

Faculté des sciences

Controlled Latent ODE-RNN for Continuous-Time Blood Glucose Modeling

Representation Learning, Factual and
Counterfactual Dietary Predictions

Author: **Louis-David PIRON**
Supervisor: **Prof. Laura SYMUL**
Reader: **Prof. Donatien HAINAUT**
Academic year 2025–2026
Master [120] en science des données, orientation statistique

“Nature uses only the longest threads to weave her patterns, so each small piece of her fabric reveals the organization of the entire tapestry.”

RICHARD P. FEYNMAN

Abstract

Modeling and predicting individual continuous-time blood glucose trajectories is a challenging problem at the intersection of dynamical systems modeling and precision medicine. The underlying process is continuous, governed by highly non-linear physiological dynamics, perturbed by discrete meal intake, and observed only through noisy, irregularly sampled measurements. While explicit mechanistic models have been proposed, some may make assumptions about the dynamics that are difficult to experimentally validate. This motivates turning to data-driven approaches relying on overparametrized models. Specifically, Neural differential equation approaches handle irregularly sampled data while enjoying the toolbox of dynamical systems as well as the expressivity of neural networks. In particular, the Latent ODE-RNN of [Rubanova et al. \(2019\)](#), a Deep State-Space model, provides a principled uncertainty quantification framework, yet lacks an explicit mechanism for exogenous meal inputs, which is essential for blood glucose modeling.

In this work, we introduce CLODE (Controlled Latent ODE-RNN), a generative framework that extends the Latent ODE-RNN by incorporating meal intake as a time-varying exogenous control signal into the latent dynamics. Two meal signal parametrizations are compared: a Dirac impulse and a β -kernel distributing carbohydrate absorption over a physiologically plausible ingestion window.

CLODE was evaluated on two simulated populations of 500 *in silico* individuals, one physiologically homogeneous and one exhibiting inter-individual variability, and compared against the ODE-LSTM ([Lechner and Hasani, 2020](#); [Fitzgerald et al., 2023](#)) and the Latent ODE-RNN models. CLODE (β -kernel) substantially outperformed both baselines on factual forecasting and counterfactual prediction across three dietary scenarios. Latent space analysis further suggested that explicitly separating exogenous meal inputs from latent dynamics encourages the emergence of physiologically structured representations, though alignment with ground-truth parameters remained low. We conjecture that conditioning on longer observation windows, conditioning on additional inputs that would be available in realistic clinical conditions, or refining the architecture would allow to better disambiguate individual physiological profiles. Overall, these results provide encouraging evidence that controlled latent dynamics constitute a promising direction for personalized glucose modeling, dietary planning, or any "homeostasis-and-perturbation" system where individual-specific latent dynamics are perturbed by exogenous inputs and observed through noisy, irregularly sampled measurements.

Keywords: Glucose Modeling, Deep state-space Modeling, Neural Differential Equations, Representation Learning, Counterfactual simulation, Irregularly sampled medical time series.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Laura Symul, for her guidance, availability, and thoughtful feedback throughout this project. I genuinely enjoyed our discussions and learned a great deal from them.

I am also grateful to Guillaume Deside, PhD candidate at UCLouvain, for providing the physiological glucose simulator that enabled the empirical validation of this work, and for his helpful feedback and participation in our meetings.

My thanks also go to Prof. Donatien Hainaut for agreeing to read and evaluate this thesis.

Computational resources were provided by the *Consortium des Équipements de Calcul Intensif* (CECI), whose support is gratefully acknowledged.

Finally, my deepest gratitude goes to my parents for their unwavering support throughout my studies and beyond.

Note

Contributions are reported using the "Contributor Role Taxonomy" (CRediT) taxonomy in Appendix C.1.

Contents

Abstract	3
Acknowledgements	4
1 Introduction	8
1.1 Research Motivation and Context	8
1.1.1 Medical Stakes, Challenges and Opportunities	9
1.1.2 The Modeling Landscape for Postprandial Glucose.	10
1.1.3 Gaps and Contribution	11
1.2 Research Question and Objectives	12
1.3 Organization	13
2 Background and Related Work	14
2.1 Classical Dynamical System Modeling	14
2.1.1 Continuous-Time Dynamical Systems	14
2.1.1.1 Existence and uniqueness	15
2.1.1.2 Numerical Integration	16
2.1.2 From Mechanistic to Data-Driven Dynamical Systems	19
2.1.2.1 State-Space Modeling	19
2.1.2.2 Parameter estimation and identifiability	20
2.1.2.3 Stochastic Extension	23
2.1.2.4 Application to Glucose	24
2.1.2.5 A change of perspective	25
2.2 Modeling Unknown Dynamics	26
2.2.1 SINDy: A Sparsity-Promoting Technique	26
2.2.1.1 Application to Glucose	28
2.2.2 Neural ODEs: An Overparametrized Technique	29
2.2.2.1 Rationale behind NODEs	29
2.2.2.2 Overparametrization	30
2.2.2.3 Incorporating Observations	31
2.2.2.4 Out-of-sample performance	32
2.2.2.5 Universal Approximation and Augmentation	36

2.2.2.6	Incorporating Exogenous Control	38
2.2.2.7	Application to glucose	39
2.3	Neural Differential Equations and Glycemia	40
2.3.1	Interleaving approach	40
2.3.1.1	Recurrent Neural Networks	41
2.3.1.2	Gated Architectures	42
2.3.1.3	ODE-RNNs	43
2.3.1.4	Handling Exogenous Inputs	45
2.3.1.5	Application to Glucose	46
2.3.2	Interpolation Approach	47
2.3.2.1	Controlled Differential Equations	47
2.3.2.2	Neural Controlled Differential Equations	47
2.3.2.3	NCDEs as the Continuous-Time Limit of RNNs	48
2.3.2.4	Application to glucose	48
2.3.3	Deep State-Space Modeling Approach	49
2.3.3.1	Variational Autoencoders	49
2.3.3.2	The Representation Learning Challenge	52
2.3.3.3	From VAE to Latent ODE	53
2.3.3.4	Latent ODE-RNN: Encoding Irregular Time Series	56
2.3.3.5	Application to glucose	57
2.3.4	Towards a Controlled Latent ODE-RNN for Glucose Modeling	58
3	Methodology	59
3.1	Proposed Model	59
3.1.1	Problem Formulation	59
3.1.2	Controlled Latent ODE-RNN	60
3.1.2.1	Model	61
3.1.2.2	Training	64
3.1.2.3	Properties	65
3.2	Evaluation Protocol	66
3.2.1	Benchmark Models	66
3.2.1.1	ODE-LSTM	66
3.2.1.2	Latent ODE-RNN	67
3.2.1.3	Hyperparameters Tuning and Model Selection	67
3.2.2	Simulation Study	68
3.2.2.1	Data Generating Process	68
3.2.3	Evaluation Metrics	72
3.2.3.1	Forecasting accuracy	72
3.2.3.2	Latent space structure	73
3.2.3.3	Counterfactual validity	73
4	Results	74
4.1	Forecasting Performance	74
4.2	Counterfactual Analysis	77
4.3	Latent Space Validation	80
5	Conclusion	82

A	Additional Mathematical Material	91
A.1	Universal Approximation properties of NODEs	91
A.1.1	Flow Map	91
A.1.2	The flow maps of an NODE is an homeomorphism	91
A.2	Inductive Bias and UDEs	92
A.3	GRU-ODE-Bayes	92
B	Additional Experimental Material	94
B.1	Architectural Details	94
B.1.1	CLODE	94
B.1.2	Latent ODE-RNN	94
B.1.3	ODE-LSTM	95
B.2	Extrapolation and Representation Learning	96
B.2.1	ODE-LSTM	96
B.2.2	Latent ODE-RNN	97
B.2.3	CLODE β Kernel	98
B.2.4	CLODE Dirac Impulse	99
B.3	Counterfactual Performance	100
B.3.1	ODE-LSTM	100
B.3.2	CLODE β Kernel	100
B.3.3	CLODE Dirac Impulse	101
B.4	Latent Space Visualization	103
B.4.1	CLODE β Kernel	103
B.4.2	CLODE Dirac Impulse	104
B.4.3	Latent ODE-RNN	105
B.5	Model Predictions Across Training Epochs	106
B.5.1	CLODE β Kernel	106
B.5.2	CLODE Dirac Impulse	107
B.5.3	Latent ODE-RNN	108
B.5.4	ODE-LSTM	109
C	Miscellaneous	112
C.1	Contributor Role Taxonomy	112
C.2	Example 2.2.2 : Additional Material	113
C.3	Example 2.3.1 : Additional Material	114

1.1 Research Motivation and Context

Most modern dynamical systems, and biological dynamical systems in particular, share three inconvenient properties: they cannot be fully observed, their governing equations are unknown, and they are subject to external perturbations.

Complex systems, such as the human body, are high-dimensional, yet one typically only has access to a small, sparse and incomplete set of noisy snapshots of the underlying process. Furthermore, their governing equations are equally puzzling. While linear dynamical systems are theoretically well-characterized, no overarching framework exists for the non-linear case ([Brunton and Kutz, 2022](#)), and biological dynamics are certainly highly non-linear. Finally, most systems are open: external inputs continuously drive them away from equilibrium, and any model that conflates this forcing explicitly risks mischaracterizing the endogenous dynamics.

This combination of challenges occurs across domains as varied as pharmacokinetics, where drug concentration rises following a dose and decays as the compound is metabolized; or thermoregulation, where an external thermal stress drives core body temperature away from its set point, before being restored by sweating and vasodilation. A framework for learning such dynamics from data, that handles : (i) partial observability, (ii) unknown non-linear dynamics, and (iii) exogenous inputs, would transfer broadly across many such settings.

This work takes a step toward such a framework in the context of *glucose dynamics*, a particularly compelling instance of this pattern. For a healthy individual, blood glucose concentration is regulated through *homeostasis*, a continuous negative feedback mechanism maintaining glucose within a narrow physiological range (typically 4-10 mmol/L), despite external perturbations. This is illustrated in Figure 1.1. When blood glucose rises, insulin, secreted by pancreatic β -cells, promotes glucose uptake by peripheral tissues and inhibits hepatic glucose production, making the glucose level drop toward a baseline. If blood glucose drops too low, pancreatic α -cells secrete glucagon, raising blood glucose back to normal.

In the case of glucose dynamics, the most prominent perturbation is food intake. When food is ingested, carbohydrates are digested and absorbed from the intestine into the

bloodstream, causing blood glucose level to rise and triggering the regulatory feedback mechanisms described earlier. This change in blood glucose following a meal is referred to as the *postprandial glycemic response* (PPGR).

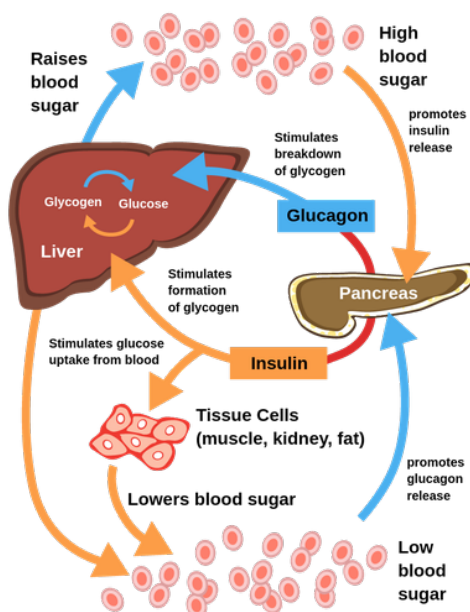


FIGURE 1.1: Glucose homeostasis regulation loop, from [labster.com](https://www.labster.com).

1.1.1 Medical Stakes, Challenges and Opportunities

Accurately predicting postprandial excursions is of great interest, even for non-diabetic subjects, as sustained hyperglycemia is a known risk factor for cardiovascular diseases, obesity and type 2 diabetes (O’Keefe and Bell, 2007; Song et al., 2023).

Beyond clinical risk prevention, accurate PPGR prediction is also a building block for personalized nutrition and precision medicine, where dietary recommendations are tailored to the individual (Zeevi et al., 2015).

In recent years, the increasing availability of electronic health records (EHRs) and wearable monitoring devices has spurred considerable interest in fully data-driven approaches (De Brouwer, 2023). In particular, continuous glucose monitors (CGMs) now enable access to high-resolution, longitudinal glucose measurements at the individual level (Bender et al., 2025). Beyond glucose, exogenous quantities can be monitored concurrently: meal intake, logged via a smartphone application, or insulin, delivered by a connected pump. This opens the possibility of learning individual-specific glucose dynamics directly from observational data.

Recent Datasets made publicly available for personalized nutrition purposes, such as Gutierrez-Osuna et al. (2025), collect data in so-called *free-living* conditions, where individuals carry out their daily activities without experimental constraints (Cisuelo et al., 2023).

In such settings, participants are not subject to standardized protocols regarding timing, diet, or activity, and instead follow their usual routines. This setup introduces additional challenges, such as increased measurement noise, irregular sampling, and uncertainty in

reported events (*e.g.*, meal timing and composition). Moreover, although the sampling conditions are more realistic, the data representation itself might be simplified or biased. This is particularly true for dietary logs, which typically record meals as instantaneous events with a timestamp and estimated nutrient content, without capturing the temporal dynamics of ingestion.

The prediction task is further complicated by *inter-individual variability* (IIV): glucose profiles differ widely across individuals due to both meal-related factors (timing, quantity, and composition of meals) and physiological factors (*e.g.* insulin sensitivity and gut microbiota). The physiological component of this variability is especially challenging: even for an identical meal, two individuals can exhibit very different PPGRs (Zeevi et al., 2015), as illustrated in Figure 1.2.

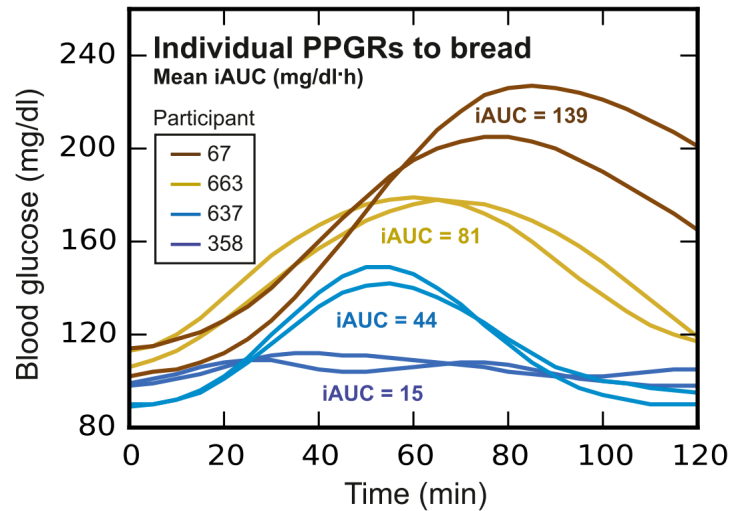


FIGURE 1.2: Postprandial glycaemic responses to bread across four participants (two replicates per participant, consumed on two different mornings), from Zeevi et al. (2015).

1.1.2 The Modeling Landscape for Postprandial Glucose.

For decades, the prevailing approach to modeling glucose dynamics has been mechanistic. From the minimal model (Bergman et al., 1979) to the UVA/Padova simulator (Man et al., 2014), one has long relied on mathematical representation of a system based on fundamental physical, chemical, or biological principles. These models offer physiological interpretability and support counterfactual simulation, but are sometimes difficult to estimate from sporadically observed data.

Recognizing the difficulty of fully characterizing glucose dynamics from first principles and sporadically observed data, several authors, such as Sirlanci et al. (2023), have proposed stochastic differential equation models as "grey-box" probabilistic alternatives, yielding models with a reduced physiological fidelity, but better estimability.

The increasing availability of CGM data also spurred interest in fully data-driven approaches. Recurrent neural networks (RNNs) and their gated variants have been successfully applied to glucose forecasting (Afsaneh et al., 2022). However, standard recurrent architectures assume regularly sampled discrete-time sequences, inconsistent with the irregular nature of CGM data and the continuous-time nature of glucose dynamics. While

temporal binning and mean imputation are commonly used as workarounds, irregular time gaps may themselves carry information about the individual’s latent physiological state and should not be discarded (Che et al., 2018).

This has motivated a growing literature on neural differential equations (NDE), which parametrize the dynamics of a system with neural networks, while naturally handling irregular sampling. De Brouwer (2023) organizes the NDE paradigm into three families: the *interleaving* approach, which alternates between ODE-governed evolution and discrete updates at observation times (Rubanova et al., 2019; Lechner and Hasani, 2020); the *interpolation* approach, which drives the latent state via a continuous control path constructed from the data (Kidger et al., 2020; Morrill et al., 2021a,b); and the *deep state-space*¹ approach, encoding an observed trajectory into a distribution over a latent initial condition and evolving it forward in time, before emitting observations from it (Chen et al., 2018; Rubanova et al., 2019).

The first two families have been applied to glucose forecasting (Fitzgerald et al., 2023; Malpica-Morales et al., 2025). The third has not, and is the one we adopt in this work. The deep state-space paradigm is particularly appealing. First, it frames the model as a deep state-space model: a latent continuous-time process coupled to an observation model (Lin and Michailidis, 2024), a two-level structure well-suited to noisy CGM measurements. Second, because the model is generative, it supports uncertainty quantification (UQ) by construction: sampling from the posterior over the initial latent state yields a distribution over trajectories rather than a point forecast. Third, latent ODE models enjoy universal approximation guarantees for continuous-time dynamics (Kidger et al., 2020). Finally, and crucially, this is the only paradigm that produces an explicit latent representation of each individual’s trajectory, one that can in principle be structured to reflect inter-individual variability (Rubanova et al., 2019).

Modeling glucose as a state-space model is not new. Wang et al. (2014), developed a personalized state-space model with an Extended Kalman Filter (EKF), explicitly incorporating meal intake and insulin as exogenous inputs. Their results confirmed the importance of meal information for early detection of postprandial excursions. However, EKFs typically struggle with state estimation in the presence of highly non-linear dynamics (Lin and Michailidis, 2024).

A further source of inspiration comes from the structurally analogous problem of pharmacokinetics (PK), which attempts to describe, predict, and forecast how drug concentration evolves in the body following repeated drugs intake. Like PPGR, PK involves an exogenous input (drug dose, or meal) perturbing a continuous-time physiological process with individual-specific latent dynamics. Lu et al. (2021) recently applied a Latent ODE architecture to PK modeling, demonstrating substantially better generalization to unseen dosing regimens than LSTM or gradient-boosted alternatives (a cross-regimen robustness that directly mirrors the cross-dietary-regime generalisation we seek in the glucose setting). The encouraging performance of the Latent ODE in this structurally similar problem motivates its application to PPGR. However, the architecture of Lu et al. (2021) relies on a relatively rigid formulation, unaligned with the best practices of the dynamical system and NDE literatures.

¹Termed *autoencoder* approach by (De Brouwer, 2023). Following Lin and Michailidis (2024), we prefer the deep state-space terminology as it allows us to underline the link of this family with traditional state-space approaches.

1.1.3 Gaps and Contribution

Despite this promise, existing Latent ODE formulations (Chen et al., 2018; Rubanova et al., 2019) include no mechanism to incorporate exogenous inputs, a gap acknowledged by Chapfuwa et al. (2022). For glucose, this is critical: trajectories are determined not only by an individual’s physiology but also by discrete meal events inducing postprandial excursions of varying amplitude and duration. This refinement further allows generating glucose trajectories conditional on a prospective meal scenario and therefore can support counterfactual reasoning.

This master thesis addresses the latter gap by proposing a *Controlled Latent ODE-RNN* (CLODE) framework for individual-level blood glucose dynamics modeling, in the presence of carbohydrates intake. The architecture extends the Latent ODE-RNN of Rubanova et al. (2019) by explicitly incorporating meal intake as a time-varying exogenous input into the latent continuous-time dynamics, yielding a generative model capable of producing individualized glucose trajectories conditioned on a prospective meal scenario. Beyond glucose, such a framework could be applicable to any "homeostasis-and-perturbation" system where individual-specific latent dynamics are driven by exogenous inputs and observed through noisy, irregularly sampled measurements.

1.2 Research Question and Objectives

More formally, we seek to answer the following research question:

Can a controlled Latent ODE-RNN learn a physiologically meaningful latent representation of individual blood glucose dynamics that enables reliable generation of factual and counterfactual trajectories ?

This question is operationalized through three empirical objectives:

1. **Controlled dynamics modeling.** Assess whether the proposed architecture accurately predicts individual glucose trajectories conditioned on meal intake, acting as exogenous control inputs.
2. **Latent representation validation.** Evaluate whether the learned latent space recovers physiologically interpretable structure. Specifically, whether latent initial conditions correlate with known simulation parameters and whether individuals with similar metabolic profiles cluster in latent space.
3. **Counterfactual analysis.** Gauge if the model supports individual-level counterfactual predictions under alternative meal scenarios.

1.3 Organization

The remainder of this master thesis is organized as follows.

Chapter 2 (Background and Related Work) develops the theoretical foundations underlying the proposed approach. It introduces dynamical systems and their modeling and surveys the neural differential equations literature.

Chapter 3 (Methodology) details the proposed Controlled Latent ODE-RNN framework as well as the evaluation protocol. It formalizes the problem statement, describes the full architecture, and presents evaluation metrics, the benchmarks as well as the physiological simulator we use in the experiments.

Chapter 4 (Results) presents and discusses the experimental results of the experimental study, evaluating CLODE along three axes: factual forecasting predictions, counterfactual predictions, and latent space structure.

Chapter 5 (Conclusion) summarizes the contributions, discusses limitations, and identifies directions for future work.

Background and Related Work

2.1 Classical Dynamical System Modeling

From the physiological description of glucose presented in Section 1.1, it naturally follows that glucose is a continuously evolving process, subject to endogenous physiological forces and exogenous perturbations, food intake being the most prominent.

This structure is precisely that of dynamical systems, and the question of how to model them is therefore inseparable from our research question. Therefore, in this section, we progressively present classical dynamical system modeling approaches.

Occasionally, we use the well-known Lotka-Volterra predator-prey system as a concrete example, in order to illustrate some theoretical concepts.

2.1.1 Continuous-Time Dynamical Systems

A dynamical system is a system whose state $\mathbf{z}$ evolves over time according to a set of rules f . This broad definition suits many physical, biological, and physiological phenomena.

This master thesis primarily focuses on continuous-time dynamical systems, given the continuous-time nature of glucose. More specifically, for now, we will mostly assume that the dynamics are governed by an Ordinary Differential Equation (ODE) :

$$\frac{d\mathbf{z}}{dt}(t) = f(\mathbf{z}(t); \boldsymbol{\beta}) \quad \text{with} \quad t \in [0; T] \subset \mathbb{R} \quad (2.1)$$

Equation 2.1 is characterized by three elements :

1. A *state vector* $\mathbf{z}(t)$, which represents the system's state at time t and takes values in a s -dimensional *state space* ;
2. A *vector field* f , encoding the rules governing the system's evolution ;
3. A *parameter vector* $\boldsymbol{\beta}$, specifying how f is calibrated.

In Equation 2.1, we assume that we are dealing with an *autonomous* system, meaning that $f : \mathbb{R}^s \rightarrow \mathbb{R}^s$ depends only on the current state $\mathbf{z}(t)$, not on time itself (*i.e.* the rules do not change over time). One might as easily consider *non-autonomous* systems, that have a time-dependent vector field $f : [0, T] \times \mathbb{R}^s \rightarrow \mathbb{R}^s$, or that include exogenous inputs $\mathbf{u}(t)$ (*c.f. infra* 2.1.2.1), or both.

Similarly, for the parameter vector, we assume time-independence. Time-varying parameters might be considered, which may be more appropriate than time-invariant ones, for instance, when modeling an ill patient, whose metabolism is expected to evolve over time.

Note that the state vector $\mathbf{z}(t)$ is commonly referred to as the *trajectories* of the system. For continuous-time systems, these trajectories are continuous curves evolving in the state space, obtained by solving the differential equation defined in 2.1.

2.1.1.1 Existence and uniqueness

In the previous section, we referred to $\mathbf{z}(t)$ as the *trajectories* of the system. The reason is that there are infinitely many solutions to the differential equation in 2.1, each corresponding to a different initial condition $\mathbf{z}(t_0) := \mathbf{z}_0$. To uniquely determine a solution (or trajectory), we have to specify an initial value problem (IVP):

$$\mathbf{z}(t_0) = \mathbf{z}_0, \quad \frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t); \boldsymbol{\beta}). \quad (2.2)$$

Provided that f is *Lipschitz continuous* (2.1.1), Picard's Existence Theorem (2.1.1), guarantees that there exists a unique solution $\mathbf{z}(t)$ to the IVP in 2.2. This solution can be expressed in integral form as:

$$\mathbf{z}(t) = \mathbf{z}_0 + \int_{t_0}^t f(\mathbf{z}(\tau); \boldsymbol{\beta}) d\tau \quad (2.3)$$

In practice, one rarely uses equation 2.3, as most differential equations cannot be solved analytically. This is the case for glucose, whose dynamics are complex and highly non-linear. One therefore typically resorts to numerical methods to approximate the solution at discrete time points. This is done using *numerical integrators*, which will be introduced in Section 2.1.1.2.

Theorem 2.1.1 (Picard's Existence Theorem). *Let $f : [0, T] \times \mathbb{R}^s \rightarrow \mathbb{R}^s$ be continuous in t and uniformly Lipschitz in $\mathbf{z}$. Let $\mathbf{z}_0 \in \mathbb{R}^s$. Then there exists a unique differentiable function $\mathbf{z} : [0, T] \rightarrow \mathbb{R}^s$ satisfying*

$$\mathbf{z}(t_0) = \mathbf{z}_0, \quad \frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t); \boldsymbol{\beta}).$$

Definition 2.1.1. A function $f : \mathbb{R}^s \rightarrow \mathbb{R}^s$ is *locally Lipschitz continuous* on $\mathbb{R}^s$, or simply *Lipschitz continuous*, if for every $R > 0$ there exists a constant $M > 0$ such that

$$\|f(x) - f(y)\| \leq M\|x - y\|, \quad \text{for all } x, y \in \mathbb{R}^s \text{ with } \|x\|, \|y\| \leq R.$$

To summarize, three ingredients are therefore necessary (and sufficient) to fully characterize the trajectory of a dynamical system: a vector field f , its parameters β , and an initial condition $\mathbf{z}_0$. In many settings, these are not trivially available. The functional form of f could be only partially known, β may present identification issues, vary across individuals or environments, and $\mathbf{z}_0$ may not be directly observable. The next section will review the foundations of numerical integration, and the following section will explore how the challenge of identifying f , β , and $\mathbf{z}_0$ can be addressed.

2.1.1.2 Numerical Integration

Even assuming known f , β and $\mathbf{z}_0$, closed-form solutions to Equation (2.3) exist only for a restricted class of vector fields f . In general, $\mathbf{z}(t)$ must be approximated numerically, and a broad family of methods has been developed for this purpose. These *numerical integrators* will be central to the data-driven approaches reviewed in Section 2.2.

To illustrate numerical integration, we use the *Lotka-Volterra equations*, a classical predator-prey model that admits no closed form solution of $\mathbf{z}(t)$ (Lotka, 1926). The state vector $\mathbf{z}(t) = [x(t), y(t)]^\top$ consists of the prey population $x(t)$ and the predator population $y(t)$ of an ecosystem at time t . The Lotka-Volterra equations are:

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \kappa_1 x(t) - \kappa_2 x(t)y(t) \\ \kappa_3 x(t)y(t) - \kappa_4 y(t) \end{bmatrix} \quad (2.4)$$

where $\kappa_1, \kappa_2, \kappa_3$, and κ_4 are positive real parameters governing the population dynamics. Here, $\beta := (\kappa_1, \kappa_2, \kappa_3, \kappa_4)^\top$ is the parameter vector.

Given the absence of analytical solution for Equation 2.4, we resort to *numerical integrators*. They take their name from the simple fact that they try to numerically approximate the integral in Equation 2.3. In the specific case of ODEs, we use the terminology of *ODE Solver*.

ODE solvers can be seen as functions that take as input the vector field f , the initial state $\mathbf{z}_0$, and a set of time points $\{t_k\}_{k=0}^{N-1}$, and return the approximated states $\{\mathbf{z}(t_k)\}_{k=0}^{N-1}$ at those time points. The simplest ODE solver is directly derived from the definition of the derivative:

$$\frac{d\mathbf{z}}{dt}(t) := \lim_{\Delta t \rightarrow 0} \frac{\mathbf{z}(t + \Delta t) - \mathbf{z}(t)}{\Delta t} \approx \frac{\mathbf{z}(t + \Delta t) - \mathbf{z}(t)}{\Delta t} \quad (2.5)$$

As our objective is to find a solution to 2.1:

$$\frac{d\mathbf{z}}{dt}(t) = f(\mathbf{z}(t); \beta) \approx \frac{\mathbf{z}(t + \Delta t) - \mathbf{z}(t)}{\Delta t}$$

Hence,

$$\mathbf{z}(t + \Delta t) \approx \mathbf{z}(t) + \Delta t f(\mathbf{z}(t); \beta) \quad (2.6)$$

Equation 2.6 defines the *Explicit Euler method*¹. Starting from $\mathbf{z}_0$, it allows us to approximate $\mathbf{z}(t)$ using a sufficiently small Δt . The Euler method is the simplest member of the broader *Runge-Kutta* (RK) family, which can achieve higher accuracy by evaluating f at multiple intermediate points within each step.

The celebrated RK4 method, for instance, uses four such evaluations and achieves fourth-order accuracy $\mathcal{O}(\Delta t^4)$:

$$\mathbf{z}(t + \Delta t) \approx \mathbf{z}(t) + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4) \quad (2.7)$$

where the four slope evaluations are:

$$\begin{aligned} k_1 &= f(\mathbf{z}(t); \boldsymbol{\beta}) & k_2 &= f\left(\mathbf{z}(t) + \frac{\Delta t}{2} k_1; \boldsymbol{\beta}\right) \\ k_3 &= f\left(\mathbf{z}(t) + \frac{\Delta t}{2} k_2; \boldsymbol{\beta}\right) & k_4 &= f(\mathbf{z}(t) + \Delta t k_3; \boldsymbol{\beta}) \end{aligned}$$

The advantages of numerical ODE solvers are obvious: we can approximate a continuous-time process, even when no closed-form solution exists. Thanks to that approximation, we can make predictions at arbitrary time points, even those not included in the observed data, allowing for both interpolation and extrapolation. Moreover, since a trajectory is fully determined by $f(\cdot; \boldsymbol{\beta})$ and $\mathbf{z}_0$, modifying the initial condition yields a different trajectory making ODE solvers a natural engine for *counterfactual simulation*.

Example 2.1.1 (Fox and Rabbit Population). *Consider a synthetic predator-prey ecosystem of foxes and rabbits. Over time, the count of each population was measured $N = 30$ times, yielding the following observed data:*

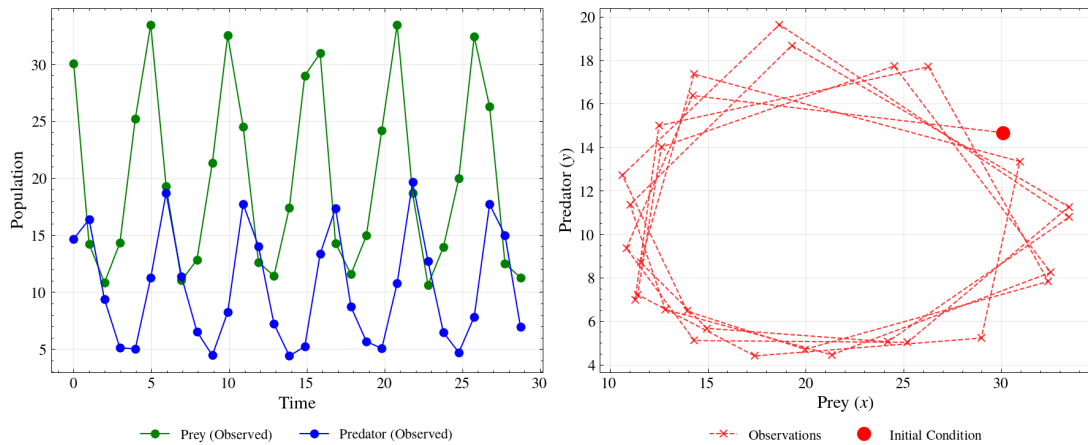


FIGURE 2.1: Observed data from the fox-rabbit system. **Left:** Time series of rabbit and fox populations. **Right:** Phase space trajectory.

Figure 2.1 showcases observations in two complementary ways. First, as time series showing how each population evolves over time. Second, in phase space, where the predator population is plotted against the prey population, revealing the underlying dynamical structure of the system.

¹A numerical method is said to be explicit when it only uses earlier time steps for the approximation, as opposed to *implicit methods*, which also involve the current step (e.g., Backward Euler).

One conjecture that the dynamics of this ecosystem are governed by the Lotka-Volterra equations with parameters $\kappa_1 = 1.0, \kappa_2 = 0.1, \kappa_3 = 0.075, \kappa_4 = 1.5$. The vector field associated to those parameters is represented in Figure 2.2.

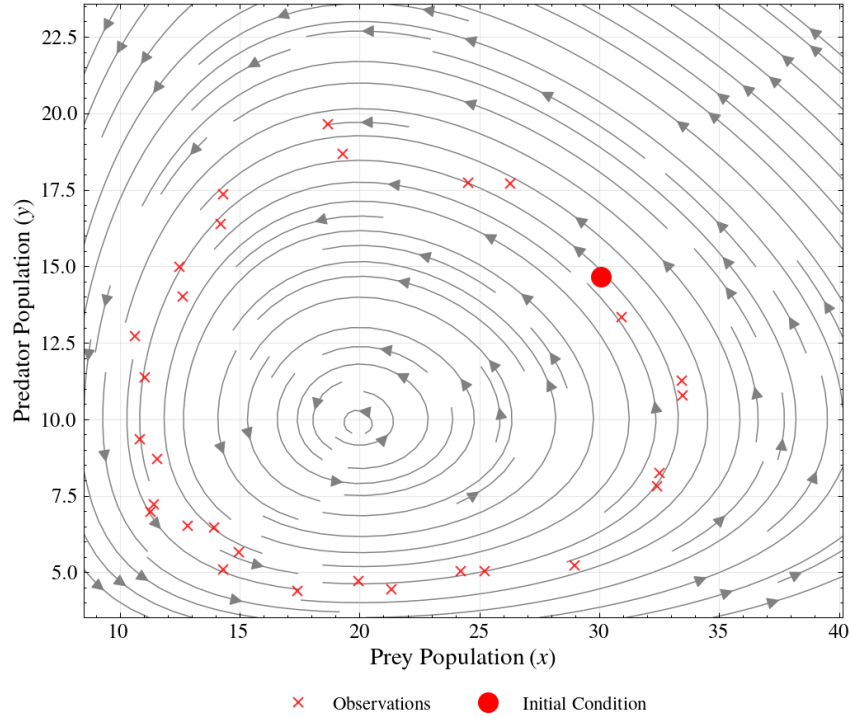


FIGURE 2.2: Vector field of the Lotka-Volterra equations with parameters $\kappa_1 = 1.0, \kappa_2 = 0.1, \kappa_3 = 0.075, \kappa_4 = 1.5$.

Given the initial state $\mathbf{z}_0 = [30, 15]^\top$ and our "belief" in f and the parameters β , we can approximate the populations dynamics over time using numerical ODE solvers. A numerical approximation is given in Figure 2.3.

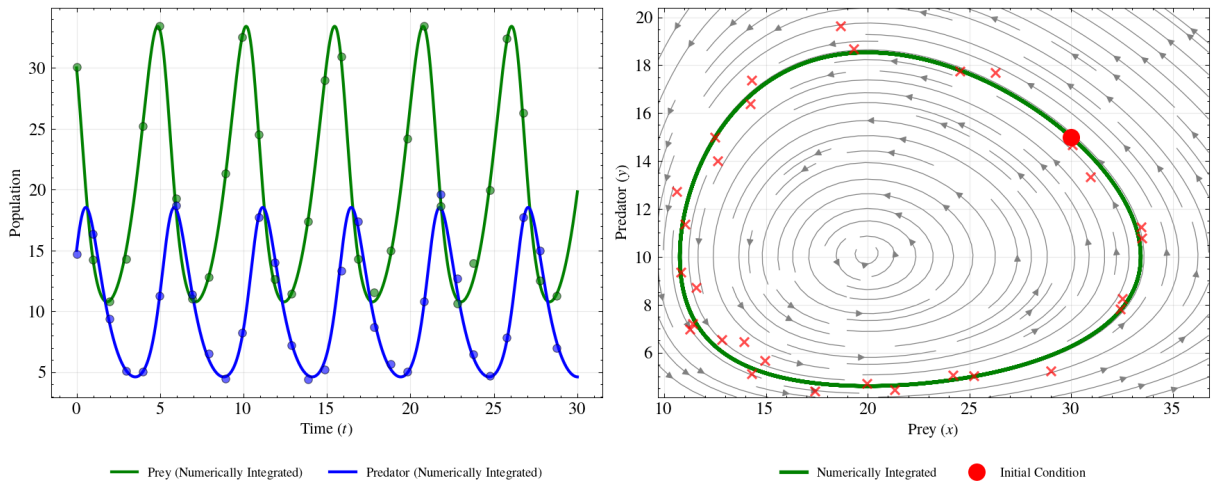


FIGURE 2.3: Numerical solution of the Lotka-Volterra equations using `odeint` default Dormand-Prince 5 (`dopri5`) method, with initial state $\mathbf{z}_0 = [30, 15]^\top$. **Left:** Time series of rabbit and fox populations. **Right:** Phase space trajectory.

2.1.2 From Mechanistic to Data-Driven Dynamical Systems

Up to now, we have assumed the full knowledge of f , β and $\mathbf{z}_0$. In practice, they are not trivially available. We therefore progressively relax those assumptions, starting from the *mechanistic* modeling paradigm, where f is prescribed *a priori* based on so-called first principles, *i.e.* fundamental laws of science.

Within this paradigm, we will answer three questions : (i) how to formally account for noisy, partial observations of the state space and exogenous perturbations (Section 2.1.2.1); (ii) how to estimate unknown parameters from data (Section 2.1.2.2); (iii) and how to account for system stochasticity (Section 2.1.2.3).

We then discuss how this framework has been applied to glucose modeling (Section 2.1.2.4) and why a fully data-driven approach can be beneficial.

2.1.2.1 State-Space Modeling

In practice, $\mathbf{z}(t)$ is not directly observable. One typically has access only to noisy, partial measurements of the true state. This motivates introducing an *observation model* that describes how the measurements $\mathbf{x}(t) \in \mathbb{R}^o$ are generated from the, now latent, $\mathbf{z}(t)$.

We also want to account for possible exogenous inputs $\mathbf{u}(t) \in \mathbb{R}^p$, not evolving as part of the system, but influencing its trajectory. Those exogenous inputs are often referred to as *control inputs* in engineering.²

This motivates the following two-level structure, called *state-space model* (SSM):

$$\frac{d\mathbf{z}}{dt}(t) = f(\mathbf{z}(t), \mathbf{u}(t); \beta) + \delta_t \quad (2.8)$$

$$\mathbf{x}(t) = g(\mathbf{z}(t), \mathbf{u}(t); \alpha) + \varepsilon_t \quad (2.9)$$

where δ_t is a *process noise* term, capturing stochasticity in the process, and ε_t is the *measurement noise*.

When $\delta_t = 0$, the state process is deterministic and all randomness resides in the observation emission. The stochastic extension, where $\delta_t \neq 0$, is briefly discussed in Section 2.1.2.3.

Unlike $\mathbf{z}(t)$, now considered a *latent* variable, $\mathbf{u}(t)$ and $\mathbf{x}(t)$ are viewed as *manifest* variables, because the latter is measured, *manifesting* itself to the observer (Dahleh et al., 2011).

Note

In classical state-space modeling, the latent state typically has lower dimension than the observations ($s \ll o$). In the deep learning paradigm, relying on Cover's theorem, we quite often do the opposite ($s \gg o$): the observations are lifted into a higher-dimensional latent space. We adopt this approach in Section 2.2. Justifications for it will be given in Section 2.2.2.5.

²For our glucose application, we postulate that $u(t) \in \mathbb{R}$, represents the meal perturbations. If, say, we wanted to extend to a model applicable to diabetic subject, we could take a bivariate control $\mathbf{u}(t) \in \mathbb{R}^2$, consisting of the meal intakes and insulin deliveries.

2.1.2.2 Parameter estimation and identifiability

Even when the functional form of f is known, its parameters β are rarely universal. They vary across ecosystem or patient, and must therefore be estimated from observed data. This is the *parameter estimation* problem.

One way to proceed is to frame that problem in a Bayesian setting, where we characterize the full posterior distribution over the parameters given N observed snapshots $\mathbf{X} = [\mathbf{x}_{t_0}, \dots, \mathbf{x}_{t_{N-1}}]^\top$.

The Bayesian approach to parameter estimation consists of three steps. First, one specifies a prior over the parameters and the initial condition, and an *observation model*, in the spirit of SSMs :

$$\beta \sim p(\beta) \quad (2.10)$$

$$\mathbf{z}_0 \sim p(\mathbf{z}_0 | \beta) \quad (2.11)$$

$$\mathbf{x}_{t_k} \sim p(\mathbf{x}_{t_k} | \mathbf{z}(t_k; \beta, \mathbf{z}_0)) \quad (2.12)$$

where $\mathbf{z}(t_k; \beta, \mathbf{z}_0)$ denotes the state reached at time t_k starting from $\mathbf{z}_0$.³

The choice of the observation model $p(\mathbf{x}_{t_k} | \mathbf{z}(t_k; \beta, \mathbf{z}_0))$ encodes assumptions about the noise structure, while the prior $p(\beta)$ allows to inject prior knowledge about the parameters if any. Otherwise uninformative (diffuse) priors can be used.

Second, the likelihood follows by factoring over observations and marginalizing over the initial condition:

$$p(\mathbf{X} | \beta) = \int p(\mathbf{z}_0 | \beta) \prod_{k=0}^{N-1} p(\mathbf{x}_{t_k} | \mathbf{z}(t_k; \beta, \mathbf{z}_0)) d\mathbf{z}_0 \quad (2.13)$$

Third, combining the likelihood with the prior via Bayes' theorem yields the posterior:

$$p(\beta | \mathbf{X}) \propto p(\mathbf{X} | \beta) p(\beta) \quad (2.14)$$

This posterior has no closed form and is typically approximated via Markov Chain Monte Carlo (MCMC) methods.

Beyond prior knowledge incorporation, one of the advantage motivating the Bayesian approach is that it allows for *uncertainty quantification* (UQ), which is of central interest when modeling dynamical systems. More specifically, the Bayesian posterior quantifies one specific source of uncertainty : our incomplete knowledge of β given finite, noisy snapshots $\mathbf{X}$.

A prerequisite for reliable estimation is *structural identifiability*, the question of whether distinct parameter values can produce identical trajectories. A model is structurally identifiable if β can in principle be uniquely recovered from perfect, noise-free observations. Even when this condition holds, *practical identifiability* may fail in finite, noisy datasets,

³Our formulation is inspired by Särkkä (2016), but not identical.

particularly when the number of parameters is large relative to the information content of the available measurements (Villaverde, 2019). This motivates the design philosophy of minimal models for the mechanistic framework, but also encourages moving beyond it using stochastic closure techniques for instance (Sirlanci et al., 2023).

Example 2.1.2 (Fox and Rabbit Population - Bayesian Parameter Estimation). *We return to the fox-rabbit ecosystem and ask whether the parameters β can be recovered from the $N = 30$ noisy observations introduced earlier. We use a particular instance of (2.10), where the parameters are reparametrized on the log scale with weakly informative Normal priors, ensuring positivity by construction. The initial condition $\mathbf{z}_0 = [x_0, y_0]^\top$ is treated as unknown and assigned a prior centered on the first noisy observation:*

$$\begin{aligned}\log \kappa_1 &\sim \mathcal{N}(0.0, 0.5^2) \\ \log \kappa_2 &\sim \mathcal{N}(-2.0, 0.5^2) \\ \log \kappa_3 &\sim \mathcal{N}(-2.0, 0.5^2) \\ \log \kappa_4 &\sim \mathcal{N}(0.0, 0.5^2) \\ \log x_0 &\sim \mathcal{N}(\log \mathbf{z}_{t_0}^{(1)}, 0.5^2) \\ \log y_0 &\sim \mathcal{N}(\log \mathbf{z}_{t_0}^{(2)}, 0.5^2)\end{aligned}$$

The observation model is assumed log-normal:

$$\log \mathbf{x}_{t_k} \sim \mathcal{N}(\log \hat{\mathbf{z}}(t_k; \beta, \mathbf{z}_0), \sigma^2)$$

where $\hat{\mathbf{z}}$ is the numerical approximation to $\mathbf{z}$, and σ is the observation noise (inferred jointly, with prior $\sigma \sim \text{LogNormal}(-1.0, 1.0)$). Posterior sampling was performed using *PyMC* (Abril-Pla et al., 2023) with the *DEMetropolisZ* algorithm (4 chains, 10 000 draws, 10 000 tuning steps)⁴.

