Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

OPCNet: Oscillatory Pattern Capturing Network for Highly Oscillatory Data

SangUk Hwang
Catholic University of Korea
Hyeong-Ohk Bae
Ajou University
Byungjoon Lee
blee@catholic.ac.kr

Catholic University of Korea

Research Article

Keywords:

Posted Date: August 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9635718/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

OPCNet: Oscillatory Pattern Capturing Network for Highly Oscillatory Data

SangUk Hwang*, Hyeong-Ohk Bae†, Byungjoon Lee‡

May 12, 2026

Abstract

We introduce a novel neural network-based model, called the oscillatory pattern capturing network (OPCNet), to fit highly oscillatory data by identifying the data with the solution of the linear ODE in [14] with perturbations. Given oscillatory data, the model first learns the oscillatory patterns of the data, which can be inferred from the dynamics of the linear ODE, and then fits the data with the solution of the ODE. Incorporating with a suitable loss function, the proposed model succeeds to find the ODE system to represent the oscillatory behavior of the given data. Several numerical results validate that the proposed model predicts the oscillatory pattern of the data well and demonstrate its outperformance compared to the result from the existing method.

1 Introduction

As mentioned in [14], oscillating problems have been introduced by astronomers in 1957 for orbits of artificial satellites. Several approaches have been developed to solve oscillating problems (refer to references in [14]). One of the popular oscillating phenomena in classical mechanics is a harmonic oscillator, which describes the behavor of a pendulum. In real oscillators, there is friction or damping, which shows a damped harmonic oscillator. These are described in the form of ordinary differential equations (ODEs) related to the conservation of momentum, or Newton's second law. If we solve ODEs, we may understand their behavior.

Related to solving ODE, there might be two directions: one is to solve a given ODE, and the other is to find an ODE itself, and solve to fit the given data. In the first direction, many methods have been developed; however, for highly oscillatory problems it is still not satisfactory. In the second direction, usually statistical regressions or other approaches are developed much, but still not enough in the ODE approaches. Recently, with deep neural networks, such approaches become popular. Oscillatory curves are initially treated with linear systems, then their perturbations:

$$\frac{d\boldsymbol{x}}{dt} = \frac{A}{\varepsilon}\boldsymbol{x} + g(t, \boldsymbol{x}), \quad A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \tag{1.1}$$

where $\varepsilon > 0$ is small, of which solution is approximated by a series of functions of t/ε (refer to [14]). In [14] a given initial value problem

$$y' = f(y, t), \quad y(0) = y_0, \quad 0 \le t \le L$$

*Department of Artificial Intelligence, The Catholic University of Korea, Bucheon-si, Gyeonggi-do, 14663, S. Korea. Email: lall21@catholic.ac.kr

†Department of Financial Engineering, Ajou University, Suwon-si, Gyeonggi-do, 16499, S. Korea. Email: hobae@ajou.ac.kr

‡**Corresponding author**. Department of Mathematics, The Catholic University of Korea, Bucheon-si, Gyeonggi-do, 14663, S. Korea. Email: blee@catholic.ac.kr

is solved approximately by the concept of quasi-envelope. In [1] a numerical routine is provided for the efficient solution of one dimensional, second-order, ordinary differential equations with rapidly oscillating solutions. The method is based on a Runge-Kutta-like stepping procedure that makes use of the Wentzel-Kramers-Brillouin approximation to skip regions of integration where the characteristic frequency varies slowly.

In [10], highly-oscillatory orbits are obtained by solving the linear oscillator $y'' + g(t)y = 0$, where $g(t) \to \infty$ as $t \to \infty$. In [6], numerical methods for oscillatory, multi-scale Hamiltonian systems are reviewed. Recent advances such as the modulated Fourier expansion approach [17] and the multiscale time integrators for highly oscillatory equations [2] further extend these ideas, providing uniformly accurate schemes even in extreme frequency regimes.

Recently, as a deep learning technique to solve ODEs, neural ODE is introduced in [5]. A given data may be approximated by this neural ODE. In [3, 8], they fit neural ODEs and ODEs to the oscillating Hare-Lynx data for the classic ecological data. Although such oscillatory pattern predictions are very important in various fields, we cannot find any other research to fit highly oscillatory data based on the neural network model, as far as we know.

In this paper, we introduce a novel neural network-based model, called the oscillatory pattern capturing network (OPCNet), to predict the oscillatory pattern of the given data equipped with a proper ODE system. More precisely, we make use of the assumption from [14] that highly oscillatory curve is able to be seen as the solution of the linear ODE (1.1) with a small perturbation to construct the prediction model. With this assumption, we allow OPCNet to learn the dynamics of the ODE and its solution simultaneously, so that the model predicts the oscillatory pattern of the data correctly.

In section 2, several works related to predicting the pattern of highly oscillatory data are reviewed. The detailed architecture of OPCNet is introduced in section 3, and numerical results are presented in section 5 to validate the outperformance of OPCNet. Section 6 finalizes the article presenting the conclusion and future works.

