Biologically informed NeuralODEs for genome-wide regulatory dynamics

Intekhab Hossain[1*], Viola Fanfani[1], John Quackenbush[1] and Rebekka Burkholz[2]

[1*]Department of Biostatistics, Harvard T.H. Chan School of Public Health, Boston, MA, USA.
[2]Helmholtz Center for Information Security (CISPA), Saarbrücken, Germany.

*Corresponding author(s). E-mail(s): ihossain@g.harvard.edu;
Contributing authors: vfanfani@hsph.harvard.edu;
johnq@hsph.harvard.edu; burkholz@cispa.de;

Abstract

Models that are formulated as ordinary differential equations (ODEs) can accurately explain temporal gene expression patterns and promise to yield new insights into important cellular processes, disease progression, and intervention design. Learning such ODEs is challenging, since we want to predict the evolution of gene expression in a way that accurately encodes the causal gene-regulatory network (GRN) governing the dynamics and the nonlinear functional relationships between genes. Most widely used ODE estimation methods either impose too many parametric restrictions or are not guided by meaningful biological insights, both of which impedes scalability and/or explainability. To overcome these limitations, we developed PHOENIX, a modeling framework based on neural ordinary differential equations (NeuralODEs) and Hill-Langmuir kinetics, that can flexibly incorporate prior domain knowledge and biological constraints to promote sparse, biologically interpretable representations of ODEs. We test accuracy of PHOENIX in a series of *in silico* experiments benchmarking it against several currently used tools for ODE estimation. We also demonstrate PHOENIX's flexibility by studying oscillating expression data from synchronized yeast cells and assess its scalability by modelling genome-scale breast cancer expression for samples ordered in pseudotime. Finally, we show how the combination of user-defined prior knowledge and functional forms from systems biology allows PHOENIX to encode key properties of the underlying GRN, and subsequently predict expression patterns in a biologically explainable way. PHOENIX will be available as open source at: https://github.com/QuackenbushLab/phoenix.

Keywords: gene regulatory networks, time series, dynamical systems modelling, neural ordinary differential equations (NeuralODEs)

1 Introduction

Biological systems are complex with phenotypic states, including those representing health and disease, defined by the expression states of the entire genome. Transitions between these states occur over time through the action of highly interconnected regulatory processes driven by transcription factors. Modeling molecular mechanisms that govern these transitions is essential if we are to understand the behavior of biological systems, and design interventions that can more effectively induce a specific phenotypic outcome. But this is challenging since we want not only to predict gene expression at unobserved time-points, but also to make these predictions in a way that explains any prior knowledge of transcription factor binding sites. Models that accurately encode such interactions between transcription factors and target genes within gene regulatory networks (GRN) can provide insights into important cellular processes, such as disease-progression and cell-fate decisions [1–4].

Given that many dynamical systems can be described using ordinary differential equations (ODEs), a logical approach to modeling GRNs is to estimate ODEs for gene expression using an appropriate statistical learning technique [3–6]. Although estimating gene regulatory ODEs ideally requires time-course data, obtaining such data in biological systems is difficult (if not impossible given the destructive nature of the associated assays). One can instead use pseudotime methods applied to cross-sectional data to order samples and subsequently estimate ODEs that captures the regulatory structure [19, 29].

While a variety of ODE estimation methods have been proposed, most suffer from key issues that limit their applicability in modeling genome-wise regulatory networks. Some systems biology models formulate ODEs based solely on biochemical principles of gene regulation, and use the available data to parameterize these equations [22]. However, such methods impose several restrictions on the ODEs and cannot flexibly adjust to situations where the underlying assumptions do not hold; this increases the risk of model misspecification and hinders scalability to large networks, particularly given the enormous number of parameters necessary to specify a genome-scale model [21, 27]. Other methods are based on non-parametric methods to learning regulatory ODEs, using tools such as sparse kernel regression [3], random forests [5], variational auto-encoders [7, 28, 29], diffusion processes [4], and neural ordinary differential equations [6], but these fail to include biologically relevant associations between regulatory elements and genes as constraints on the models.

These latter models can be broadly placed into two classes based on the inputs required to estimate the gradient f of the gene regulatory dynamics, where $f(\boldsymbol{x}) = \frac{d\boldsymbol{x}}{dt}$. The first class consists of methods like PRESCIENT [4] and RNAForecaster [6] that can learn f based only on time series gene-expression input $\{\boldsymbol{x}_{t_0};\ \boldsymbol{x}_{t_1};\ \ldots;\ \boldsymbol{x}_{t_T}\}$ without additional steps or consideration of other regulatory inputs [4, 6, 8]. In the process of learning transitions between

consecutive time points, these "one-step" methods *implicitly* learn the local derivative (often referred to as "RNA velocity" [10]) $\frac{d\boldsymbol{x}}{dt}\,|_{\boldsymbol{x}=\boldsymbol{x}_{t_m}}$, as an intermediary to estimating f. One significant issue with these approaches is scalability, and studying meaningfully large dynamical systems (ideally those describing the entire genome) has been too costly in terms of runtime and performance loss [4, 6, 9, 20]. This leads to potential issues with generalizability as regulatory processes operate genome-wide and even small perturbations can have wide-ranging regulatory effects.

The second class consists of "two-step" methods such as dynamo [3], RNA-ODE [5], and scDVF [7] that aim to alleviate this performance loss by replacing the difficult task of learning f with two separate steps [3, 5, 7]. First, the RNA velocity ($\frac{d\boldsymbol{x}}{dt}$) is *explicitly* estimated for each data point in a preprocessing step, using spliced and unspliced transcript counts, and one of many available velocity estimation tools [3, 10–15]. In the next step, the original task of learning f is reduced to **learning a vector field** from expression-velocity tuples $[\boldsymbol{x}_i, (\frac{d\boldsymbol{x}}{dt})_i]$ and a suitable learning algorithm is deployed. Apart from needing additional inputs that may not always be available (for example, spliced and unspliced counts are not available for microarray data), these "two step" methods are also sensitive to the velocity estimation tool used, many of which suffer from a multitude of weaknesses [15]. Still, the Jacobian of the estimated vector field can help inform whether the learned dynamics are biologically meaningful [2, 3, 5, 19].

While the flexibility of both classes of models is helpful in estimating arbitrary dynamics, they are"black-box" methods whose somewhat opaque nature not only makes them prone to overfitting, but it makes it difficult to extract interpretable mechanistic insights about regulatory control [1, 6]. These models are optimized solely to predict RNA velocity or gene expression levels and so the predictions are not explainable in the sense that they cannot be related back to a sparse causal GRN [16]. Another major issue is the scalability of these methods; because of their computational complexity, they cannot feasibly scale to thousands of genes–and definitely not to the entire genome. Consequently, most of these methods either restrict themselves to a small set of highly variable genes or resort to dimension-reduction techniques (PCA, UMAP, latent-space embedding, etc.) as a preprocessing step [3, 4, 28, 29]. This leads to certain biological pathways being masked in the dynamics and impedes the recovery of causal GRNs. Finally, these models generally lack a means of incorporating biological constraints and prior knowledge to guide model selection and prevent overfitting [1, 17].

We developed **PHOENIX** (**P**rior-informed **H**ill-like **O**DEs to **E**nhance **N**euralnet **I**ntegrals with e**X**plainability) as a scalable method for estimating dynamical systems governing gene expression through an ODE-based machine learning framework that is flexible enough to avoid model misspecification

and is **guided by insights** from systems biology that facilitate biological interpretation of the resulting models [18, 41]. At its core, PHOENIX models temporal patterns of gene expression using neural ordinary differential equations (NeuralODEs) [31, 33], an advanced computational method commensurate with the scope of human gene regulatory networks – with more than 25,000 genes and 1,600 TFs – and the limited number of samples. We implement an innovative NeuralODE architecture that inherits the universal function approximation property (and thus the flexibility) of neural networks while resembling Hill–Langmuir kinetics (which have been used to model dynamic transcription factor biding site occupancy [24, 26, 27]) so that it can reasonably describe gene regulation by modeling the sparse yet synergystic interactions of genes and transcription factors.

Importantly, PHOENIX operates on the original gene expression space and does not require any dimensionality reduction, thus preventing information loss (especially for lowly-expressed genes that are nonetheless important for cell fate) [4]. This together with the incorporation of user-defined prior knowledge of likely network structure ensures that a trained PHOENIX model is **explainable** – it not only predicts temporal gene expression patterns, but also encodes an extractable GRN that captures key mechanistic properties of regulation such as activating (and repressive) edges and strength of regulation.

2 The PHOENIX model

Given a time series gene expression data set, the NeuralODEs of PHOENIX *implicitly* estimate the local derivative (RNA velocity) at an input data point with a neural network (NN). We designed activation functions that resemble Hill-kinetics and thus allow the NN to sparsely represent different patterns of transcriptional co-regulation by combining separate additive and multiplicative blocks that operate on the linear and logarithmic scales respectively. An ODE solver then integrates the estimated derivative to reconstruct the steps taken from an input x_i at time t_i to a predicted output $\widehat{x}_{i+1}$ at time t_{i+1} [31]. The trained neural network block thus encodes the ODEs governing the dynamics of gene expression, and hence encodes the underlying vector field and GRN. An important advantage of incorporating an ODE solver is that we can predict expression-changes for arbitrarily long time intervals without relying on predefined Euler discretizations, as is required by many other methods [4, 7, 20]. We further augmented this framework by allowing users to include **prior knowledge** of gene regulation in a flexible way, which acts as a domain-knowledge-informed regularizer or soft constraint of the NeuralODE [17] (**Figure 1**). By combining the **mechanism-driven** approach of systems biology inspired functional forms and prior knowledge with the **data-driven** approach of powerful machine learning tools, PHOENIX scales up to full-genome data sets and learns meaningful models of gene regulatory dynamics.

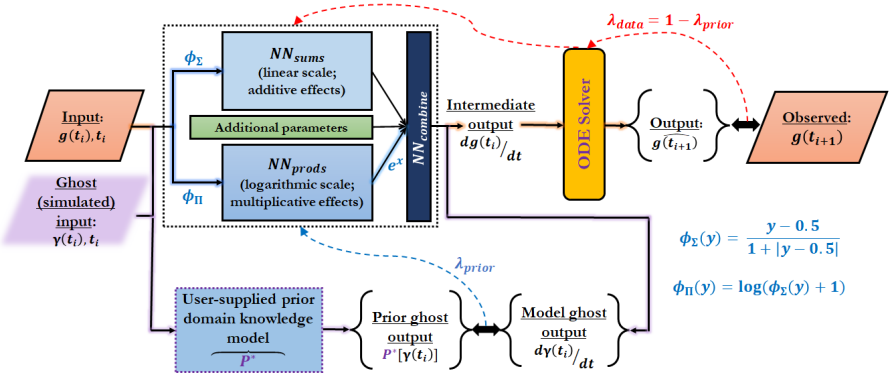

Fig. 1 PHOENIX is powered by a NeuralODE engine. Given an expression vector $\boldsymbol{g}(t_i) \in \mathbb{R}^{\#\text{genes}}$ at time t_i, a neural network (dotted rectangle) estimates the local derivative $d\boldsymbol{g}(t_i)/dt$ and an ODE solver integrates this value to predict expression at subsequent time points $\hat{\boldsymbol{g}}(t_{i+1})$. The neural network is equipped with activation functions (ϕ_Σ and ϕ_Π) that resemble Hill-Langmuir kinetics, and two separate single-layer blocks (NN_{sums} and NN_{prods}) that operate on the linear and logarithmic scales to model additive and multiplicative co-regulation respectively. A third block ($\text{NN}_{combine}$) then flexibly combines the additive and multiplicative synergies. PHOENIX incorporates two levels of back-propagation to parameterize the neural network while inducing domain knowledge-specific properties; the first (red arrows with weight λ_{data}) aims to match the observed data, while the second (blue arrow with weight λ_{prior}) uses simulated (ghost) expression vectors $\boldsymbol{\gamma}(t_i)$ to implement soft constraints defined by user-supplied prior models ($\mathcal{P}^*$) of putative regulatory interactions.

2.1 Neural ordinary differential equations (NeuralODEs)

NeuralODEs [31] learn dynamical systems by parameterizing the underlying derivatives with neural networks:

$$\frac{d\boldsymbol{x}(t)}{dt} = f(\boldsymbol{x}(t), t) \approx \text{NN}_\theta(\boldsymbol{x}(t), t)$$

Given an initial condition, the output at any given time-point can now be approximated using a numerical ODE solver $\mathcal{S}$ of adaptive step size:

$$\widehat{\boldsymbol{x}(t_1)} = \boldsymbol{x}(t_0) + \int_{t_0}^{t_1} \text{NN}_\theta(\boldsymbol{x}(t), t)dt = \mathcal{S}(\boldsymbol{x}(t_0);\ \text{NN}_\theta;\ t_0;\ t_1)$$

This is the basic architecture of a NeuralODE [31], and it lends itself to loss functions L (e.g. ℓ_2 loss) of the form:

$$L\Big(\boldsymbol{x}(t_1), \widehat{\boldsymbol{x}(t_1)}\Big) = L\Big(\boldsymbol{x}(t_1), \mathcal{S}(\boldsymbol{x}(t_0);\ \text{NN}_\theta;\ t_0;\ t_1)\Big).$$

To perform back-propagation, the gradient of the loss function with respect to all parameters θ must be computed, which is done using the adjoint sensitivity method [31]. Building off of the NeuralODE author's model implementation in `PyTorch` [33], we made biologically-motivated modifications to the architecture, and incorporated user-defined prior domain knowledge.

2.2 Model formulation and neural network architecture

Most models for co-regulation of gene expression are structured as a simple feedback process [24]. Given that gene regulation can be influenced by perturbations across an entire regulatory network of n genes, the gene expression of all genes $g_j(t)$ can have an effect on a specific $g_i(t)$ at time point t:

$$\frac{d\boldsymbol{g}(t)}{dt} = f_{reg}\Big(\boldsymbol{g}(t)\Big) - \boldsymbol{g}(t)$$

where $\boldsymbol{g}(t) = \{g_i(t)\}_{i=1}^{n}$, $f_{reg} : \mathbb{R}^n \to \mathbb{R}^n$, and f_{reg} is approximated with a neural network. To model additive as well as multiplicative effects within f_{reg}, we used an innovative neural network architecture equipped with activation functions that emulate – and can thus sparsely encode – Hill-kinetics (see **Figure 1**). The Hill–Langmuir equation $H(P)$ was originally derived to model the binding of ligands to macromolecules [26], and can be used to model transcription factor occupancy of gene regulatory binding sites [27]:

$$H(P) = \frac{P^\alpha}{\kappa^\alpha + P^\alpha} = \frac{(P/\kappa)^\alpha}{1 + (P/\kappa)^\alpha} = \frac{Y}{1+Y}, \text{with } Y = (P/\kappa)^\alpha,$$

which resembles the softsign activation function $\phi_{\text{soft}}(y) = 1/(1 + |y|)$. For better neural network trainability, however, we shifted it to the center of the expression values. To approximate suitable exponents α, we further log transformed H, since composing additive operations in the log transformed space with a Hadamard $\exp\circ$ function can represent multiplicative effects. Thus,

$$\phi_\Sigma(x) = \frac{x - 0.5}{1 + \mid x - 0.5 \mid}, \qquad \phi_\Pi(x) = \log\left(\frac{x - 0.5}{1 + \mid x - 0.5 \mid} + 1\right).$$

were employed as activation functions to define two neural network blocks (NN_{sums} and NN_{prods}), representing additive and multiplicative effects:

$$\boldsymbol{c_\Sigma}(\boldsymbol{g}(t)) = \boldsymbol{W_\Sigma}\phi_\Sigma(\boldsymbol{g}(t)) + \boldsymbol{b_\Sigma} \qquad \boldsymbol{c_\Pi}(\boldsymbol{g}(t)) = \exp\circ(\boldsymbol{W_\Pi}\phi_\Pi(\boldsymbol{g}(t)) + \boldsymbol{b_\Pi}).$$

The concatenated vectors $\boldsymbol{c_\Sigma}(\boldsymbol{g}(t)) \oplus \boldsymbol{c_\Pi}(\boldsymbol{g}(t))$ served as input to a third block $\text{NN}_{combine}$ (with weights $\boldsymbol{W}_\cup \in \mathbb{R}^{n \times 2m}$) that flexibly combined these additive and multiplicative effects. We found that a single linear layer was sufficient for this purpose. To simplify the representation of steady state for genes without upstream transcription factors $\left(\frac{dg_i(t)}{dt} = 0, \forall t\right)$, we introduced gene specific parameters $\boldsymbol{v} \in \mathbb{R}^n$. Accordingly, the output derivative for each gene i was multiplied with $\text{ReLU}(v_i) = \max\{(v_i, 0)\}$. We expressed this using the Hadamard product ($\odot$) of the previous output and the elementwise ReLU of $\boldsymbol{v}$:

$$\widehat{\frac{d\boldsymbol{g}(t)}{dt}} = \text{ReLU}(\boldsymbol{v}) \odot \Big[\boldsymbol{W}_\cup\{\boldsymbol{c_\Sigma}(\boldsymbol{g}(t)) \oplus \boldsymbol{c_\Pi}(\boldsymbol{g}(t))\} - \boldsymbol{g}(t)\Big].$$

The trainable parameters $\boldsymbol{\theta} = (\boldsymbol{W_\Sigma}, \boldsymbol{W_\Pi}, \boldsymbol{b_\Sigma}, \boldsymbol{b_\Pi}, \boldsymbol{W}_\cup, \boldsymbol{v})$ were learned based on observed data and prior domain knowledge (details in SM.1).

