Supplementary Notes for "SDEvelo: a deep generative approach for transcriptional dynamics with cell-specific latent time and multivariate stochastic modeling"

Xu Liao[1†], Lican Kang[2†], Yihao Peng[1], Xiaoran Chai[3], Chengqi Lin[4],

Hongkai Ji[5], Yuling Jiao[6*], and Jin Liu[1*]

[1]School of Data Science, The Chinese University of Hong Kong-Shenzhen, Shenzhen, China,

[2]Cardiovascular and Metabolic Disorders Program, Duke-NUS Medical School, Singapore,

[3]Cancer and Stem Cell Biology Program, Duke-NUS Medical School, Singapore,

[4]Key Laboratory of Developmental Genes and Human Disease, School of Life Science and Technology, Southeast University, Nanjing, China,

[5]Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, USA,

[6]School of Mathematics and Statistics, Wuhan University, Wuhan, China.

[†]Equal contributions.

[*]Corresponding author. Email: yulingjiaomath@whu.edu.cn, liujinlab@cuhk.edu.cn.

Contents

In the supplementary notes, we detail the proposed SDEvelo method and provide additional insights into its implementation, including initialization, parameter optimization, and hyper-parameter settings for the numerical experiments. Furthermore, we elaborate on the simulation setting details for the SDE-based model and deep dimension reduction (DDR) techniques [1] for downstream tasks, including pseudo-code of the algorithm, network architectures, SGD optimizers [2, 3], and hyper-parameter configurations used. In general, we offered an exhaustive description of the methodologies and computational strategies deployed in SDEvelo, ensuring clarity and reproducibility.

1 Details for SDEvelo

In this section, we detail the generation of SDE trajectories using the SDEvelo method, emphasizing the Euler-Maruyama method [4, 5] for simulations. We revisit the SDEvelo model's fundamentals, including the training parameters and their biological implications, empirical objective functions, optimization algorithms like SGD and Adam [6] and the algorithm for parameter iterations. We further demonstrated parameter initialization, detailed optimization strategies, and the setting of hyper-parameters in our experiments.

1.1 SDE trajectories from SDEvelo generated by Euler-Maruyama method

First we revisit our SDEvelo model which characterized the stochastic dynamic kinetics for multiple genes:

$$\begin{aligned} d\boldsymbol{U}(t) &= (\boldsymbol{\alpha}(t) - \boldsymbol{\beta} \odot \boldsymbol{U}(t))dt + \boldsymbol{\sigma}_1 d\boldsymbol{B}(t), \\ d\boldsymbol{S}(t) &= (\boldsymbol{\beta} \odot \boldsymbol{U}(t) - \boldsymbol{\gamma} \odot \boldsymbol{S}(t))dt + \boldsymbol{\sigma}_2 d\boldsymbol{B}(t), \end{aligned} \tag{1}$$

where, $\boldsymbol{U}(t)$ represents the normalized vector of unspliced mRNA counts, while $\boldsymbol{S}(t)$ denotes the normalized vector of spliced mRNA counts, both measured across p genes over a shared latent time t. The dynamic of these vectors is governed by the transcription, splicing, and degradation rates, denoted by $\boldsymbol{\alpha}(t)$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$, respectively. Additionally, the model incorporates stochastic elements through a Wiener process, $\boldsymbol{B}(t)$, to account for intrinsic and extrinsic biological variability. The parameters $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ quantify the noise levels associated with unspliced and spliced dynamics, respectively.

To parameterize the multivariate SDE model and sample the generated trajectories from the SDE model in the current step l, we utilized several parameters and hyper-parameters to characterize the dynamic kinetics:

- Function with parameters $\boldsymbol{\alpha}(t)$: The transcription rate function, directly linked to the transcription dynamics of mRNA reads.

$$\alpha(t)_i = \frac{c_i}{1 + \exp\{b(t-a)\}}, \quad i = 1, \ldots, p. \tag{2}$$

  - Parameter $\boldsymbol{c} = (c_1, \ldots, c_p)$: Represents the transcription rate intensity.
  - Parameter $\boldsymbol{a} = (a_1, \ldots, a_p)$: is a vector of the switch time points.
  - Hyper-parameter b: A scaling value to control the shape of the Sigmoid function.

- Parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$: Splicing and degradation rate vectors, crucial for modeling the transition from spliced to unspliced mRNA and degradation of mRNA.

- Parameter $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$: Noise level vectors for unspliced and spliced mRNA reads, capturing the model's stochastic nature.

