Spectral neural approximations for models of transcriptional dynamics

Gennady Gorin[1,*], Maria Carilli[1,2,*], Tara Chari[2], and Lior Pachter[2,3,**]

[1]Division of Chemistry and Chemical Engineering, California Institute of Technology, Pasadena, CA, 91125
[2]Division of Biology and Biological Engineering, California Institute of Technology, Pasadena, CA, 91125
[3]Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, 91125
[*]These authors contributed equally to this work.
[**]Address correspondence to Lior Pachter (lpachter@caltech.edu)

June 16, 2022

Abstract

Transcriptional systems involving discrete, stochastic events are naturally modeled using Chemical Master Equations (CMEs). These can be solved for microstate probabilities over time and state space for a better understanding of biological rates and system dynamics. However, closed form solutions to CMEs are available in only the simplest cases. Probing systems of higher complexity is challenging due to the computational cost of finding solutions and often compromises accuracy by treating infinite systems as finite. We use statistical understanding of system behavior and the generalizability of neural networks to approximate steady-state joint distribution solutions for a two-species model of the life cycle of RNA. We define a set of kernel functions using moments of the system and learn optimal weights for kernel functions with a neural network trained to minimize statistical distance between approximated and numerically calculated distributions. We show that this method of kernel weight regression (KWR) approximation is as accurate as lower-order generating-function solutions to the system, but faster; KWR approximation reduces the time for likelihood evaluation by several orders of magnitude. KWR also generalizes to produce probability predictions for system rates outside of training sets, thereby enabling efficient transcriptional parameter exploration and system analysis.

1 Introduction

Single-cell RNA sequencing (scRNA-seq) data that reflect stochastic transcription dynamics can be naturally modeled by discrete-valued, continuous-time Markov chains (CTMCs) that are formalized by chemical master equations (CMEs) [1]. Such models track the probabilities of system microstates, which can be interpreted as likelihoods and used to fit biophysical parameters [2] as well as to obtain insights into transcriptional biophysics [3]. However, models that combine nontrivial regulatory schema with multimodal data are not typically tractable [3,4].

To circumvent the challenges of finding closed-form solutions to models, CTMCs are commonly simulated using Gillespie's stochastic simulation algorithm [5], which generates statistically valid realizations from the underlying distributions. However, as is typical with Monte Carlo (MC) methods, the simulated distributions approach the true distributions only with inverse-square root convergence in the number of realizations. Furthermore, simulation procedures are computationally expensive, rendering them inefficient for likelihood evaluation. Alternatively, it is possible to truncate the state spaces, reducing the possible molecule count microstates to a "large enough" finite subdomain of some fixed size ß, and apply standard matrix methods to solve the CME. This approach is called the finite state projection (FSP) algorithm [6,7], and it has been successfully applied to inference [2,8–10] and experiment design [11]. However, the FSP algorithm requires $\mathcal{O}(ß^3)$ time complexity. Thus, the procedure is extremely inefficient in the typical case where ß is much larger than the number of data points. The method also fails to provide gradients for likelihood optimization.

Some classes of CMEs can be solved using generating functions that transform infinite-dimensional CMEs to complex partial differential equations (PDEs). These PDEs have the same number of spatial dimensions as number of species [12] and generalize to coupled series of PDEs when the systems involve states on a finite subset of $\mathbb{N}$, such as binary promoter activation or deactivation [4]. Generating function approaches are not always applicable, and can require numerical integration. When feasible, generating function evaluation has space and time complexity of $\mathcal{O}(ß)$. However, converting from a generating function domain to a count domain involves an inverse fast Fourier transform [13], which requires storing an array of size ß and has a formal time complexity of $\mathcal{O}(ß \log ß)$. Thus, as with the FSP algorithm, if ß is large, generating function methods are infeasible due to the "curse of dimensionality" and high memory requirements. Finally, such approaches still require truncation and are not amenable to calculating gradients.

In a narrow set of systems, generating functions can be inverted by differentiation [14], Cauchy contour integration [15], or Taylor expansion [16] to explicitly obtain microstate probabilities. Such closed-form solutions are available for RNA splicing and degradation coupled to a constitutive, or unregulated, transcription process [17], and several stationary solutions are available for a single RNA species produced by regulated transcription [18,19]. However, even when available, calculating the probability of a single microstate requires summing over probabilities of all microstates with fewer species [10,14,18,20–22].

We present an alternative approach using kernel function weight regression (KWR) to represent and approximate the multidimensional solutions to CMEs used to model transcriptional dynamics. We investigate a CME system involving transcriptional bursting, splicing and degradation (Figure 1a), and use the qualitative features of its univariate conditional disitributions to define an approximation in terms of a finite mixture of negative binomial kernel functions. The weights for these kernel functions are learned by a neural network (Figure 1b). By applying conditional probability, we use this method to approximate bivariate distributions (Figure 1c). Our approach

is motivated in part by the Poisson representation of CMEs [23], which reframes discrete processes as infinite Poisson mixtures. As the KWR procedure is compatible with automatic differentiation, the approximation allows for gradient calculations, which are useful for likelihood optimization.

Despite the considerable success of basis function learning in the study of continuous dynamical systems [24,25], such methods, and neural approximations in general, have been exceedingly rare in the single-cell biophysics field. [26] apply recurrent neural networks to directly learn CME solutions, building on previous work on continuous systems [27]. However, the solution domain size ß is predefined, meaning the method cannot extrapolate beyond it. [28] have proposed neural approximations for continuous-valued statistics of complex CME systems; unfortunately, this approach cannot be used to perform inference. Parallel work by [29] leverages neural networks to approximate a time-homogeneous non-Markovian system by a non-homogeneous Markovian system with learned time dependence, demonstrating the potential breadth of applications of neural approximations; however, the method still requires solving the resulting system by FSP.

Most pertinently, [30] have independently proposed a method for the approximation of CME solutions by a set of negative binomial mixture functions, implemented using the neural Nessie framework. However, the approach is qualitatively different to the one we present. Nessie learns mixture function locations and weights, whereas we only learn weights and a single scaling factor to place the kernel functions. This is possible by exploiting the known unimodality and lower moments of the solutions. Our approach simplifies the learning process, but requires more preliminary mathematical analysis and understanding of the distribution shapes. Further, Nessie is designed to fit univariate distributions, whereas we present a method for learning multivariate distributions by conditioning.

2 Results

We consider the system illustrated in Figure 1a, that encodes a simple but standard model of the mRNA life-cycle [31]. Transcriptional events arrive according to a Poisson process with rate α. Every transcriptional event generates B nascent mRNAs ($\mathcal{N}$), where B is a geometric distribution with mean b. After an exponentially-distributed delay with rate β, nascent mRNAs are spliced into mature mRNAs ($\mathcal{M}$). Finally, after an exponentially-distributed delay with rate γ, mature mRNAs are degraded. This system induces a stationary distribution with PMF $P(n, m; b, \beta, \gamma)$, where β and γ are considered in units of α with no loss of generality at steady state. The resultant PMF is not known in closed form, but its nascent marginal $P(n)$ is negative binomially distributed.