2.3 Structural domain knowledge incorporation

One challenge we found in interpreting PHOENIX is that NeuralODEs have multiple solutions [43], of which many are inconsistent with our understanding of the process by which specific transcription factors (TFs) regulate the expression of other genes within the genome. Most solutions accurately represent gene-gene correlations, but do not necessarily reflect biologically established TF-gene regulation processes. Inspired by recent developments in physics-informed deep learning [17], we introduced biologically-motivated soft constraints to regularize the search for a parsimonious approximation. We started with the NeuralODE prediction (see 2.1) for the gene expression vector:

$$\begin{aligned}\widehat{\boldsymbol{g}(t_1)} &= \boldsymbol{g}(t_0) + \int_{t_0}^{t_1} \mathrm{ReLU}(\boldsymbol{v}) \odot \Big[\boldsymbol{W}_\cup\{\boldsymbol{c_\Sigma}(\boldsymbol{g}(t)) \oplus \boldsymbol{c_\Pi}(\boldsymbol{g}(t))\} - \boldsymbol{g}(t)\Big] dt \\ &= \mathcal{S}\Big(\boldsymbol{g}(t_0);\ \mathrm{ReLU}(\boldsymbol{v}) \odot \Big[\boldsymbol{W}_\cup\{\boldsymbol{c_\Sigma}(\boldsymbol{g}(t)) \oplus \boldsymbol{c_\Pi}(\boldsymbol{g}(t))\} - \boldsymbol{g}(t)\Big];\ t_0;\ t_1\Big)\end{aligned}$$

We observed that the unregularized PHOENIX provides an **observed gene expression-based** approximation for the local derivative $\frac{d\boldsymbol{g}(t)}{dt}$, but often we have additional structural information available about which TFs are more likely to regulate certain target genes. Hence, one could also formulate a domain knowledge-informed $\mathcal{P}^*\big(\boldsymbol{g}(t)\big)$ that is a **prior-based** approximation:

$$\underbrace{\mathrm{ReLU}(\boldsymbol{v}) \odot \Big[\boldsymbol{W}_\cup\{\boldsymbol{c_\Sigma}(\boldsymbol{g}(t)) \oplus \boldsymbol{c_\Pi}(\boldsymbol{g}(t))\} - \boldsymbol{g}(t)\Big]}_{\text{PHOENIX (based on observed gene expression data)}} \approx \frac{d\boldsymbol{g}(t)}{dt} \approx \underbrace{\mathcal{P}^*\Big(\boldsymbol{g}(t)\Big)}_{\text{prior-based}}.$$

By promoting our NeuralODE to flexibly align with such structural domain knowledge, we automatically searched for biologically more realistic models that still explained the observed gene expression data. To this end, we designed a modified loss function $\mathcal{L}_{\text{mod}}$ that incorporated the effect of prior model $\mathcal{P}^*$ using a set of K randomly generated expression vectors $\{\boldsymbol{\gamma_k} \in \mathbb{R}^n\}_{k=1}^K$. This induced a preference for consistency with prior domain knowledge.

$$\begin{aligned}&\mathcal{L}_{\text{mod}}\Big(\boldsymbol{g}(t_1), \widehat{\boldsymbol{g}(t_1)}\Big) \\ &= \overbrace{\lambda\, L\Big[\boldsymbol{g}(t_1), \mathcal{S}\Big(\boldsymbol{g}(t_0);\ \mathrm{ReLU}(\boldsymbol{v}) \odot \Big[\boldsymbol{W}_\cup\{\boldsymbol{c_\Sigma}(\boldsymbol{g}(t)) \oplus \boldsymbol{c_\Pi}(\boldsymbol{g}(t))\} - \boldsymbol{g}(t)\Big];\ t_0;\ t_1\Big)\Big]}^{\text{loss based on matching observed gene expression data}} \\ &\quad + (1-\lambda)\underbrace{\frac{1}{K}\sum_{k=1}^{K} L\Big[\mathcal{P}^*\big(\boldsymbol{\gamma_k}\big), \mathrm{ReLU}(\boldsymbol{v}) \odot \Big[\boldsymbol{W}_\cup\{\boldsymbol{c_\Sigma}(\boldsymbol{\gamma_k}) \oplus \boldsymbol{c_\Pi}(\boldsymbol{\gamma_k})\} - \boldsymbol{\gamma_k}\Big]\Big]}_{\text{loss based on matching prior model}}\end{aligned}$$

Here, λ is a tuning parameter for flexibly controlling how much weight is given to the prior-based optimization, which we tuned with cross-validation, and $L[\boldsymbol{x}, \widehat{\boldsymbol{x}}]$ is the primary loss function, set to the $L = \ell_2$ loss in our experiments.

While our modeling framework is flexible regarding the nature of the prior model $\mathcal{P}^*$, we incorporated a simple linear model, a common choice for chemical reaction networks or simple oscillating physical systems [44]. We used the adjacency matrix $\boldsymbol{A}$ of likely network structure based on prior domain knowledge (e.g. experimentally validated interactions, motif map of promoter targets, etc.) with known activating and repressive edges set to +1 and -1 respectively. For cases where the sign (activating/repressive) was unknown, we randomly assigned +1 or -1 with uniform probability.

$$\mathcal{P}^*(\boldsymbol{\gamma_k}) = \boldsymbol{A} \cdot \boldsymbol{\gamma_k} - \boldsymbol{\gamma_k} = (\boldsymbol{A} - \boldsymbol{I}) \cdot \boldsymbol{\gamma_k}$$

3 Results

We demonstrate the utility of PHOENIX for estimating gene expression dynamics by performing a series of *in silico* benchmarking experiments, where PHOENIX exceeds even the most optimistic performance of popular black-box RNA dynamics estimation methods. We demonstrate the scalability of PHOENIX by applying it to genome-scale breast cancer samples ordered in pseudotime and investigate how scaling to the complete data set improves a representation of key pathways. Finally, we apply PHOENIX to yeast cell cycle data to show that it can capture oscillatory dynamics by flexibly deviating from Hill-like assumptions when necessary.

3.1 PHOENIX accurately and explainably learns temporal evolution of *in silico* dynamical systems

We began our validation studies with simulated gene expression time-series data so that the underlying dynamical system that produced the system's patterns of gene expression was known. We adapted `SimulatorGRN` [23] (a simulator used extensively by the well cited `R`/Bioconductor package `dcanr` [24]) to generate time-series expression data from two synthetic *S. cerevisiae* gene regulatory systems (SIM350 and SIM690, consisting of 350 and 690 genes respectively). The activating and repressive interactions in each *in silico* system was used to synthesize noisy expression "trajectories" for each gene across multiple time-points (see Methods 5.1.1 and 5.1.2). We split up the trajectories into training (88%), validation (6% for hyperparameter tuning), and testing (6%), and compared PHOENIX predictions on the test set against the "known"/ground truth trajectories (detailed results in SR.1). Since PHOENIX uses user-defined prior knowledge as a regularizer, we also corrupted the prior model ($p(\boldsymbol{\gamma_k})$ from 2.3) at a level commensurate with the "experimental" noise level (see SM.4.2), reflecting the fact that transcription factor-gene binding is itself noisy.

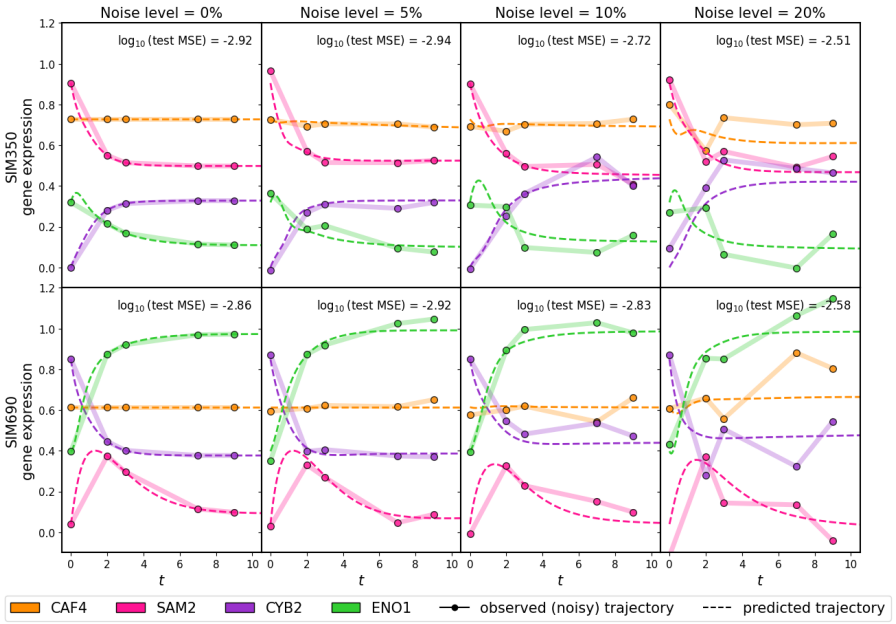

Fig. 2 We applied PHOENIX to simulated gene expression data originating from two different *in silico* dynamical systems SIM350 (top row) and SIM690 (bottom row) that simulate temporal expression of 350 and 690 genes respectively. Each simulated trajectory consisted of five time-points ($t = 0, 2, 3, 7, 9$) and was subjected to varying levels of Gaussian noise ($\frac{\text{noise } \sigma}{\text{mean}}$ = 0%, 5%, 10%, 20%). Since PHOENIX uses a user-defined prior network model as a regularizer, we also corrupted the prior-models up to an amount commensurate with the noise level. For each noise setting we trained PHOENIX on 140 of these "observed" trajectories and validated on 10. The performance on the validation trajectories was used to determine the optimal value of λ_{prior}. We then tested the trained model (with the optimal choice of λ_{prior}) on 10 new test set trajectories. We display both observed and predicted test set trajectories for four arbitrary genes in both SIM350 and SIM690, across all noise settings. We display the mean squared error (MSE) between the predictions and the 10 test set trajectories pre-noise.

We found that PHOENIX accurately learned the temporal evolution of the SIM350 and SIM690 data sets (**Figure 2**) and was able to recover the *true* test set trajectories (that is, test set trajectories pre-noise) with a reasonably high accuracy even when the training trajectories and prior knowledge model had included high levels of noise. Furthermore, the shapes of the predicted trajectories (and hence the predicted steady state levels) obtained from feeding initial values (expression at $t = 0$) into the trained model remained robust to noise, suggesting that a trained PHOENIX model could be used to estimate the temporal effects of cellular perturbations.

Since the primary prediction engine of PHOENIX is a NeuralODE, we wanted to benchmark its performance relative to "out-of-the-box" (OOTB)

NeuralODE models (such as RNAForecaster [6]) to understand the contributions of our modifications to the NeuralODE architecture. We tested a range of OOTB models (activation functions: ReLU, sigmoid, and tanh) where we adjusted the total number of trainable parameters to be similar to that of PHOENIX (see SM.3.2). Because PHOENIX uses a domain prior of likely gene-regulation interactions in its optimization scheme, we also tested a version (PHX_0) where the weight of the prior was set to zero ($\lambda_{\text{prior}} = 0$). For each of SIM350 and SIM690, we observed that PHOENIX outperformed OOTB NeuralODEs on the test set in noiseless settings (**Figure SR1** and **Table SR2**). When we added noise, the PHOENIX models still generally outperformed the OOTB models, especially in the larger SIM690. The validation MSEs were more comparable between all the models in the high noise setting in SIM350. The consistently strong performance of PHOENIX suggests that using a Hill-kinetics inspired NeuralODE architecture better captures the dynamics of the regulatory process, in part because it models the binding kinetics of transcription factor-gene interactions.

In terms of contribution of the prior to PHOENIX's performance, we observed that PHOENIX was generally outperformed by PHX_0, its unregularized version (**Figure SR1** and **Table SR2**). However, given that the prior can be interpreted as soft biological constraints on the estimated dynamical system [17], an important question is whether PHX_0 (as well as OOTB models) makes accurate temporal predictions by correctly learning elements of the causal biology, or whether the lack of prior information results in an alternate learned representation of the dynamics, which - despite predicting *these particular* trajectories well - does not explain the true biological regulatory process.

To this end, we recognized that the parameters of a trained PHOENIX model encode an estimate of the ground-truth gene regulatory network (GRN) that causally governs the system's evolution over time. We therefore inferred encoded GRNs from trained PHOENIX models and compared them to the ground truth networks GRN_{350} and GRN_{690} used to synthesize SIM350 and SIM690 respectively (see SM.2). Given PHOENIX's simple NeuralODE architecture, we were able to develop a GRN inference algorithm that could predict edge existence, direction, strength, and sign, using just model coefficients, without any need for time consuming sensitivity analyses (unlike other approaches [2, 7]). For comparison, we wanted to extract GRNs from the most predictive OOTB models; given their black-box nature, OOTB model GRNs had to be obtained via sensitivity analyses (see SM.3.2).

We compared inferred and ground truth GRNs in terms of several metrics, including edge recovery, out-degree correlations, and induced sparsity. We obtained near-perfect edge recovery for PHOENIX (AUC= 0.96 - 0.99) as well as high out-degree correlations across all noise settings (**Figure 3** and **Table SR3**). Most notably, we observed that PHOENIX predicted dynamics

in a more robustly explainable way than PHX_0 and the OOTB models. We measured induced sparsity by reverse engineering a metric $\mathcal{C}_{max}$ based on maximizing classification accuracy (see SM.2), and found that PHOENIX resulted in much sparser dynamics than PHX_0 (**Table SR4**). To further assess this phenomenon, we computed the estimated model effect between every gene pair in SIM350, and compared these values between PHOENIX and PHX_0. We found that that the incorporation of priors helped PHOENIX identify core elements of the dynamics, and predict gene expression patterns in a biologically parsimonious manner (**Figure SR2**).

Since the inclusion of such static prior knowledge greatly increased the explainability of the inferred dynamics, we also investigated how explainability was affected by misspecification of the prior. In our *in silico* experiments, we had randomly corrupted (misspecified) the prior by an amount commensurate with the noise level (see SM.4.2). We compared network representations of these misspecified prior constraints to GRNs extracted from the PHOENIX models that used these very priors. We found that PHOENIX was able to appropriately learn causal elements of the dynamics beyond what was encoded in the priors (**Table SR1**). This suggests that even though the user-defined priors enhance explainability, PHOENIX can deviate from them when necessary, and learn regulatory interactions from just the data itself.

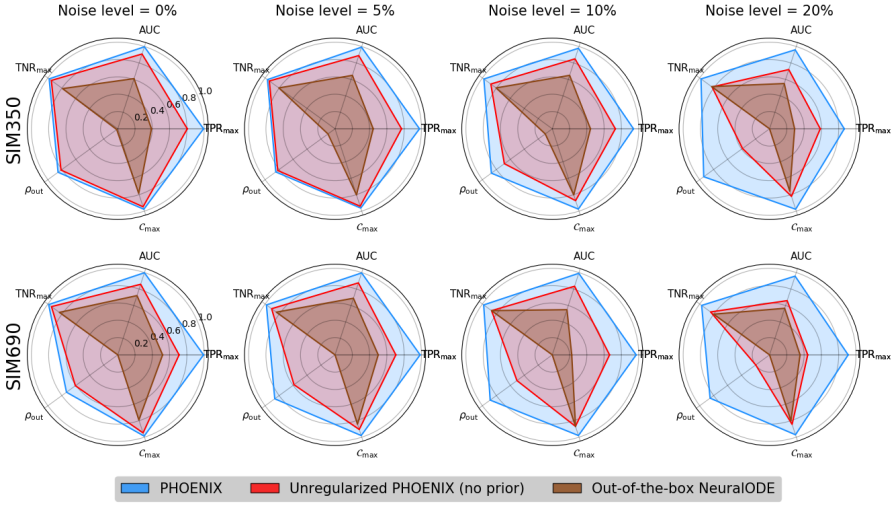

Fig. 3 We extracted encoded GRNs from the trained PHOENIX models and the best performing out-of-the-box NeuralODE models, for both *in silico* dynamical systems SIM350 (**top row**) and SIM690 (**bottom row**) across all noise settings. We compared these GRN estimates to the corresponding ground truth GRNs used to formulate SIM350 and SIM690, and obtained AUC values as well as out-degree correlations (ρ_{out}). We also reverse-engineered a metric ($\mathcal{C}_{max}$) to inform how sparsely PHOENIX had inferred the dynamics (see SM.2). Furthermore, we used these $\mathcal{C}_{max}$ values to obtain optimal true positive and true negative rates (TPR_{max} and TNR_{max}) that were independent of any cutoff value, allowing us to compare between "best possible" networks across all settings.

3.2 PHOENIX exceeds the most optimistic performances of current black-box methods *in silico*

Having established PHOENIX models as both predictive and explainable, we compared its performance to other existing methods for gene expression ODE estimation *in silico* (**Table 1**). As discussed earlier, these can be placed into two groups based on the input data. The "one-step" methods estimate dynamics by directly using expression trajectories; these include RNAForecaster [6] (which is an out-of-the-box NeuralODE), and PRESCIENT [4]; PHOENIX is more similar to these methods.

"Two-step" methods such as Dynamo [3], RNA-ODE [5], and scDVF [7] estimate dynamics by first reconstructing RNA velocity using inputs such as spliced and unspliced mRNA counts and then estimating a vector field mapping expression to velocity. To avoid the need for uncommon input data types and to also emulate the theoretically optimal performance in the first step of these "two-step" velocity-based methods, we used the **noiseless ground truth velocities** as input into their second step since the true velocities were known in our *in silico* experiments (see SM.3.1). Further, we used the validation set to optimize key hyperparameters of all the methods (**Table 1**) before finally testing predictive performance on expression values from held-out trajectories. Most of the methods also provide a means for extracting a gene network which we used to evaluate each method's explainability (see SM.3.3).

In these comparisons, we found that the "one-step" trajectory-based methods generally yielded better predictions than the "two-step" velocity-based methods (although Dynamo sometimes achieved performance compared to the single step methods). This makes sense since trajectory-based methods are optimized to predict gene expression trajectories while velocity-based methods predict trajectories by first optimizing RNA velocity estimations [1]. Overall, PHOENIX outperformed even the optimistic versions of the black-box methods by large margins both in terms of predicting gene-expression (**Table SR2**) and explainability (based on consistency with the ground truth network **Table SR3**). We found that Dynamo was generally the most explainable competing method but that, in some settings, RNA-ODE and scDVF were more explainable. Finally, we found that the dynamics estimated by PHOENIX were much sparser than any other method, and that this sparsity remained fairly insensitive to noise levels (**Table SR4**).

Further ODE estimation approaches and their functionalities are discussed in **Table SR5**. Code for performing such methodological benchmarks will be included with the PHOENIX release [34].