- Parameter $\boldsymbol{u}_0$ and $\boldsymbol{s}_0$: Initial conditions for the unspliced and spliced mRNA dynamics.

- Parameter $\boldsymbol{u}_{\text{shift}}$ and $\boldsymbol{s}_{\text{shift}}$: Adjustment parameters for initial conditions $\boldsymbol{u}_0$ and $\boldsymbol{s}_0$, ensuring stability of SDEvelo model.

- Hyper-parameter h: Step size for numerical Euler-Maruyama method, default set is 0.01 for the discretization of the model.

In practice, we can formulate the iteration for step l with the given parameters, which can be mathematically expressed as follows:

$$\begin{aligned}
\boldsymbol{U}^{(l)}(t+h) &= \boldsymbol{U}^{(l)}(t) + \left(\frac{\boldsymbol{c}^{(l)}}{1 + \exp(\boldsymbol{b} \cdot (t - \boldsymbol{a}^{(l)}))} - \boldsymbol{\beta}^{(l)} \odot \boldsymbol{U}^{(l)}(t) \right) \cdot h + \boldsymbol{\sigma}_1^{(l)} \cdot \sqrt{h} \cdot \boldsymbol{Z}_1, \\
\boldsymbol{S}^{(l)}(t+h) &= \boldsymbol{S}^{(l)}(t) + (\boldsymbol{\beta}^{(l)} \odot \boldsymbol{U}^{(l)}(t) - \boldsymbol{\gamma}^{(l)} \odot \boldsymbol{S}^{(l)}(t)) \cdot h + \boldsymbol{\sigma}_2^{(l)} \cdot \sqrt{h} \cdot \boldsymbol{Z}_2,
\end{aligned}$$

where h denotes the Euler-Maruyama discretization step size, and $\boldsymbol{Z}_1, \boldsymbol{Z}_2 \sim \mathcal{N}(0, I_p)$ are p-dimensional vectors of independent standard normal random variables representing the intrinsic and extrinsic stochastic fluctuations in unspliced and spliced mRNA abundance, respectively. The whole trajectories for unspliced and spliced data in step l were generated from $t = 0$ to $t = 1$ with the step size h. Given the initial conditions $\boldsymbol{U}^{(l)}(0) = \boldsymbol{u}_0 + \boldsymbol{u}_{\text{shift}}^{(l)}$ and $\boldsymbol{S}^{(l)}(0) = \boldsymbol{s}_0 + \boldsymbol{s}_{\text{shift}}^{(l)}$. We defined the generated whole trajectories as $\boldsymbol{U}^{(l)}$ and $\boldsymbol{S}^{(l)}$ on the step l and used the generated data to calculate the objective function.

Therefore, the complete set of parameters for step l was defined as:

$$\Theta^{(l)} = \left\{ \boldsymbol{c}^{(l)}, \boldsymbol{a}^{(l)}, \boldsymbol{\beta}^{(l)}, \boldsymbol{\gamma}^{(l)}, \boldsymbol{\sigma}_1^{(l)}, \boldsymbol{\sigma}_2^{(l)}, \boldsymbol{u}_{\text{shift}}^{(l)}, \boldsymbol{s}_{\text{shift}}^{(l)}, \boldsymbol{u}_0, \boldsymbol{s}_0 \right\}. \tag{3}$$

These parameters were critical to update generated trajectories that described the dynamics of unspliced ($\boldsymbol{U}^{(l)}$) and spliced ($\boldsymbol{S}^{(l)}$) mRNA abundance over time, from $t = 0$ to $t = 1$, with a Euler-Maruyama discretization step size h.

1.2 Objective function and parameters updates

We aim to estimate $\Theta^{(l)}$ by minimizing the empirical MMD objective, which is a measure of the distributional discrepancy between the generated trajectories and the real data samples. In step l, the MMD loss with incorporating a kernel trick was formulated as:

$$\begin{aligned}\mathcal{L}_{\mathrm{MMD}^2}(\Theta^{(l)}) = & \frac{1}{{N_g}^2}\sum_{i=1}^{N_g}\sum_{i'=1}^{N_g} k\left(g_i(\Theta^{(l)}), g_{i'}(\Theta^{(l)})\right) \\ & - \frac{2}{N_g N_r}\sum_{i=1}^{N_g}\sum_{j=1}^{N_r} k\left(g_i(\Theta^{(l)}), r_j\right) \\ & + \frac{1}{{N_r}^2}\sum_{j=1}^{N_r}\sum_{j'=1}^{N_r} k\left(r_j, r_{j'}\right).\end{aligned} \tag{4}$$

where:

