Simulation-free Structure Learning for Stochastic Population Dynamics

Anonymous authors
Paper under double-blind review

Abstract

Modeling dynamical systems and unraveling their underlying structural dependencies is central to many domains in the natural sciences. Various physical systems, such as those arising in cell biology, are inherently high-dimensional and stochastic in nature, and admit only partial, noisy state measurements. Our primary motivating setting is single-cell biology, where destructive measurements yield unpaired population snapshots rather than longitudinal trajectories of the same cells. This poses a significant challenge for addressing the problems of modeling the underlying dynamics and inferring the network structure of these systems. Existing methods are typically tailored either for structure learning or modeling dynamics at the population level, but are limited in their ability to address both problems together. In this work, we address both problems simultaneously: we present StructureFlow, a novel and principled simulation-free training approach for jointly learning the structure and stochastic population dynamics of physical systems. We showcase the utility of StructureFlow for the tasks of structure learning from interventions and dynamical (trajectory) inference of conditional population dynamics. We empirically evaluate our approach on high-dimensional synthetic systems, a set of biologically plausible simulated systems, and an experimental single-cell dataset. We show that StructureFlow can learn the structure of underlying systems while simultaneously modeling their conditional population dynamics — a key step toward model-based mechanistic understanding of systems behavior.

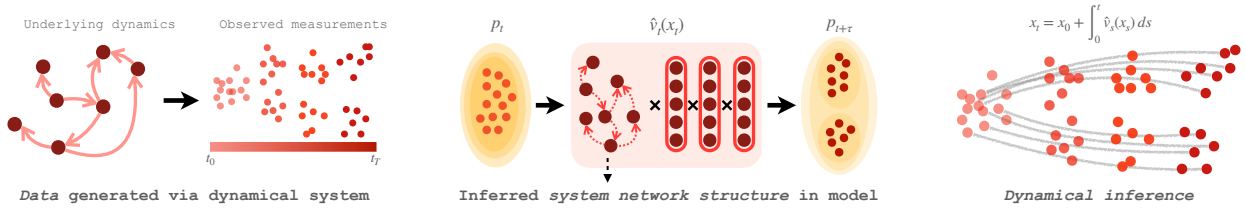


Figure 1: **Overview of StructureFlow for joint structure learning and dynamical inference.** Left: an underlying system, described by a directed interaction graph, generates the (visible) data; destructive measurement yields unpaired population snapshots at times $t_0, \ldots, t_T$, not individual trajectories. Middle: from snapshots p_t and $p_{t+\tau}$, StructureFlow learns an autonomous drift $\hat{v}(x_t)$ from a neural graphical model whose first layer encodes the interaction network. The network structure is recovered directly from the learned weights. Right: integrating the learned field, $x_t = x_0 + \int_0^t \hat{v}_s(x_s)\, ds$, gives continuous trajectories. The network is read directly from the model, so structure and dynamics are recovered jointly.

1 Introduction

Unraveling the underlying structure of physical systems from their stochastic dynamics, given noisy and partial measurements, is a central problem in many areas of science. Many physical systems, notably in cell and molecular biology, operate in out-of-equilibrium regimes, are high-dimensional, and are subject to intrinsic stochasticity (Elowitz et al., 2002; Schiebinger et al., 2019). This poses a significant challenge for the task of deciphering the underlying system structure and modeling the resulting dynamics. Effectively

addressing this problem is a crucial step toward gaining a mechanistic understanding of systems and the ability to predict their behavior under natural and perturbational conditions. This would, in turn, provide practitioners with a map for how to control and guide a system's response towards desirable states (Dixit et al., 2016; Norman et al., 2019). In cellular and molecular biology, such a tool would have significant implications in advancing our ability to interpret and model cell fate, development, response to disease (Rizvi et al., 2017; Binnewies et al., 2018; Gulati et al., 2020; Molè et al., 2021), predict perturbational responses in tissues and patients (Ji et al., 2021; Bunne et al., 2023; Peidli et al., 2024; Atanackovic et al., 2024), and facilitate experimental design (Zhang et al., 2023; Huang et al., 2024).

Currently, there exist two broad classes of approaches for deciphering the stochastic dynamics of physical systems from *data*. The first class, which we refer to as **dynamical inference**[1], commonly deals with constructing an estimate of the underlying vector field from partial and noisy measurements. From a good estimate of the vector fields, qualitative properties of dynamics can be interpreted, valuable for understanding cell dynamics and cell fate (Qiu et al., 2022; MacArthur, 2022). In this work, we focus on the preceding task of learning good estimates of vector fields from partially observed population snapshot data.

The second (and in our work, concurrent) class, **structure learning**[2], aims to reconstruct the *directional* relationships between each of the variables of a given system. Throughout the paper, the learned graph should be interpreted as a directed structural-dependency estimate under the Neural Graphical Model (NGM) and population-dynamics assumptions that are discussed in Section 3.1 and Appendix A.1, rather than as unrestricted causal recovery from snapshots alone. Existing methods are typically designed to tackle either task in isolation and are limited in their ability to address both problems together (Zheng et al., 2018; Huynh-Thu and Geurts, 2018; Gao et al., 2018; Brouillard et al., 2020; Atanackovic et al., 2023; Zhang, 2024). We posit that knowledge of the underlying vector field is advantageous for this task, and thus we address both problems simultaneously under a single-model.

A key feature of many of the systems we wish to model is that the measurement is *destructive*. For example, profiling a cell via single-cell RNA sequencing requires lysing the cell, precluding repeated observation of the same individual over time. Experimental data therefore arrive as population-level *snapshots* at discrete timepoints, with no temporal correspondence maintained between cells across timepoints (Schiebinger et al., 2019; Weinreb et al., 2018). While snapshot data is the canonical regime for single-cell biology, it also arises whenever longitudinal sampling is impractical, e.g., in costly survey designs or other settings with destructive readouts. We note that, while the methodology we develop is generally applicable to underdamped Langevin systems, the experimental scope of this work is centered on the single-cell setting.

To achieve this, we propose StructureFlow, a principled and *simulation-free* training method for joint dynamical inference and structure learning. Building on recent advances in score and flow matching ([SF]šM) (Tong et al., 2024) and entropy-regularized optimal transport (EOT) (Cuturi, 2013; Shi et al., 2023), StructureFlow learns a probability flow ordinary differential equation (PF-ODE) from snapshot data. This allows it to model the continuous evolution of a system's population while simultaneously inferring the underlying network structure embedded directly within the model's parameterization. The novelty is the integration and specialization of score/flow matching, an NGM-based structural drift, and intervention-conditioned population modeling for the joint snapshot setting; we do not claim that intervention masks or NGMs alone are new.

The simulation-free framing of StructureFlow is critical for high-dimensional systems common in cellular biology, as it avoids computationally expensive trajectory simulations during training. This is distinct from inference-time rollout, where we integrate the learned probability-flow ODE (PF-ODE) or stochastic differential equation (SDE) to evaluate held-out distributions. To best facilitate the joint learning task, we introduce a novel parameterization: an autonomous (time-independent) vector field represented by a Neural Graphical Model (NGM) (Bellot et al., 2022; Dinh and Ho, 2020) which captures the stationary system structure, while a time-dependent score function captures the evolving stochastic dynamics. We present an overview of our framework in Figure 1 and outline our core contributions below:

[1]This task is also commonly labeled and referred to as *trajectory inference* (Hashimoto et al., 2016; Weinreb et al., 2018; Schiebinger et al., 2019; Tong et al., 2020; Neklyudov et al., 2022). Since the concept of "trajectory" has been frequently used in the literature to refer to several non-equivalent things, in this work, we prefer to name this task *dynamical inference*.

[2]Also commonly referred to in literature as a *network inference*, *causal discovery*, and *system identification*.

- We formulate joint dynamical inference and structure learning as a multi-timepoint Schrödinger Bridge (SB) problem.
- We introduce StructureFlow, a novel simulation-free training approach tailored for simultaneously learning the underlying network structure and conditional population dynamics of a stochastic dynamical system from noisy and partial observations.
- We construct a comprehensive empirical evaluation for the joint inference task over an assortment of systems: (i) on high-dimensional synthetic systems, (ii) an collection of biologically plausible simulated systems, and (iii) an experimental single cell dataset with genetic interventions. We use our evaluation pipeline to showcase the application of StructureFlow for the joint structure learning and dynamical inference tasks, all while building an extensive benchmark of state-of-the-art methods.

2 Background and Preliminaries

2.1 The Single-Cell Setting

As single-cell biology is our primary application, we briefly discuss and connect it to the formalism we use. Single-cell RNA sequencing (Macosko et al., 2015; Klein et al., 2015) measures the expression of $d \gg 1$ genes for each cell in a sampled population. Each measured cell generates a vector of gene expression levels, represented as a point $\boldsymbol{x} \in \mathbb{R}^d$ in "gene expression space". A cell population can therefore be viewed as an empirical distribution, or point cloud, in this space (Schiebinger et al., 2019), and its evolution over time can be modeled as a stochastic process, such as the SDE in equation 1. Under this model, the population observed at any instant t is the time marginal of the process at t. Because sequencing a cell destroys it, the same cell cannot be measured twice; experiments therefore yield *unpaired* population snapshots at discrete timepoints, rather than tracked single-cell trajectories. This gives rise to the statistical problem of recovering dynamics from snapshot data described below. Perturbational single-cell experiments add controlled changes to this snapshot setting: rather than observing only a natural time course, one also observes populations after specific genes are perturbed and compares how their distributions evolve relative to the unperturbed condition. Genetic interventions (e.g. CRISPR knockouts Jinek et al. (2012); Dixit et al. (2016)) thus define the conditions c; recovering the directed graph $\boldsymbol{A}$ corresponds to gene regulatory network inference, and dynamical inference corresponds to predicting how the population evolves over time and under perturbation.

2.2 Problem: Modeling Stochastic Population Dynamics

We consider a dynamical model in $\mathbb{R}^d$ described by the stochastic differential equation (SDE):

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{v}_t(\boldsymbol{x}_t)\mathrm{d}t + \boldsymbol{\sigma}\mathrm{d}\boldsymbol{B}_t, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ is the state at time t, $\boldsymbol{v}_t : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is the drift, $\mathrm{d}\boldsymbol{B}_t$ are Brownian motion increments, and $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ is the diffusion coefficient matrix. For simplicity, we shall assume that $\boldsymbol{\sigma} = \sigma\mathbf{I}$ for some $\sigma > 0$, although the concepts we introduce can all be generalized to the setting of anisotropic noise (see Appendix A.9 for details).

Given $\boldsymbol{x}_0 \sim p_0$, where p_0 is a density over $\mathbb{R}^d$, the dynamics of equation 1 gives rise to a family of *marginals* $(p_t)_{t\in[0,1]}$, where p_t denotes the marginal probability distribution of the random variable $\boldsymbol{x}_t$ at time t, which are characterized by the accompanying *Fokker–Planck equation* (FPE):

$$\partial_t p_t = -\nabla \cdot (p_t(\boldsymbol{x}_t)\boldsymbol{v}_t(\boldsymbol{x}_t)) + \tfrac{1}{2}\nabla \cdot (\sigma^2 \nabla p_t(\boldsymbol{x}_t)). \tag{2}$$

The solution to the FPE at time t, $p_t(\boldsymbol{x})$, gives the probability density of the population at time t, starting from $p_0(\boldsymbol{x})$. Importantly, equation 2 can be reformulated as an equivalent *probability flow* ordinary differential equation (ODE) (Song et al., 2021; Maoutsa et al., 2020):

$$\partial_t p_t(\boldsymbol{x}) = -\nabla \cdot (p_t(\boldsymbol{x})\boldsymbol{u}_t(\boldsymbol{x})), \qquad \boldsymbol{u}_t(\boldsymbol{x}) = \boldsymbol{v}_t(\boldsymbol{x}) - \tfrac{1}{2}\sigma^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}). \tag{3}$$

In the above, $\boldsymbol{u}_t$ is the *probability flow field* and is related to the drift of the SDE equation 1 up to the addition of a term involving the score, $\boldsymbol{s}_t(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$. As will be apparent in Section 2.3, the probability flow formulation of the FPE equation 2, and hence the SDE equation 1, opens up an avenue for addressing dynamics inference without the need for costly simulation-based methods.

Inference problem setup. We consider a setting where empirical snapshot observations are available at multiple timepoints $t_0, \ldots, t_{T-1}$ under different conditions $c \in \mathcal{C}$, yielding a set of empirical marginal distributions $\{\hat{p}^{(c)}_{t_i}\}_0^{T-1}$ over $\boldsymbol{x}_{t_i} \in \mathbb{R}^d$ for each condition c. Each marginal comprises N_i i.i.d. samples assumed to arise from some unknown SDE of the form equation 1. We assume the noise level $\boldsymbol{\sigma}$ is known, while the vector field $\boldsymbol{v}_t$ is unknown[3]. From here, our objective is twofold: **(1; *dynamical inference*)** approximate the (*conditional*) vector field $\boldsymbol{v}_t$ of the underlying stochastic dynamics, and **(2; *structure learning*)** recover a directed graph $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ that captures dependencies among d variables and represents the underlying data generative process of the system. We formulate this joint inference task through the lens of Schrödinger Bridges.

2.3 Simulation-Free Schrödinger Bridges via Score and Flow Matching

The Schrödinger Bridge Problem (SBP) is concerned with finding the most likely stochastic evolution transporting a source density q_0 to a target density q_1, given a reference process that encodes prior knowledge of the dynamics. In its dynamical form, the SBP is typically formulated in terms of *laws of stochastic processes*, i.e. probability measures on the path space $C([0,1], \mathbb{R}^d)$ that describe the distribution of entire sample trajectories. Writing $\mathbb{P}$ to be the law of a process transporting q_0 to q_1 and $\mathbb{Q}$ to be the reference measure, we seek a stochastic process $\mathbb{P}^*$ satisfying:

$$\mathbb{P}^* = \underset{\mathbb{P}:p_0=q_0,p_1=q_1}{\arg\min} \; \mathrm{KL}(\mathbb{P}\|\mathbb{Q}) \tag{4}$$

for marginals p_t of $\mathbb{P}$, and $\mathrm{KL}(\mathbb{P}\|\mathbb{Q}) = \int \mathrm{d}\mathbb{P} \log(\mathrm{d}\mathbb{P}/\mathrm{d}\mathbb{Q})$ is the Kullback–Leibler divergence. When the reference process $\mathbb{Q}$ is the Brownian motion, a key result from Schrödinger Bridge theory (Föllmer, 1988) is that the solution of equation 4 takes the form of a mixture of Brownian bridges $\mathbb{Q}_{xy}$ with respect to a *coupling* π of the distributions (q_0, q_1):

$$\mathbb{P}^\star = \int \mathbb{Q}_{xy} \mathrm{d}\pi(x,y). \tag{5}$$

Furthermore, the SBP coupling π amounts to the solution of an entropic optimal transport problem (Léonard, 2014):

$$\pi^\star = \arg\min_{\pi\in\Pi(q_0,q_1)} \tfrac{1}{2}\mathbb{E}_\pi\|x-y\|_2^2 + \varepsilon \mathrm{KL}(\pi|q_0\otimes q_1) \tag{6}$$

with regularization parameter $\varepsilon = \sigma^2 > 0$, where $\Pi(q_0, q_1)$ denotes the set of admissible couplings between q_0 and q_1. In practice when (q_0, q_1) are discrete distributions, this can be solved extremely efficiently using the Sinkhorn algorithm (Cuturi, 2013). While the coupling π is easy to compute from samples, finding a solution to the dynamical SBP in continuous time is desirable for modelling continuous dynamics. With a notable exception being the case of Gaussian measures (Bunne et al., 2022) where analytical expressions are available, the dynamical SBP equation 4 does not admit a straightforward solution for $\boldsymbol{v}_{\mathrm{SB}}(t, \boldsymbol{x})$, defined as the drift of the SDE, $\mathrm{d}\boldsymbol{x}_t = \boldsymbol{v}_{\mathrm{SB}}(t, \boldsymbol{x}_t)\mathrm{d}t + \sigma \mathrm{d}\boldsymbol{B}_t$, which underlies the Schrödinger Bridge process $\mathbb{P}$.

Numerous works aim to build approximations to $\boldsymbol{v}_{\mathrm{SB}}$ and hence solutions to the dynamical SBP (Chen et al., 2022; Bortoli et al., 2021; Shi et al., 2023; Tong et al., 2024). In particular, Tong et al. (2024) proposes to use score matching (Hyvärinen, 2005) and flow matching (Lipman et al., 2023a) ([SF]^{2}M) as a natural methodology for building the approximation to $\boldsymbol{v}_{\mathrm{SB}}$ in order to solve for $\mathbb{P}$. Note $\boldsymbol{v}_{\mathrm{SB}}(t, \boldsymbol{x})$ simply corresponds to the drift $\boldsymbol{v}_t$ defined by the SBP. The key observation is to leverage the reciprocal process characterization of $\mathbb{P}$ (Léonard, 2014) as a mixture of Brownian bridges, and the fact that Brownian bridges conditioned on endpoints $(\boldsymbol{x}_0, \boldsymbol{x}_1) =: \boldsymbol{z}$ admit closed form expressions for their probability flow $\boldsymbol{v}^\circ_t(\boldsymbol{x}|\boldsymbol{z})$ and score $\nabla \log p_t(\boldsymbol{x}|\boldsymbol{z})$:

$$\boldsymbol{v}^\circ_t(\boldsymbol{x}|\boldsymbol{z}) = \frac{1-2t}{t(1-t)}(\boldsymbol{x} - (t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_0)) + (\boldsymbol{x}_1 - \boldsymbol{x}_0); \quad \nabla \log p_t(\boldsymbol{x}|\boldsymbol{z}) = \frac{t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_0 - \boldsymbol{x}}{\sigma^2 t(1-t)}. \tag{7}$$

Together, these provide the probability flow characterisation of the Brownian bridge. With this in hand, and leveraging the fact that the dynamical SBP can be constructed as a mixture of Brownian bridges (equation 5), Tong et al. (2024) propose to construct a neural approximation to the SB flow field

[3]Throughout we *assume* this datageneration model holds for every condition c.

$\boldsymbol{v}^\theta(t,\boldsymbol{x}) \approx \boldsymbol{v}(t,\boldsymbol{x}) := \boldsymbol{v}_t^\circ(t,\boldsymbol{x}) - \frac{\sigma^2}{2}\nabla \log p_t(\boldsymbol{x})$. The flow field $\boldsymbol{v}^\theta(t,\boldsymbol{x})$ is trained together with a score field $\boldsymbol{s}^\theta(t,\boldsymbol{x})$ using the conditional score and flow matching loss

$$\mathcal{L}_{[\mathrm{SF}]^2\mathrm{M}}(\theta) = \mathbb{E}_{t,\boldsymbol{z},\boldsymbol{x}}\left[\|\boldsymbol{v}_t^\theta(\boldsymbol{x}) - \boldsymbol{v}_t^\circ(\boldsymbol{x}|\boldsymbol{z})\|^2 + \ \lambda(t)\|\boldsymbol{s}_t^\theta(\boldsymbol{x}) - \nabla \log p_t(\boldsymbol{x}|\boldsymbol{z})\|^2\right]. \tag{8}$$

In the above, $\lambda(t) > 0$ weighs the score as a function of time t, and $\boldsymbol{z} := (\boldsymbol{x}_0, \boldsymbol{x}_1) \sim \pi(\boldsymbol{x}_0, \boldsymbol{x}_1)$ where π is the entropic OT coupling in equation 5 obtained via the Sinkhorn algorithm. Following Proposition 3.4 of Tong et al. (2024), minimising $\mathcal{L}_{[\mathrm{SF}]^2\mathrm{M}}(\theta)$ then approximates the solution to the SBP in equation 4 under mild conditions. Additionally, Chen et al. (2023); Theodoropoulos et al. (2025); Rohbeck et al. (2025) model a multi-marginal SBP, where successive time-marginal couplings are not computed independently, but rather globally across all timepoints.