All chains converged ($\hat{R} \leq 1.01$) and contain a large enough effective sample size for reliable inference ($ESS_{\text{bulk}} \geq 1489$, $ESS_{\text{tail}} \geq 2625$). Table 2.1 reports the posterior summaries. Figure 2.4 shows the marginal posterior distributions alongside the prior, and Figure 2.5 the corresponding posterior predictive trajectories.

Table 2.1: Posterior summaries for the Lotka-Volterra parameters. True values are coloured **green** if contained within the 95% highest probability density intervals (HPDI) and **red** otherwise.

Parameter	Ground Truth	Posterior Mean	HPDI 2.5%	HPDI 97.5%
κ_1	1.000	1.014	0.948	1.077
κ_2	0.100	0.103	0.095	0.110
κ_3	0.075	0.074	0.069	0.079
κ_4	1.500	1.478	1.388	1.577

⁴Post-burnin chains are available in Appendix C.

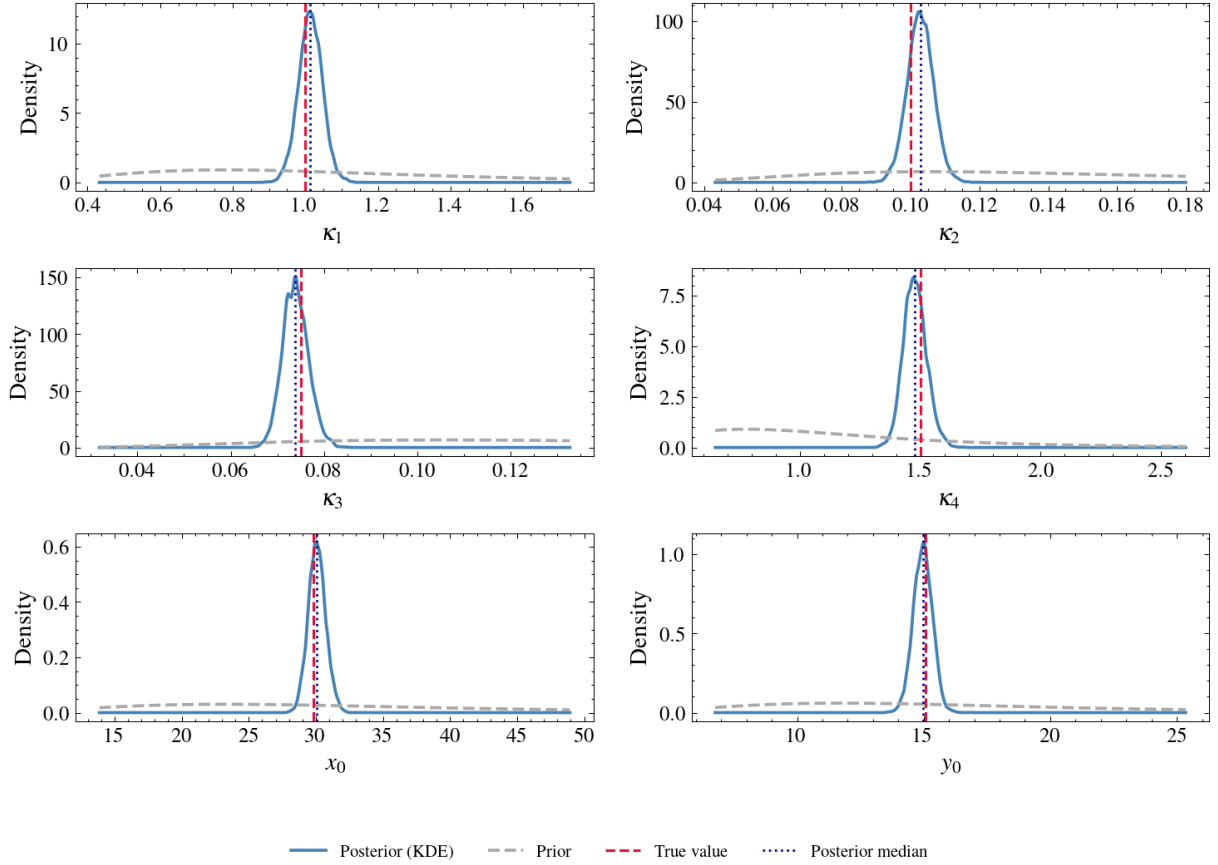


FIGURE 2.4: Marginal posterior distributions of the Lotka-Volterra parameters. The dashed red line indicates the true value; the dotted blue line indicates the posterior median.

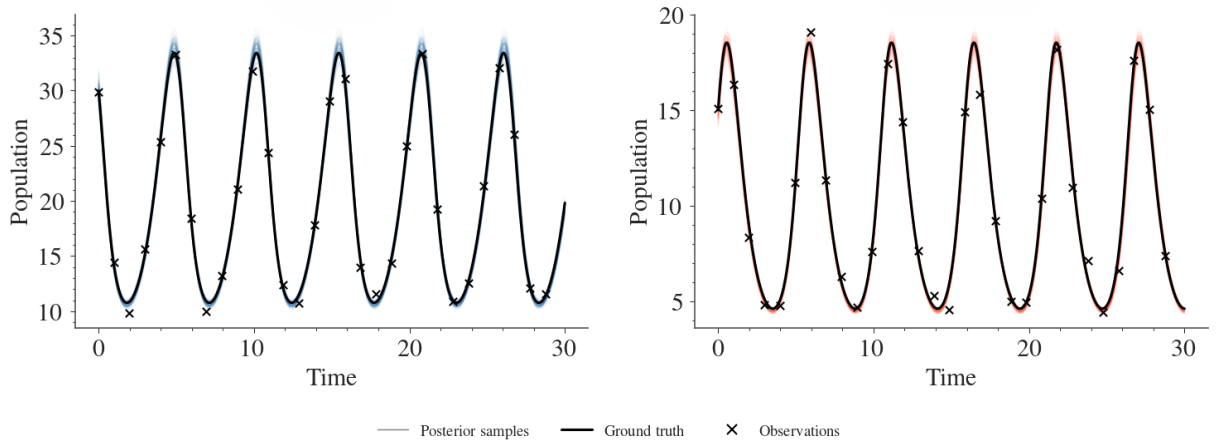


FIGURE 2.5: Posterior predictive check (500 samples). **Left:** prey population. **Right:** predator population.

The posteriors are well-concentrated around the true parameter values, all of which fall within the 95% HPDI (Table 2.1). The pair plot (Figure 2.6) provides an intuitive view of the identifiability. Some parameters exhibit strong pairwise correlations (particularly

κ_1 , the prey growth rate, and κ_4 , the predator mortality rate), while initial conditions are least correlated with the other model parameters.

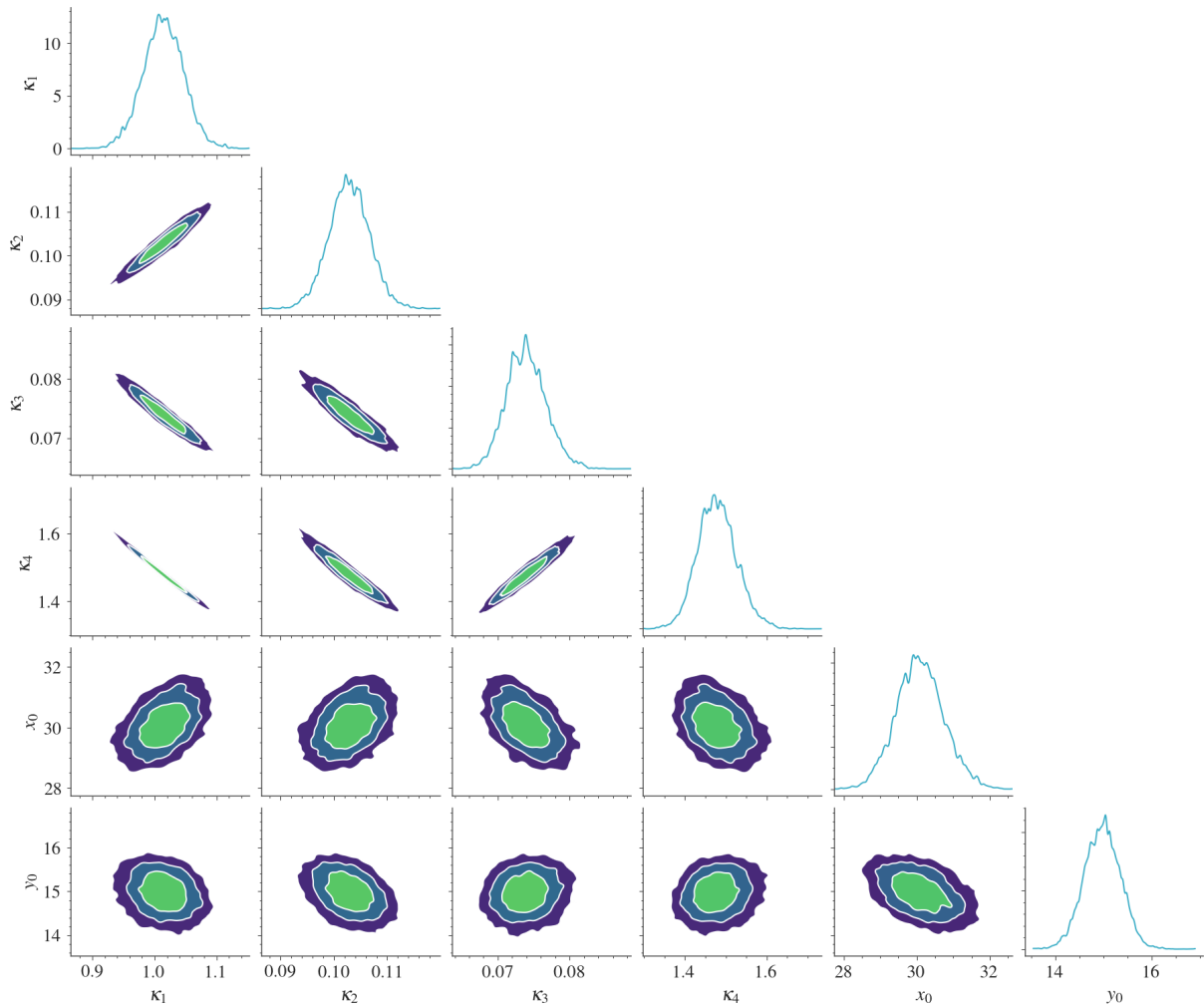


FIGURE 2.6: Pairwise posterior distributions of the Lotka-Volterra parameters.

The strong pairwise correlation observed between κ_1 and κ_4 illustrates that, even in a well-identified system, the data alone may not be sufficient to fully disentangle individual parameter contributions. This can become even more challenging when the deterministic model is misspecified or when intrinsic stochasticity is present.

Note that the example is inspired by [Brunkhorst \(2022\)](#).

2.1.2.3 Stochastic Extension

As discussed earlier, even when the full mechanistic model of a system is known, parameter estimation can be challenging. One practical approach is therefore to work with a reduced deterministic model and augment it with a stochastic correction term that captures the effects of neglected dynamics. In addition, as pointed out in the previous section, real-world systems can often exhibit intrinsic variability that cannot be explained deterministically (process noise) which can also be incorporated explicitly as a stochastic term.

This leads naturally to a stochastic differential equation (SDE) formulation:

$$d\mathbf{z}(t) = f(\mathbf{z}(t); \boldsymbol{\beta})dt + \sigma(\mathbf{z}(t))d\mathbf{W}_t, \quad (2.15)$$

where $\mathbf{W}_t$ denotes a multivariate Wiener process and $\sigma(\mathbf{z}(t))$ controls the intensity of the stochastic forcing. The deterministic component $f(\mathbf{z}(t); \boldsymbol{\beta})$ is referred to as the *drift* term and describes the average (or mean) evolution of the system. The stochastic component $\sigma(\mathbf{z}(t))d\mathbf{W}(t)$ is called the *diffusion* term and represents random fluctuations around this mean behaviour.

2.1.2.4 Application to Glucose

From a dynamical systems perspective, glucose dynamics of patient i can be described as the observable component of an underlying physiological state, evolving according to equations (2.8)-(2.9):

$$\frac{d\mathbf{z}_i}{dt}(t) = f(\mathbf{z}_i(t), \mathbf{u}_i(t); \boldsymbol{\beta}_i) + \boldsymbol{\delta}_{i,t} \quad (2.16)$$

$$G_i(t) = g(\mathbf{z}_i(t); \boldsymbol{\alpha}) + \varepsilon_t \quad (2.17)$$

where $\mathbf{z}_i(t) = [z_i^{(1)}(t), z_i^{(2)}(t), \dots, z_i^{(g)}(t), \dots, z_i^{(s)}(t)]^\top$ denotes the latent physiological state of patient i . In particular, the component $z_i^{(g)}(t)$ represents the true blood glucose concentration, while $G_i(t)$ corresponds to its noisy realization obtained from patient i 's CGM.⁵

This is a general specification. In what follows, we discuss three approach to glucose modeling using differential equations : a minimal model (Bergman et al., 1979), a physiologically faithful model (Man et al., 2014), and a model relying on stochastic closure (Sirlanci et al., 2023). Note that we drop the i subscript below, but parameters are always inferred at the patient level.

Minimal Model The *minimal model* of Bergman et al. (1979) is among the most influential mechanistic model of glucose regulation. It was introduced to estimate the insulin sensitivity from intravenous glucose tolerance test (IVGTT). It instantiates (2.16) with $\boldsymbol{\delta} = 0$, and a three-dimensional deterministic state $\mathbf{z}(t) = [G(t), X(t), I(t)]^\top$:

$$\begin{cases} \frac{dG}{dt}(t) = (p_1 - X(t)) G(t) - p_1 G_b \\ \frac{dX}{dt}(t) = p_2 X(t) + p_3 (I(t) - I_b) \\ \frac{dI}{dt}(t) = \max(0, \gamma(G(t) - h) t) - \nu I(t) \end{cases} \quad (2.18)$$

where $G(t)$ and $I(t)$ are the plasma glucose and plasma insulin, $X(t)$ is an instrumental variable which is assumed to mediate the action of insulin on glucose disappearance, G_b is baseline glucose, I_b is baseline insulin, p_1 is *glucose effectiveness*. The full parameter vector is $\boldsymbol{\beta} = (p_1, p_2, p_3, \gamma, h, \nu)^\top$, and is estimated using non-linear least squares. Insulin sensitivity index is recovered as $S_I = -p_3/p_2$. The observation equation (2.17) reduces to

⁵ $\boldsymbol{\alpha}$ and the measurement noise are not assumed patient-specific. Differences could stem from the CGM, but are not considered here.

the identity: $g(\mathbf{z}^{(1)}(t)) = G(t)$. Despite its simplicity, the model remains a cornerstone of quantitative endocrinology (Bergman, 2021)⁶.

UVA/Padova Simulator For more comprehensive physiological representation specific to type 1 diabetes, the UVA/Padova simulator (Man et al., 2014) extends (2.16) to a multi-compartment system with over $s = 30$ state variables, encoding gut absorption, endogenous glucose production, and hepatic dynamics, with $\delta = 0$ (no stochasticity). Unlike the minimal model, it is not designed to be identified from individual patient data, but rather to serve as a high-fidelity *in silico* environment for testing closed-loop glucose control algorithms based on a simulated population (of 100 adults, 100 adolescents, and 100 children). It was subsequently approved by the FDA for the development of artificial pancreas systems, illustrating how mechanistic models can encode sufficient physiological complexity to carry regulatory weight.

Stochastic closure Sirlanci et al. (2023) point out that often available data is too sparse to resolve high fidelity models. In their paper, they adopt a stochastic closure approach, that enables effective estimability, at the cost of reduced physiological fidelity. In particular, they instantiate (2.8) with a nonhomogeneous Ornstein-Uhlenbeck diffusion term ($\delta \neq 0$):

$$dG(t) = \alpha(G_b - G(t)) + s_I(t) I(t) + m(t) + \sqrt{2\alpha\sigma^2} dW(t) \quad (2.19)$$

where G_b is basal glucose, α controls the mean-reversion rate, $s_I(t)$ is insulin sensitivity, $I(t)$ is insulin input, and $m(t)$ is the rate of glucose appearance from meals. This directly instantiates the SDE formulation of Section 2.1.2.3, enabling explicit uncertainty quantification over glucose trajectories. Unlike Bergman et al. (1979), the authors here rely on Bayesian estimation framework.

The structure of (2.19) is physiologically interpretable term by term. The drift decomposes into three contributions: (i) a mean-reverting force $\alpha(G_b - G(t))$ pulling glucose back toward its baseline level, (ii) an insulin-mediated disappearance term $s_I(t) I(t)$, and (iii) a meal-driven appearance term $m(t)$. The diffusion term $\sqrt{2\alpha\sigma^2} dW(t)$ captures residual stochasticity not accounted for by the deterministic drift.

2.1.2.5 A change of perspective

These models share a common strength: interpretability. They potentially encode measurable physiological quantities, and their predictions are constrained to be physiologically consistent by construction. But they also share a common limitation: the functional form of f is fixed *a priori*. Any dynamics not anticipated by the modeller cannot be captured regardless of the amount of available data. This misspecification risk motivates a different question:

Rather than prescribing f , can we *discover* it directly from observed trajectories?

This question will be addressed thoroughly in the next section.

⁶We use the differential equation described in Van Herpe et al. (2006), which slightly differ from the original minimal model.

2.2 Modeling Unknown Dynamics

In the previous section, we explored how dynamical systems can be modelled by prescribing f from first principles, relying on domain knowledge. We now turn to the question of learning f directly from observed trajectories, addressing more broadly the *system identification* problem.

We explore two representative philosophies: one seeking to learn a parsimonious representation of the governing equations of a system (Section 2.2.1) ; the other, based on neural networks, trading interpretability for expressivity (Section 2.2.2). This second philosophy, which underpins the neural differential equation literature, is central to the model developed in this work.

2.2.1 SINDy: A Sparsity-Promoting Technique

SINDy provides a natural bridge between mechanistic models and expressive data-driven techniques such as Neural ODEs. Rather than prescribing f from first principles or approximating it with an unconstrained neural network, it learns a parsimonious representation of f from observed snapshots by selecting a small number of active terms in a pre-specified library of candidate functions. Although not used in our experiments, we introduce it briefly to situate the data-driven spectrum before turning to Neural ODEs.

The Sparse Identification of Nonlinear Dynamics (SINDy) algorithm ([Brunton et al., 2016](#)) relies on Occam's razor principle :

"Entia non sunt multiplicanda praeter necessitatem"

| Among all theories explaining the world equally well, the simplest is the best.

Mathematically, given a snapshot matrix $\mathbf{X} \in \mathbb{R}^{N \times o}$ and their time derivatives $\dot{\mathbf{X}} \in \mathbb{R}^{N \times o}$, [Brunton et al. \(2016\)](#) translate this idea by seeking a sparse representation of the vector field f as a linear combination of candidate functions:

$$\dot{\mathbf{X}} \approx \Theta(\mathbf{X}) \Xi \quad (2.20)$$

where $\Theta(\mathbf{X}) \in \mathbb{R}^{N \times l}$ is a library of l candidate functions evaluated on the snapshots, say:

$$\Theta(\mathbf{X}) = [\mathbf{1}, \mathbf{X}, \dots, \mathbf{X}^2, \dots, \sin(\mathbf{X}), \dots]$$

and $\Xi \in \mathbb{R}^{l \times o}$ is a coefficient matrix, estimated using a sparse regression algorithm such as the LASSO ([Tibshirani, 1996](#)). Each column of Ξ thus identifies the active terms in f for one state dimension.

Explicitly, letting $\Theta(\mathbf{x}^\top)$ denote the row vector of candidate functions evaluated at a single state $\mathbf{x} \in \mathbb{R}^o$, the vector field is approximated as:

$$f(\mathbf{x}) \approx \Xi^\top \Theta(\mathbf{x}^\top)^\top$$

Each column $\xi_k \in \mathbb{R}^l$ of Ξ encodes the active terms for the k -th state dimension. This is illustrated for a Lorenz system on Figure 2.7.

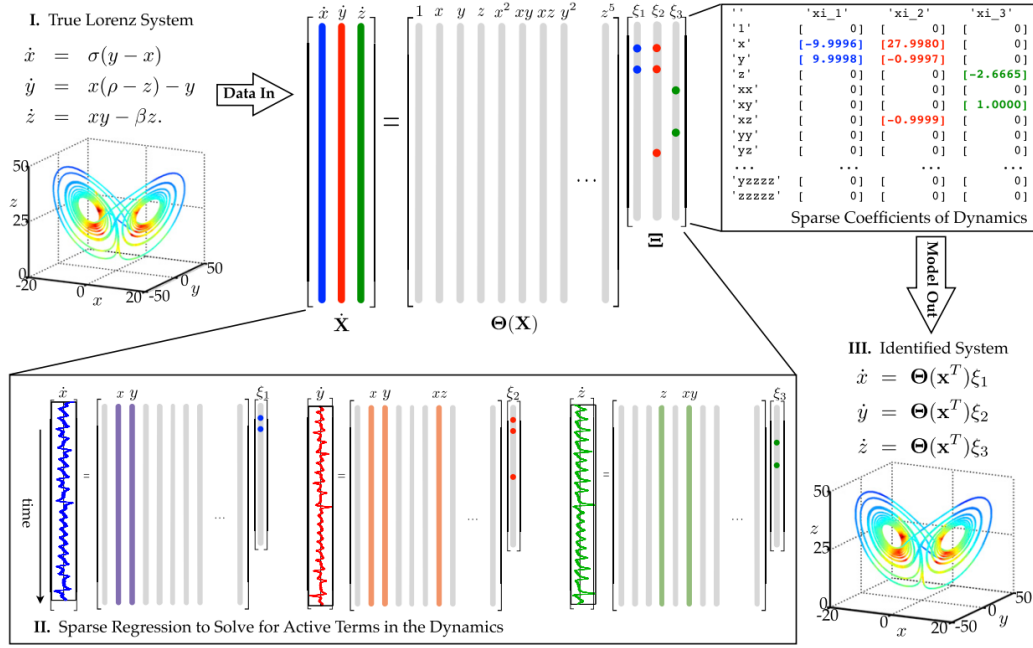


FIGURE 2.7: Schematic representation of SINDy on the Lorenz system. Sparse regression identifies the coefficient matrix Ξ , whose columns ξ are almost entirely zero. The few non-zero entries recover the three governing equations $\dot{x} = \Theta(\mathbf{x}^T)\xi_1$, $\dot{y} = \Theta(\mathbf{x}^T)\xi_2$, $\dot{z} = \Theta(\mathbf{x}^T)\xi_3$. Illustration from (Brunton et al., 2016).

Though interesting by its simplicity and interpretability, SINDy rests on some strong assumptions. First, it assumes that the o observed states are sufficient to explain their own dynamics, *i.e.* that no latent state variables are needed. Second, it assumes that the snapshots are observed in a way that enables the estimation of $\dot{\mathbf{X}}$ ⁷. Third, it assumes that f is a linear combination of some terms of the library.

Note that SINDy has spurred several extensions such as : E-SINDy (Fasel et al., 2022), which accounts for parameter uncertainty through library and observation bootstrapping, SINDy Autoencoder (Champion et al., 2019), which combines SINDy with an autoencoder to identify governing equations in a learned (lower dimensional) latent coordinate system rather than the observation space or, VENI VINDy VICI (Conti et al., 2026), which further extends the Autoencoder framework to a fully Bayesian variational framework for uncertainty-aware equation discovery.

Example 2.2.1 (Fox and Rabbit Population - NODE). *We fit SINDy on our Fox and Rabbit Population, now assuming 1000 observations are available for the sake of simplicity. Estimation of $\dot{\mathbf{X}}$ was performed using a Savitzky-Golay filter (Savitzky and Golay, 1964). The library $\Theta(\mathbf{X})$ is a second order polynomial basis.*

While SINDy recovers well the dynamics with a low noise sample (Figure 2.8), it struggles for a noisier one (Figure 2.9). This is a direct consequence of the difficulty of estimating $\dot{\mathbf{X}}$ from $\mathbf{X}$ in noisier settings. Even for low noise samples, derivatives depends on the chosen method, as illustrated in Appendix C.2. That issue would be further exacerbated for glucose trajectories, which are typically irregularly sampled.

⁷That second issue can be mitigated. For instance, in the supporting material joined to their paper, Brunton et al. (2016) address the noise problem by denoising derivatives via total variation regularization.

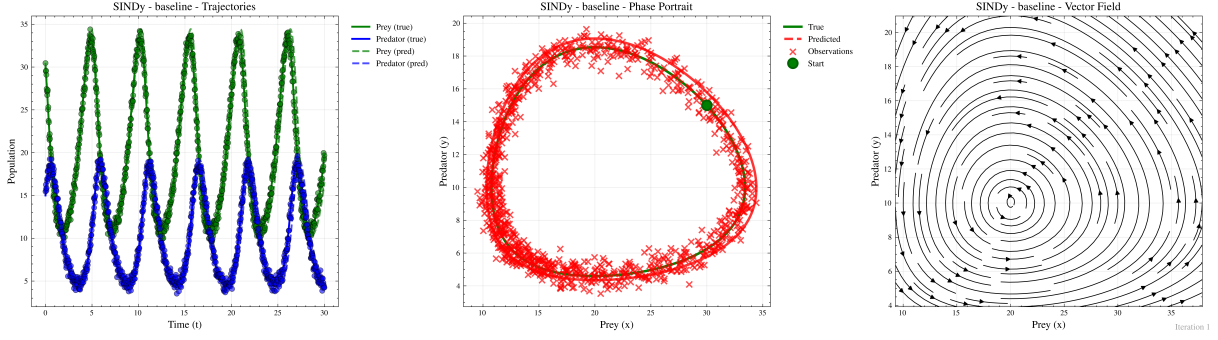


FIGURE 2.8: Baseline Sample, $n = 1000$, gaussian noise $\sigma^2 = 0.5$. **Left** : Predator (blue) and Prey (green) temporal trajectories. **Middle** : Phase-space trajectory. **Right** : Predicted vector field. For the first two panels, observed trajectories are continuous while predicted one are dotted. Observed samples are also displayed

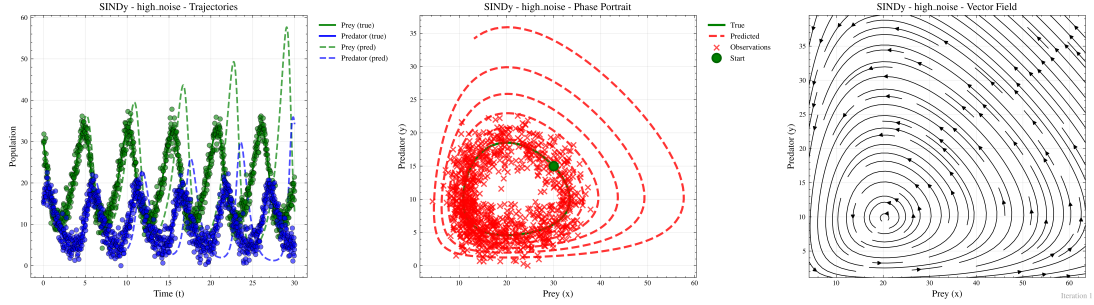


FIGURE 2.9: Noisy Sample, $n = 500$, gaussian noise $\sigma^2 = 2$. **Left** : Predator (blue) and Prey (green) temporal trajectories. **Middle** : Phase-space trajectory. **Right** : Predicted vector field. For the first two panels, observed trajectories are continuous while predicted one are dotted. Observed samples are also displayed

2.2.1.1 Application to Glucose

To the author's knowledge, the only application of SINDy (or any of its variants) to glucose dynamics modeling is [Joedicke et al. \(2022\)](#), who applied SINDy to identify a sparse differential equation governing blood glucose from CGM data of a single type 1 diabetic patient. Using carbohydrate intake and insulin they recover:

$$\begin{aligned} \frac{dG}{dt}(t) = & \kappa_0 + \kappa_1 b(t) + \kappa_2 C(t) + \kappa_3 G(t) + \kappa_4 B(t)b(t) + \kappa_5 b(t)^2 + \kappa_6 b(t)C(t) + \kappa_7 b(t)G(t) \\ & + \kappa_8 C(t)G(t) + \kappa_9 G(t)^2 \end{aligned} \quad (2.21)$$

where $G(t)$ is blood glucose, $C(t)$ carbohydrate intake (time-shifted), $B(t)$ bolus insulin (time-shifted), and $b(t)$ basal insulin.

The estimated parameters are $\hat{\kappa}_0 = -1.14$, $\hat{\kappa}_1 = 102.39$, $\hat{\kappa}_2 = -0.14$, $\hat{\kappa}_3 = -7.69$, $\hat{\kappa}_4 = -0.21$, $\hat{\kappa}_5 = 648.80$, $\hat{\kappa}_6 = 12.80$, $\hat{\kappa}_7 = -185.42$, $\hat{\kappa}_8 = -0.96$, $\hat{\kappa}_9 = 11.39$.

However, the predictive performance remains limited, with a reported RMSE of 36.44 mg/dL (approximately 2.02 mmol/L), which is substantial relative to the normal glycaemic range of 4-10 mmol/L.

2.2.2 Neural ODEs: An Overparametrized Technique

Neural ODEs (Chen et al., 2018) solve some of the limitations of SINDy by replacing the prescribed library Θ with an unconstrained neural network f_θ , trading interpretability for expressivity.

Crucially, in line with the theory of SSMs, it is common in the neural differential equation paradigm to define the dynamics in a latent space $\mathbf{z}(t) \in \mathbb{R}^s$ rather than the observation space⁸. This can be done by leaning on a learnable map $\ell_\theta : \mathbb{R}^o \rightarrow \mathbb{R}^s$, for instance applied to the observed initial condition $\mathbf{x}_0$:

$$\ell_\theta(\mathbf{x}_0) = \mathbf{z}_0, \quad \frac{d\mathbf{z}}{dt}(t) = f_\theta(\mathbf{z}(t)) \quad (2.22)$$

This strategy is known as augmentation and will be discussed more in details in Section 2.2.2.5. It allows to relax one of the SINDy assumption: that the observed variables (or their transformation through Θ) are sufficient to explain the endogenous dynamics. Instead, we assume that a latent representation can be inferred from what has been measured, and then used to perform predictions.

In what follows, we first discuss the rationale behind this parameterization and its connection to ResNets (Section 2.2.2.1), before examining its expressivity (Section 2.2.2.5), and finally how exogenous control can be incorporated into the framework (Section 2.2.2.6).

2.2.2.1 Rationale behind NODEs

The philosophy behind Neural ODEs (NODEs) is quite straightforward : approximate f using a neural network f_θ , usually a feedforward neural network or a shallow multi-layer perceptron (MLP) :

$$\frac{d\mathbf{z}}{dt}(t) = f_\theta(\mathbf{z}(t)) \quad (2.23)$$

The trajectory can then be obtained by numerical integration :

$$\{\mathbf{z}_{t_k}\}_{k=0}^{N-1} = \text{ODESolve}(f_\theta, \mathbf{z}_0, \{t_k\}_{k=0}^{N-1}) \quad (2.24)$$

where `ODESolve` denotes a numerical integration scheme (*e.g.* Runge-Kutta) that propagates the initial condition $\mathbf{z}_0$ forward in time under the dynamics f_θ , returning the (approximated) latent state at each query time t_k .

This idea gained considerable traction with the seminal work of Chen et al. (2018), who observed that the ResNet update rule (He et al., 2016) of the form :

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t; \theta) \quad (2.25)$$

is precisely an Euler discretization of a continuous transformation.

⁸We take the notation $\mathbf{z}$, as in our work the NODE operates in a latent space $\mathbb{R}^s$ with $s > o$. However, a NODE can equivalently be applied directly in observation space, treating the raw snapshots $\mathbf{x}$ as a noisy realizations of the true state, implicitly assuming $\mathbf{x} \approx \mathbf{z}$. This is the setting of Example 2.3.2, where no encoder ℓ_θ is used and the NODE is fitted directly on the observed trajectory.

We can understand that statement by looking back at the Euler Explicit method (2.6), and setting $\Delta t = 1$. Equation 2.6 then becomes :

$$\begin{aligned}\mathbf{z}(t + \Delta t) &\approx \mathbf{z}(t) + \Delta t f_{\theta}(\mathbf{z}(t)) \\ \mathbf{z}(t + 1) &\approx \mathbf{z}(t) + 1 f_{\theta}(\mathbf{z}(t))\end{aligned}$$

This means that residual networks are *idealized as continuous-time dynamical systems*. We replace the discrete T layers by a continuous variable t , resulting in a continuous-time dynamics described by an ODE, parametrized by a neural network (Li et al., 2022).⁹

This idealization can be understood in many ways. One of them is that, instead of relying on recursive architectures, which require function composition for which we lack mathematical understanding, one can now enjoy the full toolbox of dynamical systems theory.

On the other hand, and not the least, since `ODESolve` can be queried at arbitrary time points, NODEs are naturally suited to handle irregularly sampled data. It also inherits the properties of numerical solvers introduced earlier, *i.e.* the ability to perform interpolation, extrapolation, and counterfactual simulations.

The introduction of NODEs sparked a rich line of work on *neural differential equations*, including Latent ODE (Chen et al., 2018), ODE-RNN (Rubanova et al., 2019), ODE-LSTM (Lechner and Hasani, 2020), Neural CDEs (Kidger et al., 2020), GRU-ODE-Bayes (De Brouwer et al., 2019), CF-ODE (De Brouwer et al., 2022), IMODE (Gwak et al., 2020), and many others.

We refer the reader to Kidger (2022) for a rigorous and in-depth treatment of neural differential equations, and to Oh et al. (2025) for a recent survey of this landscape. Applications to medical time series are discussed in depth in De Brouwer (2023).

2.2.2.2 Overparametrization

Obviously, the parametrization of f with a neural network is far away from Occam’s razor principle. The cardinality of θ will vastly exceed that of the true vector field parameters, and there is no reason to expect that f_{θ} will recover the true governing equations.

Nevertheless, f_{θ} remains a powerful tool for approximating the true trajectory. By construction, the solution of a NODE is continuous and smooth, providing a dense, differentiable approximation of the latent state, even from sparse, irregularly sampled observations. This smooth trajectory can then serve as a substrate for downstream analysis¹⁰.

⁹Note that strictly speaking, NODEs cannot be considered the true deep limit of ResNets (Massaroli et al., 2020). In a ResNet, each residual block has its own parameter vector θ_t , whereas a NODE uses a single shared f_{θ} with depth as a continuous variable. The true deep limit would require a hypernetwork parametrizing $\theta(t)$ as a function of t (Zhang et al., 2019; Ha et al., 2016).

¹⁰One can, for instance, apply SINDy on the recovered $\hat{\mathbf{z}}(t)$ to attempt identifying the true governing equations, as illustrated in Figure 2.14.

2.2.2.3 Incorporating Observations

Parameter estimation requires incorporating observations in the model. In Section 2.1.2, this was achieved via a probabilistic observation model and Bayesian inference; in Section 2.2.1, regression was performed directly on the snapshots. For Neural ODEs, several strategies have been developed. The most straightforward, standard in the deep learning literature, optimizes θ by minimizing a reconstruction loss of the form:

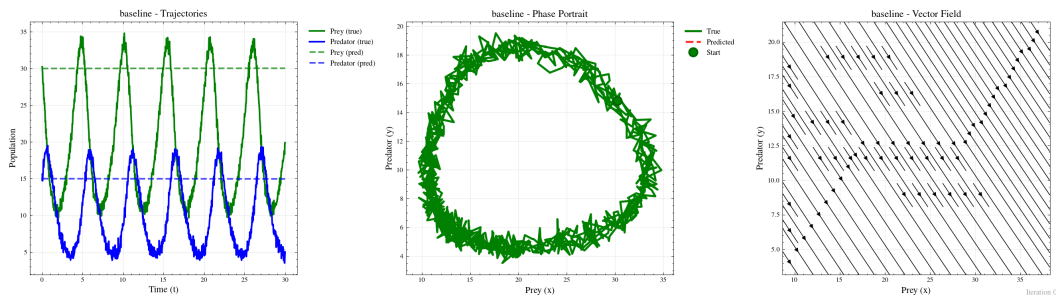
$$\theta^* = \arg \min_{\theta \in \Theta} \frac{1}{B} \sum_{b=1}^B \mathcal{L} \left(\{\mathbf{x}_{t_k}^{(b)}\}_{k \notin \mathcal{K}_{\text{mask}}^{(b)}}, \{\hat{\mathbf{z}}(t_k; \theta, \mathbf{z}_0^{(b)})\}_{k \notin \mathcal{K}_{\text{mask}}^{(b)}} \right) \quad (2.26)$$

where $b = 1, \dots, B$ indexes randomly sampled sub-windows of the observed trajectory (or trajectories), $k_0^{(b)} \in \{0, \dots, N - T_b\}$ is the starting index, T_b is the window length, $\mathcal{K}_{\text{mask}}^{(b)} \subseteq \{k_0^{(b)}, \dots, k_0^{(b)} + T_b - 1\}$ is a random observation mask, and $\mathbf{z}_0^{(b)} := \mathbf{x}_{t_{k_0^{(b)}}}$ is the observed state at the window start, used directly as initial condition for the ODE solver.

The point is that at each iteration, both the starting index $k_0^{(b)}$ and the observation mask $\mathcal{K}_{\text{mask}}^{(b)}$ are resampled, exposing the model to different temporal windows and different observation patterns. This encourages the learned f_θ to generalize beyond the specific times and densities seen during training.

Example 2.2.2 (Fox and Rabbit Population - NODE). *We now train a classical NODE on the same Fox and Rabbit Population as SINDy. f_θ is a fully connected MLP ($2 \rightarrow 50 \rightarrow 50 \rightarrow 2$) with tanh activations. Training follows a curriculum strategy in which batches consist of $B = 20$ sub-trajectories of length T_b , each starting from a randomly sampled initial condition drawn from the observed trajectory (without replacement). The window length T_b is progressively increased over training: starting at $T_b = 10$ for the first 500 iterations, then extended to 20, 50, and finally 150, forcing the model to capture longer dependencies as training progresses. The model evaluated at the beginning, middle, and end of the training, is presented in Figure 2.10.*

Figure 2.11 illustrates the same architecture, and same training procedure, on a twice lower sparser dataset ($N = 500$, with irregularly sampled data). We see that, thanks to the masking procedure, NODE is still able to recover well the anulus structure and interpolate observed trajectories.



(A) Iteration 0

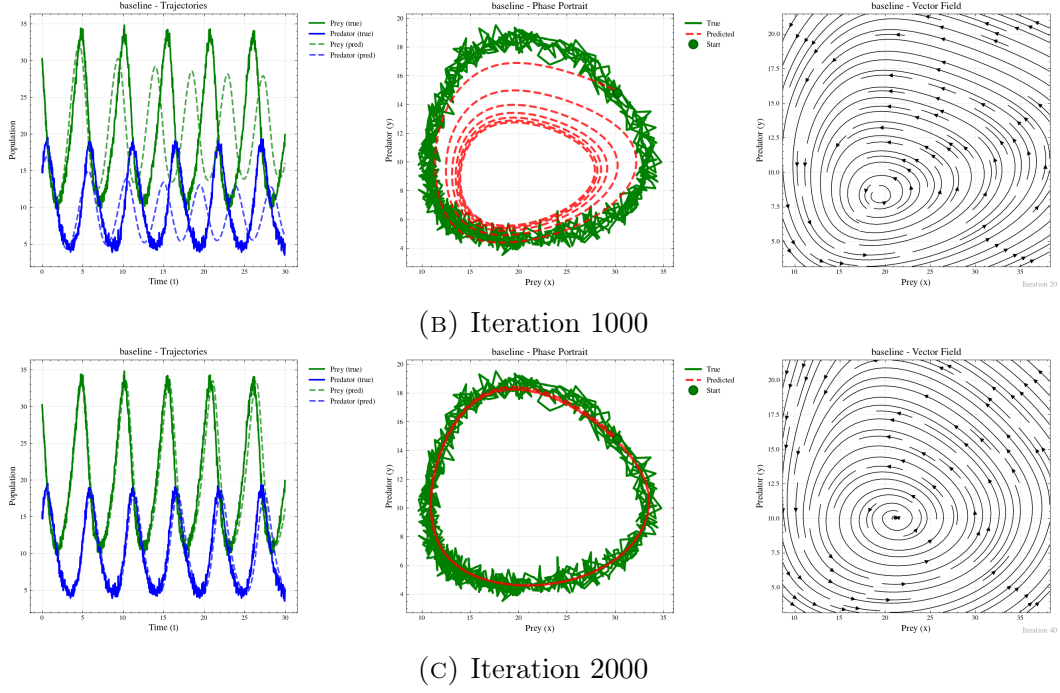


FIGURE 2.10: Evolution of the learned vector field f_θ at successive training iterations. For each sub-figure, **Left** : Predator (blue) and Prey (green) temporal trajectories. **Middle** : Phase-space trajectory. **Right** : Predicted vector field. For the first two panels, observed trajectories are continuous while predicted ones are dotted.

2.2.2.4 Out-of-sample performance

A common pitfall of deep learning methods is *overfitting*. Because neural networks are highly expressive function approximators, they can memorize the training data rather than learning the underlying structure, and subsequently fail to generalize to unseen inputs.

Intuitively, for a NODE, each training trajectory acts as a constraint on f_θ . When only a single trajectory is available, many vector fields can satisfy it (the problem is underdetermined). As more diverse trajectories are added, the feasible set of vector fields shrinks, progressively forcing f_θ closer to the true underlying dynamics. This is illustrated in Figure 2.12.

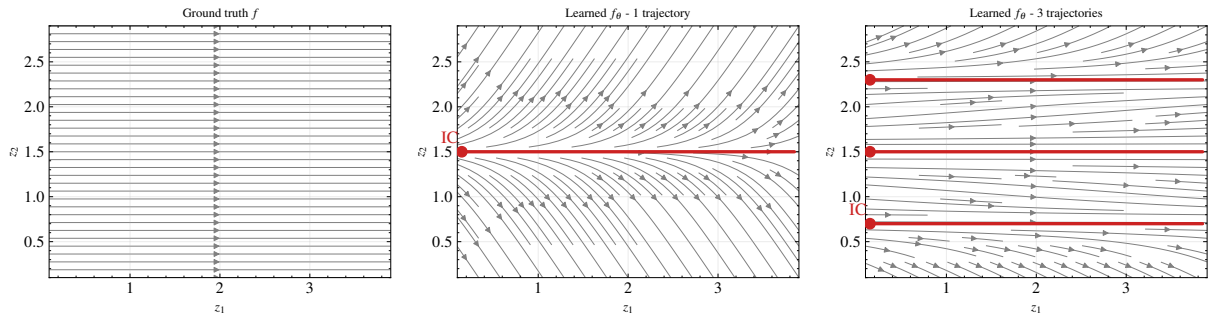


FIGURE 2.12: Illustration of how training trajectory diversity constrains the learned vector field f_θ (synthetic example).