- $\boldsymbol{G}(\Theta^{(l)}) = \{g_i(\Theta^{(l)})\}_{i=1}^{N_g} = \{[\boldsymbol{U}_{i\cdot}^{(l)}, \boldsymbol{S}_{i\cdot}^{(l)}]\}_{i=1}^{N_g}$ is defined as the collection of sample vectors obtained from the trajectories of SDE at the lth iteration; similarly, $\boldsymbol{R} = \{r_j\}_{j=1}^{N_r} = \{[\boldsymbol{U}_{j\cdot}^{R}, \boldsymbol{S}_{j\cdot}^{R}]\}_{j=1}^{N_r}$ represents the set of real data samples.
- Each $g_i \in \mathbb{R}^{2p}$ is a vector which concatenates generated unspliced and spliced data for p genes in the ith cell. Likewise, each r_j in $\mathbb{R}^{2p}$ is a vector representing the real data for the corresponding genes.
- For the kernel function k, we employed as $k(x, x') = \sum_{q=1}^{K} \exp\left(-\frac{1}{2w_q}|x - x'|^2\right)$, which integrated different bandwidth parameters w_q to capture the diverse scales of the data distribution effectively.

The goal function for SDEvelo is defined as $\mathcal{L}_{\mathrm{MMD}}\left(\Theta^{(l)}\right) = \sqrt{\mathcal{L}_{\mathrm{MMD}^2}\left(\Theta^{(l)}\right)}$, which aims to optimize the SDE trajectories.

Given the differentiable nature of the MMD loss function (4), we employ SGD to efficiently estimate the parameters $\Theta^{(l)}$. The update form at lth-iteration for each parameter $\theta \in \Theta^{(l)}$ can be mathematically expressed as follows:

$$\theta^{(l+1)} = \theta^{(l)} - \eta \cdot \nabla_\theta \mathcal{L}_{\mathrm{MMD}}(\Theta^{(l)}), \tag{5}$$

where:

- $\theta^{(l+1)}$ and $\theta^{(l)}$ are the values of the parameter θ after and before the update, respectively.
- η represents a positive scalar learning rate, indicating the step size in the direction of the negative gradient.
- $\nabla_\theta \mathcal{L}_{\mathrm{MMD}}(\Theta^{(l)})$ denotes the gradient of the MMD loss function with respect to the parameter θ at iteration l, implemented by the backpropagation algorithm using the PyTorch platform.

To improve parameter optimization for the complex transcription dynamics, we employed the Adam optimization algorithm. Adam has ability to enhance parameter-specific learning rates through estimating gradient moments, accelerating convergence and further mitigate the need for a constant learning rate. The update for $\Theta^{(l)}$ with Adam at iteration l can be expressed as:

$$\iota_\theta^{(l+1)} = \tau_1 \cdot \iota_\theta^{(l)} + (1-\tau_1) \cdot \nabla_\theta \mathcal{L}_{\text{MMD}}(\Theta^{(l)}), \tag{6}$$

$$\vartheta_\theta^{(l+1)} = \tau_2 \cdot \vartheta_\theta^{(l)} + (1-\tau_2) \cdot \left(\nabla_\theta \mathcal{L}_{\text{MMD}}(\Theta^{(l)})\right)^2, \tag{7}$$

$$\hat{\iota}_\theta = \frac{\iota_\theta^{(l+1)}}{1-\tau_1^{l+1}}, \tag{8}$$

$$\hat{\vartheta}_\theta = \frac{\vartheta_\theta^{(l+1)}}{1-\tau_2^{l+1}}, \tag{9}$$

$$\theta^{(l+1)} = \theta^{(l)} - \frac{\eta}{\sqrt{\hat{\vartheta}_\theta} + \epsilon} \cdot \hat{\iota}_\theta, \tag{10}$$

where:

- $\iota_\theta^{(l)}$ and $\vartheta_\theta^{(l)}$ are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients, respectively.
- τ_1 and τ_2 are exponential decay rates for these moment estimates, typically close to 1.
- $\hat{\iota}_\theta$ and $\hat{\vartheta}_\theta$ are bias-corrected versions of $\iota_\theta^{(l+1)}$ and $\vartheta_\theta^{(l+1)}$, respectively.
- η is the positive learning rate, and ϵ is a small scalar to improve numerical stability.

In practice, we employed values as $\tau_1 = 0.9$, $\tau_2 = 0.999$, and $\epsilon = 10^{-8}$ in all numerical experiments, providing a trade-off between momentum and adaptability.