To approximate full joint distributions, we train a neural network to approximate the distributions of mature RNA molecules conditional on a nascent copy number n (Figure 1b), i.e., the univariate PMFs $P(m|n; b, \beta, \gamma)$. To accomplish this, we generate rate vectors $\theta := \{b, \beta, \gamma\}$ from a log-uniform distribution, compute high-quality generating functions, use Fourier inversion to obtain joint PMFs $P(m, n; \theta)$, and divide by the nascent probabilities $P(n; \theta)$ to extract the conditional PMFs. To construct a basis for each conditional distribution given nascent value n, we use a function $\psi(\theta, n)$ to calculate kernel means μ_k^* and standard deviations σ_k for a set of kernel negative binomial distributions, where k indexes the kernels. Next, we construct a neural network $\mathcal{F}$, which maps $\tilde{\theta}$, a transformation of θ and n, to a set of weights $\mathbf{w}$ and a kernel width scaling factor h. To approximate conditional distributions using the kernel basis, we train $\mathcal{F}$ by iterating over values of $\tilde{\theta}$ in the training set and minimizing the Kullback-Leibler divergence between the true distributions

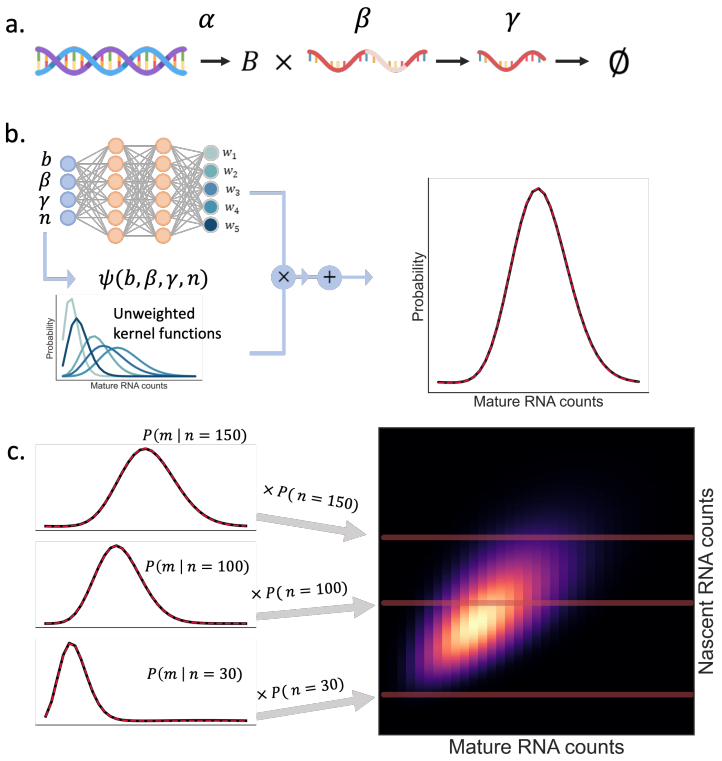

Figure 1: A schematic representation of the KWR approximation procedure. **a.** The biological process under consideration: bursty transcription of nascent RNA, followed by splicing to produce mature RNA and mRNA degradation. **b.** We approximate univariate conditional distributions by using a neural network to learn weights for a set of kernel functions. The kernel functions are weighted and added to approximate true distributions. **c.** Joint distributions are reconstructed by approximating conditional probabilities for the number of mature RNA molecules, then multiplying by marginal probabilities for the number of nascent RNA molecules.

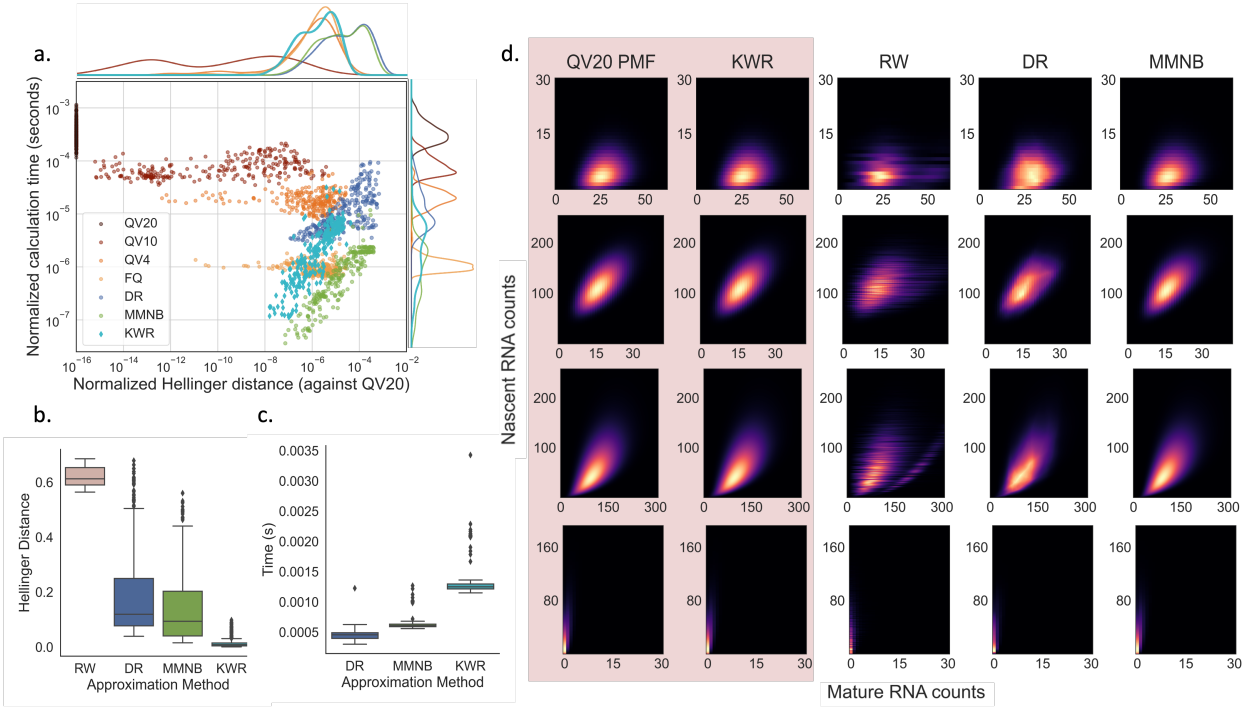

Figure 2: KWR approximation accurately reconstructs joint probability distributions, with relatively low runtime requirements. **a.** Time to generate joint distributions over state space versus Hellinger distance, both normalized by state space size, for 256 testing rate vectors. The methods compared are three generating-function based evaluation methods (QV10, QV4, and FQ), direct regression (DR), a moment-matched negative binomial (MMNB) approximation, and kernel weight regression (KWR). All Hellinger distances are calculated against QV20 distributions. **b.** Hellinger distance (unnormalized) over state space for 768 testing rate vectors using four approximation methods: kernels weighted with random weights (RW), direct regression (DR), moment-matched negative binomial (MMNB), and kernel weight regression (KWR). **c.** Time to calculate a single microstate probability $P(m, n; \theta)$ for 768 testing rate vectors using three approximations: direct regression (DR), moment-matched negative binomial (MMNB), and kernel weight regression (KWR). d. Typical distributions obtained by generating function inversion (QV20 or ground truth, left highlighted column), kernel weight regression (KWR, right highlighted column), kernels weighted with random weights (RW), direct regression (DR), and moment-matched negative binomial approximation (MMNB).