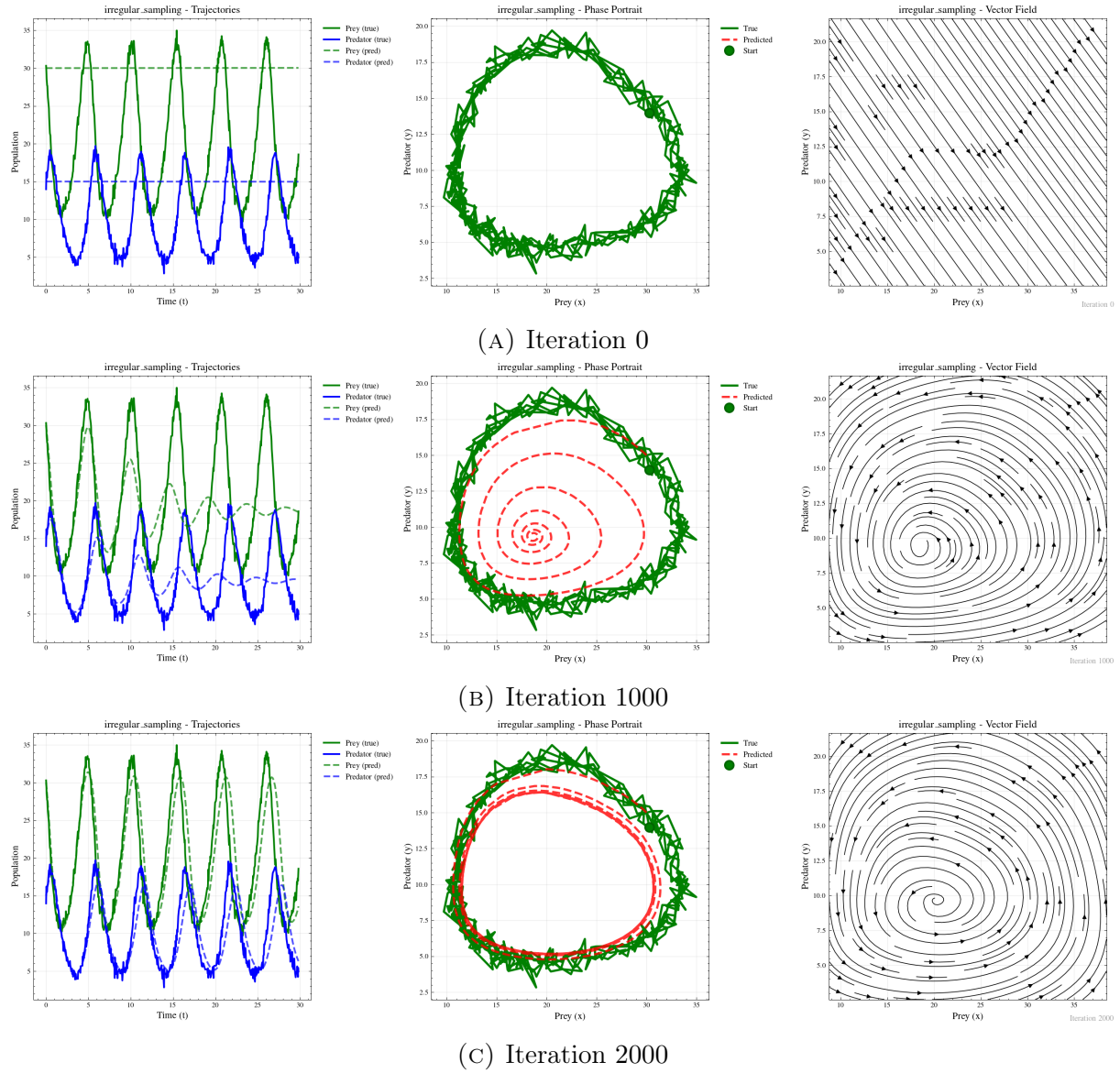


FIGURE 2.11: Evolution of the learned vector field f_θ at successive training iterations for a smaller ($n = 500$) dataset, with irregular sampling. For each sub-figure, **Left** : Predator (blue) and Prey (green) temporal trajectories. **Middle** : Phase-space trajectory. **Right** : Predicted vector field. For the first two panels, observed trajectories are continuous while predicted one are dotted.

Example 2.2.3 (Fox and Rabbit Population - Overfitting). *We now consider the NODE of the previous example, which was trained on a single Lotka-Volterra trajectory, whose initial condition was $\mathbf{z}_0 = [30, 15]^\top$. We evaluate the trained f_θ on three held-out initial conditions drawn from a different region of phase space. Figure 2.13 shows the predicted trajectories against the ground truth.*

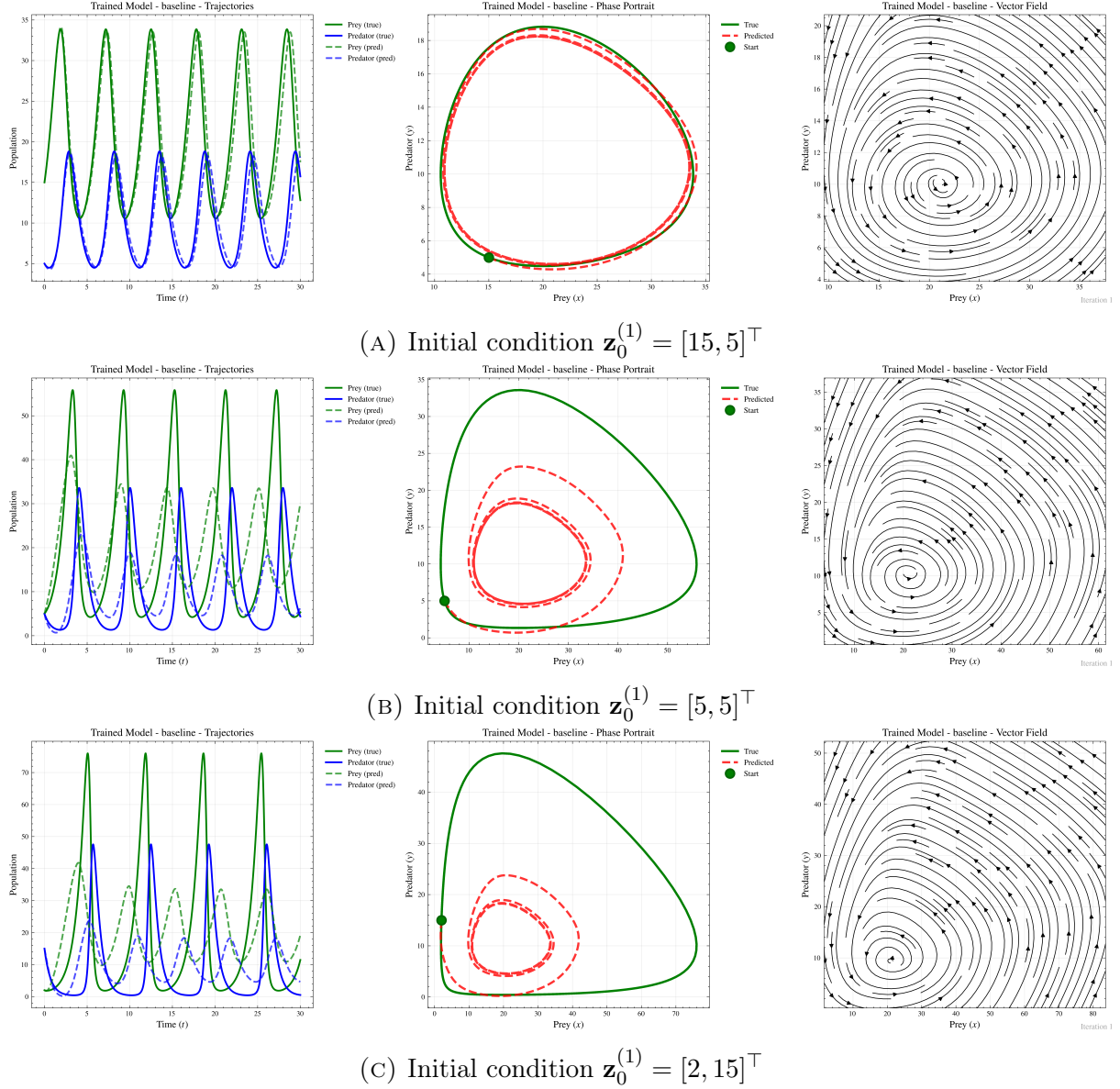


FIGURE 2.13: Out-of-sample predictions for three held-out initial conditions. For each sub-figure, **Left** : Predator (blue) and Prey (green) temporal trajectories. **Middle** : Phase-space trajectory. **Right** : Predicted NODE vector field. For the first two panels, ground truth trajectories are continuous while predicted one are dotted.

We see that the further we go from the training initial conditions, the worse it gets. In particular, we see that both IC 2 and 3 converge to the orbit of the observed IC $\mathbf{z}_0 = [30, 15]^\top$. In order to generalize, it is often sufficient to train the model on a more diverse set of initial conditions.

Another approach could be to use a "NODE-then-SINDy" pipeline. Based on smoothed NODE trajectories, this approach enables a much more robust estimation of $\dot{\mathbf{X}}$, thereby allowing for faithful extrapolation outside the observed orbit.

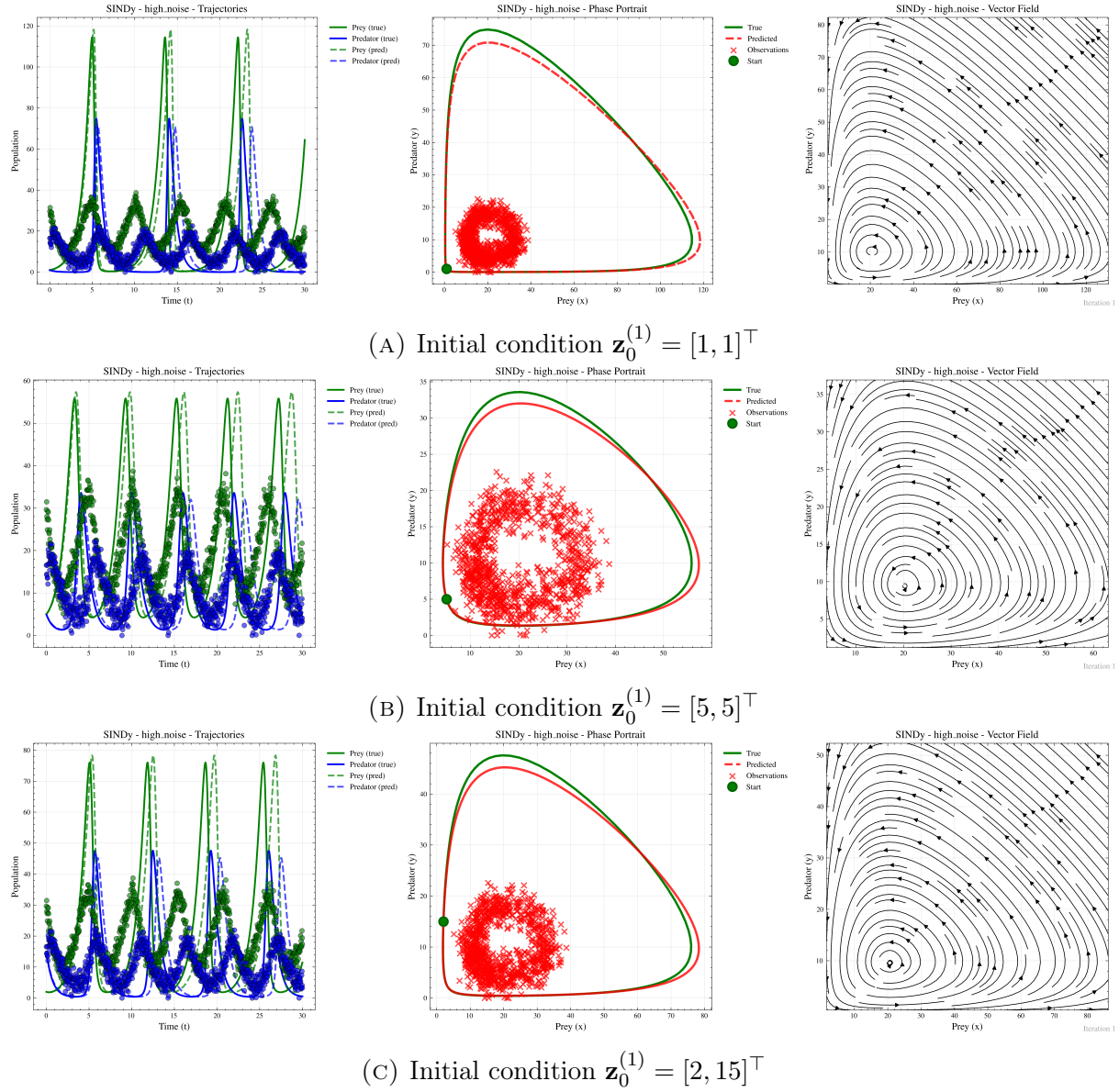


FIGURE 2.14: Out-of-sample predictions for three held-out initial conditions. For each sub-figure, **Left** : Predator (blue) and Prey (green) temporal trajectories. **Middle** : Phase-space trajectory. **Right** : Predicted vector SINDy field. For the first two panels, ground truth trajectories are continuous while predicted one are dotted. Observed samples are also displayed.

2.2.2.5 Universal Approximation and Augmentation

It is well established that a feedforward network with at least one hidden layer and a sigmoidal activation function can approximate any Borel-measurable function to arbitrary precision (Hornik et al., 1989), provided enough hidden units¹¹. Since f_θ is typically parametrized by such a network, one might expect NODEs to inherit this property. This turns out to be false in general, for a fundamental reason rooted in the geometry of ODE flows (See Appendix A.1).

In practice, one may still restore the universal approximation property, by augmenting (*i.e.* lifting) the state space dimension, resulting in the so-called *Augmented Neural ODE* (ANODE) model family.

By evolving in the higher-dimensional state space $\mathbb{R}^s$, the ODE avoid topological obstacles, learning simpler flows at lower computational cost (Dupont et al., 2019; Massaroli et al., 2020).

Beyond the topological argument, this is what we did intuitively in the introduction with Equation 2.22. Since $\mathbf{x}(t) \in \mathbb{R}^o$ cannot fully explain its own dynamics, the learnable map $\ell_\theta : \mathbb{R}^o \rightarrow \mathbb{R}^s$ with $s > o$ lifts the observations into an augmented space where the dynamics can be more easily and properly represented.

Formally, ANODE lifts the state space from $\mathbb{R}^o$ to $\mathbb{R}^s$ with $s > o$, by applying a learnable map $\ell_\theta : \mathbb{R}^o \rightarrow \mathbb{R}^s$ to the initial condition:

$$\mathbf{z}(0) = \ell_\theta(\mathbf{x}(0)), \quad \frac{d\mathbf{z}}{dt}(t) = f_\theta(\mathbf{z}(t)) \quad (2.27)$$

As stressed by Kidger (2022), common choices for ℓ_θ are : (i) zero-augmentation $\ell_0(\mathbf{x}) = [\mathbf{x}, \mathbf{0}]^\top$, (ii) learned augmentation $\ell_\theta(\mathbf{x}) = [\mathbf{x}, \tilde{\ell}_\theta(\mathbf{x})]^\top$ for some learned $\tilde{\ell}_\theta$, (iii) a learnable affine map $\ell_\theta(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$ with $A \in \mathbb{R}^{s \times o}$, $\mathbf{b} \in \mathbb{R}^s$, stacked into θ .

Another interesting property to point out is that ANODE are universal approximators, even if their vector field are not (Kidger, 2022). We summarize the rationale of Kidger (2022) in proposition 2.2.1.

Proposition 2.2.1 (Universal Approximation Property of ANODEs, Kidger (2022)). *ANODEs are universal approximators. Two cases arise:*

1. *If f_θ is itself a universal approximator, then for fixed augmented dimension s , complexity is obtained through the vector field. This is analogous to the deep and narrow universal approximation theorem for feedforward networks.*
2. *If f_θ is not a universal approximator, then the augmented dimension $s \rightarrow \infty$ compensates, with complexity obtained through the affine maps. This is analogous to the classical wide universal approximation theorem.*

This proposition is particularly relevant to the architecture reviewed in Section 2.3.3.3, that consists of an augmentation. Massaroli et al. (2020) warn in their section "*Mind your input network*" that in encoder-NODE architectures, an overly expressive encoder may resolve the task before the NODE integration begins, causing f_θ to collapse toward an identity map.

¹¹This result guarantees the *existence* of a network that can represent the function, but not that the training algorithm will find it. Learning can fail for two distinct reasons: the optimizer may not converge to the desired parameters, or the model may overfit and fail to generalize (Goodfellow et al., 2016).

Example 2.2.4 (Fox and Rabbit Population - ANODE). *In the Example 2.3.3, we did not use augmentation, as we assumed that no variable other than the Prey and Predator populations were needed. Augmenting the statespace should be pointless here, as there are no latent variables nor topological obstacles. Here, for the sake of completeness of the examples, we illustrate augmentation on the Lotka-Volterra population. By adding an additional 0 dimension, we still recover the annulus structure of the phase space, as visible on Figure 2.15. All the training procedure is identical, except for the number of iteration (fixed to 2500) and learn a vector field $f_\theta : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, which gains a new dimension.*

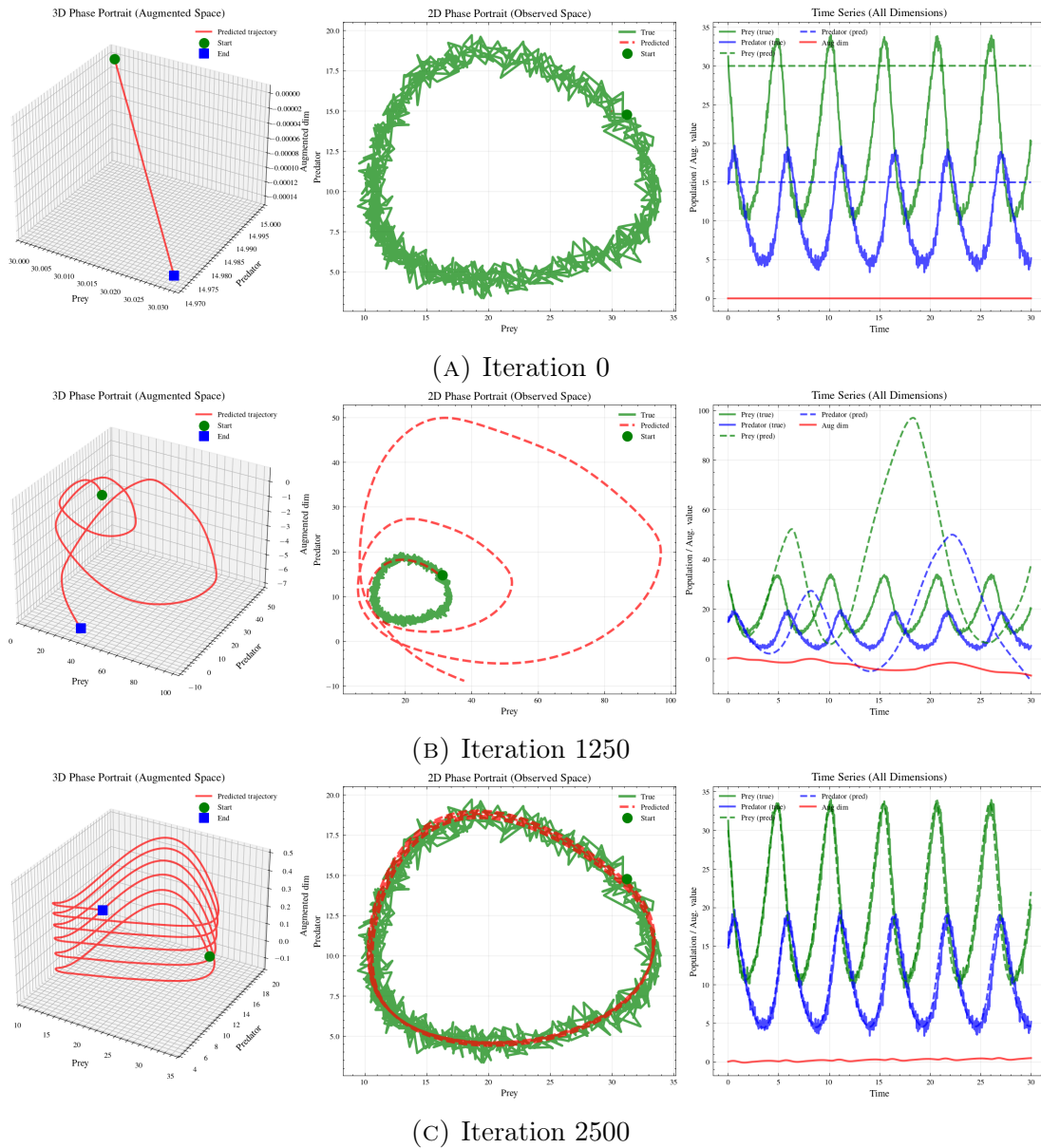


FIGURE 2.15: Evolution of the learned vector field f_θ at successive training iterations. For each sub-figure, **Left** : Augmented Phase-space trajectory. **Middle** : Observed Phase-space trajectory. **Right** : Predator (blue), Prey (green) and 0-augmented temporal trajectories. The predicted trajectory is presented in red.

2.2.2.6 Incorporating Exogenous Control

So far, the vector field f_θ has been assumed to govern a *closed* system, where the dynamics depend solely on the current state $\mathbf{z}(t)$, with no external inputs, or control.

In many real-world settings, however, the system is *open* and subject to *exogenous inputs* $\mathbf{u}(t)$ that perturb its trajectory without being governed the system's own rules, meaning that these inputs force the system from outside without being explained by it.

This distinction is often overlooked in the neural differential equations literature, where $\mathbf{u}(t)$ is silently absorbed into an augmented state vector $\tilde{\mathbf{z}}$, yielding:

$$\frac{d\tilde{\mathbf{z}}}{dt}(t) = f_\theta(\tilde{\mathbf{z}}(t)) \quad (2.28)$$

where $f_\theta : \mathbb{R}^{s+p} \rightarrow \mathbb{R}^{s+p}$.

This formulation is problematic. By including $\mathbf{u}(t)$ into $\tilde{\mathbf{z}}$, we implicitly treat $\mathbf{u}$ as an endogenous variable governed by a learned ODE, when it should instead be read as an external signal at each solver step.

The proper formulation keeps $\mathbf{u}(t)$ as an argument to f_θ , not as a component of the state:

$$\frac{d\mathbf{z}}{dt}(t) = f_\theta(\mathbf{z}(t), \mathbf{u}(t)) \quad (2.29)$$

where $f_\theta : \mathbb{R}^{s+p} \rightarrow \mathbb{R}^s$. Here $\mathbf{u}(t)$ is *read* at each solver step via external interpolation, but is not treated as a state variable with its own dynamics.

The trajectory is then given by:

$$\mathbf{z}(t) = \mathbf{z}_0 + \int_{t_0}^t f_\theta(\mathbf{z}(\tau), \mathbf{u}(\tau)) d\tau \quad (2.30)$$

which is approximated at desired time points as:

$$\{\mathbf{z}(t_k)\}_{k=0}^{N-1} = \text{ODESolve}(f_\theta, \mathbf{z}_0, \{t_k\}_{k=0}^{N-1}, \{\mathbf{u}\}_{k=0}^{N-1}) \quad (2.31)$$

Similarly, [Brunton et al. \(2016\)](#) make this point by applying SINDy to a forced Lotka-Volterra system without taking $\mathbf{u}(t)$ as an exogenous input, yielding an unstable model that fails to generalize. In contrast, which explicitly includes functions of $\mathbf{u}(t)$ in the candidate library, correctly recovers the governing equations.

Note that (2.29) rests on an *exogeneity assumption*: $\mathbf{u}(t)$ is treated as a given signal, independent of $\mathbf{z}(t)$. This is a modeling choice, not a physical fact. In reality, feedback loops may exist (*e.g.*, low glucose triggering hunger and subsequent food intake). We adopt this assumption throughout, as is standard in the control and system identification literature.¹²

Finally, (2.29) requires $\mathbf{u}(t)$ to be available as a continuous signal, whereas in practice control inputs are recorded at discrete times. Evaluating $\mathbf{u}(t)$ between observations therefore

¹²We also believe that this it is a reasonable assumption to make as meal timing is often governed by social and behavioural schedules, and can thus be seen as independent of the physiological state.

requires either treating events as instantaneous impulses (*i.e.* using a Dirac representation) or performing an explicit transformation (*e.g.* interpolation), which, in the NODE framework, is typically *ad hoc* and decoupled from the model. The specific parameterisation of $\mathbf{u}(t)$ adopted in the present work is discussed in Section 3.1.2.

Neural Controlled Differential Equations (Section 2.3.2) take another approach by constructing a continuous path directly from the data and letting it drive the latent dynamics via a Riemann-Stieltjes integral, absorbing the interpolation step into the architecture itself.

2.2.2.7 Application to glucose

To our knowledge, no paper have tried to model glucose dynamics relying solely on NODEs. Still, the closely related universal differential equations (UDEs), which hybridizes a mechanistic model f with a flexible neural correction term f_θ :

$$\frac{d\mathbf{z}}{dt}(t) = f(\mathbf{z}(t); \boldsymbol{\beta}) + f_\theta(\mathbf{z}(t)) \quad (2.32)$$

have been applied to glucose modeling through the Hybrid² Neural ODE Causal Modeling (H²NCM, Zou et al. (2024)) architecture.

The latter is given the qualificative Hybrid² as it relies on two types of inductive bias¹³: (i) a soft bias, taking the form of a causal loss regularization term that encourages correct ranking of intervention effects, and (ii) a hard bias, taking the form of reduced UVA/Padova mechanistic model f .

We do not explore this model further, as it differs from our setting along several dimensions: it targets post-exercise glycaemic response in individuals with type 1 diabetes rather than postprandial dynamics in a healthy population, and operates in a deterministic framework rather than a latent probabilistic one. Nevertheless, we still point out to the article as two of its ideas remain relevant future directions: (i) incorporating a causal loss to guide the learning of more physiologically meaningful latent representations, and (ii) further hybridizing the dynamics with mechanistic components.

¹³For more details on UDEs and the notion of inductive bias, see details Appendix A.2.

2.3 Neural Differential Equations and Glycemia

In the previous section, we shed light on two philosophically different ways of approaching data-driven dynamical system modeling (SINDy and NODE). We already established that the last approach, NODE, would be particularly interesting to model glucose dynamics, as it natively handles continuous-time processes and irregularly sampled data.

While the vanilla NODE can be effective (Section 2.2.2.3), other architectures have been introduced to incorporate observations more elegantly. De Brouwer (2023) organizes the most prominent modeling strategies under three families. First, the *Deep SSM* family¹⁴ takes inspiration directly from the Bayesian SSM framework of Section 2.1.2.2. Second, the *interleaving approach* is a continuous-time extension of the recurrent neural network family. Third, Neural CDEs, representing the *interpolation* family, are grounded in the theory of controlled differential equations.

Figure 2.16 situates the three families chronologically. The *Deep SSM* family emerged with the seminal Latent ODE of Chen et al. (2018), submitted in 2018; the *interleaving approach* followed in 2019 with the ODE-RNN of Rubanova et al. (2019); and the *interpolation* family was introduced with Neural CDEs (Kidger et al., 2020) in 2020. We address in turn each of these modeling approaches and their applications to glucose in Sections 2.3.1 to 2.3.3.



FIGURE 2.16: Chronology of the three NDE families based on submission year.

2.3.1 Interleaving approach

As illustrated on Figure 2.17, the interleaving approach consists of an alternation of two operations: a discrete *update* of the hidden state $\mathbf{h}$ upon reception of a new observation, and a continuous *evolution* of $\mathbf{h}$ during the interval separating consecutive observations.

Following Fitzgerald et al. (2023), we can present all architectures of the interleaving family under an overarching framework. Let $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{y}_k, t_k)\}_{k=0}^{N-1}$ be an irregularly-sampled dataset of input-target pairs, where $\mathbf{x}_k \in \mathbb{R}^{d_x}$ may encompass observations, previous predictions, missingness masks and additional covariates, and $\mathbf{y}_k \in \mathbb{R}^{d_y}$ is the associated target at time t_k . An interleaving model is defined by an update map ϕ_θ , an evolve map Φ_θ , and an output map g_θ , operating as:

$$\mathbf{h}(t_k^+) = \phi_\theta(\mathbf{h}(t_k^-), \mathbf{x}_k) \quad (2.33)$$

$$\mathbf{h}(t) = \Phi_\theta(\mathbf{h}(t_k^+), t_k^+, t) \quad t \in [t_k^+, t_{k+1}^-] \quad (2.34)$$

$$\hat{\mathbf{y}}(t) = g_\theta(\mathbf{h}(t)) \quad (2.35)$$

¹⁴That De Brouwer (2023) terms "autoencoder".

where $\mathbf{h}(t_k^+)$ ($\mathbf{h}(t_k^-)$) denotes the hidden state just after (before) the update, and θ collects all parameters.

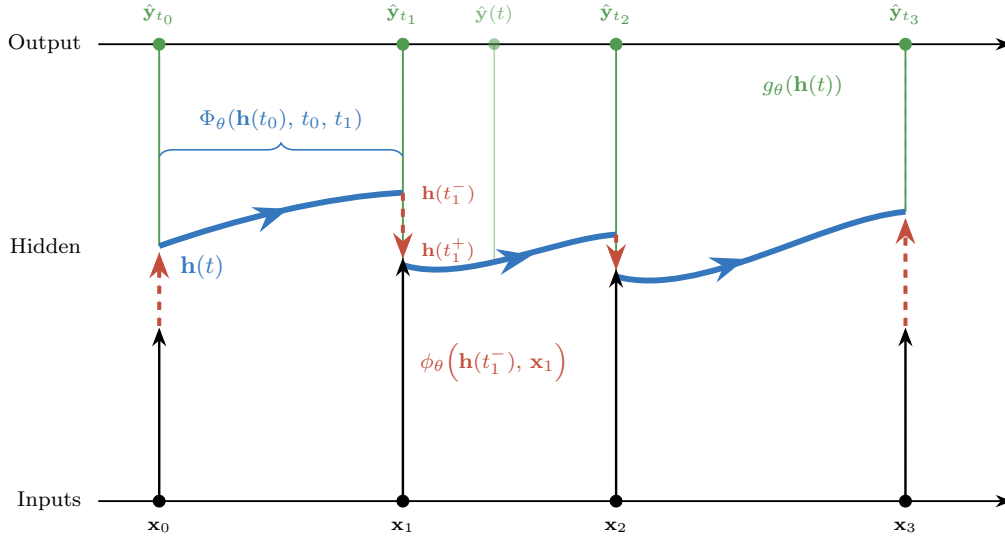


FIGURE 2.17: Schematic of the interleaving approach. At each observation time t_k , the hidden state is updated discontinuously by ϕ_θ . Between observations, $\mathbf{h}(t)$ evolves continuously according to Φ_θ . The output map g_θ can be queried at any time.

In the following sections, we explore some instances of the interleaving framework. In Section 2.3.1.1, we introduce the celebrated Recurrent Neural Networks (RNNs) family, which correspond to the case where Φ_θ is the identity. Then, in Section 2.3.1.3, we discuss the ODE-RNN and ODE-LSTM architectures, which parameterize Φ_θ as the flow map of a NODE, allowing inter-observation dynamics to be learned from data. Finally, Section 2.3.1.4 points out that, similarly to the vanilla NODE, most approaches in the interleaving literature do not distinguish between exogenous control variables and endogenous realizations of the underlying dynamical system.

2.3.1.1 Recurrent Neural Networks

RNNs were originally introduced by [Rumelhart et al. \(1986\)](#) to process sequences of variable length by relying on *parameter sharing*, meaning that the same parameter vector θ governs every update step, regardless of sequence length.

Given an input sequence $\{\mathbf{x}_k\}_{k=0}^{N-1}$, the hidden state $\mathbf{h}_k \in \mathbb{R}^{d_h}$ and d_y -dimensional output $\hat{\mathbf{y}}_k$ are sequentially updated thanks to:

$$\mathbf{h}_k = \sigma_1(\mathbf{W}_h \mathbf{h}_{k-1} + \mathbf{W}_x \mathbf{x}_k + \mathbf{b}_h) \quad (2.36)$$

$$\hat{\mathbf{y}}_k = \sigma_2(\mathbf{W}_y \mathbf{h}_k + \mathbf{b}_y) \quad (2.37)$$

where $\mathbf{W}_h \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{W}_x \in \mathbb{R}^{d_h \times d_x}$, $\mathbf{W}_y \in \mathbb{R}^{d_y \times d_h}$, $\mathbf{b}_h \in \mathbb{R}^{d_h}$, $\mathbf{b}_y \in \mathbb{R}^{d_y}$ are the learnable parameters, collected in θ . σ_1 is usually a tanh or sigmoid activation, while σ_2 is dependent on the modeling goal (*e.g.* sigmoid for binary classification or softmax for multilabel classification).

Equation (2.36) corresponds to the update map, and (2.37) corresponds to the output map. There is virtually no evolve map for the vanilla RNN as the hidden state remains constant between observations.

It transpires that RNNs are inherently linked to dynamical systems. Looking back to the equations of a SSM (2.9), we see that our RNN can be seen as a discretized SSM. Many other links with the dynamical system theory can be made, see for instance [Brunton and Kutz \(2022\)](#) or [Pascanu et al. \(2013\)](#).

Vanilla RNNs feature many drawbacks, justifying moving towards more advanced architectures. One of the most well known flaws of RNNs are the so-called *vanishing* and *exploding* gradient problems, which make challenging to learn long-term dependencies ([Pascanu et al., 2013](#))¹⁵.

2.3.1.2 Gated Architectures

Gated architectures were specifically designed to overcome the vanishing gradient¹⁶ and short-term memory problems of RNNs.

The first gated architecture, introduced by [Hochreiter and Schmidhuber \(1997\)](#), is the Long Short-Term Memory (LSTM). It overcomes the vanishing gradient and short-term memory problems by introducing two distinct cell states $\mathbf{h}$ and $\mathbf{C}$:

$$\begin{aligned}\mathbf{h}_k &= O_k \odot \tanh(\mathbf{C}_k) \\ \mathbf{C}_k &= F_k \odot \mathbf{C}_{k-1} + I_k \tilde{\mathbf{C}}_k\end{aligned}$$

relying on a candidate memory state

$$\tilde{\mathbf{C}}_k = \tanh(\mathbf{W}_x^{(c)} \mathbf{x}_k + \mathbf{W}_h^{(c)} \mathbf{h}_{k-1} + \mathbf{b}^{(c)})$$

and three learnable gates

- Forget gate : $F_k = \sigma(\mathbf{W}_x^{(f)} \mathbf{x}_k + \mathbf{W}_h^{(f)} \mathbf{h}_{k-1} + \mathbf{b}^{(f)})$
- Input gate : $I_k = \sigma(\mathbf{W}_x^{(in)} \mathbf{x}_k + \mathbf{W}_h^{(in)} \mathbf{h}_{k-1} + \mathbf{b}^{(in)})$
- Output gate : $O_k = \sigma(\mathbf{W}_x^{(o)} \mathbf{x}_k + \mathbf{W}_h^{(o)} \mathbf{h}_{k-1} + \mathbf{b}^{(o)})$

where $\odot$ is the Hadamard (element-wise) product, σ is a sigmoid activation, $\mathbf{W}_x^{(\cdot)} \in \mathbb{R}^{d_h \times d_x}$, $\mathbf{W}_h^{(\cdot)} \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{b}^{(\cdot)} \in \mathbb{R}^{d_h}$.

Because the sigmoid activation $\sigma : \mathbb{R} \rightarrow [0, 1]$, the three gates act as switches (proportions) that regulate the information flux in the cell. First, the forget gate determines how much of the previous long-term memory is retained (*i.e.* "not forgotten"). Second, the input gate determines how much of candidate long-term memory state $\tilde{\mathbf{C}}_k$ is incorporated into the updated long-term memory. Third, the output gate determines what parts of the updated cell state is transmitted to the hidden state.

For a LSTM, the output map takes the same form as the Vanilla RNN, *i.e.* $\hat{\mathbf{y}}_k = \sigma_2(\mathbf{W}_y \mathbf{h}_k + \mathbf{b}_y)$, for some activation σ_2 .

¹⁵Vanishing (Exploding) gradient refers to when the norm of the gradients decreases (increases) exponentially towards 0 (∞).

¹⁶The exploding gradient problem, on the other hand, is often tackled using a gradient clipping strategy, which mechanically clip the gradient when it exceeds a user-defined threshold.

Gated Recurrent Units (GRUs) were then introduced (Chung et al., 2014) as a more parsimonious alternative to LSTMs, reducing the number of gates from three to two :

- Reset gate: $R_k = \sigma(\mathbf{W}_x^{(r)} \mathbf{x}_k + \mathbf{W}_h^{(r)} \mathbf{h}_{k-1} + \mathbf{b}^{(r)})$
- Update gate: $Z_k = \sigma(\mathbf{W}_x^{(z)} \mathbf{x}_k + \mathbf{W}_h^{(z)} \mathbf{h}_{k-1} + \mathbf{b}^{(z)})$

and eliminating the separate cell state $\mathbf{C}_k$, using only a hidden state :

$$\begin{aligned}\tilde{\mathbf{h}}_k &= \tanh(\mathbf{W}_x^{(c)} \mathbf{x}_k + \mathbf{W}_h^{(c)} (R_k \odot \mathbf{h}_{k-1}) + \mathbf{b}^{(c)}) \\ \mathbf{h}_k &= (1 - Z_k) \odot \mathbf{h}_{k-1} + Z_k \odot \tilde{\mathbf{h}}_k\end{aligned}$$

The update gate Z_k blends the previous hidden state and the candidate, playing the combined role of the LSTM's forget and input gates.

Those three architecture fall in the scope of classical recurrent neural network, where hidden states are kept constant between observations. A representation of each architecture's cell is given in Figure 2.18

These architectures have long been state-of-the-art for sequential data where the order of observations matters but elapsed time does not (*e.g.* in some NLP applications, where the relative order of words carries meaning, but the time between words is irrelevant). In such settings, the discrete index k is a sufficient notion of "time". However, this does not hold for irregularly sampled time series, where the gap $\Delta t_k = t_k - t_{k-1}$ carries information about the system's evolution. Standard RNNs discard it, since they apply the same update regardless of how much time has elapsed.

Early attempts addressed this by feeding Δt_k as an additional input, or by introducing a fixed exponential decay of the hidden state between observations (Che et al., 2018; Rubanova et al., 2019). However, these approaches either leave the evolve map as the identity, or fix its functional form *a priori*, leaving little flexibility to adapt to the data.

2.3.1.3 ODE-RNNs

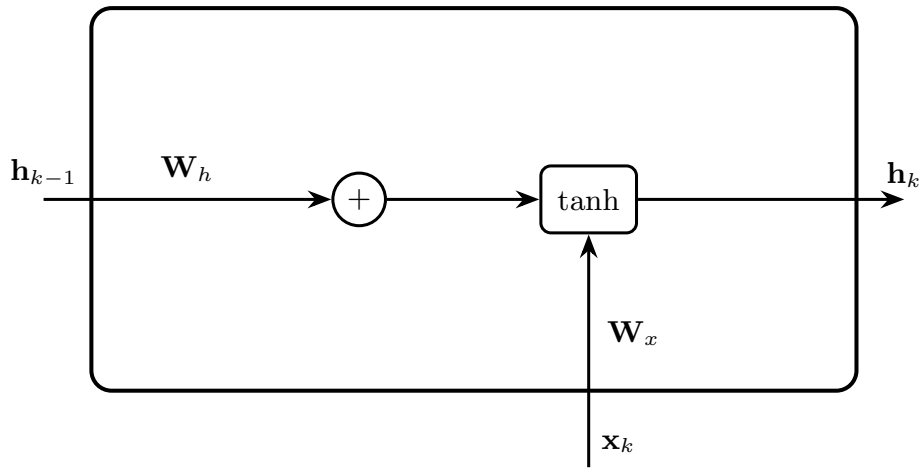
The ODE-RNN (Rubanova et al., 2019) is a natural resolution to the limitations of the early approaches. Rather than fixing the form of Φ_θ , it parameterises it as the flow map of a Neural ODE, *i.e.* $\Phi_\theta(\mathbf{h}(t_k^+), t_k^+, t_k^-) = \mathbf{h}(t_k^+) + \int_{t_k^+}^{t_k^-} f_\theta(\mathbf{h}(\tau)) d\tau$. Numerically, the evolve step becomes:

$$\mathbf{h}(t_k^-) = \text{ODESolve}(f_\theta, \mathbf{h}(t_{k-1}^+), (t_{k-1}^+, t_k^-)) \quad (2.38)$$

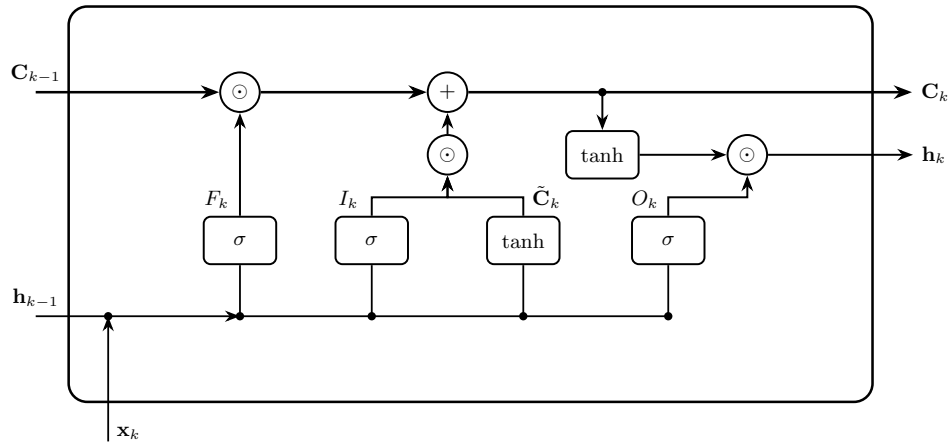
where $f_\theta : \mathbb{R}^h \rightarrow \mathbb{R}^h$ is a feedforward neural network parameterising the hidden state dynamics. The update step then incorporates the new observation via a GRU cell:

$$\mathbf{h}(t_k^+) = \text{GRUCell}(\mathbf{h}(t_k^-), \mathbf{x}_k) \quad (2.39)$$

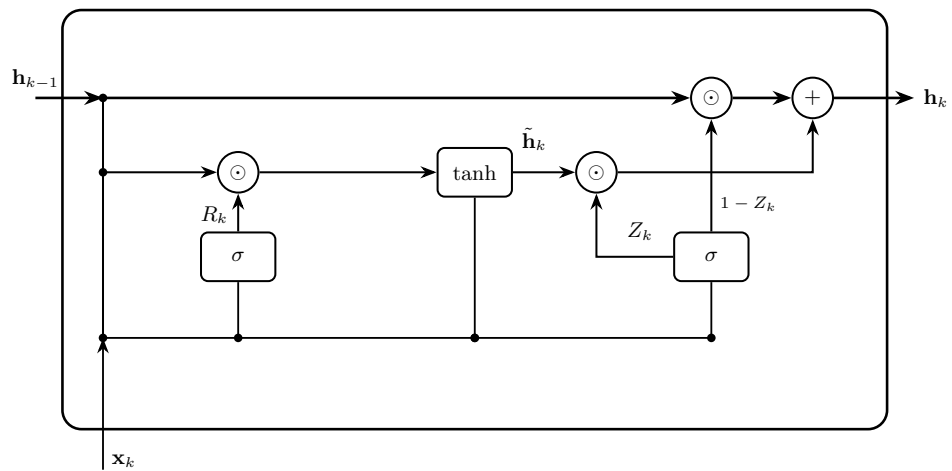
As for traditional RNNs, we then map the hidden state to the observations using an output network g_θ . The full algorithm is given in Algorithm 1.



(A) Vanilla Recurrent Neural Network.



(B) Long Short-Term Memory.



(C) Gated Recurrent unit.

FIGURE 2.18: From top to bottom: vanilla RNN, LSTM and GRU.

Algorithm 1 ODE-RNN (Rubanova et al., 2019)

Require: Observations and timestamps $\{(\mathbf{x}_k, t_k)\}_{k=0}^{N-1}$

- 1: $\mathbf{h}_0 \leftarrow \mathbf{0}$ ▷ Initialisation
- 2: **for** $k = 1, \dots, N - 1$ **do**
- 3: $\mathbf{h}(t_k^-) \leftarrow \text{ODESolve}(f_\theta, \mathbf{h}(t_{k-1}^+), (t_{k-1}^+, t_k^-))$ ▷ Evolve
- 4: $\mathbf{h}(t_k^+) \leftarrow \text{GRUCell}(\mathbf{h}(t_k^-), \mathbf{x}_k)$ ▷ Update
- 5: **end for**
- 6: **return** $\{g_\theta(\mathbf{h}_k)\}_{k=0}^{N-1}$

A residual limitation concerns gradient stability. ODE-RNNs still suffer from vanishing and exploding gradients, regardless of the choice of solver. The ODE-LSTM model (Lechner and Hasani, 2020) addresses this by replacing the GRU update cell with an LSTM cell. The full algorithm is given in Algorithm 2.