1.3 Algorithm for SDEvelo

This subsection introduces the SDEvelo algorithm with the pseudo-code. It includes initialization, SDE trajectory simulation with the Euler-Maruyama method, parameter optimization through MMD loss minimization, and Adam optimizer updates, iterating to update parameters. The pseudo-code for the SDEvelo algorithm is given below:

Pseudo-code for the SDEvelo algorithm

- Input real samples of unspliced and spliced mRNA abundance for n cells and p genes: $\{\boldsymbol{U}^R, \boldsymbol{S}^R\}$. Tuning hyper-parameters: learning rate η, weight decay rates τ_1, τ_2, scaling vector $\boldsymbol{b}$, Euler-Maruyama discretization step size h.
- *Initialization*:
  - Initialize parameters $\Theta^{(0)} = \{\boldsymbol{c}^{(0)}, \boldsymbol{a}^{(0)}, \boldsymbol{\beta}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{\sigma}_1^{(0)}, \boldsymbol{\sigma}_2^{(0)}, \boldsymbol{u}_{\text{shift}}^{(0)}, \boldsymbol{s}_{\text{shift}}^{(0)}, \boldsymbol{u}_0, \boldsymbol{s}_0\}$.
  - Set initial conditions $\boldsymbol{U}^{(0)}(0) = \boldsymbol{u}_0 + \boldsymbol{u}_{\text{shift}}^{(0)}$ and $\boldsymbol{S}^{(0)}(0) = \boldsymbol{s}_0 + \boldsymbol{s}_{\text{shift}}^{(0)}$.
- *For each iteration l until convergence*:

- Simulate $\{\boldsymbol{U}^{(l)}(t), \boldsymbol{S}^{(l)}(t)\}$ using Euler-Maruyama method based on current $\Theta^{(l)}$.
- Calculate MMD loss $\mathcal{L}_{\text{MMD}}(\Theta^{(l)})$ using Equation (4).
- *Parameter update using SGD or Adam*:
  - Calculate gradients $\nabla_\theta \mathcal{L}_{\text{MMD}}(\Theta^{(l)})$ for each $\theta \in \Theta^{(l)}$.
  - Update $\Theta^{(l+1)}$ using Adam optimizers with gradients, learning rate η, and weight decay rates τ_1, τ_2.
- Update parameters in $\Theta^{(l+1)}$ by clipping the values.

- *End of iteration*
- Output optimized parameters $\Theta^{(L)} = \{\boldsymbol{c}^{(L)}, \boldsymbol{a}^{(L)}, \boldsymbol{\beta}^{(L)}, \boldsymbol{\gamma}^{(L)}, \boldsymbol{\sigma}_1^{(L)}, \boldsymbol{\sigma}_2^{(L)}, \boldsymbol{u}_{\text{shift}}^{(L)}, \boldsymbol{s}_{\text{shift}}^{(L)}, \boldsymbol{u}_0, \boldsymbol{s}_0\}$ and fitted mRNA dynamics $\{\boldsymbol{U}^{(L)}(t), \boldsymbol{S}^{(L)}(t)\}$.

1.4 Implementation details for SDEvelo

In this subsection, we presented the implementation details of SDEvelo, including parameter initialization, optimization strategies, and dataset details. It includes parameter summaries, initialization strategies, and optimization approaches, focusing on computational efficiency and model accuracy. We also elaborated the techniques involved during the training process, including subsampling, mini-batch techniques, and the optimizer scheduler for computational efficiency. We further summarized the dataset details for clarity, covering species platform, and running time costs.

Subsampling and mini batch. In the pipeline of SDEvelo, we randomly subsampled $5,000$ cells as input of the parameter updating process to enhance the computational stability and efficiency. During the training process, we used the mini batch with default size 200 to ensure computational efficiency and alleviate the requirement for the memory usage.

Initialization. We initialized transcription-splicing-degradation rate as $\boldsymbol{c}^{(0)} = \boldsymbol{\beta}^{(0)} = \boldsymbol{\gamma}^{(0)} = \max_{\text{row}} \left(\boldsymbol{U}^R\right)$, and noise levels for unspliced and spliced stochastic process as $\boldsymbol{\sigma_1} = \boldsymbol{\sigma_2} = 0.01\mathbf{1}_p$. For the switch time point, we initialized

$$\boldsymbol{a}^{(0)} = \frac{\sum_{i=1}^{n} \mathbf{1}\left(U_i > S_i \boldsymbol{\beta}_{\text{robust}}^{\top}\right)}{n_{\text{cell}}},$$

where $\boldsymbol{\beta}_{\text{robust}}$ represents the slope of robust quantile regression fitting the 5th and 95th percentiles.