$P(m|n;\theta)$ and the kernel approximations Q:

$$Q(m|n;\theta) = \sum_k w_k P_{\text{ker}}(\mu_k^*, h\sigma_k). \tag{1}$$

Here, P_{ker} is a negative binomial distribution parametrized by its mean and standard deviation and w_k are the entries of $\mathbf{w}$. Finally, to reconstruct the joint distributions, we multiply each $Q(m|n;\theta)$ by $P(n;\theta)$, as in Figure 1c. We call this method kernel weight regression (KWR). The method is fully described in Section 6; a qualitative overview is provided in Section 6.3, model architecture illustrated in Figure S1, and parameters listed in Table S1.

To benchmark the algorithm's performance, we investigated the precision achieved in recovering PMFs on a grid, as well as the runtime, for a series of testing rate vectors (Figure 2a). To quantify precision, we calculate the Hellinger distance [30], a metric of discrepancy between probability distributions:

$$H(P,Q;\theta) = \frac{1}{\sqrt{2}}\sqrt{\sum_{n=0}^{ß_N}\sum_{m=0}^{ß_M}\left(\sqrt{P(n,m;\theta)} - \sqrt{Q(n,m;\theta)}\right)^2}, \tag{2}$$

where $ß_N$ and $ß_M$ are the state space bounds for each θ. Although the system in Figure 1a does not afford a ground truth, we can approximate it by a high-quality generating function approximation. This approximation is denoted by QV20, and was obtained by adaptive quadrature and evaluated on a grid spanning 20 standard deviations above the means, i.e., with bounds $\mu_z + 20\sigma_z$ in each dimension $z \in \{N, M\}$. To compare distributions with different nascent and mature means, we normalized calculation times and Hellinger distances by state space, dividing by $ß_N \times ß_N$.

We compared KWR to four generating function-based solutions. Three of these solutions (QV20, QV10, QV4) use adaptive quadrature on smaller grids of 20, 10, and 4 standard deviations above nascent and mature means respectively, whereas the fourth (FQ) uses 60th-order Gaussian quadrature on a grid of 4 standard deviations above the means. We also benchmarked a novel coarse approximation (MMNB), obtained by moment-matching a lognormal conditional in Section 6.5 to a negative binomial distribution (Section S2). We found that the KWR approximation achieves accuracy comparable to lower-order approximations when evaluating probabilities over a grid, but reduces evaluation time (Figure 2a). For example, the average state-space normalized Hellinger distances for 256 testing rate vectors were 3.216×10^{-6} and 4.515×10^{-6} for QV4 and KWR, respectively, but KWR approximation was an order of magnitude faster with an average time of 1.949×10^{-5} seconds for QV4 evaluation and 3.329×10^{-6} seconds for KWR approximation (Figure 2a).

We also compared the KWR approximation to a method of function approximation more typical in machine learning: a direct prediction of joint probability using the neural network to directly map from input $\{\theta, n, m\}$ to a single-valued output $P(m, n;\theta)$. We refer to this method as direct regression (DR) and describe its implementation in Section S1 with testing results shown in Figure S2 and network parameters listed in Table S2. DR is typically less accurate than the KWR approximation and the coarse MMNB approximation.

As a control, we investigated the approximations obtained by using a learned scaling factor h with random weights $\mathbf{w}$ in Equation 1. As shown in Figure 2b, the control displays poor performance, motivating the use of learned weights.

In the context of statistical inference from data, the optimization of likelihoods involves evaluation of a single microstate $P(n, m; \theta)$. On this task, KWR outperforms generating function-based methods by design, as it bypasses the grid evaluation. Generating-function based calculation on grids of 20 or 10 standard deviations above nascent and mature means (QV20 and QV10) can take up to several hundred seconds (see Figure S3); evaluating single microstate probabilities using KWR approximation took on average 0.0015 seconds, decreasing evaluation time by four orders of magnitude (Figure 2c). We find that DR and MMNB slightly outperform KWR on microstate timing (Figure 2c). However, their accuracy is substantially lower (Figure 2b).

Some examples of probability distributions obtained by the approximation procedures are shown in Figure 2d, where we treat QV20 as the best approximation to the ground truth. The KWR results do not demonstrate any substantial qualitative discrepancies, suggesting that it is appropriate for moderate-precision inference. The MMNB results show qualitative discrepancies (e.g., row 2 of Figure 2d), suggesting that while it is appropriate as a coarse approximation, there is an accuracy trade-off for the decrease in evaluation time. The random-weight control recapitulates the general shape of the distribution, owing to the kernel constraints, but introduces considerable error into microstate probabilities, confirming that the learned weights are essential for an accurate approximation. Finally, the DR results appear to considerably distort the distribution shapes, suggesting that the moderate-sized network used in its implementation is insufficient for accurate approximation.

In addition to facilitating PMF evaluation, the neural network enables substantial compression. The training, testing, and evaluation examples and ground truth distributions required 4.36 GB of storage, while the neural network and scripts used to calculate kernel functions are a mere 285 KB in size, requiring about 10,000 times less storage space. After training, examples and ground truth distributions no longer need to be stored, greatly reducing memory requirements for subsequent inference or joint distribution evaluation.

3 Discussion

After training on a relatively small set of example rate vectors that parameterize the transcriptional model and their ground truth probability distributions, the KWR neural network generalizes to produce highly accurate and unbounded joint distribution approximations out of sample (Figure 3). Thus, it can greatly reduce the amount of time necessary to explore steady state distributions across a wide range of transcriptional parameters for inference, parameter sensitivity investigations, and experimental design [11] under the current model.

The implementation we have designed is strongly reliant on the structure of the system at hand: we have taken considerable advantage of known distributional properties, such as moments and understanding of the qualitative shape. To judiciously place bases around the region of high probability density, we constructed an analytically tractable approximation to the distribution. Nevertheless, it appears plausible that more sophisticated models can be treated using analogous formulas, such as a lognormal mixture for distributions known to be multimodal.