Table 1 Qualitative comparison of black-box methods for estimating gene expression dynamics that we benchmarked against PHOENIX *in silico*. We provide details about inputs, learning algorithms, and key performance metrics

	Method	Description of approach for estimating dynamical system	Scalability		Explainability		Notes
			Large-scale predictive performance (10^4+ genes)	Efficient without dimension reduction	Can extract GRN that describes dynamics	Flexibly incorporates prior/domain knowledge	✳ Hyperparameters we optimized using val. set ✂ GRN extraction details
Velocity based (two step)	Dynamo [3]	Uses time-resolved metabolically labelled spliced and unspliced scRNA to estimate RNA velocity ($d\boldsymbol{x}/dt$), then fits a sparse vector field mapping expression to velocity using Gaussian kernel regression	✘ (fewer than 300 genes)	✘ (top30 PC/ 2D UMAP)	✓	✘	✳ Control points (M) and sparsity parameter (λ) ✂ Monte Carlo approach to estimate GRN using Jacobians provided by tool (see SM.3.3)
	RNA-ODE [5]	Uses scRNA transcriptome and RNA velocity to fit random forests mapping expression to velocity	✘ (3001 genes)	✓	✓	✘	✳ Number of trees (nTrees) ✂ Tool provides GRN (see SM.3.3)
	scDVF [7]	Uses mRNA counts and corresponding and RNA velocity (from scVelo [11]) to fit variational autoencoders mapping expression to velocity	✘ (3000 high var. genes)	✓	✓	✘	✂ Tool provides gene correlation matrix by simulating retrograde trajectories (see SM.3.3)
Traj. based (one step)	PRESCIENT [4]	Uses time-series scRNA-seq and cell-growth rate data to learn a potential function Ψ with a neural network. Final drift model is obtained using automatic differentiation $\mu = -\nabla\Psi$	✘ (2500 highly var. genes)	✘ (top50 PC/ top30 PC)	✘	✘	✳ Number of neurons inside each hidden layer (k_{dim}) (see SM.3.3)
	Out-of-the-box NeuralODE [6, 9]	Uses time-series expression and NeuralODE out-of-the-box, with traditional activation funcions	✘ (2000 highly expr. genes)	✓	✓	✘	✂ Monte Carlo approach using sensitivity analysis (see SM.3.2)
	PHOENIX	Our method	✓	✓	✓	✓	✳ Prior weight (λ_{prior}) ✂ Algo. described in SM.2

3.3 PHOENIX predicts temporal evolution of yeast cell-cycle genes in an explainable way

We then tested PHOENIX using an experimental data set from cell-cycle synchronized yeast cells, consisting of two technical replicates of expression values for 3551 genes across 24 time points (see Methods 5.2.1 for data processing).

Since there were two technical replicates (or trajectories), one approach for training PHOENIX was to use one replicate for training and the other for validation. However, given the extreme similarity between the two replicates in terms of gene expression values across all 24 times points, this would have led to artificially good performance on the validation trajectory. Instead of splitting by trajectories here, we used an alternate strategy and split the data based on transition pairs (see Methods 5.2.2), splitting the 46 transition pairs into training (40, 86%), validation (3, 7%), and test (3, 7%). For the domain prior we used a simplistic adjacency-matrix-based prior model derived from a motif map of promoter targets for each of the 3551 genes (see Methods 5.2.2). We tuned the prior weight (to $\lambda_{\text{prior}} = 0.10$) using the validation set to induce higher explainability by promoting a sparse GRN structure.

PHOENIX was able to learn the temporal evolution of gene expression across the yeast cycle; it was able to explain over 84% of the variation in the test set (**Figure 4**, bottom-left). Notably, when we visualized the estimated dynamics by extrapolating from just initial values (expression at $t = 0$), we found that PHOENIX plausibly predicted continued periodic oscillations in gene expression, even though the training data consisted of only two full cell cycles (**Figure 4**, top). The amplitude of the predicted trajectories dampened across time points, which is expected given that yeast array-data tends to exhibit underdamped harmonic oscillation during cell division possibly reflecting de-synchronization of the yeast cells [46]. This performance on oscillatory dynamics is indicative of the high flexibility of PHOENIX, which inherits the universal function approximation property from NeuralODEs, allowing it to deviate from Hill-like assumptions when necessary, while still remaining explainable due to the integration of prior knowledge.

To test the biological explainability of the learned dynamical system, we extracted the encoded GRN from the trained PHOENIX model (with optimal $\lambda_{\text{prior}} = 0.10$ as determined by the validation set) and compared to a validation network of ChIP-chip transcription factor (TF) binding data [40]. PHOENIX had impressive accuracy in predicting TF binding (AUC = 0.86), indicating that it had learned transcription factor binding information in the process of explaining temporal patterns in expression (**Figure 4**, bottom-right). In the absence of any prior-knowledge ($\lambda_{\text{prior}} = 0$), the explainability was poor, highlighting the importance of such knowledge-based guidance in black-box models [17, 18].

Similar to the *in silico* experiments, we saw that PHOENIX's ability to predict TF binding was greater than that obtained by comparing just the prior to the validation data (**Table SR1**). This suggested that PHOENIX had used the prior knowledge of cell cycle progression as a starting point to anchor the dynamics, and then used the data itself to learn improved regulatory rules.

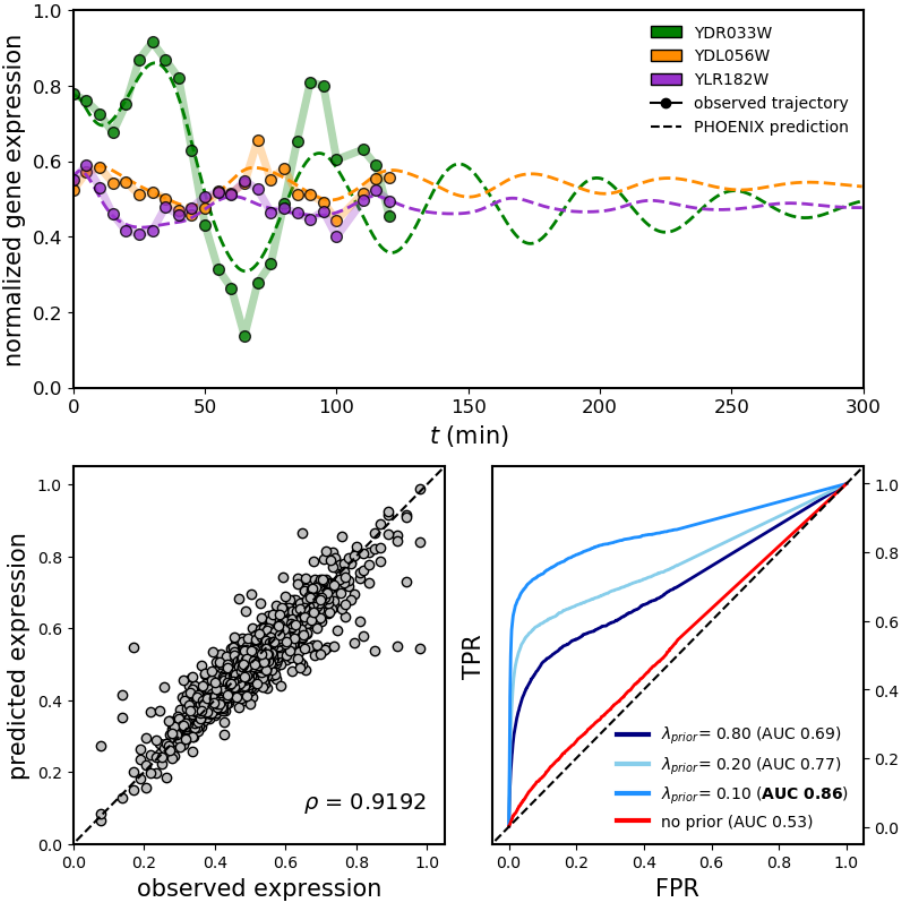

Fig. 4 (top) We applied PHOENIX ($\lambda_{prior} = 0.10$) to 2 technical replicates of gene expression of 3551 genes each, collected across 24 time points in a yeast cell-cycle time course [45]. We trained on 40 transition pairs, used 3 for validation, and tested predictive accuracy on the remaining 3. We display both observed and predicted trajectories for 3 arbitrary genes, where the predicted trajectories are extrapolations into future time points based on just initial values (gene-expression at $t = 0$). **(bottom-left)** We correlated observed versus predicted expression levels of all 3551 genes for the 3 expression vectors in the test set; $\rho = 0.919$ implying $R^2 = 0.844$. **(bottom-right)** We tested the explainability of the learned dynamics by comparing encoded GRNs retrieved from a series of trained models (of varying prior-dependencies) against ChIP-chip data [40] to obtain ROC curves. The $\lambda_{\text{prior}} = 0.10$ model was the one chosen based on validation set MSE (see SM.1).

3.4 PHOENIX infers genome-wide dynamics of breast cancer progression and identifies central pathways

Although there are a number of tools for inferring dynamics of regulatory networks, most do not scale beyond a few hundreds of genes, falling far short of the 25,000 genes in the human genome (**Table 1**). Given the performance improvements we saw that were driven by PHOENIX's use of soft constraints, we wanted to test whether PHOENIX could be extended to human-genome scale networks. Due to the dearth of longitudinal human studies with genome-wide expression measurements, we used data from a cross-sectional breast cancer study (GEO accession GSE7390 [47]) consisting of microarray expression values for 22000 genes from 198 breast cancer patients, and ordered these samples in pseudotime. For consistency in pseudotime ordering, we reused a version of this data that was already preprocessed and ordered (using a random walk based pseudotime approach) in the PROB paper [19].

We limited our analysis to the genes that had measurable expression and appeared in our regulatory prior, and obtained a single pseudotrajectory of expression values for $n_g = 11165$ genes across 186 patients, each at a distinct pseudo-timepoint. To explore whether PHOENIX's performance depends on the size of the data set, we created pseudotrajectories for $n_g = 500$, 2000, and 4000 genes by subsetting the data set to its n_g most variable genes. Similar to the yeast example, we split up the 185 transition pairs into training (170, 90%), validation (8, 5%), and test (7, 5%). For the domain prior network, we again used a simplistic prior model derived from a motif map of promoter targets (see Methods 5.3.2), and tuned λ_{prior} using the validation set.

For each pseudotrajectory of size n_g, we fit a separate PHOENIX model and calculated variation explained in the test set. We observed very impressive predictive performance, with test set R^2 values in the 97% - 99% range (**Figure 5**, top). We found that even though PHOENIX's computational cost increased as we scaled to $n_g = 11165$ genes, the cost was not excessive even at this genome-scale (see **Table SR7**). It is noteworthy that this type of feasible scalability to $n_g > 10^4$ genes is unprecedented in tools for estimating gene regulatory dynamical systems (**Table 1**); other methods either focus on a subset of highly variable genes [6, 7] or model dynamics in a less interpretable, lower dimensional PCA/UMAP/latent space [3, 4, 28].

Next, we investigated PHOENIX's ability to identify biologically relevant and actionable information regarding gene regulation in breast cancer. First, we tested the performance of the learned dynamical system to reconstruct a gene regulatory network and predict TF-gene interactions. While the ground truth GRN is unknown, we can estimate performance by comparing a validation network of experimental ChIP-chip binding information [48] to a subnetwork of the encoded GRN of a trained PHOENIX model. We found good alignment between the two GRNs (AUC = 0.81 - 0.91) even when we scaled up to

$n_g = 11165$ genes (**Figure 5**). It is important to note that the PHOENIX-based concordance with experimental data was generally greater than that obtained by comparing just the prior knowledge to the validation network (**Table SR1**), indicating that PHOENIX was improving upon the GRN suggested by the prior knowledge, in addition to learning a dynamical model.

To better understand the benefits of PHOENIX's scalability, we investigated how estimating regulatory dynamics based on a subset of only the n_g most variable genes can alter the perceived importance of individual genes to the regulatory system in question. We reasoned that a model trained on all assayed genes should reconstruct biological information better than those that are restricted to a subset of genes [6, 7]. First, we performed a gene-level analysis by perturbing *in silico* the PHOENIX-estimated dynamical system from each value of n_g (500, 2000, 4000, 11165). This yielded "influence scores" representing how changes in initial ($t = 0$) expression of each gene affected subsequent ($t > 0$) predicted expression of all other genes (see Methods 5.3.3). As might be expected, the influence scores grew increasingly more concordant with centrality measures in the ChIP validation network, consistent with the key roles played by transcription factor genes in large GRNs (**Table SR7**).

We observed that highly variable genes with known involvement in breast cancer (such as WT1 [53], ESR1 [54], AR [51], and FOXM1 [52]) were generally influential across all values of n_g (**Figure SR3**). It is interesting to note that both WT1 and FOXM1 were very influential in the $n_g = 500$ system, but their score drops in the full genome ($n_g = 11165$) system. This is likely due to the way in which we constructed the smaller subsets of the whole genome – by selecting the most variable genes. One would expect that the most variable transcription factor genes falling within any subset would be highly correlated in expression with other genes falling in the same set, and that the overall effect would be diluted by adding more - potentially uncorrelated - genes to the system. It is more interesting that genes missing in the smaller subsets (due to low expression variability) were identified as central to the dynamics in the full ($n_g = 11165$) system . Among these genes, we can find some encoding cancer-relevant transcription factors such as E2F1 and CTCF, members of the TP53 family (TP73), DNA methyltransferase enzymes (DNMT1), and members of the KLF family.

We found that the more computationally manageable systems ($n_g = 500, n_g = 2000$) yielded an incomplete picture of gene-level influences, since the the method used in constructing these subsets hinders the mechanistic explainability of the resulting regulatory model. Certain genes exhibit relatively low variability in expression but are still central to disease-relevant genome-level dynamics; compared to methods that exclude such genes to make computation tractable [4, 6, 7], PHOENIX can correctly identify such as central because of its ability to model subtle but important genome-scale dynamics.

Finally, we performed a pathway-based functional enrichment analysis by translating these gene influence scores to pathway influence scores using permutation tests on the **Reactome pathway** database [58] (see Methods 5.3.4). Not surprisingly, the dynamical systems with fewer genes missed many pathways known to be associated with breast cancer that were identified as over-represented in the genome-scale ($n_g = 11165$) system (**Figure 5** and **Table SR6**). Notably, the pathways missed in the smaller networks include apoptosis regulation (a hallmark of cancer [59]), estrogen-related signalling (whose role in breast cancer is well documented [60]), regulation of beta cells (relevant for immune processes [61]), and TP53 regulation of caspases (relevant to apoptosis control in tumors [62]).

In a parallel analysis testing for functional enrichment of **GO biological process** terms, we again found the smaller systems to overlook important pathways that were clearly influential in the genome-scale analysis; these included developmental processes associated with epithelial tumor development, germ cell development (possibly linked to the regulation of cancer testis antigens), and a wide array of RNA metabolism processes that are increasingly recognized as being significant to breast cancer development [63] (**Figure SR4**). Similarly, for **GO molecular function** terms, the smaller gene subsets missed key processes such as estrogen receptor binding (**Figure SR5**).

These results clearly demonstrate the importance of scalable methods such as PHOENIX that can model genome-wise dynamics. Our reduced gene sets from which we built the smaller PHOENIX models consisted of the 500, 2000, or 4000 most variable genes. These gene sets likely consist of variable genes that are correlated with each other, meaning that we are sampling only a portion of the biological processes driving the temporal changes in breast cancer; the full picture only emerges when looking at regulatory processes across the spectrum of genes that can contribute. Alternative approaches, such as concentrating on specific pathways, risk introducing self-fulfilling biases in the discovery process. Similarly, methods that use low-dimensional embedding to reduce the complexity of modeling dynamics risk obscuring losing valuable, biologically relevant insights. PHOENIX's scalability offers the best potential for discovery of interpretable insights with high explainability relative to the phenotypes under study.

Fig. 5 (top panel) We applied PHOENIX to a pseudotrajectory of 186 breast cancer samples (ordered along subsequent "pseudotimepoints") consisting of $n_g = 11165$ genes [47]. We trained on 170 transition pairs, used 8 for validation (to tune λ_{prior}), and tested predictive accuracy on the remaining 7. We also repeated the analysis on smaller subsets of genes $n_g = 500, 2000, 4000$, where we subsetted the full trajectory to only the n_g most variable genes in the pseudotrajectory. We display both the predictive performance (R^2 on the test set) and the explainability performance (AUC from comparing encoded GRNs from trained models against a ChIP-seq validation network [48]). **(main panel)** We used the trained PHOENIX models to extract permutation-based influence scores for pathways in the **Reactome database** [58] (see Methods 5.3.3 and 5.3.4), and visualized influence scores for a collection of the most central pathways. See Table SR6 for detailed results.

4 Discussion

Given the importance of regulatory networks and their dynamics, there has been a tremendous interest in inferring and modeling their physical and temporal behavior. The use of NeuralODEs represents an extremely promising technology for inferring such networks, but so far, attempts to implement NeuralODE-based network modeling have encountered significant problems, not the least of which has been their inability to scale to modeling genome-wide dynamics in a biologically explainable manner.

PHOENIX represents an important new methodological extension to the NeuralODE framework that is not only scaleable to the full human genome, but also biologically well interpretable and able to capture explicitly both additive as well as multiplicative ways in which transcription factors cooperate in regulating gene expression. For a simplified analysis, the underlying gene regulatory network can also be extracted from a learned model and compared with experimental evidence. An optional feature of PHOENIX that contributes significantly to its explainability is that it can be guided by (structural) domain knowledge. Notably, PHOENIX also remains flexible to deviate from domain knowledge when necessary, and learn novel insights consistent with the training data

The predictive accuracy, scaleabilty, flexibility, and biological explainability can be attributed primarily to two things. First, our novel NeuralODE architecture that includes the use of Hill-like activation functions for capturing the kinetic properties of molecular binding provides a massive advantage in terms of predictive power. And second, the introduction of soft constraints based on prior knowledge of putative network structure leads to a scalable and biologically explainable estimate of the underlying dynamics.