3 StructureFlow: Joint Inference of Structure and Dynamics

Our objective is to jointly infer the network structure and population dynamics of the underlying stochastic system. StructureFlow achieves this in a novel simulation-free manner during training. In this section we introduce the central components of our framework, namely: **(i)** improved parameterization tailored for the joint task, **(ii)** modeling conditional stochastic population dynamics from interventional data, and **(iii)** simulation-free training for learning conditional stochastic population dynamics.

3.1 Structural Vector Field Parameterization

We consider a time-independent (or *autonomous*) vector field, which models the underlying structure of a system, and a time-dependent score function accounting for stochasticity. We assume that the underlying system structures (graphs) are stationary between adjacent marginals $(\hat{p}_{t_i}^{(c)}, \hat{p}_{t_{i+1}}^{(c)})$. We posit that this assumption yields an easier *joint* inference problem, allowing us to parameterize *a single* structure within the autonomous vector field.

Autonomous vector field and time-dependent score parameterization. From equation 1 we seek to recover the autonomous vector field $\boldsymbol{v}$, while the objective in (Tong et al., 2024) targets the SB flow $\boldsymbol{v}_t^\circ$. The quantities of the ground truth process equation 1 are related via $\boldsymbol{v}_t^\circ(\boldsymbol{x}_t) = \boldsymbol{v}(\boldsymbol{x}_t) - \frac{\sigma^2}{2}\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$, where $\boldsymbol{v}_t^\circ, \nabla \log p_t$ are the probability flow field and the score of the Brownian bridge pinned at $(t_i, x_i), (t_{i+1}, x_{i+1})$ respectively. We therefore make the parametrization choice $\hat{\boldsymbol{v}}_t(\boldsymbol{x}_t) = \boldsymbol{v}^\theta(\boldsymbol{x}_t) - \frac{\sigma^2}{2}\boldsymbol{s}_t^\phi(\boldsymbol{x}_t)$. Since time-independent vector fields, $\boldsymbol{v}$, are a function of state, and not spurious temporal correlations, they are better suited for modeling underlying structures which are stationary over time, as we assume. This leads to a more principled modeling approach.

Modeling dynamic structural dependencies via neural graphical vector fields. We consider a neural structural model following the definition of Bellot et al. (2022), where we assume there exists functions $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$ such that for $j = 1, \ldots, d$, and $\boldsymbol{v}_j : \mathcal{X}^d \to \mathbb{R}$, where $\mathcal{X}^d$ is a bounded subset of $\mathbb{R}^d$. We can rewrite equation 1 under the neural dynamic structural model formalism as

$$\mathrm{d}x_j(t) = \boldsymbol{v}_j(\boldsymbol{x}(t))\mathrm{d}t + \boldsymbol{\sigma}_j \mathrm{d}B_j(t), \quad \boldsymbol{x}(0) \sim p_0, \tag{9}$$

where $\mathrm{d}x_j(t)$ denotes the structural dependency for the instantaneous rate of change of the jth variable dependent on all state observations $\boldsymbol{x}$ at time t.[4] We use an NGM (Bellot et al., 2022) to parameterize the dynamic structural relationships defined in equation 9.

NGMs are a class of neural graphical model parameterizations designed to represent complex variable (features) dependencies in a computationally efficient manner (Shrivastava and Chajewska, 2023). Bellot et al. (2022) extend this formulation for the structure learning of continuous-time dynamical systems, motivating their application in our framework. The NGM lets us parameterize the structural relationships of a system with d variables directly within the *autonomous* vector field $\boldsymbol{v}(\boldsymbol{x}_t)$. We use an NGM to approximate the

[4]We note that we are using $\boldsymbol{x}(t)$ interchangeably with $\boldsymbol{x}_t$.

autonomous component of the ground truth process described in equation 1 ($\boldsymbol{v}(\boldsymbol{x}_t)$ instead of $\boldsymbol{v}_t(\boldsymbol{x}_t)$). The NGM parameterization of the *autonomous* vector field is defined as:

$$\boldsymbol{v}^{\theta_j}(\boldsymbol{x}_t) = \psi(\cdots\psi(\psi(\boldsymbol{x}_t\theta_j^A)\theta_j^1)\cdots)\theta_j^K, \quad j = 1, \ldots, d, \tag{10}$$

where $\theta_j^A \in \mathbb{R}^{d\times h}$ denotes a weight matrix (or graph layer) representing the dynamic dependencies between $\mathrm{d}x_j(t)$ and $\boldsymbol{x}(t)$, $\theta_j^k \in \mathbb{R}^{h\times h}$ for $k = 1, \ldots, K-1$ are the weight matrices of each corresponding hidden layer, $\theta_j^K \in \mathbb{R}^{h\times 1}$ is the final layer's weight matrix yielding a scalar output, and $\psi(\cdot)$ is an activation function. We assume variable dependencies defined by θ_j^A are sparse and enforce sparsity on θ_j^A using group lasso regularization during training. We include further details regarding the NGM and its optimization for dynamic structure learning in Appendix A.1, as well as principled theoretical justification for recovering the underling network structure in Appendix A.3.

3.2 Modeling Conditional Population Dynamics

StructureFlow explicitly learns conditional stochastic dynamics (trajectories) using both interventional and observational data and simultaneously infers the underlying system structure. We model interventions as ideal *knockouts*, i.e. where for a given intervention, a variable is entirely removed from the system, and the intervention does not directly affect other variables of the system. For an intervention c, we define a binary mask $\boldsymbol{M}^{(c)} \in \{0,1\}^{d\times d}$ as:

$$M_{ji}^{(c)} = \begin{cases} 0 \text{ if } i = c, j \neq c \\ 1 \text{ otherwise} \end{cases} \tag{11}$$

Where i is the possible source of influence and j is the target. The mask is structured to represent the severed outgoing influences of the variable c. In the observational setting, no mask is applied. Under condition c, the vector field's parameters become $\theta_j^{A(c)}$, obtained by an element-wise product with $\boldsymbol{M}^{(c)}$: $\theta_j^{A(c)} = \boldsymbol{M}^{(c)} \odot \theta_j^A$. The drift is then evaluated as:

$$\boldsymbol{v}_c^{\theta_j}(\boldsymbol{x}_t|c) = \psi(\cdots\psi(\psi(\boldsymbol{x}_t\theta_j^{A(c)})\theta_j^1)\cdots)\theta_j^K. \tag{12}$$

We can then recover our underlying matrix estimate $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ as $A_{ji} = \|(\theta_j^A)_{i,:}\|_2$. Similarly, we condition the score $\boldsymbol{s}_t^\phi(\boldsymbol{x}_t)$ on c, i.e., $\nabla_{\boldsymbol{x}_t}\log p_t(\boldsymbol{x}_t|\boldsymbol{k}^{(c)})$ where $\boldsymbol{k}^{(c)} \in \{0,1\}^d$ is a conditional input vector with $k_i^{(c)} = 1$ if i is perturbed under c, otherwise $k_i^{(c)} = 0$.

3.3 Simulation-free end-to-end Training

StructureFlow builds on the $[\mathrm{SF}]^2\mathrm{M}$ framework introduced by Tong et al. (2024) to learn a neural approximation of $\boldsymbol{v}$ without simulation. We use Entropic Optimal Transport (EOT) to pair points drawn from temporally adjacent snapshots $\hat{p}_{t_i}^{(c)}$ and $\hat{p}_{t_{i+1}}^{(c)}$. We use the Sinkhorn algorithm (Cuturi, 2013) to estimate the probabilistic EOT couplings between pairs of distributions only a single time before training starts, and do not need to be recomputed. From each coupling, we draw paired samples (x_0, x_1) across $[t_i, t_{i+1}]$ and compute the corresponding target drift of the probability flow $\boldsymbol{v}_t^\circ$ and target score $\boldsymbol{s}_t = \nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}_t \mid c)$. We parametrize the drift of the probability flow as:

$$\hat{\boldsymbol{v}}_{t,c}(\boldsymbol{x}_t;\Theta|\boldsymbol{M}^{(c)},\boldsymbol{k}^{(c)}) = \boldsymbol{v}_c^\theta(\boldsymbol{x}_t|\boldsymbol{M}^{(c)}) - \frac{\sigma^2}{2}\boldsymbol{s}_t^\phi\left(\boldsymbol{x}_t|\boldsymbol{k}^{(c)}\right). \tag{13}$$

for conditional probability flow parametrization $\hat{\boldsymbol{v}}_{t,c}$ under condition c. We optimize model parameters $\Theta = (\theta, \phi)$ by regressing to the targets $\boldsymbol{v}_t^\circ$ and $\boldsymbol{s}_t$ via the StructureFlow loss

$$\mathcal{L}_{\mathrm{SF}}(\Theta) = \sum_c \mathbb{E}\left[(1-\alpha)\,||\hat{\boldsymbol{v}}_{t,c}(\boldsymbol{x}_t;\Theta|\boldsymbol{M}^{(c)},\boldsymbol{k}^{(c)})t - \boldsymbol{v}_t^\circ||^2 + \alpha||\boldsymbol{s}_{t,c}^\phi(\boldsymbol{x}_t|\boldsymbol{k}^{(c)}) - \boldsymbol{s}_t||^2\right], \tag{14}$$

where the expectation is over $t \sim \mathcal{U}(0,1)$ and $\boldsymbol{x}_t \sim p_t(\boldsymbol{x})$, and $0 \leq \alpha \leq 1$ weighs the vector field loss and score loss. During training, we apply group lasso on θ_j^A to encourage the model to learn sparse dependencies. We include detailed pseudo code for training StructureFlow in Algorithm 1.

We note that we choose to solve the EOT problem to pair successive time marginals ($\hat{p}^{(c)}_{t_i}$ and $\hat{p}^{(c)}_{t_{i+1}}$) independently of the other marginals $\hat{p}_t \mid t \notin \{i, i+1\}$, not globally across all time marginals like is done in Chen et al. (2023); Theodoropoulos et al. (2025); Rohbeck et al. (2025). As noted by Theodoropoulos et al. (2025), coupling adjacent snapshots independently, as StructureFlow does, can limit the capture of long-range temporal dependencies and the coherence of inferred trajectories across the full sequence of timepoints. Global multi-marginal couplings (Chen et al., 2023; Rohbeck et al., 2025; Theodoropoulos et al., 2025) instead condition on all timepoints jointly to learn smoother, globally consistent trajectories. We do not investigate the use of such couplings in StructureFlow, and leave this as potential direction for future work.

4 Related Work

Structure learning for continuous dynamics. There exist several works for structure learning of continuous dynamics. Notably, Bellot et al. (2022) introduce a continuous-time framework for inferring structure of the underlying dynamical system from time-series data. Wang et al. (2023) extend this approach to the stochastic and Bayesian setting. However, these approaches rely in computationally expensive neural ODE solvers during training, Recent lines of work have introduced structure learning methods for continuous dynamics specifically tailored for biological systems and the inference of gene regulatory networks (GRNs) (Huynh-Thu et al., 2010a; Huynh-Thu and Geurts, 2018; Gao et al., 2018; Ishikawa et al., 2023). However, all these methods assume a selection of fixed ODE parameterizations, linear dynamics, or perturbation-specific recovery, limiting their ability to model stochastic trajectories and complex network interactions. Symbolic regression-based approaches offer a complimentary line of work (Brunton et al., 2016; Sun et al., 2023; Dakhmouche et al., 2025). Dakhmouche et al. (2025) also solve for continuous population dynamics (dynamical/trajectory inference) and provide an interpretable model via the learned symbolic equations, but do not explicitly recover a network structure. Brunton et al. (2016); Sun et al. (2023) also use symbolic regression to recover closed-form equations, but assume fully observed trajectories.

Structure learning from time-series data. A substantial body of work addresses structure learning when longitudinal observations of individual particles are available. Classical approaches infer dependencies via Granger causality or lag-based conditional independence testing, with PCMCI and its variants (Runge et al., 2019) being a widely used representative. Etesami and Kiyavash (2014; 2016) formalize this perspective through directed information graphs for general dynamical systems. More recent neural approaches include neural relational inference (Kipf et al., 2018), which jointly learns interaction graphs and trajectories under deterministic dynamics, and SCOTCH (Wang et al., 2023), which extends Bayesian structure learning to SDEs. A complementary line of work directly recovers governing equations from observed trajectories via sparse regression. SINDy (Brunton et al., 2016) fits a sparse linear combination of candidate basis functions to time derivatives of state observations, with the recovered nonzero coefficients revealing which variables drive each other's dynamics, and stochastic extensions (Boninsegna et al., 2018) adapt this framework to SDEs. These methods are powerful in their setting but rely on maintained particle identity across timepoints, an assumption violated in the destructive-measurement regime considered here. We also empirically compare against PCMCI adapted to snapshots via OT pseudo-trajectories in Appendix E.13, table 22

Joint inference of structure and dynamics. While both dynamical inference and structure learning are central to understanding stochastic systems, many existing methods treat these tasks in a decoupled fashion (Qiu et al., 2022; Sha et al., 2024). These procedures are centered around learning a dense vector field from the observed data, subsequently extracting structural information post-hoc. Moreover, they do not incorporate any prior on sparsity or explicitly model structure during training. Tong et al. (2024) provide a preliminary investigation into the use of the NGM architecture in conjunction with $[\mathrm{SF}]^2\mathrm{M}$ for jointly inferring structure and dynamics, but do so in a limited setting, i.e. don't consider interventions and assume access to a significant quantity of time-points. We demonstrate later (Section 5 Figure 3) that the trivial application of the NGM architecture under the $[\mathrm{SF}]^2\mathrm{M}$ setup does not achieve competitive performance. Lin et al. (2025) propose a neural ODE-based approach for the joint inference task, but do not model stochastic dynamics, only consider two time-points, and require simulation during training (akin to Bellot et al. (2022)). Concurrent work also explores masking-based intervention modeling in single-cell generative models (Jiang

et al., 2025; Lorch et al., 2026), highlighting joint structure-and-dynamics learning under interventions as an actively studied problem. StructureFlow addresses these limitations for the joint task.

5 Experiments

We evaluate StructureFlow on both synthetic and real-world datasets to assess its ability to recover underlying network structure and infer stochastic population dynamics. Our experiments focus on three key capabilities: structure learning of dynamical systems from population data, dynamical inference across left-out time-points, and prediction of responses to unseen interventions (knockouts). We also include a scaling study to highlight our method's computational efficiency. We provide further details for these experiments in Appendix A, B, and E.

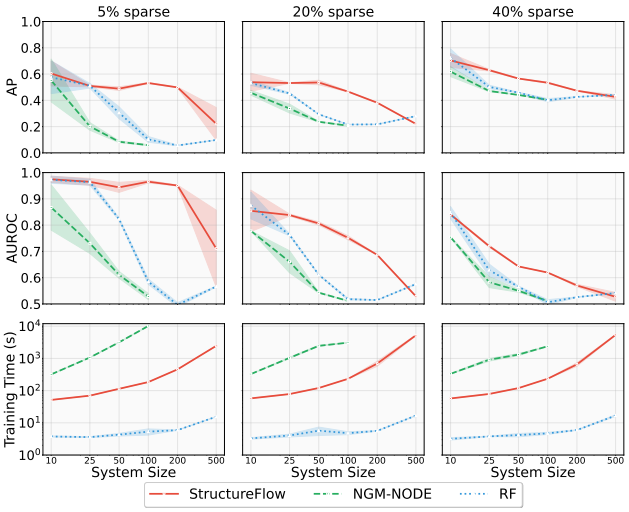


Figure 2: **StructureFlow yields a favorable structure-learning accuracy–runtime tradeoff when scaled to high-dimensional systems.** We compare with NGM-NODE and RF on synthetic linear systems with varying dimensionality (d) and system (graph) sparsity levels (5%, 20%, 40%).

5.1 Experiments on Synthetic High-dimensional Systems

We evaluate StructureFlow for the structure learning task across varying system dimensionality using randomly generated Erdos–Renyi graphs (Erdös and Rényi, 1959) and derive synthetic data via linear SDE simulation (see Appendix A.6 for details). We vary the dimensionality of the system d from 10 to 500 variables and the sparsity (proportion of non-zero edges) from 5% to 40% of the underlying graph which defines the data-generative process of the dynamical system. We consider a simulation-based approach, NGM-NeuralODE (NGM-NODE) (Bellot et al., 2022), and Reference Fitting (RF) (Zhang, 2024), as baselines, since they are capable of the joint structure learning and trajectory inference task. We report results in Figure 2 showing area under the receiver operating characteristic curve (AUROC), average precision (AP), and training time (seconds). We observe that StructureFlow exhibits favorable scaling performance as dimensionality of the system increases. While the baseline methods are competitive on small systems, we see that StructureFlow consistently maintains strong performance in high-dimensional settings while exhibiting improved computational efficiency in terms of training time compared to NGM-NODE, especially for large d. Thus, our scaling claim is about the accuracy vs. runtime tradeoff against alternative simulation-based approaches, rather than uniformly cheaper than every baseline. We remark that RF is the most computationally efficient in this regard as it makes assumptions of linearity of the underlying system. Due to this, RF suffers from poor expressivity and exhibits poor performance on the second element of the joint task – dynamical (trajectory) inference. We show this in the following sections.

5.2 Experiments on Simulated Biological Systems

We consider six simulated biological systems: trifurcating (**TF**), cyclical (**CY**), long linear (**LL**), swirling (**SW**), bifurcating (**BF**), and a curated system mimics hematopoietic stem cell (**HSC**) differentiation. All datasets are simulated via BoolODE (Pratapa et al., 2020) under both observational and interventional (knockout) conditions across multiple time-points (see details in Appendix A). See Appendix E for extended synthetic system results and ablations. In this section, we empirically evaluate StructureFlow for the joint structure learning and dynamics inference tasks.