For each gene j, we calculated the Euclidean distance d_{ij} of each cell i 's spliced and unspliced counts from the origin:

$$\boldsymbol{d}_{ij} = \sqrt{\boldsymbol{s}_{ij}^2 + \boldsymbol{u}_{ij}^2}, \quad \forall i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, p\}.$$

For each gene j, find the cell i_j^* that corresponds to the minimum Euclidean distance:

$$i_j^* = \arg\min_i \boldsymbol{d}_{ij}, \quad \forall j \in \{1, 2, \ldots, p\}.$$

For each gene j, we initialized $\boldsymbol{s}_{0j}$ and $\boldsymbol{u}_{0j}$ as:

$$\boldsymbol{s}_{0j} = \boldsymbol{s}_{i_j^* j} \quad \text{and} \quad \boldsymbol{u}_{0j} = \boldsymbol{u}_{i_j^* j}, \quad \forall j \in \{1, 2, \ldots, p\}.$$

Thus, the corresponding initialization $\boldsymbol{s}_0 = [\boldsymbol{s}_{01}, \boldsymbol{s}_{02}, \ldots, \boldsymbol{s}_{0p}]^T$ and $\boldsymbol{u}_0 = [\boldsymbol{u}_{01}, \boldsymbol{u}_{02}, \ldots, \boldsymbol{u}_{0p}]^T$ were obtained. In summary, the details of initialization for other parameters and default values of hyper parameters can be found in Table S1 and Table S2, respectively.

Table S1: Summary of SDEvelo parameters and their initialization and optimization

Parameter	**Description**	**Initialization**	**Optimization**	**Clipping**
$\boldsymbol{a}$	Switch time point	$\frac{\sum_{i=1}^{n_{\text{cell}}} 1\left(U_i > S_i \boldsymbol{\beta}_{\text{robust}}^{\top}\right)}{n_{\text{cell}}}$	Adam (0.1)	$[0.1, 1]$
$\boldsymbol{c}$	Transcription rate	$\max_{\text{row}}\left(\boldsymbol{U}^R\right)$	Adam (0.5)	$[0.01, 1000]$
$\boldsymbol{\beta}$	Splicing rate	$\max_{\text{row}}\left(\boldsymbol{U}^R\right)$	Adam (0.5)	$[0.01, 1000]$
$\boldsymbol{\gamma}$	Degradation rate	$\max_{\text{row}}\left(\boldsymbol{U}^R\right)$	Adam (0.5)	$[0.01, 100]$
$\boldsymbol{\sigma}_1$, $\boldsymbol{\sigma}_2$	Noise levels	$0.01\mathbf{1}_p$	Adam (0.1)	$[0.01, 10]$
$\boldsymbol{u}_{\text{shift}}$, $\boldsymbol{s}_{\text{shift}}$	Adjustments for initial conditions	Zeros	Adam (0.1)	$[-10, 10]$

Table S2: Summary of SDEvelo's hyper parameters

Parameter	**Description**	**Default value**
b	Scaling hyper parameter to control the Sigmoid shape	100
h	Step size of the Euler-Maruyama discretization	0.01

Optimization. In our model, we utilized the Adam optimizer with an initial learning rates listed in Table S1. To refine the optimization, we applied a torch.optim.lr_scheduler.MultiStepLR function scheduler with a milestone at the 200-th epoch, where the learning rate will be reduced to 10% of the current rate when updating $\left\{\boldsymbol{c}^{(l)}, \boldsymbol{\beta}^{(l)}, \boldsymbol{\gamma}^{(l)}\right\}$.

Velocity calculation. The observed unspliced and spliced mRNA data are directly utilized, and noise is introduced to reflect the stochastic nature of biological systems, the RNA velocity at iteration L is calculated by:

$$\text{RNA Velocity } = \left(\boldsymbol{U}^R \boldsymbol{\beta}^{(L)\top} - \boldsymbol{S}^R \boldsymbol{\gamma}^{(L)\top}\right) \cdot h + \boldsymbol{\sigma_2}^{(L)} \sqrt{h} \cdot \boldsymbol{Z},$$

where h denotes the Euler-Maruyama discretization step size, and $\boldsymbol{Z} \sim \mathcal{N}(0, \boldsymbol{I}_{n \times p})$.

Datasets. Table S3 outlines the characteristics of the datasets used in this study, including data species, sequencing platforms, dataset dimensions as input of SDEvelo, applied dimension reduction (DR) techniques, and processing times with the proposed SDEvelo pipeline.