We have used a closed-form conditioning probability $P(n)$ to convert the two-dimensional problem into a one-dimensional one. This does not appear to be a fundamental constraint to generalizing the current approach. For example, if the nascent marginal were not known in closed form, we could "bootstrap" by first learning a neural approximation $Q(n)$, then fitting $Q(m|n)$, to produce bivariate approximations. We anticipate that this approach can extend to distributions

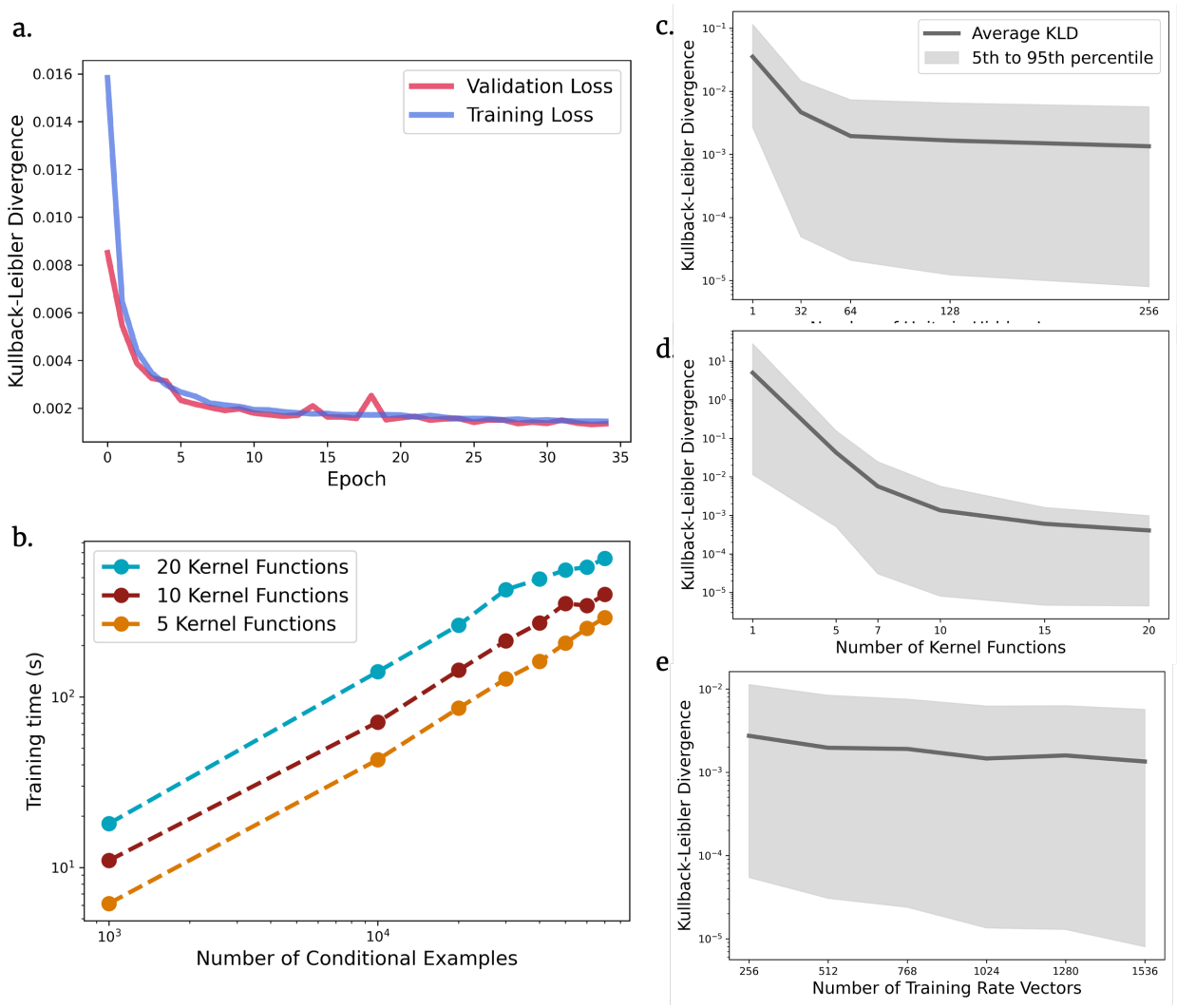

Figure 3: The training, validation, and testing performance of the KWR method. **a.** The average Kullback-Leibler divergence (KLD) between true and KWR approximated conditional distributions for training and validation examples decreases to zero as the model is trained. **b.** The training time increases linearly with the size of the training set and the number of kernel functions used. **c.** KLDs for testing examples decrease with increasing the number of nodes in hidden layer, **d.** increasing the number number of kernel functions, and **e.** increasing the number of training rate vectors.

in more dimensions, potentially bypassing the "curse of dimensionality" more generally, so long as ground truth distributions can be generated and stored.

The method we have presented is focused on the time-independent stationary distribution. Time-dependent distributions present challenges, because they can be arbitrarily far from steady state or a unimodal distribution: the initial distributions are fundamentally unconstrained. However, it may be feasible to train a network to learn a more constrained version of the problem, where the initial distribution is at steady state.

Furthermore, in our work we have approximated the solutions for a single class of distributions induced by a memoryless bursty transcription process. We anticipate our approach to generalize to considerably more general classes of unimodal distributions. For example, we expect that the solutions of CMEs with a transcription rate governed by stochastic differential equation [3] can be implemented with only minor modifications to the kernel construction procedure.

Finally, our method can be adapted to learn distributions based on simulations, as in the Nessie framework [30], substantially expanding the range of models. Specifically, the recent advances in simultaneous quantification [32] and modeling [33] of mRNA and proteins presents a natural application. The trivariate unspliced mRNA – spliced mRNA – protein system does not afford an analytical solution that can be fit to data, but it is easily simulated and suggests directions for future applications.

4 Data and code availability

All training and validation datasets, trained models, scripts to generate manuscript figures, and a small Google Colaboratory demonstration are available at `https://github.com/pachterlab/GCCP_2022`.

5 Acknowledgments

We thank Dr. Yisong Yue, Yongin Choi, Dr. John J. Vastola, and Dr. Zachary Fox for valuable discussions. G.G. and L.P. were partially funded by NIH U19MH114830. Figures 1, 2, 3, and S3 use color palettes from MetBrewer, developed by Blake R. Mills and available at `https://github.com/BlakeRMills/MetBrewer`.

6 Methods

6.1 Statistical preliminaries

The Poisson distribution over the non-negative integers $x \in \mathbb{N}_0$ is parameterized by its mean λ:

$$P_{Poiss}(x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}. \tag{3}$$

The negative binomial (NB) distribution over $x \in \mathbb{N}_0$ can be parameterized by the shape r and the success probability p:

$$P_{NB,rp}(x; r, p) = \frac{\Gamma(r+x)}{x!\Gamma(r)}(1-p)^x p^r. \tag{4}$$

This is the probability mass function (PMF) in its combinatorial form, which follows the convention used in MATLAB, `numpy`, and `scipy`.