StructureFlow effectively infers network structure of simulated biological systems. To evaluate how well StructureFlow recovers the underlying network structure, we extract the inferred graph from

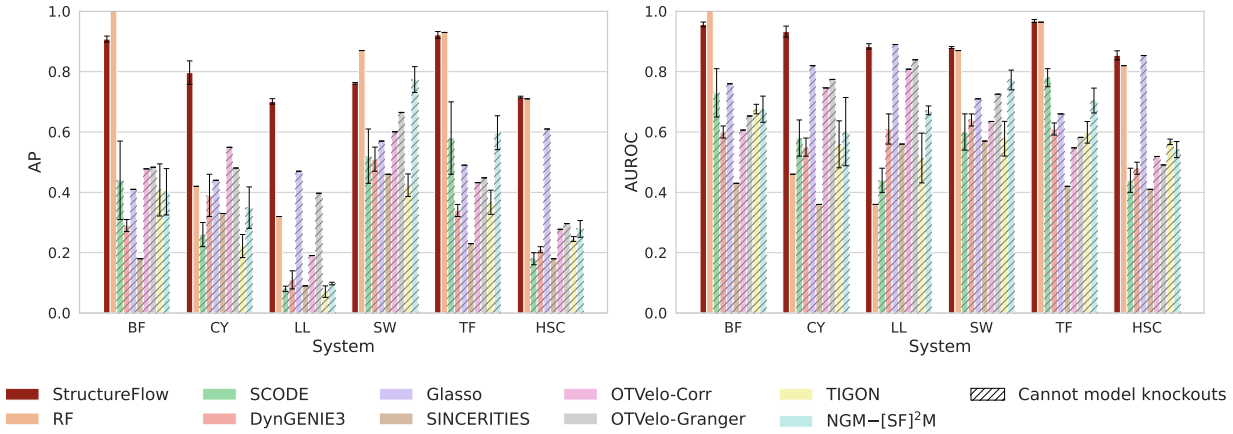
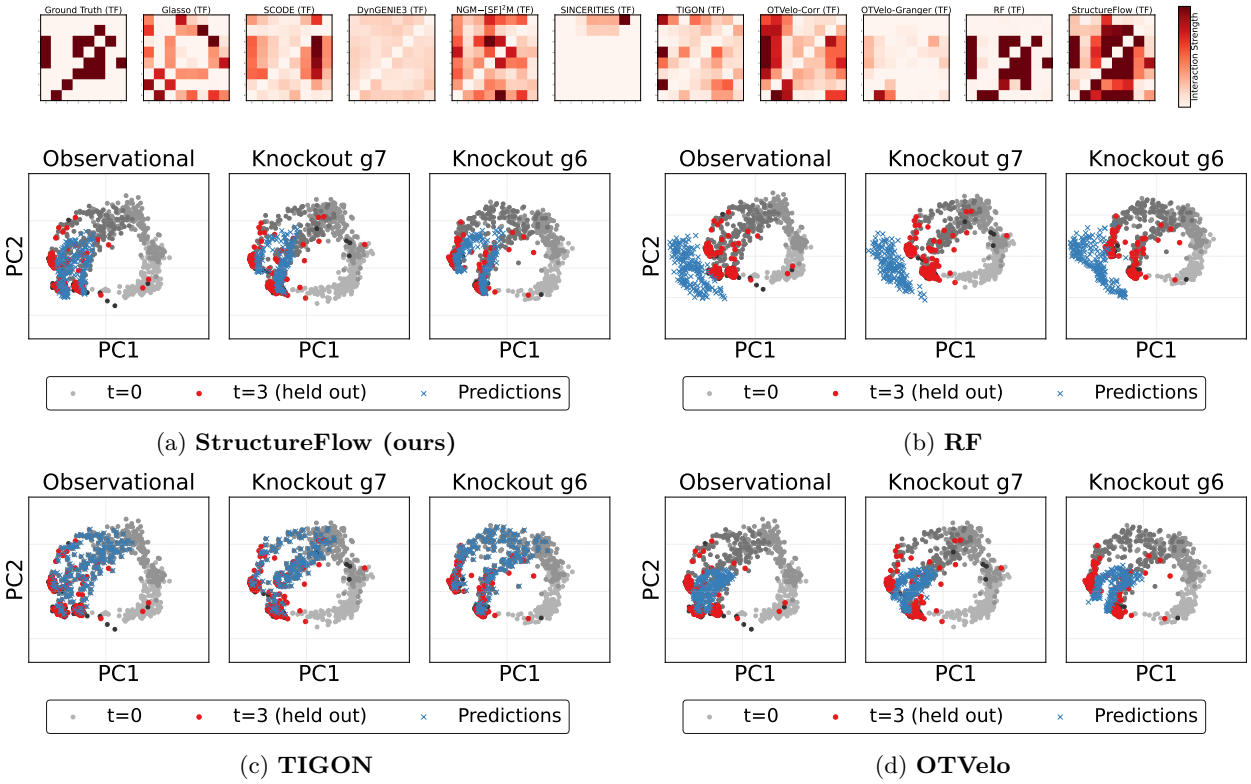


Figure 3: **StructureFlow is consistently a top performing *structure learning* method across simulated biological systems.** Here, we use interventional (with *knockouts*) and observational (no *knockouts*) data, and report average precision (AP) and area under the ROC curve (AUROC) scores.



(a) **StructureFlow (ours)** (b) **RF**

(c) **TIGON** (d) **OTVelo**

Figure 4: **Summary of inferred structure and visualization of model predicted dynamics.** (top) connectivity matrices (heatmaps) showing ground truth and inferred graphs for the **TF** (trifurcating) system. The x/y-axis labels correspond to system variables and shading indicates edge interaction strength. (bottom) visual comparison of dynamical inference using four different methods: StructureFlow, RF, TIGON, and OTVelo, respectively. We show 2D plots of left-out timepoint prediction for the observational (wild-type) setting and for two *seen* interventions (knockouts) selected for their diverse trifurcating paths. This figure is intended as a qualitative visualization of structure and representative trajectory behavior; the performance comparison is quantified in Figure 3 and Table 1.

Table 1: **StructureFlow achieves competitive performance for *dynamical inference* on simulated biological systems.** Shown is a comparison of dynamical inference methods for learning conditional population dynamics across synthetic biological systems. We include [SF]^{2}M for comparison, but note that [SF]^{2}M does not infer underlying network structure (i.e. cannot address joint task). Colors indicate **1st best performer** and 2nd best performer for models that perform the joint inference task.

	TF		CY		LL		SW		BF		HSC (Curated)	
	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD
[SF]^{2}M	0.761 ± 0.014	0.134 ± 0.005	0.517 ± 0.015	0.153 ± 0.002	0.847 ± 0.053	0.190 ± 0.010	0.572 ± 0.008	0.215 ± 0.005	0.660 ± 0.007	0.140 ± 0.004	0.664 ± 0.011	0.083 ± 0.003
RF	1.376 ± 0.000	0.191 ± 0.000	2.086 ± 0.000	0.326 ± 0.000	2.120 ± 0.007	0.296 ± 0.000	1.371 ± 0.000	0.290 ± 0.000	1.495 ± 0.000	0.242 ± 0.000	0.950 ± 0.000	0.066 ± 0.000
OTVelo	1.874 ± 0.262	0.582 ± 0.039	1.972 ± 0.144	0.622 ± 0.034	2.878 ± 0.361	0.647 ± 0.015	1.940 ± 0.144	0.577 ± 0.022	1.847 ± 0.277	0.591 ± 0.042	1.942 ± 0.006	0.603 ± 0.002
TIGON	1.409 ± 0.294	**0.063 ± 0.022**	1.224 ± 0.053	**0.044 ± 0.003**	1.840 ± 0.293	**0.066 ± 0.039**	0.957 ± 0.190	**0.017 ± 0.009**	1.394 ± 0.294	**0.066 ± 0.021**	**0.666 ± 0.363**	**0.012 ± 0.014**
StructureFlow	**0.789 ± 0.012**	0.135 ± 0.004	**0.576 ± 0.012**	0.140 ± 0.004	**0.842 ± 0.031**	0.196 ± 0.008	**0.610 ± 0.010**	0.220 ± 0.006	**0.694 ± 0.017**	0.142 ± 0.007	0.683 ± 0.011	0.068 ± 0.003

the first layer of the parameterized vector field (defined in equation 10) which interprets the learned weight matrix as a proxy for variable-variable (gene-gene) interactions. We use AP and AUROC to evaluate how closely the StructureFlow inferred graphs recapitulate the ground truth structure used to simulate the respective synthetic biological systems.[5] We compare StructureFlow to a variety of baseline methods (some of which are tailored for structure learning in biological systems): RF (Zhang, 2024), OTVelo (Zhao et al., 2024), TIGON (Sha et al., 2024), dynGENIE3 (Huynh-Thu and Geurts, 2018), SINCERITIES (Gao et al., 2018), SCODE (Matsumoto et al., 2017), graphical lasso (Glasso), and NGM-[SF]^{2}M (Tong et al., 2024). All models are trained across five random seeds.

We show the quantitative results for this experiment and evaluation in Figure 3 and visualizations of the inferred structures in Figure 4 (top). We observe that StructureFlow exhibits higher consistency as a top performing method for inferring the underlying structure across all systems and both metrics. In comparison, certain baseline methods at best show favorable performance in specific systems and on specific metrics, but struggle to consistently perform across the board. For instance, on structure learning (Figure 3) RF matches StructureFlow on BF, SW, TF, and HSC but drops on the cyclical (CY) and long-linear (LL) systems (AP $\approx$ 0.32–0.42 vs. $\approx$ 0.70–0.80), reflecting RF's single global linear OU matrix versus StructureFlow's nonlinear NGM drift.

This structure-level competitiveness does not carry over to the concurrent task: on dynamical inference (Table 1) RF is consistently worse than StructureFlow on every system ($W_2 \approx$ 0.95–2.1 vs. $\approx$ 0.58–0.84), as a linear drift cannot capture the nonlinear stochastic population dynamics. RF is computationally cheap and effective when a global linear drift suffices, but StructureFlow remains a generalizable structure-learning method that also models the nonlinear, stochastic population dynamics that RF cannot.

StructureFlow effectively infers dynamics across left-out time-points in simulated biological systems. To evaluate dynamical inference performance, we consider a leave-one-out cross-validation experiment across discrete timepoints, where we iteratively exclude each timepoint t_k for $k \in \{1, \ldots, T-2\}$ during model training, then simulate via ODE from the ground truth data at t_{k-1} to t_k. We use the Wasserstein-2 distance (W_2) and maximum mean discrepancy (MMD) to evaluate the distributional distances between the predicted and ground truth distributions at t_k.

We consider RF, OTVelo, and TIGON as baselines since they are designed to address both inference tasks. We additionally include a comparison to [SF]^{2}M as a baseline method for dynamical inference, but note that [SF]^{2}M is not capable of jointly inferring the network structure. We show quantitative results for this experiment in Table 1 and visualization of the predictions in Figure 4a and Figure 4b. StructureFlow achieves the best W_2 among methods that perform the joint task across the six systems, while TIGON achieves the lowest MMD (with StructureFlow second on MMD).

Ablations of core design choices. Two choices central to StructureFlow are the group lasso regularization on the structural (FC1) layer and the weight α balancing the stochastic score and deterministic flow components. On the TF system, group lasso improves structure recovery up to an intermediate strength (AP $0.80 \rightarrow 0.92$, AUROC $0.94 \rightarrow 0.97$ at weight 0.04) before over-sparsification degrades it, and the score/flow

[5] In this work, we do not threshold the learned graphs/structures. Instead, we use AP/AUROC metrics to evaluate structure learning performance across all possible thresholds. We leave extensions for learning thresholded structures for future work.

Table 2: **Unbalanced StructureFlow**

	AUROC	AP	W_2	MMD
TIGON	0.540 ± 0.014	0.427 ± 0.047	2.355 ± 0.591	0.038 ± 0.041
StructureFlow	0.735 ± 0.005	0.358 ± 0.006	1.672 ± 0.532	0.029 ± 0.031
U-StructureFlow	**0.838 ± 0.010**	**0.509 ± 0.025**	**1.670 ± 0.438**	**0.027 ± 0.033**

balance is best at $\alpha \in [0.1, 0.5]$ with a drop at higher score weight ($\alpha = 0.8$). Full sweeps (including model capacity) are reported in Appendix E.6, Tables 11, 12, and 13.

StructureFlow infers structure and dynamics in the unbalanced setting. We show a use case of StructureFlow in the unbalanced setting. To extend StructureFlow to the unbalanced setting (**U-StructureFlow**), we replace the balanced Sinkhorn algorithm in our EOT procedure with the unbalanced Sinkhorn divergence and incorporate a growth rate field to model proliferation-induced density shifts. We evaluate our method on a growth-based bifurcation system. Following the setup of Zhang et al. (2025), we use the underlying gene regulatory network and boolean logic of the standard BoolODE bifurcation dataset, but we impose lineage specific proliferation rates where one of the attractors has a high proliferation rate and the other has a near-zero growth rate. We observe unbalanced-StructureFlow yields the best performance for the joint task (Table 2). We discuss further details in Appendix B.3.

5.3 Experiments on Real Data (Renge) System

We consider an interventional time-series dataset of human induced pluripotent stem cells (iPSC) introduced by (Ishikawa et al., 2023). Interventions are conducted via clustered regularly interspaced short palindromic repeats (CRISPR) knockouts on 23 transcription factors (TFs). Cell states are then sequenced via single-cell RNA-seq across 4 time bins for each CRISPR intervention. We process the dataset following the procedure of Zhang (2024) and select the 8 interventions with greatest observed change in population-level gene expression; see Appendix A.8.

StructureFlow recovers underlying structure of gene regulatory networks competitively to baselines. We evaluate StructureFlow on its ability to recover the 103×103 directed network of inferred TF-TF interactions from the processed real data. We remark that a central challenge in evaluating structure learning performance on real biological systems is that there rarely exists a definitively known ground truth network. In this setting, we have access to an approximation of the ground-truth network for 18 genes, determined using a ChIP-seq reference; see Appendix E.9 for details. Accordingly, AP and AUROC measure agreement with this partial reference and should not be interpreted as a complete validation of the full regulatory graph. Models are trained using all 103 genes, but evaluation is done over 18×103 *ground-truth* networks. We show results in Figure 5 and observe that StructureFlow yields competitive performance to counterpart baselines. Because performance varies across random seeds, we avoid interpreting the best seed as a definitive ranking; instead, these results indicate that StructureFlow is competitive with the strongest baselines under the available partial-reference benchmark. We include additional results in Appendix E.

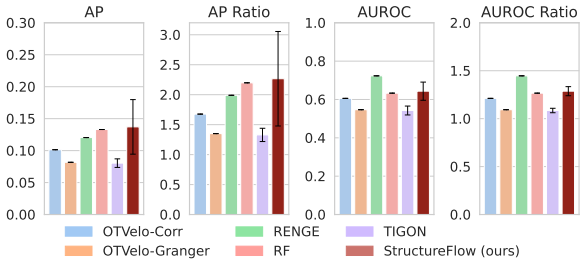


Figure 5: **StructureFlow achieves competitive benchmark agreement for the *structure learning* tasks in the real-data (Renge) system.** We use the AP/AUROC ratio to denote structure recovery performance w.r.t. a random predictor.

StructureFlow effectively models cell trajectories and improves prediction of left-out interventions (knockouts) in real-data setting. We report results for dynamical inference over left out time-points and left-out knockouts on the real biological system in Table 3 and Table 4, respectively. In the left-out intervention (knockout) setting, we withhold all time marginals $t_k, k = [0, ..., T-1]$ for a given inter-

ventional condition during training, then evaluate performance on predicting the final timepoint. Similar to the simulated biological systems, we use W_2 and ED to evaluate performance. OTVelo and TIGON do not model interventions, and thus are not included in the left-out intervention evaluation. For left-out-time point dynamical inference (Table 3), we observe that StructureFlow achieves the best W_2 among evaluated methods and yields competitive results on ED. StructureFlow is the best performing joint method on the left-out intervention task, and although it shows improved generalization in this setting, implying some plausibly improved learning of underlying mechanisms, we make no claim that StructureFlow truly models the *causal* dependencies of the underlying data generative process. The inferred networks and intervention responses should therefore be treated as model-based hypotheses requiring biological validation. We include an equivalent experiment for the simulated biological systems in Appendix E Table 8.

Table 3: **StructureFlow shows competitive performance for *dynamical inference* of left-out marginals on the real (Renge) biological dataset.** We show a comparison of dynamical inference methods for learning conditional population dynamics across left-out timepoints $k = 1, 2$ on the Renge dataset. We report results over withheld k as well as the average of the two using Wasserstein-2 (W_2) and Energy Distance (ED) (Rizzo and Székely, 2016) metrics. Bold indicates **1st best performer** and the underline indicates 2nd best performer for models that perform the joint inference task.

	Timepoint 1		Timepoint 2		Average	
	W_2	ED	W_2	ED	W_2	ED
[SF]^{2}M	5.673 ± 0.034	0.533 ± 0.024	5.841 ± 0.030	0.524 ± 0.024	5.757 ± 0.032	0.528 ± 0.024
RF	10.193 ± 0.004	12.389 ± 0.009	9.242 ± 0.004	10.089 ± 0.008	9.717 ± 0.004	11.239 ± 0.008
OTVelo	7.106 ± 0.000	4.269 ± 0.000	6.996 ± 0.000	3.658 ± 0.000	7.051 ± 0.000	3.963 ± 0.000
TIGON	6.401 ± 0.033	**0.173 ± 0.011**	6.402 ± 0.002	**0.125 ± 0.001**	6.402 ± 0.001	**0.149 ± 0.024**
StructureFlow	**5.589 ± 0.026**	0.692 ± 0.048	**5.679 ± 0.029**	0.765 ± 0.025	**5.634 ± 0.027**	0.729 ± 0.037

Table 4: **StructureFlow shows competitive performance for *dynamical inference* of left-out interventions on the real (Renge) biological dataset.** We show a comparison of dynamical inference methods for learning conditional population dynamics on the Renge dataset for left-out interventions. We report results for 3 left-out gene knockout conditions using Wasserstein-2 (W_2) and Energy Distance (ED) metrics, as well as the average performance across conditions. **Bold** indicates the best performing method that can perform the joint task. We provide a visualization of this result in Appendix E fig. 12.

	NANOG		POU5F1		PRDM14		Average	
	W_2	ED	W_2	ED	W_2	ED	W_2	ED
[SF]^{2}M	5.726 ± 0.015	0.467 ± 0.014	5.931 ± 0.024	0.826 ± 0.020	6.002 ± 0.043	0.639 ± 0.030	5.886 ± 0.027	0.644 ± 0.021
RF	9.859 ± 0.002	11.492 ± 0.006	10.211 ± 0.002	12.140 ± 0.004	9.889 ± 0.005	11.461 ± 0.013	9.986 ± 0.003	11.698 ± 0.008
StructureFlow	**5.517 ± 0.041**	**0.818 ± 0.015**	**5.717 ± 0.021**	**1.136 ± 0.039**	**5.816 ± 0.023**	**0.973 ± 0.053**	**5.683 ± 0.028**	**0.976 ± 0.036**

6 Conclusion

In this work, we introduced StructureFlow, a novel simulation-free training method tailored for jointly learning the underlying network structure and the conditional population dynamics of high-dimensional stochastic systems. Our work highlights the benefits of addressing the joint inference task with a single model trained via score-and-flow matching and tailored for the problem — i.e. improved performance on both tasks. We demonstrate this through a suite of empirical experiments and show that StructureFlow exhibits improved performance on several structure-learning and W_2 dynamical-inference metrics compared to existing approaches, which either tackle the individual tasks in isolation and/or are limited in their ability to address both problems. Moreover, we showcase the application of our method on a challenging real system for recovering network structure and simultaneously predicting the system response of left-out (*unseen*) interventions.

Limitations & future work. In this work, we focused on the problem of joint structure learning and dynamical inference for stochastic population dynamics. With this come non-standard challenges, e.g. evaluating structure learning performance requires knowledge of the ground truth directed graphs which define

the data generative process of the dynamical system. To address this, we considered a suite of synthetic and biologically meaningful simulated systems (Pratapa et al., 2020), such that both tasks can be evaluated in unison, which emulate real systems. However, these simulated systems may be limited in their ability to fully reflect system behavior in real settings. For example, our simulated knockout conditions may not exactly reflect realistic CRISPR effects, such as imperfect knockouts. We believe exploring settings with imperfect intervention is a fruitful direction for future work. Sensitivity analyses over the assumed diffusion scale σ, the number of timepoints, and the number of samples per timepoint are similarly valuable directions. More generally, inferred networks and predicted perturbation responses should be used as hypotheses for follow-up experiments rather than as clinical or therapeutic evidence without independent validation.

In a similar vein, evaluating structure learning performance in real-data settings requires experimental validation (which was accomplished by Ishikawa et al. (2023) to produce what we label as the Renge dataset. Such experimental validation is costly to acquire and unrealistic in large systems. Thus, this experimental validation is limited to a small set of system variables, and is still at best an estimate of the ground truth network. Stronger validation remains an open challenge: this would require larger or orthogonal evidence sources, such as additional chromatin accessibility data or curated regulatory databases. This would require substantial new data generation or curation and is outside the scope of the present work. Another limitation is that StructureFlow outputs unsigned edge scores. If signed interactions are needed, a sign can be estimated from the corresponding entry of the Jacobian of the learned drift, e.g. as done in Qiu et al. (2022). Exploring other approaches to directly infer signed edges is a potential future direction. We also do not consider certain aspects of physical processes that are prevalent in natural systems, such as modeling population dynamics while considering interacting particles (Atanackovic et al., 2024) and/or structure learning of non-stationary networks (Lu et al., 2025), but view these directions as natural extensions for StructureFlow. Incorporating biological priors (either through the choice of reference process (Petrović et al., 2025; Zhang and Stumpf, 2025) or by directly imposing knowledge of structural relationships on the network (Roohani et al., 2024)) is another interesting direction for future work. Lastly, using optimal couplings acquired using marginals across all timepoints is a directly applicable extension to StructureFlow, and may help improve overall performance of both structure learning and dynamical inference (Chen et al., 2023; Theodoropoulos et al., 2025; Rohbeck et al., 2025).