2 Simulated gene dynamics using an SDE-Based model

In this section, we introduce SDe-based model to simulate dynamics, including formulation for unspliced and spliced mRNA and settings of parameter distributions in the simulation setting.

2.1 Model formulation

First, we revisit our simulated SDE model. For each gene i within a single cell trajectory k, the temporal dynamics are characterized by the levels of unspliced mRNA, $U_{i,k}(t)$, and spliced

Table S3: Summary of datasets and their characteristics

Dataset	**Species**	**Platform**	**Dimension**	**DR**	**Time**
PBMC	Human	10x Genomics	$65,877 \times 601$	t-SNE	780 seconds
HCC1	Human	Visium	$2,982 \times 1,985$	PCA	606 seconds
HCC2	Human	Visium	$1,334 \times 1,985$	PCA	309 seconds
HCC3	Human	Visium	$2,732 \times 1,985$	PCA	574 seconds
HCC4	Human	Visium	$2,764 \times 1,985$	PCA	405 seconds
Reprogramming	Mouse	10x Genomics	$85,010 \times 2,000$	PCA	798 seconds
Erythroid	Mouse	10x Genomics	$9,815 \times 2,000$	UMAP	$1,156$ seconds

mRNA, $S_{i,k}(t)$, governed by the following SDEs:

$$
\begin{aligned}
dU_{i,k}(t) &= \left(\frac{c_i}{1 + \exp(-b \cdot (t - a_i))} - \beta_i U_{i,k}(t) \right) dt + \sigma_{1,i} \sqrt{dt} \cdot \xi_{1,i,k}(t), \\
dS_{i,k}(t) &= \left(\beta_i U_{i,k}(t) - \gamma_i S_{i,k}(t) \right) dt + \sigma_{2,i} \sqrt{dt} \cdot \xi_{2,i,k}(t),
\end{aligned}
$$

where

- a_i represents the gene-specific activation time,
- c_i denotes the transcription rate,
- β_i and γ_i are the rates of mRNA splicing and degradation, respectively,
- $\sigma_{1,i}$ and $\sigma_{2,i}$ are the noise intensities associated with unspliced and spliced mRNA production,
- $\xi_{1,i,k}(t)$ and $\xi_{2,i,k}(t)$ are independent standard Wiener processes.

2.2 Parameter distributions

In our simulated experiments, we utilized SDE-based models with predefined parameters, ensuring the capture of biological variability and stochasticity. Parameters were sampled from uniform distributions as detailed below:

- Switch time point $\boldsymbol{a}_i \sim \mathcal{U}(0.2, 0.8)$,
- Transcription rates $\boldsymbol{c}_i \sim \mathcal{U}(2, 12)$,
- Splicing rates $\boldsymbol{\beta}_i \sim \mathcal{U}(2, 12)$,
- Degradation rates $\boldsymbol{\gamma}_i \sim \mathcal{U}(2, 12)$,
- Noise intensities for unspliced and spliced mRNA, $\boldsymbol{\sigma}_{1,i}$ and $\boldsymbol{\sigma}_{2,i}$, were sampled from $\mathcal{U}(0, 2)$.

3 Incorporate the latent time estimated by SDEvelo using deep dimension reduction

This section delves into integrating deep dimension reduction (DDR) with latent time estimations from SDEvelo for gene expression data analysis. We outline DDR's formulation, emphasizing its capacity to retain, disentangle, and sufficiently encode gene expression information. The DDR process, described via pseudo-code, employs optimization to align representations with a Gaussian distribution. Implementation details, including network architectures and hyper-parameter settings, are provided to facilitate the application of DDR in extracting insights from complex biological data.

3.1 Formulation for deep dimension reduction

We used the deep dimension reduction algorithm to extract the latent time $\mathbf{T} \in \mathbb{R}^{n\times 1}$ relevant information from gene expression data $\mathbf{X} \in \mathbb{R}^{n\times p}$, in the meanwhile we guarantee the sufficiency, promote disentanglement, and maintain the information from gene expression to learn the representation.

For the sufficiency [7–11], we utilized the conditional independence [1, 12–14] to characterize $\mathbf{X} \perp\!\!\!\perp \mathbf{T}|R(\mathbf{X})$, which indicates that $R(\mathbf{X})$ capture all relevant information from gene expression about the latent time estimated from SDEvelo. To maintain the original information of gene expression, we managed to maximize the distance correlation between the gene expression X and the learned representation $R(\mathbf{X})$.