Alternatively, the NB distribution can be parametrized by the shape r and the mean μ:

$$P_{NB,r\mu}(x; r, \mu) = \frac{\Gamma(r+x)}{x!\Gamma(r)} \left(\frac{r}{r+\mu}\right)^r \left(\frac{\mu}{r+\mu}\right)^x. \tag{5}$$

The shape r is specified given a mean and standard deviation (as long as $\sigma^2 > \mu$):

$$r(\mu, \sigma) = \frac{\mu^2}{\sigma^2 - \mu}. \tag{6}$$

We use the resultant mean-variance (μ, σ^2) parametrization of the NB distribution:

$$P_{NB}(x; \mu, \sigma) = P_{NB,r\mu}(r(\mu, \sigma), \mu). \tag{7}$$

The lognormal distribution over $x \in \mathbb{R}_+$ can be parametrized by the mean and standard deviation of the underlying exponentiated normal distribution (from here, referred to as the log-mean μ and the log-standard deviation σ):

$$f(x; \mu_l, \sigma_l) = \frac{1}{x\sigma_l\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \mu_l)^2}{2\sigma_l^2}\right). \tag{8}$$

The parameters can be set according to the distribution's desired mean and variance:

$$\begin{aligned} \mu_l &= \ln \frac{\mu^2}{\sqrt{\sigma^2 + \mu^2}}, \\ \sigma_l &= \left[\ln \frac{\sigma^2 + \mu^2}{\mu^2}\right]^{1/2}. \end{aligned} \tag{9}$$

The quantiles of the lognormal distribution have the following form:

$$q(x; \mu_l, \sigma_l) = \exp\left(\mu_l + \sigma_l \Phi^{-1}(x)\right), \tag{10}$$

where Φ is the cumulative distribution function of the standard normal distribution and Φ^{-1} is its quantile function.

The bivariate lognormal distribution over $x, y \in \mathbb{R}_+$ can be similarly parametrized, with an additional parameter that governs the correlation of the underlying bivariate normal distribution (from here, referred to as the log-correlation ρ_l):

$$\begin{aligned} f(x, y; \mu_{xl}, \mu_{yl}, \sigma_{1l}, \sigma_{yl}, \rho_l) &= \frac{1}{xy \times 2\pi\sigma_{1l}\sigma_{yl}\sqrt{1-\rho_l^2}} \exp\left(-\frac{A^2 + B^2 - 2\rho_l AB}{2(1-\rho_l^2)}\right), \\ A &= \frac{\ln x - \mu_{1l}}{\sigma_{1l}}, \\ B &= \frac{\ln y - \mu_{yl}}{\sigma_{yl}}, \end{aligned} \tag{11}$$

where μ_{xl} and μ_{yl} are the marginal log-means, σ_{xl} and σ_{yl} are the marginal log-standard deviations [34]. The log-correlation ρ_l can be set according to the distribution's desired marginal moments and correlation ρ:

$$\rho_l = \frac{1}{\sigma_{xl}\sigma_{yl}} \ln\left(\rho\sigma_1\sigma_2 e^{\mu_{xl}+\mu_{yl}+(\sigma_{xl}+\sigma_{yl})/2} + 1\right). \tag{12}$$

Finally, the conditional distribution of y given x is lognormal, with the following log-mean and log-standard deviation:

$$\begin{aligned} \mu_{yl,x} &= \mu_{yl} + \rho_l \frac{\sigma_{yl}}{\sigma_{xl}}(\ln x - \mu_{yl}), \\ \sigma_{yl,x} &= \sigma_{yl}\sqrt{1-\rho_l^2}. \end{aligned} \tag{13}$$

6.2 System definition

We consider the following system:

$$\varnothing \xrightarrow{\alpha} B \times \mathcal{N} \xrightarrow{\beta} \mathcal{M} \xrightarrow{\gamma} \varnothing, \tag{14}$$

where $\mathcal{N}$ denotes a nascent mRNA transcript and $\mathcal{M}$ denotes a mature mRNA transcript (Figure 1a). The system's rates and burst size are defined in Table 1. We set the burst frequency α to unity and consider all other rates in units of α with no loss of generality at equilibrium.

The joint generating function solution was derived in [31]:

$$\begin{aligned} G(x_N, x_M) &= \sum_{x_N, x_M = 0}^{\infty} x_N^n x_M^m P(n, m) \\ u_z &= x_z - 1 \\ U(u_N, u_M, s) &= u_M \frac{\beta}{\beta - \gamma} e^{-\gamma s} + \left(u_N - u_M \frac{\beta}{\beta - \gamma}\right) e^{-\beta s} \\ \ln G := \phi(u_N, u_M) &= \int_0^\infty \left[\frac{1}{1 - bU(u_N, u_M, s)} - 1\right] ds. \end{aligned} \tag{15}$$

Symbol	Definition
N	Nascent mRNA species index
M	Mature mRNA species index
z	Generic species subscript (N or M)
n	Nascent mRNA copy number
m	Mature mRNA copy number
$P(n, m)$	True bivariate PMF of nascent and mature RNA counts
B	Geometric burst size distribution defined on $\mathbb{N}_0$
b	Mean burst size
β	Splicing rate, normalized to burst frequency
γ	Degradation rate, normalized to burst frequency
$\mu_N = b/\beta$	Mean of nascent count distribution
$\mu_M = b/\gamma$	Mean of mature count distribution
μ_z	Mean of generic species z
$\sigma_N^2 = \mu_N(1+b)$	Variance of nascent count distribution
$\sigma_M^2 = \mu_M\left(1+\frac{b\beta}{\beta+\gamma}\right)$	Variance of mature count distribution
σ_z^2	Variance of generic species z
$\mathrm{Cov}(N, M) = \frac{b^2}{\beta+\gamma}$	Covariance of nascent and mature RNA counts
$\rho = \frac{\mathrm{Cov}(N,M)}{\sigma_N \sigma_M}$	Pearson correlation of nascent and mature RNA counts

Table 1: Statistical and biological quantities of interest.

Here x_N and x_M are complex generating function arguments. The nascent marginal of this distribution is given by a negative binomial law [31,35]:

$$P(n) = P_{NB,r\mu}\left(\frac{k}{\beta}, \frac{kb}{\beta}\right). \tag{16}$$

Salient lower moments can be computed (Table 1), and the distribution is known to be unimodal [12]. However, the microstate, conditional, and mature marginal probabilities do not have closed-form solutions.

The distribution can be evaluated on a grid of size $ß_N \times ß_M$, or the direct product of species-specific microstates $[0, 1, ..., ß_N - 1] \times [0, 1, ..., ß_M - 1]$, by evaluating G on the complex bivariate unit sphere and performing an inverse fast Fourier transform (IFFT) [13,31]. As the IFFT sums to unity, the selected state space sizes $\{ß_N, ß_M\}$ must be sufficiently large to limit truncation error. Therefore, it is typically set according to standard concentration inequalities: for example, Chebyshev's inequality guarantees that the amount of probability mass at $x \geq \mu + 4\sigma$ will be at most 1/16.