We remark, StructureFlow is designed to be built upon and extended to all aforementioned settings. In this work, we provide a methodological and experimental foundation amenable to further development.

Reproducibility Statement

To facilitate the reproducibility of our results and findings, the source code is released anonymously at `https://anonymous.4open.science/r/StructureFlow/`. We also provide the necessary mathematical preliminaries, background, and introduction to our methods in Section 2 and 3, respectively. We further outline experimental details in Appendix A, where we expand on the mathematical details of NGM and RF, provide the experimental details we used to train StructureFlow, and clarify our data setups across synthetic (linear), BoolODE, and single-cell CRISPR perturbation settings. In Appendix B, we provide details on our implementation, including Algorithm 1, which outlines the core StructureFlow training loop.

References

Atanackovic, L., Tong, A., Wang, B., Lee, L. J., Bengio, Y., and Hartford, J. S. (2023). Dyngfn: Towards bayesian inference of gene regulatory networks with gflownets. *Advances in Neural Information Processing Systems*, 36:74410–74428.

Atanackovic, L., Zhang, X., Amos, B., Blanchette, M., Lee, L. J., Bengio, Y., Tong, A., and Neklyudov, K. (2024). Meta flow matching: Integrating vector fields on the wasserstein manifold. *arXiv preprint arXiv:2408.14608*.

Bellot, A., Branson, K., and van der Schaar, M. (2022). Neural graphical modelling in continuous-time: Consistency guarantees and algorithms. *International Conference on Learning Representations*.

Binnewies, M., Roberts, E. W., Kersten, K., Chan, V., Fearon, D. F., Merad, M., Coussens, L. M., Gabrilovich, D. I., Ostrand-Rosenberg, S., Hedrick, C. C., Vonderheide, R. H., Pittet, M. J., Jain, R. K., Zou, W., Howcroft, T. K., Woodhouse, E. C., Weinberg, R. A., and Krummel, M. F. (2018). Understanding the tumor immune microenvironment (time) for effective therapy. *Nature Medicine*, 24(5):541550.

Boninsegna, L., Nüske, F., and Clementi, C. (2018). Sparse learning of stochastic dynamical equations. *The Journal of Chemical Physics*, 148(24):241723.

Bortoli, V. D., Thornton, J., Heng, J., and Doucet, A. (2021). Diffusion schrödinger bridge with applications to score-based generative modeling. In *Neural Information Processing Systems (NeurIPS)*.

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. (2020). Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33:21865–21877.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937.

Bunne, C., Hsieh, Y.-P., Cuturi, M., and Krause, A. (2022). The Schrödinger bridge between gaussian measures has a closed form. *arXiv preprint 2202.05722*.

Bunne, C., Stark, S. G., Gut, G., Del Castillo, J. S., Levesque, M., Lehmann, K.-V., Pelkmans, L., Krause, A., and Rätsch, G. (2023). Learning single-cell perturbation responses using neural optimal transport. *Nature Methods*, 20(11):1759–1768.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31.

Chen, T., Liu, G.-H., Tao, M., and Theodorou, E. A. (2023). Deep momentum multi-marginal schrödinger bridge.

Chen, T., Liu, G.-H., and Theodorou, E. A. (2022). Likelihood training of Schrödinger bridge using forward-backward SDEs theory. *International Conference on Learning Representations (ICLR)*.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. (2019). Unbalanced optimal transport: Dynamic and kantorovich formulation.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Neural Information Processing Systems (NIPS)*.

Dakhmouche, R., Lunati, I., and Gorji, H. (2025). Robust symbolic regression for dynamical system identification. *Transactions on Machine Learning Research*.

Dinh, V. C. and Ho, L. S. (2020). Consistent feature selection for analytic deep neural networks. *Advances in Neural Information Processing Systems*, 33:2420–2431.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., et al. (2016). Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *cell*, 167(7):1853–1866.

Elowitz, M. B., Levine, A. J., Siggia, E. D., and Swain, P. S. (2002). Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186.

Erdös, P. and Rényi, A. (1959). On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297.

Etesami, J. and Kiyavash, N. (2014). Directed information graphs: A generalization of linear dynamical graphs. In *2014 American Control Conference*, pages 2563–2568. IEEE.

Etesami, J. and Kiyavash, N. (2016). Measuring causal relationships in dynamical systems through recovery of functional dependencies. *IEEE Transactions on Signal and Information Processing over Networks*, 3(4):650–659.

Eyring, L., Klein, D., Uscidda, T., Palla, G., Kilbertus, N., Akata, Z., and Theis, F. (2024). Unbalancedness in neural monge maps improves unpaired domain translation.

Föllmer, H. (1988). Random fields and diffusion processes. *École d'Été de Probabilités de Saint-Flour*.

Gao, N. P., Ud-Dean, S. M. M., Gandrillon, O., and Gunawan, R. (2018). Sincerities: Inferring gene regulatory networks from time-stamped single cell transcriptional expression profiles. *Bioinformatics*, 34(2):258–266.

Gulati, G. S., Sikandar, S. S., Wesche, D. J., Manjunath, A., Bharadwaj, A., Berger, M. J., Ilagan, F., Kuo, A. H., Hsieh, R. W., Cai, S., Zabala, M., Scheeren, F. A., Lobo, N. A., Qian, D., Yu, F. B., Dirbas, F. M., Clarke, M. F., and Newman, A. M. (2020). Single-cell transcriptional diversity is a hallmark of developmental potential. *Science*, 367(6476):405411.

Hashimoto, T. B., Gifford, D. K., and Jaakkola, T. S. (2016). Learning population-level diffusions with generative recurrent networks. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2417–2426.

Huang, K., Lopez, R., Hütter, J.-C., Kudo, T., Rios, A., and Regev, A. (2024). Sequential optimal experimental design of perturbation screens guided by multi-modal priors. In *International Conference on Research in Computational Molecular Biology*, pages 17–37. Springer.

Huynh-Thu, V. A. and Geurts, P. (2018). dyngenie3: Dynamical genie3 for the inference of gene networks from time series expression data. *Scientific Reports*, 8(1):3384.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010a). Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):e12776.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010b). Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):e12776.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709.

Ishikawa, M., Sugino, S., Masuda, Y., et al. (2023). Renge infers gene regulatory networks using time-series single-cell rna-seq data with crispr perturbations. *Communications Biology*, 6:1290.

Ji, Y., Lotfollahi, M., Wolf, F. A., and Theis, F. J. (2021). Machine learning for perturbational single-cell omics. *Cell Systems*, 12(6):522–537.

Jiang, J. et al. (2025). D-spin constructs gene regulatory network models from multiplexed scrna-seq data revealing organizing principles of cellular perturbation response. *bioRxiv*.

Jinek, M., Chylinski, K., Fonfara, I., Hauer, M., Doudna, J. A., and Charpentier, E. (2012). A programmable dual-rna–guided dna endonuclease in adaptive bacterial immunity. *Science*, 337(6096):816–821.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems.

Klein, A. M. et al. (2015). Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201.

Lin, Z., Chang, S., Zweig, A., Kang, M., Azizi, E., and Knowles, D. A. (2025). Interpretable neural odes for gene regulatory network discovery under perturbations. *arXiv preprint arXiv:2501.02409*.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023a). Flow matching for generative modeling.

Lipman, Y. et al. (2023b). Flow matching for generative modeling. *arXiv:2210.02747v2*.

Lorch, L. et al. (2026). Latent causal diffusions for single-cell perturbation modeling. *arXiv preprint*.

Lu, Z., Zhang, W., Le, T., Wang, H., Sümbül, U., SheaBrown, E. T., and Mi, L. (2025). Netformer: An interpretable model for recovering dynamical connectivity in neuronal population dynamics. In *The Thirteenth International Conference on Learning Representations*.

Léonard, C. (2014). A survey of the schrödinger problem and some of its connections with optimal transport. *Discrete and Continuous Dynamical Systems*, 34(4):1533–1574.

MacArthur, B. D. (2022). The geometry of cell fate. *Cell Systems*, 13(1):1–3.

Macosko, E. Z. et al. (2015). Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214.

Maoutsa, D., Reich, S., and Opper, M. (2020). Interacting particle solutions of fokkerplanck equations through gradientlogdensity estimation. *Entropy*, 22(8):802.

Matsumoto, H., Kiryu, H., Furusawa, C., Ko, M. S. H., Ko, S. B. H., Gouda, N., Hayashi, T., and Nikaido, I. (2017). Scode: an efficient regulatory network inference algorithm from single-cell rna-seq during differentiation. *Bioinformatics*, 33(15):2314–2321.

Molè, M. A., Coorens, T. H. H., Shahbazi, M. N., Weberling, A., Weatherbee, B. A. T., Gantner, C. W., Sancho-Serra, C., Richardson, L., Drinkwater, A., Syed, N., Engley, S., Snell, P., Christie, L., Elder, K., Campbell, A., Fishel, S., Behjati, S., Vento-Tormo, R., and Zernicka-Goetz, M. (2021). A single cell characterisation of human embryogenesis identifies pluripotency transitions and putative anterior hypoblast centre. *Nature Communications*, 12(1).

Moscardó García, M., Aalto, A., Montanari, A. N., et al. (2025). Multi-omic network inference from time-series data. *npj Systems Biology and Applications*, 11(114).

Neklyudov, K., Severo, D., and Makhzani, A. (2022). Action matching: A variational method for learning stochastic dynamics from samples.

Norman, T. M., Horlbeck, M. A., Replogle, J. M., Ge, A. Y., Xu, A., Jost, M., Gilbert, L. A., and Weissman, J. S. (2019). Exploring genetic interaction manifolds constructed from rich single-cell phenotypes. *Science*, 365(6455):786–793.

Oki, S., Ohta, T., Shioi, G., Hatanaka, H., Ogasawara, O., Okuda, Y., Kawaji, H., Nakaki, R., Sese, J., and Meno, C. (2018). Chip-atlas: a data-mining suite powered by full integration of public chip-seq data. *EMBO Reports*, 19(12):e46255.

Peidli, S., Green, T. D., Shen, C., Gross, T., Min, J., Garda, S., Yuan, B., Schumacher, L. J., Taylor-King, J. P., Marks, D. S., et al. (2024). scperturb: harmonized single-cell perturbation data. *Nature Methods*, pages 1–10.

Petrović, K., Atanackovic, L., Moro, V., Kapuśniak, K., Ceylan, I. I., Bronstein, M., Bose, A. J., and Tong, A. (2025). Curly flow matching for learning non-gradient field dynamics. *arXiv preprint arXiv:2510.26645*.

Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A., and Murali, T. (2020). Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods*, 17(2):147–154.

Qiu, X., Zhang, Y., Martin-Rufino, J. D., Weng, C., Hosseinzadeh, S., Yang, D., Pogson, A. N., Hein, M. Y., Min, K. H. J., Wang, L., Grody, E. I., Shurtleff, M. J., Yuan, R., Xu, S., Ma, Y., Replogle, J. M., Lander, E. S., Darmanis, S., Bahar, I., Sankaran, V. G., and Weissman, J. S. (2022). Mapping transcriptomic vector fields of single cells. *Cell*, 185(4):690–711.e45.

Rizvi, A. H., Camara, P. G., Kandror, E. K., Roberts, T. J., Schieren, I., Maniatis, T., and Rabadan, R. (2017). Single-cell topological rna-seq analysis reveals insights into cellular differentiation and development. *Nature Biotechnology*, 35(6):551560.

Rizzo, M. L. and Székely, G. J. (2016). Energy distance. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(1):27–38.

Rohbeck, M., Bunne, C., Brouwer, E. D., Huetter, J.-C., Biton, A., Chen, K. Y., Regev, A., and Lopez, R. (2025). Modeling complex system dynamics with flow matching across time and conditions. In *The Thirteenth International Conference on Learning Representations*.

Roohani, Y., Huang, K., and Leskovec, J. (2024). Predicting transcriptional outcomes of novel multigene perturbations with gears. *Nature Biotechnology*, 42(6):927–935.

Runge, J., Nowack, P., Kretschmer, M., Flaxman, S., and Sejdinovic, D. (2019). Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996.

Schiebinger, G., Shu, J., Tabaka, M., et al. (2019). Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943.e22.

Sha, Y., Qiu, Y., Zhou, P., Nguyen, H. H., Xie, S., Zou, J., Lee, D., Gehring, J., Alizadeh, A., Tibshirani, R., Pritchard, J. K., and Zou, E. (2024). Reconstructing growth and dynamic trajectories from single-cell transcriptomics data. *Nature Machine Intelligence*, 6:25–39.

Shi, Y., De Bortoli, V., Campbell, A., and Doucet, A. (2023). Diffusion Schrödinger bridge matching. *arXiv preprint 2303.16852*.

Shrivastava, H. and Chajewska, U. (2023). Neural graphical models. In *European Conference on Symbolic and Quantitative Approaches with Uncertainty*, pages 284–307. Springer.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations.

Sun, F., Liu, Y., Wang, J.-X., and Sun, H. (2023). Symbolic physics learner: Discovering governing equations via monte carlo tree search.

Theodoropoulos, P., Saravanos, A. D., Theodorou, E. A., and Liu, G.-H. (2025). Momentum multi-marginal schrödinger bridge matching.

Tong, A. et al. (2024). Simulation-free schrödinger bridges via score and flow matching. *International Conference on Artificial Intelligence and Statistics*.

Tong, A., Huang, J., Wolf, G., van Dijk, D., and Krishnaswamy, S. (2020). Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics.

Wang, B., Jennings, J., and Gong, W. (2023). Neural structure learning with stochastic differential equations. *arXiv preprint arXiv:2311.03309*.

Wang, D., Jiang, Y., Zhang, Z., Gu, X., Zhou, P., and Sun, J. (2025). Joint velocity-growth flow matching for single-cell dynamics modeling.

Wang, S., Al-Radhawi, M. A., Lauffenburger, D. A., and Sontag, E. D. (2024). Recovering biomolecular network dynamics from single-cell omics data requires three time points. *NPJ Systems Biology and Applications*, 10(1):97.

Weinreb, C., Wolock, S., Tusi, B. K., Socolovsky, M., and Klein, A. M. (2018). Fundamental limits on dynamic inference from single-cell snapshots. 115(10):E2467–E2476.

Zhang, J., Cammarata, L., Squires, C., Sapsis, T. P., and Uhler, C. (2023). Active learning for optimal intervention design in causal models. *Nature Machine Intelligence*, 5(10):1066–1075.

Zhang, S., Maddu, S., Qiu, X., and Chardès, V. (2025). Inferring stochastic dynamics with growth from cross-sectional data.

Zhang, S. Y. (2024). Joint trajectory and network inference via reference fitting. *Proceedings of the 19th Machine Learning in Computational Biology meeting*.

Zhang, S. Y. and Stumpf, M. P. (2025). Learning non-equilibrium diffusions with schr\" odinger bridges: from exactly solvable to simulation-free. *arXiv preprint arXiv:2505.16644*.

Zhao, W., Larschan, E., Sandstede, B., and Singh, R. (2024). Optimal transport reveals dynamic gene regulatory networks via gene velocity estimation. *bioRxiv*. [Preprint].

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31.

A Experimental Details

A.1 Neural Graphical Models for Dynamical Systems

Unlike static graphical models, where variables are assumed to be independent and identically distributed, dynamical systems require modeling dependencies in continuous time.

Neural graphical models (NGMs) parameterize the vector field $\boldsymbol{v}(\boldsymbol{x}(t))$ using deep neural networks (Bellot et al., 2022). Specifically, a neural dynamic structural model (NDSM) is defined as:

$$dx_j(t) \;=\; v_j^\theta(\boldsymbol{x}(t))\,dt \;+\; dw_j(t), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0$$

where $v_j^\theta \in \mathcal{F}$ are analytic functions parameterized by neural networks with trainable weights $\theta \in \Theta$. Each function v_j^θ can be expressed as:

$$v_j^\theta(\boldsymbol{x}) \;=\; \psi\Big(\psi\big(\cdots\psi(\,\boldsymbol{x}\,\theta_j^A)\,\theta_j^1\cdots\big)\,\theta_j^K\Big),$$

where $\theta_j^A \in \mathbb{R}^{d\times h}$ is the first (graph) layer, $\theta_j^k \in \mathbb{R}^{h\times h}$ for $k = 1,\ldots,K-1$ and $\theta_j^K \in \mathbb{R}^{h\times 1}$ are the deeper layers, and ψ is an activation function. Directed edges in the inferred graph correspond to non-zero partial derivatives of these networks, and indicate model-based direct structural dependencies under the NGM assumptions between variables. The adjacency can be inferred if:

$$G_{ji} \neq 0 \quad\Longleftrightarrow\quad \big\|\partial_i\, v_j^\theta\big\|_{L_2} > 0$$

Now to recover G, structure learning can be formulated as a penalized optimization problem:

$$\arg\min_{v^\theta}\; R_n(\theta) \;=\; \min_{v^\theta}\; \frac{1}{n}\sum_{m=1}^{n}\big\|\boldsymbol{x}(t_m) - \hat{\boldsymbol{x}}(t_m)\big\|_2^2,$$

$$\text{subject to}\;\; d\boldsymbol{x}(t) = \boldsymbol{v}^\theta(\boldsymbol{x}(t))\,dt \quad\text{and}\quad \rho(\theta) \leq \eta,$$

where $\rho(\theta)$ is a regularization term. In (Bellot et al., 2022) this is often implemented using group lasso (GL) or adaptive group lasso (AGL):

$$\rho_{\text{GL}}(\theta) \;=\; \lambda_{\text{GL}}\sum_{j=1}^{d}\sum_{i=1}^{d}\big\|\big(\theta_j^A\big)_{i,:}\big\|_2, \qquad \rho_{\text{AGL}}(\theta) \;=\; \lambda_{\text{AGL}}\sum_{j=1}^{d}\sum_{i=1}^{d}\frac{\big\|\big(\theta_j^A\big)_{i,:}\big\|_2}{\big\|\big(\hat{\theta}_j^A\big)_{i,:}\big\|_2^\gamma}.$$

Here, $\lambda_{\text{AGL}}, \lambda_{\text{GL}}$ control the regularization strength, and $\big(\hat{\theta}_j^A\big)_{i,:}$ is an initial estimate of the parameters.

Given an estimate of $\boldsymbol{v}^\theta$, NGMs can be extended to irregularly sampled and non-linear time series data by numerically computing the forward trajectory with an ODE solver:

$$\hat{\boldsymbol{x}}(t_1), \hat{\boldsymbol{x}}(t_2), \ldots, \hat{\boldsymbol{x}}(t_n) \;=\; \text{ODESolve}\big(\boldsymbol{v}^\theta,\, \boldsymbol{x}(t_0),\, t_1, t_2, \ldots, t_n\big)$$

enforcing the constraint $d\boldsymbol{x}(t) = \boldsymbol{v}^\theta(\boldsymbol{x}(t))\,dt$ while optimizing for $R_n(\theta)$ (Chen et al., 2018). While the NGM was originally developed to identify causal dependencies in deterministic systems governed by ODEs, our work extends their applicability to stochastic systems by using the probability flow formulation of the underlying SDE. We use this terminology in the technical sense of the cited NGM framework, and do not claim general causal identifiability from population snapshots. Differently from Bellot et al. (2022), we train an NGM which parameterizes the probability-flow ODE, whose instantaneous drift is

$$\hat{\boldsymbol{v}}_t(\boldsymbol{x};\Theta) \;=\; \boldsymbol{v}^\theta(\boldsymbol{x}) \;-\; \frac{\sigma^2}{2}\,\boldsymbol{s}_t^\phi(\boldsymbol{x}),$$

which is equivalent in marginal behavior to the original SDE but removes the stochasticity by incorporating the score function $\nabla \log p_t(\boldsymbol{x})$. This allows us to retain the NGM for structure discovery, while still modeling data generated from inherently noisy dynamics.