2 Related Works

2.1 Classical and Numerical Approaches for Highly Oscillatory ODEs

Many researchers have investigated numerical approximation methods for highly oscillatory ordinary differential equations (ODEs). Petzold [14] proposed an efficient approach to solving such problems, exploiting the "quasi-envelope" structure observed in the solutions of

$$\frac{d\boldsymbol{x}}{dt} = \frac{A}{\varepsilon}\boldsymbol{x} + g(t, \boldsymbol{x}), \quad A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \tag{2.1}$$

where $\varepsilon > 0$ is small. Cohen et al. [6] developed trigonometric and adaptive integrators for multi-scale Hamiltonian systems, while Bunder et al. [4] proposed a generalized Filon quadrature for oscillatory problems. Khanamiryan [11] introduced asymptotic and Filon-type methods for linear systems and waveform relaxation for nonlinear systems. Condon et al. [7] further extended these ideas using asymptotic expansions in inverse powers of the oscillation parameter. Recent developments such as modulated Fourier expansions [17] and uniformly accurate multiscale time integrators [2] have pushed the boundary of accuracy and stability even in extreme frequency regimes.

These approaches demonstrate that while numerical methods can approximate highly oscillatory dynamics effectively, they typically rely on specialized analytical assumptions or frequency-scale separation. Consequently, their applicability to complex data-driven settings remains limited.

2.2 Neural ODEs and Limitations

With the rise of deep learning, several methods have emerged to infer and solve differential equations using neural networks. Among them, the Neural Ordinary Differential Equations (Neural ODEs)

framework [5] has attracted significant attention. Neural ODEs reinterpret the residual connections in ResNets as a forward Euler discretization of an ODE; in the continuous limit, the network's hidden state evolves according to

$$\frac{d\boldsymbol{h}(t)}{dt} = f_\theta(\boldsymbol{h}(t), t),$$

where f_θ is a neural network. This continuous-depth formulation allows end-to-end learning of dynamical systems from data.

Applications of Neural ODEs have been reported in diverse domains, including the modeling of ecological time-series data such as the oscillatory hare-lynx population [3, 8], and system identification in biology [16]. However, despite their elegance and flexibility, Neural ODEs struggle to represent highly oscillatory dynamics. Their learned vector fields tend to smooth out rapid temporal variations, leading to inaccurate phase and amplitude predictions when the underlying process exhibits fast oscillations or small-scale periodic structures.

Building on this framework, the Latent ODEs [15] were proposed as an extension designed to mitigate these limitations, particularly for irregular or complex time-series data. Latent ODEs combine an ODE-RNN encoder that processes irregularly sampled observations, a Neural ODE that evolves a continuous latent trajectory $h(t)$, and a decoder that reconstructs the observed signal from this trajectory, and thus were intended to compensate for the shortcomings of standard Neural ODEs. However, our experiments in Section 5 show that despite this more expressive architecture, Latent ODEs still struggle with highly oscillatory data; rapid changes in phase and amplitude are only partially captured, and the learned trajectories remain overly smoothed.

2.3 Motivation for OPCNet

To address this limitation, we introduce the Oscillatory Pattern Capturing Network (OPCNet)-a neural network architecture designed to accurately model and predict oscillatory patterns while preserving the dynamical structure of the underlying ODE. Leveraging the assumption from [14] that highly oscillatory trajectories can be viewed as solutions of perturbed linear ODEs such as (1.1), OPCNet learns both the governing dynamics and their oscillatory solutions simultaneously. In particular, unlike the unperturbed formulation, we explicitly adopt the perturbed ODE setting described in [14], where additional terms account for deviations from the nominal linear dynamics. This perspective enables OPCNet to capture complex oscillatory behavior beyond idealized linear solutions. This design allows the model to retain the interpretability of ODE-based systems while achieving superior performance over existing Neural ODE approaches on oscillatory data.

3 Architecture of OPCNet

In this section, we present our newly developed neural network model, called OPCNet, for fitting the highly oscillatory data with the proper ODE dynamics. OPCNet consists of two different networks: one is called the function network, and the other is called the solution network.

To enhance clarity and improve accessibility, we present OPCNet within a standard machine learning pipeline framework in Figure 1. Given time-series inputs $\boldsymbol{t}$ and corresponding observations $\boldsymbol{x}$, the solution network (SolNet) first predicts the trajectory SolNet($\boldsymbol{t}$). The time derivative of this prediction is then obtained via automatic differentiation and serves as a supervisory signal for the function network (FuncNet), which learns the underlying dynamics $f(t, \boldsymbol{x})$. The two networks are trained jointly by minimizing the loss function in (3.2), thereby enforcing consistency between the predicted solution and the learned dynamics. This unified learning pipeline enables OPCNet to simultaneously learn both the oscillatory trajectory and its governing ODE.

In order for the model to predict the dynamics inherent to the oscillatory data, the function network utilizes the ODE model (1.1) for general oscillatory curve in [14]. Meanwhile, the solution network learns the data itself to obtain the information of the rate of change in the data. Once the information

Algorithm 1 Training precedure of OPCNet

Input : $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \in \mathbb{R}^{n\times 2}$ where $\boldsymbol{x}_i = (x_i, y_i), \boldsymbol{t} \in \mathbb{R}^{n\times 1}, \alpha > 0$

Output : $\theta^m = (W^m_{\text{FuncNet}}, \boldsymbol{b}^m_{\text{FuncNet}}, W^m_{\text{SolNet}}, \boldsymbol{b}^m_{\text{SolNet}})$