Using simulated data we have shown that PHOENIX outperforms other models for inferring regulatory dynamics (including other NeuralODE-based models), particularly in the presence of experimental noise. Also, an application to data from the yeast cell cycle elucidates PHOENIX's flexibility in modelling arbitrary dynamics. More importantly, PHOENIX is the only NeuralODE method capable of extending its modeling to capture genome-scale regulatory processes. Using data from breast cancer patients organized in pseudotime we illustrate not only the ability of PHOENIX to faithfully model genome-scale networks, but also demonstrate the power of extending regulatory modeling to capture seemingly subtle but biologically important regulatory processes.

5 Methods

5.1 Testing on simulated data

5.1.1 Defining a ground truth dynamical system

We created a ground truth gene regulatory network (GRN) by sampling from S. cerevisiae (yeast) regulatory networks obtained from the SynTReN v1.2 supplementary data in simple interaction format (SIF) [35]. The SynTReN file provides a directional GRN containing 690 genes and 1094 edges with annotations (activating vs repressive) for edge types; we defined this GRN to be ground truth network G_{690}. To obtain G_{350}, we sampled a subnetwork of 350 genes and 590 edges from G_{690}. We used the connectivity structure of G_{350} and G_{690}, to define systems of ODEs (SIM350 and SIM690) with randomly assigned coefficients. This entire pipeline was executed using `SimulatorGRN` [23], a framework used extensively by the `R`/Bioconductor package `dcanr` [24]. Please see SM.4.1 for futher ODE formulation details from `SimulatorGRN`.

5.1.2 Simulating time series gene expression data

For each $n \in \{350, 690\}$, we used the ground truth dynamical system SIMn to generate expression vectors $\boldsymbol{g}(t) \in \mathbb{R}^n$, across time points t. We started by i.i.d. sampling 160 standard uniform $\mathbb{R}^n$ vectors to act as initial ($t = 0$) conditions. We used these initial conditions to integrate SIMn and obtain 160 expression trajectories across $t \in T = \{0, 2, 3, 7, 9\}$ using `R`'s `desolve` package: $\{\{\boldsymbol{g}(t)_i\}_{t\in T}\}_{i=1}^{160}$. We used only five time points to emulate potential scarcities of time-series information in real data sets, while the range $t = 0$ to 9 generally covered the transition from initial to steady state. Lastly, we added Gaussian noise vectors $\boldsymbol{\varepsilon}(t, \sigma)_i \overset{i.i.d}{\sim} \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ of varying σ to get noisy data sets: $\left\{\{\{\boldsymbol{g}(t)_i + \boldsymbol{\varepsilon}(t, \sigma)_i\}_{t\in T}\}_{i=1}^{160}\right\}_{\sigma\in S}$. Since the average simulated expression value was ≈ 0.5, using $\sigma \in S = \{0, \frac{1}{40}, \frac{1}{20}, \frac{1}{10}\}$ corresponded roughly to average noise levels of $0\%, 5\%, 10\%, 20\%$.

5.1.3 Model setup for training and testing

For each simulation scenario, there were 160 simulated trajectories, out of which we used 140 (88%) for training, 10 (6%) for validation (hyperparameter tuning) and 10 (6%) for testing. We provide some details on PHOENIX implementation (e.g. training strategy, prior incorporation, etc) in SM.1, and include finer technicalities (e.g. exact learning rates) in our GitHub repository [34]. For prior domain knowledge model we used the simple linear model: $\mathcal{P}^*(\boldsymbol{\gamma_k}) = \boldsymbol{A}^{\boldsymbol{\sigma}\%} \cdot \boldsymbol{\gamma_k} - \boldsymbol{\gamma_k}$, where we chose $\boldsymbol{A}^{\boldsymbol{\sigma}\%}$ to be noisy/corrupted versions of the adjacency matrices of ground truth networks G_{350} and G_{690} (see SM.4.2 for details). We set activating edges in $\boldsymbol{A}^{\boldsymbol{\sigma}\%}$ to +1 and repressive edges to -1. To validate explainability we extracted GRNs from trained models, and compared to ground truth G_{350} and G_{690} for existence of edges, out-degree correlations, and induced sparsity (see SM.2 for details).

5.2 Testing on experimental yeast cell cycle data

5.2.1 Data processing and normalization

GPR files were downloaded from the Gene Expression Omnibus (accession GSE4987 [45]), and consisted of **two dye-swap technical replicates** measured every five minutes for 120 minutes. Each of two replicates were separately ma-normalized using the `maNorm()` function in the `marray` library in `R/Bioconductor` [37]. The data were batch-corrected [38] using the `ComBat()` function in the `sva` library [39] and probe-sets mapping to the same gene were averaged, resulting in expression values for 5088 genes across fifty conditions. Two samples (corresponding to the 105 minute time point) were excluded for data-quality reasons, as noted in the original publication, and genes without motif information were then removed, resulting in a expression data-set containing 48 samples (24 time points in each replicate) and 3551 genes. The expression values were then normalized to be between 0 and 1.

5.2.2 Model setup for training and testing

Given the extreme similarity between the two replicates in terms of gene expression values, the approach of using one replicate for training and the other for validation would have led to artificially good performance on the validation trajectory. Instead, we noted that the data set contained 46 different transition pairs, where a transition pair consists of two consecutive expression vectors in the data set $(\boldsymbol{g}(t_i), \boldsymbol{g}(t_{i+1}))$. We split these 46 transition pairs into training (40, 86%), validation (3, 7%), and test (3, 7%). We provide some details on PHOENIX implementation (e.g. training strategy, prior incorporation, etc) in SM.1, and include finer technicalities (e.g. learning rate schedule, initialization scheme, etc) in our GitHub repository [34].

For prior domain knowledge model we used the simple linear model: $\mathcal{P}^*(\boldsymbol{\gamma_k}) = \boldsymbol{A} \cdot \boldsymbol{\gamma_k} - \boldsymbol{\gamma_k}$. We based our choice of $\boldsymbol{A}$ on the regulatory network structure of a motif map, similar to that used in other methods, such as PANDA [41]. We downloaded predicted binding sites for 204 yeast transcription factors [40]. These data include 4360 genes with tandem promoters. 3551 of these genes are also covered on the yeast cell cycle gene expression array. 105 total transcription factors in this data set target the promoter of one of these 3551 genes. The motif map between these 105 transcription factors and 3551 target genes provides the adjacency matrix $\boldsymbol{A}$ of 0s and 1s, where we randomly assigned each of the 1s to either +1 (activating) or -1 (repressive).

We used ChIP-chip data from Harbison et al. [40] to create a network of TF-target interactions, and used this as a validation network to test explainability. The targets of transcription factors in this ChIP-chip data set were filtered using the criterion $p < 0.001$. We calculated AUC values by comparing the encoded GRN retrieved from the trained models (see SM.2) to the validation network.

5.3 Testing on breast cancer pseudotime data

5.3.1 Data procurement and psuedotime ordering

The original data set comes from a cross-sectional breast cancer study (GEO accession GSE7390 [47]) consisting of microarray expression values for 22000 genes from 198 breast cancer patients, which we aimed to sort along a pseudotime axis. We noted that the same data set was also used in the PROB [19] paper. PROB is a GRN inference method that infers a random-walk based pseudotime to sort cross-sectional samples and reconstruct the GRN. For consistency and convenience in pseudotime inference, we obtained the same version of this data that was already preprocessed and sorted by PROB. We normalized the expression values to be between 0 and 1. We limited our analysis to the genes that had measurable expression and appeared in our prior model, and obtained a pseudotrajectory of expression values for 11165 genes across 186 patients. We also created pseudotrajectories for $n_g = 500$, 2000, and 4000 genes by subsetting to the n_g highest variance genes.

5.3.2 Model setup for training and testing

We noted that the data set contained 185 different transition pairs, where a transition pair consists of two consecutive expression vectors in the data set $(\boldsymbol{g}(t_i), \boldsymbol{g}(t_{i+1}))$. We split up the 185 transition pairs into training (170, 90%), validation (8, 5%), and test (7, 5%); Please find further implementation details in SM.1 and our GitHub repository [34].

For prior domain knowledge model we used the simple linear model: $\mathcal{P}^*(\boldsymbol{\gamma_k}) = \boldsymbol{W_0} \cdot \boldsymbol{\gamma_k} - \boldsymbol{\gamma_k}$. We based our choice of $\boldsymbol{W_0}$ on a motif map, similar to that used in the breast cancer analysis in OTTER [50]. The network $\boldsymbol{W_0}$ is derived from the human reference genome, for the breast tissue specifically. W_0 is a binary matrix with $\boldsymbol{W_0}_{i,j} \in \{0, 1\}$ where 1 indicates a TF sequence motif in the promoter of the target gene. Sequence motif mapping was performed using the `FIMO` software [65] from the `MEME` suite [66] and the `R` package `GenomicRanges` [67]. Note that $\boldsymbol{W_0}$ carries no sign information so that we cannot infer whether TFs inhibit or activate the expression of a gene. Hence we assigned each of the non-zero entries as +1 or -1 with uniform probability.

Validation of explainability was challenging, since there are only few data sets that have ChIP-seq data for many TFs from the same cells. We used ChIP-seq data from the MCF7 cell line (breast cancer, 62 TFs) in the ReMap2018 database [48] to create a validation network of TF-target interactions. We calculated AUC values by comparing the encoded GRNs retrieved from the trained models (see SM.2) to the validation network.

5.3.3 Gene influence scores

Given $\mathcal{M}_{n_g}$ a PHOENIX model trained on the pseudotrajectory consisting of only the n_g most variable genes ($n_g \in \{500, 2000, 4000, 11165\}$), we performed perturbation analyses to compute gene influence scores $\mathcal{IS}_{n_g,j}$. We randomly generated 200 initial (i.e. $t = 0$) expression vectors via i.i.d standard uniform sampling $\{\boldsymbol{g(0)_k} \in \mathbb{R}^{n_g}\}_{k=1}^{200}$. Next, for each gene j in $\mathcal{M}_{n_g}$, we created a perturbed version of these initial value vectors $\{\boldsymbol{g^j(0)_k}\}_{k=1}^{200}$, where only gene j was perturbed in each unperturbed vector of $\{\boldsymbol{g(0)_k}\}_{k=1}^{200}$. We then fed both sets of initial values into $\mathcal{M}_{n_g}$ to obtain two sets of predicted trajectories $\{\{\widehat{\boldsymbol{g}}\boldsymbol{(t)_k}\}_{t\in T} \in \mathbb{R}^{n_g}\}_{k=1}^{200}$ and $\{\{\widehat{\boldsymbol{g}}^{\boldsymbol{j}}\boldsymbol{(t)_k}\}_{t\in T} \in \mathbb{R}^{n_g}\}_{k=1}^{200}$ across a set of time points T. We calculated influence as the average absolute difference between the two sets of predictions, which represented how changes in *initial* ($t = 0$) expression of gene j affected *subsequent* ($t > 0$) predicted expression of all other genes in the n_g-dimensional system:

$$\mathcal{IS}_{n_g,j} = \frac{1}{200}\sum_{k=1}^{200}\left[\frac{1}{|T|}\sum_{\substack{t\in T\\ t\neq 0}}\left(\frac{1}{n_g}\sum_{\substack{i=1\\ i\neq j}}^{n_g}\left|\widehat{g}_i(t)_k - \widehat{g}_i{}^j(t)_k\right|\right)\right]$$

5.3.4 Pathway influence scores

Having computed gene influence scores $\mathcal{IS}_{n_g,j}$ for each gene j in each dynamical system of dimension n_g genes, we translated these gene influence scores into pathway influence scores. We used the Reactome pathway data set, GO biological process terms, and GO molecular function terms from MSigDB [49], that map each biological pathway/process, to the genes that are involved in it. For each system of size n_g, we obtained the pathway (p) influence scores ($\mathcal{PS}_{n_g,p}$) as the sum of the influence scores of all genes involved in that pathway:

$$\mathcal{PS}_{n_g,p} = \sum_{j\in p}\mathcal{IS}_{n_g,j}$$

We statistically tested whether each pathway influence score is higher than expected by chance using empirical null distributions. We randomly permuted the gene influence scores across the genes to recompute "null" values $\mathcal{PS}^0_{n_g,p}$. For each pathway, we performed $K = 1000$ permutations to obtain a null distribution $\{\mathcal{PS}^0_{n_g,p,k}\}_{k=1}^{K}$ that can be compared to $\mathcal{PS}_{n_g,p}$. We could then compute an empirical p-value as $p = \frac{1}{K}\sum_{k=1}^{K}\mathbb{I}_{\mathcal{PS}^0_{n_g,p,k} > \mathcal{PS}_{n_g,p}}$, where $\mathbb{I}$ is the indicator function. Finally, we used the mean ($\mu_{0(n_g,p)}$) and variance ($\sigma^2_{0(n_g,p)}$) of the null distribution $\{\mathcal{PS}^0_{n_g,p,k}\}_{k=1}^{K}$ to obtain and visualize pathway z-scores that are comparable across pathways and subset sizes (n_g):

$$z_{(n_g,p)} = \frac{\mathcal{PS}_{n_g,p} - \mu_{0(n_g,p)}}{\sqrt{\sigma^2_{0(n_g,p)}}}$$

Supplementary information. The article is accompanied by a Supplemental Methods (**SM**) section, and a Supplemental Results (**SR**) section. Additionally, all relevant code and data will be available as open source with the PHOENIX release: https://github.com/QuackenbushLab/phoenix.

Acknowledgments. The authors thank Marouen Ben Guebila, Dawn DeMeo, Jonas Fischer, Kimberly Glass, Camila Lopes-Ramos, Panagiotis Mandros, Soel Micheletti, Enakshi Saha, and Katherine Shutta for thoughtful critiques and discussions. We also thank Daniel Karlsson and Olle Svanström for sharing with us their starter code [68] for basic NeuralODE training.

Declarations. The authors declare the following:

- Funding : IH, VF, and JQ were supported by a grant from the US National Cancer Institute (R35CA220523). JQ and VF have additional funding from the National Human Genome Research Institute (R01HG011393).
- Conflict of interest : All authors declare no competing interests.
- Availability of data and materials : Relevant data are made available through the Supplemental Methods (**SM**) and Supplemental Results (**SR**) sections, as well as the PHOENIX release: https://github.com/QuackenbushLab/phoenix.
- Code availability : An open-source implementation [34] and documentation will be available through https://github.com/QuackenbushLab/phoenix.
- Authors' contributions : IH, JQ, and RB conceived of the project. RB proposed the NeuralODE modelling approach. IH proposed the physics/biology-informed learning framework. IH carried out the bulk of the work including data processing and coding of the methodology. VF contributed to the breast cancer pathways modeling and interpretation. All authors contributed to the writing and editing of the manuscript.

References

[1] Xing, J. (2022). Reconstructing data-driven governing equations for cell phenotypic transitions: integration of data science and systems biology. Physical Biology, 19(6), 061001.

[2] Hackett, S. R., Baltz, E. A., Coram, M., Wranik, B. J., Kim, G., Baker, A., ... & McIsaac, R. S. (2020). Learning causal networks using inducible transcription factors and transcriptome-wide time series. Molecular systems biology, 16(3), e9174.

[3] Qiu, X., Zhang, Y., Martin-Rufino, J. D., Weng, C., Hosseinzadeh, S., Yang, D., ... & Weissman, J. S. (2022). Mapping transcriptomic vector fields of single cells. Cell, 185(4), 690-711.

[4] Yeo, G. H. T., Saksena, S. D., & Gifford, D. K. (2021). Generative modeling of single-cell time series with PRESCIENT enables prediction of cell trajectories with interventions. Nature communications, 12(1), 1-12.

[5] Olteanu, M., & Stefan, R. (2020). An exponential stability test for a messenger rna–micro rna ode model. University politehnica of bucharest scientific bulletin-series a-applied mathematics and physics, 82(4), 11-16.

[6] Erbe, R., Stein-O'Brien, G., & Fertig, E. J. (2022). Transcriptomic forecasting with neural ODEs. bioRxiv.

[7] Chen, Z., King, W. C., Hwang, A., Gerstein, M., & Zhang, J. DeepVelo: Single-cell Transcriptomic Deep Velocity Field Learning with Neural Ordinary Differential Equations.

[8] Monti, M., Fiorentino, J., Milanetti, E., Gosti, G., & Tartaglia, G. G. (2022). Prediction of Time Series Gene Expression and Structural Analysis of Gene Regulatory Networks Using Recurrent Neural Networks. Entropy, 24(2), 141.

[9] Hu, Y. (2022). Modeling the gene regulatory dynamics in neural differentiation with single cell data using a machine learning approach.

[10] La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., ... & Kharchenko, P. V. (2018). RNA velocity of single cells. Nature, 560(7719), 494-498.

[11] Bergen, V., Lange, M., Peidli, S., Wolf, F. A., & Theis, F. J. (2020). Generalizing RNA velocity to transient cell states through dynamical modeling. Nature biotechnology, 38(12), 1408-1414.

[12] Gu, Y., Blaauw, D., & Welch, J. D. (2022). Bayesian inference of rna velocity from multi-lineage single-cell data. bioRxiv.

[13] Gayoso, A., Weiler, P., Lotfollahi, M., Klein, D., Hong, J., Streets, A. M., ... & Yosef, N. (2022). Deep generative modeling of transcriptional dynamics for RNA velocity analysis in single cells. bioRxiv.

[14] Cui, H., Maan, H., & Wang, B. (2022). DeepVelo: Deep Learning extends RNA velocity to multi-lineage systems with cell-specific kinetics. bioRxiv.

[15] Bergen, V., Soldatov, R. A., Kharchenko, P. V., & Theis, F. J. (2021). RNA velocity—current challenges and future perspectives. Molecular systems biology, 17(8), e10282.

[16] Aliee, H., Richter, T., Solonin, M., Ibarra, I., Theis, F., & Kilbertus, N. (2022). Sparsity in Continuous-Depth Neural Networks. arXiv preprint arXiv:2210.14672.

[17] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. Nature Reviews Physics, 3(6), 422-440.

[18] Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), 67-82.

[19] Sun, X., Zhang, J., & Nie, Q. (2021). Inferring latent temporal progression and regulatory networks from cross-sectional transcriptomic data of cancer samples. PLoS computational biology, 17(3), e1008379.

[20] Mao, G., Zeng, R., Peng, J., Zuo, K., Pang, Z., & Liu, J. (2022). Reconstructing gene regulatory networks of biological function using differential equations of multilayer perceptrons. BMC Bioinformatics, 23(1), 1-17.