A.2 Reference Fitting (RF)

In contrast to StructureFlow, Reference Fitting (Zhang, 2024) (RF) starts from the principle that the inferred couplings π should minimize entropy with respect to a reference process, with the joint-optimization problem defined as:

$$\min_{A\in\mathbb{R}^{d\times d},b\in\mathbb{R}^d}\ \min_{\pi\in\mathcal{C}(\mu,\mu')}\ \sigma^2\mathrm{KL}\left(\pi|K^{\sigma}_{(A,b)}\right)+\mathcal{R}\left(A,b\right)$$

where $\mathcal{R}$ is a regularizer term, A and b are the linear and constant terms, respectively, of an Ornstein–Uhlenbeck (OU) process of the form:

$$\mathrm{d}X_t=\left(AX_t+b\right)\mathrm{d}t\ \ +\sigma\mathrm{d}B_t$$

and $K^{\sigma}_{(A,b)}$ is the transition kernel of the OU process.
Perturbations can then be modeled in this set-up by the linear interaction matrix $A^{(g)}$, where g represents a knocked-out gene, and thus the gth is zeroed out. Thus, for any gene intervention (knockout) g, the reference OU process takes the form:

$$\mathrm{d}X_t=\left(A\odot M^{(g)}\right)X^{(g)}_t\mathrm{d}t\ \ +\sigma\mathrm{d}B_t$$

where $M^{(g)}_{ij}=\mathbf{1}_{i\neq g}$ is the masking matrix for intervention g. This allows the RF process to learn interactions which may have very low signal in the observational (wild-type) data, and thus cannot be learned from the observed dynamics alone. The transition kernel, $K^{\sigma}_{(A,b)}$, is then approximated by separating the drift and noise terms to model the kernel's mean and covariance, respectively:

$$\mu_t=e^{tA}x_0,\ \Sigma_t=\sigma^2tI$$

This yields a transition kernel of the form:

$$K_t(x,x')\propto\exp\left(-\frac{||e^{tA}x-x'||_2^2}{2\sigma^2t}\right)$$

Running an alternating optimization over the couplings π and reference dynamics yields both the reference process itself and couplings which correspond to this process.

Trajectory fitting (ODE view). Given v^{θ}, irregularly sampled trajectories can be obtained by numerical integration,

$$\hat{\boldsymbol{x}}(t_1),\ldots,\hat{\boldsymbol{x}}(t_n)\ =\ \mathrm{ODESolve}\big(v^{\theta},\,\boldsymbol{x}(t_0),\,t_1,\ldots,t_n\big),$$

and one may minimize a data misfit $R_n(\theta)=\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}(t_i)-\hat{\boldsymbol{x}}(t_i)\|_2^2$ with the above sparsity penalty (Chen et al., 2018; Bellot et al., 2022).

Stochastic dynamics via probability flow. While NGMs were introduced for deterministic ODEs, we extend them to stochastic systems by parameterizing the *probability-flow ODE* associated with an SDE. Let the instantaneous probability-flow drift be

$$\hat{\boldsymbol{v}}_t(\boldsymbol{x};\Theta)\ =\ \boldsymbol{v}^{\theta}(\boldsymbol{x})\ -\ \tfrac{\sigma^2}{2}\,\boldsymbol{s}^{\phi}_t(\boldsymbol{x}),$$

where $\Theta=\{\theta,\phi\}$, $\boldsymbol{v}^{\theta}$ is the NGM (autonomous) drift defined above, and $\boldsymbol{s}^{\phi}_t(\boldsymbol{x})=\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})$ is a score network. This PF-ODE induces the same marginals p_t as the original SDE but removes stochasticity by absorbing it into the score term. We therefore retain the NGMs FC1 layer θ^A for *structure discovery* (via GL/AGL on $A_{ji}=\|(\theta^A_j)_{i,:}\|_2$) while modeling data generated by noisy dynamics. This connects structure extraction to the PF-ODE used for rollout, while the training losses remain simulation-free.

A.3 On Theoretical Guarantees for Structure Recovery from Population Dynamics via NGMs

Edge Recovery in the NGM and StructureFlow. Although our vector field is parameterized by a neural network, using a sparse weight matrix as the first layer of the network provides a principled approach to encode, and likewise to recover, the underlying interaction structure of the non-linear system (Bellot et al., 2022). For continuous-time dynamics Bellot et al. (2022) denote directed edge(s) $x_k \to x_j$ whenever the drift f_j has a **functional dependence** on x_k, i.e. $\partial_k f_j \neq 0$. They show that for any analytic neural network parameterization of f_j, every true functional dependency must pass through the input layer and conversely, if $\partial_k f_j = 0$, there exists an equivalent parameterization where the entire k-th column of the first-layer weight matrix is zero. This justifies the application group lasso solely to the first-layer weights, because zeros in that layer correspond exactly to absent edges in the underlying continuous system. Bellot et al. (2022) show that with enough data, adaptive group lasso recovers the correct sparsity pattern. StructureFlow relies on the same principle because our autonomous vector field makes use of the same formulation of the NGM.

On identifiability of underlying structure via the NGM and flow matching. In general, the question of identifiability for directed structure is challenging, even in the simplest possible case of linear dynamics (see e.g. Wang et al. (2024)). Similarly, Zhang (2024) were not able to provide an identifiability result but only show well-posedness of their optimization problem.

To our knowledge, identifiability in the joint trajectory inference and structure learning problem from population snapshots is an open problem and is particularly challenging due to the interdependence of the learned structure. We thus leave a general theoretical study of identifiability to future work. In what follows, we reason that the NGM with group lasso recovers the true underlying structure in the idealized setting where the drift function and score can be reconstructed exactly. We consider autonomous dynamics, where the drift field v is given by a NGM model with parameters θ_0

$$dX_t = v(X_t; \theta_0)dt + \sigma dB_t.$$

Equivalently, this can be written as a probability flow, with $u_t(x) = v(x) - D\nabla \log p_t$ and $D = \frac{1}{2}\sigma\sigma^\top$:

$$\partial_t p_t = -\nabla \cdot (p_t(x)u_t(x)).$$

Consider the case where samples of the flow $u_t(x)$ and score $\nabla \log p_t(x)$ can be accessed exactly, where $(t, x) \sim p_t(x)$. Writing $\hat{v}_t(x; \theta)$ to be the reconstructed vector field, parameterized as a NGM with parameter θ. The corresponding probability flow is

$$\hat{u}_t(x) = \hat{v}(x) - D\nabla \log p_t.$$

The flow matching objective with target $u_t(x)$ is thus

$$\min_\theta \mathbb{E}_{t\sim[0,1],x\sim p_t} \|\hat{u}_t(x) - u_t(x)\|_2^2 \Longrightarrow \min_\theta \mathbb{E}_t \mathbb{E}_{x\sim p_t} \|\hat{v}(x; \theta) - v(x; \theta_0)\|^2.$$

In this idealized setting, flow matching amounts to least-square regression on the target vector field $v(x, \theta_0)$. Let $\{x_i\}_{i=1}^N$ a sample of size N drawn following $t \sim [0, 1]$ and $x \sim p_t$. Then, consider the empirical and population risks R_N, R

$$R_N(\theta) = \min_\theta \frac{1}{N} \sum_{i=1}^N \|\hat{v}(x_i; \theta) - v(x; \theta_0)\|^2,$$

$$R(\theta) = \mathbb{E}_{t,x} \|\hat{v}(x; \theta) - v(x; \theta_0)\|^2.$$

Our goal is to characterize identifiability of the structural graph from samples. Then this falls into the framework considered by Dinh and Ho (2020). Specifically, the results of [Dinh and Ho (2020), Theorem 3.6] guarantees local consistency of the learned graph from minimizing (1) in the limit where $n \to \infty$.

A.4 Experiment Details

Model architectures. The learned flow model is a NGM as described above. The score and residual models are standard multilayer perceptrons with ReLU activations. These support time-varying and conditionally encoded inputs (knockout vectors). All networks are initialized using Gaussian weights: $\mathcal{N}(0, 0.1)$ for weights and $\mathcal{N}(0, 1 \times 10^{-2})$ for biases.

Note on Entropic Optimal Transport We note that we choose to solve the EOT problem to pair successive time marginals ($\hat{p}^{(c)}_{t_i}$ and $\hat{p}^{(c)}_{t_{i+1}}$) independently of the other marginals $\hat{p}_t \mid t \notin \{i, i+1\}$, not globally across all time marginals like is done in Chen et al. (2023); Theodoropoulos et al. (2025); Rohbeck et al. (2025). StructureFlow is flexible however, in that one would simply swap the coupling algorithm to solve the global multi-marginal problem instead with no additional changes to training. This could be used for other means as well, such as incorporating biological priors.

Trajectory simulation. Simulations are run between pairs of discrete timepoints (t_k, t_{k+1}) where $k = 0, \ldots, T-1$ using 100 Euler steps. We consider two rollout strategies: (a) stochastic differential equation (SDE) sampling and (b) probability flow ODE (PF-ODE) sampling. Both start from p_{t_k}, the empirical distribution of cells at timepoint t_k, but differ in how the dynamics are integrated.

ODE simulation The probability flow ODE (PF-ODE), given by

$$\dot{x}_t = \boldsymbol{v}(x_t) - \tfrac{\sigma^2}{2}\boldsymbol{s}^{\phi}_t(x_t), \quad x_0 \sim p_{t_k}, \tag{15}$$

describes the deterministic evolution of the density p_t using our learned score $\boldsymbol{s}^{\phi}_t$ and flow $\boldsymbol{v}(x_t)$. The score corrects the flow to match the evolving density, while the deterministic rollout removes sample-path noise, yielding smoother trajectories but potentially underestimating variability.

Evaluation metrics. We report Wasserstein-2 (W2) distance, squared Maximum Mean Discrepancy (MMD^2), and Energy Distance Rizzo and Székely (2016) between predicted and ground truth distributions. W2 is computed using the exact Earth Movers Distance from the POT library with a squared Euclidean cost matrix.

MMD^2 is computed using a radial basis function (RBF) kernel. For each evaluation, we average over five kernel bandwidths. Each bandwidth is defined via a kernel scale parameter $\sigma \in \{0.01, 0.1, 1, 10, 100\}$. The final MMD^2 score is the mean of the values computed under each of these five kernel settings.

Energy distance is computed according to Rizzo and Székely (2016). For random vectors $X,\ Y \in \mathbb{R}^d$ with laws F, G, the squared energy distance is

$$\mathcal{D}^2(F, G) = 2\mathbb{E}\|X - Y\| \ - \ \mathbb{E}\|X - X'\| \ - \ \mathbb{E}\|Y - Y'\| \ \geq 0,$$

where X' and Y' are i.i.d. copies of X and Y. As described, it is the excess average cross-distance between two samples after subtracting their interal (within-sample) average distances, and is zero only when the two distributions coincide.

For structure learning evaluation, we compute the Area Under the Receiver Operating Characteristic curve (AUROC) and Average Precision (AP) of the learned graph against the ground-truth. AUROC measures the trade-off between true positive and false positive rates across all classification thresholds. AP summarizes the precision-recall curve as the weighted mean of precisions achieved at each threshold.

Experiment protocols.

- **Leave-one-out (timepoint):** For each dataset, we iteratively exclude data from a single discrete timepoint t_k for $k \in \{1, \ldots, T-2\}$ from training (we only hold out the interior timepoints). The model is trained on the remaining timepoints $\{t_j : j \in \{0, \ldots, T-1\} \setminus \{k\}\}$ and then used to simulate the transition from timepoint t_{k-1} to t_k. We report the average performance across all held-out timepoints by training a separate model for each excluded t_k.

- **Leave-one-out (knockout):** For each dataset, we select the first three gene knockouts and exclude one per trained model. Each model is then used to simulate transitions from t_k to t_{k+1} conditioned on the held-out inter across discrete timepoints $k = 0, \ldots, T-2$. A separate model is trained for each held-out knockout, and results are averaged across the three excluded interventions.

Seed values. All experiments are repeated across fixed seed values. For structure learning experiments, we use 5 seeds: {1, 2, 3, 4, 5}. For computationally intensive experiments (leave-one-timepoint-out and leave-one-knockout-out) we use 3 seeds: {1, 2, 3}

Hardware. All experiments were implemented in PyTorch and run on a MacBook Pro with an Apple M1 Pro CPU and 32GB RAM. No GPU is needed. All models train in under 10 minutes.

A.5 Hyperparameters

We report all hyperparameters used in training our models. Unless otherwise specified, we use the AdamW optimizer with a fixed weight decay of 1×10^{-2}.

Synthetic datasets. Models are trained for 15,000 steps using a batch size of 64 and a learning rate of 3×10^{-3}. Regularization and model-specific settings are provided below:

Table 5: Hyperparameters for synthetic datasets.

Hyperparameter	**Value**
Training steps	15,000
Batch size	64
Learning rate	3×10^{-3}
Weight decay	1×10^{-2}
α (Score term influence)	0.1
ℓ_2 regularization	5×10^{-6}
Residual regularization	1×10^{-3}
group lasso regularization	0.04
Knockout hidden layers	[100]
Score model hidden layers	[100, 100]
Residual model hidden layers	[64, 64]
SDE noise scale σ	1.0

Real data. For the real data experiments, some hyperparameters are overriden:

Table 6: Overriden Hyperparameters for real data (RENGE).

Hyperparameter	**Value**
Training steps per fold	10,000
Learning rate	0.07
ℓ_2 regularization	1×10^{-9}
α (Score term influence)	0.73
group lasso regularization	0.0008
Knockout hidden layers	[128]
Score model hidden layers	[128, 128]
Residual model hidden layers	[128, 128]

We used Optuna to perform hyperparameter optimization, and selected configurations that maximized the area under the precision-recall curve (AUPR). The search space included key hyperparameters such as

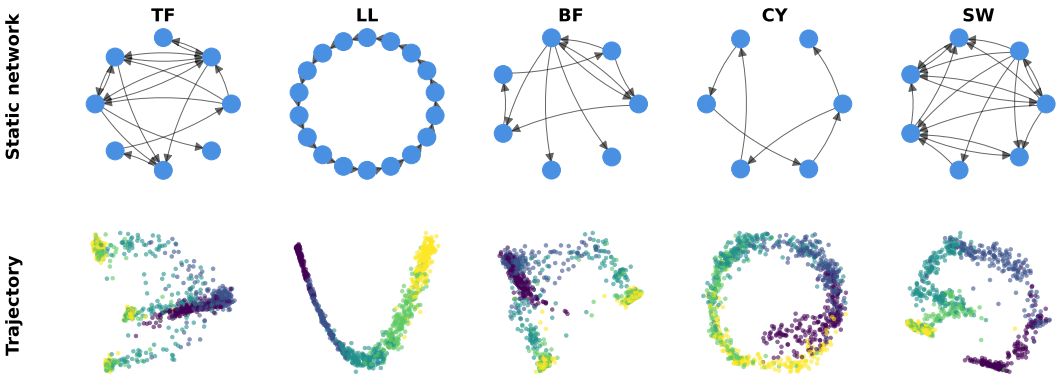


Figure 6: **BoolODE synthetic biological systems.** Network topologies and example trajectory visualizations for the six synthetic biological systems used in our experiments: trifurcating (TF), cyclical (CY), long linear (LL), swirling (SW), bifurcating (BF), and hematopoietic stem cell (HSC) systems. Figure is sourced from Zhang (2024).

the learning rate, regularization strength, group lasso regularization, batch size, and model capacity (e.g., number of hidden units). For each trial, model performance was evaluated using cross-validation, and the trial achieving the highest mean AUPR across folds was selected as the optimal configuration. AUPR is computed only on the available ChIP-seq reference edges for the 18 TFs used in evaluation; because the available reference edges are only partial, the resulting AP/AUROC should be interpreted as benchmark agreement with known or curated edges rather than an unbiased estimate of whole-network recovery. We note that the score term was of higher importance here. This is likely due to the fact that scRNA-sequencing data is much noisier than the synthetic systems.

A.6 Synthetic Linear System

We generate synthetic linear systems to evaluate structure learning performance across varying dimensionality and sparsity levels. The underlying network structure is generated using Erdos–Renyi random graphs (Erdös and Rényi, 1959), where edges are placed independently with probability p to achieve the desired sparsity level (5%, 20%, or 40% of possible edges).

For a system with d variables, we construct a random adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ where non-zero entries are sampled to avoid weak interactions. Specifically, edge weights are drawn from the union of two uniform distributions: $\mathcal{U}(-1.0, -0.5) \cup \mathcal{U}(0.5, 1.0)$, ensuring all edges have magnitude ≥ 0.5. Positive and negative edges occur with equal probability, representing activating and inhibiting interactions respectively. The linear SDE governing the system dynamics is:

$$d\boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{x}_t dt + \sigma d\boldsymbol{B}_t \tag{16}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ represents the system state, $\boldsymbol{A}$ encodes the linear interactions, and $\sigma = 0.1$ is the noise level. We simulate trajectories over $T = 5$ time points, generating $N = 1000$ samples per time point. The initial conditions are sampled from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, 0.5\boldsymbol{I})$.

A.7 BoolODE

BoolODE was employed to simulate 1000 cells independently for each trajectory. To obtain time-resolved snapshots, we divided the simulation into T=5 discrete intervals. Expression outputs from simulations were log-transformed for use in subsequent analysis. Knockout trajectories were produced by altering Boolean rules so that the targeted gene retained only self-activation, while its initial expression level was set to zero.

To evaluate performance across different conditions, we considered gene expression values across across the following conditions for each dataset type:

- HSC: Gata1, Fli1, Fog1, Eklf, Scl, Gfi1, EgrNab, cJun
- TF, BF, CY, SW, and LL: observational (wild-type), g2, g3, g4, g5, g6, g7, g8

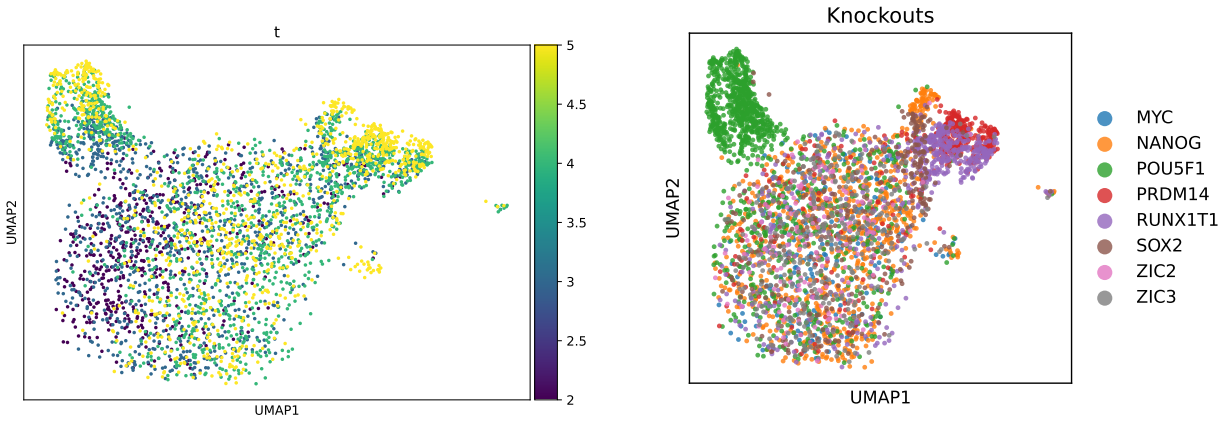
Since we considered 1000 cell expression trajectories, each measured across 800 continuous time and pseudo-time values, and divided across eight conditions, we were left with 100,000 cell measurements per condition. This was then further discretized into T=5 time bins.