1. $\lambda \leftarrow 1/\hat{\sigma}^2_{sol}$
2. $\mu \leftarrow 1/\hat{\sigma}^2_{der}$
3. $m \leftarrow 0$

while θ^m does not convergence **do**

1. $m \leftarrow m + 1$
2. $\boldsymbol{f} \leftarrow \text{FuncNet}(\boldsymbol{x}, \boldsymbol{t})$
3. $\boldsymbol{x}_{\text{pred}} \leftarrow \text{SolNet}(\boldsymbol{t})$
4. $L_{\text{dynamics}} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \left\| \boldsymbol{f}_i - \left(\frac{d\boldsymbol{x}_{\text{pred}}}{dt}\right)^{(*)}_i \right\|^2$
5. $L_{\text{solution}} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \left\| \boldsymbol{x}_i - (\boldsymbol{x}_{\text{pred}})_i \right\|^2$
6. $L_{\text{Total}} \leftarrow \lambda L_{\text{solution}} + \mu L_{dynamics}$
7. $\theta^m \leftarrow \theta^{m-1} - \alpha \frac{\partial L_{\text{Total}}}{\partial \theta^{m-1}}$

end while

$(*)$ $\frac{d\boldsymbol{x}_{\text{pred}}}{dt}$ is obtained from backpropagation procedure of SolNet.

of the data is captured by the solution network, then the derivative of the data, computed by the backward process in the solution network, is used to update the the function network. Algorithm 1 summarizes the training procedure of OPCNet.

3.1 Function and Solution Networks

The function network is devised to learn the dynamics of the given highly oscillatory data, with the assumption that the dynamics of data can be explained by the solution of

$$\frac{d\boldsymbol{x}}{dt} = \frac{A}{\epsilon}\boldsymbol{x} + g(t, \boldsymbol{x}), \; A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{3.1}$$

for sufficiently small $\epsilon > 0$ and some nonlinear function g. For this purpose, we construct the network to contain two subnetworks capable of learning a small number $\epsilon > 0$ and its corresponding dynamics $\frac{A}{\epsilon}\boldsymbol{x} + g(t, \boldsymbol{x})$. We set the subnetwork for ϵ to be one linear regressor and the subnetwork for the dynamics to be a general fully connected neural network whose input is $\boldsymbol{x} = (x, y, t)$ and output is $f(t, \boldsymbol{x}) = \frac{A}{\epsilon}\boldsymbol{x} + g(t, \boldsymbol{x})$. The network first learns a small number ϵ from the initial guess ϵ_0 and learns the dynamics of data $f(t, \boldsymbol{x})$ with fully connected layers, as in Algorithm 2.

We define the solution network to be a simple fully connected layer whose input is the time t and output is the predicted solution at t of (3.1), summarized in Algorithm 3. Throughout this paper, we refer to the function network as FuncNet and the solution network as SolNet. Figure 1 depicts two networks in this subsection.

Algorithm 2 The forward propagation FuncNet $(\boldsymbol{x}, \boldsymbol{t})$ of FuncNet

Input : $\boldsymbol{x} \in \mathbb{R}^{n\times 2}, \boldsymbol{t} \in \mathbb{R}^{n\times 1}, \epsilon_0 > 0$

Parameters : $w_\epsilon, W_1, W_2, \cdots, W_L, \boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_L$

Output : $\boldsymbol{f} \in \mathbb{R}^{n\times 2}$

1. $\boldsymbol{g} \leftarrow \tanh\left(concat\left(\boldsymbol{x}, \boldsymbol{t}\right) W_1 + \boldsymbol{b}_1\right)$

   for $l = 2$ **to** $L - 1$ **do**

   2. $\boldsymbol{g} \leftarrow \tanh\left(\boldsymbol{g} W_l + \boldsymbol{b}_l\right)$

   end for

3. $\boldsymbol{g} \leftarrow \boldsymbol{g} W_L + \boldsymbol{b}_L$
4. $\epsilon \leftarrow \epsilon_0 w_\epsilon$
5. $\boldsymbol{f} \leftarrow \boldsymbol{g} + \frac{1}{\epsilon} A\boldsymbol{x},\ where\ A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

Algorithm 3 The forward propagation SolNet $(\boldsymbol{x}, \boldsymbol{t})$ of SolNet

Input : $\boldsymbol{t} \in \mathbb{R}^{n\times 1}$

Parameters : $W_1, W_2, \cdots, W_L, \boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_L$

Output : $\boldsymbol{x}_{\text{pred}} \in \mathbb{R}^{n\times 2}$

1. $\boldsymbol{x}_{\text{pred}} \leftarrow \tanh\left(\boldsymbol{t} W_1 + \boldsymbol{b}_1\right)$

   for $l = 2$ **to** $L - 1$ **do**

   2. $\boldsymbol{x}_{\text{pred}} \leftarrow \tanh\left(\boldsymbol{x}_{\text{pred}} W_l + \boldsymbol{b}_l\right)$

   end for

3. $\boldsymbol{x}_{\text{pred}} \leftarrow \boldsymbol{x}_{\text{pred}} W_L + \boldsymbol{b}_L$

3.2 Loss Function

The loss function of the model must be defined to account for both the solution and the dynamics simultaneously, so that the predicted dynamics can be corrected by the derivative of the predicted solution in (3.1). To this end, we define the loss function L as

$$L = \frac{1}{n}\sum_{i=1}^{n} \lambda \left\|\text{SolNet}\left(\boldsymbol{t}\right)_i - \boldsymbol{x}_i\right\|^2 + \mu \left\|\text{FuncNet}\left(\boldsymbol{x}, \boldsymbol{t}\right)_i - \frac{d}{dt}\text{SolNet}\left(\boldsymbol{t}\right)_i\right\|^2 \tag{3.2}$$

where $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \in \mathbb{R}^{n\times 2}$ denotes the given data, each data is $\boldsymbol{x}_i = (x_i, y_i)$, and λ and μ are hyperparameters controlling the relative weighting of the two loss components. The first term in the summation enforces the predicted solution SolNet $(\boldsymbol{t})$ to be as close as possible to the given data $\boldsymbol{x}$,