6.3 Overview of approximation procedure

In the most general formulation, a CME model of biology converges to a stationary distribution $P(z; \theta)$ over $z \in \mathbb{N}_0$, which can be computed at considerable computational cost through generating function inversion. This distribution may be marginal or conditional. In certain cases, P affords a

real Poisson representation [23]:

$$P(z;\theta) = \int_0^\infty P_{\text{Poiss}}(z;\lambda)dF_\lambda, \tag{17}$$

where F is a mixing cumulative distribution function (CDF), or stationary solution to an underlying continuous-valued Fokker-Planck equation. If such a representation exists, we can always write down a finite approximation over K Poisson kernels:

$$Q(z;\theta) = \sum_{k=1}^{K} w_k P_{\text{Poiss}}(z;\lambda_k), \tag{18}$$

where w_k are weights on a K-dimensional simplex. Formally, this approximation substitutes the continuous CDF F with a piecewise constant CDF F^*:

$$F^*(\lambda) = \sum_{k=1}^{K} w_k \mathbb{I}_{\lambda \leq \lambda_k}, \tag{19}$$

where $\mathbb{I}$ is the indicator function. In principle, every P can be analyzed to produce an optimal set of weights w_k and mixture components λ_k, i.e., ones which minimize a measure of statistical divergence between P and Q. Further, as $K \to \infty$, $Q \to P$. However, the problem of actually obtaining the optimal w_k and λ_k may be challenging. More problematically, convergence in the number of kernels in K is typically slow.

Our approach is to first accelerate convergence in K. To accomplish this, we use negative binomial kernels, which are continuous Poisson mixtures with a gamma mixing density:

$$\begin{aligned} P_{\text{ker}}(z;\mu^*,\sigma^*) &= P_{\text{NB}}(z;\mu_k^*,\sigma_k^*) \text{ if } (\sigma_k^*)^2 > \mu_k^*; \\ &= P_{\text{Poiss}}(z;\mu_k^*) \text{ otherwise,} \end{aligned} \tag{20}$$

where μ_k^* and σ_k^* are the mean and standard deviation of the negative binomial kernel. Therefore, the approximation is a mixture of K kernels:

$$Q(z;\theta) = \sum_{k=1}^{K} w_k P_{\text{ker}}(m;\mu_k^*,\sigma_k^*). \tag{21}$$

To implement this approximation, we need to find values of w_k, μ_k^*, and σ_k^*. In principle, this can be accomplished by training a neural network to predict these quantities based on θ; this is the approach taken in the Nessie framework [30]. However, we find that μ_k^* and σ_k^* can be assigned based on the shape of the approximated distribution.

We seek to judiciously place the kernel locations μ_k^*. These locations should be adaptive with respect to rate vector θ, and prioritize high-probability mass regions of the distribution P. We assume that $P(z;\theta)$ resembles a lognormal distribution, which is typical of biomolecules [36] and consistent with the knowledge that P is defined on the positive numbers, unimodal, and skewed to the right. If we have such an approximating lognormal distribution with parameters μ_l and σ_l, we can place μ_k^* at the quantiles of this distribution, given by Equation 10. We found that quantiles placed at the Chebyshev nodes (rescaled to $[0,1]$) gave the best performance:

$$\begin{aligned} x_k &= \frac{1}{2}\left[\cos\left(\frac{2k-1}{2K}\pi\right)+1\right], \\ \mu_k^* &= q(x_k;\mu_l,\sigma_l). \end{aligned} \tag{22}$$

Next, we seek to specify the kernel standard deviations σ_k^*. Knowing that Q should be unimodal, we implement an adaptive procedure:

$$\begin{aligned} \sigma_k &:= \mu_{k+1} - \mu_k, \\ \sigma_k^* &\propto \sigma_k, \end{aligned} \tag{23}$$

i.e., the width of each kernel is controlled by the spacing between neighboring kernels. This approach prevents the individual kernels from becoming too sharply peaked. We define the constant of proportionality as h, such that $\sigma_k^* = h \times (\mu_{k+1} - \mu_k) := h\sigma_k$, where σ_k is the unscaled kernel width. We set the final parameter σ_K^* to $\sqrt{\mu_K^*}$, making the Kth kernel Poisson.

Given this framework, we can split the problem of evaluating Q into two parts. The analytical part uses the lognormal approximation to obtain the kernel locations and unscaled widths:

$$\psi : \theta \mapsto \begin{bmatrix} (\mu_1^*, \sigma_1) \\ ... \\ (\mu_k^*, \sigma_k) \end{bmatrix}. \tag{24}$$

As a shorthand, we use the following component-specific notation:

$$\psi_k : \theta \mapsto (\mu_k^*, \sigma_k), \tag{25}$$

with the caveat that this shorthand corresponds to extracting entries of the K-dimensional array in Equation 24, as σ_k are computed in a coupled fashion.

The second part of the problem involves evaluating $\mathbf{w}$ and h. Conceptually, these are implemented using a neural network $\mathcal{F}$:

$$\mathcal{F} : \tilde{\theta} \mapsto w_1, ..., w_K, h = \mathbf{w}, h, \tag{26}$$

where $\tilde{\theta}$ is an transformation of θ.

This procedure generates a prediction $Q(z; \theta)$ for a microstate m under θ. By minimizing a measure of statistical distance between P and Q, we can train the neural network $\mathcal{F}$ to output optimal $\mathbf{w}$ and h.

6.4 Measures of statistical distance

To train $\mathcal{F}$, we use a variant of the Kullback-Leibler divergence (KLD) between P and Q as our loss function. Given a target distribution P and an approximator Q, the KLD is defined as follows:

$$D(P \parallel Q) = -\sum_{z=0}^{\infty} P(z) \ln \frac{Q(z)}{P(z)}. \tag{27}$$

Although this loss function is standard [30], it poses problems in the current context. Although the KLD is guaranteed to be positive, its truncated version (with the sum taken up to a finite limit ß) is not. To construct a well-defined measure of error, we use the distributions' truncated analogues:

$$P'(z) := \frac{P(z)}{\sum_{z=0}^{ß} P(z)}. \tag{28}$$

Therefore, the loss function takes the following form:

$$L = D(P' \parallel Q') = -\sum_{z=0}^{ß} P'(z) \ln \frac{Q'(z)}{P'(z)}. \tag{29}$$

We use this version whenever we refer to the optimization of KL divergence. It is conceivable that the truncation procedure in Equation 28 can introduce error. Therefore, to benchmark the performance of the algorithm, we use an orthogonal metric, the Hellinger distance. This metric is given in Equation 2; since squares are strictly positive, the truncated version can be safely treated as a lower bound on the Hellinger distance over the entire state space.

6.5 Approximating joint distributions

To approximate the joint distribution of nascent and mature RNA, we use nascent marginals and approximate mature conditionals using the approximation procedure described in Section 6.3 based on $P(n, m) = P(n)P(m|n)$. The nascent marginal $P(n)$ is known (Equation 16). To obtain the joint distribution, we need to solve the univariate problem of approximating $P(m|n)$ by a judiciously chosen ψ and neural network input $\tilde{\theta}$.