A.8 Single-cell CRISPR perturbation time-series

Following Zhang (2024), we use raw count data from Ishikawa et al. (2023), retrieved from the Gene Expression Omnibus database (accession GSE213069). Zhang et al. use the following procedure:

- Columns corresponding to gRNAs were removed.
- Counts were normalized using the `scanpy.pp.normalize_total` function with default options.
- The data was log-transformed.
- Prior to dimensionality reduction, highly variable genes were selected using scanpy.pp.highly_variable_genes.

We considered the set of 103 transcription factors that Zhang (2024) uses and included only cells that received a single knockout. To construct the ChIP-seq reference, we obtained experimental binding information from ChIP-Atlas Oki et al. (2018) for the following TFs (within a 1kb window): Chd7, Ctnnb1, Dnmt1, Foxh1, Jarid2, Kdm5b, Med1, Myc, Nanog, Nr5a2, Pou5f1, Prdm14, Sall4, Sox2, Tcf3, Tcf7l1, Ubtf, Znf398 with a 1kb window. We then further reduced the knockouts to a group of 8 most prominent ones as referenced in Zhang (2024). We show a visualization of the dateset in UMAP space in Figure 7.



(a) Time (Days 2-5)

(b) Interventions (knock-outs)

Figure 7: **Visualization of real (Renge) dataset.** (a) UMAP visualization showing temporal progression of wildtype (unconditional) data across days 2-5, with colors indicating time points. (b) UMAP visualization of intervention conditions showing trajectories under different gene knock-out perturbations, with colors indicating the specific knocked-out gene.

A.9 Extension to Anisotropic Diffusion

For clarity of exposition, Sections 2 and 3 assumed isotropic diffusion $\boldsymbol{\sigma} = \sigma\mathbf{I}$. The framework extends naturally to anisotropic diffusion $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$ with only minor changes to the conditional targets and the EOT cost.

Anisotropic conditional targets. Let $\boldsymbol{\Sigma} = \boldsymbol{\sigma}\boldsymbol{\sigma}^\top$. The Brownian bridge conditioned on endpoints $\boldsymbol{z} = (\boldsymbol{x}_0, \boldsymbol{x}_1)$ has conditional score

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x} \mid \boldsymbol{z}) = \frac{1}{t(1-t)} \boldsymbol{\Sigma}^{-1}\big(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_0 - \boldsymbol{x}\big),$$

while the conditional probability flow $\boldsymbol{v}_t^\circ(\boldsymbol{x} \mid \boldsymbol{z})$ retains the same form as in the isotropic case, since the displacement field does not depend on $\boldsymbol{\Sigma}$.

Anisotropic EOT coupling. The squared Euclidean cost in Eq. (6) is inappropriate when $\boldsymbol{\Sigma}$ is non-isotropic. The correct EOT problem uses a Mahalanobis-type cost,

$$\pi^\star = \arg\min_{\pi \in \Pi(q_0, q_1)} \ \tfrac{1}{2}\,\mathbb{E}_\pi\big[(\boldsymbol{x}-\boldsymbol{y})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{y})\big] \ + \ \varepsilon\,\mathrm{KL}(\pi \,\|\, q_0 \otimes q_1),$$

which still admits efficient solution via Sinkhorn iterations after a preconditioning step.

Reduction to diagonal diffusion. Beyond these direct extensions, linear SDEs with arbitrary diffusion can be reduced to an equivalent SDE with diagonal diffusion via a change of basis, as formalized below.

Lemma 1. *Up to a change of basis, any linear SDE is equal in law to a linear SDE with diagonal diffusion. Concretely, take* $\mathrm{d}\boldsymbol{X}_t = \boldsymbol{A}\boldsymbol{X}_t\mathrm{d}t + \boldsymbol{\sigma}\,\mathrm{d}\boldsymbol{B}_t$ *with drift* $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ *and diffusion* $\boldsymbol{\sigma} \in \mathbb{R}^{d\times d}$*. Then there exists an orthogonal* $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ *such that* $\boldsymbol{Y}_t = \boldsymbol{U}^\top \boldsymbol{X}_t$ *obeys a linear SDE with drift matrix* $\boldsymbol{U}^\top\boldsymbol{A}\boldsymbol{U}$ *and a diagonal diffusion matrix* $\boldsymbol{\Lambda}$*.*

Proof. Let $\boldsymbol{D} = \boldsymbol{\sigma}\boldsymbol{\sigma}^\top/2$. By the spectral theorem, $\boldsymbol{D} = \boldsymbol{U}\boldsymbol{\Lambda}^2\boldsymbol{U}^\top/2$ for some orthogonal $\boldsymbol{U}$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda}) \succeq 0$. Consider the two SDEs

$$\text{(a)} \quad \mathrm{d}\boldsymbol{X}_t = \boldsymbol{A}\boldsymbol{X}_t\,\mathrm{d}t + \boldsymbol{\sigma}\,\mathrm{d}\boldsymbol{B}_t,$$
$$\text{(b)} \quad \mathrm{d}\boldsymbol{X}_t = \boldsymbol{A}\boldsymbol{X}_t\,\mathrm{d}t + \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top\,\mathrm{d}\boldsymbol{B}_t.$$

The generator of (a) is $(\mathcal{L}f)(\boldsymbol{x}) = \langle \partial_{\boldsymbol{x}} f(\boldsymbol{x}), \boldsymbol{A}\boldsymbol{x}\rangle + \frac{1}{2}\mathrm{tr}\big(\boldsymbol{\sigma}^\top \partial_{\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{x})\,\boldsymbol{\sigma}\big)$, and that of (b) is $(\tilde{\mathcal{L}}f)(\boldsymbol{x}) = \langle \partial_{\boldsymbol{x}} f(\boldsymbol{x}), \boldsymbol{A}\boldsymbol{x}\rangle + \frac{1}{2}\mathrm{tr}\big(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top \partial_{\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{x})\,\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top\big)$. Since $\mathrm{tr}(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top \partial_{\boldsymbol{x}\boldsymbol{x}} f\,\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top) = \mathrm{tr}(\boldsymbol{\sigma}^\top \partial_{\boldsymbol{x}\boldsymbol{x}} f\,\boldsymbol{\sigma})$, the generators coincide and the SDEs are equal in law.

Orthogonal transformations preserve Brownian increments, so $\boldsymbol{U}^\top\,\mathrm{d}\boldsymbol{B}_t = \mathrm{d}\tilde{\boldsymbol{B}}_t$ for a Brownian motion $\tilde{\boldsymbol{B}}_t$. Substituting into (b), we obtain $\mathrm{d}\boldsymbol{X}_t = \boldsymbol{A}\boldsymbol{X}_t\,\mathrm{d}t + \boldsymbol{U}\boldsymbol{\Lambda}\,\mathrm{d}\tilde{\boldsymbol{B}}_t$, and the result follows. □ □

B Implementation Details

B.1 StructureFlow Algorithm

Algorithm 1 StructureFlow Training Loop

Require: Training data across conditions c, hyperparameters $\alpha, \sigma, \lambda_{\mathrm{GL}}$, number of steps N, d variables

Initialize parameters $\Theta = \{\theta, \phi\}$, Optimizer *Opt*
for step $n = 1$ to N **do**
  $\boldsymbol{M}^{(c)} \leftarrow M_{ji}^{(c)} = 0$ if $i = c, j \neq c, M_{ji}^{(c)} = 1$ otherwise **for all** $j, i \in 1, ..., d \times 1, ..., d$
  $\boldsymbol{k}^{(c)} \leftarrow k_j^{(c)} = 1$ if j perturbed under c, else $k_j^{(c)} = 0$ **for all** j
  Sample $(\boldsymbol{x}_t, t, \boldsymbol{s}_t, \boldsymbol{v}_t^\circ)$ from SB for condition c
  $\boldsymbol{s}_t^\phi \leftarrow \boldsymbol{s}_t^\phi(\boldsymbol{x}_t|\boldsymbol{k}^{(c)})$
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}^{\theta_j}(\boldsymbol{x}_t \mid \boldsymbol{M}^{(c)}) - \frac{\boldsymbol{\sigma}^2}{2}\,\boldsymbol{s}_t^\phi\big(\boldsymbol{x}_t \mid \boldsymbol{k}^{(c)}\big)$ ▷ Using $\boldsymbol{M}^{(c)} \odot \theta_j^A$
  $\mathcal{L} \leftarrow \sum_c \mathbb{E}\left[(1-\alpha)\left\|\hat{\boldsymbol{v}}_t(\boldsymbol{x}_t; \Theta|\boldsymbol{M}^{(c)}, \boldsymbol{k}^{(c)}) - \boldsymbol{v}_t^\circ\right\|^2 + \alpha\left\|\boldsymbol{s}_t^\phi(\boldsymbol{x}_t|\boldsymbol{k}^{(c)}) - \boldsymbol{s}_t\right\|^2\right]$
  $\boldsymbol{g} \leftarrow \nabla_\Theta \mathcal{L}$
  Update parameters $\Theta \leftarrow Opt(\Theta, \boldsymbol{g})$
  $\theta_j^A \leftarrow \mathrm{Prox}_{\eta\,\lambda_{\mathrm{GL}}\|\cdot\|_{\mathrm{group}}}\big(\theta_j^A\big)$ ▷ group-wise proximal update on first layer
end for

B.2 Gene Regulatory Network Inference Baselines

All baselines were trained with the same pre-processing steps, train-test splits, and evaluation metrics. In addition, we experimented with the hyper-parameters of the baseline methods. For methods that considered the same BoolODE dataset(s) (Zhang, 2024; Zhao et al., 2024), we used the respective hyper-parameters reported in the work. For other methods, we experimented with hyper-parameters settings, but found that the hyper-parameters set by the authors of the respective baseline methods performed generally well.

SCODE: Sparse COding for Differential Equations (SCODE) from Matsumoto et al. (2017) is a method for reconstructing GRNs from single-cell time series by fitting a linear ODE system to gene expression trajectories. The method infers a sparse gene interaction matrix such that simulated expression profiles recapitulate observed temporal dynamics. In the synthetic case, we take the transposed expression matrix and pseudotime vector that is given to us in both the observational and interventional settings and write them to disk in a format compatible with SCODE. We then execute the model via the Ruby + R wrapper developed by Matsumoto et al. (2017), and read the mean interaction matrix from the resulting output files, which gives us an adjacency matrix. SCODE does not work with interventional data so we do not expect an increase in performance.

SINCERITIES: Single-Cell Network Inference by Covariance Regression for Time Series, from Gao et al. (2018) is an R-based algorithm for inferring directed GRNs from time-course single-cell data. SINCERITIES combines information from lagged covariances and local linear regression to estimate causal effects, and it outputs signed edge weights. We use the R script developed by the authors and the given expression matrix and psuedotime matrix to create a `GRNPrediction.txt` file which gives a gene $\times$ gene matrix.

GLASSO: Graphical Lasso is a classical method for inferring sparse undirected graphical models from high dimensional data. GLASSO estiamtes the precision (inverse covariance) matrix under an l_1 penalty to enforce sparsity, such that nonzero entries correspond to conditional dependencies between genes. This method does not work with time-series data. We first standardize the gene expression matrix, and choose the regularization parameter via cross-validation. The estimated precision matrix is negated (higher values correspond to stronger interactions), and the diagonal is set to zero. Our result is an adjacency matrix with symmetric edge weights that represent partial correlations between genes.

dynGENIE3: This method from Huynh-Thu and Geurts (2018) is an extension of the GENIE3 algorithm Huynh-Thu et al. (2010b) to reconstruct GRNs from time-series single-cell or bulk gene expression data. The method frames network inference as a feature selection problem, where for each gene, a random forest regression model is trained to predict its expression at each time point as a function of all other genes' expressions at earlier time points. The regulatory strength is aggregated over all of the trees and time lags, and yields a directed adjacency matrix. We aggregated our expression matrix `adata.X`, which is an annData object, by time points (mean expression per time) and store it in a list `X`, and keep the corresponding points in `t`. The dynGENIE3 model is then called with these two parameters as input and we save the resulting adjacency matrix. By averaging across all cells at each time t we effectively create pseudo-bulk expressions, then stack these per-timepoint means into a matrix so as to be a single trajectory. This approach is corroborated by Moscardó García et al. (2025) who use a nearly identical procedure for dynGENIE3.

RENGE: RENGE Ishikawa et al. (2023) can use both temporal and interventional data. Renge uses pseudotime trajectories from single-cell data, and CRISPR knockout information, with a two-step inference strategy, that involves a regression model to predict expression as a function of genes, and network aggregation to aggregate the inferred effects across all perturbations and time points. This gives us a gene $\times$ gene matrix based on both the endogenous dynamics and systematic perturbation present in the dataset. The main experiment from Ishikawa et al. (2023) was what was used in our paper and in Zhang (2024). We use the model that Ishikawa et al. (2023) provide in their codebase and do further experimentation, using that matrix as a baseline.

Table 7: Assumptions and capabilities of StructureFlow and baseline methods. We indicate whether each method uses interventional (knockout) data, requires discrete time labels, relies on pseudo-trajectories or OT pairings to recover temporal correspondence, applies a dimensionality-reduction step (PCA/autoencoder) before inference, and whether it addresses the *joint* structure-and-dynamics task or only one of the two.

Method	Interventions	Time labels	Pseudo-traj.	PCA/AE	Joint task
StructureFlow (ours)	Yes (masks)	Yes	No	No	Yes
$[SF]^2M$	Cond. input	Yes	No	No	No (dynamics only)
RF	Yes (mask on A)	Yes	No	No	Yes (linear)
TIGON	No	Yes	No	Optional	Yes (Jacobian)
OTVelo	No	Yes	OT pairings	No	Yes
dynGENIE3	No	Yes	Pseudo-bulk	No	Structure only
SINCERITIES	No	Yes	No	No	Structure only
SCODE	No	Yes (pseudotime)	No	No	Structure only
Glasso	No	No	No	No	Structure only (undirected)
RENGE	Yes	Yes (pseudotime)	No	No	Structure only
PCMCI (adapted)	Both	Yes	OT pseudo-traj.	No	Structure only

TIGON: Trajectory Inference with Growth via Optimal transport and Neural networks (TIGON) (Sha et al., 2024) models time-series scRNA-seq as a continuum of cell states whose density evolves under a reaction-transport dynamic. It represents a time-dependent cell-state density represented by two fields: a velocity field that models state transitions, and a growth field that accounts for birth/death and mass change. TIGON integrates the learned dynamics using ODE integration per time step, and learns to minimize a the Wasserstein-Fisher-Rao distance. At inference, TIGON learns to reconstruct the continuous trajectories and velocities for individual cells, estimates time-resolved growth maps, and can learn a gene-regulatory structure from the Jacobian of its flow. TIGON uses either a PCA layer or an autoencoder to first encode the ambient data. This is a step for the "dimensionless-solver" used within TIGON, which is optimized to operate in 10 dimensional space. All of the other baselines, and StructureFlow, are trained directly in the ambient space. Thus, in order to provide a comparative setting, we train TIGON in the ambient space as well and forgo the PCA/autoencoder step. We did not encounter challenges with applying TIGON to the high dimensional real (Renge) biological system. TIGON uses Gaussian mixture models to represent p_t (equivalent to our Eq. 2). We also train TIGON in PCA space, and then convert to the ambient space using the following formula: $J_X = W^T J_z W$ where J_z is the latent Jacobian. We note no difference in performance for structure learning.

OTVelo: OTVelo Zhao et al. (2024) estimates gene velocity from cell trajectories reconstructed via optimal transport. The method proceeds in two stages. The first stage is trajectory inference where it aligns cell across consecutive time-points using Fused Gromov–Wasserstein (FGW) optimal transport, which uses both expression state and local geometric structure. The second part uses OT pairings to estimate the gene velocities via two different approaches. We consider two versions of this model, OTVelo-Corr and OTVelo-Granger. OTVelo-Corr uses time-lagged correlation analysis that generalizes the standard time-lagged correlation to distributions of cells matched by the OT pairings. OTVelo-Granger, on the other hand, is a regularized linear regression model (Elastic Net) where the velocity of a target gene at time $t+1$ is predicted from the velocities of regulating gene at time t. Given that we discretize all of our data into bins, and OTVelo tests their model on the same datasets, simulated using BoolODE, we apply the model from Zhao et al. (2024) directly in our pipeline.

B.3 Unbalanced StructureFlow

In order to properly model cell proliferation and death, we formalize the extension of StructureFlow to biological systems where probability mass is not conserved. We generalize the probability flow formulation to include a growth term, derived from Unbalanced Optimal Transport (UOT).

Unbalanced Optimal Transport In the standard StructureFlow framework, couplings $(x_0, x_1) \sim \pi(x_0, x_1)$ are acquired via a balanced Sinkhorn solver, which enforces strict mass conservation ($\sum \pi_i = \frac{1}{N}$). To model cell proliferation and death, we replace this with UOT. The universalness of our method allows us to easily incorporate unbalancedness into StructureFlow by rescaling source and target measures Eyring et al. (2024). We compute the coupling π^* between empirical snapshots $\hat{\rho}_{t_k}$ and $\hat{\rho}_{t_{k+1}}$ by minimizing the unbalanced Sinkhorn Divergence Wang et al. (2025); Chizat et al. (2019):

$$\pi^* = \arg \min_{\pi \in \mathbb{R}_+^{N \times M}} \sum_{i,j} C_{ij}\pi_{ij} \ + \epsilon H(\pi) \ + \rho[KL(\pi_{\#0}|\hat{\rho}_{t_k})] \ + KL(\pi_{\#1}|\hat{\rho}_{t_{k+1}})]$$

where $\epsilon H(x)$ is the entropic regularization coefficient, ρ is the marginal relaxation parameter, and $\pi_{\#0}$ are row sums which control the amount of mass transported away from each starting cell. If $\pi_{\#0} < \hat{\rho}_{t_k}$ then the transport plan used less mass than was available and if $\pi_{\#1} > \hat{\rho}_{t_{k+1}}$ then the transport plan delivered more mass than was originally available relative to the source.