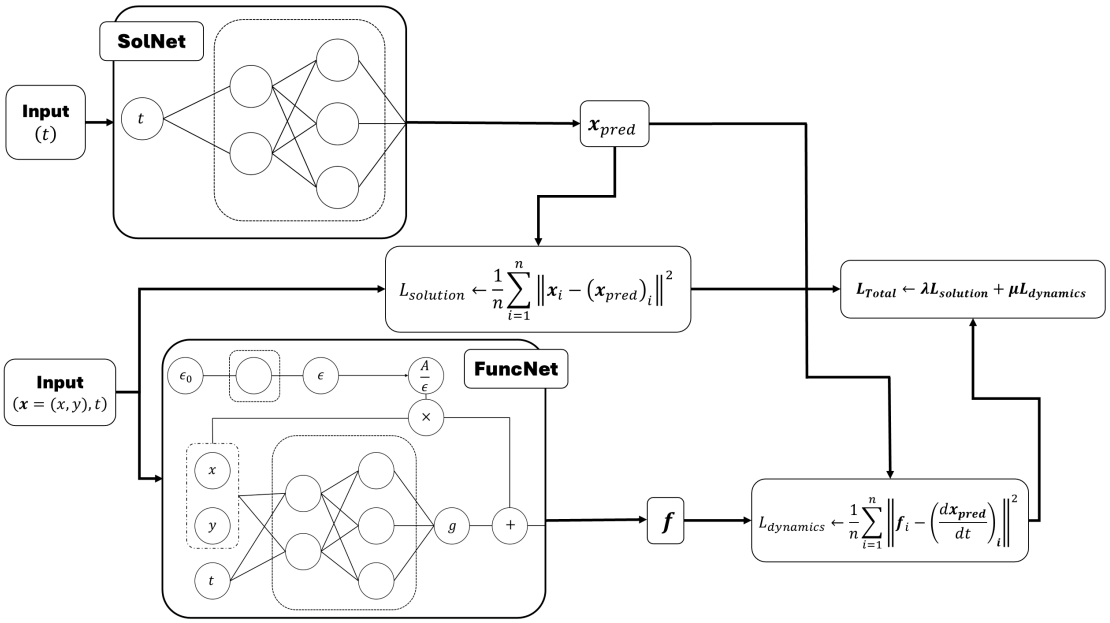


Figure 1: Schematic illustration of the OPCNet learning pipeline. The solution network (SolNet) predicts the trajectory from time input, whose derivative is obtained via automatic differentiation. This derivative guides the function network (FuncNet) to learn the underlying dynamics. Both networks are trained jointly through a coupled loss enforcing consistency between the predicted solution and dynamics.

while the second term encourages the predicted dynamics FuncNet $(\boldsymbol{x}, \boldsymbol{t})$ to match the derivative of the predicted solution, $\frac{d}{dt}\text{SolNet}(\boldsymbol{t})$. Here, the derivative $\frac{d}{dt}\text{SolNet}(\boldsymbol{t})$ can be directly obtained through the backpropagation process in SolNet. Note that our loss function L is the sum of two empirical second moments of residuals, one from the solution and the other from its derivative, which guides the selection of the relative weights λ and μ. To make this selection principled, we consider the following lemma, which provides a simple scale-balancing rationale for setting λ and μ in a consistent manner across different magnitudes of residuals.

Lemma 3.1 (Scale balancing). Let R_1 and R_2 be two residuals with empirical variances $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$, computed on a representative dataset (or minibatch). Consider the weighted quadratic loss

$$L = \lambda\, \mathbb{E}[R_1^2] + \mu\, \mathbb{E}[R_2^2].$$

If the residuals are rescaled as $R_1 \mapsto c_1 R_1$ and $R_2 \mapsto c_2 R_2$, the equivalent loss is obtained by replacing

$$(\lambda, \mu) \;\mapsto\; \left(\frac{\lambda}{c_1^2}, \frac{\mu}{c_2^2}\right).$$

Proof. Let

$$L(\lambda, \mu; R_1, R_2) \;=\; \lambda\, \mathbb{E}[R_1^2] + \mu\, \mathbb{E}[R_2^2].$$

If $R_1 \mapsto c_1 R_1$ and $R_2 \mapsto c_2 R_2$, then

$$L(\lambda, \mu; c_1 R_1, c_2 R_2) = \lambda\, \mathbb{E}[(c_1 R_1)^2] + \mu\, \mathbb{E}[(c_2 R_2)^2] = \lambda c_1^2 \mathbb{E}[R_1^2] + \mu c_2^2 \mathbb{E}[R_2^2].$$

Hence

$$L(\lambda, \mu; c_1 R_1, c_2 R_2) = L(\lambda c_1^2,\ \mu c_2^2;\ R_1, R_2),$$

which is equivalent to the statement that scaling the residuals by (c_1, c_2) is the same as replacing the weights by $(\lambda, \mu) \mapsto (\lambda/c_1^2, \mu/c_2^2)$ when one wishes to express the new loss in the original residual variables. □

The lemma directly implies that the relative scaling of residuals can be compensated by inverse-square weighting of their empirical variances. In practice, we approximate $\mathbb{E}[R_i^2] \approx \hat{\sigma}_i^2$ using representative data and choose the weights $\lambda = \alpha/\hat{\sigma}_1^2$ and $\mu = \alpha/\hat{\sigma}_2^2$ for some positive constant α, such that each term $\lambda\mathbb{E}[R_1^2]$ and $\mu\mathbb{E}[R_2^2]$ contributes approximately the same magnitude α to the total loss. This yields a scale-invariant and empirically balanced formulation. Without loss of generality, we set $\alpha = 1$, that is,

$$\lambda = \frac{1}{\hat{\sigma}_{\text{sol}}^2}, \qquad \mu = \frac{1}{\hat{\sigma}_{\text{der}}^2}, \tag{3.3}$$

where $\hat{\sigma}_{\text{sol}}$ and $\hat{\sigma}_{\text{der}}$ denote the empirical standard deviations of the residuals associated with the solution and its derivative, respectively.