To define ψ, we first construct a univariate lognormal distribution that resembles the true conditional distribution $P(m|n;\theta)$. As the conditional moments of $P(m, n;\theta)$ are not known in closed form, we first approximate the joint distribution $P(m, n;\theta)$ using the the bivariate lognormal distribution described in Equation 12, matched to μ_N, σ_N^2, μ_M, σ_M^2, and ρ of $P(m, n;\theta)$ (Table 1). To do so, we apply Equation 9 for N and M to obtain the marginal parameters, then Equation 12 to obtain the correlation. Finally, we evaluate Equation 13 at $x = n + 1$ to obtain the parameters μ_l and σ_l of a coarse lognormal approximation of conditional distributions $P(m|n;\theta)$. The kernel locations and unscaled widths follow from Equations 22 and 23.

Next, we define conditional examples $\hat{\theta}$, transformations of θ, n, to use as inputs to the neural network. We found that $\hat{\theta} = \{\log_{10} b, \log_{10} \beta, \log_{10} \gamma, \mu_l, \sigma_l, ß_M, n\}$, where $ß_M = \mu_M + 4\sigma_M$ is the state space bound for mature counts, reduced training time as conditional means and variances were pre-calculated. Each rate vector $\theta = b, \beta, \gamma$ corresponds to $ß_N$ conditional examples $\tilde{\theta}$ (Section 6.5). With the kernels and the network inputs specified, we iterate over pre-computed distributions $P(m|n;\theta)$ and train the network by minimizing the loss in Equation 29.

6.6 Data generation

To generate training and validation distributions, we sampled vectors of biophysical rate vectors θ within the following biologically motivated bounds:

$$\begin{aligned} b &\in [1, 300], \\ \beta &\in [0.05, 50], \\ \gamma &\in [0.05, 50], \end{aligned} \tag{30}$$

where the sampling measure was log-uniform. As described previously, β and γ are unitless quantities, scaled to the value of α. We generated a vector of three random numbers within the bounds. Next, we computed μ_N and μ_M, the average nascent and mature copy numbers (Table 1). If either of these averages exceeded 1,000, we discarded and regenerated the rate vector; we adopted this

rejection procedure to ensure the training distributions matched the low-copy number regime seen in transcriptomic data. Next, we evaluated the solution in Equation 15 up to the bounds $\mu_z + 20\sigma_z$ using adaptive quadrature (implemented in the *SciPy* function `integrate.quad_vec` [37]) and the inverse real fast Fourier transform (implemented in the *SciPy* function `fft.irfft2` [38]). We denote this high-precision procedure by QV20. After evaluating $P(n, m; \theta)$ on a large grid, we truncated this ground truth PMF to the more practical size $ß_z = \mu_z + 4\sigma_z$ and stored it to disk. To generate conditional training data, we extracted the entries corresponding to each n and multiplied them by $P(n; \theta)$ (Equation 4, implemented in the *SciPy* function `stats.nbinom.pmf` [37] with $r = \beta^{-1}$ and $p = (1 + b)^{-1}$ [35]). The resulting conditional distributions were stored to disk, alongside their values of θ and n.

Symbol	Definition
K	Number of approximating kernels
k	Approximating kernel index
$\mathbf{w} := w_1, ..., w_K$	Weights of approximating kernels
Θ_k	Parameters of approximating kernel
h	Scaling Factor
$\mathcal{F}$	Generic multivariate function implemented through a neural network
P_{ker}	Univariate PMF of approximating kernel
Q	Approximating mixture
$ß_N$	Nascent state space size or grid dimension
$ß_M$	Mature state space size or grid dimension
$ß_z$	State space size in a generic dimension
$ß = ß_N \times ß_M$	Total state space size
$P_{\text{NB}}(x; \mu, \sigma)$	Negative binomial PMF
$P_{\text{Poiss}}(x; \lambda)$	Poisson PMF

Table 2: Approximation variables.

6.7 Kernel weight regression with neural network

The final architecture for the neural network $\mathcal{F}$ used to predict $\mathbf{w}$ and h is a multilayer perceptron (MLP) that takes as input rate vector $\tilde{\theta} \in \mathbb{R}^7$ and outputs weight vector $\mathbf{w} \in \mathbb{R}^{10}$, corresponding to 10 kernel functions, and scaling factor h. The input $\tilde{\theta}$ is passed through two, fully connected layers of 256 nodes, each with ReLU activation functions, to output $\mathbf{w}$, which is normalized by the softmax activation function to ensure weights $\mathbf{w}$ sum to unity. To produce h, outputs from the first 256 node layer are also passed to a single node with the sigmoid activation function to produce a number in the range [0,1], which is then re-scaled to the range [1,6]. Model architecture is illustrated in Figure S1 and network parameters listed in Table S1.

Weights $\mathbf{w}$, scaling factor h, and kernel negative binomial distributions are combined to predict an approximate conditional probability $Q(m|n; \theta)$ as described in Section 6.3. The network is trained to minimize truncated KL divergence (Section 6.4) between the approximation Q and the QV20 conditional ground truth (Section 6.6). We implemented the MLP in PyTorch and used the Adam optimizer to update node weights with a learning rate of 0.001 [39]. We did not implement a learning decay rate.

Under these conditions, the model successfully decreased the KL divergence between ground truth and learned conditional PMFs for training and validation sets during the training process (Figure 3a). The trained model also performed well on a testing set of 756 rate vectors (104,954 conditional examples), with 1% and 99% quantiles of testing KLD values of 3.06×10^{-6} and 0.0268, respectively. This network required 10.8 hours to train for 35 epochs.

To choose our final architecture, we tested the effects of changing model and training configurations on timing and KLD minimization, as shown in Figure 3b - e. Wherever not indicated, we used 10 kernel functions and two hidden layers of 256 nodes as defaults with 1,536 training rate vectors (211,515 conditional examples), 512 validation rate vectors (72,906 conditional examples), and 768 testing rate vectors (109,685 conditional examples). Models in Figure 3c and d were trained for 30 epochs, and models in Figure 3e were trained for 35 epochs. Models in Figure 3b were trained for 1 epoch with the indicated number of training conditional examples and 1,000 validation conditional examples. Training time scaled linearly with the training set size and increased with the number of kernel functions (Figure 3b).

Increasing the number of nodes in each layer allows the network to learn more complex representations: we found that increasing the number of nodes in each hidden layer decreased the testing KLDs, with diminishing returns at 64 nodes (Figure 3c). The number of kernel functions also governs the diversity of approximable distributions: adding more kernels did not produce a substantial effect beyond $K = 10$, which we implement in our final architecture (Figure 3d). As we increased the training set size, we found performance improved only marginally (Figure 3e). This suggests the function being approximated by $\mathcal{F}$ is relatively simple: the network can be trained on a relatively small number of examples, yet effectively extrapolate to examples outside of the training set.

References

[1] Crispin Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry, and the Natural Sciences*. Springer, third edition, 2004.

[2] Brian Munsky, Guoliang Li, Zachary R. Fox, Douglas P. Shepherd, and Gregor Neuert. Distribution shapes govern the discovery of predictive models for gene regulation. *Proceedings of the National Academy of Sciences*, 115(29):7533–7538, 2018.