Algorithm 2 ODE-LSTM (Lechner and Hasani, 2020)

Require: Observations and timestamps $\{(\mathbf{x}_k, t_k)\}_{k=0}^{N-1}$

- 1: $\mathbf{h}_0 \leftarrow \mathbf{0}, \quad \mathbf{C}_0 \leftarrow \mathbf{0}$ ▷ Initialisation
- 2: **for** $k = 1, \dots, N - 1$ **do**
- 3: $\mathbf{h}(t_k^-) \leftarrow \text{ODESolve}(f_\theta, \mathbf{h}(t_{k-1}^+), (t_{k-1}^+, t_k^-))$ ▷ Evolve
- 4: $(\mathbf{C}_k, \mathbf{h}(t_k^+)) \leftarrow \text{LSTMCell}(\mathbf{C}_{k-1}, \mathbf{h}(t_k^-), \mathbf{x}_k)$ ▷ Update
- 5: **end for**
- 6: **return** $\{g_\theta(\mathbf{h}_k)\}_{k=0}^{N-1}$

2.3.1.4 Handling Exogenous Inputs

Among others, ODE-RNN and ODE-LSTM belong to the broad family that Gwak et al. (2020) term *Neural Jump Differential Equations* (NJDEs). Their common structure can be written as an impulsive differential equation:

$$\begin{cases} \dot{\mathbf{h}}(t) = f_\theta(\mathbf{h}(t)) & t \neq t_k \\ \mathbf{h}(t_k^+) = \phi_\theta(\mathbf{h}(t_k^-), x_k) & t = t_k \end{cases} \quad (2.40)$$

As noted by Gwak et al. (2020): *"Although it is possible to use NJDEs to naively model systems with interventions, by, say, simply concatenating them with observations, this approach is limited as it does not explicitly separate autonomous dynamics of the system from these effects. This, in turn, makes it challenging to properly reconstruct the evolution of a system subject to external forces."*

In response, the author propose Intervention-Modeling ODE (IMODE), which introduces a second hidden state dedicated to tracking the effect of external interventions, alongside the standard hidden state encoding autonomous dynamics. We do not explore IMODE further as when benchmarked on glucose forecasting by Fitzgerald et al. (2023), it consistently underperforms relative to ODE-LSTM across both simulated and real-world settings. We give more details on their application to glucose forecasting in the following section.

2.3.1.5 Application to Glucose

[Fitzgerald et al. \(2023\)](#) apply several of the interleaving architectures introduced above (ODE-GRU, ODE-LSTM, and IMODE) to probabilistic forecasting of blood glucose in the intensive care unit (ICU), on both a synthetic dataset generated from an instance of Equation 2.19 and a real-world Electronic Medical Records (EMRs) dataset.

The modeling task is next-step forecasting, *i.e.* given an irregularly observed patient history $\mathcal{H}_i$ forecast the distribution of blood glucose at the next available measurement time :

$$p_\theta(y_i(t) | \mathcal{H}_i) = \mathcal{N}(\mu(\mathbf{h}(t)), \sigma^2(\mathbf{h}(t))), \quad t > T_i \quad (2.41)$$

where $\mathbf{h}(t)$ is the hidden state of the CTRNN at time t , and $\mu(\mathbf{h}(t)), \sigma^2(\mathbf{h}(t))$ are produced by a heteroscedastic output head g_θ mapping $\mathbf{h}(t)$ to the parameters of a univariate Gaussian predictive distribution.

The patient history $\mathcal{H}_i = \{y_i, \mathbf{x}_i, \mathbf{x}_{0,i}\}$ is made of previous blood glucose measurements y_i , co-varying clinical inputs $\mathbf{x}_i$ (nutritional and insulin inputs), and static features $\mathbf{x}_{0,i}$ (*e.g.*, diabetic status).

Given that the authors use a heteroscedastic output head, all interleaving architectures are trained with a specific two-term objective, identical to the one of the GRU-ODE-Bayes architecture ([De Brouwer et al., 2019](#)). See Equation (A.8) from Appendix A.3 for more details.

On the EMR dataset, the ODE-LSTM model outperforms all traditional and deep learning architectures, while being closely followed by Catboost ([Prokhorenkova et al., 2018](#)) which is easier and faster to train. The authors also highlight that the ODE-LSTM model was also marginally better when forecasting the blood glucose of patient taking insulin, making such architectures good candidates for counterfactual predictions.

2.3.2 Interpolation Approach

As said earlier, most prominent architectures based on neural differential equations can be thought as *impulsive differential equations* (Gwak et al., 2020).

A conceptually different strategy, introduced by Kidger et al. (2020), is to incorporate observations not as discrete jumps, but as a continuous-time control signal driving the dynamics. This is the *interpolation* approach. Rather than modifying $\mathbf{h}$ at each observation time, one constructs a continuous control path $\mathcal{C} : [t_0, t_N] \rightarrow \mathbb{R}^{v+1}$ from the data¹⁷ and let it drive the hidden state dynamics at all times via a Riemann-Stieltjes integral. The hidden state then has continuous dependence on the observations, not just at the moments they arrive. This is illustrated in Figure 2.19.

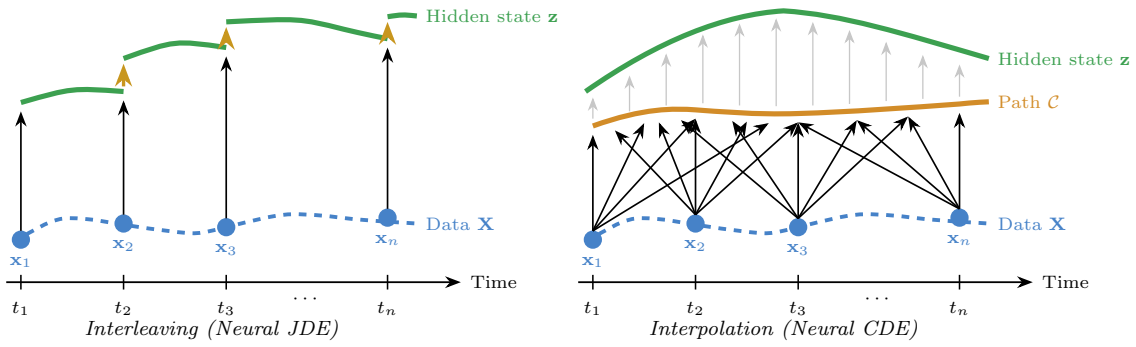


FIGURE 2.19: Comparison of the interleaving and interpolation approaches. **Left:** In the interleaving approach, the hidden state is modified discretely at each observation and evolves freely between them. **Right:** In the interpolation approach, a continuous control path $\mathcal{C}(t)$ is constructed from the data and drives the hidden state continuously at all times. Adapted from Kidger et al. (2020).

2.3.2.1 Controlled Differential Equations

The mathematical foundation of the interpolation approach is the theory of *controlled differential equations* (CDEs), that we introduce here briefly.

Let $\mathcal{C} : [t_0, t_N] \rightarrow \mathbb{R}^a$ be a continuous path of bounded variation, and let $f : \mathbb{R}^b \rightarrow \mathbb{R}^{b \times a}$ be a Lipschitz continuous function. A controlled differential equation driven by $\mathcal{C}$ is defined as:

$$\mathbf{z}(t) = \mathbf{z}_0 + \int_{t_0}^t f(\mathbf{z}(\tau)) d\mathcal{C}(\tau), \quad t \in (t_0, t_N], \quad (2.42)$$

where the integral is a Riemann-Stieltjes integral.

The key insight is that $\mathcal{C}$ acts as a control signal that continuously modulates the dynamics of $\mathbf{z}(t)$. This stands in contrast to a Neural ODE, which is driven only by time itself and has no direct mechanism for incorporating external data after t_0 .

¹⁷The "+1" accounts for the time component, as $\mathcal{C}$ augments the interpolated observations with t itself.

2.3.2.2 Neural Controlled Differential Equations

Given observations $\{(\mathbf{x}_k, t_k)\}_{k=0}^N$, a *Neural CDE* (Kidger et al., 2020) is defined as the solution of:

$$\mathbf{z}(t_0) = \ell_\theta(\mathbf{x}_0, t_0), \quad \mathbf{z}(t) = \mathbf{z}(t_0) + \int_{t_0}^t f_\theta(\mathbf{z}(\tau)) d\mathcal{C}(\tau), \quad t \in (t_0, t_N]. \quad (2.43)$$

where $f_\theta : \mathbb{R}^s \rightarrow \mathbb{R}^{s \times (v+1)}$ is a neural network and $\mathcal{C} : [t_0, t_N] \rightarrow \mathbb{R}^{v+1}$ is a continuous interpolation of the observation-time pairs $(\mathbf{x}_k, t_k)$.

ℓ_θ is, as for the ANODEs, a neural network lifting the initial condition into a higher-dimensional space. Observations can then be reconstructed at arbitrary times using an output network g_θ .

When $\mathcal{C}$ is differentiable and has bounded derivative (as is typically the case in practice) the Riemann-Stieltjes integral reduces to an ordinary integral:

$$\mathbf{z}(t) = \mathbf{z}_0 + \int_{t_0}^t f(\mathbf{z}(\tau)) \frac{d\mathcal{C}}{d\tau}(\tau) d\tau, \quad (2.44)$$

which is precisely an ODE with time-varying vector field $g_{\theta, \mathcal{C}}(\mathbf{z}, \tau) = f_\theta(\mathbf{z}) \dot{\mathcal{C}}(\tau)$. CDEs can therefore be solved and trained using exactly the same software as Neural ODEs (Kidger et al., 2020).

The construction of $\mathcal{C}$ from the discrete observations is a modeling choice with significant practical consequences. Morrill et al. (2021a) point out to the following choices, depending on the modeling and inference requirements :

- **Continuously online or online with missing data:** use rectilinear controls, the only scheme that is fully causal and handles missing observations natively.
- **Discretely online without missing data, or offline:** use cubic Hermite splines with backward differences, which achieve comparable performance at faster speeds.
- **Offline, speed over accuracy:** use natural cubic splines, which are faster but non-causal, making them incompatible with online prediction.

2.3.2.3 NCDEs as the Continuous-Time Limit of RNNs

Neural CDEs have a precise theoretical relationship with RNNs. As shown in Kidger et al. (2020), an RNN is exactly a discretized Neural CDE, making the latter the formal continuous-time limit of the recurrent architecture.¹⁸

For a rigorous demonstration, we refer the reader to the original paper. However, intuitively, an RNN maps a discrete sequence $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ to a sequence of hidden states $(\mathbf{h}_1, \dots, \mathbf{h}_N)$; a Neural CDE maps a continuous path $\mathcal{C}(t)$ to a continuous response $\mathbf{z}(t)$. The former is precisely what you obtain when you discretize the latter.

¹⁸The interleaving approach on the other hand, provides informal continuous-time RNNs. Kidger et al. (2020) further show that their formulation is strictly more expressive than the alternative of the interleaving framework. That is, model such as GRU-ODE-Bayes can be represented exactly by a Neural CDE, but the converse does not hold.

2.3.2.4 Application to glucose

To our knowledge, the only direct application of NCDEs to glucose modeling is [Malpica-Morales et al. \(2025\)](#), who use vagus nerve recordings as the control path $\mathcal{C}$ to drive glucose predictions in anaesthetised rats, demonstrating the suitability of NCDEs for capturing complex physiological driver-response relationships. Their setting however differs fundamentally from ours, targeting a bioelectronic medicine application in an animal model rather than postprandial glucose modeling in humans.

In their experiments, the NCDE significantly outperformed feed-forward, convolutional, and recurrent (GRU) baselines in terms of mean squared error, suggesting that the path-driven formulation better captures the temporal structure of physiological signals than standard sequence-to-sequence architectures.

2.3.3 Deep State-Space Modeling Approach

The interleaving and interpolation approaches reviewed above both process observations sequentially, without ever forming an explicit probabilistic representation of the latent state underlying a trajectory. They produce a *deterministic* latent trajectory $\mathbf{z}(t)$: once the initial state $\mathbf{z}_0$ is fixed, the dynamics are fully determined. Uncertainty can only enter through the output network g_θ , which may parametrize a predictive distribution $p(\mathbf{x}_k | \mathbf{z}(t_k))$, but the latent state itself carries no uncertainty.

The *deep state-space modeling* (Deep SSM) approach on the other hand takes a fundamentally different perspective. Following the Bayesian SSM structure of Section 2.1.2.2, it not only offers a probabilistic observation model $p(\mathbf{x}_k | \mathbf{z}(t_k))$, but also performs probabilistic inference over the initial latent state, approximating the posterior $p(\mathbf{z}_0 | \mathbf{X})$ from the observed snapshots. This makes the entire latent path a random object, enabling uncertainty quantification at the level of the dynamics rather than solely at the observation level.

The key distinction from classical SSMs is that all components are parametrized by neural networks and trained end-to-end via a variational learning pipeline ([Lin and Michailidis, 2024](#)).

The variational autoencoder (VAE) is the foundational building block of the Deep SSM architecture, and is introduced in Section 2.3.3.1. Before arriving at the full Latent ODE architecture, we discuss in Section 2.3.3.2 the challenge of learning structured, interpretable latent representations. Section 2.3.3.3 then derives the Latent ODE model by combining the VAE and NDE frameworks. Finally, Section 2.3.3.4 introduces the Latent ODE-RNN of [Rubanova et al. \(2019\)](#), which is particularly suited to irregularly-sampled data.

2.3.3.1 Variational Autoencoders

Many real-world processes are best understood as driven by a small number of latent factors that are never directly observed but govern the observations. Uncovering these factors from observations alone, what [Kingma and Welling \(2019\)](#) call *unsupervised representation learning*, is one of the central challenges of modern machine learning. Variational Autoencoders ([Kingma et al., 2014](#)) provide a principled framework for learning such latent representations and have been extensively used for that task in recent years.

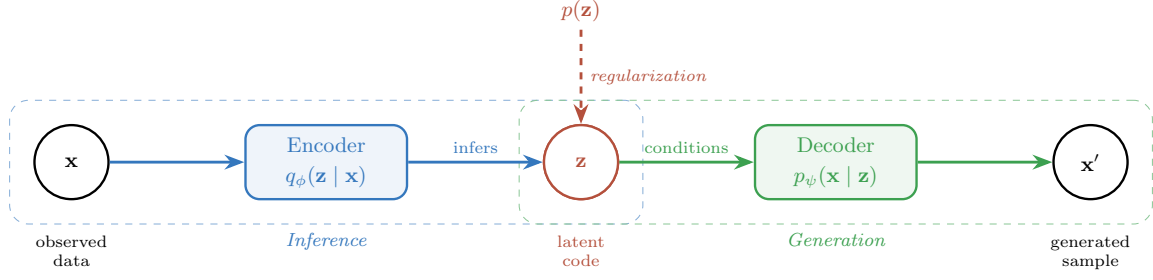


FIGURE 2.20: Schematic representation of the VAE framework.

A VAE consists of two coupled parameterized networks: (i) a *probabilistic encoder*, also called the *inference model* or *recognition model*, $q_\phi(\mathbf{z} | \mathbf{x})$, which recognizes the most plausible latent codes by inferring a distribution over $\mathbf{z}$ given the observation $\mathbf{x}$; and (ii) a *probabilistic decoder*, also called the *observation model*, $p_\psi(\mathbf{x} | \mathbf{z})$, which generates observations by decoding a sampled latent code. The prior $p(\mathbf{z})$ regularizes the latent space by constraining the distribution of codes produced by the encoder. These components are illustrated in Figure 2.20.

More formally, we can frame the problem as a generative model :

$$\mathbf{z} \sim p(\mathbf{z}) \quad (2.45)$$

$$\mathbf{x} | \mathbf{z} \sim p_\psi(\mathbf{x} | \mathbf{z}) \quad (2.46)$$

where $\mathbf{x} \in \mathbb{R}^o$ is an observed data point and $\mathbf{z} \in \mathbb{R}^s$ a corresponding latent variable.

For our generative model, the central object of interest is the *posterior* distribution over the latent variable given the observation, which by Bayes' theorem is:

$$p_\psi(\mathbf{z} | \mathbf{x}) = \frac{p_\psi(\mathbf{x} | \mathbf{z}) p(\mathbf{z})}{p_\psi(\mathbf{x})} \quad (2.47)$$

The denominator $p_\psi(\mathbf{x}) = \int p_\psi(\mathbf{x} | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$, known as the *evidence*, requires integrating over all possible latent configurations. This expression is often intractable.

VAE resolves this issue by approximating the intractable posterior with a tractable variational family $q_\phi(\mathbf{z} | \mathbf{x})$.

Training objective. We want to jointly optimize the generative parameters ψ and the inference parameters ϕ such that: (i) $q_\phi(\mathbf{z} | \mathbf{x})$ is as close as possible to the true posterior $p_\psi(\mathbf{z} | \mathbf{x})$; and (ii) the decoder $p_\psi(\mathbf{x} | \mathbf{z})$ produces realistic observations.

A natural way to quantify (i) is the KL divergence $D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\psi(\mathbf{z} | \mathbf{x})) \geq 0$, which equals zero if and only if the two distributions coincide.

However, $p_\psi(\mathbf{z} | \mathbf{x})$ is intractable, so this KL term cannot be optimized directly. Instead, we derive a tractable lower bound by manipulating the log-evidence, which will be shown to address objectives (i) and (ii) simultaneously:

$$\begin{aligned}
\log p_\psi(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\psi(\mathbf{x})] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\psi(\mathbf{x} | \mathbf{z}) p(\mathbf{z})}{p_\psi(\mathbf{z} | \mathbf{x})} \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\psi(\mathbf{x} | \mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \cdot \frac{q_\phi(\mathbf{z} | \mathbf{x})}{p_\psi(\mathbf{z} | \mathbf{x})} \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\psi(\mathbf{x} | \mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{q_\phi(\mathbf{z} | \mathbf{x})}{p(\mathbf{z})} \right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{q_\phi(\mathbf{z} | \mathbf{x})}{p_\psi(\mathbf{z} | \mathbf{x})} \right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\psi(\mathbf{x} | \mathbf{z})] - D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z}))}_{\mathcal{L}(\psi, \phi; \mathbf{x}) = \text{ELBO}} + D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\psi(\mathbf{z} | \mathbf{x}))
\end{aligned}$$

Since $D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\psi(\mathbf{z} | \mathbf{x})) \geq 0$, the quantity $\mathcal{L}(\psi, \phi; \mathbf{x})$ is a lower bound on the log-evidence. Hence the name *Evidence Lower Bound*, or ELBO in short. Maximizing the ELBO thus simultaneously tightens the variational approximation to the true posterior and maximizes the marginal likelihood (log-evidence), as displayed in Figure 2.21.

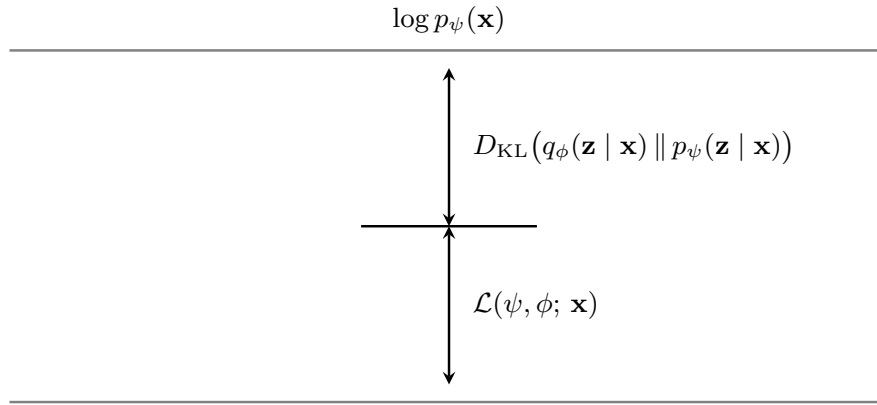


FIGURE 2.21: Schematic representation of the ELBO.

The ELBO itself further decomposes into two interpretable terms:

$$\mathcal{L}(\psi, \phi; \mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\psi(\mathbf{x} | \mathbf{z})]}_{\text{reconstruction}} - \underbrace{D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z}))}_{\text{KL regularization}} \quad (2.48)$$

The reconstruction term encourages the decoder to faithfully recover $\mathbf{x}$ from the latent code; the KL term regularizes the posterior toward the prior, preventing the encoder from deviating too far from the prior, thereby ensuring the latent space remains structured and well-behaved.

A common pitfall of VAEs is the so-called *posterior collapse*. The latter happens when the KL regularization term dominates the objective, making the posterior collapse toward the prior. A common remedy is to introduce a scalar $\beta_{\text{KL}} > 0$ to control the relative weight of the regularization term, yielding the β -VAE objective (Higgins et al., 2017):

$$\mathcal{L}_\beta(\psi, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\psi(\mathbf{x} | \mathbf{z})] - \beta_{\text{KL}} D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z})) \quad (2.49)$$

Setting $\beta_{\text{KL}} < 1$ reduces the regularization pressure, allowing the encoder to retain more information about the input in the latent code (Fu et al., 2019). Conversely, Higgins et al. (2017) show that setting $\beta_{\text{KL}} > 1$ encourages *disentanglement* of the latent dimensions by increasing the regularization pressure, at the cost of reconstruction quality (a discussion we return to in Section 2.3.3.2). In practice, a standard alternative to a fixed β_{KL} is *KL cost annealing*, starting with $\beta_{\text{KL}} = 0$ and gradually increasing over training until reaching 1, giving the encoder time to learn how to reconstruct data before the regularization kicks in (Bowman et al., 2016).

Gaussianity assumptions. In the standard VAE formulation, one often assumes that the distributions involved are Gaussian.

The conventional choice for the prior is the isotropic gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. That prior is then commonly associated to a posterior which is a diagonal Gaussian $q_\phi(\mathbf{z} | \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}_\phi^2(\mathbf{x})))$, whose parameters $(\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi)$ are the outputs of the encoder network.

In such a case the KL term in (2.48) admits a closed solution:

$$D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}_\phi^2(\mathbf{x}))) \| \mathcal{N}(\mathbf{0}, \mathbf{I})) = \frac{1}{2} \sum_{j=1}^s (\mu_{\phi,j}(\mathbf{x})^2 + \sigma_{\phi,j}^2(\mathbf{x}) - 1 - \log \sigma_{\phi,j}^2(\mathbf{x})) \quad (2.50)$$

This choice of prior encourages independence between latent dimensions.

Then, the observation model is in general a Gaussian:

$$p_\psi(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\psi(\mathbf{z}), \boldsymbol{\Sigma}_\psi(\mathbf{z})) \quad (2.51)$$

where $\boldsymbol{\mu}_\psi(\mathbf{z}) \in \mathbb{R}^o$ and $\boldsymbol{\Sigma}_\psi(\mathbf{z}) \in \mathbb{R}^{o \times o}$ are outputs of the decoder network. A diagonal structure $\boldsymbol{\Sigma}_\psi(\mathbf{z}) = \text{diag}(\boldsymbol{\sigma}_\psi^2(\mathbf{z}))$ is commonly assumed to reduce the number of parameters. In the diagonal case, the reconstruction term in (2.48) is:

$$\mathbb{E}_{q_\phi} [\log p_\psi(\mathbf{x} | \mathbf{z})] = -\frac{p}{2} \log(2\pi) - \frac{1}{2} \sum_{k=1}^o \left(\log \sigma_{\psi,k}^2(\mathbf{z}) + \frac{(x_k - \mu_{\psi,k}(\mathbf{z}))^2}{\sigma_{\psi,k}^2(\mathbf{z})} \right) \quad (2.52)$$

In practice, many implementations simplify by fixing $\boldsymbol{\Sigma}_\psi \equiv \sigma_{\text{obs}}^2 \mathbf{I}$ as a constant hyperparameter, in which case (2.52) depends on a scaled mean squared error between $\mathbf{x}$ and the decoder output $\boldsymbol{\mu}_\psi(\mathbf{z})$:

$$-\frac{1}{2\sigma_{\text{obs}}^2} \sum_{k=1}^o (x_k - \mu_{\psi,k}(\mathbf{z}))^2 \quad (2.53)$$

Each of these three choices can be revisited. Alternative priors include the Laplacian, which promotes sparsity and also admit a closed form solution for the KL term in (2.48) (Conti et al., 2026), and normalizing flow priors (Rezende and Mohamed, 2015), which learn a flexible $p_\lambda(\mathbf{z})$ at the cost of additional parameters λ .

2.3.3.2 The Representation Learning Challenge

The latent space $\mathbf{z}$ is the central object of the VAE (and deep SSM) framework. Shaped jointly by the prior $p(\mathbf{z})$ and the training objective, it is meant to encode the latent factors

that causally generate the observations. In the best case, each coordinate of $\mathbf{z}$ captures an independent, semantically meaningful degree of freedom of the underlying true data generating process.

Paraphrasing [Kingma and Welling \(2019\)](#) : *unsupervised representation learning* is the quest for disentangled, statistically independent, and causally interpretable factors of variation.

In practice, of course, this task is non-trivial. [Locatello et al. \(2019\)](#) establish that without inductive biases on both the model and the data, infinitely many generative models share the same marginal distribution of observations, making unsupervised identification of the true factors impossible in general. A well-functioning generative model and a well-structured latent space are therefore two distinct objectives.

The β -VAE loss 2.49 explicitly trades one against the other. By setting $\beta_{\text{KL}} > 1$, it sacrifices some reconstruction fidelity to encourage disentanglement.

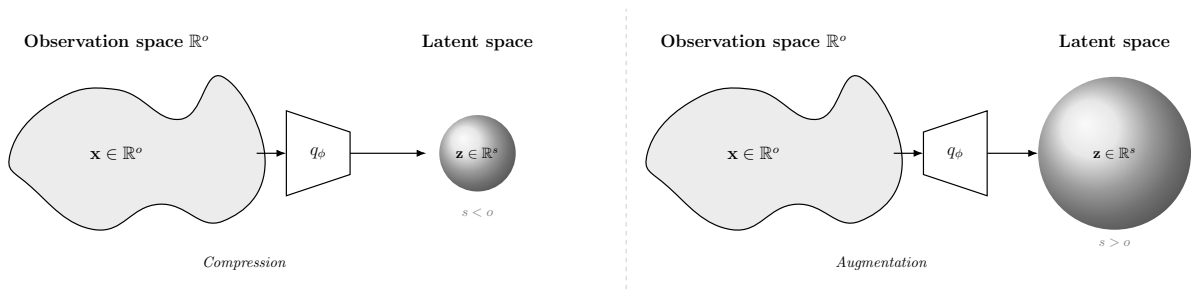


FIGURE 2.22: Two regimes of latent representation learning. **Left:** compression ($s < o$) **Right:** augmentation ($s > o$) where the encoder lifts a low-dimensional observation into a richer latent space where the dynamics are easier to represent.

We want to highlight that, even though latent spaces are usually chosen as lower-dimensional representations, one can equivalently choose higher-dimensional ones. Figure 2.22 illustrates the two philosophies. The first seeks *compression*: the encoder distills a high-dimensional observation into a smaller number of explanatory factors, $s < o$. The second seeks *augmentation*: the encoder lifts a low-dimensional observation into a richer embedding where the data is easier to represent, $s > o$. The latter approach directly mirrors the augmentation we performed in the NDE paradigm.

In both cases, the underlying rationale is the same: finding a new coordinate system in which the structure of the data is more transparent.

2.3.3.3 From VAE to Latent ODE

The Deep SSM family is the last of the three families to introduce, and it is the modeling paradigm adopted (and adapted) in the present work. It was first proposed in the seminal work of [Chen et al. \(2018\)](#) under the name *Latent ODE*. We follow [Lin and Michailidis \(2024\)](#) in terming this family *Deep SSM*, as it is a direct neural analogue of the classical SSM framework of Section 2.1.2.1.

It extends the VAE framework of Section 2.3.3.1 to continuous-time dynamical systems by replacing the static latent variable with a latent trajectory governed by a Neural ODE.

Rather than mapping a single observation $\mathbf{x}$ to a latent code $\mathbf{z}$, the model maps an entire observed trajectory¹⁹ $\mathbf{X} = [x_{t_0}, \dots, x_{t_{N-1}}]$ to the parameters of a distribution over an initial latent condition $\mathbf{z}_0$, by encoding $\mathbf{X}$ backward in time using a RNN and applying to the output hidden vector a translation network τ :

$$\mathbf{h}_0 = \overleftarrow{\text{RNN}}_\phi(\mathbf{X}), \quad [\boldsymbol{\mu}(\mathbf{X}), \boldsymbol{\sigma}(\mathbf{X})] = \tau_\phi(\mathbf{h}_0) \quad (2.54)$$

where $\overleftarrow{\cdot}$ specifies that the RNN is run backward in time, and the inference parameters ϕ contains both the RNN and the translation network²⁰ weights.

Together, these two networks are the analogous of the probabilistic encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$ in the classical VAE. The distinction is that we now parametrize a distribution over the initial condition $\mathbf{z}_0$ given the observed trajectory $\mathbf{X}$:

$$q_\phi(\mathbf{z}_0 \mid \mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{X}), \text{diag}(\boldsymbol{\sigma}^2(\mathbf{X}))) \quad (2.55)$$

We then postulate that the latent dynamics evolve according to an ODE, parametrized by a vector field f_θ :

$$\frac{d\mathbf{z}}{dt}(t) = f_\theta(\mathbf{z}(t)) \quad (2.56)$$

In practice, when an observed trajectory $\mathbf{X}$ is available, the initial condition $\mathbf{z}_0$ is sampled from the variational posterior $q_\phi(\mathbf{z}_0 \mid \mathbf{X})$, which approximates the intractable true posterior $p(\mathbf{z}_0 \mid \mathbf{X})$. One can also generate plausible trajectories by sampling directly from the prior.

The latent trajectory is approximated at desired time points $\{t_k\}_{k=0}^{N-1}$ using numerical integration:

$$\{\mathbf{z}(t_k)\}_{k=0}^{N-1} = \text{ODESolve}(f_\theta, \mathbf{z}_0, \{t_k\}_{k=0}^{N-1}) \quad (2.57)$$

A shallow MLP then constitutes the decoder, mapping each latent state $\mathbf{z}(t_k)$ to the mean of the observation model:

$$p_\psi(x_k \mid \mathbf{z}(t_k)) = \mathcal{N}(\mu_\psi(\mathbf{z}(t_k)), \sigma_{\text{obs}}^2) \quad (2.58)$$

where σ_{obs} is typically fixed as a hyperparameter in the original formulation ([Chen et al., 2018](#)), following the same rationale than Section 2.3.3.1.

Altogether, our generative model can be seen as an instance the mechanistic SSM (2.8)-(2.9) :

$$\mathbf{z}_0 \sim p(\mathbf{z}_0) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (2.59)$$

$$\frac{d\mathbf{z}}{dt}(t) = f_\theta(\mathbf{z}(t)) \quad (2.60)$$

$$x_k \sim p_\psi(x_k \mid \mathbf{z}(t_k)) \quad (2.61)$$

¹⁹In what follows, without loss of generality, we assume that the observed time serie is univariate (as for glucose).

²⁰The translation network τ_ϕ is simply a shallow MLP "translating" the RNN's output hidden vector to the parameters of the variational posterior.

The full architecture is illustrated on Figure 2.23.

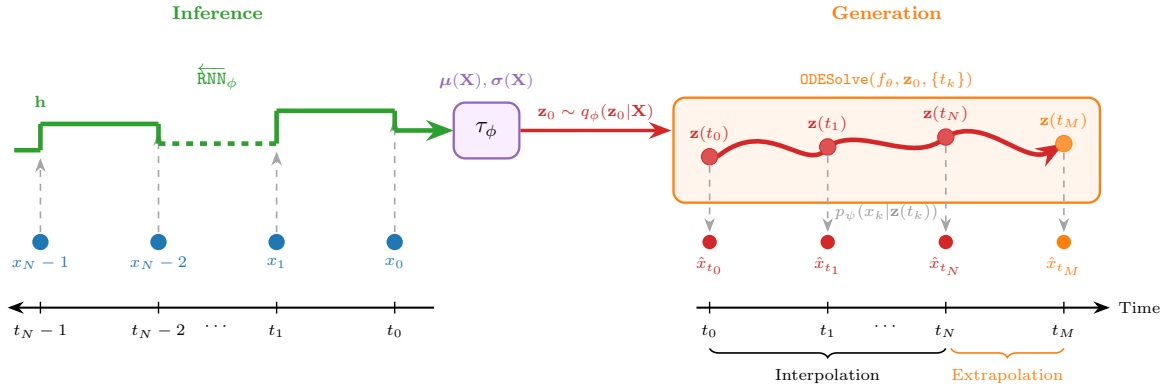


FIGURE 2.23: Schematic of the Latent ODE architecture. First the univariate time series $\mathbf{X}$ is encoded backward in time into a hidden state $\mathbf{h}_0$, which is then mapped by the translation network τ_ϕ to the parameters $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ of the variational posterior. Then, an initial condition $\mathbf{z}_0$ is sampled from this posterior and evolved forward under the learned dynamics f_θ . Finally, observations are emitted from the latent trajectory thanks to an output network, which is typically parametrizing the mean of a predictive Gaussian distribution $p_\psi(x_k | \mathbf{z}(t_k)) = \mathcal{N}(\mu_\psi(\mathbf{z}(t_k)), \sigma_{\text{obs}}^2)$.

Training objective Similarly to the VAE setting, the training objective is given by the ELBO :

$$\mathcal{L}(\theta, \psi, \phi; \mathbf{X}) = \mathbb{E}_{q_\phi} \left[\sum_{k=0}^{N-1} \log p_\psi(x_k | \mathbf{z}(t_k)) \right] - D_{\text{KL}}(q_\phi(\mathbf{z}_0 | \mathbf{X}) \| p(\mathbf{z}_0)) \quad (2.62)$$

Properties The Latent ODE enjoys several properties that make it particularly attractive for dynamical system modeling. It is *generative*: sampling $\mathbf{z}_0 \sim p(\mathbf{z}_0)$ and integrating forward enables *in silico* simulation of new trajectories. It provides *uncertainty quantification* over the initial condition: the posterior $q_\phi(\mathbf{z}_0 | \mathbf{X})$ propagates observation uncertainty into a distribution over trajectories, rather than a single point forecast. Finally, since the ODE solver can be queried at arbitrary times, it can be used both for interpolation and extrapolation.

Note that, regarding the second property, though it is already made explicit through the notations, it is worth emphasising that, as for a VAE, each observation $\mathbf{x}$ (or here entire trajectory $\mathbf{X}$) is mapped to its own parameters. This is a very interesting property as it theoretically allows the model to grasp a certain idea of inter-individual variability, projecting similar observations in nearby regions of the latent space, and dissimilar ones further apart.

Dimensionality of the latent space. The latent state typically lives in a higher-dimensional space than the observations. This augmentation serves a precise mathematical purpose. As established in Section 2.2.2.5, NODEs are not universal approximators in their native space. Lifting to a higher-dimensional space resolves this topological obstruction and restores the universal approximation property.

Beyond the approximation-theoretic argument, seeking a coordinate system in which dynamics simplify is a recurring strategy in physics and data-driven system identification. As for VAEs, two complementary philosophies coexist [Brunton et al. \(2021\)](#): one seeks a *low-dimensional* embedding that extracts a few key latent variables to parameterise the dynamics (an approach taken by the SINDy Autoencoder, [Champion et al. \(2019\)](#)); the other lifts the data to a *higher-dimensional* space where the evolution becomes approximately linear, exploiting what [Brunton et al.](#) describe as related to Koopman Operator Theory and Cover’s theorem. The Latent ODE, with its higher-dimensional latent space, is closer to the second paradigm, though the connection to Koopman theory remains informal and the encoder q_ϕ is not explicitly designed to learn a Koopman embedding.

2.3.3.4 Latent ODE-RNN: Encoding Irregular Time Series

The Latent ODE of [Chen et al. \(2018\)](#) uses a standard RNN as part of the recognition network, which treats all time steps as equidistant and is therefore poorly suited to irregularly-sampled data. [Rubanova et al. \(2019\)](#) resolve this by replacing the encoder with an ODE-RNN, yielding a fully ODE-based architecture. The posterior over $\mathbf{z}_0$ becomes:

$$q_\phi(\mathbf{z}_0 \mid \mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{X}), \text{diag}(\boldsymbol{\sigma}^2(\mathbf{X}))), \quad [\boldsymbol{\mu}(\mathbf{X}), \boldsymbol{\sigma}(\mathbf{X})] = \tau_\phi(\overleftarrow{\text{ODE-RNN}}_\phi(\mathbf{X})) \quad (2.63)$$

The improvement over a vanilla RNN encoder is most visible at extrapolation. A Latent ODE-RNN encoder maintains periodic dynamics far beyond the training interval, while the RNN-based version tend to fail doing so ([Rubanova et al., 2019](#)).

[Rubanova et al. \(2019\)](#) also formalize two inference regimes for latent variable models, illustrated in Algorithms 3 and 4. The key difference lies in the direction and scope of the encoding step.

Algorithm 3 Interpolation Regime

Require: $\mathbf{X} = \{(x_k, t_k)\}_{k=0}^{N-1}$
 1: $\mathbf{h}_0 \leftarrow \overleftarrow{\text{ODE-RNN}}_\phi(\mathbf{X})$ ▷ Encode backward in time
 2: $\boldsymbol{\mu}(\mathbf{X}), \boldsymbol{\sigma}(\mathbf{X}) \leftarrow \tau_\phi(\mathbf{h}_0)$
 3: $\mathbf{z}_0 \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{X}), \text{diag}(\boldsymbol{\sigma}(\mathbf{X})^2))$ ▷ Sample initial latent condition
 4: $\{\mathbf{z}(t_k)\}_{k=0}^{N-1} \leftarrow \text{ODESolve}(f_\theta, \mathbf{z}_0, \{t_k\}_{k=0}^{N-1})$ ▷ Integrate $t_0 \rightarrow t_{N-1}$ (interpolate)
 5: **return** $\{p_\psi(x_k \mid \mathbf{z}(t_k))\}_{k=0}^{N-1}$

Algorithm 4 Extrapolation Regime

Require: $\mathbf{X} = \{(x_k, t_k)\}_{k=0}^{N-1}$, query times $\{t_k\}_{k=N}^M$
 1: $\mathbf{h}_{N-1} \leftarrow \overleftarrow{\text{ODE-RNN}}_\phi(\mathbf{X})$ ▷ Encode forward in time
 2: $\boldsymbol{\mu}(\mathbf{X}), \boldsymbol{\sigma}(\mathbf{X}) \leftarrow \tau_\phi(\mathbf{h}_{N-1})$
 3: $\mathbf{z}_{N-1} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{X}), \text{diag}(\boldsymbol{\sigma}(\mathbf{X})^2))$ ▷ Sample latent condition at t_{N-1}
 4: $\{\mathbf{z}(t_k)\}_{k=N}^M \leftarrow \text{ODESolve}(f_\theta, \mathbf{z}_{N-1}, \{t_k\}_{k=N}^M)$ ▷ Integrate $t_{N-1} \rightarrow t_M$ (extrapolate)
 5: **return** $\{p_\psi(x_k \mid \mathbf{z}(t_k))\}_{k=N}^M$

Properties It is also worth contrasting the Latent ODE-RNN with autoregressive models such as the ODE-RNN. In autoregressive models, the joint distribution factorises as $p(\mathbf{X}) = \prod_k p(x_k | x_{k-1}, \dots, x_0)$, and both the dynamics and the conditioning on observations are entangled in the hidden state updates, making them hard to interpret. In contrast, the Latent ODE-RNN explicitly decouples the dynamics (f_θ), the observation model (p_ψ), and the recognition model (q_ϕ), each of which can be examined or specified independently. Furthermore, while autoregressive models are often sufficient for densely sampled data, they perform worse when observations are sparse (Rubanova et al., 2019), a known challenge for CGM data.

A particularly promising property of the Latent ODE-RNN is that Rubanova et al. (2019) demonstrated empirically is that the latent space can recover physically meaningful structure. The authors showcase this on the *MuJoCo Hopper* simulation, where the learned $\mathbf{z}_0$ aligns closely with the physical parameters that most strongly determine the future trajectory.

2.3.3.5 Application to glucose

To date, there is to our knowledge no applications of the Deep SSM family to glucose modeling, which motivates the present work.

Nevertheless, Lu et al. (2021) propose a related application in pharmacokinetic (PK) modeling, predicting drug plasma concentration of T-DM1 conjugate for two dosing regimens (weekly and once every three weeks) after the first dosing cycle.

In particular, they use a slightly modified version of the Latent ODE of Chen et al. (2018), which is illustrated on Figure 2.24.

First, though they also use RNN encoder to parametrize the distribution of the latent initial conditions, and a NODE to parametrize the latent dynamics, they use a simple root mean square error as training objective, meaning that their model is not a valid variational model per se.

Second, their latent dynamics are not integrated globally over the full time horizon, but rather solved sequentially between consecutive time points, with the drug dosing amount injected as an additive perturbation to the latent state $\mathbf{z}$ at each dosing event:

$$\mathbf{z}_{t_k} = \text{ODESolve}(f_\theta, \mathbf{z}_{t_{k-1}} + \text{AMT}_{t_{k-1}}, [t_{k-1}, t_k]) \quad (2.64)$$

where the scalar dosing amount $\text{AMT}_{t_{k-1}}$ is embedded into the first dimension of the latent space $[\text{AMT}_t, 0, \dots, 0]^\top \in \mathbb{R}^s$.

Third, at the emission stage, their decoder does not map the latent state $\mathbf{z}_{t_i}$ alone to the predicted concentration. It concatenates it with the observed PK measurements from the first dosing cycle, used as a patient-specific context signal:

$$\hat{y}_{t_i} = g_\psi(\mathbf{z}_{t_i}, \mathbf{c}_{\text{cycle}_1}) \quad (2.65)$$

Empirically, their model achieves strong results, outperforming LSTM, lightGBM (Ke et al., 2017), and classical NLME models both within and *across* dosing regimens. However, several design decisions they made remain debatable: (i) the absence of a variational

objective forgoes the theoretical framework; (ii) the piecewise integration prevents the ODE from modeling smooth dynamics between perturbation events; (iii) and embedding the dosing amount in a single latent dimension via zero-padding arbitrarily concentrates the perturbation’s effect, whereas a learned control signal could distribute it across the full latent state in a more physiologically grounded manner.

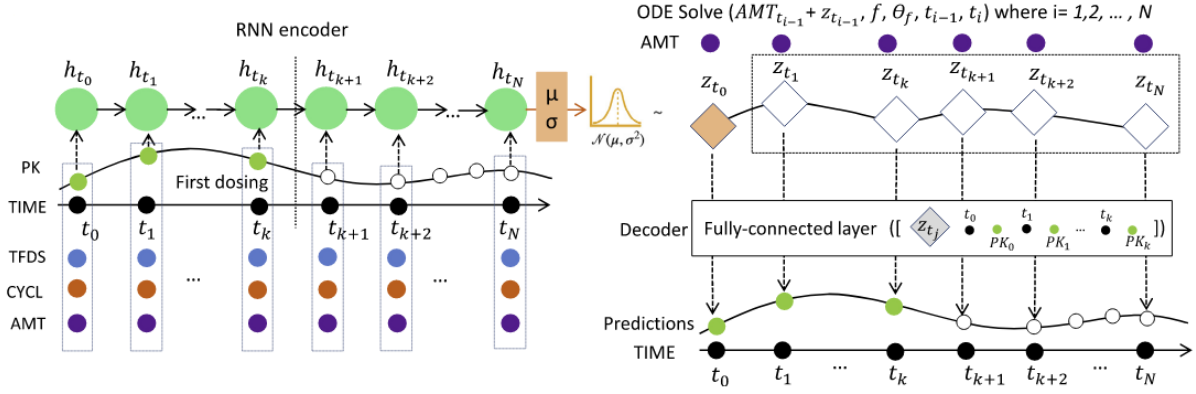


FIGURE 2.24: Schematic of the Neural PK architecture, from [Lu et al. \(2021\)](#)

2.3.4 Towards a Controlled Latent ODE-RNN for Glucose Modeling

Among the three paradigms reviewed in this section, the Deep SSM approach appears the most suited to our objectives. Its variational framework provides uncertainty quantification over the latent trajectory and the observation process ; its continuous-time formulation handles irregular observations naturally ; and its separation between higher dimensional latent dynamics and observation model aligns with the physiological structure of postprandial glucose responses.

The remaining open question concerns the incorporation of exogenous inputs. Most approaches either ignore this distinction entirely, or handle meal inputs through *ad hoc* state augmentation, as illustrated by the PK application of [Lu et al. \(2021\)](#). A more theoretically grounded approach would incorporate control signals directly into the latent dynamics.

Doing so would also enable for counterfactual simulations, by substituting $\mathbf{u}$ with an prospective control $\mathbf{u}'$, and would also allow the model to learn a more faithful latent representation, by disentangling the intrinsic system dynamics from the effect of exogenous perturbations.

In the next chapter, we introduce CLODE, a Controlled Latent ODE-RNN, which extends the Deep SSM framework with a meal control signal, a full variational training pipeline, targeting the three objectives outlined in Chapter 1: (i) accurate trajectory prediction conditioned on meal inputs, (ii) structured latent representation of individual glycemic response, (iii) and counterfactual dietary simulation.

3.1 Proposed Model

3.1.1 Problem Formulation

We consider a population of n individuals monitored in free-living conditions. For each individual i , we observe over a 48-hour window:

- (i) a sequence of CGM measurements $\mathbf{X}^{(i)} = \{(x_k^{(i)}, t_k^{(i)})\}_0^{M_i}$, where $x_k^{(i)} \in \mathbb{R}_+$ denotes the glucose reading at time $t_k^{(i)}$, and M_i varies across individuals due to sensor dropout;
- (ii) a dietary log $\mathbf{C}^{(i)} = \{(c_j^{(i)}, t_j^{(i)})\}_{j=0}^{C_i-1}$, where $c_j^{(i)} \in \mathbb{R}_+$ is the carbohydrate amount of the j -th meal recorded at time $t_j^{(i)}$.