To enhance the robustness and promote the rotation invariance, we constrain the learned representation $R(\mathbf{X})$ to obey the standard Gaussian distribution, that is $R(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. To achieve this, we utilized a divergence measure $\mathbb{D}$, which can quantify the distributional difference between $\mu_{R(\mathbf{X})}$ and the standard normal distribution γ_d. The condition required is that for every measurable function R, the divergence $\mathbb{D}\left(\mu_{R(\mathbf{X})}|\gamma_d\right)$ should be greater than or equal to zero, with equality holding true solely when R is a member of the set $\mathcal{M} = \{R : \mathbb{R}^p \to \mathbb{R}^d | R(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)\}$. Such a constraint is fulfilled by f-divergences, inclusive of the KL-divergence.

Therefore, the optimal function R^* falls within $\mathcal{M}$ precisely when it achieves the minimum of $\mathbb{D}\left(\mu_{R(X)}|\gamma_d\right)$. In practice, we use variational gradient flow framework [15, 16] to establish a process to push the distribution of the learned representation forward a standard normal distribution.

Consider V to represent the degree of nonlinear correlation between two random variables X and Y, characterized under three main aspects: (i) $V[X, Y]$ is always non-negative, achieving a value of zero solely when X and Y are independent; (ii) For every measurable function R, $V[X, Y]$ is always greater than or equal to $V[R(X), Y]$; (iii) The condition $V[X, Y] = V[R(X), Y]$ is met if and only if R is a member of the set $F = \{R : \mathbb{R}^p \to \mathbb{R}^d | Y \perp\!\!\!\perp X\}$.

Therefore, to obtain a sufficient, encoding and disentangle map R^*, we formulated a constrained minimization problem:

$$\text{argmin}_R -\mathcal{V}[R(\mathbf{X}), \mathbf{T}] - \mathcal{V}[R(\mathbf{X}), \mathbf{X}] \text{ subject to } \mathbb{D}\left(\mu_{R(\mathbf{X})} \| \gamma_d\right) = 0.$$

The Lagrangian form of this minimization problem with neural networks pasteurization is

$$\mathcal{L}(\zeta) = -\mathcal{V}[R_\zeta(\mathbf{X}), \mathbf{T}] - \lambda_{gene}\mathcal{V}[R_\zeta(\mathbf{X}), \mathbf{X}] + \lambda_{reg}\mathbb{D}\left(\mu_{R_\zeta(\mathbf{X})} \| \gamma_d\right),$$

where $\lambda_{gene} \geq 0$ and $\lambda_{reg} \geq 0$ are tuning hyper parameters. Those parameters can provide a balance among the sufficiency property, encoding original information and the disentanglement constraint. A small λ_{gene} or λ_{reg} leads to a representation with more information about the latent time estimated from SDEvelo.

3.2 Algorithm for deep dimension reduction

Thus, we can derive the pseudo-code for the DDR algorithm to obtain the sufficient, encoding and disentangle representation from the gene expression profiles, as shown in below:

Pseudo-code for the DDR algorithm designed for SDEvelo

- Input $\{\mathbf{X}_i, \mathbf{T}_i\}_{i=1}^n$. Tuning hyper parameters: $s, \eta, \lambda_{gene}, \lambda_{reg}, d$. Sample $\{\mathbf{W}_i\}_{i=1}^n \sim \gamma_d$.
- *Outer loop for* ζ
  - *Inner loop (particle method)*
    - Let $\mathbf{Z}_i = R_\zeta(\mathbf{X}_i), i = 1, \ldots, n$.
    - Solve

$$\widehat{D}_\psi \in \arg\min_{D_\psi} \frac{1}{n}\sum_{i=1}^n \{\log[1 + \exp(D_\psi(\mathbf{Z}_i))] + \log[1 + \exp(-D_\psi(\mathbf{W}_i))]\}.$$

    - Define the residual map

$$\mathbb{T}(\mathbf{z}) = \mathbf{z} - s\nabla f'(\hat{r}(\mathbf{z}))$$

      with $\hat{r}(\mathbf{z}) = \exp(-\widehat{D}_\psi(\mathbf{z}))$.
    - Update the particles $\mathbf{Z}_i = \mathbb{T}(\mathbf{Z}_i)$, $i = 1, 2, ..., n$.
  - *End inner loop*
  - The gradient of the loss function L with respect to ζ is given by:

$$\begin{aligned}\nabla_\zeta L = &- \nabla_\zeta \widehat{\mathcal{V}}_n\left[R_\zeta(\mathbf{X}), \mathbf{T}\right] - \lambda_{\text{gene}}\nabla_\zeta \widehat{\mathcal{V}}_n\left[R_\zeta(\mathbf{X}), \mathbf{X}\right] \\ &+ \lambda_{\text{reg}}\nabla_\zeta \sum^n \left\|R_\zeta\left(\mathbf{X}_i\right) - \mathbf{Z}_i\right\|^2 / n. \end{aligned} \tag{11}$$