Stochastic Dynamics with Growth To account for cellular growth, we adopt the formulation proposed in Zhang et al. (2025), where the population density $\rho_t(x)$ evolves according to a Fokker–Planck equation with a source term:

$$\partial_t \rho_t(x) = -\nabla \cdot (\rho_t(\boldsymbol{x}_t)\boldsymbol{v}_t(\boldsymbol{x}_t)) + \tfrac{1}{2}\nabla \cdot (\sigma^2 \nabla \rho_t(\boldsymbol{x}_t)) \ + g_t(x)\rho_t(x)$$

where $\boldsymbol{v}_t(\boldsymbol{x})$ represents the drift term and $\frac{1}{2}\nabla \cdot (\sigma^2 \nabla \rho_t(\boldsymbol{x}_t))$ is the diffusion coefficient. Unlike eq. (2) where probability mass is conserved, the total mass $M_t = \int \rho_t(x)dx$ varies over time according to the net growth rate of the system. We model the joint evolution where the local change in log-density is governed by both the divergence of the flow and the local growth rate, defined as:

$$\frac{d}{dt} \log \rho_t(x_t) = g_t(x_t) - \nabla \cdot u_t(x_t)$$

where $u_t(x)$ is defined in eq. (3). This formulation then allows us to disentangle the vector field v_t from the growth field, so that the density increase is due to the growth field, and the vector field only models transitions. Lastly to learn the growth field $g_\phi(x)$, we adopt the simulation-free strategy proposed in Wang et al. (2025). Instead of solving a differential equation for mass evolution during training, we interpret the row-sums of the unbalanced coupling matrix as static proxies for net proliferation accumulated along a trajectory. For a source sample x_0^i, the target growth G_i is defined by the log-marginal of the transport plan π^*:

$$G_i = \log(\sum_j \pi_{ij}^*)$$

This value represents the total probability mass at t_1 attributed to the lineage of x_0^i. If the sum is greater than 1, the cell has proliferated, otherwise the cell has undergone apoptosis (died). To learn the growth network, we regress the neural network $g_\phi(x, t)$ to the static target across the interpolated trajectory:

$$\mathcal{L}_{\text{growth}}(\phi) = \mathbb{E}_{t,(x_0^i, x_1^j)\sim\pi^*}[\|g_\phi(x_t) - G_i\|^2]$$

C Novelty Statement

This work introduces key methodological and empirical contributions to joint structure learning and dynamics inference:

Decomposition of Stochastic Dynamics. We decompose the probability flow into an autonomous drift $\boldsymbol{v}(\boldsymbol{x})$ capturing system structure and a time-dependent score function $\boldsymbol{s}_t(\boldsymbol{x})$ modeling stochasticity. This aligns with biophysical reality where regulatory mechanisms are time-invariant.

Conditional Dynamics via Interventions. We incorporate interventional data through knockout conditioning masks, enabling learning from perturbational experiments and interventional trajectory inference on unseen perturbations.

Simulation-Free Training for High-Dimensional Systems. Our framework extends simulation-free training to conditional settings, enabling tractable learning on high-dimensional systems (up to 500 variables) while jointly inferring structure and dynamics.

Comprehensive Evaluation. We establish extensive benchmarks including scalability analysis, evaluation on six biologically plausible systems with interventional data, and validation on real single-cell CRISPR time-series data with realistic constraints (4-5 time points).

D Broader Impact

The genome is the fundamental identity of every living thing. The expression of DNA dictates life. It is a multi-dimensional representation of the state of any living thing. The ultimate goal of our work is to enable improve our understanding of how an organisms genes interact, allowing us to predict the trajectory of its life, and ultimately to computationally develop interventions that can positively change that trajectory. The stochastic dynamical systems we consider here are used ubiquitously in the biophysics and modelling literature. In a biological context, our vector field $\boldsymbol{v}$ can be related to the metaphorical "Waddington's landscape".

E Additional Results

E.1 Structure Learning on Synthetic Systems with Observational Data

In 8, we observe that StructureFlow achieves competitive results in most systems across both AP and AUROC metrics. Whereas other methods tend to fluctuate widely in performance across these synthetic systems, StructureFlow underperforms all other methods only on the SW system, while beating most other benchmarks otherwise.

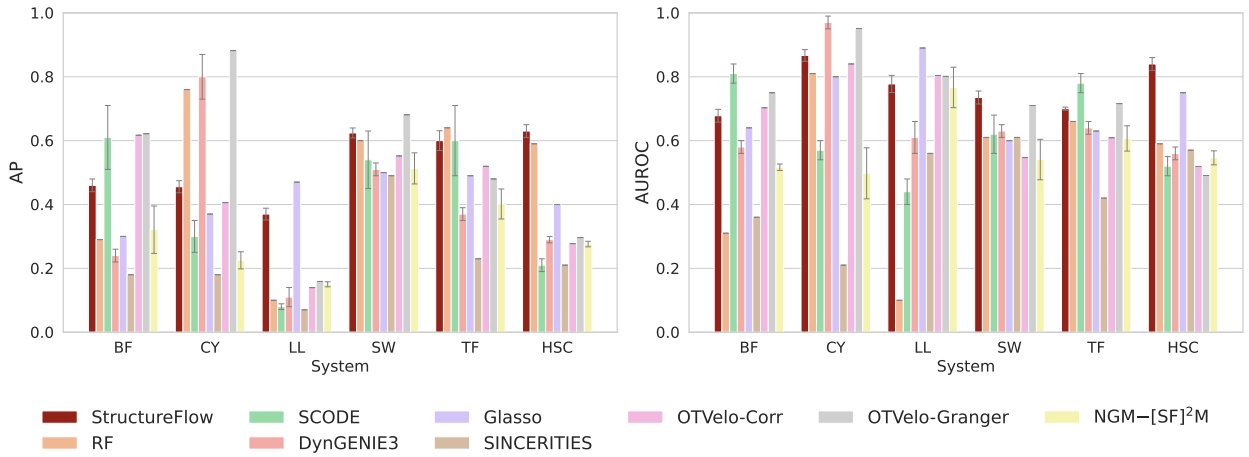
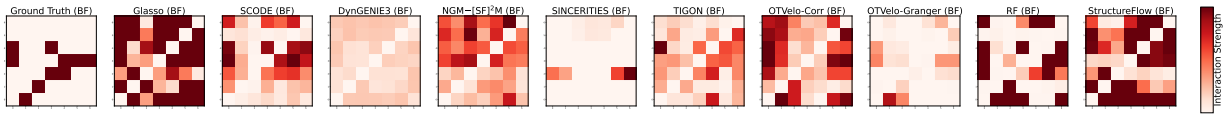


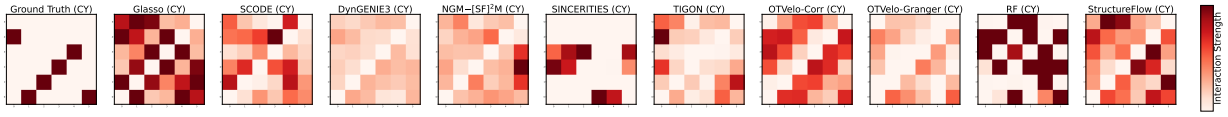
Figure 8: **Structure learning performance across synthetic systems using only observational data**. Shown are average precision (AP) and area under the ROC curve (AUROC) scores for models trained on only observational data

E.2 Dynamical Inference of Conditional Population Dynamics on Synthetic Systems using [SF]^{2}M

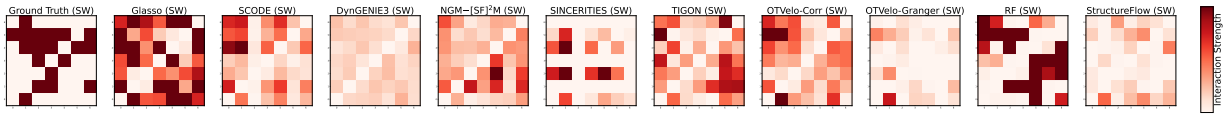
We note in Figure 11 that [SF]^{2}M predictions cluster around the synthetic systems' ground-truth trajectories across both observational and gene knockout settings. This is a result of training across both unconditional and conditional data.



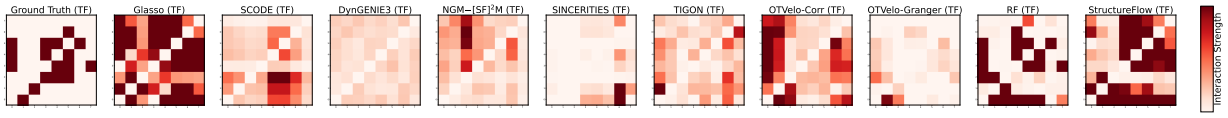
(a) Dataset BF (Observational)



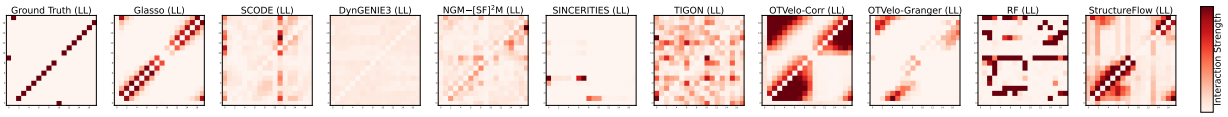
(b) Dataset CY (Observational)



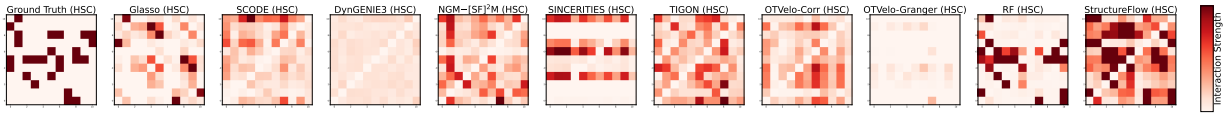
(c) Dataset SW (Observational)



(d) Dataset TF (Observational)

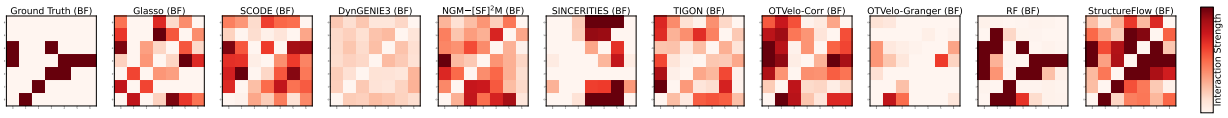


(e) Dataset LL (Observational)

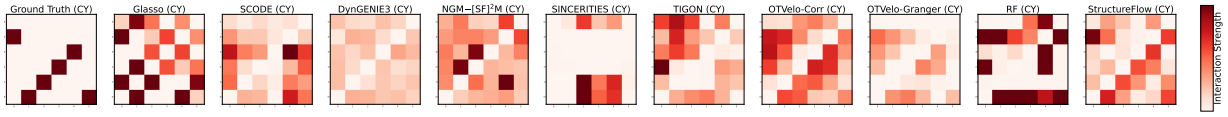


(f) Dataset HSC (Observational)

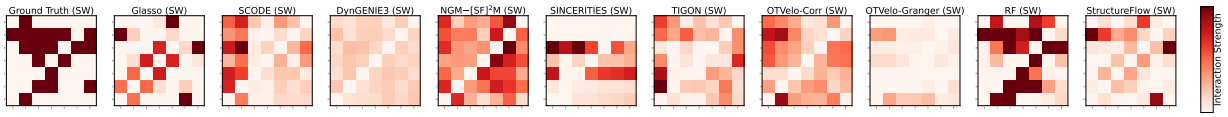
Figure 9: Six of the synthetic datasets with only observational information. Refer to panels (a)(f) when discussing individual datasets.



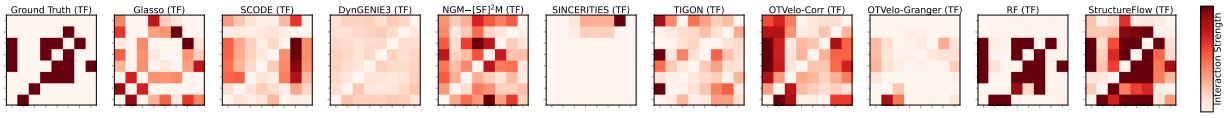
(a) Dataset BF (Observational + Interventional)



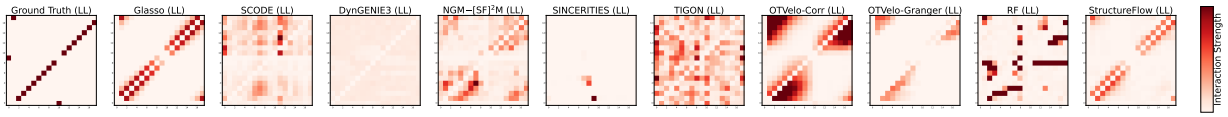
(b) Dataset CY (Observational + Interventional)



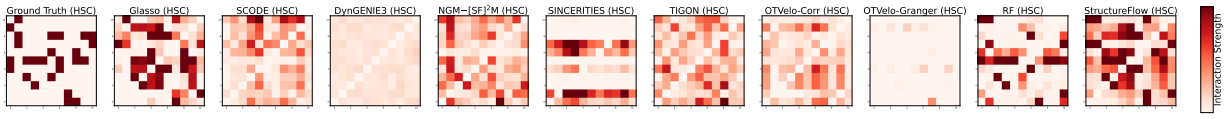
(c) Dataset SW (Observational + Interventional)



(d) Dataset TF (Observational + Interventional)



(e) Dataset LL (Observational + Interventional)



(f) Dataset HSC (Observational + Interventional)

Figure 10: Six of the synthetic datasets with the addition of interventional information. Refer to panels (a)(f) when discussing individual datasets.

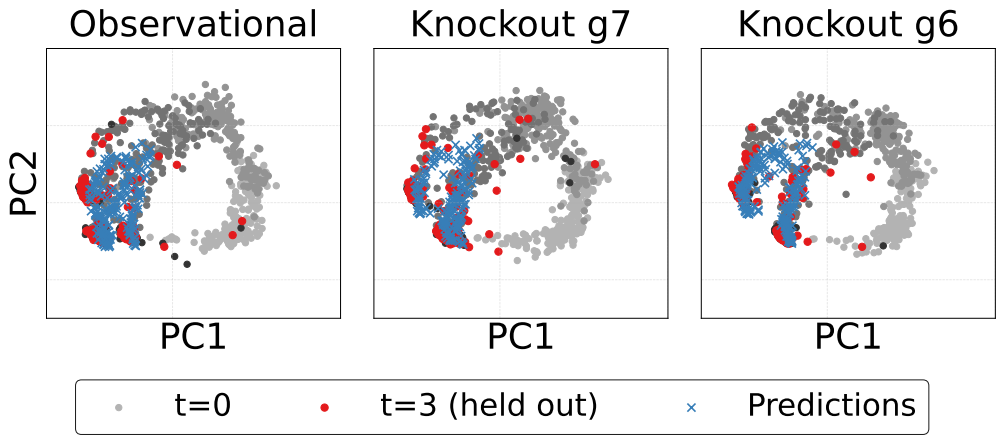


Figure 11: [SF]^{2}M **trajectory inference visualization.** Performance on observational (wild-type) data and two interventions selected for their diverse trifurcating paths. These examples illustrate the trajectory inference problem, where timepoint 3 was excluded during training, and the model simulates with the ODE from timepoint 2 to timepoint 3. Ground truth data is shown in grey, with progressively darker shades as time increases, except for timepoint 3 which is highlighted in red. Blue represent the model's predictions.

E.3 Prediction to Unseen Interventions (Knock-outs) on Synthetic Systems

Table 8 shows performance of StructureFlow and baselines on the leave-one-knockout-out trajectory inference task. StructureFlow improves over RF on W_2 for the TF and CY systems, while RF remains stronger on several MMD comparisons and the Curated (HSC) system.

Table 8: **Comparison of left-out intervention (knock-out) prediction on simulated biological systems.** For each dataset, we report Wasserstein-2 (W_2) and MMD metrics. Lower is better.

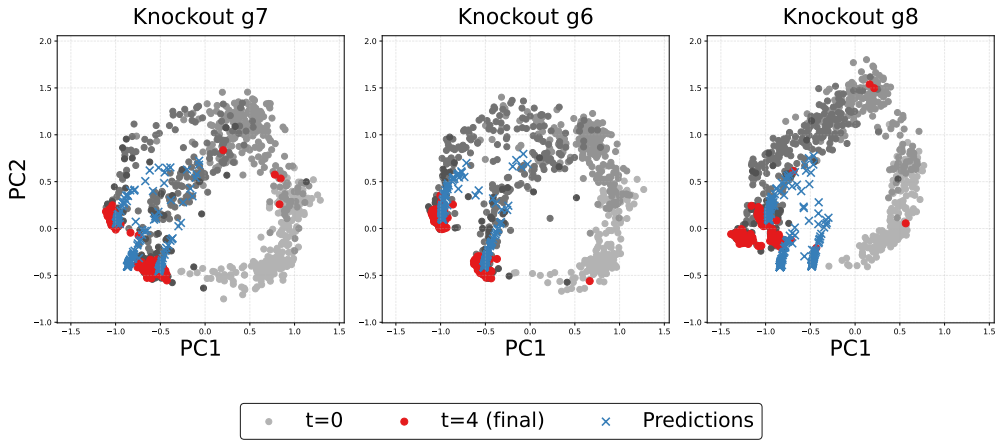
	TF		CY		LL		HSC		BF		SW	
	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD
[SF]^{2}M	0.964 ± 0.016	0.103 ± 0.003	0.967 ± 0.028	0.277 ± 0.017	1.082 ± 0.095	0.157 ± 0.010	0.886 ± 0.011	0.104 ± 0.003	1.138 ± 0.031	0.220 ± 0.005	1.404 ± 0.051	0.358 ± 0.015
RF	1.062 ± 0.000	0.098 ± 2.8e-8	1.889 ± 0.000	0.329 ± 1.5e-8	1.887 ± 0.005	0.232 ± 6.5e-4	0.984 ± 3.4e-10	0.065 ± 1.7e-8	1.176 ± 0.000	0.134 ± 1.1e-8	1.285 ± 0.000	0.284 ± 3.4e-8
StructureFlow	1.012 ± 0.010	0.110 ± 4.1e-3	0.924 ± 0.036	0.250 ± 1.8e-2	1.123 ± 0.037	0.141 ± 6.1e-3	0.852 ± 0.011	0.120 ± 4.2e-3	1.079 ± 0.016	0.213 ± 7.1e-3	1.159 ± 0.032	0.330 ± 1.1e-2

E.4 OT-CFM Ablation on Synthetic (Trifurcating) System

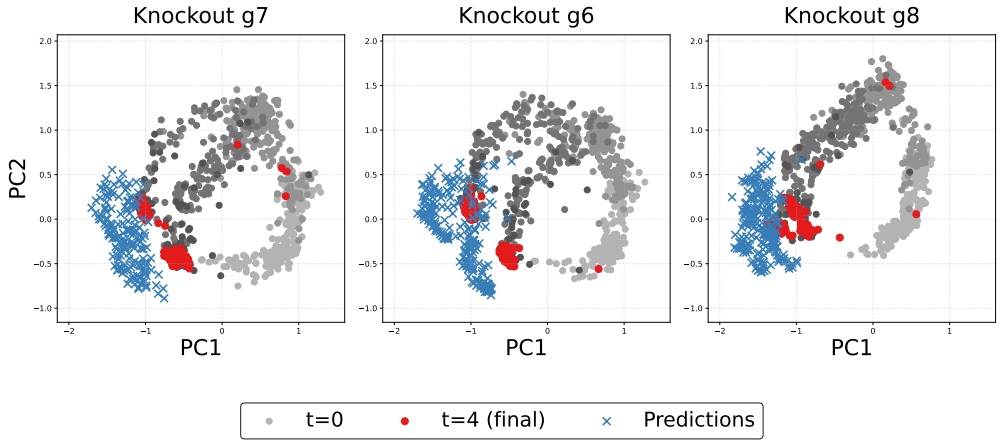
In Table 9 we remove the stochastic noise term from the conditional flow matching objective, reducing the formulation to an ODE that models the drift component. This corresponds to the classical flow matching objective Lipman et al. (2023b), where the dynamics align deterministic trajectories. StructureFlow is trained to model the process as a Schrödinger Bridge, where the forward-backward SDE explicitly models both drift and diffusion. From the results, we observe that discarding the stochastic component (OT-CFM) substantially degrades both classification (AUROC/AUPR) and distributional alignment metrics (W_2), highlighting that noise modeling contributes meaningfully to the structure learning task. For dynamical systems such as cellular trajectories observed across time points, modeling the process as a Schrödinger Bridge rather than an ODE provides a more accurate representation of developmental variability and uncertainty.

Table 9: **Comparison of StructureFlow with and without stochastic component on the TF synthetic system.** Results show AUROC, AUPR, and distributional distance metrics (W_2, MMD).