[21] Alon, U. (2006). An introduction to systems biology: design principles of biological circuits. Chapman and Hall/CRC.

[22] Ghosh, S., Matsuoka, Y., Asai, Y., Hsin, K. Y., & Kitano, H. (2011). Software for systems biology: from tools to integrated platforms. Nature Reviews Genetics, 12(12), 821-832.

[23] Hossain, I. (2022). PHOENIX package [Computer software]. `https://github.com/QuackenbushLab/phoenix`

[24] Bhuva, D. D., Cursons, J., Smyth, G. K., & Davis, M. J. (2019). Differential co-expression-based detection of conditional relationships in transcriptional data: comparative analysis and

application to breast cancer. Genome biology, 20(1), 1-21

[25] Schaffter, T., Marbach, D., & Floreano, D. (2011). GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. Bioinformatics, 27(16), 2263-2270.

[26] Gesztelyi, R., Zsuga, J., Kemeny-Beke, A., Varga, B., Juhasz, B., & Tosaki, A. (2012). The Hill equation and the origin of quantitative pharmacology. Archive for history of exact sciences, 66(4), 427-438.

[27] Kraeutler, M. J., Soltis, A. R., & Saucerman, J. J. (2010). Modeling cardiac B-adrenergic signaling with normalized-Hill differential equations: comparison with a biochemical model. BMC systems biology, 4(1), 1-12.

[28] Farrell, S., Mani, M., & Goyal, S. (2022). Inferring single-cell dynamics with structured dynamical representations of RNA velocity. bioRxiv.

[29] Li, Q. (2022). scTour: a deep learning architecture for robust inference and accurate prediction of cellular dynamics. bioRxiv.

[30] Weinreb, C., Wolock, S., Tusi, B. K., Socolovsky, M., & Klein, A. M. (2018). Fundamental limits on dynamic inference from single-cell snapshots. Proceedings of the National Academy of Sciences, 115(10), E2467-E2476.

[31] Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. Advances in neural information processing systems, 31

[32] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[33] Chen, R. T. Q. (2021). torchdiffeq (Version 0.2.2) [Computer software]. `https://github.com/rtqichen/torchdiffeq`

[34] Hossain, I. (2022). PHOENIX package [Computer software]. `https://github.com/QuackenbushLab/phoenix`

[35] Van den Bulcke, T., Van Leemput, K., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., ... & Marchal, K. (2006). SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. BMC bioinformatics, 7(1), 1-12.

[36] Soetaert, K., Petzoldt, T., & Setzer, R. W. (2010). Solving differential equations in R: package deSolve. Journal of statistical software, 33, 1-25.

[37] Yang, Y. H., & Paquet, A. C. (2005). Preprocessing two-color spotted arrays. In Bioinformatics and Computational Biology Solutions Using R and Bioconductor (pp. 49-69). Springer, New York, NY.

[38] Johnson, W. E., Li, C., & Rabinovic, A. (2007). Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics, 8(1), 118-127.

[39] Leek, J. T., Johnson, W. E., Parker, H. S., Jaffe, A. E., & Storey, J. D. (2012). The sva package for removing batch effects and other unwanted variation in high-throughput experiments. Bioinformatics, 28(6), 882-883.

[40] Harbison, C. T., Gordon, D. B., Lee, T. I., Rinaldi, N. J., Macisaac, K. D., Danford, T. W., ... & Young, R. A. (2004). Transcriptional regulatory code of a eukaryotic genome. Nature, 431(7004), 99-104.

[41] Glass, K., Huttenhower, C., Quackenbush, J., & Yuan, G. C. (2013). Passing messages between biological networks to refine predicted interactions. PloS one, 8(5), e64832.

[42] Kuijjer, M. L., Tung, M. G., Yuan, G., Quackenbush, J., & Glass, K. (2019). Estimating sample-specific regulatory networks. Iscience, 14, 226-240.

[43] Aliee, H., Theis, F. J., & Kilbertus, N. (2021). Beyond Predictions in Neural ODEs: Identification and Interventions. arXiv preprint arXiv:2106.12430.

[44] Cheng, S., & Sabes, P. N. (2006). Modeling sensorimotor learning with linear dynamical systems. Neural computation, 18(4), 760-793.

[45] Pramila, T., Wu, W., Miles, S., Noble, W. S., & Breeden, L. L. (2006). The Forkhead transcription factor Hcm1 regulates chromosome segregation genes and fills the S-phase gap in the transcriptional circuitry of the cell cycle. Genes & development, 20(16), 2266-2278.

[46] Sirovich, L. (2020). A novel analysis of gene array data: yeast cell cycle. Biology Methods and Protocols, 5(1), bpaa018.

[47] Desmedt, C., Piette, F., Loi, S., Wang, Y., Lallemand, F., Haibe-Kains, B., ... & TRANSBIG Consortium. (2007). Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the TRANSBIG multicenter independent validation series. Clinical cancer research, 13(11), 3207-3214.

[48] Chèneby, J., Gheorghe, M., Artufel, M., Mathelier, A., & Ballester, B. (2018). ReMap 2018: an updated atlas of regulatory regions from an integrative analysis of DNA-binding ChIP-seq experiments. Nucleic acids research, 46(D1), D267-D275.

[49] Liberzon, A., Birger, C., Thorvaldsdóttir, H., Ghandi, M., Mesirov, J. P., & Tamayo, P. (2015). The molecular signatures database hallmark gene set collection. Cell systems, 1(6), 417-425.

[50] Weighill, D., Guebila, M. B., Lopes-Ramos, C., Glass, K., Quackenbush, J., Platig, J., & Burkholz, R. (2021, May). Gene regulatory network inference as relaxed graph matching. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 11, pp. 10263-10272).

[51] Kensler, K. H., Regan, M. M., Heng, Y. J., Baker, G. M., Pyle, M. E., Schnitt, S. J., ... & Tamimi, R. M. (2019). Prognostic and predictive value of androgen receptor expression in postmenopausal women with estrogen receptor-positive breast cancer: results from the Breast International Group Trial 1–98. Breast Cancer Research, 21(1), 1-11.

[52] Lu, X. F., Zeng, D., Liang, W. Q., Chen, C. F., Sun, S. M., & Lin, H. Y. (2018). FoxM1 is a promising candidate target in the treatment of breast cancer. Oncotarget, 9(1), 842.

[53] Artibani, M., Sims, A. H., Slight, J., Aitken, S., Thornburn, A., Muir, M., ... & Hohenstein, P. (2017). WT1 expression in breast cancer disrupts the epithelial/mesenchymal balance of tumour cells and correlates with the metabolic response to docetaxel. Scientific reports, 7(1), 1-15.

[54] Brett, J. O., Spring, L. M., Bardia, A., & Wander, S. A. (2021). ESR1 mutation as an emerging clinical biomarker in metastatic hormone receptor-positive breast cancer. Breast Cancer Research, 23(1), 1-15.

[55] Zhang, J., Li, G., Feng, L., Lu, H., & Wang, X. (2020). Krüppel-like factors in breast cancer: Function, regulation and clinical relevance. Biomedicine & Pharmacotherapy, 123, 109778.

[56] Hollern, D. P., Swiatnicki, M. R., Rennhack, J. P., Misek, S. A., Matson, B. C., McAuliff, A., ... & Andrechek, E. R. (2019). E2F1 drives breast cancer metastasis by regulating the target gene FGF13 and altering cell migration. Scientific reports, 9(1), 1-13.

[57] Wei, L. L., Wu, X. J., Gong, C. C., & Pei, D. S. (2017). Egr-1 suppresses breast cancer cells proliferation by arresting cell cycle progression via down-regulating CyclinDs. International Journal of Clinical and Experimental Pathology, 10(10), 10212.

[58] Gillespie, M., Jassal, B., Stephan, R., Milacic, M., Rothfels, K., Senff-Ribeiro, A., ... & D'Eustachio, P. (2022). The reactome pathway knowledgebase 2022. Nucleic acids research, 50(D1), D687-D692.

[59] Hanahan, D., & Weinberg, R. A. (2011). Hallmarks of cancer: the next generation. cell, 144(5), 646-674.

[60] Le Romancer, M., Poulard, C., Cohen, P., Sentis, S., Renoir, J. M., & Corbo, L. (2011). Cracking the estrogen receptor's posttranslational code in breast tumors. Endocrine reviews, 32(5), 597-622.

[61] Citro, A., Campo, F., Dugnani, E., & Piemonti, L. (2021). Innate immunity mediated inflammation and beta cell function: Neighbors or enemies?. Frontiers in Endocrinology, 1129.

[62] Okal, A., Matissek, K. J., Matissek, S. J., Price, R., Salama, M. E., Janát-Amsbury, M. M., & Lim, C. S. (2014). Re-engineered p53 activates apoptosis in vivo and causes primary tumor regression in a dominant negative breast cancer xenograft model. Gene therapy, 21(10), 903-912.

[63] Wang, X., & Yang, D. (2021). The regulation of RNA metabolism in hormone signaling and breast cancer. Molecular and cellular endocrinology, 529, 111221.

[64] Cleland, W. W. (1975). What limits the rate of an enzyme-catalyzed reaction. Accounts of Chemical Research, 8(5), 145-151.

[65] Grant, C. E., Bailey, T. L., & Noble, W. S. (2011). FIMO: scanning for occurrences of a given motif. Bioinformatics, 27(7), 1017-1018.

[66] Bailey, T. L., Boden, M., Buske, F. A., Frith, M., Grant, C. E., Clementi, L., ... & Noble, W. S. (2009). MEME SUITE: tools for motif discovery and searching. Nucleic acids research, 37(suppl_2), W202-W208.

[67] Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., ... & Carey, V. J. (2013). Software for computing and annotating genomic ranges. PLoS computational biology, 9(8), e1003118.

[68] Karlsson, D., & Svanström, O. (2019). Modelling Dynamical Systems Using Neural Ordinary Differential Equations. [master's thesis], Chalmers University of Technology

Supplemental results (SR)

SR.1 PHOENIX *in silico* predictive accuracy: unabridged version

We began our validation studies with simulated gene expression time-series data so that the underlying dynamical system that produced the system's patterns of gene expression was known. We adapted `SimulatorGRN` [23] (a simulator used extensively by the well cited `R`/Bioconductor package `dcanr` [24]) to generate time-series gene expression data; the strategy is similar to the approach used by other studies [25, 27]. We synthetically designed two gene regulatory networks, GRN_{350} and GRN_{690}, consisting of 350 and 690 genes respectively, where each regulatory edge connecting a transcription factor and its target gene was labelled as either activating or repressive. The simulator then used these regulatory network structures to create a "ground-truth" dynamical system of 350 (or 690) ODEs, and integrate them to simulate the temporal evolution of gene expression (see Methods 5.1.1 for details); we refer to these systems as SIM350 and SIM690.

We then used these ground truth ODEs to simulate time-series expression data for each gene across multiple time-points. By varying the initial gene expression levels (reflected in the initial conditions of the ODEs) we obtained gene expression "trajectories" for each gene in each model. To mimic experimental gene expression data, we added Gaussian noise to these trajectories to create *in silico* data sets. We used these data as input to PHOENIX, testing the trajectories predicted by the PHOENIX-estimated dynamical systems against the "known" trajectories to measure performance of the algorithm.

For each of the SIM350 and SIM690 systems, we simulated 160 noisy trajectories (with varying magnitudes of Gaussian noise) consisting of 5 time-points ($t = 0, 2, 3, 7, 9$) each, where every trajectory was characterized by a random vector of initial ($t = 0$) expression values. We used 140 of these trajectories for training, 10 for validation (and hyper parameter tuning), and 10 for testing. We introduced Gaussian noise to our trajectories with standard deviations of 0.01, 0.025, 0.05, and 0.1 which roughly corresponded to noise levels of 2%, 5%, 10% and 20% respectively, and investigated the performance of PHOENIX under increasingly noisy settings. Since PHOENIX uses user-defined prior knowledge as a regularizer, we added noise to the prior at a level commensurate with the "experimental" noise level (see SM.4.2), reflecting the fact that transcription factor-gene binding is itself noisy. Using the SIM350 and SIM690 training and validation sets, we trained and tuned PHOENIX on the simulated expression trajectories at each noise level. We measured the accuracy of PHOENIX in predicting expression values from held-out (test set) trajectories, since accurate prediction of unseen trajectories is suggestive of a coherence between the learned dynamical system and the ground truth.

For all noise settings, PHOENIX accurately learned the temporal evolution of the SIM350 and SIM690 data sets; the model explained most of the variance in gene expression across time (**Figure 2**) and was able to recover the *true* test set trajectories (that is, test set trajectories pre-noise) with a reasonably high accuracy (MSE $\approx 10^{-3}$) even when the training trajectories and prior knowledge model had included high levels of noise. Furthermore, the shapes of the predicted trajectories (and hence the predicted steady state levels) obtained from feeding initial values (expression at $t = 0$) into the trained model, remained robust to noise. These results also suggest that a trained PHOENIX model could be used to estimate the temporal effects of cellular perturbations as one could feed in *any* set of initial expression values and obtain predicted trajectories across arbitrary time points.

Since the primary prediction engine of PHOENIX is a NeuralODE, we wanted to benchmark its performance relative to "out-of-the-box" (OOTB) NeuralODE models on *in silico* data sets of varying noise levels, and understand how the various modifications introduced in developing PHOENIX contributed to its performance. We first explored how our choice of the Hill-like shifted softsign activation functions, and synergies of additive and multiplicative effects influenced performance, by comparing PHOENIX to "plain" NeuralODEs (such as RNAForecaster [6]). We tested the OOTB models with three of the most widely used activation functions, ReLU, sigmoid, and tanh, adjusted the total number of trainable parameters to be similar to that of PHOENIX (see SM.3.2). Because PHOENIX uses a domain prior of likely gene-regulation interactions in its optimization scheme, we set the weight of the prior to zero ($\lambda_{\text{prior}} = 0$). We repeated each analysis several times and measured performance variability based on how well the method was able to recover pre-noise test set trajectories.

For each of SIM350 and SIM690, we observed that OOTB NeuralODEs using either ReLU, sigmoid or tanh activation functions did not perform as well as PHOENIX on the test set in noiseless settings (**Figure SR1** and **Table SR2**). When we added noise, the PHOENIX models still generally outperformed the OOTB models, especially in the larger SIM690. The validation MSEs were more comparable between all the models in the high noise setting in SIM350. We found that the tanh models were consistently the best performing amongst the OOTB models in both SIM350 and SIM690. The consistently strong performance of PHOENIX suggests that using a Hill-kinetics inspired NeuralODE architecture better captures the dynamics of the regulatory process, in part because it models the binding kinetics of transcription factor-gene interactions.

Next, we tested the contribution of the prior to PHOENIX's performance. In most cases (except noise levels 5% and 10% in SIM690), we observed

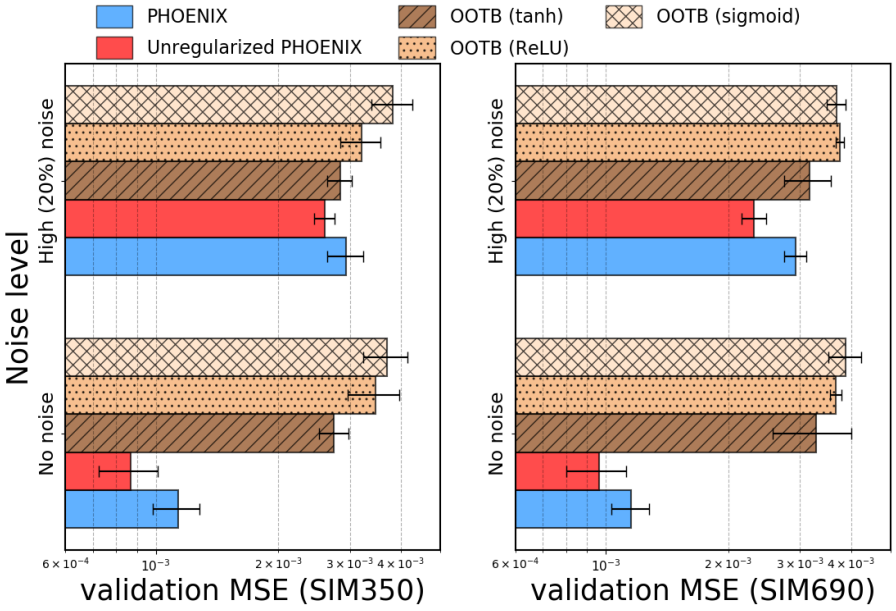

Fig. SR1 We measured the marginal contribution of PHOENIX's architecture and incorporation of prior information by comparing against the baseline contributions of out-of-the-box NeuralODE models with three different activation functions, across both *in silico* dynamical systems SIM350 (**left**) and SIM690 (**right**). We display the performance of the out-of-the-box models, as well as PHOENIX (λ_{prior} tuned using the validation set) and its unregularized version ($\lambda_{\text{prior}} = 0$), in terms of how well held out time points from pre-noise test set trajectories could be predicted after training on trajectories from different noise settings. Here high noise implies $\frac{\text{noise } \sigma}{\text{mean}} = 20\%$. The experiment was repeated five times to generate average mean-squared error (MSE) values and error bars.

that PHOENIX was outperformed in terms of temporal prediction by its unregularized ($\lambda_{\text{prior}} = 0$) version (**Figure SR1** and **Table SR2**). However, given that the prior can be interpreted as soft biological constraints on the estimated dynamical system [17], an important question is whether unregularized PHOENIX (as well as OOTB models) respect the underlying biology. In other words, we wanted to understand whether unregularized PHOENIX made accurate temporal predictions by correctly learning elements of the causal biology governing the dynamics, or whether the lack of prior information resulted in an alternate learned representation of the dynamics, which - despite predicting *these particular* held out trajectories accurately - was not reflective of the true biological regulatory process. Therefore, this "explainability" (or lack thereof) is what we investigated next.

SR.2 PHOENIX *in silico* explainability: unabridged version

While PHOENIX was able to accurately predict temporal gene expression patterns, we also found that the parameters of a trained PHOENIX model encoded an estimate of the ground-truth gene regulatory networks (GRNs) that causally governed the system's evolution over time. To assess whether PHOENIX was predicting held-out expression values in an explainable way, we infered encoded GRNs from trained PHOENIX models and compared the inferred GRNs to the ground truth GRNs used to generate the *in silico* data.