4 Theoretical Insights of OPCNet

In this section, we endeavor to provide theoretical support for why OPCNet, although designed to capture the intrinsic structure of oscillatory data, achieves superior stability and generalization compared with conventional continuous-time baselines such as Neural ODE models, which learn an unconstrained black-box vector field $f_\theta(t, x)$ and obtain predictions by numerically integrating $\dot{x} = f_\theta(t, x)$. To support this claim, we focus on three complementary mechanisms: (i) an inductive bias that explicitly separates fast and slow dynamics, (ii) stability induced by derivative coupling between the solution and function networks, and (iii) a short-time consistency bound whose quality does not deteriorate as the oscillation frequency increases.

4.1 Inductive Bias

Neural ODEs approximate a full vector field $f_\theta(t, \boldsymbol{x})$ and thus must represent both fast oscillatory and slow modulatory components of the underlying dynamics. When the governing system exhibits strong scale separation, the model faces stiffness that inflates the number of function evaluations (NFEs) and complicates optimization. In contrast, OPCNet introduces an explicit fast-slow decomposition that embeds the oscillatory structure within its architecture. The following proposition characterizes its theoretical advantage.

Proposition 4.1 (Fast-Slow Decomposition Advantage). Let the target dynamics be

$$\dot{\boldsymbol{x}}(t) = \frac{A}{\varepsilon^*}\boldsymbol{x}(t) + g^*(t, \boldsymbol{x}(t)), \qquad A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

where $0 < \varepsilon^* \ll 1$, and suppose g^* is Lipschitz and band-limited to frequencies $\omega_g \ll 1/\varepsilon^*$. Then there exists an OPCNet whose parameter complexity depends only on the bandwidth of g^*, independent of ε^*.

Proof. Let $\boldsymbol{x}(t) = R_{t/\varepsilon^*}\boldsymbol{y}(t)$ with $R_\theta = e^{\theta A}$. Then

$$\dot{\boldsymbol{y}}(t) = R_{-t/\varepsilon^*}\, g^*\big(t, R_{t/\varepsilon^*}\boldsymbol{y}(t)\big),$$

whose spectral content remains $O(\omega_g)$. Hence, $\boldsymbol{y}(t)$ evolves on a slow time scale, and an OPCNet that explicitly encodes the fast rotation through its learned parameter ε needs only to approximate this slow residual term. Consequently, its parameter complexity depends solely on the bandwidth of g^* and does not scale with $1/\varepsilon^*$. □

In contrast, a standard Neural ODE must directly represent the full high-frequency field $A\boldsymbol{x}/\varepsilon^* + g^*$, forcing either increased representational bandwidth (capacity scaling with $1/\varepsilon^*$) or smaller integration steps (NFEs scaling with $1/\varepsilon^*$) to maintain comparable accuracy. More precisely, Neural ODEs exhibit two typical failure modes in the small-ε regime: (i) increased training time or numerical instability caused by the large number of NFEs required for stiff integration, and (ii) overfitting or excessive smoothing when the model capacity is insufficient to capture fast oscillations. By explicitly factoring out the stiff component, OPCNet avoids both issues and achieves stable learning with substantially lower computational cost.

4.2 Stability by Derivative Coupling

A distinguishing feature of OPCNet is the derivative-matching term in the training loss,

$$L = \frac{1}{n}\sum_{i=1}^{n} \lambda \|\mathrm{SolNet}(t_i) - x_i\|^2 + \mu \|\mathrm{FuncNet}(t_i, x_i) - \tfrac{d}{dt}\mathrm{SolNet}(t_i)\|^2,$$

which enforces consistency between the learned vector field and the temporal derivative of the learned solution.

Proposition 4.2 (Derivative-Consistent Regularization). Assume FuncNet is L_f-Lipschitz in x. Let $\eta(t) = \|\mathrm{FuncNet}(t, \boldsymbol{x}(t)) - \frac{d}{dt}\mathrm{SolNet}(t)\|$ and suppose $\sup_t \eta(t) \le \eta$. If $\tilde{\boldsymbol{x}}(t)$ is the trajectory obtained by integrating $\dot{\tilde{\boldsymbol{x}}} = \mathrm{FuncNet}(t, \tilde{\boldsymbol{x}})$ with $\tilde{\boldsymbol{x}}(0) = \mathrm{SolNet}(0)$, then

$$\|\tilde{\boldsymbol{x}}(t) - \mathrm{SolNet}(t)\| \le \frac{\eta}{L_f}\big(e^{L_f t} - 1\big), \qquad t \ge 0.$$

Proof. Let $e(t) = \tilde{\boldsymbol{x}}(t) - \mathrm{SolNet}(t)$. Then

$$\dot{e}(t) = \mathrm{FuncNet}(t, \tilde{\boldsymbol{x}}(t)) - \mathrm{FuncNet}(t, \mathrm{SolNet}(t)) + \big[\mathrm{FuncNet}(t, \mathrm{SolNet}(t)) - \tfrac{d}{dt}\mathrm{SolNet}(t)\big].$$