[3] Gennady Gorin, John J Vastola, Meichen Fang, and Lior Pachter. Interpretable and tractable models of transcriptional noise for the rational design of single-molecule quantification experiments. Preprint, bioRxiv: 2021.09.06.459173, September 2021.

[4] John J Vastola. *In search of a coherent theoretical framework for stochastic gene regulation*. PhD thesis, Vanderbilt, March 2021.

[5] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, December 1976.

[6] Brian Munsky and Mustafa Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of Chemical Physics*, 124(4):044104, 2006.

[7] Ankit Gupta, Jan Mikelson, and Mustafa Khammash. A finite state projection algorithm for the stationary solution of the chemical master equation. *The Journal of Chemical Physics*, 147(15):154101, October 2017.

[8] Zachary Fox, Gregor Neuert, and Brian Munsky. Finite state projection based bounds to compare chemical master equation models using single-cell data. *The Journal of Chemical Physics*, 145(7):074101, August 2016.

[9] Brian Munsky, Zachary Fox, and Gregor Neuert. Integrating single-molecule experiments and discrete stochastic models to understand heterogeneous gene transcription dynamics. *Methods*, 85:12–21, 2015.

[10] Heng Xu, Samuel O. Skinner, Anna Marie Sokac, and Ido Golding. Stochastic Kinetics of Nascent RNA. *Physical Review Letters*, 117(12):128101, 2016.

[11] Zachary R Fox and Brian Munsky. The finite state projection based Fisher information matrix approach to estimate information and optimize single-cell experiments. *PLOS Computational Biology*, 15(1):e1006365, January 2019.

[12] Gennady Gorin and Lior Pachter. Modeling bursty transcription and splicing with the chemical master equation. *Biophysical Journal*, 121(6):1056–1069, February 2022.

[13] Pavol Bokes, John R. King, Andrew T. A. Wood, and Matthew Loose. Exact and approximate distributions of protein and mRNA levels in the low-copy regime of gene expression. *Journal of Mathematical Biology*, 64(5):829–854, April 2012.

[14] Lifang Huang, Zhanjiang Yuan, Peijiang Liu, and Tianshou Zhou. Feedback-induced counterintuitive correlations of gene expression noise with bursting kinetics. *Physical Review E*, 90(5):052702, November 2014.

[15] Srividya Iyer-Biswas, F. Hayot, and C. Jayaprakash. Stochasticity of gene products from transcriptional pulsing. *Physical Review E*, 79(3):031911, March 2009.

[16] Casper H. L. Beentjes, Ruben Perez-Carrasco, and Ramon Grima. Exact solution of stochastic gene expression models with bursting, cell cycle and replication dynamics. *Physical Review E*, 101(3):032403, March 2020.

[17] Tobias Jahnke and Wilhelm Huisinga. Solving the chemical master equation for monomolecular reaction systems analytically. *Journal of Mathematical Biology*, 54:1–26, September 2006.

[18] Lifang Huang, Zhanjiang Yuan, Peijiang Liu, and Tianshou Zhou. Effects of promoter leakage on dynamics of gene expression. *BMC Systems Biology*, 9, March 2015.

[19] Arjun Raj, Charles S Peskin, Daniel Tranchina, Diana Y Vargas, and Sanjay Tyagi. Stochastic mRNA Synthesis in Mammalian Cells. *PLoS Biology*, 4(10):e309, September 2006.

[20] Peijiang Liu, Zhanjiang Yuan, Lifang Huang, and Tianshou Zhou. Roles of factorial noise in inducing bimodal gene expression. *Physical Review E*, 91(6):062706, June 2015.

[21] Niraj Kumar, Thierry Platini, and Rahul V. Kulkarni. Exact Distributions for Stochastic Gene Expression Models with Bursting and Feedback. *Physical Review Letters*, 113(26):268105, December 2014.

[22] Shanjie Zhang and Jianming Jin. *Computation of Special Functions*. Wiley, New York, 1 edition, 1996.

[23] C. W. Gardiner and S. Chaturvedi. The poisson representation. I. A new technique for chemical master equations. *Journal of Statistical Physics*, 17(6):429–468, December 1977.

[24] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016.

[25] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, November 2019.

[26] Shangying Wang and Simone Bianco. AI-assisted Biology: Predict the Conditional Probability Distributions from Noisy Measurements. Preprint, bioRxiv: 2021.10.07.463577, October 2021.

[27] Shangying Wang, Kai Fan, Nan Luo, Yangxiaolu Cao, Feilun Wu, Carolyn Zhang, Katherine A. Heller, and Lingchong You. Massive computational acceleration by using neural networks to emulate mechanism-based biological models. *Nature Communications*, 10(1):4354, December 2019.

[28] Ankit Gupta, Christoph Schwab, and Mustafa Khammash. DeepCME: A deep learning framework for computing solution statistics of the chemical master equation. *PLOS Computational Biology*, 17(12):e1009623, December 2021.

[29] Qingchao Jiang, Xiaoming Fu, Shifu Yan, Runlai Li, Wenli Du, Zhixing Cao, Feng Qian, and Ramon Grima. Neural network aided approximation and parameter inference of non-Markovian models of gene expression. *Nature Communications*, 12(1):2618, December 2021.

[30] Augustinas Sukys, Kaan Öcal, and Ramon Grima. Approximating Solutions of the Chemical Master Equation using Neural Networks. Preprint, bioRxiv: 2022.04.26.489548, April 2022.

[31] Abhyudai Singh and Pavol Bokes. Consequences of mRNA Transport on Stochastic Variability in Protein Levels. *Biophysical Journal*, 103(5):1087–1096, September 2012.

[32] Marlon Stoeckius, Christoph Hafemeister, William Stephenson, Brian Houck-Loomis, Pratip K Chattopadhyay, Harold Swerdlow, Rahul Satija, and Peter Smibert. Simultaneous epitope and transcriptome measurement in single cells. *Nature Methods*, 14(9):865–868, September 2017.

[33] Gennady Gorin, Valentine Svensson, and Lior Pachter. Protein velocity and acceleration from single-cell multiomics experiments. *Genome Biology*, 21:39, February 2020.

[34] Yung-Sung Cheng. Bivariate Lognormal Distribution for Characterizing Asbestos Fiber Aerosols. *Aerosol Science and Technology*, 5(3):359–368, January 1986.

[35] Lisa Amrhein, Kumar Harsha, and Christiane Fuchs. A mechanistic model for the negative binomial distribution of single-cell mRNA counts. Preprint, bioRxiv: 657619, June 2019.

[36] Martin Bengtsson, Anders Ståhlberg, Patrik Rorsman, and Mikael Kubista. Gene expression profiling in single cells from the pancreatic islets of Langerhans reveals lognormal distribution of mRNA levels. *Genome Research*, 15(10):1388–1392, October 2005.

[37] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, SciPy 1.0 Contributors, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman

Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, and Yoshiki Vázquez-Baeza. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020.

[38] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.