While there are a number of methods to infer encoded GRNs from dynamics estimators, most rely on computationally expensive sensitivity analyses [2]. Given PHOENIX's simple NeuralODE architecture, we were able to develop an inference algorithm that only relied on the coefficients of the trained model without any need for sensitivity analyses. Roughly speaking, we predicted the existence of a directed edge from gene A to gene B ($A \xrightarrow{+} B$) if the coefficients describing the effect of A on B in the trained PHOENIX model were sufficiently large enough. We assign edge-signs (indicating either activating+ vs repressive− interactions) based on the signs of the coefficients (see SM.2). To this end, we took trained PHOENIX and unregularized PHOENIX models from both SIM350 and SIM690 under each of the different noise settings and inferred the corresponding GRNs. For comparison, we wanted to compare with GRNs inferred from the best performing (in terms of temporal prediction) OOTB models as well; given the black-box nature of the OOTB models these GRNs had to be obtained using a time consuming sensitivity analysis approach (Methods SM.3.2).

We compared the inferred GRNs to the ground truth GRNs in terms of several metrics, including the existence of edges, out-degree correlations, and induced sparsity. We obtained near-perfect AUC values for PHOENIX in the 0.96-0.99 range as well as high out-degree correlations, all of which remained robust to high noise levels (**Figure 3** and **Table SR3**). Most notably, we observed that PHOENIX predicted dynamics in a more explainable way than its unregularized version, and heavily outperformed the OOTB models. While PHOENIX remained robustly explainable to noise, both unregularized PHOENIX and the OOTB models became less-and-less explainable in terms of all metrics of interest. Since the ground truth GRNs were relatively sparse (GRN_{350} average degree = 1.57, GRN_{690} average degree = 1.56, reflecting the fact that there are less than approximately 200 transcription factors among the 6000 genes in the yeast genome) it was important that the inferred GRN was sparse. We measured induced sparsity by reverse engineering a metric $\mathcal{C}_{\max}$ based on maximizing classification accuracy (see SM.2), and found that PHOENIX resulted in much sparser inferred GRNs than its unregularized version (**Table SR4**). To further assess this phenomenon, we computed the

estimated model effect for every gene pair in SIM350, and compared these values between PHOENIX and unregularized PHOENIX across all noise settings (**Figure SR2**). These effects indicate that the incorporation of the priors helped PHOENIX identify core elements of the dynamics, resulting in the prediction of temporal gene expression in a biologically meaningful and explainable manner. We believe that this behavior will allow PHOENIX to identify key drivers of gene regulation, and in the process make more accurate predictions in response to genomic perturbations.

It should be noted that the means of including the gene-regulation domain knowledge into PHOENIX is extremely flexible. The prior model is integrated into the optimization steps of PHOENIX through a modified loss function using randomly generated expression values (see 2.3). Hence, users can not only tune how much weight is given to prior information (λ_{prior}), but can also incorporate domain knowledge through *any* prior model of choice. Since the prior models essentially induce soft biological constraints on the inferred dynamics [17], our framework can be used to subject the dynamics to context-specific constraints (for example, that the effect of transcription factors binding to their targets should reach saturation at high concentrations [1, 64]). In our experiments, we see significant gains in explainability using just a simple adjacency-matrix-based prior model derived from prior knowledge of the underlying GRN.

Given that the inclusion of such static prior knowledge greatly increased the explainability of the inferred dynamics, we also investigated how explainability was affected by **misspecification** of the prior. In our *in silico* experiments, we had randomly corrupted (misspecified) the prior by an amount commensurate with the noise level (see SM.4.2). So we compared network representations of these misspecified prior constraints to GRNs extracted from the PHOENIX models that used these very priors (**Table SR1**). We found that PHOENIX was able to appropriately learn causal elements of the dynamics beyond what was encoded in the priors. This suggests that even though PHOENIX uses user-defined priors to enhance explainability, it can deviate from the prior model when necessary, and learn biologically meaningful dynamics from just the data itself. This behavior is observed in real data applications as well, where PHOENIX learned regulatory interactions encoded in the motif-prior models, and also learned interactions that were not described by the prior, but were nevertheless experimentally validated regulatory links (**Table SR1**).

Table SR1 Assessing the contribution of priors to PHOENIX explainability, as well as the effect of prior misspecification, in both *in silico* experiments and real experimental data

	Genes	Noise	PHOENIX	Prior constraints
SIM350[1]	350	0%	0.998	1.000
	350	5%	0.992	0.970
	350	10%	0.979	0.950
	350	20%	0.962	0.900
SIM690[1]	690	0%	0.998	1.000
	690	5%	0.996	0.975
	690	10%	0.991	0.950
	690	20%	0.958	0.900
Yeast[2]	3551	-	0.860	0.790
Breast[2]	500	-	0.910	0.820
	2000	-	0.910	0.840
	4000	-	0.873	0.810
	11165	-	0.810	0.810

[1]For *in silico* experiments (SIM350, SIM690) the prior knowledge model was corrupted by an amount commensurate with the noise level (see SM.4.2). Then for each scenario, a network representation of the misspecified prior model was checked for how well it aligned with the ground truth GRN in terms of AUC. This AUC was compared to that obtained by aligning the ground truth GRN against a network extracted from a PHOENIX model trained using the misspecified prior in question.

[2]The same approach was repeated for the real data sets (yeast cell cycle and breast cancer), but this time the prior networks came from the motiff-prior models used (see Methods 5.2.2 and 5.3.2), and the "ground-truth" GRNs were experimentally verified ChIP-Seq networks describing transcription factor binding [40].

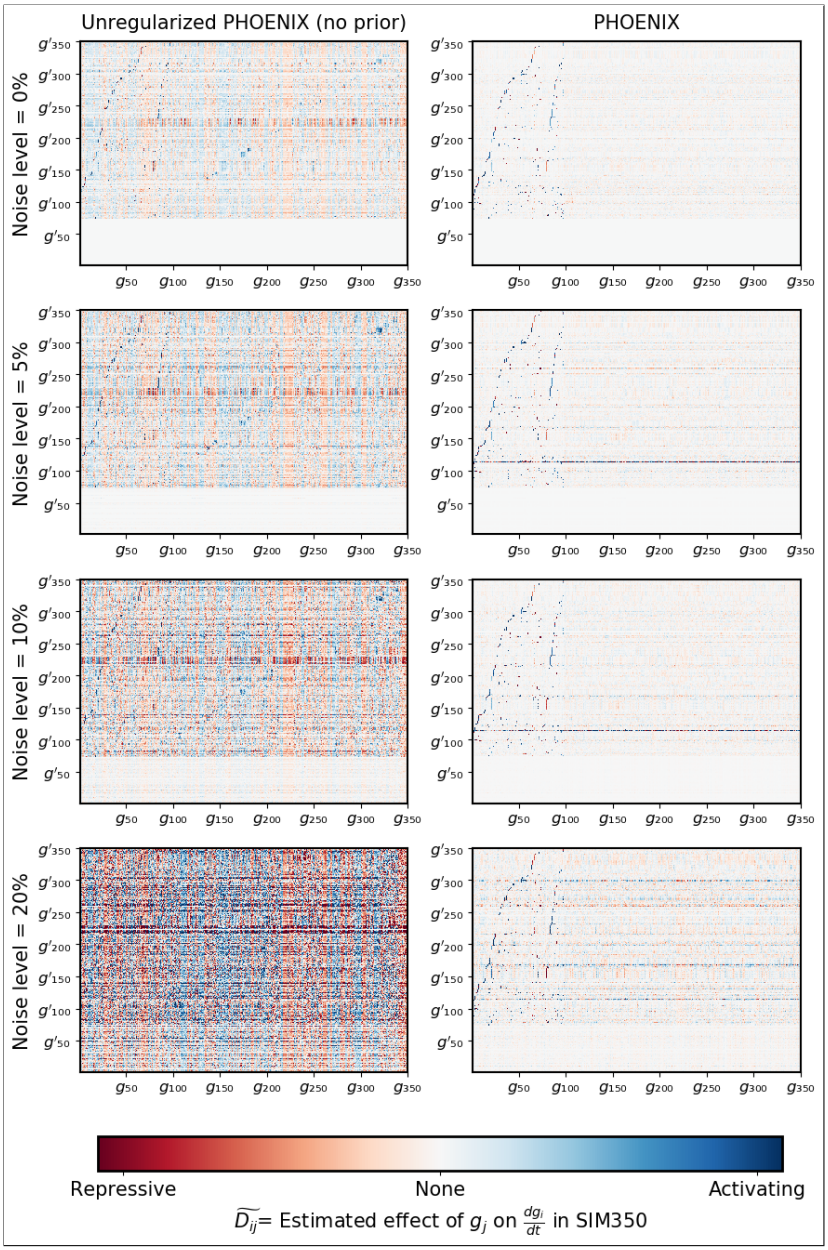

Fig. SR2 We investigated *how* PHOENIX used prior information to sparsify the learned dynamics by computing the encoded 350×350 dynamics matrices $\widetilde{D}$ (see SM.2), across all noise settings in SIM350. We display $\widetilde{D}$ for both PHOENIX (**right column**) as well as its unregularized ($\lambda_{\text{prior}} = 0$) version (**left column**) as heatmaps of effect size.

SR.3 Comparing PHOENIX to black-box models: additional results

Table SR2 Benchmarking PHOENIX against other methods *in silico*, in terms of trajectory-recovery performance (performance metric is MSE on test set)

		Trajectory based				Velocity based		
	Noise	PHX[1]	PHX_0[2]	OOTB[3]	PRESC[4]	Dynamo[5]	RNODE[6]	scDVF[7]
SIM350	0%	0.0011	**0.0009**	0.0028	0.0036	0.0029	0.0068	0.0096
	5%	0.0012	**0.0009**	0.0026	0.0039	0.0039	0.0074	0.0103
	10%	0.0019	**0.0016**	0.0027	0.0039	0.0043	0.0073	0.0094
	20%	0.0029	**0.0026**	0.0029	0.0039	0.0048	0.0075	0.0099
SIM690	0%	0.0011	**0.0009**	0.0026	0.0038	0.0052	0.0072	0.0073
	5%	**0.0012**	0.0018	0.0023	0.0037	0.0060	0.0064	0.0069
	10%	**0.0015**	0.0026	0.0039	0.0037	0.0060	0.0076	0.0095
	20%	0.0030	**0.0024**	0.0028	0.0039	0.0059	0.0075	0.0084

[1]PHX = Regularized PHOENIX where λ_{prior} was chosen based on the validation set.

[2]PHX_0 = Unregularized PHOENIX where $\lambda_{\text{prior}} = 0$ (ignoring the network prior).

[3]OOTB = Out-of-the-box NeuralODE, resembling how plain NeuralODEs are typically used for this problem [6, 9]. We note that the method RNAForecaster [6] uses an OOTB approach with ReLU activation, but the results here are for the *tanh* activation function which had better performance than both sigmoid and ReLU on the validation set.

[4]PRESC = PRESCIENT [4], where we used the validation set to optimize the value of k_{dim}, which is the number of neurons use in PRESCIENT's hidden layer.

[5]For Dynamo [3] we used the validation set to optimize the sparsity regularization penalty (λ), as well as the number of kernel basis functions use to approximate the vector field (M)

[6]RNODE = RNA-ODE [5], where we used the validation set to optimize the number of trees used in the random forest function

[7]In scDVF [7] the encoder and the decoder consisted of four dense layers (size 64 for the intermediate layers and size 16 for the latent layer) with ReLU activation

Table SR3 Benchmarking PHOENIX against other methods *in silico*, in terms of explainability. A GRN describing the inferred dynamics was extracted for each fitted model, and was compared to the ground truth GRN to calculate an AUC

		Trajectory based				Velocity based		
	Noise	PHX[1]	PHX_0[1]	OOTB[2]	PRESC[3]	Dynamo[4]	RNODE[5]	scDVF[6]
SIM350	0%	**0.998**	0.909	0.610	N/A	0.778	0.721	0.787
	5%	**0.992**	0.854	0.649	N/A	0.778	0.599	0.723
	10%	**0.979**	0.812	0.600	N/A	0.775	0.594	0.618
	20%	**0.962**	0.720	0.550	N/A	0.742	0.535	0.562
SIM690	0%	**0.998**	0.858	0.720	N/A	0.686	0.701	0.810
	5%	**0.996**	0.843	0.688	N/A	0.699	0.604	0.749
	10%	**0.991**	0.800	0.549	N/A	0.692	0.535	0.665
	20%	**0.958**	0.659	0.566	N/A	0.673	0.488	0.550

[1]The architecture of PHOENIX allowed GRNs to be extracted from both regularized and unregularized versions using a very simple and efficient algorithm (see SM.2)

[2]GRNs were extracted from out-of-the-box NeuralODEs, using sensitivity analyses (see SM.3.2)

[3]PRESCIENT does not provide a straightforward means for extracting a GRN describing the dynamics [4]

[4]Dynamo provides a function that calculates the Jacobian matrix of the fitted model at any data point [3]. We used this function to calculate an average Jacobian matrix across several randomly generated data points and estimate a GRN (see SM.3.3)

[5]RNA-ODE provides a function for GRN inference by ranking regulatory links using estimated effect sizes [5] (see SM.3.3)

[6]scDVF provides a function for estimating gene correlation networks from simulated retrograde trajectories [7]. We generated $n = 200$ retrograde trajectories using 200 randomly generated initial conditions (see SM.3.3)

Table SR4 Benchmarking PHOENIX against other methods *in silico*, in terms of sparsity of inferred dynamics. Sparsity was calculated here as the average out-degree of the extracted GRN from each of the inferred dynamical systems

		Trajectory based				Velocity based		
	Noise	PHX	PHX_0	OOTB	PRESC[1]	Dynamo	RNA-ODE	scDVF
SIM350	0%	**8**	18	75	N/A	147	82	46
	5%	**13**	18	69	N/A	147	64	43
	10%	**9**	25	41	N/A	141	41	37
	20%	**8**	63	85	N/A	168	23	99
SIM690	0%	**9**	36	144	N/A	424	206	58
	5%	**14**	33	109	N/A	347	39	90
	10%	**14**	74	89	N/A	336	71	102
	20%	**20**	108	138	N/A	464	45	178

[1]PRESCIENT [4] does not provide a straightforward means for extracting a GRN that describes the dynamics

Table SR5 Details about additional black-box methods for estimating gene expression dynamics that we excluded from our *in silico* benchmarking experiments. We provide details about each method, and some reasoning behind its exclusion

Method	Approach for estimating dynamics	Notes on exclusion
PROB[19]	Uses a local Euler approximation to calculate RNA velocity $d\boldsymbol{x}/dt$. Then describes this RNA velocity as a function of gene-expression through a linear model with quadratic interaction terms between all pairs of genes. Bayesian Lasso is used to induce sparsity.	The linear model is too simplistic, and may not accurately reflect complex regulatory patterns. Also, the method seeks to directly estimate pairwise gene effects ($\mathcal{O}(n^2)$ parameters), and hence has not been shown to scale beyond 100 genes.
LatentVelo[28]	Embeds spliced and unspliced RNA counts into a low dimensional latent space using a variational autoencoder (VAE), and then infers dynamics in this latent space with a NeuralODE, with soft constraints on the interplay between latent spliced and latent unspliced counts.	Dynamics can only be inferred in a low-dimensional latent space, making it difficult to derive interpretable insights and compare explainability. We already benchmarked against a VAE-based model (scDVF [7]) which - unlike LatentVelo - provides a GRN interpreting the inferred dynamics.
scTour[29]	Embeds spliced RNA counts into a low dimensional latent space using a variational autoencodeer, and then infers dynamics in this latent space with a NeuralODE.	Same reasoning as above.
PBA[30]	Uses spectral-graph theory to solve multi-dimensional Fokker-Plank equations on an empirical grid formed by expression values. It assumes that velocity fields are gradients of a potential landscape in gene expression space $\boldsymbol{J} = -\nabla F$.	PBA ostensibly ignores oscillatory gene expression dynamics (the cell cycle) [30], and hence may not be flexible enough. Also, we already benchmarked against another method (PRESCIENT [4]), which assumes dynamics to be the gradient of a scalar potential.

SR.4 Applying PHOENIX to breast cancer data: additional results

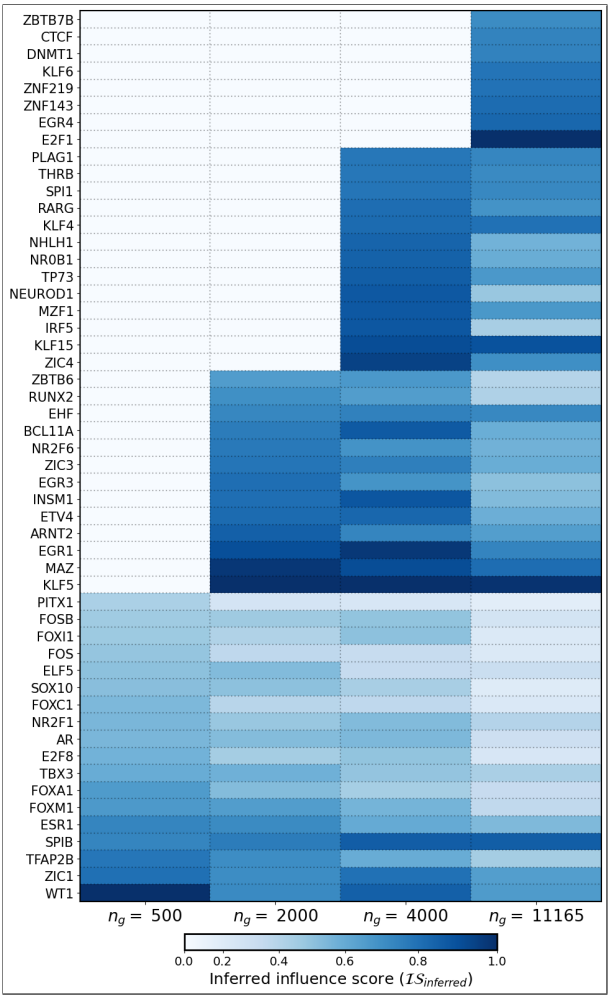

Fig. SR3 We applied PHOENIX to a pseudotrajectory of 186 breast cancer samples (ordered along subsequent "pseudotimepoints") consisting of $n_g = 11165$ genes [47]. We also repeated the analysis on smaller subsets of genes $n_g = 500, 2000, 4000$, where we subsetted the full trajectory to only the n_g most variable genes in the pseudotrajectory. We used the trained PHOENIX models to extract influence scores for individual genes in the estimated system (see Methods 5.3.3), and visualized influence scores for the most central genes across different values of n_g. For visualization purposes, the influence scores are normalized within each column (i.e. each value of n_g) to be between 0 and 1. Genes that are excluded from the subset of n_g most variable genes were assigned $\mathcal{IS}_{inferred} = 0$.