The 48-hour window is partitioned into a *context window* $\mathbf{X}_{\leq 24h}^{(i)} = \{(x_k^{(i)}, t_k^{(i)})\}_{k=0}^{N_i-1}$ and an *extrapolation window* $\mathbf{X}_{>24h}^{(i)} = \{(x_k^{(i)}, t_k^{(i)})\}_{k=N_i}^{M_i}$, with N_i and M_i varying across individuals due to sensor dropout. Similarly, $\mathbf{C}_{>24h}^{(i)}$ denotes the prospective meal plan over the extrapolation window.

Only the context window is used at prediction time. The extrapolation window is used for training and evaluation.

Once the model is calibrated, given $\mathbf{X}_{\leq 24h}^{(i)}$ and a prospective meal plan $\mathbf{C}_{>24h}^{(i)}$, we seek to predict the glucose trajectory over the subsequent 24 hours with a measure of predictive uncertainty:

$$p\left(G^{(i)}(t) \mid \mathbf{X}_{\leq 24h}^{(i)}, \mathbf{C}_{>24h}^{(i)}\right), \quad t \in (24h, 48h] \quad (3.1)$$

This modelling task presents three major challenges, already highlighted in the introduction. First, glycemia is a continuous-time process but only noisy snapshots are available at irregular time points. Second, glucose trajectories are subject to exogenous perturbations, primarily meal intake, which are typically unrealistically logged as point events. Third, glucose dynamics differ substantially across individuals, even for individuals with an identical diet. These three constraints jointly motivate the architecture described in the following subsection.

3.1.2 Controlled Latent ODE-RNN

The three constraints identified above naturally point out to a model in the Deep SSM paradigm, with a latent process governing the dynamics, and a probabilistic observation model linking that process to noisy glucose measurements.

Specifically, the Latent ODE-RNN introduced in Section 2.3.3 theoretically satisfies two out of the three constraints natively. Indeed, the first constraint is fulfilled, as the model handles irregular time series by design, and the third was demonstrated empirically by Rubanova et al. (2019) as highlighted in Section 2.3.3.4.

However, the vanilla Latent ODE-RNN provides no mechanism for incorporating exogenous inputs into the latent dynamics, leaving the second constraint unaddressed.

Given that limitation, we introduce a new model in the Deep SSM paradigm that is inspired by the Latent ODE-RNN, but treats differently manifest variables that are genuinely governed by a differential equation (such as glucose) from those that are exogenous to the system dynamics (such as carbohydrate intake).

We refer to that new model as *Controlled Latent ODE-RNN* (CLODE), whose architecture is displayed in Figure 3.1. The model requires that we transform the prospective meal log into a continuous exogenous signal $u^{(i)} : [t_N^{(i)}, t_M^{(i)}] \rightarrow \mathbb{R}_+$ ^{1 2}. Glucose measurements are then modelled as the observable component of a latent physiological process $\mathbf{z}^{(i)}(t) \in \mathbb{R}^s$, with control $u^{(i)}(t) \in \mathbb{R}_+$:

$$\frac{d\mathbf{z}^{(i)}}{dt}(t) = f_\theta(\mathbf{z}^{(i)}(t), u^{(i)}(t)) \quad (3.2)$$

$$x_k^{(i)} \sim p_\psi(x_k^{(i)} | \mathbf{z}^{(i)}(t_k^{(i)})) \quad (3.3)$$

where f_θ parameterises the latent dynamics and p_ψ is a probabilistic observation model. The individual-specific initial condition $\mathbf{z}_{N_i-1}^{(i)}$ is treated as a latent random variable inferred from the first 24 hours of CGM observations. It is produced by a recognition network, which together with the controlled NODE and the emission network forms the three learned components of CLODE. The full pipeline is summarized in Algorithm 5.

In the remainder of this chapter, we drop the superscript (i) and describe the model for a generic individual with N first 24-hour glucose measurements $\mathbf{X}_{\leq 24h} = \{(x_k, t_k)\}_{k=0}^{N-1}$.

Algorithm 5 CLODE

Require: Observed glucose $\mathbf{X}_{\leq 24h} = \{(x_k, t_k)\}_{k=0}^{N-1}$, meal signal $u(t)$, query times $\{t_k\}_{k=N}^M$

- 1: $\boldsymbol{\mu}(\mathbf{X}_{\leq 24h}), \boldsymbol{\sigma}(\mathbf{X}_{\leq 24h}) \leftarrow \tau_\phi(\overline{\text{ODE-RNN}}_\phi(\mathbf{X}_{\leq 24h}))$ ▷ Encode forward in time
- 2: $\mathbf{z}_{N-1} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{X}_{\leq 24h}), \text{diag}(\boldsymbol{\sigma}(\mathbf{X}_{\leq 24h})^2))$ ▷ Sample initial condition at t_{N-1}
- 3: $\{\mathbf{z}(t_k)\}_{k=N}^M \leftarrow \text{ODESolve}(f_\theta, \mathbf{z}_{N-1}, u, \{t_k\}_{k=N}^M)$ ▷ Integrate under meal control
- 4: **return** $\{p_\psi(x_k | \mathbf{z}(t_k))\}_{k=N}^M$ ▷ Decode at query times

¹If one is interested in interpolation rather than extrapolation, it suffices to define $u^{(i)}$ over the observation window $[t_0^{(i)}, t_{N_i-1}^{(i)}]$ instead, and to encode backward in time following the interpolation regime of Algorithm 3.

²The specific parameterisation of $u^{(i)}$ is discussed in Section 3.1.2.1.

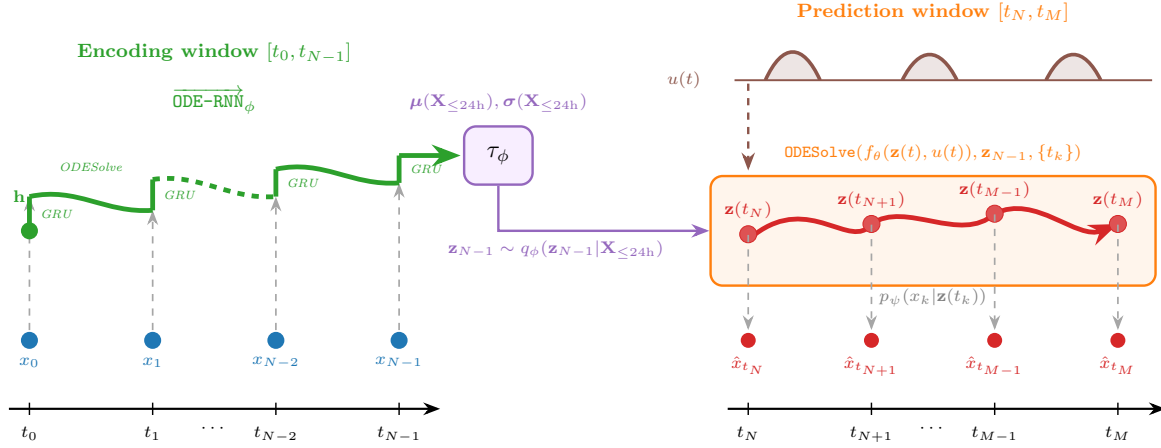


FIGURE 3.1: Schematic of the CLODE architecture. First the context glucose $\mathbf{X}_{\leq 24h}$ is encoded forward in time into a hidden state $\mathbf{h}_{N-1}$, which is then mapped by the translation network τ_ϕ to the parameters $\boldsymbol{\mu}(\mathbf{X}_{\leq 24h}), \boldsymbol{\sigma}(\mathbf{X}_{\leq 24h})$ of the variational posterior. Then, the latent initial condition at time t_{N-1} , $\mathbf{z}_{N-1}$ is sampled from this posterior and evolved forward under the learned dynamics f_θ , under a control input $u(t)$. Finally, observations are emitted from the latent trajectory thanks to an output network, typically parametrizing the mean of a predictive Gaussian distribution $p_\psi(x_k | \mathbf{z}(t_k)) = \mathcal{N}(\mu_\psi(\mathbf{z}(t_k)), \sigma_{\text{obs}}^2)$.

3.1.2.1 Model

The generative model is:

$$\mathbf{z}_{N-1} \sim p(\mathbf{z}_{N-1}) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (3.4)$$

$$\frac{d\mathbf{z}}{dt}(t) = f_\theta(\mathbf{z}(t), u(t)) \quad (3.5)$$

$$x_k \sim p_\psi(x_k | \mathbf{z}(t_k)) \quad (3.6)$$

In the extrapolation regime adopted here, the initial condition $\mathbf{z}_{N-1}$ is not sampled from the prior but inferred from the observed glucose history $\mathbf{X}_{\leq 24h} = \{(x_k, t_k)\}_{k=0}^{N-1}$ via the recognition network : $\mathbf{z}_{N-1} \sim q_\phi(\mathbf{z}_{N-1} | \mathbf{X}_{\leq 24h})$. The prior $p(\mathbf{z}_{N-1})$ intervenes only if we want to generate new plausible time series (hence the denomination *generative*.)

Each component of the model is detailed below.

Meal Control Signal

The representation of meal records in datasets typically consists, for each meal event j , of a carbohydrate amount $c_j \in \mathbb{R}_+$ at a timestamp t_j . This discrete representation is not suited for the ODE solver to do predictions at arbitrary times. The meal log must therefore be transformed to a continuous signal $u(t)$. Two different parametrization for $u(t)$ are presented below.

Dirac Impulse representation. A first naïve representation that one could use, which does not require any transformation for the meal log is the Dirac impulse³. The corresponding meal signal is:

$$u(t) = \sum_{j=0}^{C-1} c_j \delta(t - t_j) \quad (3.7)$$

However, the Dirac representation inherits another limitation of the meal logs, which is that meal events are presented as point perturbations, as if the full carbohydrate amount c_j was eaten instantaneously at t_j . This representation is not representative of the true meal ingestion process, which distribute that amount on a time window.

Beta kernel representation. ⁴ To represent more faithfully the meal ingestion process, we choose to replace the Dirac impulse with a Beta Kernel, that allows us to flexibly spread the ingestion of meal j over a physiologically plausible window $[t_j, t_j + W]$ as illustrated in Figure 3.2:

$$\tilde{u}(t) = \frac{c_j}{W \cdot B(a, b)} \left(\frac{t - t_j}{W} \right)^{a-1} \left(1 - \frac{t - t_j}{W} \right)^{b-1} \cdot \mathbb{I}_{t \in [t_j, t_j + W]}$$

where $B(a, b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ is the Beta function, $a, b > 0$ are the shape parameters, and $W > 0$ is the ingestion window duration.

The total meal signal is :

$$u(t) = \sum_{j=0}^{C-1} \tilde{u}_j(t) \quad (3.8)$$

As for the Dirac impulse representation, for which $\int u(\tau) d\tau = \sum_{j=0}^{C-1} c_j$, mass is also conserved thanks to the normalization of the Beta density.

The flexibility of the Beta distribution allows the kernel to capture different ingestion patterns, as illustrated in Figure 3.3. In the present work, a , b , and W are treated as global hyperparameters, shared across all individuals and meals. Conditioning them on the carbohydrate amount c_j or to external covariate (*e.g.* biological sex or age) would allow the kernel to adapt to individual ingestion patterns, but is left for future work.

³Recall that the Dirac delta $\delta(t - t_j)$ is a distribution satisfying $\delta(t - t_j) = 0$ for $t \neq t_j$, $\delta(t - t_j) = +\infty$ for $t = t_j$, and $\int_{-\infty}^{+\infty} \delta(t - t_j) dt = 1$, concentrating unit mass at t_j .

⁴We borrow the term *Beta kernel* from the nonparametric density estimation ([Chen, 2000](#)) literature. But our formulation and usage is distinct.

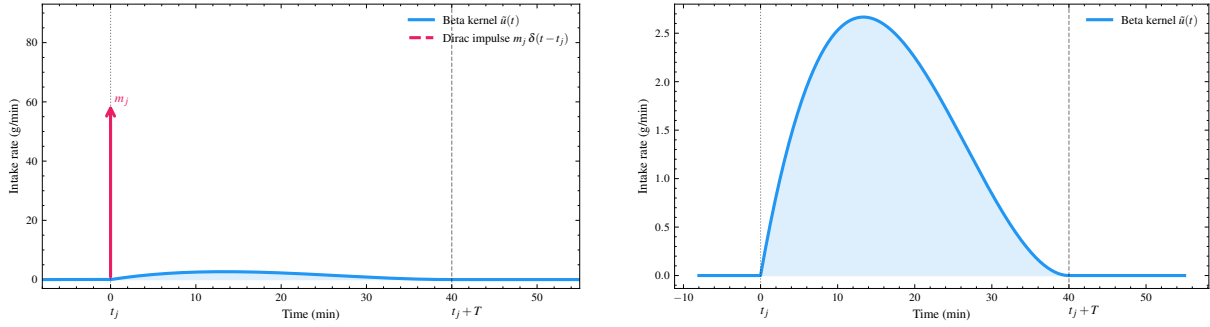


FIGURE 3.2: Continuous representations of meal ingestion. **Left:** Comparison of Dirac and Beta(2,3) kernel representations for a single meal ($c_j = 60$ g, $W = 40$ min). **Right:** Zoom on the Beta kernel.

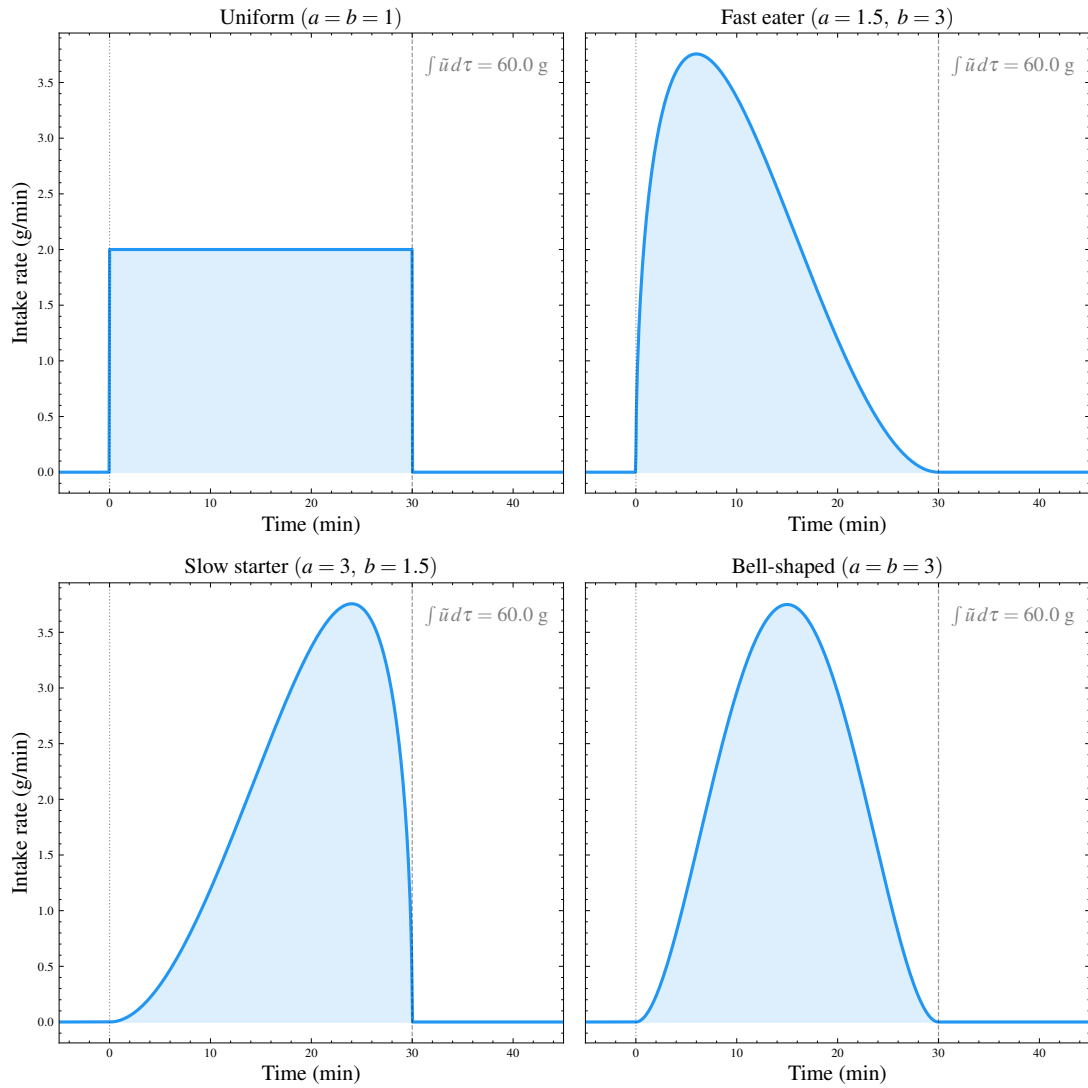


FIGURE 3.3: Beta kernel archetypes for $c_j = 60$ g and $W = 40$ min, illustrating the flexibility of the shape parameters a and b . All kernels integrate to exactly c_j regardless of (a, b) .

Comparison with Neural CDEs. We prefer the Beta kernel over Neural CDEs (Kidger et al., 2020), which allow the incorporation of control by constructing a continuous control path $\mathcal{C}(t)$ via an interpolation scheme. Applied to the meal log, interpolation would imply overestimation of the meal ingestion, given that the meal log does not represent the true ingestion at time t_j but the total amount ingested. Our Beta kernel avoids this by construction. The signal spreads the meal ingestion over a physiologically plausible window, while preserving mass and remaining zero outside each meal window.

Recognition Network

The encoder follows the extrapolation regime of Rubanova et al. (2019) (Algorithm 4), processing the observed glucose history $\mathbf{X}_{\leq 24\text{h}} = \{(x_k, t_k)\}_{k=0}^{N-1}$ forward in time via an ODE-RNN (Section 2.3.1.3). The final hidden state is passed through a translation network τ_ϕ to produce the variational posterior:

$$q_\phi(\mathbf{z}_{N-1} \mid \mathbf{X}_{\leq 24\text{h}}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{X}_{\leq 24\text{h}}), \text{diag}(\boldsymbol{\sigma}^2(\mathbf{X}_{\leq 24\text{h}}))) \quad (3.9)$$

where $[\boldsymbol{\mu}(\mathbf{X}_{\leq 24\text{h}}), \boldsymbol{\sigma}(\mathbf{X}_{\leq 24\text{h}})] = \tau_\phi(\overrightarrow{\text{ODE-RNN}_\phi(\mathbf{X}_{\leq 24\text{h}})})$. We underscore that the encoder vector field is separate from the latent vector field. A distinction sometimes overlooked in the literature.

Emission Network

The emission network is a shallow MLP mapping each latent state $\mathbf{z}(t_k)$ to the predicted glucose mean, with a fixed observation noise σ_{obs} :

$$p_\psi(x_k \mid \mathbf{z}(t_k)) = \mathcal{N}(\mu_\psi(\mathbf{z}(t_k)), \sigma_{\text{obs}}^2) \quad (3.10)$$

where σ_{obs} is a global hyperparameter shared across individuals and time points, following Chen et al. (2018).

3.1.2.2 Training

Following the VAE derivation of Section 2.3.3.1, the training objective is the ELBO adapted to the sequential setting of (2.62):

$$\mathcal{L}(\theta, \psi, \phi; \mathbf{X}_{\leq 24\text{h}}) = \mathbb{E}_{q_\phi(\mathbf{z}_0 \mid \mathbf{X}_{\leq 24\text{h}})} \left[\sum_{k=N}^M \log p_\psi(x_k \mid \mathbf{z}(t_k)) \right] - \beta_{\text{KL}} D_{\text{KL}}(q_\phi(\mathbf{z}_0 \mid \mathbf{X}_{\leq 24\text{h}}) \parallel p(\mathbf{z}_0)) \quad (3.11)$$

where the reconstruction sum runs over the extrapolation window $[t_N, t_M]$. The weight β_{KL} of the KL term is either fixed or annealed following Bowman et al. (2016); some specific configurations are discussed in Section 3.2.

At each training step, the encoder processes a randomly subsampled subset of the N observations in the encoding window, with between $\lfloor N/4 \rfloor$ and N valid observations retained at each gradient step. This exposes the model to different temporal windows and observation densities during training, encouraging the encoder to produce posterior estimates under varying missingness patterns.

3.1.2.3 Properties

We have already established that CLODE explicitly separates manifest variables governed by differential equations (the latent physiological state) from those that are exogenous to the system dynamics (carbohydrate intake), in contrast to the vanilla Latent ODE-RNN.

We now discuss two other properties of particular interest given our research question: the ability to perform counterfactual dietary predictions, and the potential for physiologically structured latent representations.

Counterfactual Dietary Simulation

At inference time, the recognition network infers $\mathbf{z}_{N-1}$ from the *factual* (observed) glucose history. This initial condition is intended to represent the individual’s latent physiological state at time t_{N-1} . Based on it, we can forecast the glucose trajectory conditionally on a hypothetical prospective meal signal $u'(t)$, keeping $\mathbf{z}_{N-1}$ fixed:

$$\{\mathbf{z}'(t_k)\}_{k=N}^M = \text{ODESolve}(f_\theta, u', \mathbf{z}_{N-1}, \{t_k\}_{k=N}^M) \quad (3.12)$$

This yields a predicted glucose trajectory for the same individual under an alternative dietary scenario. In our setting, this means that given the first 24 hours of observed glucose, one can either extrapolate the next 24 hours conditioning on the known meal log (which amounts to factual reconstruction), or produce counterfactual predictions conditioning on a hypothetical meal scenario. Hence, CLODE can be used as a tool for answering questions of the form: *how would this individual have responded under a different perturbation scenario?*

Physiologically Structured Latent Space

In the Latent ODE-RNN framework, the initial condition $\mathbf{z}_{N-1}$ is the only individual-specific quantity in the model. Ideally, it should encode the physiological state of the individual (*e.g.* their metabolic baseline, insulin sensitivity).

The point of lifting into a latent space (beyond the approximation-theoretic argument), is that individuals do not share identical glucose dynamics. Parametrizing a single neural differential equation directly in the glucose observation space would therefore fail. By lifting observations into a higher-dimensional latent space instead, the recognition network can project individuals into different regions of $\mathbb{R}^s$, each corresponding to a different initial condition $\mathbf{z}_{N-1}$ from which individuals evolve differently under the same shared vector field f_θ .

CLODE is further designed to favor a clean physiological interpretation of that latent space. By supplying meal intake as an explicit control input, the latent space is incentivized to capture genuine inter-individual physiological variability. Indeed, a naive application of the Latent ODE-RNN would instead force the network to simultaneously represent the individual’s physiology and the effect of their meal schedule in $\mathbf{z}_{N-1}$. Worse, asking the ODE to learn dynamics for $u(t)$ (which by definition has no dynamics) would risk atrophying the training, as the vector field would be tasked with explaining a signal that cannot be.

Whether CLODE achieves this in practice is an empirical question, which we investigate in Chapter 4.

3.2 Evaluation Protocol

3.2.1 Benchmark Models

Models are selected to challenge the proposed model on three specific tasks along complementary axes, as illustrated in Table 3.1.

On the one hand, the ODE-LSTM with heteroskedastic prediction head and mixed objective of [Fitzgerald et al. \(2023\)](#) is an autoregressive benchmark that handles irregular time series and exogenous inputs, but provides no explicit latent representation.

On the other hand, the Latent ODE-RNN of [Rubanova et al. \(2019\)](#) is the direct precursor to CLODE, sharing the same VAE backbone and ODE-RNN encoder, but lacking any mechanism for incorporating exogenous control into the latent dynamics. Results obtained with this model can therefore be interpreted as those of an ablation study.

Both baselines are adapted to our experimental setup to enable long-horizon forecasting and counterfactual evaluation. Implementation details and adaptations are provided below.

Table 3.1: Comparison of model capabilities across the three evaluation axes.

Model	Factual Extrapolation	CF Extrapolation	Representation Learning
ODE-LSTM	✓*	✓*	✗
Latent ODE-RNN	✓	✗	✓
CLODE	✓	✓	✓

* ODE-LSTM supports extrapolation but is not designed for long-horizon trajectory prediction.

3.2.1.1 ODE-LSTM

The ODE-LSTM follows the architecture of [Fitzgerald et al. \(2023\)](#), with a preprocessing MLP encoding glucose observations before each LSTM update, and a NODE flow map interpolating the hidden state between observations (Section 2.3.1.5). The model is trained using ADAM ([Kingma and Ba, 2017](#)) on the full 48-hour sequence with a double loss consisting of (i) a predictive negative log-likelihood computed from the hidden state after the ODE evolution step, and (ii) a KL regularization term computed from the hidden state immediately after the LSTM update, weighted by a mixing rate λ_{mix} .

Usually, ODE-LSTM are primarily designed for 1-step ahead prediction or short-term extrapolation relying on *teacher forcing*, meaning that the true glucose value is fed back at each step during training. Applied to our 24-hour extrapolation task, errors can accumulate quickly ([Bengio et al., 2015](#)) because, at inference, the model must rely on its own predictions as inputs, a regime never encountered during training. To mitigate this, we adopt a curriculum learning strategy inspired by the scheduled sampling procedure ([Bengio et al., 2015](#)), linearly decaying the teacher forcing probability from 1 to 0 over training:

$$\epsilon(e) = 1 - \frac{e}{E}, \quad e \in \{1, \dots, E\} \quad (3.13)$$

where e is the current epoch and E the total number of training epochs, so that $\epsilon(0) = 1$

(full teacher forcing) and $\epsilon(E) = 0$ (fully autoregressive)⁵.

Validation loss is computed under a fully autoregressive regime to ensure that model selection reflects the true inference regime.

3.2.1.2 Latent ODE-RNN

The Latent ODE-RNN follows the extrapolation regime of Algorithm 4. The encoder processes the first 24 hours of CGM observations jointly with the meal log.

Both glucose measurements and meal events are placed on a common time grid formed by the union of all CGM timestamps and meal timestamps across the batch. Each timestep is represented by a four-dimensional vector $[G, c, m_G, m_c]$, where G and c are the glucose and carbohydrate values (zero-padded where unobserved), and $m_G, m_c \in \{0, 1\}$ are binary masks indicating which channels are observed at each time.

The training objective is the same as the one used by CLODE, *i.e* the ELBO as defined in (3.11), and is optimized using ADAM (Kingma and Ba, 2017), with a learning rate of 0.001 and a weight decay of 0.0001.

The subsampling procedure is also similar to CLODE (see Section 3.1.2.2). However, unlike CLODE, where meal information bypasses the encoder entirely and enters the model as a control input in the latent NODE, the Latent ODE-RNN feeds meal events directly into the encoder. Since meals are assumed fully known, meal timestamps are always retained and never masked during subsampling.

3.2.1.3 Hyperparameters Tuning and Model Selection

Best models are selected based on validation performance in terms of the model-specific loss. Given the computationally demanding nature of the training process, for each model, only a set of hyperparameters pre-selected based on preliminary experiments is explored. Results and architecture details are reported in Appendix B.1 and B.2.

For the ODE-LSTM, we vary the mixing rate $\lambda_{\text{mix}} \in \{0.0001, 0.001, 0.01\}$.

For CLODE and the Latent ODE-RNN models, we vary the latent space dimension $s \in \{3, 4, 6\}$ in the IIV setting (fixed at $s = 4$ in the No-IIV setting), and the KL regularization term, which is either chosen constant $\beta_{\text{KL}} \in \{0.5, 1.0, 2.0\}$, or linearly increased from 0 to 1 over the first 100 epochs:

$$\beta_{\text{KL}}(e) = \min\left(1, \frac{e}{100}\right) \quad (3.14)$$

We additionally explore a decoder output bias initialization of 4.3 mmol/L, approximately corresponding to the 10th percentile of the glucose distribution in the training data, as an alternative to the default zero initialization.

For CLODE specifically, we compare the Dirac and β -Kernel meals signals. Since we assume no prior knowledge about individual meal ingestion patterns, we fix the shape parameters of the β -Kernel to $a = b = 1$, which corresponds to a uniform distribution over the ingestion window W , set to 30 minutes.

⁵A similar approach was used by Rubanova et al. (2019), who applied scheduled sampling with a fixed probability of 0.5 when training ODE-RNN for extrapolation.

3.2.2 Simulation Study

3.2.2.1 Data Generating Process

Physiological Simulator

We simulate glucose trajectories using a physiologically inspired model of glucose-insulin dynamics, adapted from the minimal model of [Bergman et al. \(1979\)](#) and extended to incorporate circadian variation and a bi-modal meal absorption profile, capturing fast and slow glucose appearance phases via a mixture of two log-normal densities.⁶ The state vector is $[G(t), X(t)]^\top$, where $G(t)$ is blood glucose concentration and $X(t)$ is the remote insulin action in peripheral tissues. The dynamics are governed by:

$$\frac{dG}{dt}(t) = -G(t) X(t) - p_1(G(t) - G_b(t)) + R(t) \quad (3.15)$$

$$\frac{dX}{dt}(t) = -p_2(X(t) - S_G Z(t)) \quad (3.16)$$

where the circadian baseline $G_b(t)$, the glucose appearance rate $R(t)$, and the insulin secretion function $Z(t)$ are defined as:

$$G_b(t) = \mu_b + A_{\text{circ}} \cos\left(\frac{2\pi(t - \varphi)}{24}\right) + B_{\text{circ}} \sin\left(\frac{2\pi(t - \varphi)}{24}\right) \quad (3.17)$$

$$R(t) = \frac{N_E}{V} \sum_{j=0}^{C-1} c_j [(1 - R_H) f_{\text{LN}}(\tau; P_1, \omega_1) + R_H f_{\text{LN}}(\tau; P_2, \omega_2)], \quad \tau = t - t_m \quad (3.18)$$

$$Z(t) = \frac{G(t) - G_b(t)}{1 + e^{-\alpha(G(t) - G_b(t))}} + \gamma R(t) \quad (3.19)$$

and where $f_{\text{LN}}(\tau; P, \omega)$ is a log-normal density parameterised by peak time P and width ω :

$$f_{\text{LN}}(\tau; P, \omega) = \frac{1}{\tau \sqrt{\pi \omega}} \exp\left(-\frac{(\log(\tau/P) + \omega/2)^2}{\omega}\right), \quad \tau > 0 \quad (3.20)$$

In $R(t)$, c_j denotes the carbohydrate amount of meal j , consistent with the meal log notation of Section 3.1.1. $V > 0$ is the glucose distribution volume. $N_E > 0$ is a nutrient effect coefficient, parametrizing the overall glycemic impact of carbohydrate intake. Finally, $R_H \in (0, 1)$ is the mixing weight of the slow absorption component: a fraction R_H of each meal is absorbed via the slow log-normal pathway (peak time P_2), and the remaining $1 - R_H$ via the fast pathway (peak time P_1).

Population Design

We simulate two *in silico* populations of $N = 500$ healthy individuals over a 48-hour window.

For each population, individuals are randomly partitioned as follows : 30% of individuals ($n = 150$) are held out as a test set. The remaining 70% are split 80/20 into a training set ($n = 280$) and a validation set ($n = 70$) used for early model selection.

⁶The simulator was kindly provided by PhD Candidate Guillaume Deside (UCLouvain, ISBA). We use the simulator equations without modification. We only adapt the equations to fit our single-nutrient experimental setup.

The first population is *physiologically homogeneous*. All individuals share identical physiological parameters ($p_1, p_2, \mu_b, S_G, N_E, P_1, \phi, V, \gamma, \alpha, R_H, P_2, \omega_1, \omega_2$), and differ only in their eating habits. Each individual is assigned one of three eating profiles (*picky*, *average*, or *big* eater) which govern their two degrees of freedom with profile-specific distributions : the number of daily meals (between 1 and 3), and carbohydrate amount per meal. Meal timing constitutes a third degree of freedom, shared across all profiles. Each meal is assigned to one of three time slots (Breakfast, Lunch, Dinner), with timing drawn uniformly within the slot (see Table 3.2).

Table 3.2: Eating profile definitions.

	Picky	Average	Big
Active slots per day	1 or 2	2 or 3	3
Carbohydrates (g/meal)	[20, 50]	[40, 80]	[70, 120]
Number of individuals	125	250	125
3 slots : Breakfast (07:00–09:00) Lunch (12:00–14:00) Dinner (18:00–20:00).			

This population serves mainly as a sanity check for the encoder. Since $\mathbf{z}_{N-1}$ is intended to encode the physiological state of the individual, it should exhibit no systematic discrimination between eating profiles in the absence of physiological variability. Any apparent clustering would indicate that the model conflates the effect of exogenous inputs with genuine metabolic differences.

The second population introduces *inter-individual variability* (IIV) by randomizing a subset of physiological parameters across individuals : the baseline glucose level μ_b , the insulin sensitivity S_G , the circadian phase shift ϕ ; and two parameters governing the shape and timing of the postprandial glucose appearance curve : the nutrient effect coefficient N_E and the first absorption peak time P_1 . Parameters are drawn independently from the uniform distributions specified in Table 3.3

Table 3.3: Physiological parameter distributions for the IIV setting. Parameters are drawn independently per individual from uniform distributions. In the No-IIV setting, all parameters are fixed at $\pi_{\text{no-iiv}}$.

Parameter	Symbol	Unit	Distribution	$\pi_{\text{no-iiv}}$
<i>Physiological parameters</i>				
Baseline glucose	μ_b	mmol/L	$\mathcal{U}(4.50, 6.50)$	5.50
Insulin sensitivity	S_G	$\text{min}^{-1}/(\text{mmol/L})$	$\mathcal{U}(0.010, 0.020)$	0.015
Phase Shift	ϕ	hour	$\mathcal{U}(5, 7)$	6
<i>Meal parameters</i>				
Effect coefficient	N_E	–	$\mathcal{U}(0.25, 1.00)$	0.625
First absorption peak	P_1	min	$\mathcal{U}(20, 90)$	55.0

All remaining parameters are fixed at their default values across both populations: $p_1 = 0.020$, $p_2 = 0.050$, $V = 1.50$, $\gamma = 10.0$, $\alpha = 0.018$, $A_{\text{circ}} = 0.50$, $B_{\text{circ}} = 0.30$, $P_2 = 120.0 \text{ min}$, $\omega_1 = 0.80$, $\omega_2 = 1.20$, $R_H = 0.425$.

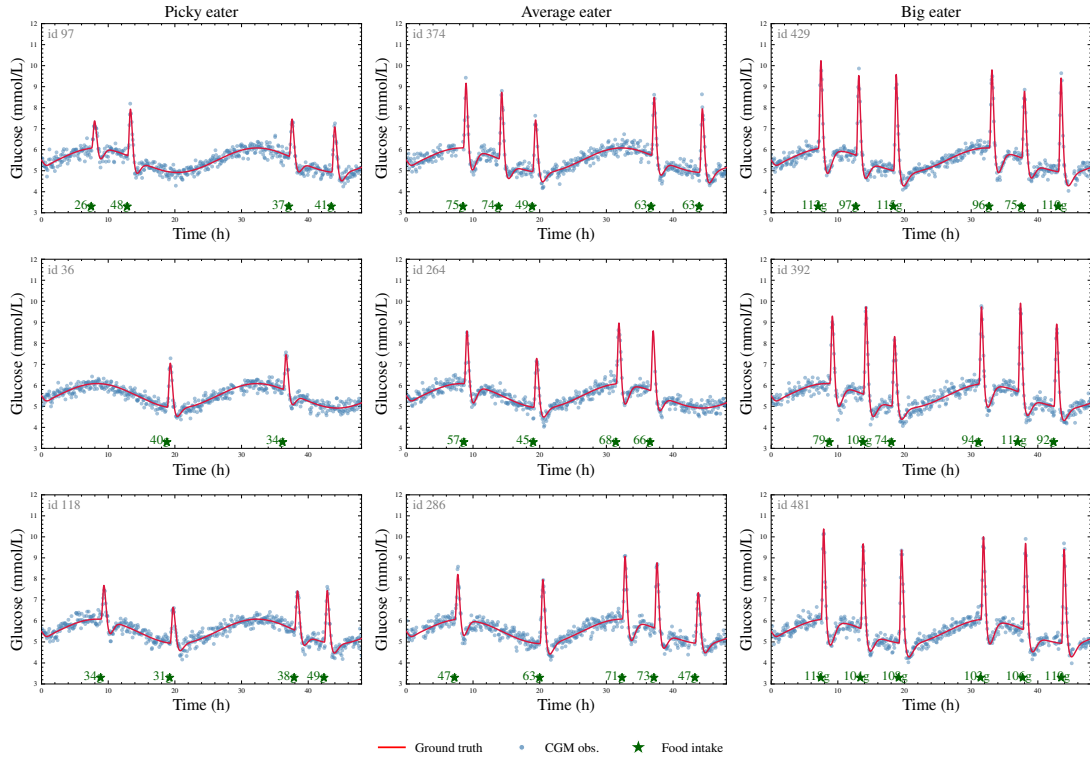


FIGURE 3.4: Random trajectories from the No-IIV population, stratified by eating profile. Each panel shows the ground-truth glucose trajectory (red) and CGM observations (blue).

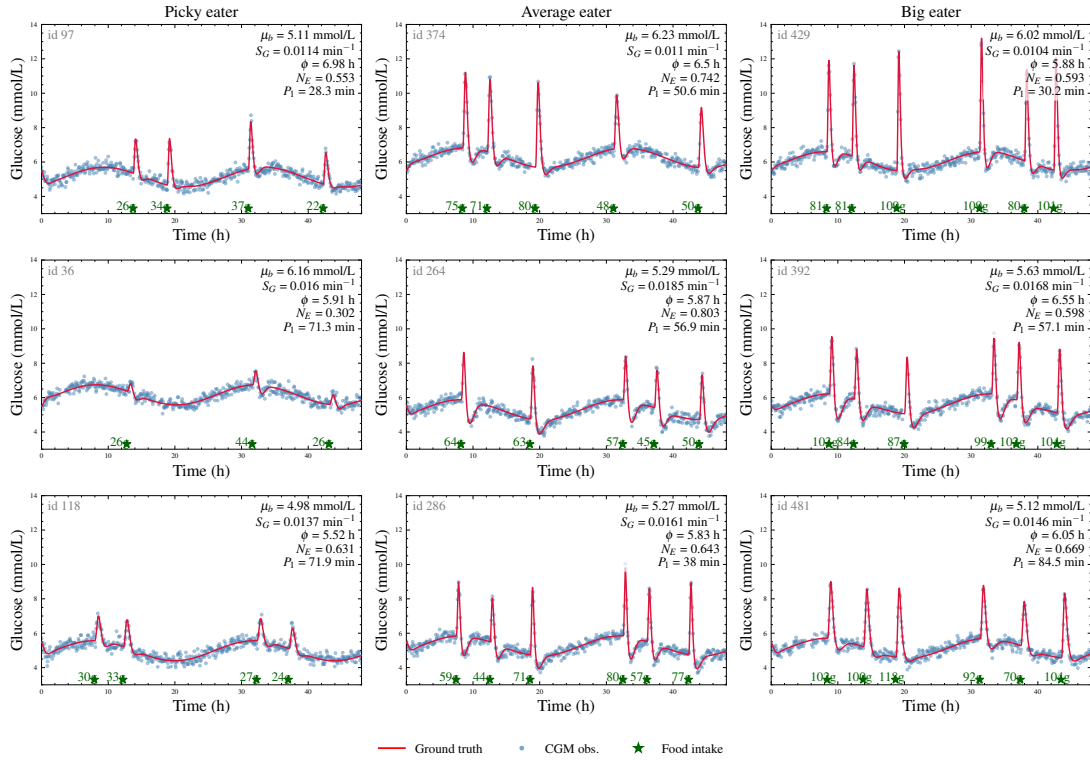


FIGURE 3.5: Random trajectories from the IIV population. Individual physiological parameters are reported in each panel.

From each population, CGM traces are first simulated on a regular 5-minute grid $\{t_1, \dots, t_K\}$ with $K = 48 \times 12 \text{ points/h} + 1 = 577$ points over 48 hours. Three sources of sampling irregularity are then applied sequentially to mimic those observed in real-life CGM measurements.

- First, each 24-hour window is independently exposed to dropout blocks with probability 0.30; when present, one to two contiguous segments of 30 to 60 minutes are removed from the grid, simulating sensor removal or sensor memory overload. We denote $\mathcal{D}_i \subset \{t_1, \dots, t_K\}$ the set of timepoints (potentially) removed by this dropout block process for individual i .
- Second, each remaining timestamp $t_k \notin \mathcal{D}_i$ is perturbed as $\tilde{t}_k = t_k + \delta_k$, $\delta_k \sim \mathcal{N}(0, 1^2)$ minutes, reflecting clock drift or Bluetooth transmission delays.
- Third, each perturbed observation is retained independently with probability 0.95, via $\gamma_k \sim \text{Bernoulli}(0.95)$, with dropped points simulating isolated signal losses.

The final observation grid for individual i is thus $\mathcal{T}_i = \{\tilde{t}_k : t_k \notin \mathcal{D}_i \text{ and } \gamma_k = 1\}$.

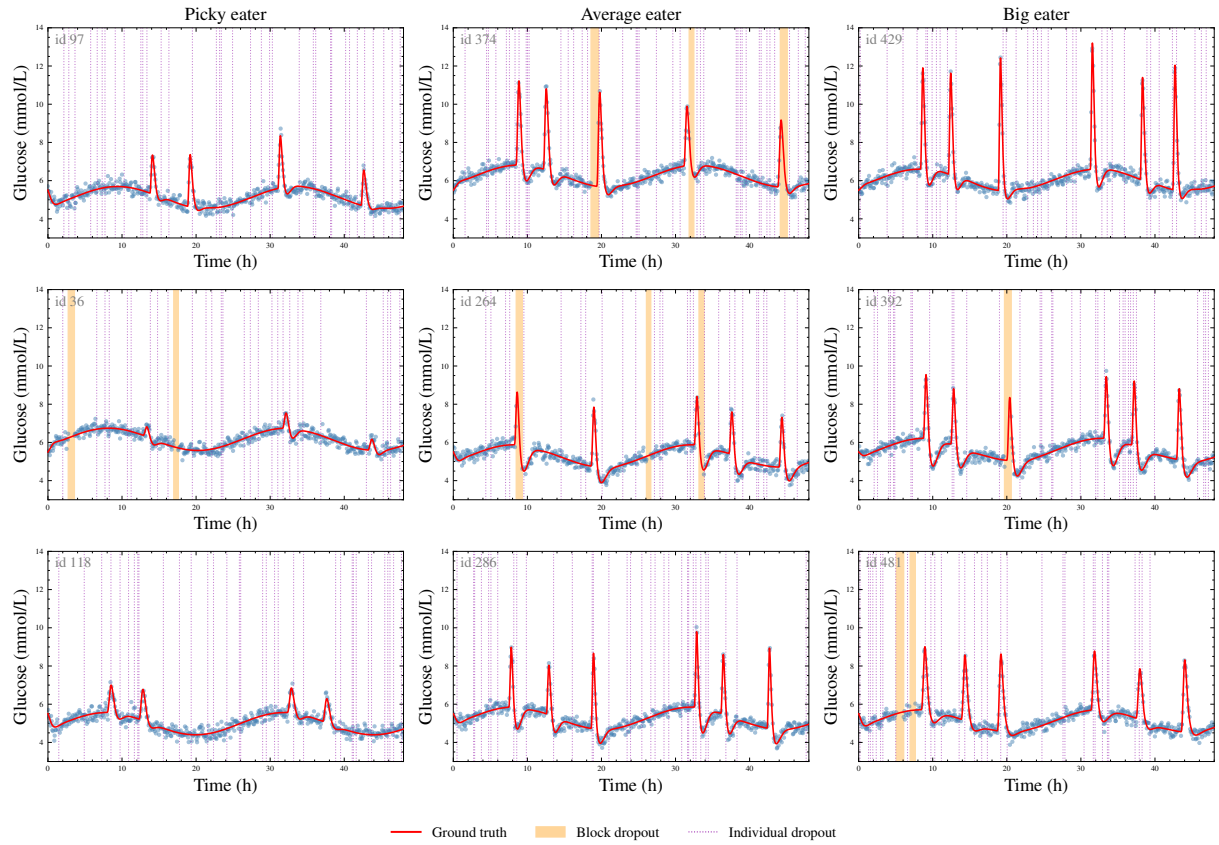


FIGURE 3.6: Illustration of the observation process on the IIV population. Orange bands indicate block dropout segments; purple dotted lines mark isolated individual-level gaps. The CGM measurements (blue) is the result of applying all three irregularity sources sequentially.