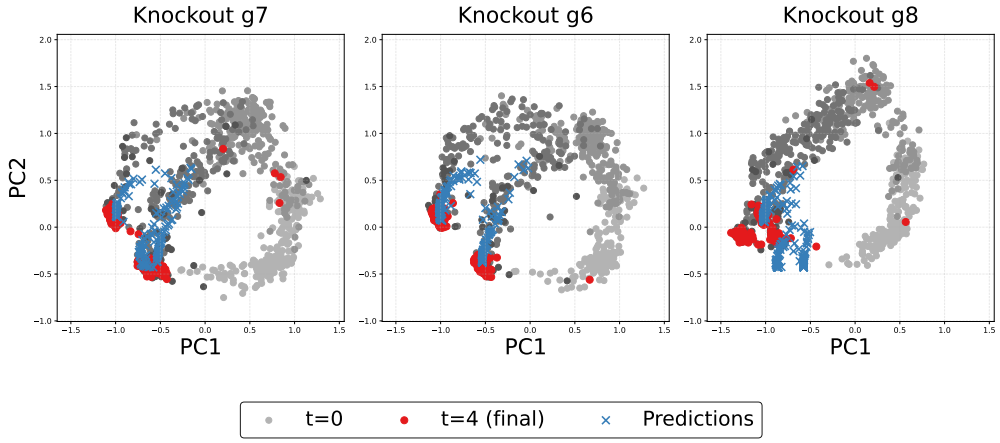
	AUROC	AUPR	W_2	MMD
StructureFlow	**0.968 ± 0.005**	**0.932 ± 0.011**	**0.795 ± 0.007**	0.136 ± 0.002
StructureFlow (OT-CFM)	0.791 ± 0.013	0.634 ± 0.037	0.846 ± 0.022	**0.062 ± 0.004**



(a) [SF]^{2}M **(no structure)**



(b) **RF**



(c) **StructureFlow (ours)**

Figure 12: **Comparison of trajectory inference performance for left-out interventions (knockouts).** We show PCA visualizations of model predictions for interventions that were excluded during training. Ground truth data is shown in grey, with model predictions overlaid. This demonstrates each method's ability to generalize to unseen perturbational conditions.

E.5 Scaling and runtime of various baselines and our method

Here we provide an additional experiment comparing training time of all methods on the BoolODE **TF** system. For this experiment, all methods are run on the same device (GPU, Quadro RTX 6000), but we note StructureFlow can be efficiently trained using a CPU. We only report parameter counts for neural network-based methods. For StructureFlow and the flow matching baseline, we use $(\cdot)$ to denote the additional time used to compute EOT. We use the same number of timepoints (5), dimensionality (8), and number of cells (8000). We add flow matching as a reference point to StructureFlow, where we remove regularization terms, the score model, and replace the NGM backbone with a traditional MLP backbone. The comparison of wall-clock timing, together with the synthetic scaling study (Figure 2) and the Renge runtimes (Table 16), use a common data setup and were run on a standard desktop workstation. For baselines, hyperparameters were set to author-recommended default values. We report parameter counts as a proxy for memory footprint, with StructureFlow being small and training on CPU within 32 GB of RAM.

Table 10: Model wall-clock time on the Trifurcating (**TF**) BoolODE system. We include parameter counts for all neural network-based methods. We use $(\cdot)$ to denote runtime of EOT.

Time [s]	Model	Parameter Size
263 + (3.3)	StructureFlow	20716
1511.18	TIGON	43140
82.01	RF	-
3228.61	NGM-NODE	8008
116.17 + (3.3)	Flow matching	8008
208.58 + (3.3)	$[\text{SF}]^2$M	14892
131.22	OTVelo	-
5771.41	SDFL	-

E.6 Additional Ablations of Hyperparameters

In order to assess the stability of our method we vary three hyperparameters that are crucial to the performance of our model: α, which varies the weight of the score and flow term; group lasso regularization which determines how sparse the adjacency matrix recovered by the autonomous flow term is; and the size of the hidden dimensions in our flow module (We use one layer in our case).

Table 11: Ablation over the group lasso regularization term.

group lasso Reg.	0.0	0.0001	0.001	0.01	0.04	0.1	0.2	0.5
AUROC	0.943 ± 0.004	0.946 ± 0.004	0.944 ± 0.004	0.952 ± 0.002	**0.971±0.008**	0.853 ± 0.032	0.851 ± 0.017	0.879 ± 0.016
AP	0.804 ± 0.014	0.816 ± 0.015	0.805 ± 0.014	0.833 ± 0.008	**0.918±0.026**	0.677 ± 0.032	0.688 ± 0.015	0.708 ± 0.021
W_2	0.7623 ± 0.0047	0.7623 ± 0.0030	**0.7608±0.0050**	0.7665 ± 0.0077	0.7886 ± 0.0048	0.8329 ± 0.0049	0.8398 ± 0.0059	0.8414 ± 0.0058

Table 12: Ablation over α. $\alpha = 1$ pertains to only using the score model. $\alpha = 0$ pertains to only using flow model.

α	0.1	0.2	0.5	0.8
AUROC	**0.971±0.008**	0.965 ± 0.007	**0.971±0.009**	0.937 ± 0.005
AP	0.917 ± 0.025	0.886 ± 0.026	**0.918±0.040**	0.851 ± 0.011
W_2	0.7886±0.0048	**0.7814±0.0007**	0.7887±0.0035	0.8085±0.0042

Table 13: Ablation over size of hidden dimension (# of neurons) per layer for the NGM flow model.

Hidden dimension for NGM	10	50	100	200	256
AUROC	0.925 ± 0.010	0.964 ± 0.004	0.971 ± 0.008	0.969 ± 0.003	**0.975±0.003**
AP	0.832 ± 0.020	0.910 ± 0.013	0.917 ± 0.025	0.916 ± 0.014	**0.932±0.012**
W_2	0.7950±0.0024	**0.7853±0.0026**	0.7886±0.0048	0.7972±0.0055	0.7858±0.0208

E.7 Assessing the stability of the score and flow network in StructureFlow

Here we test two settings where we try freezing the weights of one of the modules. We jointly train s_ϕ and v_θ for $K \in \{10000, 15000\}$ iterations, then freeze the parameters of s_ϕ and continue training only v_θ for an additional $M = 5000$ iterations. We do the same experiment but freeze the flow term instead. In both settings our performance stays constant for structure learning, but is more sensitive for dynamics.

Table 14: Ablation over freezing flow and score model post-training when models pre-trained for 10000 iterations.

Freeze model	AP	AUROC	W_2
score	0.9011	0.9681	1.5462 ± 0.0520
flow	0.9109	0.9700	0.9366 ± 0.1992

Table 15: Ablation over freezing flow and score model post-training when models pre-trained for 15000 iterations.

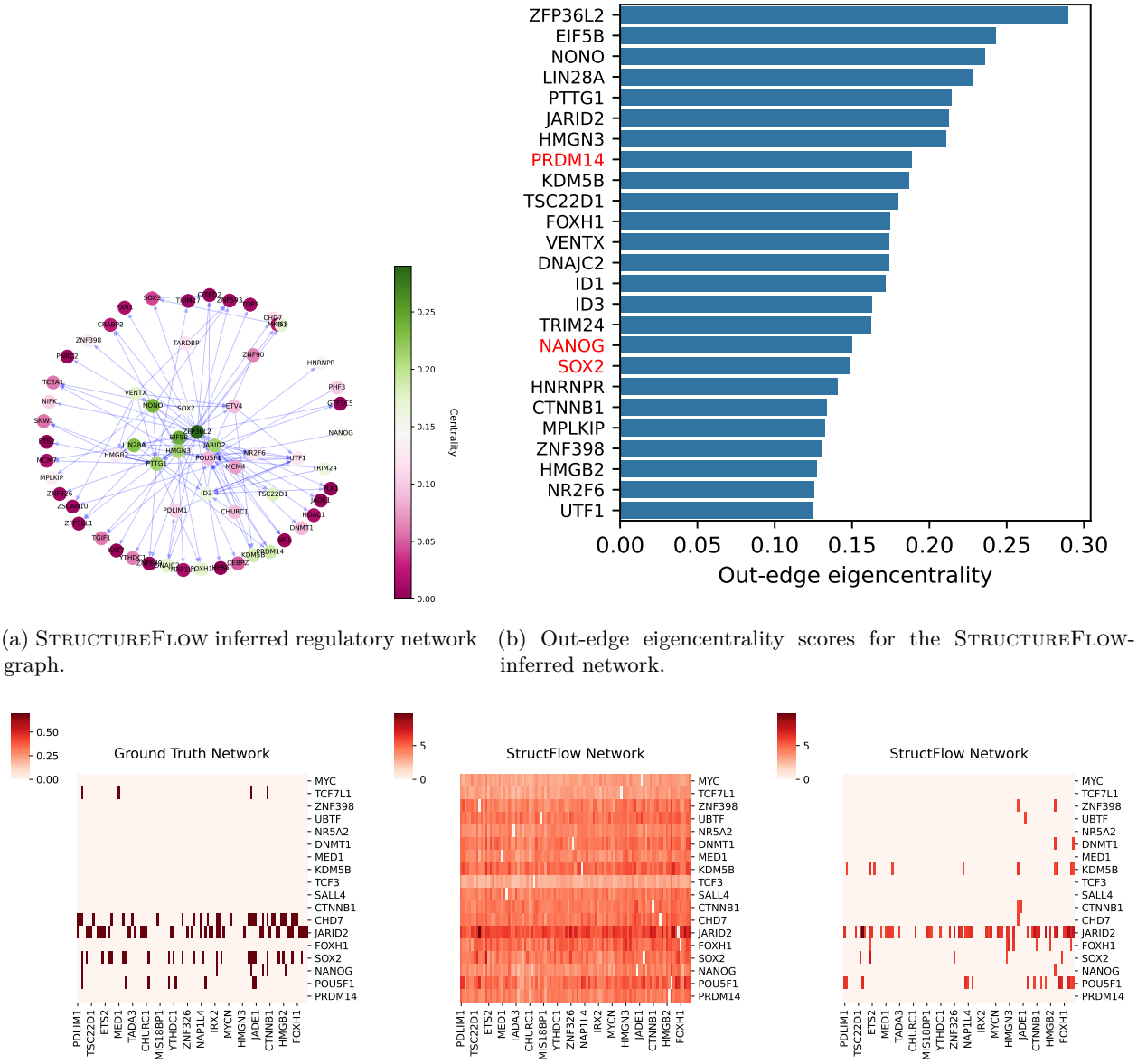
Freeze model	AP	AUROC	W_2
score	0.9019	0.9668	1.5200 ± 0.0569
flow	0.8843	0.9650	0.9337 ± 0.2181

E.8 Scaling and runtime of various baselines and our method on Renge data

Table 16: Model wall-clock time on the Real (Renge) dataset. We include parameter counts for all neural network-based methods. We note that for StructureFlow and $[SF]^2M$ the OT pairing was negligble and thus not included.

Time [s]	Model	Parameter Size
319.31 ± 21.29	StructureFlow	1440646
1542.88 ± 26.67	TIGON (PCA)	43140
14112.49	TIGON (Ambient)	409448
338.12 ± 0.78	RF	-
101.31 ± 1.43	$[SF]^2M$	175310
12.21 ± .125	OTVelo	-

E.9 Structure Experiments for Real Data



(a) StructureFlow inferred regulatory network graph.

(b) Out-edge eigencentrality scores for the StructureFlow-inferred network.

(c) Ground truth regulatory network (heatmap).

(d) StructureFlow-inferred network (no threshold).

(e) StructureFlow-inferred network (99th quantile threshold).

Figure 13: **Analysis of gene regulatory network inference results.** **(a)** StructureFlow graph representation of the inferred gene regulatory network. **(b)** The top 25 Centrality scores (out-edge eigencentrality) for nodes in the StructureFlow-inferred network. **(c)** Ground truth regulatory network shown as a heatmap. This network is from the experimental binding information from the ChIP-atlas. **(d)** StructureFlow-inferred network adjacency matrix (no thresholding). **(e)** StructureFlow-inferred network thresholded at the 99th quantile to highlight the strongest edges.

Table 18: **StructureFlow achieves the best W_2 among joint methods for *dynamical inference* on simulated biological systems.** Shown is a comparison of dynamical inference methods for learning conditional population dynamics across synthetic biological systems. We include [SF]^{2}M for comparison, but note that [SF]^{2}M does not infer underlying network structure (i.e. cannot address joint task). Colors indicate **1st best performer** and 2nd best performer for models that perform the joint inference task.

	TF		CY		LL		SW		BF		HSC (Curated)	
	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD	W_2	MMD
[SF]^{2}M	0.761 ± 0.014	0.134 ± 0.005	0.517 ± 0.015	0.153 ± 0.002	0.847 ± 0.053	0.190 ± 0.010	0.572 ± 0.008	0.215 ± 0.005	0.660 ± 0.007	0.140 ± 0.004	0.664 ± 0.011	0.083 ± 0.003
RF	1.376 ± 0.000	0.191 ± 0.000	2.086 ± 0.000	0.326 ± 0.000	2.120 ± 0.007	0.296 ± 0.000	1.371 ± 0.000	0.290 ± 0.000	1.495 ± 0.000	0.242 ± 0.000	0.950 ± 0.000	0.066 ± 0.000
OTVelo	1.874 ± 0.262	0.582 ± 0.039	1.972 ± 0.144	0.622 ± 0.034	2.878 ± 0.361	0.647 ± 0.015	1.940 ± 0.144	0.577 ± 0.022	1.847 ± 0.277	0.591 ± 0.042	1.942 ± 0.006	0.603 ± 0.002
TIGON	1.035 ± 0.002	**0.063 ± 0.022**	0.762 ± 0.001	**0.044 ± 0.003**	1.853 ± 0.001	**0.066 ± 0.039**	0.669 ± 0.190	**0.017 ± 0.009**	0.730 ± 0.006	**0.066 ± 0.021**	0.730 ± 0.006	**0.012 ± 0.014**
StructureFlow	**0.789 ± 0.012**	0.135 ± 0.004	**0.577 ± 0.012**	0.140 ± 0.004	**0.842 ± 0.031**	0.196 ± 0.008	**0.610 ± 0.010**	0.220 ± 0.006	**0.694 ± 0.017**	0.142 ± 0.007	**0.683 ± 0.011**	0.068 ± 0.003

E.10 Evaluating StructureFlow with Irregularly Sampled Time-Points

We evaluate StructureFlow and Reference Fitting on the TF BoolODE system using population snapshots from non-uniform timepoints. To accomplish this, we bin the continuous timepoint data into randomly non-uniformly spaced time bins with a standard deviation of 30% of the time bin width (whereas our original experiments bin into discrete equally spaced timebins of $t = 1, 2, 3, \ldots$). We run this experiment over 3 seeds, where the seeding randomizes the time bin spacing (making the data non-deterministic between runs, and thus RF is non-deterministic in this specific setting).

Table 17: Comparison of StructureFlow and RF on irregularly sampled time-points for the TF system.

Metric	StructureFlow	RF
GRN Graph AP	0.8501 ± 0.0348	0.9614 ± 0.0282
GRN Graph AUROC	0.9326 ± 0.0180	0.9812 ± 0.0099
Trajectory W_2	0.8491 ± 0.0168	1.4060 ± 0.0665
Trajectory MMD	0.1408 ± 0.0022	0.1977 ± 0.0152

E.11 Comparison with Symbolic Distribution Flow Learning (SDFL)

We compare StructureFlow to Symbolic Distribution Flow Learning (SDFL) (Dakhmouche et al., 2025), which tackles the population dynamics problem through learning explicit symbolic equations. While SDFL provides interpretable outputs through per-dimension equations, it requires significant computation time to run and cannot explicitly recover a learned matrix structure for the structure learning task. We evaluate SDFL on the trifurcating (TF) system for the trajectory inference task and report results below (we include StructureFlow results from Table 1 for easy comparison). Note due to the methods training instability, we were unable to get 3 consecutive successful seeds, and therefore report only 1 seed result.

Table 19: Comparison of dynamical inference performance with SDFL on the TF system.

Method	W_2	MMD
SDFL	3.4880	0.0785
StructureFlow	0.789 ± 0.012	0.135 ± 0.004

E.12 Full Results for Renge (real) Data Results

We provide the full results across held-out timepoints 1 and 2 for the *dynamical inference* of left-out marginals on the real (Renge) biological dataset.

Table 20: **StructureFlow shows competitive performance for *dynamical inference* of left-out marginals on the real (Renge) biological dataset.** We show a comparison of dynamical inference methods for learning conditional population dynamics across left-out timepoints $k = 1, 2$ on the Renge dataset. We report results over withheld k as well as the average of the two using Wasserstein-2 (W_2) and Energy Distance (ED) (Rizzo and Székely, 2016) metrics. Bold indicates **1st best performer** and the underline indicates 2nd best performer for models that perform the joint inference task.

	Timepoint 1		Timepoint 2		Average	
	W_2	ED	W_2	ED	W_2	ED
[SF]^{2}M	5.673 ± 0.034	0.533 ± 0.024	5.841 ± 0.030	0.524 ± 0.024	5.757 ± 0.032	0.528 ± 0.024
RF	10.193 ± 0.004	12.389 ± 0.009	9.242 ± 0.004	10.089 ± 0.008	9.717 ± 0.004	11.239 ± 0.008
OTVelo	7.106 ± 0.000	4.269 ± 0.000	6.996 ± 0.000	3.658 ± 0.000	7.051 ± 0.000	3.963 ± 0.000
TIGON	6.401 ± 0.033	**0.173 ± 0.011**	6.402 ± 0.002	**0.125 ± 0.001**	6.402 ± 0.001	**0.149 ± 0.024**
StructureFlow	**5.589 ± 0.026**	0.692 ± 0.048	**5.679 ± 0.029**	0.765 ± 0.025	**5.634 ± 0.027**	0.729 ± 0.037

Table 21: **StructureFlow shows competitive performance for *dynamical inference* of left-out interventions on the real (Renge) biological dataset.** We show a comparison of dynamical inference methods for learning conditional population dynamics on the Renge dataset for left-out interventions. We report results for 3 left-out gene knockout conditions using Wasserstein-2 (W_2) and Energy Distance (ED) metrics, as well as the average performance across conditions. **Bold** indicates the best performing method that can perform the joint task. We provide a visualization of this result in Appendix E fig. 12.

	NANOG		POU5F1		PRDM14		Average	
	W_2	ED	W_2	ED	W_2	ED	W_2	ED
[SF]^{2}M	5.726 ± 0.015	0.467 ± 0.014	5.931 ± 0.024	0.826 ± 0.020	6.002 ± 0.043	0.639 ± 0.030	5.886 ± 0.027	0.644 ± 0.021
RF	9.859 ± 0.002	11.492 ± 0.006	10.211 ± 0.002	12.140 ± 0.004	9.889 ± 0.005	11.461 ± 0.013	9.986 ± 0.003	11.698 ± 0.008
StructureFlow	**5.517 ± 0.041**	**0.818 ± 0.015**	**5.717 ± 0.021**	**1.136 ± 0.039**	**5.816 ± 0.023**	**0.973 ± 0.053**	**5.683 ± 0.028**	**0.976 ± 0.036**

E.13 PCMCI Adapted to Snapshot Settings

Table 22: **StructureFlow achieves higher AUPRC than the adapted PCMCI baseline in the snapshot setting.** We adapt PCMCI to the snapshot regime by constructing pseudo-longitudinal trajectories sampled from OT couplings between successive marginals. Results are reported on the trifurcating (TF) and cyclical (CY) BoolODE systems, in both the observational (obs) and observational + interventional (obs+int) settings.

Model	Dataset	AUPRC observational	AUPRC obs. + interventional
PCMCI	TF	0.29	0.36
StructureFlow	TF	**0.60**	**0.92**
PCMCI	CY	0.22	0.21
StructureFlow	CY	**0.47**	**0.80**