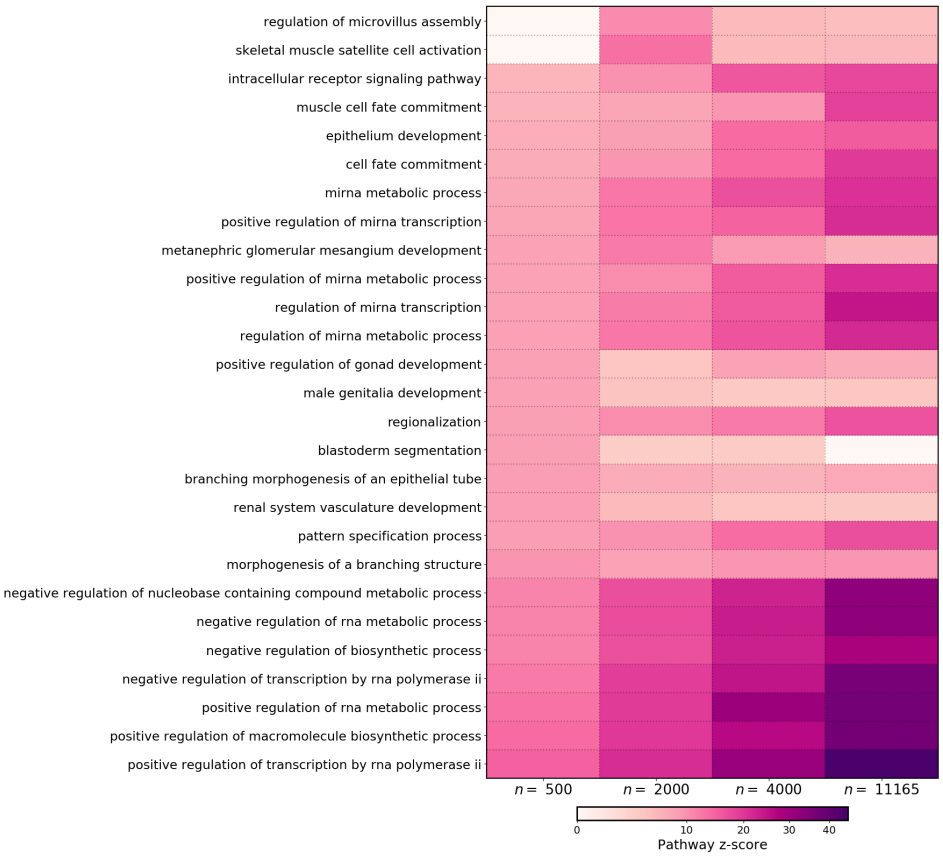

Fig. SR4 We applied PHOENIX to a pseudotrajectory of 186 breast cancer samples (ordered along subsequent "pseudotimepoints") consisting of $n_g = 11165$ genes [47]. We also repeated the analysis on smaller subsets of genes $n_g = 500, 2000, 4000$, where we subsetted the full trajectory to only the n_g most variable genes in the pseudotrajectory. We used the trained PHOENIX models to extract influence scores for pathways in the **Gene Ontology (biological process)** database (see Methods 5.3.4), and visualized influence scores for the most central pathways across different values of n_g.

Fig. SR5 We applied PHOENIX to a pseudotrajectory of 186 breast cancer samples (ordered along subsequent "pseudotimepoints") consisting of $n_g = 11165$ genes [47]. We also repeated the analysis on smaller subsets of genes $n_g = 500, 2000, 4000$, where we subsetted the full trajectory to only the n_g most variable genes in the pseudotrajectory. We used the trained PHOENIX models to extract influence scores for pathways in the **Gene Ontology (molecular function)** database (see Methods 5.3.4), and visualized influence scores for the most central pathways across different values of n_g.

Table SR6 Detailed results of permutation tests (see Methods 5.3.4) used to obtain pathway influence scores from trained PHOENIX models and the **Reactome pathway database** [58]. The mean (μ_0) and standard deviation (σ_0) of each permutation test null distribution are tabulated for each subset (n_g) in question, and the corresponding z-scores are visuallized in **Figure 5**. Missing values indicate that the **all** genes involved in that pathway were excluded from that particular subset based on low variability in expression.

Reactome pathway	$n_g = 500$ genes			$n_g = 2000$ genes			$n_g = 4000$ genes			$n_g = 11165$ genes		
	z	μ_0	σ_0	z	μ_0	σ_0	z	μ_0	σ_0	z	μ_0	σ_0
nuclear receptor transcription pathway	8.254	0.014	0.013	14.783	0.043	0.022	22.331	0.111	0.027	29.595	0.064	0.006
estrogen dependent gene expression	7.28	0.035	0.019	5.638	0.063	0.026	16.879	0.097	0.025	23.653	0.102	0.007
sumoylation of intracellular receptors	6.902	0.009	0.01	11.689	0.019	0.014	15.455	0.059	0.019	18.487	0.036	0.004
runx1 regulates wnt signaling	6.414	0.003	0.005	6.315	0.003	0.006	5.329	0.004	0.005	5.367	0.005	0.001
esr mediated signaling	5.987	0.049	0.022	3.764	0.108	0.034	8.921	0.204	0.036	14.155	0.179	0.01
runx1 reg. estrogen receptor mediated transcription	5.713	0.003	0.006	7.822	0.003	0.005	4.372	0.005	0.006	6.701	0.004	0.001
transcriptional regulation of testis differentiation	5.428	0.005	0.008	3.721	0.016	0.012	7.757	0.039	0.016	6.834	0.012	0.003
regulation of runx2 expression and activity	5.424	0.006	0.008	5.419	0.024	0.016	5.751	0.056	0.019	3.65	0.083	0.006
transcriptional regulation by runx2	5.243	0.012	0.012	7.127	0.055	0.023	8.925	0.119	0.028	7.146	0.134	0.008
sumoylation of transcription factors	5.146	0.003	0.006	5.579	0.01	0.011	3.518	0.026	0.014	8.458	0.022	0.003
nuclear signaling by erbb4	4.447	0.009	0.01	3.468	0.022	0.015	1.314	0.065	0.021	1.847	0.035	0.004
runx2 regulates bone development	4.225	0.003	0.006	7.511	0.021	0.015	8.74	0.044	0.017	10.138	0.033	0.004
ngf stimulated transcription	3.829	0.009	0.01	6.586	0.044	0.022	6.849	0.065	0.02	11.538	0.046	0.005
transcriptional regulation of granulopoiesis	3.174	0.003	0.005	5.275	0.02	0.015	10.197	0.052	0.018	15.124	0.036	0.004
regulation of gene expression in beta cells	2.262	0.003	0.006	2.353	0.003	0.005	10.856	0.025	0.011	10.654	0.016	0.002
estrogen dep. nucl. events (downstr. esr memb. sig.)	1.52	0.011	0.011	0.724	0.023	0.015	0.171	0.039	0.017	18.257	0.028	0.004
transcriptional regulation of pluripotent stem cells	-0.227	0.003	0.007	5.464	0.016	0.014	9.772	0.047	0.017	12.611	0.027	0.003
regulation of apoptosis	-0.444	0.011	0.011	-0.663	0.038	0.019	1.297	0.124	0.027	18.031	0.193	0.009
runx2 regulates chondrocyte maturation				7.926	0.007	0.009	6.916	0.009	0.007	4.934	0.005	0.002
aryl hydrocarbon receptor signalling				7.177	0.003	0.006	6.718	0.008	0.007	4.389	0.008	0.002
deact. of the beta catenin transactivating complex				6.669	0.01	0.01	5.831	0.035	0.016	4.578	0.046	0.005
runx1 regulates differentiation of myeloid cells				6.282	0.003	0.006	3.885	0.008	0.007	1.939	0.006	0.002
myogenesis				-0.259	0.007	0.009	1.457	0.026	0.013	10.692	0.03	0.004
mecp2 regulates transcription factors							9.653	0.008	0.007	9.156	0.005	0.002
tp53 reg. transcrip. of caspase activators and caspases							8.645	0.008	0.007	5.323	0.014	0.002
activation of puma and translocation to mitochondria							7.925	0.009	0.008	11.346	0.011	0.002

Table SR7 Additional performance metrics for PHOENIX on breast cancer data

Number of genes (n_g)[1]	Runtime (hrs)[2]	Concordance with influential ChIP genes[3]
500	0.10	11.47%
2000	0.14	18.03%
4000	0.41	31.14%
11165	2.39	85.25%

[1]The full data set consisted of $n_g = 11165$ genes [47]. We also fit PHOENIX to smaller subsets of genes $n_g = 500, 2000, 4000$, where we subsetted the full data set to only the n_g most variable genes in the pseudotrajectory.

[2]Runtime for a single run when using an AWS c5.4xlarge instance ($0.68/hour).

[3]For each n_g, we computed inferred influence scores for individual genes based on perturbation analyses on the fitted PHOENIX model (see Methods 5.3.3). We also computed harmonic centralities of the genes based on a validation network describing ChIP-binding data [48]. We then did a binary assignment, where the top 10% ($10\% \times n_g$) genes with highest inferred influence were labelled "predicted influential" and the remaining ($90\% \times n_g$) were "predicted non-influential". We measured concordance as the **sensitivity** $\frac{TP}{TP+FN}$ with which these predicted labels recovered the "truly influential" genes (i.e. those genes with non-zero harmonic centrality in the ChIP validation network).

Supplemental methods (SM)

SM.1 PyTorch implementation: details

SM.1.1 NN architecture and forward function

Both NN_{sums} and NN_{prods} are fully connected single layer NNs with Hill-like activations that take n inputs (for n genes) and produce m outputs each.

```
(NN_sums): Sequential(
    (activation_0): phi_Sigma()
    (sum_combos): Linear(in_features=n,out_features=m, bias = True))

(NN_prods): Sequential(
    (activation_0): phi_Pi()
    (prod_combos): Linear(in_features=n,out_features=m, bias = True))
```

We use $m << n$ allowing us to approximate lower dimensional sets of additive and multiplicative gene combinations. We used the following size m in each experiment, where m roughly scaled with the dimensionality n of the problem:

- Simulations: For ($n = 350, 690$ genes) we use $m = 40, 50$, respectively
- Yeast cell cycle: There were $n = 3551$ genes, and we used $m = 120$
- Breast cancer: For subset sizes $n_g = 500, 2000, 4000, 11165$ genes we used $m = 40, 100, 120, 300$ respectively

These $2m$ outputs are then fed into a final fully connected single layer $\text{NN}_{combine}$, and gene-specific parameters ($\boldsymbol{\upsilon}$/ `u_vector` $\in \mathbb{R}^n$) are applied to create the `final` estimations for the n local derivatives.

```
(NN_combine): Sequential(
    (final_outputs): Linear(in_features=2*m,out_features=n))

def forward(self, t, input):
    c_Sigma = self.NN_sums(input)
    c_Pi = torch.exp(self.NN_prods(input))
    joint = self.NN_combine(torch.cat((c_Sigma, c_Pi),dim=-1)
    final = torch.relu(u_vector)*(joint - y)
    return(final)
```

SM.1.2 Predictive performance: training, testing, and validation (choosing $\boldsymbol{\lambda}$)

We re-shaped the simulated data sets to be amenable with the `torchdiffeq` package in `PyTorch` [33], which contains the base library used for NeuralODEs. Regardless of whether the data was split based on trajectories (*in silico* experiments) or transition pairs (real data applications), *at any given training step* the data was fed into the PHOENIX model in pairwise-vector form. For instance, in the *in silico* experiments, each training trajectory consisted of 5 time-points $\in \{0, 2, 3, 7, 9\}$, and was subsequently fed to the model in the form of its 4 constituent transition pairs $(t_i, t_{i+1}) \in \{(0, 2), (2, 3), (3, 7), (7, 9)\}$, where a transition pair consists of two consecutive expression vectors in the trajectory $(\boldsymbol{g}(t_i),\ \boldsymbol{g}(t_{i+1}))$. With each transition pair fed to the model, it learned an approximation of the derivative that described the transition of $\boldsymbol{g}(t_i)$ to $\boldsymbol{g}(t_{i+1})$. We initialized each of NN_{sums}, NN_{prods}, and $\text{NN}_{combine}$ with a sparse initialization scheme that set 95% of the weight parameters to be 0. We initialized the $\boldsymbol{\upsilon}_i$s with i.i.d standard uniform values.

We used Dopri5 (included with `torchdiffeq`) as the ODESolver within PHOENIX's NeuralODE engine, and the Adam optimizer (also included with `torchdiffeq`) to optimize NN parameters. Broadly speaking, `torchdiffeq` learns by performing back propagation through the ODESolver via adjoint sensitivity analysis [31]. Next, we use a `simulated_batch` of expression values $\{\boldsymbol{\gamma_k}\}_{k=1}^{K}$ to incorporate a structural domain knowledge inspired prior model

$$\mathcal{P}^*(\boldsymbol{\gamma_k}) + \boldsymbol{\gamma_k} = \boldsymbol{A} \cdot \boldsymbol{\gamma_k}$$

So, given the user-supplied `prior_matrix` $\boldsymbol{A}$, we pre-calculated `prior_output`, $\mathcal{P}^*(\boldsymbol{\gamma_k}) + \boldsymbol{\gamma_k}$, using `torch.matmul()`.

```
simulated_batch = torch.rand(10000,1,prior_mat.shape[0])
prior_output = torch.matmul(batch_for_prior,prior_mat)
```

We tied this into model training using a modified loss function `composed_loss` with weight (λ). We obtained model `predictions` using the `odeint()` function in `torchdiffeq`, as shown in the pseudocode below:

```
def training_step(...., training_batch, target, simulated_batch,
                    prior_output, lambda, time_pts):
    predictions =  odeint(....,training_batch, time_pts)
    loss_data = torch.mean((predictions - target)**2)
    model_sim_output = forward(time_pts,simulated_batch) +
                                                simulated_batch
    loss_prior = torch.mean((model_sim_output - prior_output)**2)
    composed_loss = lambda * loss_data +
                    (1- lambda) * loss_prior
    composed_loss.backward()
    opt.step()
    return [loss_data, loss_prior]
```

We trained for up to 200 epochs on an AWS c5.9xlarge instance, where each epoch consisted of the entire training set being fed to the model in the form of constituent transition pairs. We used `torch.optim` function `ReduceLROnPlateau()` to reduce the learning rate by 10% every 3 epochs, unless the validation set performance showed reasonable improvement.

```
scheduler = optim.lr_scheduler.ReduceLROnPlateau(opt, mode='min',
            factor=0.9, patience=3, threshold=1e-05,
            threshold_mode='abs', eps=1e-09)
```

Training terminated if validation set performance failed to improve in 10 consecutive epochs. We repeated this entire pipeline for a grid of $\lambda \in \{0.1, 0.2, 0.5, 0.8, 0.9, 0.99, 0.999, 1\}$, and used the validation mean squared error (MSE) at model termination to decide on an optimal λ. For this final model, we evaluated final predictive performance using test set MSE.

For even further details (exact learning rates, `torchdiffeq` details, ODE-Solver details, optimizer details, etc.) we refer the reader to our GitHub repository [34], where the entire code base will be made available.

SM.2 Explainability performance: GRN inference

SM.2.1 Algorithm for efficiently retrieving encoded GRN from trained PHOENIX model

We start with PHOENIX's prediction for the local derivative given a gene expression vector $\boldsymbol{g}(t) \in \mathbb{R}^n$ in an n-gene system:

$$\widehat{\frac{d\boldsymbol{g}(t)}{dt}} = \mathrm{ReLU}(\boldsymbol{v}) \odot \Big[\boldsymbol{W}_{\cup}\{\boldsymbol{c}_{\boldsymbol{\Sigma}}(\boldsymbol{g}(t)) \oplus \boldsymbol{c}_{\boldsymbol{\Pi}}(\boldsymbol{g}(t))\} - \boldsymbol{g}(t)\Big], \quad \text{where}$$

$$\boldsymbol{c}_{\boldsymbol{\Sigma}}(\boldsymbol{g}(t)) = \boldsymbol{W}_{\boldsymbol{\Sigma}}\phi_{\Sigma}(\boldsymbol{g}(t)) + \boldsymbol{b}_{\boldsymbol{\Sigma}} \quad \text{and} \quad \boldsymbol{c}_{\boldsymbol{\Pi}}(\boldsymbol{g}(t)) = \exp\circ(\boldsymbol{W}_{\boldsymbol{\Pi}}\phi_{\Pi}(\boldsymbol{g}(t)) + \boldsymbol{b}_{\boldsymbol{\Pi}})$$

We observed that a trained PHOENIX model encodes interactions *between* genes primarily within the weight parameters from its neural network blocks $\boldsymbol{W}_{\boldsymbol{\Pi}}, \boldsymbol{W}_{\boldsymbol{\Sigma}} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{W}_{\cup} \in \mathbb{R}^{n\times 2m}$. This inspired an efficient means of projecting the estimated dynamical system down to a gene regulatory network (GRN) $\widehat{G_n}$. We calculated a matrix $\boldsymbol{D} \in \mathbb{R}^{n\times n}$, where $\boldsymbol{D}_{ij}$ approximated the *absolute contribution* of gene j to the derivative of gene i's expression:

$$\boldsymbol{D} = \boldsymbol{W}_{\cup}\begin{bmatrix}\boldsymbol{W}_{\boldsymbol{\Sigma}}\\ \boldsymbol{W}_{\boldsymbol{\Pi}}\end{bmatrix}$$

Next, we adapted the marginal attribution approach described by Hackett *et al.* [2] to calculate the **dynamics matrix** $\widetilde{\boldsymbol{D}}$, where $\widetilde{\boldsymbol{D}_{ij}}$ was scaled according to the *relative contribution* of gene j to the rate of change in gene i's expression:

$$\widetilde{\boldsymbol{D}_{ij}} = \frac{\boldsymbol{D}_{ij}\ \mathbb{I}_{|\boldsymbol{D}_{ij}|>|\boldsymbol{D}_{ji}|}}{\sum_{j'=1}^{n}|\boldsymbol{D}_{ij'}|\ \mathbb{I}_{|\boldsymbol{D}_{ij'}|>|\boldsymbol{D}_{j'i}|}}$$

We used the indicator function $\mathbb{I}$ to preserve only the stronger of the two effects ($\boldsymbol{D}_{ij}$ and $\boldsymbol{D}_{ji}$) for each gene pair, and also to prune self-effects ($\boldsymbol{D}_{ii}$). However, one may choose to include these phenomena by removing the $\mathbb{I}$ terms.