3.2.3 Evaluation Metrics

We evaluate the model along three axes, each with a dedicated set of metrics.

3.2.3.1 Forecasting accuracy

We assess point prediction accuracy via the root mean squared error:

$$\text{RMSE} = \sqrt{\frac{1}{M-N} \sum_{k=N}^M (x_k - \hat{x}_k)^2} \quad (3.21)$$

where x_k is the observed glucose at time t_k , $\hat{x}_k$ is the predicted mean, and the sum runs over the $M - N$ observations in the extrapolation window.

Since the model produces a full predictive distribution rather than a point estimate, following [Fitzgerald et al. \(2023\)](#), we additionally report the *Continuous Ranked Probability Score* (CRPS), a strictly proper scoring rule that evaluates the entire predictive distribution against the observed glucose x_k :

$$\text{CRPS}(F_\psi, x_k) = \int_{-\infty}^{+\infty} (F_\psi(\xi | \mathbf{z}(t_k)) - \mathbb{I}_{\{\xi \geq x_k\}})^2 d\xi \quad (3.22)$$

where F_ψ is the CDF of $p_\psi(x_k | \mathbf{z}(t_k))$. The CRPS reduces to the mean absolute error when F_ψ is a point mass, and penalises both bias and miscalibrated uncertainty simultaneously. Denoting $\hat{\mu}_k = \hat{x}_k$, the closed form for our Gaussian emission model is:

$$\text{CRPS}_k = \sigma_{\text{obs}} \left[\frac{x_k - \hat{\mu}_k}{\sigma_{\text{obs}}} \left(2\Phi\left(\frac{x_k - \hat{\mu}_k}{\sigma_{\text{obs}}}\right) - 1 \right) + 2\phi\left(\frac{x_k - \hat{\mu}_k}{\sigma_{\text{obs}}}\right) - \frac{1}{\sqrt{\pi}} \right] \quad (3.23)$$

where Φ and ϕ denote the CDF and PDF of the standard normal distribution. The reported CRPS is averaged over the prediction window: $\frac{1}{M-N} \sum_{k=N}^{M-1} \text{CRPS}_k$. Figure 3.7 illustrates the score intuitively for four predictive Gaussian distributions with varying mean and variance.

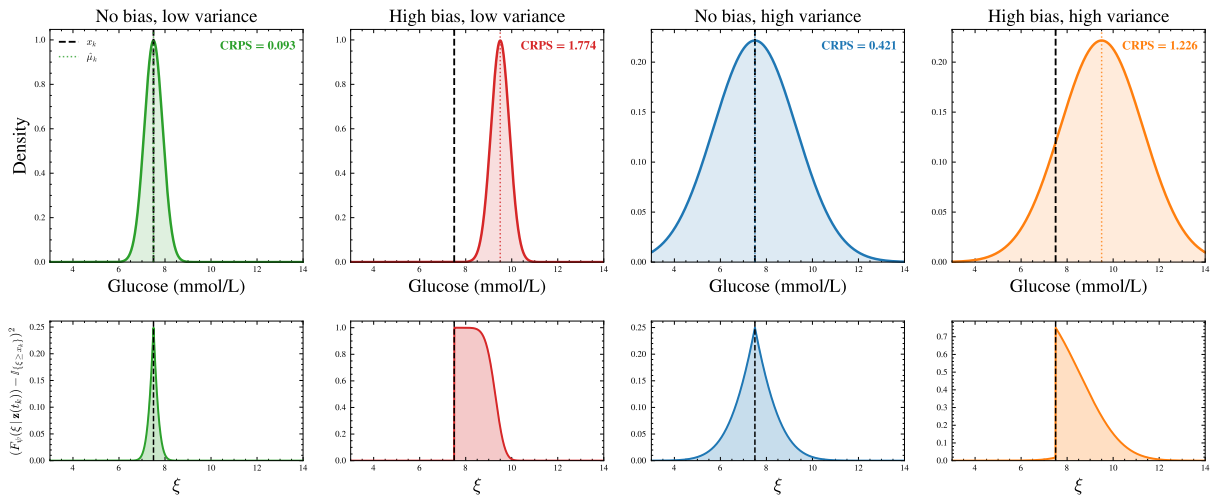


FIGURE 3.7: Illustration of the CRPS for four Gaussian predictive distributions, varying bias and variance. **Top** : predictive PDF alongside the true observation x_k (dashed vertical line). **Bottom** : corresponding integrand.

We further report *empirical coverage* at the 90% level: the fraction of observations falling within the predicted 90% prediction interval, which should equal 0.90 for a well-calibrated model:

$$\text{Coverage}_{90} = \frac{1}{M - N} \sum_{k=N}^{M-1} \mathbb{I}_{\{x_k \in [\hat{q}_{0.05,k}, \hat{q}_{0.95,k}]\}} \quad (3.24)$$

where $\hat{q}_{\alpha,k}$ denotes the α -quantile of the k -th predictive distribution.

3.2.3.2 Latent space structure

Following [Rubanova et al. \(2019\)](#), we project the inferred initial latent conditions $\{\mathbf{z}_{N-1,i}\}_{i=1}^n$ to two dimensions using UMAP ([McInnes et al., 2020](#)) and colour each point by its ground-truth physiological parameter values and eating profiles. To quantify the alignment between the latent representation and the true physiological variation, we compute the Spearman rank correlation between each latent dimension $z_{N-1,j}$ and each ground-truth parameter π_k :

$$\rho_{j,k} = \frac{\text{cov}(\text{rank}(z_{N-1,j}), \text{rank}(\pi_k))}{\sigma_{\text{rank}(z_{N-1,j})} \sigma_{\text{rank}(\pi_k)}} \quad (3.25)$$

Furthermore, we report the RV coefficient ([Robert and Escoufier, 1976](#)), a multivariate generalisation of the squared Pearson correlation that measures the global linear dependence between two data matrices. Let $\tilde{\mathbf{Z}} = \mathbf{Z} - \iota \bar{\mathbf{z}}^\top$ and $\tilde{\mathbf{\Pi}} = \mathbf{\Pi} - \iota \bar{\boldsymbol{\pi}}^\top$ denote the column-centred versions of the latent matrix $\mathbf{Z} \in \mathbb{R}^{n \times s}$ and the physiological parameter matrix $\mathbf{\Pi} \in \mathbb{R}^{n \times p}$, where $\bar{\mathbf{z}} \in \mathbb{R}^s$ and $\bar{\boldsymbol{\pi}} \in \mathbb{R}^p$ are the column means. The RV coefficient is then:

$$\text{RV}(\mathbf{Z}, \mathbf{\Pi}) = \frac{\text{tr}(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^\top \tilde{\mathbf{\Pi}}\tilde{\mathbf{\Pi}}^\top)}{\sqrt{\text{tr}(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^\top) \text{tr}(\tilde{\mathbf{\Pi}}\tilde{\mathbf{\Pi}}^\top \tilde{\mathbf{\Pi}}\tilde{\mathbf{\Pi}}^\top)}} \quad (3.26)$$

As for dimension-wise correlations, the RV coefficient takes values in $[0, 1]$, with values close to 1 indicating that the latent space closely mirrors the structure of the physiological parameter space.

3.2.3.3 Counterfactual validity

Following [De Brouwer et al. \(2022\)](#), we evaluate counterfactual predictions using two metrics. First, the *factual RMSE* and *counterfactual RMSE* are reported separately, to assess whether performance degrades when the model is queried outside the training distribution of meal inputs. Second, we report the *Precision in Estimation of Heterogeneous Effects* (PEHE), which measures the ability to correctly predict the differential effect between two interventions u and u' :

$$\text{PEHE} = \sqrt{\frac{1}{M - N} \sum_{k=N}^{M-1} [(x_k^u - x_k^{u'}) - (\hat{x}_k^u - \hat{x}_k^{u'})]^2} \quad (3.27)$$

Since the simulator provides ground-truth trajectories under any intervention by construction, PEHE can be computed exactly.

4.1 Forecasting Performance

We first evaluate the ability of CLODE to extrapolate glucose trajectories over the 24-hour prediction window. Results are reported for the best-performing configuration of each model in terms of validation loss¹.

Based on Figure 4.1, we observe that both variants of CLODE outperform the baselines. As expected, the β -Kernel variant, which distributes the observed carbohydrate amount, achieves better performance than the Dirac impulse variant, which models meals as instantaneous events. Exact values are reported in Table 4.1.

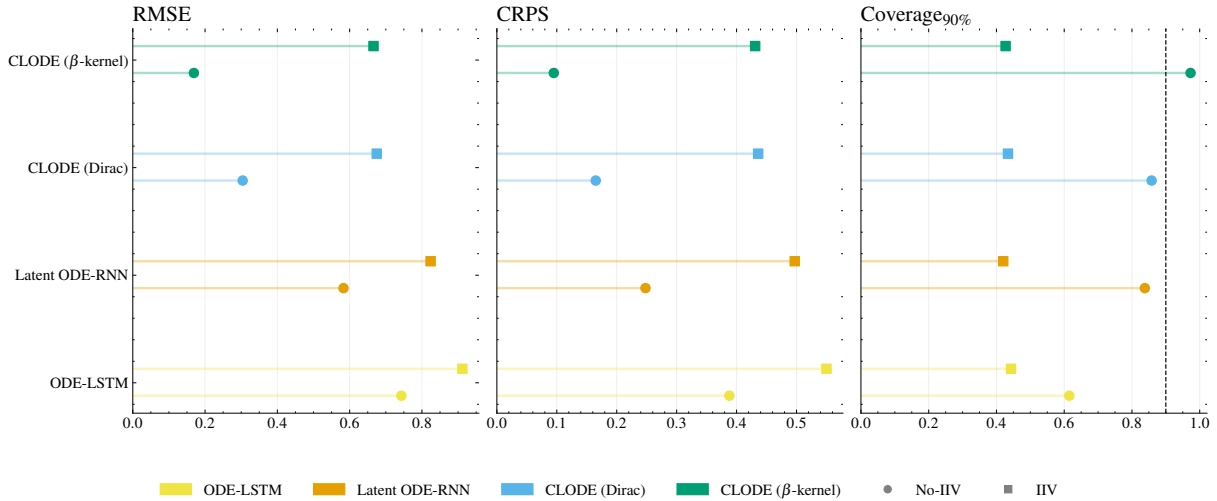


FIGURE 4.1: Forecasting performance across models and evaluation settings. Each panel corresponds to a metric; circles and squares distinguish the No-IIV and IIV settings respectively. The dashed vertical line in the Coverage panel marks the 90% target.

¹The retained configurations for the No-IIV (resp. IIV) are : for the ODE-LSTM, $\lambda_{\text{mix}} = 0.001(0.001)$; for the Latent ODE-RNN, $s = 4(6)$, $\beta_{\text{KL}} = 1.0(2.0)$ and output bias of 4.3 (4.3); for CLODE (Dirac), $s = 4(3)$, β_{KL} linearly annealed (0.5) and output bias of 4.3 (4.3); for CLODE (β -kernel), $s = 4(6)$, β_{KL} linearly annealed (0.5) and output bias of 0.0 (4.3).

Table 4.1: Forecasting performance across the two evaluation datasets. Best result per column is **bolded**; second-best is underlined.

Model	No-IIV			IIV		
	RMSE	CRPS	Coverage _{90%}	RMSE	CRPS	Coverage _{90%}
ODE-LSTM	0.743	0.388	0.615	0.912	0.550	0.443
Latent ODE-RNN	0.583	0.248	0.838	0.824	0.497	0.420
<i>Ours</i>						
CLODE (Dirac)	<u>0.304</u>	<u>0.165</u>	<u>0.858</u>	<u>0.675</u>	<u>0.436</u>	<u>0.434</u>
CLODE (β -kernel)	0.169	0.095	0.973	0.666	0.431	0.427

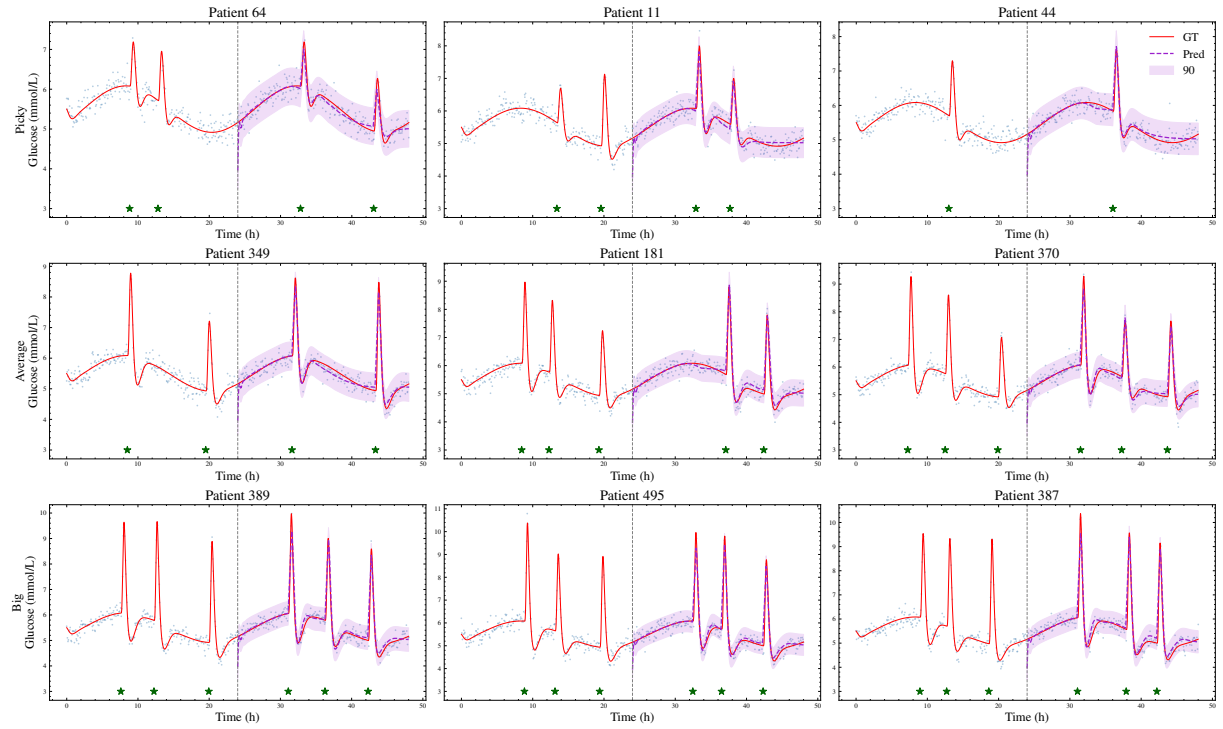
Notably, our β -Kernel variant achieves by far the best predictive performance across most metrics on both evaluation settings, with a particularly marked improvement in the No-IIV population, where it achieves a RMSE and CRPS approximately twice as low as the second best model (CLODE with Dirac Impulse). In the IIV setting, the β -Kernel variant still outperforms its Dirac counterpart, as well as the baselines, but performance gaps are narrower, especially between the two CLODE variants.

In particular, for the IIV population, the β -Kernel variant seem to succeed only for certain individuals. For instance, Patients 11 and 495 in Figure 4.2b present a systematic vertical (upward) bias in their predictions, while Patients 64 and 370 show a systematic vertical (downward) bias. Moreover, for some Patients, such as Patient 44, the magnitude of postprandial glucose excursions are underestimated. This suggests that the model may be struggling to capture the full extent of inter-individual variability in this setting. This will be further investigated in the next two sections.

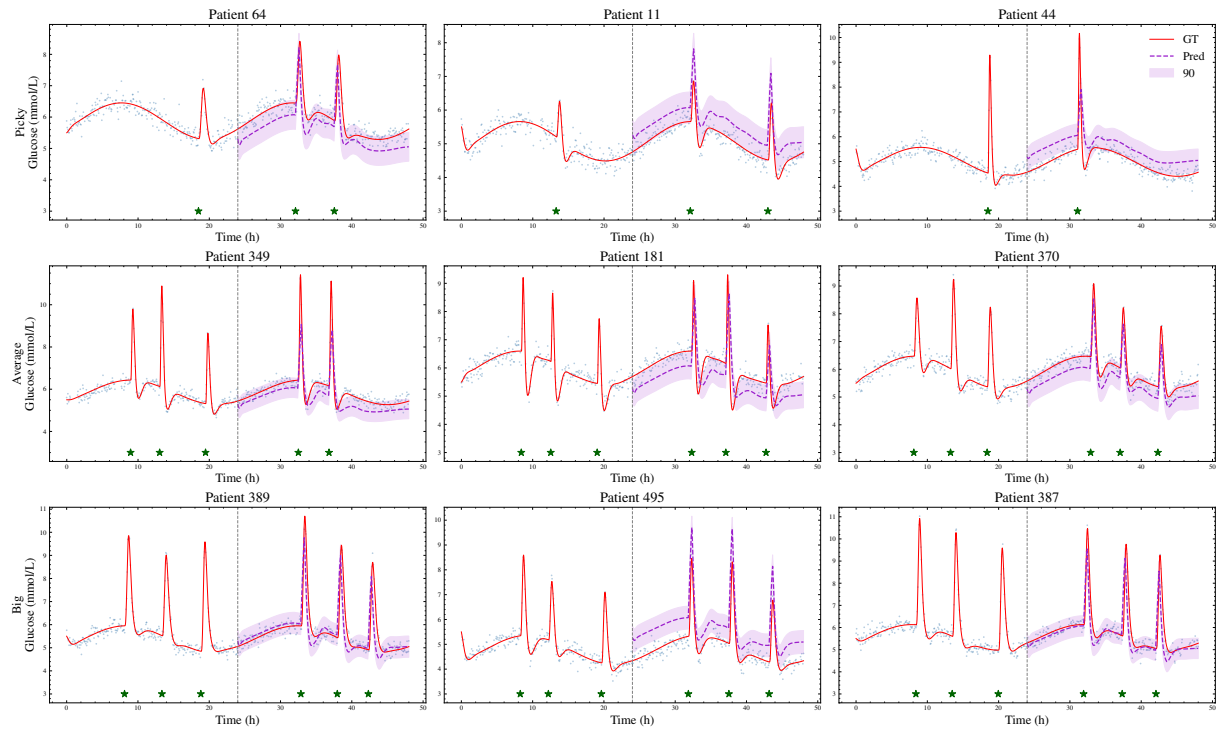
Among baselines, the Latent ODE-RNN outperforms the ODE-LSTM across all metrics (except for the Coverage in the IIV setting). This is in line with the literature, where [Rubanova et al. \(2019\)](#) showed empirically that the closely related ODE-RNN model was struggling with long-term extrapolation. We show here that this issue extends to the ODE-LSTM, even with the adopted curriculum learning strategy.

While the Latent ODE-RNN performs better than the ODE-LSTM, it is still outperformed by both CLODE variants. The training visualizations of Appendix B.5 suggests that the model is able to fit the circadian baseline reasonably well in early epochs, but fails to capture the postprandial glucose excursions, unlike CLODE (See Figure 4.2a). This highlights the importance of incorporating exogenous control into the latent ODE for accurate long-term forecasting in this setting.

The only metric in which the ODE-LSTM ranks first is the empirical coverage in the IIV setting (0.443). However, this should not be interpreted as better calibration. The training visualizations in Section B.5 showcase that the model converges to a degenerate solution in which the predicted uncertainty $\sigma(t)$ is inflated and the predicted means converge to the overall mean. The resulting wide intervals mechanically increase coverage without improving point prediction accuracy, as confirmed by the ODE-LSTM’s substantially worse RMSE and CRPS.



(A) No-IIV population.



(B) IIV population.

FIGURE 4.2: Factual glucose trajectory predictions of CLODE (β -kernel) on three random test individuals, stratified by eating profile. Red: ground truth. Purple: predicted mean with 90% prediction interval. Stars: meal events. The dashed vertical line marks the encoding/prediction split at 24 hours.

4.2 Counterfactual Analysis

We evaluate counterfactual performance under three scenarios, each probing a different aspect of the model’s ability to generate counterfactual predictions. In each scenario, the recognition network processes the first 24 hours of observed glucose² to infer $\mathbf{z}_{N-1}$. The model is then queried with a hypothetical meal signal $u'(t)$ over the held-out 24-hour window.

The three scenarios are:

- S1 Dose-response:** same individual, same meal timing, carbohydrate amount varied by $\pm 20\%$ relative to the observed intake;
- S2 Timing shift:** same individual, same carbohydrate amount, meal timing shifted by ± 2 hours;
- S3 No meal:** no meal input, testing whether the model correctly recovers the individual’s circadian baseline $G_b(t)$.

Based on Tables 4.2 and 4.3, we observe that both variants of CLODE outperform the ODE-LSTM baseline across all scenarios and evaluation settings, with the β -Kernel variant consistently achieving the best performance.

Table 4.2: PEHE and CF-RMSE across the three scenarios on the **No-IIV** dataset. Best result per column is **bolded**.

Model	PEHE			CF-RMSE		
	S1	S2	S3	S1	S2	S3
ODE-LSTM	0.078	0.814	0.555	0.723	0.729	0.445
<i>Ours</i>						
CLODE (Dirac)	0.072	0.261	0.533	0.308	0.309	0.584
CLODE (β -kernel)	0.037	0.185	0.158	0.166	0.171	0.131

Table 4.3: PEHE and CF-RMSE across the three scenarios on the **IIV** dataset. Best result per column is **bolded**.

Model	PEHE			CF-RMSE		
	S1	S2	S3	S1	S2	S3
ODE-LSTM	0.080	0.817	0.559	0.884	0.891	0.676
<i>Ours</i>						
CLODE (Dirac)	0.065	0.462	0.332	0.641	0.648	0.549
CLODE (β -kernel)	0.057	0.451	0.322	0.632	0.634	0.533

When examining Figure 4.4, which shows the counterfactual predictions of CLODE (β -kernel) for the average eating profile, we observe that the model is able to capture well

²Under factual meal scenario.

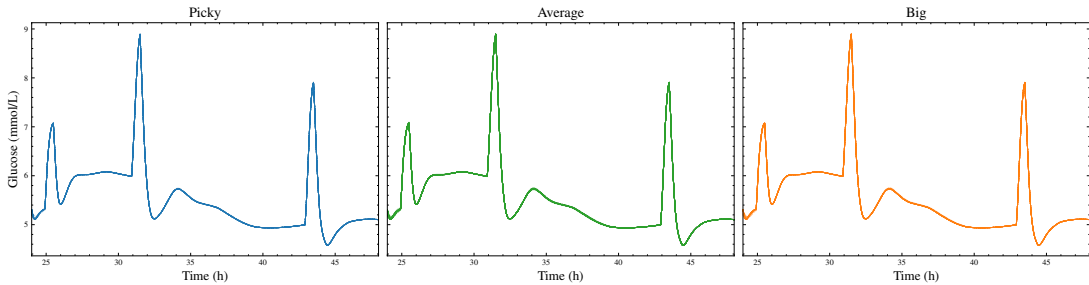
glucose trajectories under the counterfactual meal scenarios, especially in the No-IIV population.

Notably, in Scenario S3 (No meal), it appears that the model is able to correctly recover the individual's circadian baseline. This suggests that CLODE (β -kernel) is effectively disentangling the effects of exogenous inputs from the underlying physiological dynamics.

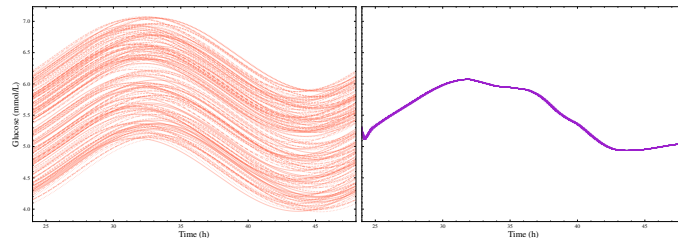
However, when moving to the IIV setting, the bias issues highlighted earlier resurface. Upon closer inspection, it appears that CLODE (β -Kernel) has learned the representation of an average patient.

To investigate whether the model has been able to capture the differences in the physiological parameters shaping the baseline (μ_b, S_G, ϕ), we overlay all counterfactual trajectories for scenario S3 (No meal). We further overlay counterfactual trajectories for a new hypothetical scenario where everyone would eat the same meal, at the same time, allowing to see if the model learns PPGR parameters (N_E, P_1).

In both cases, as can be seen on Figure 4.3, we observe that all test patients have a near-identical trajectory. This means that, for this particular setting, the model has not been able to learn an initial condition $\mathbf{z}_{N-1}$ that capture inter-individual variability. Figure 4.5 however shows that the inferred latent codes exhibit variance across individuals, suggesting that the encoder has not collapsed to the prior. The failure to produce individualized trajectories might therefore likely stems from a different source: the learned vector field f_θ may be insufficiently sensitive to variations in $\mathbf{z}_{N-1}$, the encoding window may be too short to capture discriminative physiological signal, or the encoder may lack the expressivity to project individuals into well-separated regions of the latent space. Potential remedies are discussed in Chapter 5.

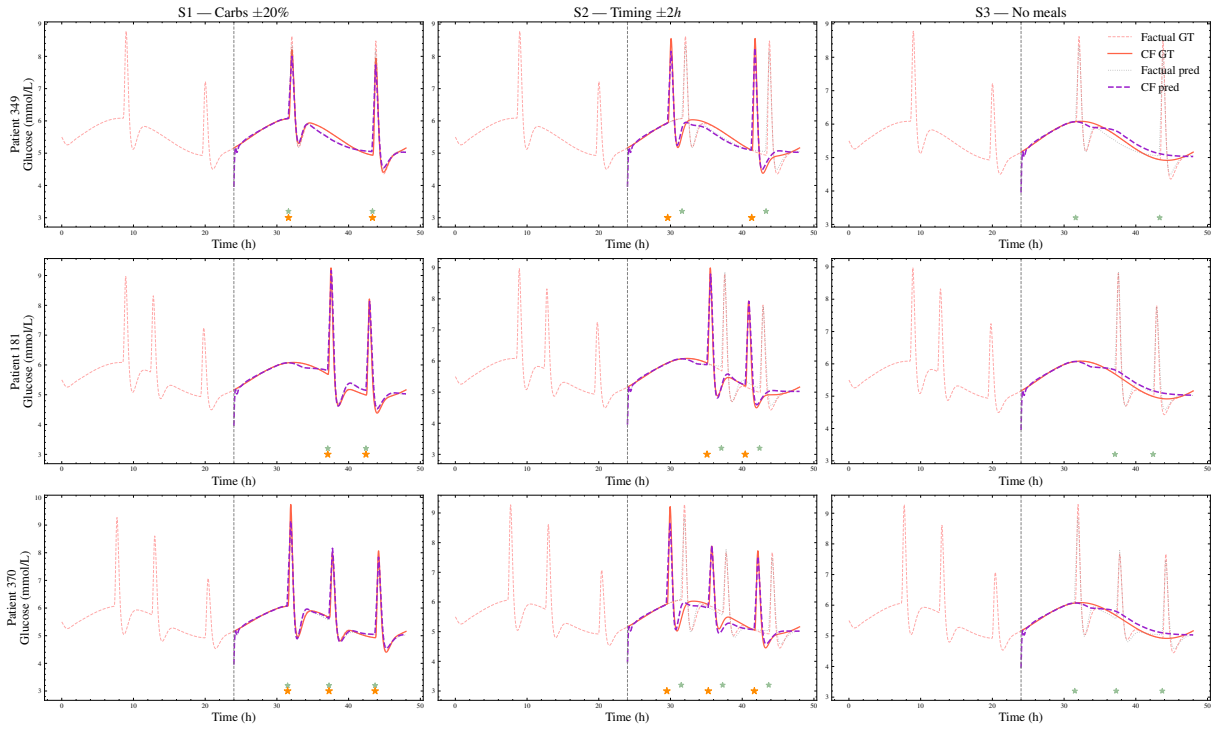


(A) Identical Diet Overlay. **Left** : Counterfactual trajectories for "picky" profile **Middle** : Counterfactual trajectories for "average" profile **Right** : Counterfactual trajectories for "big" profile.

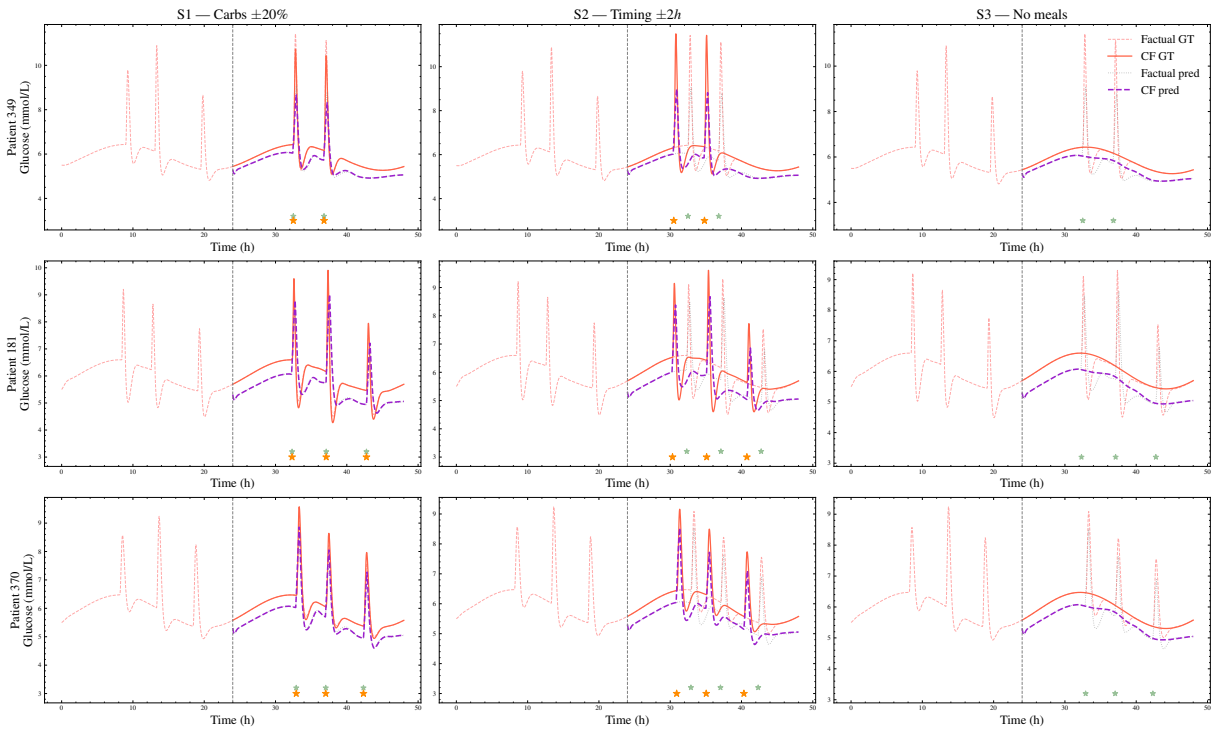


(B) Scenario S3 Overlay. **Left** : Test set individual's real circadian baselines. **Right** : Counterfactual trajectories under scenario S3 (No meal).

FIGURE 4.3: Overlay of counterfactual trajectories for Scenario S3 (No meal) and identical diet scenario.



(A) No-IIV population.



(B) IIV population.

FIGURE 4.4: Counter-Factual glucose trajectory predictions of CLODE (β -kernel) on three random test individuals, for the average eating profile. Red: Counter-Factual Ground Truth. Dashed Red : Factual Ground Truth. Purple: predicted mean with 90% prediction interval. Dashed Gray : Factual Predictions. Stars: meal events. The dashed vertical line marks the encoding/prediction split at 24 hours.

4.3 Latent Space Validation

As established in Section 4.2, all models produce near-identical trajectories under a standardized scenarios, suggesting that the inferred $\mathbf{z}_{N-1}$ captures limited IIV, though the inferred latent codes exhibit variance across individuals (Figure 4.5).

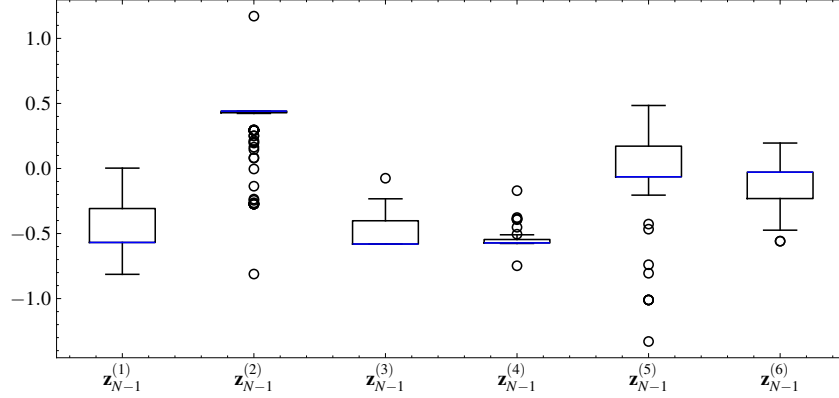


FIGURE 4.5: Boxplot of the inferred $\mathbf{z}_{N-1}$, dimension-wise.

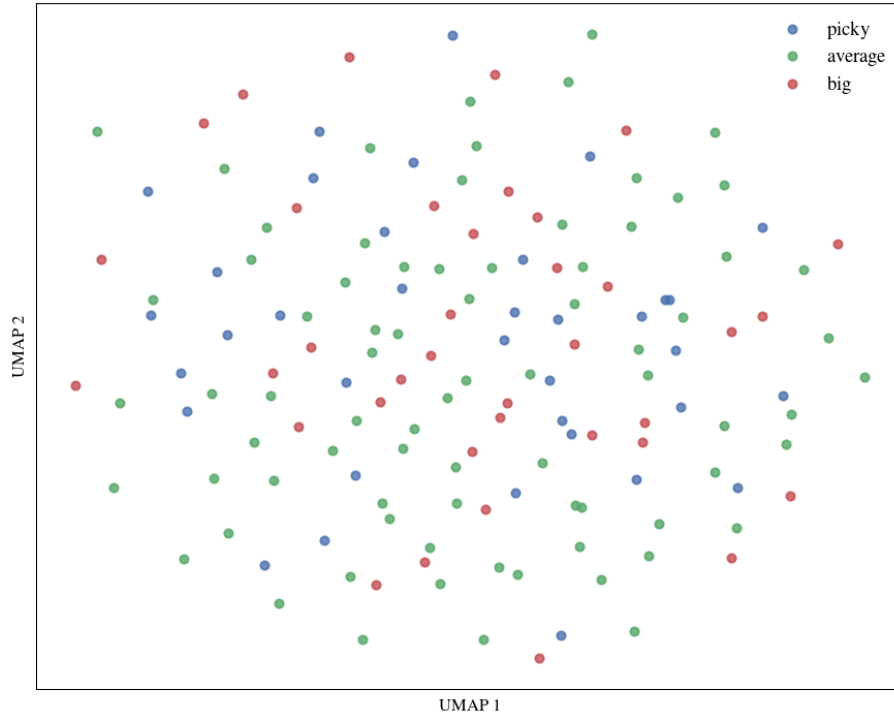
We now investigate more deeply the structure of the latent initial condition $\mathbf{z}_{N-1}$.

While the results displayed in Table 4.4 show that the RV coefficient and Spearman correlations are generally low across all models, we observe that the β -Kernel variant of CLODE achieves the highest RV coefficient, indicating a stronger global alignment between the latent space and the true physiological parameter space. Moreover, it also achieves the highest average Spearman correlation, suggesting that individual latent dimensions are more strongly associated with specific physiological parameters compared to the other models.

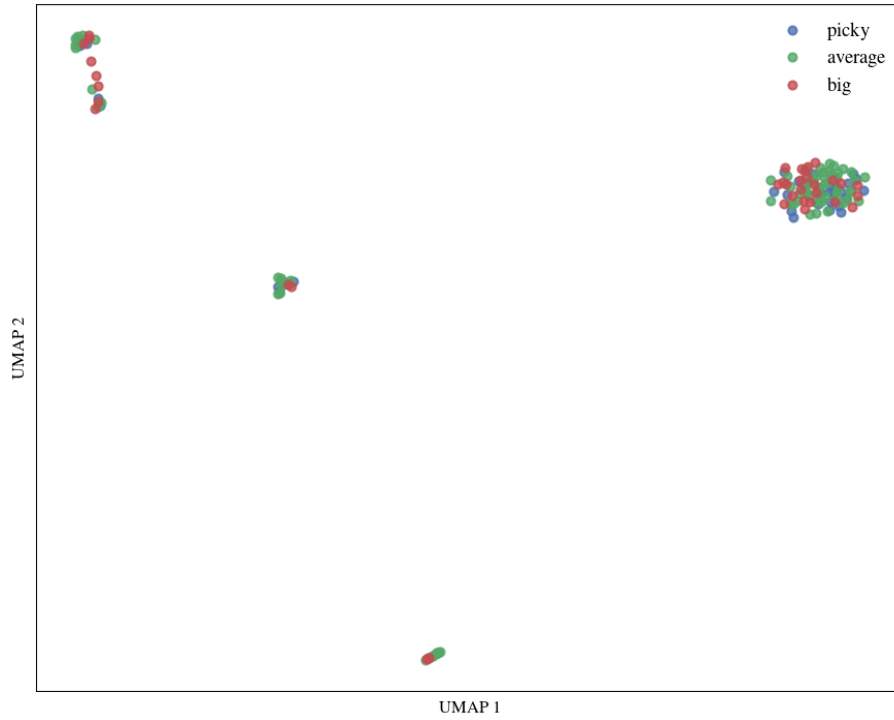
Furthermore, Figure 4.6 support the fact that, even though the latent space of CLODE (β -kernel) is not strongly correlated with the true physiological parameters, it produces some separation between individuals based on some underlying (latent) structure. In the No-IIV setting, we see that the model does not separate individuals, which was expected since all individuals are physiologically identical. In the IIV setting, on the other hand, we see that the model separates the individuals in the latent space, and that this separation is not driven by the eating profiles.

Table 4.4: Latent space validation metrics. Best result per column is **bolded**.

Model	RV Coefficient	Spearman Corr. (max)	Spearman Corr. (mean)
Latent ODE-RNN	0.0013	0.247	0.073
<i>Ours</i>			
CLODE (Dirac)	0.0169	0.124	0.059
CLODE (β -kernel)	0.0226	0.246	0.113



(A) No IIV.



(B) IIV.

FIGURE 4.6: UMAP 2D projection of the latent space $\mathbf{z}_{N-1}$ for CLODE (β -kernel) per eating profile.

²Additional visualizations, including UMAP projections coloured by physiological parameters and Spearman correlation heatmaps, are provided in Appendix B.4.

Conclusion

How a given individual will respond to a meal – and how that response would differ under an alternative dietary scenario – is a central challenge for personalized nutrition and precision medicine. Addressing it requires a model that can handle continuous-time physiological dynamics, incorporate discrete meal perturbations, and generalize across individuals.

In this work, we demonstrate that CLODE (Controlled Latent ODE-RNN), a new Deep State-Space model whose latent dynamics are controlled by a continuous-time meal signal, constitutes a promising step toward such a model because it can (i) support accurate factual and counterfactual predictions in a physiologically homogeneous population; and (ii) recover an average physiological profile in a heterogeneous population, while showing encouraging early evidence of individual-level representation learning abilities.

CLODE extends the Latent ODE-RNN of [Rubanova et al. \(2019\)](#), which provides no explicit mechanism for exogenous inputs, by incorporating punctual meal records as an explicit continuous-time control signal into the latent dynamics. Motivated by applications involving explicit control inputs in pharmacokinetics ([Lu et al., 2021](#)) and system identification ([Brunton et al., 2016](#)), this design choice incentivizes the latent space to capture genuine endogenous physiological dynamics rather than conflating them with exogenous dietary inputs.

To represent meal records as a continuous-time signal, we introduced the β -Kernel, a scaled Beta distribution spreading the recorded carbohydrate amount over a physiologically plausible ingestion window, while preserving meal mass. This representation is more faithful to the true ingestion process, which takes time, and further allows for a smoother numerical integration.

Ultimately, the question underlying this master thesis was whether CLODE (β -Kernel) could simultaneously support three complementary objectives experimentally: (i) physiologically grounded representation learning, (ii) factual prediction and (iii) counterfactual predictions, with the broader ambition of enabling continuous-time individual-level blood glucose modeling.

To address this question, we designed two *in silico* populations of 500 individuals, generated via a physiologically inspired glucose-insulin simulator. The first population, physiologically homogeneous, contained individuals sharing identical physiological parameters but varying in their eating profiles. The second population introduced inter-individual variability (IIV) by randomizing key physiological parameters across individuals. Each individual was observed over a 48-hour window. The first 24 hours served as the encoding window, and the second was used to evaluate the model’s ability to extrapolate the subsequent 24 hours conditioned on a prospective meal plan.

Performance was assessed using three sets of metrics, each mirroring one of the outlined objectives: RMSE, CRPS and Coverage (90%) for factual forecasting accuracy; PEHE and CF-RMSE (across three dietary scenarios) for counterfactual validity; and RV coefficient and Spearman rank correlation for latent space structure. Moreover, CLODE was compared against two representative baselines from the neural differential equation literature: the ODE-LSTM variant of [Fitzgerald et al. \(2023\)](#), an autoregressive model handling exogenous inputs without a latent representation, and the Latent ODE-RNN of [Rubanova et al. \(2019\)](#), which provides a latent representation but no way of incorporating control.

The two populations and two baselines served complementary diagnostic purposes. In the No-IIV setting, where all individuals are physiologically identical, a well-designed model should not discriminate individuals in the latent space, intended to represent the latent physiological state. Comparing CLODE against the Latent ODE-RNN in this setting therefore probed whether explicitly including meal inputs in the latent dynamics prevents the encoder from conflating exogenous meal effects with endogenous physiological variability. We also compared CLODE (β -Kernel) against a Dirac impulse variant of CLODE, where meals were fed as instantaneous point events rather than distributed over an ingestion window, to assess whether a physiologically motivated continuous meal representation yields better predictions than the naive discrete representation of the raw dietary logs.

On factual forecasting, CLODE (β -Kernel) substantially outperformed both baselines across both populations. It achieved an RMSE of 0.169 and a CRPS of 0.095 in the No-IIV setting, compared to 0.583 and 0.248 for the closest baseline (Latent ODE-RNN), and 0.666 and 0.431 against 0.824 and 0.497 in the IIV setting. Furthermore, the β -Kernel variant consistently outperformed the Dirac impulse variant of CLODE across both settings, confirming that distributing meal intake over a physiologically plausible ingestion window yields more accurate predictions than unrealistically treating meals as instantaneous events. Counterfactual performance followed the same pattern, with PEHE ranging from 0.037 to 0.185 across the three dietary scenarios in the No-IIV setting, and from 0.057 to 0.451 in the IIV setting. Yet, despite ranking best in both simulation settings, CLODE showed systematic upward or downward biases for individuals in the IIV setting.

That difficulty of accurately predicting some individuals in the IIV setting is tightly linked to the model’s learned latent representation. Analyzing the latter revealed a nuanced picture. In the No-IIV setting, as expected, CLODE produced no systematic clustering of individuals in the latent space based on (exogenous) eating profiles. By contrast, the Latent ODE-RNN did, suggesting that treating exogenous and endogenous effects separately is crucial in order to learn a more faithful representation. In the IIV setting, CLODE (β -

Kernel) achieved the highest RV coefficient (0.0226 *vs* 0.0013 for the Latent ODE-RNN) and the highest mean Spearman correlation (0.113 *vs* 0.073), indicating a stronger global alignment between the inferred latent space and the true physiological parameter space. Yet absolute values remained low, and the model produced near-identical trajectories under standardized dietary scenarios. This suggests that the encoder – operating on a single 24-hour observation window – is insufficiently sensitive to variations in the dynamics of the observed glucose. One could also argue that the latent dynamics – assumed to be deterministic – are not flexible enough to capture the full complexity of the underlying physiological process.

Nevertheless, despite these limitations, CLODE remained the only model capable of recovering a coherent average physiological dynamic in a heterogeneous population. This is an encouraging result in itself: it demonstrates that the model has internalized a physiologically plausible glucose response to meals, even in the presence of inter-individual variability. This represents a promising step toward personalized glucose modeling.

Taken together, these results provide a partial positive answer to our research question. CLODE demonstrates that incorporating meal intake as an explicit exogenous control signal supports long-horizon factual and counterfactual glucose trajectory prediction. However, the goal of learning a *physiologically meaningful* latent representation of individual glycemic dynamics remains only partially achieved and would be worth further investigating.

These findings also suggest two main future directions. On the data side, conditioning on longer observation windows or incorporating static covariates with known discriminative power (*e.g.* sex or BMI) could help the encoder better separate physiological profiles at a low architectural cost. On the architecture side, more expressive encoder designs – such as replacing the ODE-RNN with an ODE-LSTM (Lechner and Hasani, 2020; Coelho et al., 2024), exploring other prior families (*e.g.* Laplacian or Normalizing Flows), or incorporating physiological inductive biases in the spirit of Zou et al. (2024) – represent interesting future research directions.