By the Lipschitz continuity of FuncNet, $\|\dot{e}(t)\| \le L_f \|e(t)\| + \eta(t)$. Applying Gronwall's inequality yields

$$\|e(t)\| \le e^{L_f t}\int_0^t e^{-L_f s}\eta(s)\, ds \le \frac{\eta}{L_f}(e^{L_f t} - 1),$$

which gives the desired bound. □

Standard Neural ODE training focuses on matching integrated trajectories to observed data, without explicitly enforcing consistency between the learned vector field and the temporal derivative of a separate parametric solution. In contrast, OPCNet introduces this derivative-coupling mechanism, which naturally aligns the two learned representations. This additional structure acts as a form of regularization, promoting stability and improving extrapolation by ensuring that the dynamics predicted by the vector field remain coherent with the learned solution manifold.

4.3 Short-time Consistency Bound independent of $1/\varepsilon$

Combining the inductive-bias and derivative-consistency effects yields a short-time consistency bound for OPCNet, which crucially avoids factors that blow up as $\varepsilon^* \to 0$.

Theorem 4.3 (Consistency of OPCNet Dynamics). Let the true dynamics be $\dot{\boldsymbol{x}}^* = \frac{A}{\varepsilon^*}\boldsymbol{x}^* + g^*(t, \boldsymbol{x}^*)$, with g^* Lipschitz. Suppose FuncNet and SolNet approximate g^* and $\boldsymbol{x}^*$ with uniform error ϵ_{app}, and training achieves total loss $L \le \eta^2$. Then for any finite $T > 0$,

$$\sup_{t\in[0,T]} \|\tilde{\boldsymbol{x}}(t) - \boldsymbol{x}^*(t)\| \le C(T)\,(\epsilon_{\mathrm{app}} + \eta),$$

where $C(T)$ depends on Lipschitz constants but *not* on $1/\varepsilon^*$.

Proof. Write

$$\tilde{\boldsymbol{x}}(t) - \boldsymbol{x}^*(t) = \big(\tilde{\boldsymbol{x}}(t) - \mathrm{SolNet}(t)\big) + \big(\mathrm{SolNet}(t) - \boldsymbol{x}^*(t)\big).$$

The first term is controlled by derivative consistency: since $\|\dot{\tilde{\boldsymbol{x}}}(t) - \dot{\mathrm{SolNet}}(t)\| \le \eta$ and SolNet is Lipschitz with constant L, Gronwall's inequality gives

$$\|\tilde{\boldsymbol{x}}(t) - \mathrm{SolNet}(t)\| \le \eta\, e^{LT}.$$

The second term is bounded by the uniform approximation error, $\|\mathrm{SolNet}(t) - \boldsymbol{x}^*(t)\| \le \epsilon_{\mathrm{app}}$. Combining these by the triangle inequality yields

$$\|\tilde{\boldsymbol{x}}(t) - \boldsymbol{x}^*(t)\| \le \epsilon_{\mathrm{app}} + \eta\, e^{LT}.$$

Since the fast term A/ε^* is represented explicitly, the constant e^{LT} does not depend on $1/\varepsilon^*$, completing the proof. □

Error and stability bounds for NODEs typically involve norms or Lipschitz constants of the learned field f_θ. When f_θ must represent a large linear term A/ε^*, these constants grow with $1/\varepsilon^*$, producing bounds that degrade for highly oscillatory systems. OPCNet removes this dependence by structural design.

5 Numerical experiments

In this section, we perform several numerical experiments to validate the performance of OPCNet. Numerical results with highly oscillatory synthetic data, generated by harmonic oscillators and Bessel functions, are given in the first two subsections. In the last subsection, Hare-Lynx data from the predator-prey interaction [13], M4 hourly data [12], and ECG data [9], which are well-known to be highly oscillatory in real-world applications, are chosen for testing the performance of the network. For comparison, we compare all results for the synthetic data with the results from neural ODEs (NODE) and latent ODEs. Since all examples in this section are time-dependent problems, we set $\boldsymbol{x}$ to be the collection of $\boldsymbol{x}_i = (t_i, x_i)$, thus the FuncNet learns the dynamics of each example based on

$$\frac{d\boldsymbol{x}}{dt'} = \frac{A}{\epsilon}\boldsymbol{x} + g\left(t', \boldsymbol{x}\right),$$

introducing a fictitious time variable t'. All numerical experiments were implemented in Python and executed on a personal computer equipped with a 3.60 GHz CPU and 16 GB of memory. The hyperparameters used for training OPCNet are summarized in Table 1. The source code and datasets used in these experiments are publicly available at `https://github.com/SangukHwang0117/OPCNet`.