Finally, we subjected $\widetilde{\boldsymbol{D}}$ to a cut-off value $v_{\mathcal{C}}$ based on an appropriate percentile $\mathcal{C}$; we used $\mathcal{C} = 0.995$, meaning $v_{\mathcal{C}} = 99.5^{th}$ percentile of $\{|\ \widetilde{\boldsymbol{D}_{ij}}\ |\}_{\forall i,j}$. Specifically, we decided edge **existence** and **strength** in $\widehat{G_n}$ as follows:

$$\textbf{No edge}\text{: gene } j \overset{0}{\rightarrow} \text{gene } i \iff |\ \widetilde{\boldsymbol{D}_{ij}}\ | \leq v_{\mathcal{C}}$$

$$\textbf{Activating edge}\text{: gene } j \overset{\widetilde{\boldsymbol{D}_{ij}}}{\rightarrow} \text{gene } i \iff \widetilde{\boldsymbol{D}_{ij}} > v_{\mathcal{C}}$$

$$\textbf{Repressive edge}\text{: gene } j \overset{\widetilde{\boldsymbol{D}_{ij}}}{\rightarrow} \text{gene } i \iff \widetilde{\boldsymbol{D}_{ij}} < -v_{\mathcal{C}}$$

SM.2.2 Evaluation of explainability

We compared the estimated $\widehat{G_n}$ to the corresponding validation network in terms of out-degree correlation and edge-existence, calculating recovery AUC, true positive rate (classification TPR), and true negative rate (classification TNR). For our *in silico* experiments, we reverse-engineered a method to inform how sparsely PHOENIX had inferred the dynamics.

Since the ground truth graphs G_{350} and G_{690} were known, we found the value of $\mathcal{C} \in (0, 1)$, i.e. the aforementioned percentile cutoff of $\mid \widetilde{\boldsymbol{D}_{ij}} \mid$ values. that maximized the balanced classification accuracy ($\frac{\text{TPR}+\text{TNR}}{2}$), and used this $\mathcal{C}_{\max}$ as a measure of how sparsely the dynamics were inferred. Since $\mathcal{C}_{\max}$ reflects how well the trained PHOENIX model discriminates between true and non interactions, a low value of $\mathcal{C}_{\max}$ corresponded to dense dynamics, while a high value corresponded to sparser dynamics. When comparing explainability across different PHOENIX fits, we used $\mathcal{C}_{\max}$ to obtain corresponding values $\text{TPR}_{\max}$ and $\text{TNR}_{\max}$. This eliminated the dependence on any arbitrary cutoff and allowed us to compare networks in terms of best possible TPR and TNR.

SM.3 Benchmark experiments against competing methods

SM.3.1 Data sets used for benchmark experiments

For comparison to OOTB models, we used the same simulated data sets from SIM350 and SIM690 that were used for the PHOENIX experiments, with the same train-val-test split. For PRESCIENT [4], we found it to work only with consecutive integer time points, and so we generated 160 noisy trajectories from the ground-truth SIM350 and SIM690 at $t \in T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Each of the 1600 simulated vectors in $\{\{\boldsymbol{g}(t)_k \in \mathbb{R}^n\}_{t \in T}\}_{k=1}^{160}$ could be conceptualized as a **single cell**, with `cell_id` $= (t, k)$, at time t with expression values for n genes. We partitioned the cells via the same train-val-test split as the PHOENIX runs, and used this time-labelled "single cell training data" as PRESCIENT's input. We set all `cell_type` to `regular` in the meta data.

For "two-step" methods such as Dynamo [3], RNA-ODE [5], and scDVF [7], we know that they estimate dynamics by first reconstructing RNA velocity using inputs such as spliced and unspliced mRNA counts (step 1) and then estimating a vector field mapping expression to velocity (step 2). To avoid the need for uncommon input data types and to also emulate the **theoretically optimal performance** in step 1, we used the noiseless ground truth velocities $\frac{d\boldsymbol{g}(t)}{dt}$ as input into step 2 directly, since these true velocities could be retrieved from our simulator based on the random seeds used. Subsequently, we only tested step 2 (with the same train, validation, and test data sets used in PHOENIX and the corresponding ground truth velocities) and obtained an optimistic estimate of each method's performance.

SM.3.2 Out-of-the-box (OOTB) NeuralODE models

We tried to emulate how one might typically use OOTB NeuralODE models for the purpose of predicting gene expression dynamics [6]. Given a gene regulatory network of n genes, we assume that the gene expression of all genes $g_j(t)$ can have an effect on a specific $g_i(t)$: $\frac{d\boldsymbol{g}(t)}{dt} = f_{reg}\Big(\boldsymbol{g}(t)\Big) - \boldsymbol{g}(t)$, where $\boldsymbol{g}(t) = \{g_i(t)\}_{i=1}^{n}$ and $f_{reg} : \mathbb{R}^n \to \mathbb{R}^n$. We approximated f_{reg} with just an out-of-the-box neural network (NN_{OOTB}) with parameters $\boldsymbol{\theta}_{OOTB}$, and ReLU, tanh, or sigmoid activation functions:

$$\frac{d\boldsymbol{g}(t)}{dt} \approx \text{NN}_{OOTB}\Big(\boldsymbol{g}(t), \boldsymbol{\theta}_{OOTB}\Big) - \boldsymbol{g}(t)$$

For fair comparison, we created NN_{OOTB} such that it contained a similar number of hidden layers and trainable parameters as PHOENIX. Specifically, we designed (`basic_act_func`s used were Sigmoid, tanh, and ReLU) as follows:

```
(NN_OOTB): Sequential(
    (activation_0): basic_act_func()
    (layer_1): Linear(in_features=n, out_features=2*m, bias=True)
    (activation_1): basic_act_func()
    (layer_out): Linear(in_features=2*m, out_features=n), bias=True)

def forward(self, t, input):
    res = self.NN_OOTB(input)
    return(res - y)
```

We chose $m = 40$ for $n = 350$ (SIM350), and $m = 50$ for $n = 690$ (SIM690). We used the same basic learning strategy (initialization scheme, stopping criteria, learning rates, etc) as the experiments with PHOENIX. These details as well as other technicalities (ODESolver details, optimizer details, etc) can be found on our GitHub repository [34].

To evaluate explainability, we used the trained NN_{OOTB} to bootstrap an estimate of the encoded GRN. We randomly generated 100 input expression vectors via i.i.d standard uniform sampling $\{\boldsymbol{b_k} \in \mathbb{R}^n\}_{k=1}^{100}$. Next, for each gene j being studied in the system, we created a perturbed version of these input vectors $\{\boldsymbol{b_k^j}\}_{k=1}^{100}$, where only gene j was perturbed in each vector compared to the unperturbed $\{\boldsymbol{b_k}\}_{k=1}^{100}$. We then fed both sets of input vectors into the trained NN_{OOTB} to obtain corresponding perturbed output $\{\widehat{\boldsymbol{o}}_{\boldsymbol{k}}^{\boldsymbol{j}} \in \mathbb{R}^{n_g}\}_{k=1}^{200}$ and unperturbed output $\{\widehat{\boldsymbol{o}}_{\boldsymbol{k}} \in \mathbb{R}^{n_g}\}_{k=1}^{200}$. Next, for each perturbed gene j in the input, we measured how much the perturbed output of every other gene i changed, *as a proportion* of its unperturbed output. We observed this to generally yield better results in favor of the OOTB models than if we proceeded

without taking proportions, or normalized based on input perturbation.

$$\boldsymbol{\Delta}_{ij} = \frac{1}{200}\sum_{k=1}^{100}\left|\frac{\widehat{\boldsymbol{o}}^{\boldsymbol{j}}_{\boldsymbol{k}i} - \widehat{\boldsymbol{o}}_i}{\widehat{\boldsymbol{o}}_i}\right|$$

We used $\boldsymbol{\Delta}$ to create the normalized effects matrix $\widetilde{\boldsymbol{E}}$ such that $\widetilde{\boldsymbol{E}}_{ij}$ reflected the *relative contribution* of gene j's input to the proportion change in gene i's output: $\widetilde{\boldsymbol{E}}_{ij} = \frac{\boldsymbol{\Delta}_{ij}}{\sum_{j=1}^{n}\boldsymbol{\Delta}_{ij}}$. Finally, by thresholding the values of $\widetilde{\boldsymbol{E}}$ using an optimal cut off $\mathcal{C}_{\max}$ (see SM.2.2; this is the the percentile cut off of $\widetilde{\boldsymbol{E}}_{ij}$ values that maximizes $\frac{TPR+TNR}{2}$), we obtained adjacency matrices describing $\widehat{G_{350}}$ and $\widehat{G_{690}}$ that approximated the ground truth G_{350} and G_{690}, respectively.

SM.3.3 Assessing predictive accuracy and explainability

We provide some detail here, but source code (and even more technicalities) can be found on our GitHub repository [34], where this entire benchmarking pipeline will be made available. For assessing explainability in each case, we only explain up to how a matrix $\widetilde{\boldsymbol{E}}$ was obtained. We then processed $\widetilde{\boldsymbol{E}}$ in the same way as described at the end of SM.3.2 to obtain $\widehat{G_n}$. We calculated $\widehat{G_n}$'s average out-degree and obtained AUC by comparing to ground truth G_n.

SM.3.3.1 PRESCIENT

PRESCIENT [4] uses time-series scRNA-seq and cell-growth rate data to learn a scalar-valued potential function $\Psi(\boldsymbol{g}(t))$ with a neural network. A final drift model is obtained using automatic differentiation $\frac{\boldsymbol{g}(t)}{dt} = -\nabla\Psi(\boldsymbol{g}(t))$. We found that PRESCIENT's implementation could train only with consecutive integer time points, and so we generated 160 noisy trajectories from the ground-truth SIM350 and SIM690 at $t \in T = \{0,1,2,3,4,5,6,7,8,9\}$. Each of the 1600 simulated vectors in $\{\{\boldsymbol{g}(t)_k \in \mathbb{R}^n\}_{t\in T}\}_{k=1}^{160}$ could be conceptualized as a **single cell**, with `cell_id` $= (t,k)$, at time t with expression values for n genes. We partitioned the cells via the same train-val-test split as the PHOENIX runs, and used this time-labelled "single cell training data" as PRESCIENT's input. PRESCIENT requires a meta data file mapping each `cell_id` to its timepoint $\in T = \{0,1,2,3,4,5,6,7,8,9\}$. We also set the `cell_type` column in the meta-data file to be `regular` for all cells. PRESCIENT models can incorporate proliferation during training via computing a "growth weight" per cell using either lineage tracing data or KEGG gene signatures. In the absence of such data in our simulation, we set this weight to be 1 for all cells. We used the `prescient` package in `Python` to train a model with a single hidden layer of width k_{dim} followed by a softplus activation function. We trained for 2600 epochs and predicted trajectories using PRESCIENT's `net._drift()` function (which returns $\frac{\boldsymbol{g}(t)}{dt}$) along with `solve_ivp()` from `scipy.integrate`. We optimized k_{dim} based on validation trajectory prediction. We did not find a straightforward functionality in the package to extract a GRN.

SM.3.3.2 Dynamo

Given the input data (simulated gene expression and the corresponding ground truth RNA velocities), Dynamo [3] fits a sparse vector field mapping gene expression to RNA velocity using Gaussian kernel regression. Using the training trajectories from the input data, we created an `vf.SvcVectorField()` object (from the `dynamo` package in `Python`). The trained model (mapping expression to velocity) was used along with `solve_ivp()` from `scipy.integrate` to predict trajectories. Cross-validation was used (based on best MSE in validation trajectory prediction) to choose hyper-parameters M (number of control points) and sparsity parameter λ. MSE was calculated for final model on the test trajectories. To extract a GRN, we simulated 10^4 random expression vectors $\{\boldsymbol{g} \in \mathbb{R}^n\}_{k=1}^{10^4}$ and used `dynamo`'s Jacobian function `get_Jacobian()` to obtain an average Jacobian matrix $\widetilde{\boldsymbol{E}}$.

SM.3.3.3 RNA-ODE

Given the input data (simulated gene expression and the corresponding ground truth RNA velocities), RNA-ODE [5] fits black-box random forests mapping expression to velocity. We obtained `Python` source code from `https://github.com/RuishanLiu/VelocytoAnalysis`, to build random forest regressors. We predicted trajectories using the `predict()` function of the random forest along with `solve_ivp()` from `scipy.integrate`. We optimized the number of trees in the forest (`n_estimators`) based on validation trajectory prediction. The source code also has a `GET_GRN()` function to easily estimate a $\widetilde{\boldsymbol{E}}$ based on the fitted model using a GENIE3-like approach.

SM.3.3.4 scDVF

Given the input data (simulated gene expression and the corresponding ground truth RNA velocities), scDVF [7] fits a black-box autoencoder mapping expression to velocity. We obtained `Python` source code from `https://github.com/gersteinlab/DeepVelo`, to build a 4 layer encoder and 4 layer decoder with ℓ_1 regularization. Further details can be found on our GitHub repository [34]. We predicted trajectories using the auto-encoder's `predict()` function along with `solve_ivp()` from `scipy.integrate`. The source code also has functionalities to estimate $\widetilde{\boldsymbol{E}}$ a gene-correlation matrix of cells , based on simulating "retrograde trajectories".

SM.4 Creating *in silico* expression data: additional details

SM.4.1 Ground truth system using SimulatorGRN

We created a ground truth gene regulatory network (GRN) by sampling from S. cerevisiae (yeast) regulatory networks obtained from the SynTReN v1.2 supplementary data in simple interaction format (SIF) [35]. The SynTReN

file provides a directional GRN containing 690 genes and 1094 edges with annotations (activating vs repressive) for edge types; we defined this GRN to be ground truth network G_{690}. To obtain G_{350}, we used `SimulatorGRN` [23] in `R` to sample a subnetwork of 350 genes and 590 edges from G_{690}, using the `sampleGraph()` function. Next, to each edge (for instance from gene A to gene B), we assigned randomly generated $EC_{50}^{AB} \in (0.4, 0.6)$ and $\eta^{AB} \in (1.39, 1.8)$ values using the `randomizePrams()` function from `SimulatorGRN`; lists of edges with EC_{50}^{AB}, η^{AB} values are provided with the PHOENIX release [34]. Each EC_{50}^{AB}, η^{AB} pair defines the relationship between a regulator A and its target B. The `simulationGRN()` function then used the edges in G_{350} and G_{690} to define systems (SIM350 and SIM690) of normalised-Hill ODEs [24], where the activation of B by a single regulator A was modelled as:

$$\frac{dB}{dt} = f_{act}(A, EC_{50}^{AB}, \eta^{AB}) - B = \frac{\beta A^{\eta^{AB}}}{\beta - 1 + A^{\eta^{AB}}} - B,$$

$$\text{where } \beta = \frac{EC_{50}^{AB} - 1}{2EC_{50}^{AB} - 1}$$

Repression of B by A was simply modelled using $1 - f_{act}(A, EC_{50}^{AB}, \eta^{AB})$. Co-regulation by A_1 and A_2 was modelled as either a logical `AND`:

$$f_{act}(A_1, EC_{50}^{A_1B}, \eta^{A_1B}) \times f_{act}(A_2, EC_{50}^{A_2B}, \eta^{A_2B}),$$

or a logical `OR` gate:

$$\begin{aligned} &f_{act}(A_1, EC_{50}^{A_1B}, \eta^{A_1B}) + f_{act}(A_2, EC_{50}^{A_2B}, \eta^{A_2B}) \\ &- f_{act}(A_1, EC_{50}^{A_1B}, \eta^{A_1B}) \times f_{act}(A_2, EC_{50}^{A_2B}, \eta^{A_2B}) \end{aligned}$$

where we randomly assigned 90% of co-regulations as logical `AND`s, and the other 10% as logical `OR`s. Proceeding in this manner, `simulationGRN()` used the network structures to define dynamical systems SIM350 and SIM690, containing an ODE for each gene in each of G_{350} and G_{690} respectively. For any gene Z that has no upstream regulators, we assign $dZ/dt = 0$. We simulated expression trajectories from these ODE systems using `R`'s `desolve` package.

SM.4.2 Creating corrupted/misspecified prior models

For each noise level $\sigma_{\%} \in \{0\%, 5\%, 10\%, 20\%\}$ in our *in silico* experiments, we created a shuffled version of G_{350} (and similarly G_{690}) where we shuffled $\sigma_{\%}$ of the edges by relocating those edges to new randomly chosen origin and destination genes within the network. This yielded the shuffled network $G_{350}^{\sigma_{\%}}$ with corresponding adjacency matrix $\boldsymbol{A}^{\boldsymbol{\sigma_{\%}}}$. We set activating edges in $\boldsymbol{A}^{\boldsymbol{\sigma_{\%}}}$ to +1 and repressive edges to -1, and defined the simple linear prior domain knowledge model: $\mathcal{P}^*(\boldsymbol{\gamma_k}) = \boldsymbol{A}^{\boldsymbol{\sigma_{\%}}} \cdot \boldsymbol{\gamma_k} - \boldsymbol{\gamma_k}$.