Looking ahead, a broader challenge remains evaluating CLODE on real free-living CGM data, where the true complexity of inter-individual variability, irregular sampling, and noisy dietary logs will provide a more demanding test and better inform subsequent architectural refinements.

Beyond CLODE and glucose, we believe that it would be worth further developing such controlled latent models, which could benefit any system where individual-specific continuous-time dynamics are perturbed by discrete exogenous inputs and observed through noisy, irregularly sampled measurements.

Bibliography

- Abril-Pla, O., Andreani, V., Carroll, C., Dong, L., Fonnesbeck, C. J., Kochurov, M., Kumar, R., Lao, J., Luhmann, C. C., Martin, O. A., Osthege, M., Vieira, R., Wiecki, T., and Zinkov, R. (2023). PyMC: A modern and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, 9(e1516).
- Afsaneh, E., Sharifdini, A., Ghazzaghi, H., and Ghobadi, M. Z. (2022). Recent applications of machine learning and deep learning models in the prediction, diagnosis, and management of diabetes: A comprehensive review. *Diabetology & Metabolic Syndrome*, 14(1):196.
- Bender, C., Vestergaard, P., and Cichosz, S. L. (2025). The History, Evolution and Future of Continuous Glucose Monitoring (CGM). *Diabetology*, 6(3):17.
- Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.
- Bergman, R. N. (2021). Origins and History of the Minimal Model of Glucose Regulation. *Frontiers in Endocrinology*, 11:583016.
- Bergman, R. N., Ider, Y. Z., Bowden, C. R., and Cobelli, C. (1979). Quantitative estimation of insulin sensitivity. *American Journal of Physiology*, 236(6):E667–E677.
- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.
- Box, G. E. P. and Draper, N. R. (1987). *Empirical Model-Building and Response Surfaces*. John Wiley & Sons, New York.
- Brunkhorst, G. (2022). ODE Lotka-Volterra With Bayesian Inference in Multiple Ways. In PyMC Team, editor, *PyMC Examples*.
- Brunton, S. L., Budišić, M., Kaiser, E., and Kutz, J. N. (2021). Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*.
- Brunton, S. L. and Kutz, J. N. (2022). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2 edition.

- Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Sparse Identification of Nonlinear Dynamics with Control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715.
- Champion, K., Lusch, B., Kutz, J. N., and Brunton, S. L. (2019). Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451.
- Chapfuwa, P., Rose, S., Carin, L., Meeds, E., and Henao, R. (2022). Capturing actionable dynamics with structured latent ordinary differential equations. *Proceedings of Machine Learning Research*, 180:286–295.
- Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1):6085.
- Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.
- Chen, S. X. (2000). Beta kernel smoothers for regression curves. *Statistica Sinica*, 10(1):73–91.
- Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.
- Cisuelo, O., Stokes, K., Oronti, I. B., Haleem, M. S., Barber, T. M., Weickert, M. O., Pecchia, L., and Hattersley, J. (2023). Development of an artificial intelligence system to identify hypoglycaemia via ecg in adults with type 1 diabetes: protocol for data collection under controlled and free-living conditions. *BMJ Open*, 13(3):e067899.
- Coelho, C., P. Costa, M. F., and Ferrás, L. (2024). Enhancing continuous time series modelling with a latent ODE-LSTM approach. *Applied Mathematics and Computation*, 475:128727.
- Conti, P., Kneifl, J., Manzoni, A., Frangi, A., Fehr, J., Brunton, S. L., and Kutz, J. N. (2026). VENI, VINDy, VICI: A generative reduced-order modeling framework with uncertainty quantification. *Neural Networks*, 198:108543.
- Dahleh, M. A., Dahleh, M., and Verghese, G. (2011). *Lectures on Dynamic Systems and Control*. MIT OpenCourseWare.
- De Brouwer, E. (2023). *Machine Learning Modeling of Time-Dependent Patient Trajectories*. PhD thesis.
- De Brouwer, E., Hernandez, J. G., and Hyland, S. (2022). Predicting the impact of treatments over time with uncertainty aware neural differential equations.
- De Brouwer, E., Simm, J., Arany, A., and Moreau, Y. (2019). Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32.
- Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural odes. *Advances in neural information processing systems*, 32.
- Fasel, U., Kutz, J. N., Brunton, B. W., and Brunton, S. L. (2022). Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478(2260):20210904.

- Fitzgerald, O., Perez-Concha, O., Gallego-Luxan, B., Metke-Jimenez, A., Rudd, L., and Jorm, L. (2023). Continuous time recurrent neural networks: Overview and benchmarking at forecasting blood glucose in the intensive care unit. *Journal of Biomedical Informatics*, 146:104498.
- Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., and Carin, L. (2019). Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1301–1310.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.
- Gutierrez-Osuna, R., Kerr, D., Mortazavi, B., and Das, A. (2025). CGMacros: a scientific dataset for personalized nutrition and diet monitoring. *PhysioNet*. Version 1.0.0.
- Gwak, D., Sim, G., Poli, M., Massaroli, S., Choo, J., and Choi, E. (2020). Neural ordinary differential equations for intervention modeling. *arXiv preprint arXiv:2010.08304*.
- Ha, D., Dai, A. M., and Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.
- Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.
- Joedicke, D., Parra, D., Kronberger, G., and Winkler, S. M. (2022). Identifying differential equations to predict blood glucose using sparse identification of nonlinear systems. *arXiv preprint arXiv:2209.13852*.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 3149–3157, Red Hook, NY, USA. Curran Associates Inc.
- Kidger, P. (2022). *On Neural Differential Equations*. PhD thesis, arXiv.
- Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020). Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707.
- Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization.
- Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). Semi-Supervised Learning with Deep Generative Models.
- Kingma, D. P. and Welling, M. (2019). An Introduction to Variational Autoencoders.

- Foundations and Trends® in Machine Learning*, 12(4):307–392.
- Lechner, M. and Hasani, R. (2020). Learning long-term dependencies in irregularly-sampled time series. *arXiv preprint arXiv:2006.04418*.
- Li, Q., Lin, T., and Shen, Z. (2022). Deep learning via dynamical systems: An approximation perspective. *Journal of the European Mathematical Society*, 25(5):1671–1709.
- Lin, J. and Michailidis, G. (2024). Deep Learning-based Approaches for State Space Models: A Selective Review.
- Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4114–4124. PMLR.
- Lotka, A. J. (1926). Elements of physical biology. *Science Progress in the Twentieth Century (1919-1933)*, 21(82):341–343.
- Lu, J., Deng, K., Zhang, X., Liu, G., and Guan, Y. (2021). Neural-ode for pharmacokinetics modeling and its advantage to alternative machine learning models in predicting new dosing regimens. *Iscience*, 24(7).
- Malpica-Morales, A., Kalliadasis, S., Malliaras, G. G., and Güemes, A. (2025). Data-driven prediction of blood glucose dynamics from vagus nerve recordings using neural controlled differential equations. *Machine Learning: Science and Technology*, 6(3):035062.
- Man, C. D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., and Cobelli, C. (2014). The UVA/PADOVA type 1 diabetes simulator: New features. *Journal of Diabetes Science and Technology*, 8(1):26–34.
- Massaroli, S., Poli, M., Park, J., Yamashita, A., and Asama, H. (2020). Dissecting neural odes. *Advances in neural information processing systems*, 33:3952–3963.
- McInnes, L., Healy, J., and Melville, J. (2020). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.
- Morrill, J., Kidger, P., Yang, L., and Lyons, T. (2021a). Neural controlled differential equations for online prediction tasks. *arXiv preprint arXiv:2106.11028*.
- Morrill, J., Salvi, C., Kidger, P., and Foster, J. (2021b). Neural Rough Differential Equations for Long Time Series. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7829–7838. PMLR.
- Oh, Y., Kam, S., Lee, J., Lim, D.-Y., Kim, S., and Bui, A. (2025). Comprehensive Review of Neural Differential Equations for Time Series Analysis.
- O’Keefe, J. H. and Bell, D. S. (2007). Postprandial hyperglycemia/hyperlipidemia (postprandial dysmetabolism) is a cardiovascular risk factor. *The American Journal of Cardiology*, 100(5):899–904.
- Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.

- Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31.
- Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., and Edelman, A. (2020). Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*.
- Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.
- Robert, P. and Escoufier, Y. (1976). A unifying tool for linear multivariate statistical methods: The rv-coefficient. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 25(3):257–265.
- Rubanova, Y., Chen, R. T., and Duvenaud, D. K. (2019). Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- Särkkä, S. (2016). Lecture 8: Bayesian estimation of parameters in state space models. Lecture notes, March 30, 2016.
- Savitzky, A. and Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639.
- Sirlanci, M., Levine, M. E., Low Wang, C. C., Albers, D. J., and Stuart, A. M. (2023). A simple modeling framework for prediction in the human glucose–insulin system. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(7):073150.
- Song, J., Oh, T. J., and Song, Y. (2023). Individual Postprandial Glycemic Responses to Meal Types by Different Carbohydrate Levels and Their Associations with Glycemic Variability Using Continuous Glucose Monitoring. *Nutrients*, 15(16):3571.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.
- Van Herpe, T., Pluymers, B., Espinoza, M., Berghe, G., and De Moor, B. (2006). A minimal model for glycemia control in critically ill patients. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 1:5432–5.
- Villaverde, A. F. (2019). Observability and structural identifiability of nonlinear biological systems. *Complexity*, 2019:1–20.
- Wang, Q., Molenaar, P., Harsh, S., Freeman, K., Xie, J., Gold, C., Rovine, M., and Ulbrecht, J. (2014). Personalized State-space Modeling of Glucose Dynamics for Type 1 Diabetes Using Continuously Monitored Glucose, Insulin Dose, and Meal Intake: An Extended Kalman Filter Approach. *Journal of Diabetes Science and Technology*, 8(2):331–345.
- Zeevi, D., Korem, T., Zmora, N., Israeli, D., Rothschild, D., Weinberger, A., Ben-Yacov, O., Lador, D., Avnit-Sagi, T., Lotan-Pompan, M., et al. (2015). Personalized nutrition

by prediction of glycemic responses. *Cell*, 163(5):1079–1094.

Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., and Mahoney, M. (2019). Anodev2: A coupled neural ode evolution framework. *arXiv preprint arXiv:1906.04596*.

Zou, B. J., Levine, M. E., Zaharieva, D. P., Johari, R., and Fox, E. B. (2024). Hybrid² Neural ODE Causal Modeling and an Application to Glycemic Response.

Additional Mathematical Material

A.1 Universal Approximation properties of NODEs

A.1.1 Flow Map

The integral form (2.3) suggests a natural operator. A map that takes any initial condition $\mathbf{z}_0$ to the state reached at time t . This operator, called the *flow map*, provides a unified perspective on continuous- and discrete-time dynamics.

In continuous time, it corresponds to integrating f from t_0 to t . In discrete time, sampling the trajectory at regular intervals $t_k = t_0 + k\Delta t$ yields $\mathbf{z}_{k+1} = \varphi_{\Delta t}(\mathbf{z}_k)$, where the flow map acts as a *discrete-time propagator* (Brunton and Kutz, 2022).

Definition A.1.1 (Flow map). The *flow map* $\varphi_t : \mathbb{R}^s \rightarrow \mathbb{R}^s$ associated to f is the operator that maps any initial condition $\mathbf{z}_0$ to the state reached at time t :

$$\varphi_t(\mathbf{z}_0) := \mathbf{z}(t) = \mathbf{z}_0 + \int_{t_0}^t f(\mathbf{z}(\tau)) d\tau \quad (\text{A.1})$$

A.1.2 The flow maps of an NODE is an homeomorphism

A flow map φ_t (Definition A.1.1) maps any initial condition $\mathbf{z}_0$ to the state $\mathbf{z}(t)$ by integrating f_θ from t_0 to t .

In a NODE, $\varphi_T(\mathbf{z}_0)$ thus defines the feature mapping from input (0) to output (T). The question is: what class of functions can this map represent? The answer to that question has already been explored by Dupont et al. (2019) and hinges upon A.1.1 :

Proposition A.1.1. *The flow $\varphi_t : \mathbb{R}^s \rightarrow \mathbb{R}^s$ of a Neural ODE is a homeomorphism for all $t \in [0, T]$. Consequently, the feature mapping $\mathbf{z}(T) = \varphi_T(\mathbf{z}_0)$ preserves the topology of the input space.*

Definition A.1.2 (Homeomorphism). A function $\varphi : \mathbb{R}^s \rightarrow \mathbb{R}^s$ is a *homeomorphism* if it is a continuous bijection whose inverse φ^{-1} is also continuous.

The key consequence of A.1.1 is that NODEs can only continuously deform the input space, *e.g.* they cannot tear apart connected regions, create holes, or merge distinct components, which limits the class of function NODEs can represent. Fortunately, this can be solved by resorting to Augmentation.

A.2 Inductive Bias and UDEs

A common moto in statistical learning is that "*all models are wrong, but some are useful*" (Box and Draper, 1987). This saying is here to reminds us that no model can capture reality in its full complexity. Every modelling choice involves simplifications, whether in the functional form of the dynamics or the assumed noise structure. The question is therefore not whether to make such assumptions, but which ones to make. This choice is precisely inductive bias.

Following Kidger (2022), one can distinguish two types of inductive bias : (i) *Hard* inductive bias and (ii) *Soft* inductive bias.

A *hard* inductive bias encodes domain knowledge directly into the model structure, making deviations from the assumed form impossible by construction. This is the case of mechanistic models (Section 2.1.2), where the functional form of f is prescribed, or SINDy (Section 2.2.1), where a library of functions is fixed *a priori*. In both case, the model cannot express dynamics outside the prescribed family. ANODEs, on the other hand, do not fall in this category, as they can approximate any dynamics with arbitrary precision.

A *soft* inductive bias instead penalizes or encourages certain solutions without, typically via a regularization term added to the loss. In a sense, ANODEs belong to that category as f_θ is unconstrained, and the only bias comes from the architecture and optimization procedure.

Acknowledging the limitation of architectures with prescribed f , one might hybridise a mechanistic model f with a flexible neural correction term f_θ , yielding Universal Differential Equations (Kidger, 2022; Rackauckas et al., 2020) :

$$\frac{d\mathbf{z}}{dt}(t) = f(\mathbf{z}(t); \boldsymbol{\beta}) + f_\theta(\mathbf{z}(t)) \quad (\text{A.2})$$

We keep a hard inductive bias, but we now have a way to account for unmodelled dynamics.

A.3 GRU-ODE-Bayes

The ODE-RNN and ODE-LSTM define deterministic continuous-time dynamics for the hidden state. De Brouwer et al. (2019) take a different starting point, motivated by the modelisation of sporadically observed clinical time series.

Specifically, the authors assume that the observations $\mathbf{x}_k$ are noisy realisations of a p -dimensional stochastic process $X(t)$ governed by an unknown SDE of the form:

$$dX(t) = \boldsymbol{\mu}(X(t)) dt + \boldsymbol{\sigma}(X(t)) d\mathbf{W}(t) \quad (\text{A.3})$$

The distribution of $X(t)$ then evolves according to the corresponding Fokker-Planck equation. Under Gaussianity assumption, the goal is to learn the temporal evolution of the parameters of that distribution : $\boldsymbol{\mu}_X(t)$ and $\boldsymbol{\Sigma}_X(t)$ ¹.

¹This Gaussian assumption is exact for the Ornstein-Uhlenbeck process, whose Fokker-Planck PDF is Gaussian by construction, and constitutes an approximation for general SDEs.

De Brouwer et al. (2019) parameterise these functions by using a shallow MLP applied to some hidden state $\mathbf{h}(t)$:

$$\boldsymbol{\mu}_X(t), \log \boldsymbol{\sigma}_X^2(t) = \text{MLP}_{\text{obs}}(\mathbf{h}(t)) \quad (\text{A.4})$$

where $\log \boldsymbol{\sigma}_X^2(t) \in \mathbb{R}^p$ are the per-dimension log-variances, implying a diagonal covariance structure $\boldsymbol{\Sigma}_X(t) = \text{diag}(\sigma_1^2(t), \dots, \sigma_p^2(t))$.

The resulting per-dimension predictive distribution is:

$$X_j(t) \mid \mathbf{h}(t) \sim \mathcal{N}(\mu_j(t), \sigma_j^2(t)), \quad j = 1, \dots, p \quad (\text{A.5})$$

The hidden state $\mathbf{h}(t)$ evolves continuously between observations according to the GRU-ODE dynamics:

$$\frac{d\mathbf{h}}{dt}(t) = (1 - \mathbf{z}^{(u)}(t)) \odot (\tilde{\mathbf{h}}(t) - \mathbf{h}(t)) \quad (\text{A.6})$$

where $\mathbf{z}^{(u)}(t)$ and $\tilde{\mathbf{h}}(t)$ are the update gate and candidate state of the GRU cell, now treated as continuous functions of time.

Upon reception of an observation $\mathbf{x}_k$ at time t_k , the hidden state is updated via a discrete GRU-Bayes step that incorporates the new information through an observation mask $\mathbf{m}_k$ handling partial observability:

$$\mathbf{h}(t_k^+) = \text{GRU-Bayes}(\mathbf{h}(t_k^-), \mathbf{x}_k, \mathbf{m}_k) \quad (\text{A.7})$$

This prediction-update scheme is analogous to the Kalman filter, with the GRU-ODE playing the role of the prediction step and GRU-Bayes the filtering step. We refer to the original paper for details regarding the GRU-Bayes step.

Training objective The training objective is made of two terms :

$$\mathcal{L} = \mathcal{L}_{\text{pre}} + \lambda \mathcal{L}_{\text{post}} \quad (\text{A.8})$$

where $\lambda \geq 0$ is a mixing hyper-parameter.

$\mathcal{L}_{\text{pre}}$ is the masked gaussian negative log-likelihood, using the predictive distribution parameterised by $\mathbf{h}(t_k^-)$:

$$\mathcal{L}_{\text{pre}} = - \sum_{j=1}^p m_j \log p(x_j \mid \text{MLP}_{\text{obs}}(\mathbf{h}(t_k^-))) \quad (\text{A.9})$$

$$= \frac{1}{2} \sum_{j=1}^p m_j \left(\frac{(x_{k,j} - \mu_j)^2}{\sigma_j^2} + \log \sigma_j^2 \right) \quad (\text{A.10})$$

$\mathcal{L}_{\text{post}}$ penalises the divergence between the post-update distribution p_{post} , parameterized by $\mathbf{h}(t_k^+)$, and a tight Gaussian centred on the observation, with fixed noise standard deviation σ_{obs} :

$$\mathcal{L}_{\text{post}} = \sum_{j=1}^p m_j D_{\text{KL}} \left(\mathcal{N}(\mu_{\text{post},j}, \sigma_{\text{post},j}^2) \parallel \mathcal{N}(x_{k,j}, \sigma_{\text{obs}}^2) \right) \quad (\text{A.11})$$

where $(\mu_{\text{post},j}, \sigma_{\text{post},j}^2)$ are obtained from $\text{MLP}_{\text{obs}}(\mathbf{h}(t_k^+))$ and σ_{obs} is a fixed hyperparameter. This term forces the post-update distribution to concentrate around the observed value with a variance of order σ_{obs}^2 .

Additional Experimental Material

B.1 Architectural Details

Baselines and the main architecture have been implemented mainly in Pytorch, relying on the `torchdiffeq` (for CLODE and Latent ODE-RNN) and `torchctrnn` (for ODE-LSTM). All the code, data, and main experimental results are made publicly available at : <https://github.com/LDPuser/CLODE-blood-glucose-modeling>.

B.1.1 CLODE

Recognition Network (ODE-RNN Encoder). The ODE vector field is a 3-layer MLP with hidden width 32 and Tanh activations. The GRU cell has hidden dimension 32. The translation network is a 2-layer MLP ($32 \rightarrow 32 \rightarrow 2s$) with Tanh activation, outputting the mean and log-variance. The encoder input is the concatenation of the glucose observation and its binary observation mask.

Latent ODE. The latent vector field is a 4-layer MLP with hidden width 64 and Tanh activations, taking as input the the latent state and the scalar control input.

Emission Network. The decoder is a 2-layer MLP ($s \rightarrow 32 \rightarrow 1$) with SELU activation, outputting the predicted glucose mean. Observation noise is fixed to 0.28.

B.1.2 Latent ODE-RNN

The Latent ODE-RNN shares the same architectural components as CLODE, with two differences.

On the one hand, the encoder input is a four-dimensional vector $[G, c, m_G, m_c]$ at each timestep, where G and c are the glucose and carbohydrate values (zero-padded where unobserved), and $m_G, m_c \in \{0, 1\}$ are binary observation masks. All other components (ODE vector field, GRU cell, translation network, hidden dimensions, activations) are identical to CLODE.

On the other hand, the latent vector field does not take any control input.

B.1.3 ODE-LSTM

The ODE-LSTM closely follows the implementation of [Fitzgerald et al. \(2023\)](#), available [here](#).

Preprocessing Network. A 2-layer MLP ($1 \rightarrow 30 \rightarrow 32$) with Tanh activations pre-processes each glucose observation before the LSTM update.

ODE Vector Field. The inter-observation vector fields is a 2-layer MLP ($33 \rightarrow 50 \rightarrow 32$) with Tanh activations, taking as input the concatenation of the hidden state and the scalar meal signal.

ODE-LSTM Cell. The hidden state is updated via an ODE-LSTM cell from [torchctrnn](#).

Output Network. Two independent heads map the hidden state to the predicted mean and standard deviation. Each head consists of a linear layer ($32 \rightarrow 32$), dropout ($p = 0.2$), an activation, and a final linear layer ($32 \rightarrow 1$). The mean head uses ReLU while the variance head uses Tanh, with the output passed through an exponential to ensure positivity.

B.2 Extrapolation and Representation Learning

In this appendix, we provide additional experimental results related to the extrapolation capabilities of the models and their ability to learn meaningful representations. The model presented in the main text is **bolded**, the closest competitors is underlined.

B.2.1 ODE-LSTM

Table B.1: No-IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters	Validation Metrics		Test Metrics		
λ_{mix}	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}
0.0001	226	<u>10445.02</u>	0.7092	0.3769	0.5844
0.001	300	10306.33	0.7428	0.3876	0.6147
0.01	250	10646.57	0.7124	0.3798	0.5800

Table B.2: IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters	Validation Metrics		Test Metrics		
λ_{mix}	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}
0.0001	256	<u>12537.20</u>	0.9055	0.5316	0.4848
0.001	241	12515.29	0.9117	0.5498	0.4435
0.01	277	12774.71	0.9124	0.5481	0.4470

B.2.2 Latent ODE-RNN

Table B.3: No-IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters			Validation Metrics		Test Metrics		
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}
4	0.5	0.0	299	881.33	0.723	0.386	0.560
4	0.5	4.3	144	870.84	0.704	0.377	0.562
4	1.0	0.0	288	885.28	0.709	0.383	0.567
4	1.0	4.3	40	599.74	0.583	0.248	0.838
4	2.0	0.0	67	<u>767.44</u>	0.693	0.295	0.817
4	2.0	4.3	215	866.37	0.702	0.370	0.583
4	CA*	0.0	24	1132.39	0.825	0.498	0.364
4	CA*	4.3	8	882.14	0.709	0.384	0.549

* CA denotes Cost Annealing.

Table B.4: IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters			Validation Metrics		Test Metrics				
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}	RV Coefficient	Spearman (max/mean)
3	0.5	0.0	287	1381.87	0.933	0.571	0.418	0.0000	na / na
3	0.5	4.3	176	1360.28	0.924	0.566	0.417	0.0083	0.145 / 0.081
3	1.0	0.0	221	1275.84	0.893	0.539	0.420	0.0015	0.132 / 0.084
3	1.0	4.3	52	1370.15	0.927	0.567	0.422	0.0057	0.186 / 0.077
3	2.0	0.0	287	1401.24	0.938	0.574	0.417	0.0000	na / na
3	2.0	4.3	106	1360.24	0.925	0.566	0.418	0.0009	0.297 / 0.069
3	CA*	0.0	300	1410.38	0.938	0.580	0.406	0.0069	0.317 / 0.129
3	CA*	4.3	166	1360.30	0.925	0.566	0.418	0.0013	0.125 / 0.059
4	0.5	0.0	55	1259.02	0.905	0.540	0.421	0.0000	na / na
4	0.5	4.3	228	1360.72	0.926	0.566	0.420	0.0043	0.101 / 0.049
4	1.0	0.0	282	1122.33	0.849	0.506	0.416	0.0259	0.728 / 0.199
4	1.0	4.3	249	1360.34	0.924	0.567	0.416	0.0000	na / na
4	2.0	0.0	268	1374.28	0.944	0.575	0.418	0.0000	na / na
4	2.0	4.3	165	1360.65	0.924	0.567	0.415	0.0018	0.104 / 0.054
4	CA*	0.0	265	1381.20	0.931	0.574	0.411	0.0013	0.337 / 0.096
4	CA*	4.3	101	1361.46	0.925	0.566	0.418	0.0012	0.137 / 0.062
6	0.5	0.0	295	1375.38	0.933	0.571	0.417	0.0004	0.095 / 0.040
6	0.5	4.3	261	1095.03	0.832	0.497	0.418	0.0006	0.177 / 0.070
6	1.0	0.0	290	1366.82	0.925	0.567	0.418	0.0114	0.486 / 0.110
6	1.0	4.3	154	<u>1087.02</u>	0.839	0.500	0.415	0.0002	0.162 / 0.070
6	2.0	0.0	235	1206.09	0.867	0.527	0.421	0.0082	0.190 / 0.066
6	2.0	4.3	273	1083.70	0.824	0.497	0.420	0.0013	0.247 / 0.073
6	CA*	0.0	297	1130.03	0.849	0.513	0.414	0.0044	0.136 / 0.046
6	CA*	4.3	144	1106.41	0.845	0.506	0.420	0.0061	0.195 / 0.073

* CA denotes Cost Annealing.

A Spearman correlation reported as "na" indicates that the corresponding latent dimension exhibits insufficient variance across individuals to compute a rank correlation, which may be symptomatic of posterior collapse.

B.2.3 CLODE β Kernel

Table B.5: No-IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters			Validation Metrics		Test Metrics		
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}
4	1.0	0.0	298	193.56	0.341	0.200	0.852
4	1.0	4.3	213	<u>46.28</u>	0.198	0.112	0.968
4	2.0	0.0	203	191.51	0.326	0.176	0.848
4	2.0	4.3	121	126.02	0.295	0.161	0.853
4	CA*	0.0	297	40.43	0.169	0.095	0.973
4	CA*	4.3	234	115.16	0.282	0.156	0.858

* CA denotes Cost Annealing.

Table B.6: IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters			Validation Metrics		Test Metrics				
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}	RV Coefficient	Spearman (max/mean)
3	0.5	0.0	293	896.70	0.734	0.474	0.431	0.0000	na / na
3	0.5	4.3	297	<u>719.55</u>	0.669	0.433	0.430	0.0072	0.118 / 0.069
3	1.0	0.0	257	766.70	0.681	0.441	0.424	0.0000	na / na
3	1.0	4.3	202	844.13	0.719	0.465	0.430	0.0008	0.283 / 0.114
3	2.0	0.0	299	1043.88	0.786	0.512	0.430	0.0046	0.392 / 0.100
3	2.0	4.3	246	824.91	0.699	0.451	0.434	0.0240	0.141 / 0.100
3	4.0	0.0	287	938.67	0.745	0.481	0.435	0.0000	na / na
3	4.0	4.3	211	873.70	0.729	0.471	0.429	0.0125	0.333 / 0.088
3	CA*	0.0	247	903.79	0.737	0.476	0.431	0.0008	0.100 / 0.054
3	CA*	4.3	290	723.67	0.670	0.433	0.428	0.0007	0.217 / 0.083
4	0.5	0.0	295	978.59	0.768	0.498	0.439	0.0000	na / na
4	0.5	4.3	130	890.33	0.731	0.471	0.439	0.0000	na / na
4	1.0	0.0	291	980.03	0.772	0.501	0.436	0.0054	0.136 / 0.047
4	1.0	4.3	147	852.62	0.719	0.464	0.439	0.0009	0.100 / 0.039
4	2.0	0.0	299	984.58	0.764	0.496	0.430	0.0000	na / na
4	2.0	4.3	147	854.32	0.733	0.474	0.427	0.0043	0.169 / 0.076
4	4.0	0.0	296	1032.34	0.787	0.511	0.432	0.0007	0.167 / 0.094
4	4.0	4.3	206	854.29	0.722	0.468	0.423	0.0072	0.118 / 0.069
4	CA*	0.0	143	984.76	0.767	0.486	0.424	0.0000	na / na
4	CA*	4.3	155	823.56	0.707	0.457	0.429	0.0097	0.148 / 0.088
6	0.5	0.0	291	946.23	0.766	0.497	0.426	0.0000	na / na
6	0.5	4.3	226	717.86	0.666	0.431	0.427	0.0226	0.246 / 0.113
6	1.0	0.0	297	971.66	0.770	0.500	0.437	0.0000	na / na
6	1.0	4.3	294	767.90	0.674	0.435	0.432	0.0015	0.149 / 0.072
6	2.0	0.0	298	1065.49	0.846	0.526	0.390	0.0096	0.312 / 0.092
6	2.0	4.3	66	851.32	0.727	0.462	0.436	0.0025	0.142 / 0.059
6	4.0	0.0	285	888.01	0.725	0.464	0.431	0.0000	na / na
6	4.0	4.3	268	775.14	0.682	0.440	0.425	0.0092	0.336 / 0.102
6	CA*	0.0	296	812.71	0.724	0.463	0.410	0.0000	na / na
6	CA*	4.3	281	738.40	0.673	0.435	0.424	0.0129	0.140 / 0.063

* CA denotes Cost Annealing.

B.2.4 CLODE Dirac Impulse

Table B.7: No-IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters			Validation Metrics		Test Metrics		
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}
4	1.0	0.0	286	253.45	0.394	0.234	0.738
4	1.0	4.3	140	<u>143.86</u>	0.310	0.175	0.867
4	2.0	0.0	276	162.23	0.306	0.166	0.878
4	2.0	4.3	122	158.69	0.325	0.185	0.863
4	CA*	0.0	297	233.83	0.375	0.225	0.727
4	CA*	4.3	199	132.57	0.304	0.165	0.858

* CA denotes Cost Annealing.

Table B.8: IIV setting. Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss). All remaining metrics are computed on the test set.

Hyperparameters			Validation Metrics		Test Metrics				
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	RMSE	CRPS	Coverage _{90%}	RV Coefficient	Spearman (max/mean)
3	0.5	0.0	15	2163.50	1.143	0.719	0.351	0.0029	0.259 / 0.097
3	0.5	4.3	160	724.89	0.675	0.436	0.434	0.0169	0.124 / 0.059
3	1.0	0.0	290	956.52	0.758	0.491	0.429	0.0000	na / na
3	1.0	4.3	293	<u>746.85</u>	0.682	0.441	0.424	0.0000	na / na
3	2.0	0.0	298	991.23	0.767	0.498	0.435	0.0000	na / na
3	2.0	4.3	138	919.10	0.761	0.493	0.429	0.0108	0.165 / 0.063
3	4.0	0.0	268	940.71	0.752	0.482	0.427	0.0000	na / na
3	4.0	4.3	297	958.48	0.764	0.495	0.443	0.0000	na / na
3	CA*	0.0	296	762.45	0.683	0.442	0.421	0.0000	na / na
3	CA*	4.3	271	918.00	0.756	0.491	0.431	0.0011	0.309 / 0.119
4	0.5	0.0	271	954.86	0.766	0.497	0.437	0.0012	0.199 / 0.074
4	0.5	4.3	283	865.33	0.730	0.470	0.436	0.0000	na / na
4	1.0	0.0	245	847.56	0.728	0.465	0.421	0.0070	0.211 / 0.073
4	1.0	4.3	225	893.50	0.757	0.490	0.418	0.0210	0.149 / 0.064
4	2.0	0.0	300	1172.13	0.859	0.541	0.424	0.0000	na / na
4	2.0	4.3	210	931.71	0.758	0.492	0.432	0.0186	0.128 / 0.064
4	4.0	0.0	297	1066.46	0.815	0.527	0.425	0.0000	na / na
4	4.0	4.3	200	907.99	0.744	0.480	0.431	0.0045	0.691 / 0.183
4	CA*	0.0	292	1117.86	0.831	0.517	0.428	0.0007	0.081 / 0.035
4	CA*	4.3	259	844.13	0.720	0.464	0.432	0.0170	0.120 / 0.064
6	0.5	0.0	300	961.78	0.763	0.496	0.437	0.0000	na / na
6	0.5	4.3	291	878.62	0.732	0.474	0.427	0.0000	na / na
6	1.0	0.0	287	962.88	0.765	0.498	0.431	0.0000	na / na
6	1.0	4.3	210	881.09	0.742	0.481	0.423	0.0000	na / na
6	2.0	0.0	286	957.37	0.762	0.495	0.433	0.0000	na / na
6	2.0	4.3	178	891.66	0.731	0.470	0.438	0.0303	0.240 / 0.075
6	4.0	0.0	25	1323.98	0.879	0.538	0.422	0.0012	0.165 / 0.074
6	4.0	4.3	296	909.55	0.735	0.474	0.433	0.0000	0.145 / 0.056
6	CA*	0.0	290	996.68	0.780	0.508	0.431	0.0000	na / na
6	CA*	4.3	211	903.19	0.755	0.490	0.424	0.0000	na / na

* CA denotes Cost Annealing.

B.3 Counterfactual Performance

In this appendix, we provide additional experimental results related to the counterfactual performance of the models. The model presented in the main text is bolded, the closest competitors is underlined.

B.3.1 ODE-LSTM

Table B.9: No-IIV setting. Counterfactual evaluation on the test set (PEHE and CF-RMSE across s1/s2/s3). Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss) used for model selection.

Hyperparameters	Validation Metrics		PEHE			CF-RMSE		
λ_{mix}	Best Epoch	Best Loss	s1	s2	s3	s1	s2	s3
0.0001	226	<u>10445.02</u>	0.078	0.812	0.553	0.689	0.695	0.411
0.001	300	10306.33	0.078	0.814	0.555	0.723	0.729	0.445
0.01	250	10646.57	0.078	0.812	0.553	0.692	0.698	0.415

Table B.10: IIV setting. Counterfactual evaluation on the test set (PEHE and CF-RMSE across s1/s2/s3). Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss) used for model selection.

Hyperparameters	Validation Metrics		PEHE			CF-RMSE		
λ_{mix}	Best Epoch	Best Loss	s1	s2	s3	s1	s2	s3
0.0001	256	<u>12537.20</u>	0.080	0.817	0.558	0.878	0.885	0.666
0.001	241	12515.29	0.080	0.817	0.559	0.884	0.891	0.676
0.01	277	12774.71	0.080	0.818	0.559	0.885	0.892	0.676

B.3.2 CLODE β Kernel

Table B.11: No-IIV setting. Counterfactual evaluation on the test set (PEHE and CF-RMSE across s1/s2/s3). Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss) used for model selection.

Hyperparameters			Validation Metrics		PEHE			CF-RMSE		
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	s1	s2	s3	s1	s2	s3
4	1.0	0.0	298	193.56	0.040	0.242	0.327	0.343	0.349	0.470
4	1.0	4.3	213	<u>46.28</u>	0.037	0.192	0.547	0.192	0.217	0.564
4	2.0	0.0	203	191.51	0.056	0.256	0.490	0.323	0.332	0.567
4	2.0	4.3	121	126.02	0.050	0.257	0.479	0.295	0.303	0.532
4	CA*	0.0	297	40.43	0.037	0.185	0.158	0.166	0.171	0.131
4	CA*	4.3	234	115.16	0.042	0.218	0.451	0.281	0.289	0.528

* CA denotes Cost Annealing.

Table B.12: IIV setting. Counterfactual evaluation on the test set (PEHE and CF-RMSE across s1/s2/s3). Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss) used for model selection.

Hyperparameters			Validation Metrics		PEHE			CF-RMSE		
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	s1	s2	s3	s1	s2	s3
3	0.5	0.0	293	896.70	0.059	0.498	0.492	0.711	0.716	0.714
3	0.5	4.3	297	<u>719.55</u>	0.057	0.457	0.328	0.634	0.637	0.533
3	1.0	0.0	257	766.70	0.061	0.477	0.349	0.652	0.651	0.548
3	1.0	4.3	202	844.13	0.059	0.494	0.543	0.694	0.698	0.733
3	2.0	0.0	299	1043.88	0.064	0.513	0.381	0.772	0.770	0.707
3	2.0	4.3	246	824.91	0.061	0.498	0.563	0.670	0.679	0.713
3	4.0	0.0	287	938.67	0.066	0.499	0.446	0.724	0.728	0.714
3	4.0	4.3	211	873.70	0.067	0.515	0.574	0.702	0.708	0.722
3	CA*	0.0	247	903.79	0.062	0.503	0.468	0.714	0.719	0.709
3	CA*	4.3	290	723.67	0.059	0.458	0.328	0.636	0.639	0.536
4	0.5	0.0	295	978.59	0.055	0.458	0.334	0.750	0.750	0.697
4	0.5	4.3	130	890.33	0.063	0.493	0.456	0.710	0.711	0.706
4	1.0	0.0	291	980.03	0.059	0.466	0.345	0.754	0.754	0.702
4	1.0	4.3	147	852.62	0.062	0.497	0.570	0.694	0.698	0.739
4	2.0	0.0	299	984.58	0.058	0.492	0.353	0.747	0.749	0.701
4	2.0	4.3	147	854.32	0.061	0.502	0.484	0.709	0.712	0.696
4	4.0	0.0	296	1032.34	0.058	0.479	0.357	0.771	0.769	0.719
4	4.0	4.3	206	854.29	0.066	0.508	0.493	0.699	0.703	0.703
4	CA*	0.0	143	984.76	0.091	0.633	0.444	0.750	0.825	0.701
4	CA*	4.3	155	823.56	0.064	0.485	0.506	0.681	0.686	0.703
6	0.5	0.0	291	946.23	0.102	0.511	0.390	0.745	0.747	0.690
6	0.5	4.3	226	717.86	0.057	0.451	0.322	0.632	0.634	0.533
6	1.0	0.0	297	971.66	0.057	0.466	0.340	0.752	0.752	0.698
6	1.0	4.3	294	767.90	0.066	0.474	0.347	0.644	0.653	0.542
6	2.0	0.0	298	1065.49	0.073	0.699	0.481	0.804	0.806	0.624
6	2.0	4.3	66	851.32	0.065	0.529	0.387	0.698	0.700	0.572
6	4.0	0.0	285	888.01	0.066	0.481	0.353	0.698	0.707	0.636
6	4.0	4.3	268	775.14	0.063	0.478	0.344	0.649	0.649	0.549
6	CA*	0.0	296	812.71	0.068	0.528	0.372	0.691	0.688	0.569
6	CA*	4.3	281	738.40	0.060	0.461	0.338	0.641	0.638	0.540

* CA denotes Cost Annealing.

B.3.3 CLODE Dirac Impulse

Table B.13: No-IIV setting. Counterfactual evaluation on the test set (PEHE and CF-RMSE across s1/s2/s3). Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss) used for model selection.

Hyperparameters			Validation Metrics		PEHE			CF-RMSE		
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	s1	s2	s3	s1	s2	s3
4	1.0	0.0	286	253.45	0.041	0.205	0.165	0.395	0.401	0.428
4	1.0	4.3	140	<u>143.86</u>	0.065	0.236	0.411	0.312	0.319	0.516
4	2.0	0.0	276	162.23	0.075	0.273	0.444	0.305	0.317	0.510
4	2.0	4.3	122	158.69	0.057	0.243	0.368	0.326	0.334	0.470
4	CA*	0.0	297	233.83	0.049	0.196	0.196	0.377	0.381	0.422
4	CA*	4.3	199	132.57	0.072	0.261	0.533	0.308	0.309	0.584

* CA denotes Cost Annealing.

Table B.14: IIV setting. Counterfactual evaluation on the test set (PEHE and CF-RMSE across s1/s2/s3). Hyperparameters are reported alongside validation metrics (Best Epoch and Best Validation Loss) used for model selection.

Hyperparameters			Validation Metrics		PEHE			CF-RMSE		
Latent Dim	β_{KL}	Output Bias	Best Epoch	Best Loss	s1	s2	s3	s1	s2	s3
3	0.5	0.0	15	2163.50	0.079	0.772	0.528	1.132	1.133	0.991
3	0.5	4.3	160	724.89	0.065	0.462	0.332	0.641	0.648	0.549
3	1.0	0.0	290	956.52	0.065	0.484	0.373	0.736	0.739	0.697
3	1.0	4.3	293	<u>746.85</u>	0.071	0.469	0.337	0.649	0.652	0.547
3	2.0	0.0	298	991.23	0.060	0.462	0.332	0.749	0.751	0.699
3	2.0	4.3	138	919.10	0.065	0.460	0.341	0.739	0.745	0.693
3	4.0	0.0	268	940.71	0.065	0.525	0.492	0.728	0.733	0.715
3	4.0	4.3	297	958.48	0.058	0.457	0.328	0.744	0.746	0.692
3	CA*	0.0	296	762.45	0.062	0.478	0.342	0.652	0.659	0.555
3	CA*	4.3	271	918.00	0.059	0.464	0.344	0.737	0.741	0.690
4	0.5	0.0	271	954.86	0.059	0.460	0.334	0.747	0.748	0.697
4	0.5	4.3	283	865.33	0.064	0.497	0.458	0.706	0.711	0.697
4	1.0	0.0	245	847.56	0.074	0.613	0.407	0.699	0.757	0.577
4	1.0	4.3	225	893.50	0.083	0.509	0.453	0.725	0.730	0.690
4	2.0	0.0	300	1172.13	0.079	0.673	0.471	0.834	0.839	0.691
4	2.0	4.3	210	931.71	0.054	0.463	0.335	0.738	0.743	0.688
4	4.0	0.0	297	1066.46	0.069	0.507	0.372	0.794	0.794	0.739
4	4.0	4.3	200	907.99	0.066	0.500	0.434	0.722	0.726	0.691
4	CA*	0.0	292	1117.86	0.080	0.757	0.505	0.803	0.874	0.656
4	CA*	4.3	259	844.13	0.060	0.496	0.518	0.693	0.700	0.710
6	0.5	0.0	300	961.78	0.052	0.453	0.333	0.744	0.745	0.697
6	0.5	4.3	291	878.62	0.058	0.517	0.464	0.707	0.713	0.698
6	1.0	0.0	287	962.88	0.060	0.457	0.326	0.744	0.747	0.695
6	1.0	4.3	210	881.09	0.071	0.494	0.469	0.719	0.722	0.695
6	2.0	0.0	286	957.37	0.054	0.471	0.336	0.742	0.746	0.693
6	2.0	4.3	178	891.66	0.069	0.503	0.444	0.704	0.708	0.706
6	4.0	0.0	25	1323.98	0.077	0.591	0.420	0.861	0.863	0.760
6	4.0	4.3	296	909.55	0.068	0.530	0.477	0.710	0.714	0.710
6	CA*	0.0	290	996.68	0.057	0.505	0.370	0.762	0.763	0.702
6	CA*	4.3	211	903.19	0.055	0.462	0.367	0.734	0.738	0.694

* CA denotes Cost Annealing.

B.4 Latent Space Visualization

In this appendix, we provide additional visualizations of the latent space learned by the models under the IIV setting. We present UMAP projections of the learned initial condition for all test samples. This is done only for the best performing hyperparameter configuration for each model variant, as determined by the best validation loss.

B.4.1 CLODE β Kernel

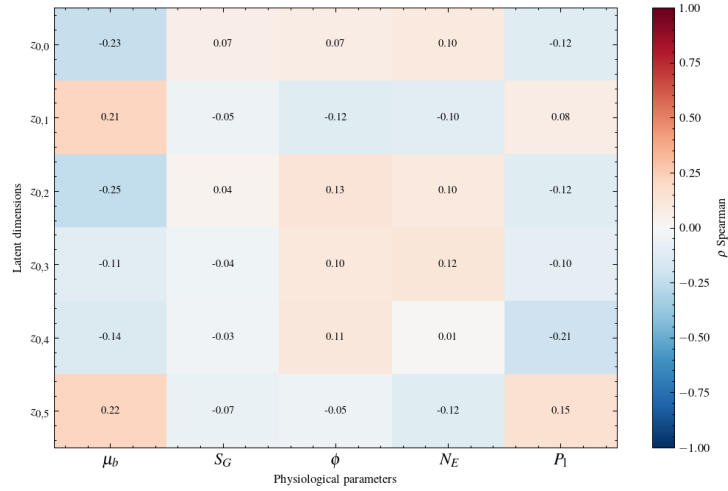


FIGURE B.1: Spearman rank correlation between latent dimensions $\mathbf{z}_{N-1}$ and ground-truth physiological parameters on the **IIV** dataset.