  - Update ζ by moving it in the opposite direction of the gradient, scaled by a learning rate η:

$$\zeta \leftarrow \zeta - \eta\nabla_\zeta L.$$

- *End outer loop*

3.3 Implementation details

This subsection details the implementation of our model using multilayer perceptrons (MLPs) for specific functions and outlines the optimization strategy with the Adam optimizer. It includes architecture specifics, learning rates, and hyper-parameter settings essential for our experiments on HCC data. In detial, MLPs were employed to parameterize D_ψ and R_ζ, as shown in Table S4.

Table S4: MLPs architectures for D_ψ and R_ζ in regression

	D_ψ		R_ζ	
Layers	Details	Output size	Details	Output size
Layer 1	Linear, LeakyReLU	8	Linear, LeakyReLU	d
Layer 2	Linear, LeakyReLU	8		
Layer 3	Linear, LeakyReLU	1		

For the optimization of R_ζ, we utilized the Adam optimizer with an initial learning rate of 0.001 and weight decay of 0. For network D_ψ, the Adam optimizer with an initial learning rate of 10^{-5} and weight decay of 0 was adopted. We summarized the values of hyper-parameters in Table S5.

Table S5: Hyper-parameters for DDR experiments on HCC data

	Description	HCC1	HCC2	HCC3	HCC4
λ_{reg}	Weight for regularization term	1.0	1.0	1.0	1.0
λ_{gene}	Weight for encoding original information	0.4	0.6	0.1	0.9
d	The reduced dimension	3	3	3	3
n_{mini}	Mini-batch size	64	64	64	64
T_1	The epoch of inner loop	1	1	1	1
T_2	The epoch of outer loop	50	50	50	50
s	The step size for the gradient flow update	0.1	0.1	0.1	0.1
lr_ζ	The learning rate of R_ζ	10^{-3}	10^{-3}	10^{-3}	10^{-3}
lr_ψ	The learning rate of D_ψ	10^{-5}	10^{-5}	10^{-5}	10^{-5}

References

[1] Huang, J., Jiao, Y., Liao, X., Liu, J. & Yu, Z. Deep dimension reduction for supervised representation learning. *IEEE Transactions on Information Theory* (2024).

[2] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323**, 533–536 (1986).

[3] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521**, 436–444 (2015).

[4] Platen, E. An introduction to numerical methods for stochastic differential equations. *Acta numerica* **8**, 197–246 (1999).

[5] Halidias, N. & Kloeden, P. E. A note on the euler–maruyama scheme for stochastic differential equations with a discontinuous monotone drift coefficient. *BIT Numerical Mathematics* **48**, 51–59 (2008).

[6] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[7] Li, K.-C. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association* **86**, 316–327 (1991).

[8] Cook, D. R. & Weisberg, S. Sliced inverse regression for dimension reduction: comment. *Journal of the American Statistical Association* **86**, 328–332 (1991).

[9] Shao, Y., Cook, D. R. & Weisberg, S. Marginal tests with sliced average variance estimation. *Biometrika* **94**, 285–296 (2007).

[10] Suzuki, T. & Sugiyama, M. Sufficient dimension reduction via squared-loss mutual information estimation. *Neural computation* **25**, 725–758 (2013).

[11] Fukumizu, K., Bach, F. R. & Jordan, M. I. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research* **5**, 73–99 (2004).

[12] Székely, G. J., Rizzo, M. L. & Bakirov, N. K. Measuring and testing dependence by correlation of distances. *The Annals of Statistics* **35**, 2769–2794 (2007).

[13] Wang, R., Karimi, A.-H. & Ghodsi, A. Distance correlation autoencoder. In *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (IEEE, 2018).

[14] Zhu, J. *et al.* Invariant and sufficient supervised representation learning. In *2023 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (IEEE, 2023).

[15] Gao, Y. *et al.* Deep generative learning via variational gradient flow. In *International Conference on Machine Learning*, 2093–2101 (2019).

[16] Gao, Y. *et al.* Deep generative learning via euler particle transport. In *Mathematical and Scientific Machine Learning*, 336–368 (PMLR, 2022).