Table 1: Training hyperparameters for OPCNet.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	1×10^{-3}
Loss function	As described in Section 3.2
Mini-batch size	Full batch (no mini-batching)
Early stopping	Not used
Training iterations	Up to 200,000 (typically converged by 60,000)
Network depth	4 layers
Activation function	Hyperbolic tangent (tanh)
Parameter initialization	Default PyTorch initialization

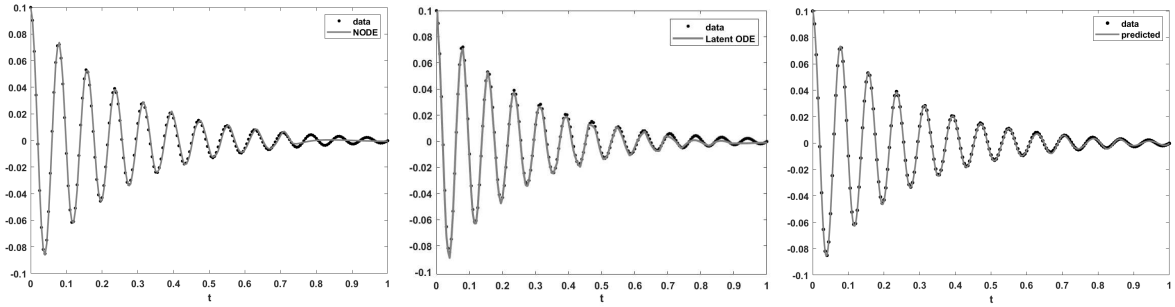


Figure 2: Numerical results of NODE (left), Latent ODE (middle), and OPCNet (right) for data generated from a damped harmonic oscillator. Although NODE and Latent ODE appear to capture the oscillatory pattern reasonably well, both fail to accurately predict the tail region of the oscillation. In contrast, OPCNet successfully captures the full oscillatory behavior of the data.

5.1 Harmonic Oscillators

For the first example, we consider highly oscillatory data obtained from the simple underdamped harmonic oscillator. More precisely, the data is chosen to be a discrete part of the solution of the following second-order ODE:

$$\frac{1}{8}y'' + y' + 802y = 0,\ y(0) = 0.1,\ y'(0) = -0.4,$$

whose explicit form is $y(t) = 0.1\cos(80t)\exp(-4t)$. We sample the data uniformly from the domain $[0, 1]$ with $\Delta t = \frac{1}{200}$.

Numerical results for neural ODE, latent ODE, and OPCNet are shown in Figure 2. While neural ODE (left) and latent ODE (middle) fit the data reasonably well, they fail to accurately capture the oscillatory behavior at later times. In contrast, OPCNet (right) successfully describes the oscillatory patterns over the entire time interval. Table 2 reports the l_2 error computed from 200 randomly sampled points in the domain $[0, 1]$. OPCNet achieves significantly lower errors than the other methods in both test cases, demonstrating its superior ability to capture highly oscillatory behavior.

	NODE [5]	**Latent ODE [15]**	**OPCNet**
Harmonic oscillators	0.512 ± 0.064	0.222 ± 0.019	$\mathbf{0.018 \pm 0.011}$
Bessel functions	0.964 ± 0.013	0.152 ± 0.215	$\mathbf{0.0030 \pm 0.0002}$

Table 2: L_2 errors (mean $\pm$ standard deviation) computed from 200 randomly sampled points in $[0, 1]$ for harmonic oscillator and Bessel function data. OPCNet achieves significantly lower errors compared to NODE and latent ODE in both cases.

5.2 Bessel Functions

Now, we select Bessel functions of the first kind of the order v

$$J_v(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!\Gamma(v+n+1)}\left(\frac{x}{2}\right)^{(2n+v)}$$

to generate highly oscillatory but non-monotonic data, which is known to be a canonical solution of the following Bessel differential equation :

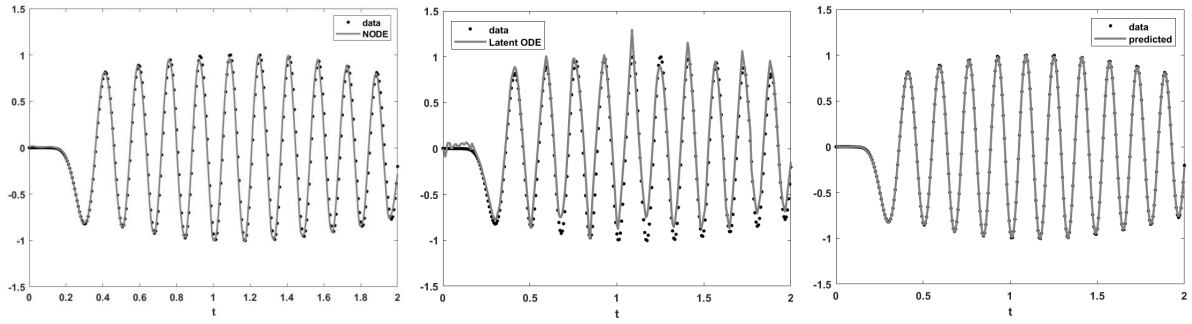


Figure 3: Numerical results of NODE (left), Latent ODE (middle), and OPCNet (right) for data generated from the Bessel function. While NODE appears to capture the oscillatory pattern reasonably well, both NODE and Latent ODE exhibit a slight overall leftward shift. In particular, the Latent ODE curve shows spurious oscillations near the initial region and fails to reproduce the shape around the peak, whereas the OPCNet result does not suffer from these issues.

$$x^2y'' + xy' + (x^2 - v^2)y = 0.$$

In order to generate oscillatory data with high frequency and diverse patterns, we consider a weighted Bessel function by a sinusoidal signal, that is,

$$y(t) = -\sin(t)\, J_v(40t).$$

We set $v = 10$ and the data is sampled uniformly from $[0, 2]$ with $\Delta t = \frac{2}{300}$.

The left and middle panels in Figure 3 show the predicted profiles from NODE and latent ODE, respectively. Although both methods appear to fit most of the data, they fail to accurately capture the oscillatory behavior near the peak regions and in the initial phase, and their predictions deteriorate as time progresses. In contrast, the result (right) in Figure 3 demonstrates that the proposed model successfully captures the oscillatory pattern regardless of the shape of the data. This qualitative observation is further supported by the quantitative results in Table 2, where the proposed model achieves significantly lower l_2 errors than both NODE and latent ODE. In particular, the error is reduced by more than an order of magnitude, highlighting the effectiveness of the proposed approach in modeling highly oscillatory data.