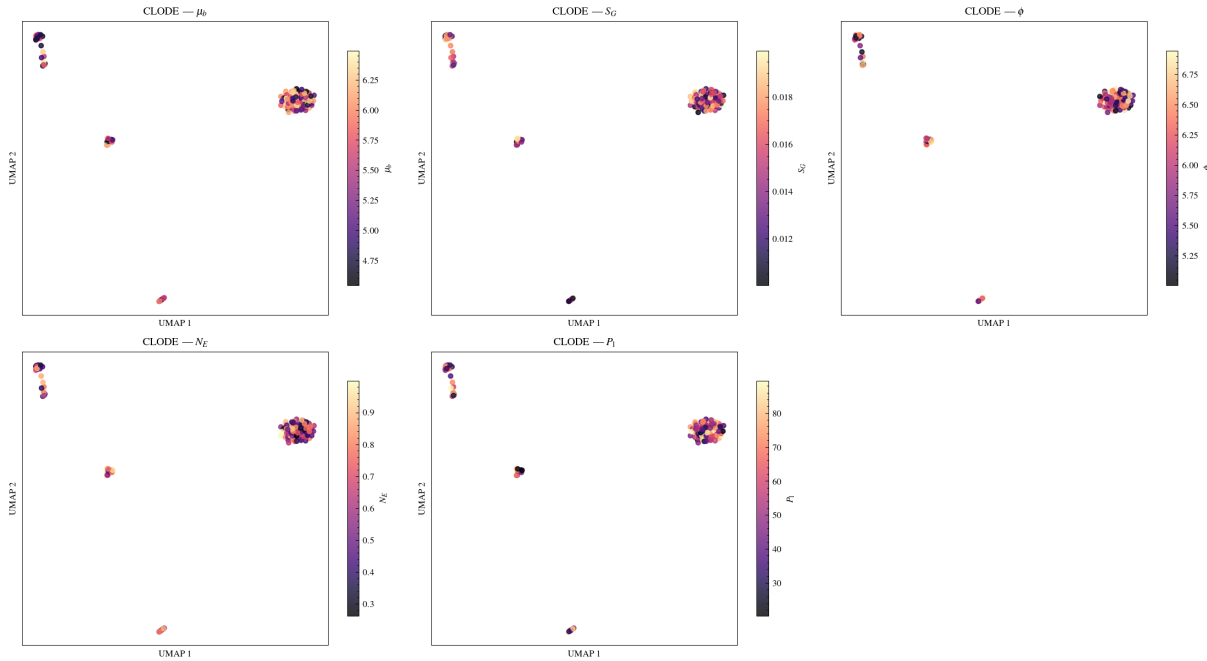


FIGURE B.2: UMAP 2D projection of the latent space $\mathbf{z}_{N-1}$ per physiological parameters on the **IIV** dataset.

B.4.2 CLODE Dirac Impulse

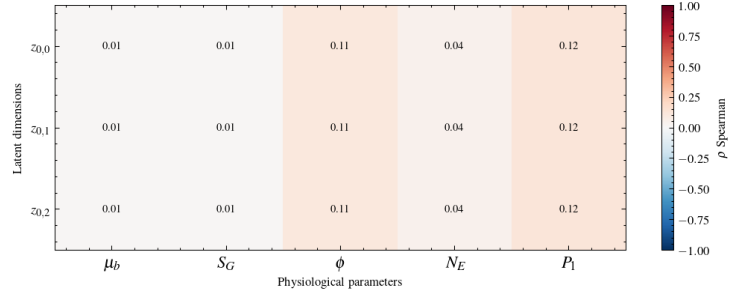


FIGURE B.3: Spearman rank correlation between latent dimensions $\mathbf{z}_{N-1}$ and ground-truth physiological parameters on the **IIV** dataset.

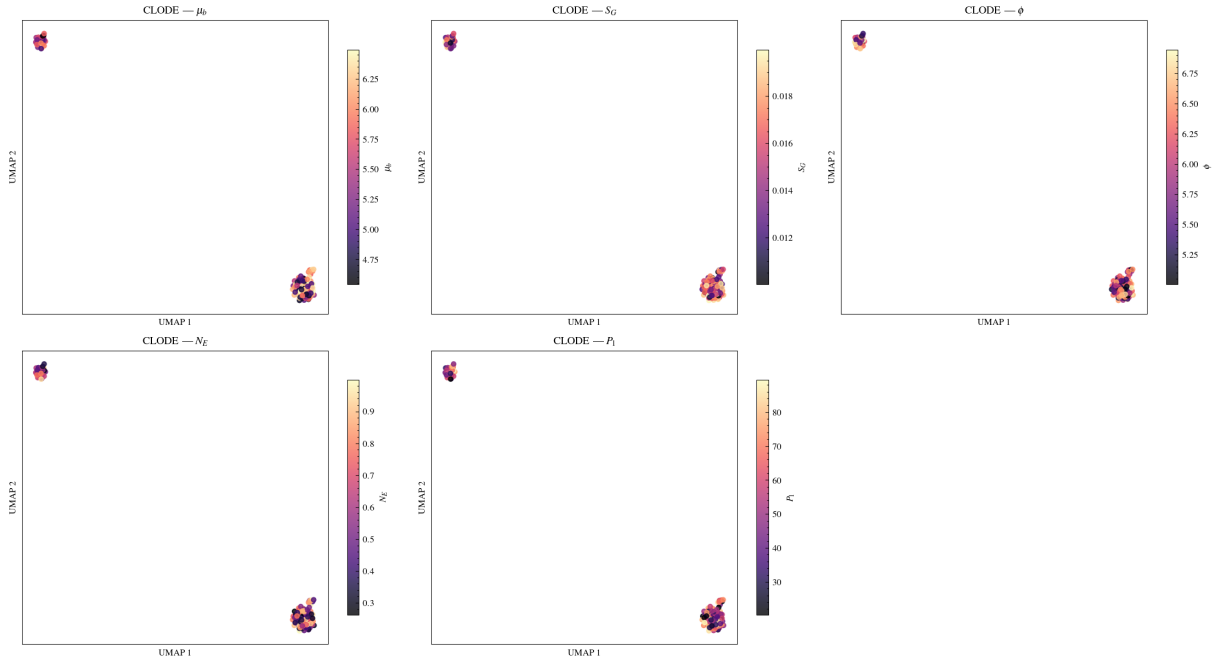


FIGURE B.4: UMAP 2D projection of the latent space $\mathbf{z}_{N-1}$ per physiological parameters on the **IIV** dataset.

B.4.3 Latent ODE-RNN

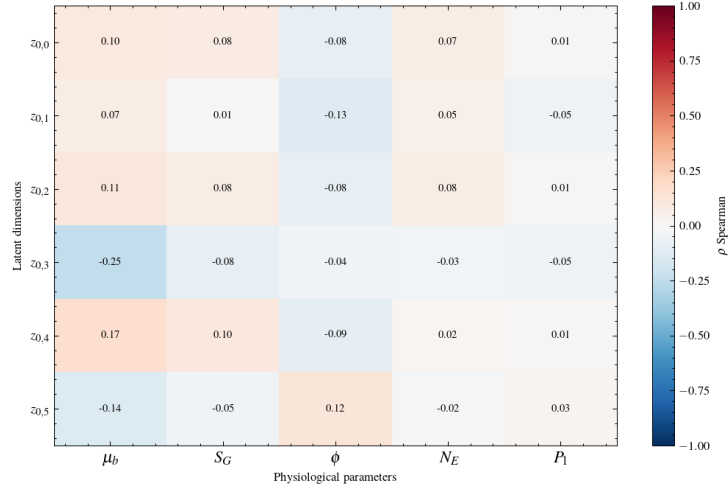


FIGURE B.5: Spearman rank correlation between latent dimensions $\mathbf{z}_{N-1}$ and ground-truth physiological parameters on the **IIV** dataset.

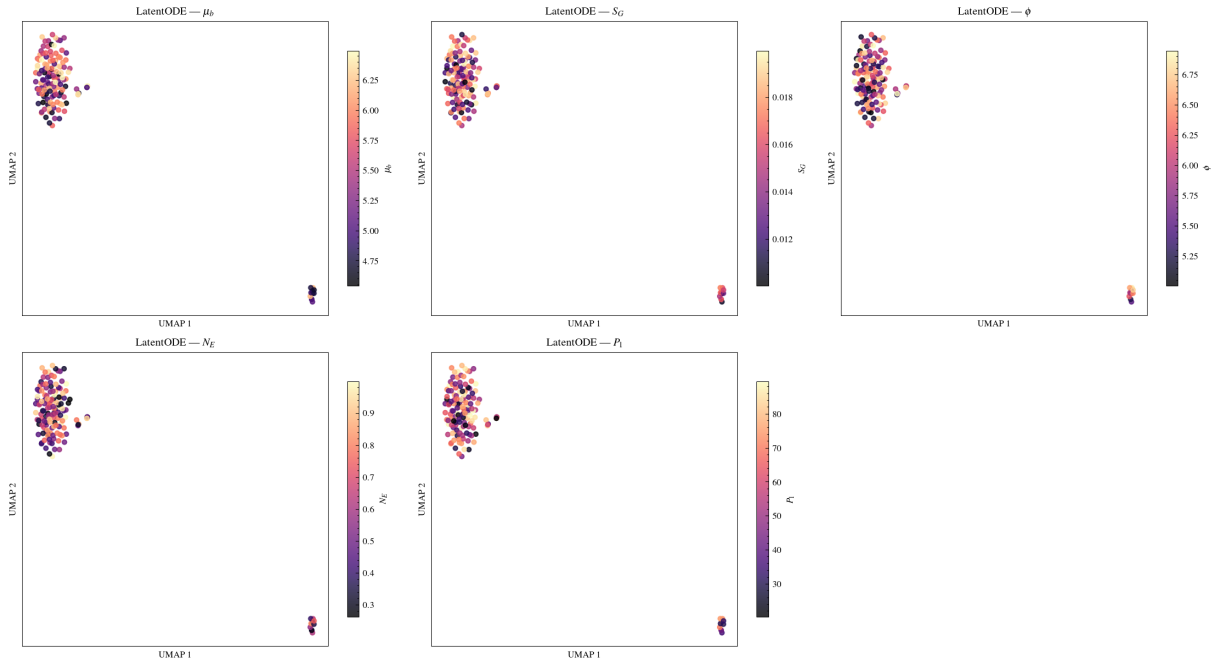


FIGURE B.6: UMAP 2D projection of the latent space $\mathbf{z}_{N-1}$ per physiological parameters on the **IIV** dataset.

B.5 Model Predictions Across Training Epochs

In this appendix, we provide visualizations of the model predictions across training epochs for a randomly selected validation patient (patient 392). Each 50 epochs, we plot the model’s predictions for the 24-hour extrapolation window. For the sake of readability, we only present the results for the best performing hyperparameter configuration for each model variant, as determined by the best validation loss.

B.5.1 CLODE β Kernel

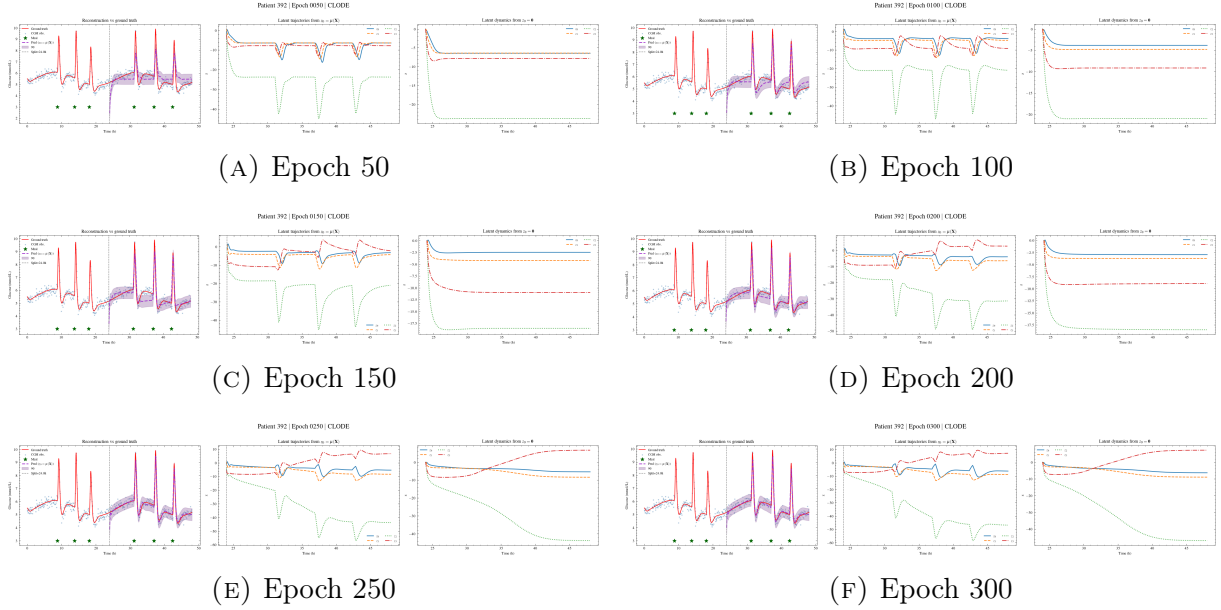


FIGURE B.7: **No-IIV setting.** Training progression of CLODE (β -kernel) on the IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the latent dynamics, starting from the estimated latent initial condition $\mathbf{z}_{N-1}$ (posterior mean), including the control inputs ; (iii) the latent dynamics, starting from $\mathbf{0}$ (prior mean), without control inputs.

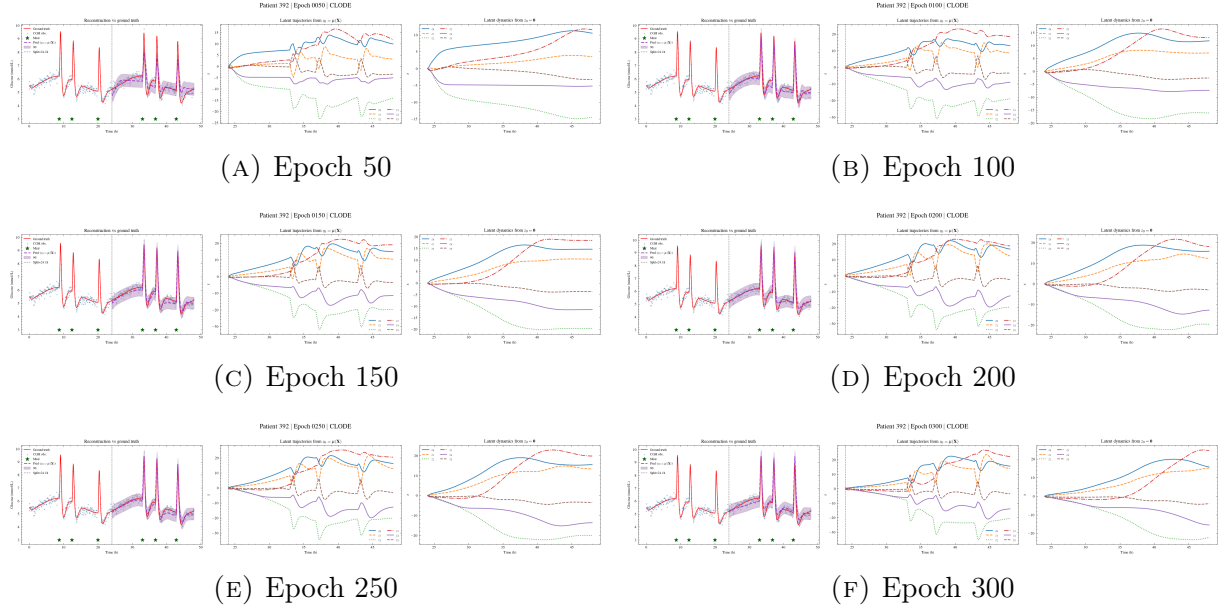


FIGURE B.8: **IIV setting.** Training progression of CLODE (β -kernel) on the IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the latent dynamics, starting from the estimated latent initial condition $\mathbf{z}_{N-1}$ (posterior mean), including the control inputs ; (iii) the latent dynamics, starting from $\mathbf{0}$ (prior mean), without control inputs.

B.5.2 CLODE Dirac Impulse

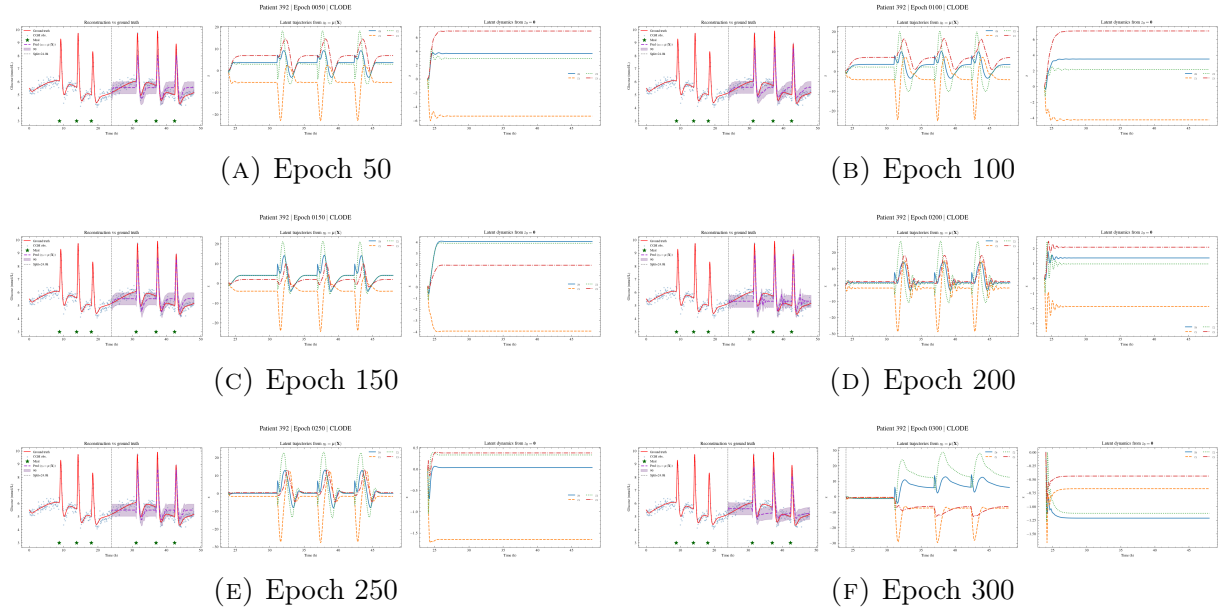


FIGURE B.9: **No-IIV setting.** Training progression of CLODE (Dirac impulse) on the IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the latent dynamics, starting from the estimated latent initial condition $\mathbf{z}_{N-1}$ (posterior mean), including the control inputs ; (iii) the latent dynamics, starting from $\mathbf{0}$ (prior mean), without control inputs.

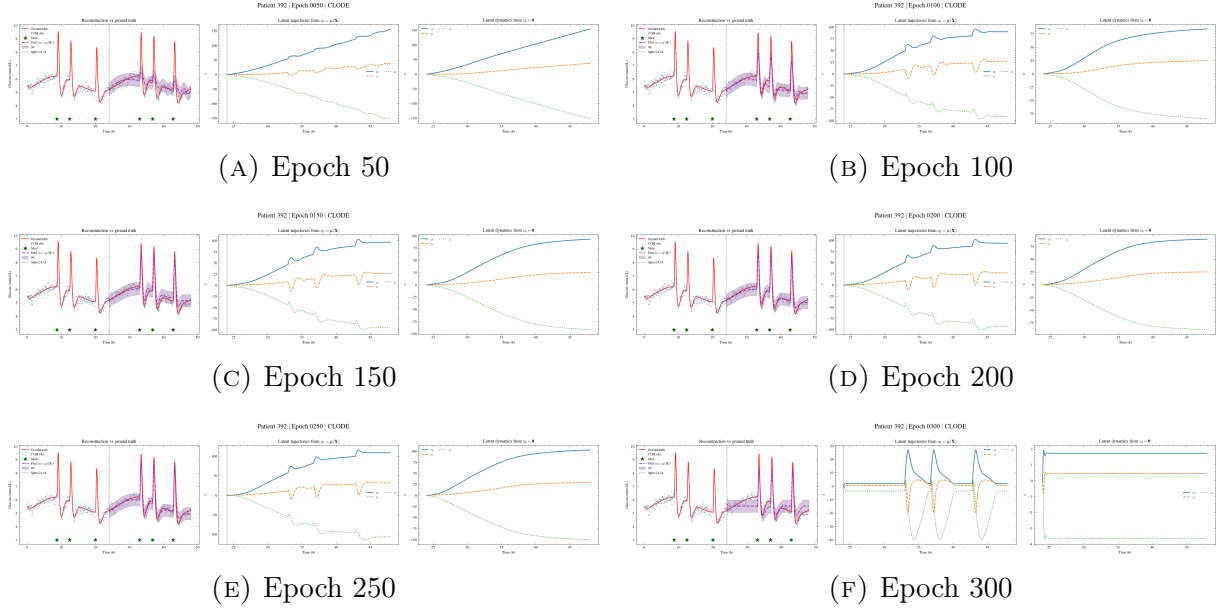


FIGURE B.10: **IIV setting.** Training progression of CLODE (Dirac impulse) on the IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the latent dynamics, starting from the estimated latent initial condition $\mathbf{z}_{N-1}$ (posterior mean), including the control inputs ; (iii) the latent dynamics, starting from $\mathbf{0}$ (prior mean), without control inputs.

B.5.3 Latent ODE-RNN

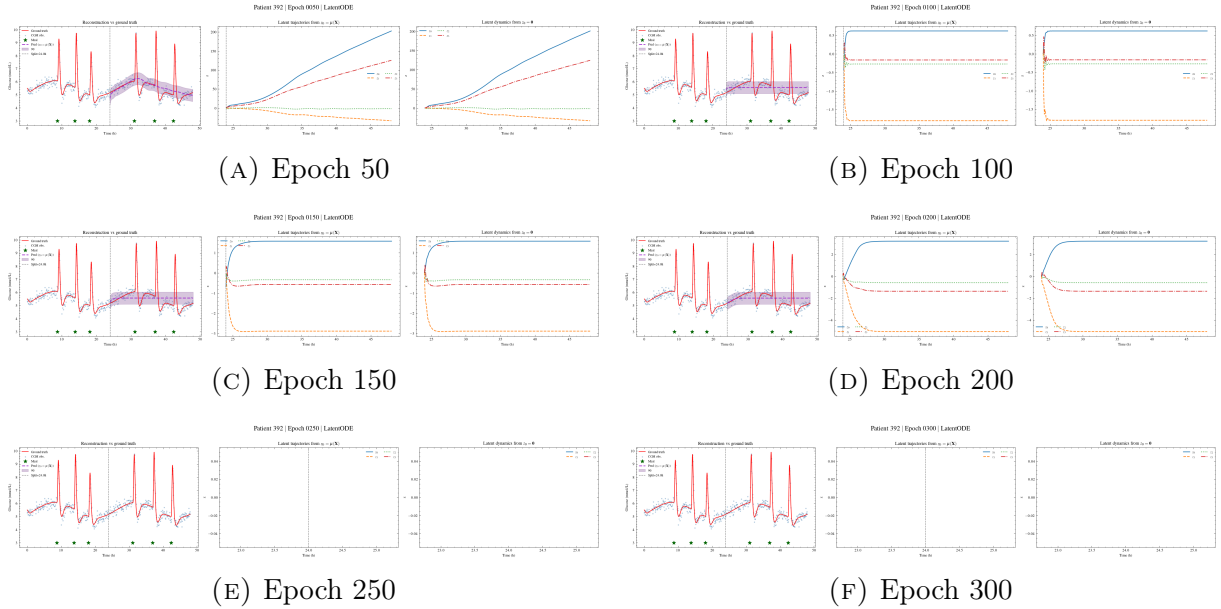


FIGURE B.11: **No-IIV setting.** Training progression of Latent ODE-RNN on the IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the latent dynamics, starting from the estimated latent initial condition $\mathbf{z}_{N-1}$ (posterior mean), including the control inputs ; (iii) the latent dynamics, starting from $\mathbf{0}$ (prior mean), without control inputs.

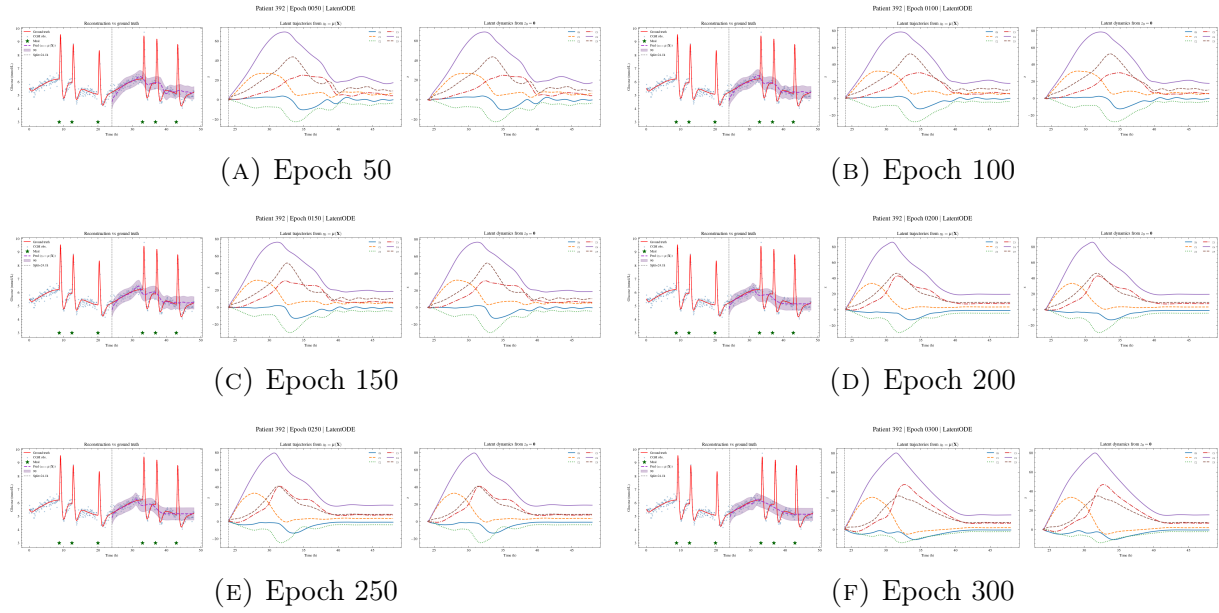


FIGURE B.12: **IIV setting.** Training progression of Latent ODE-RNN on the IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the latent dynamics, starting from the estimated latent initial condition $\mathbf{z}_{N-1}$ (posterior mean), including the control inputs ; (iii) the latent dynamics, starting from $\mathbf{0}$ (prior mean), without control inputs.

B.5.4 ODE-LSTM

Note that the performance of the ODE-LSTM might seem surprisingly poor. We believe that the model converged to a local minimum, consisting of predicting a flat trajectory with large uncertainty. This is supported by the training progression visualizations, which show that the model quickly learns to predict a flat trajectory, and then struggles to escape this local minimum.

When we shift to a one-step ahead prediction task, the ODE-LSTM performs much better, as illustrated in Figure B.15.

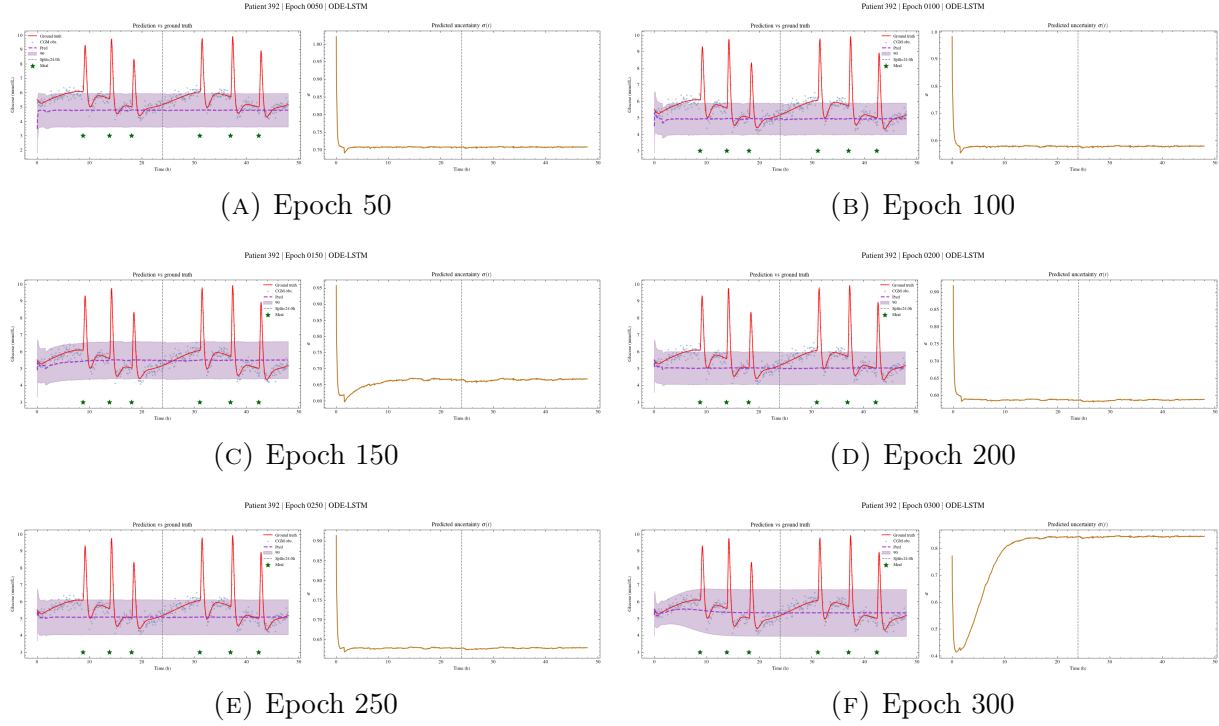


FIGURE B.13: **No-IIV setting.** Training progression of ODE-LSTM on the no-IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the predicted uncertainty $\sigma(t)$ (yellow).

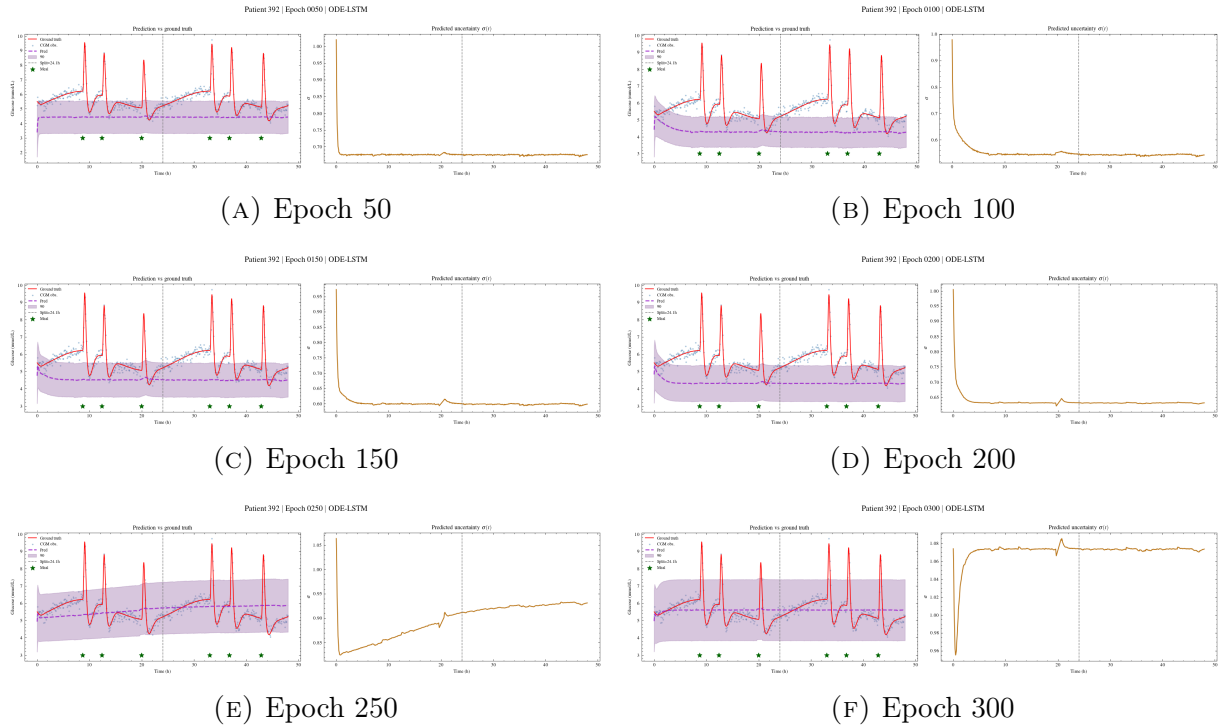
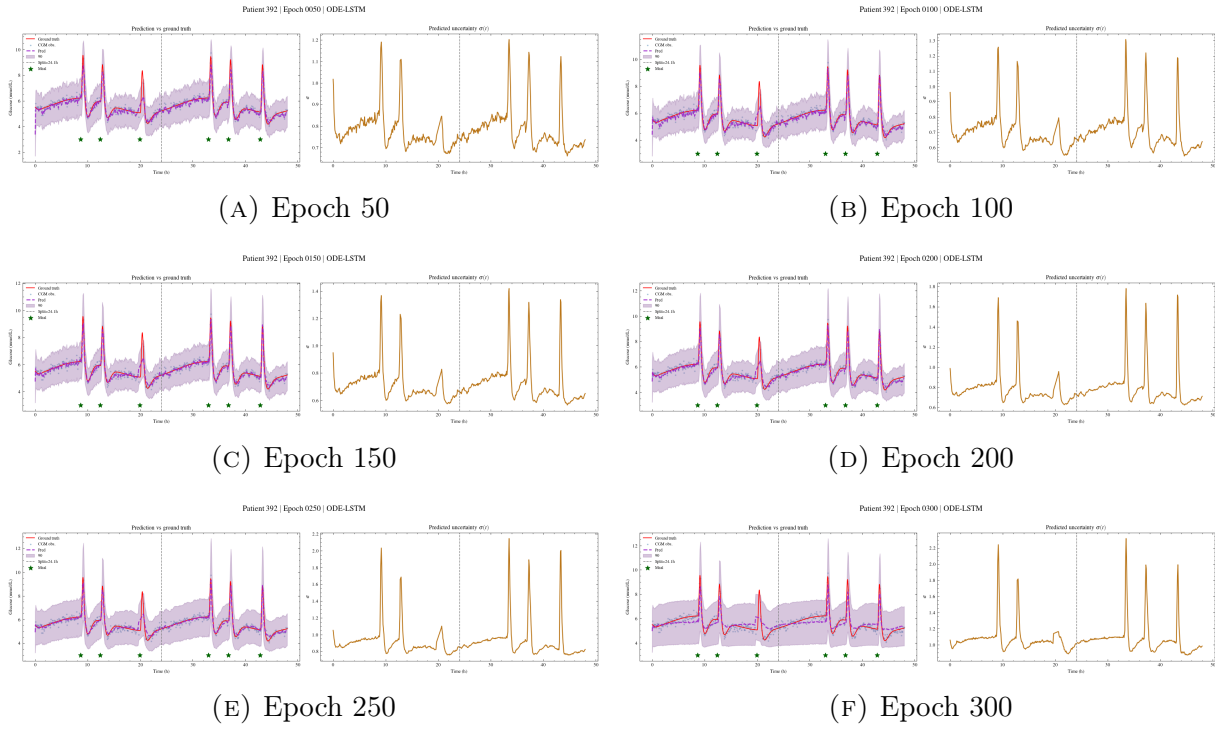


FIGURE B.14: **IIV setting.** Training progression of ODE-LSTM on the IIV dataset. Each panel shows for a training epoch : (i) the factual prediction (purple) with 90% prediction interval against the ground truth (red) ; (ii) the predicted uncertainty $\sigma(t)$ (yellow).

FIGURE B.15: **IIV setting.** One-step ahead prediction.

C.1 Contributor Role Taxonomy

Contributions are reported using the Contributor Role Taxonomy (CRediT) taxonomy.
L = Lead; **S** = Support.

CRediT Role	Louis-David Piron	Guillaume Deside, PhdC	Laura Symul, Phd
Conceptualization	L	S	S
Methodology	L	-	S
Software	L	S	-
Formal Analysis	L	-	-
Investigation	L	-	-
Resources	L	S	-
Validation	L	-	-
Visualization	L	-	-
Writing – original draft	L	-	-
Writing – review & editing	S	S	L
Supervision	-	S	L

C.2 Example 2.2.2 : Additional Material

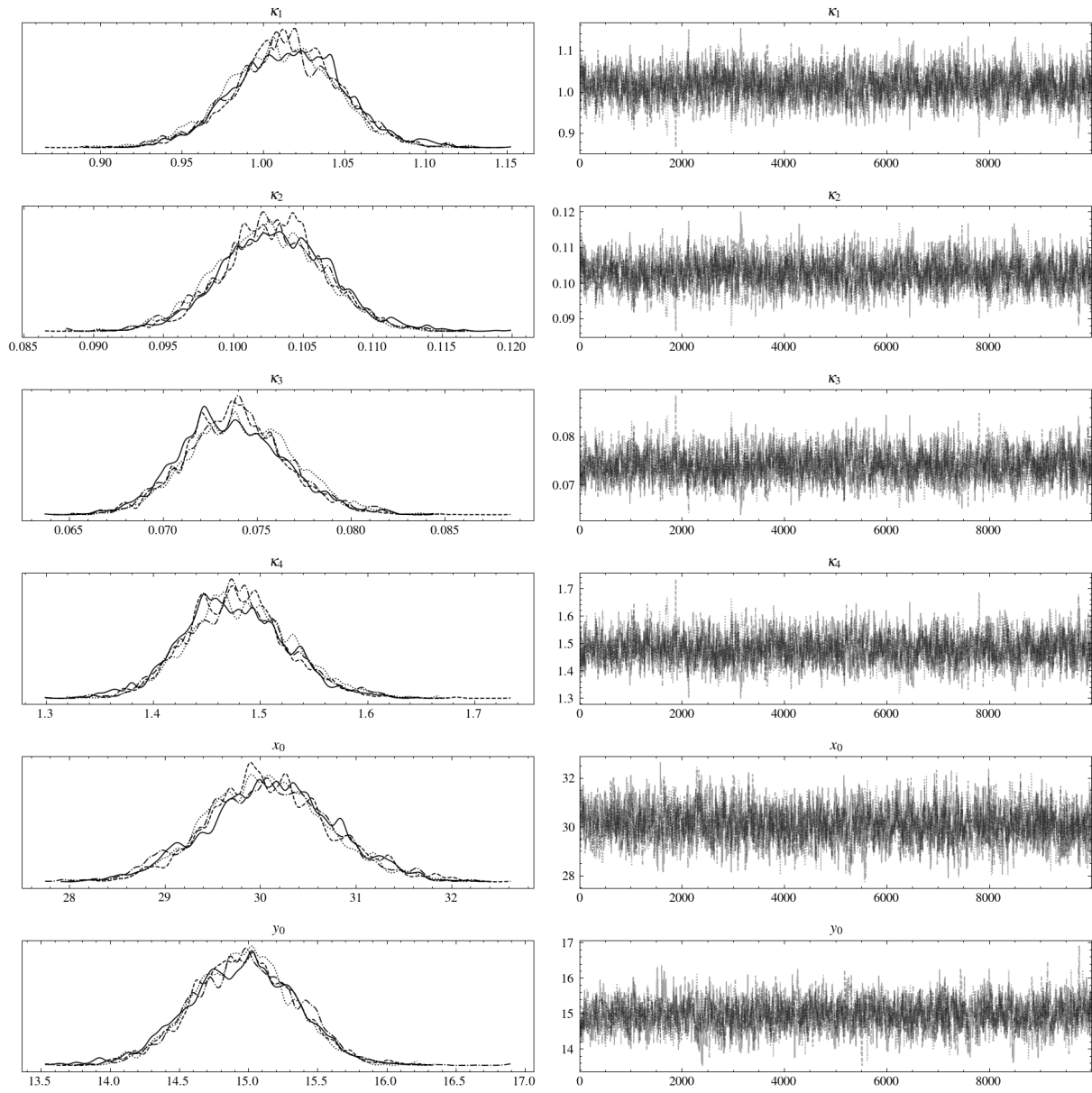


FIGURE C.1: Marginal posterior distributions and chain traceplots of the Lotka-Volterra parameters.

C.3 Example 2.3.1 : Additional Material

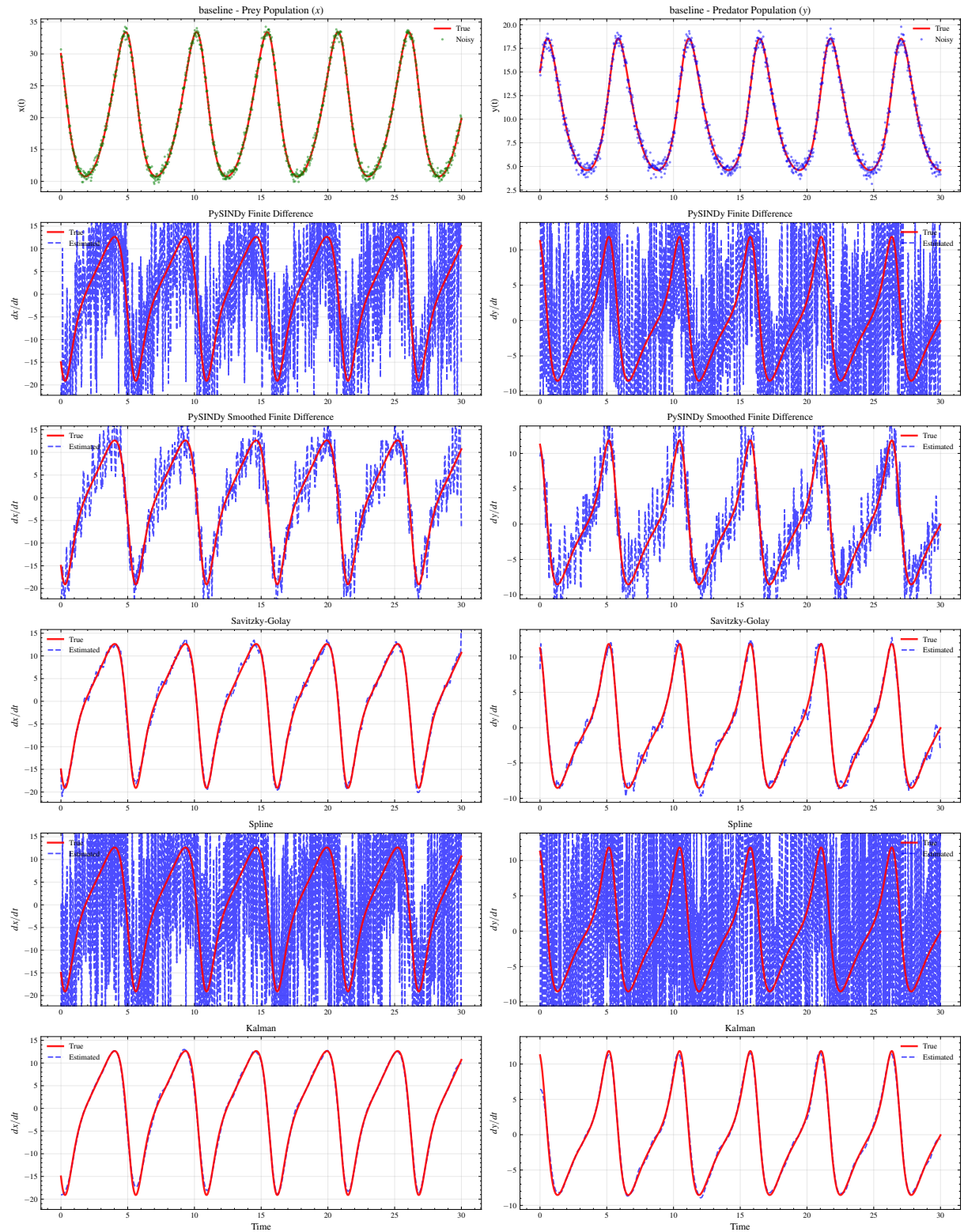


FIGURE C.2: Derivative analysis on the Baseline sample. The left (right) column is for the prey (predator) population. The first row represent raw time series and ground truths. The remaining rows present derivative estimations according to several methods.