To verify that the performance of the model does not depend on inherent properties of the data, such as periodicity, we consider more complex oscillatory data obtained from the weighted Bessel function but with irregular perturbations on the periodicity. It can be seen from Figure 4 that the proposed model (left) predicts oscillatory patterns of the data relatively well, while the result from NODE (right) does not.

5.3 Real-World Applications

In the previous sections, we examined whether OPCNet can effectively capture oscillatory patterns using synthetic data generated from numerical simulations. However, real-world data typically contain various irregularities because they arise from actual observations. To further assess the robustness and practical applicability of our model, we conducted additional experiments on three real-world datasets: Hare-Lynx, M4-Hourly, and ECG. These experiments evaluate whether OPCNet can accurately capture the oscillatory behavior present in these datasets.

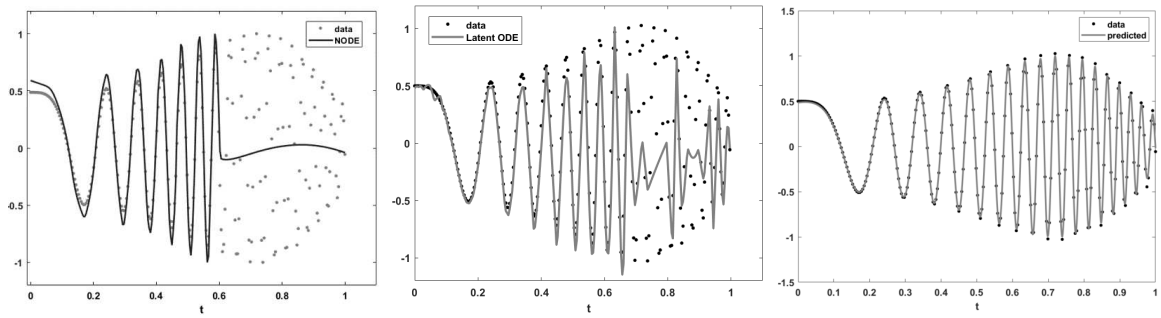


Figure 4: Numerical results of NODE (left), Latent ODE (middle), and OPCNet (right) for Bessel-like data. Both NODE and Latent ODE fail to capture the oscillatory pattern beyond approximately $t = 0.6$, whereas OPCNet successfully reproduces the oscillations despite their highly complex shape.

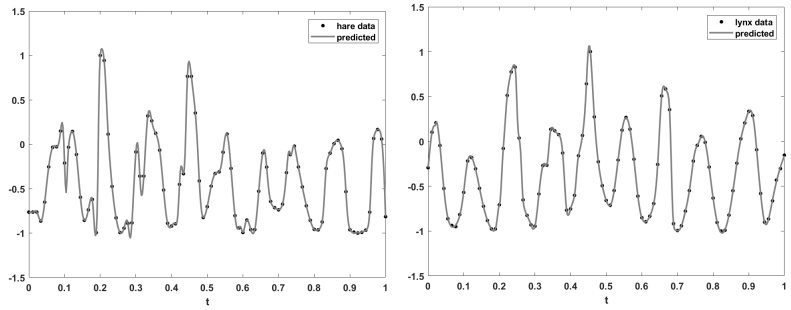


Figure 5: Numerical results of OPCNet for the data from the predator-prey interaction of Hare (left) and Lynx (right).

5.3.1 Hare-Lynx data

For the predator–prey population data, we use the well-known irregularly oscillatory time series from the interaction between the Snowshoe hare and the Canadian lynx. Researchers have observed that Canadian lynxes typically consume about two Snowshoe hares every three days, indicating that the populations of the two species are closely related. In addition, hare and lynx populations are known to exhibit dramatic changes every 8–11 years. A commonly used model for this interaction is

$$\frac{dH}{dt} = rH\left(1 - \frac{H}{k}\right) - \frac{aHL}{c+H},$$

$$\frac{dL}{dt} = a\frac{bHL}{c+H} - dL,$$

where a is the lynx–hare predation rate, b is the lynx–hare conversion factor, c is the lynx–hare Michaelis constant, d is the lynx death rate, k is the hare carrying capacity, and r is the hare reproduction rate. In this example, we use 100 years of hare and lynx population data. Figure 5 shows the results of predicting the oscillatory behavior of the hare (left) and lynx (right) populations using the proposed model. These results indicate that the proposed model fits the hare and lynx population data well.

5.3.2 M4-hourly data

The second experimental dataset is the M4 hourly dataset, which is one of the time series datasets that Makridakis uses for his competition, known for the M4 competition. The M4 competition is a

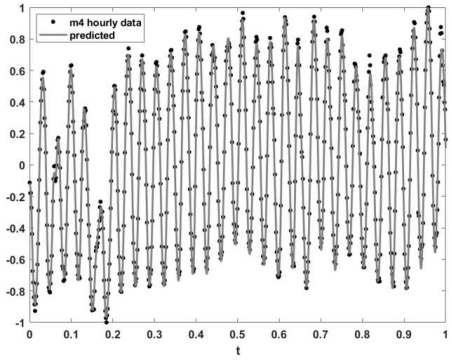


Figure 6: Numerical result of OPCNet for the M4 hourly data set.

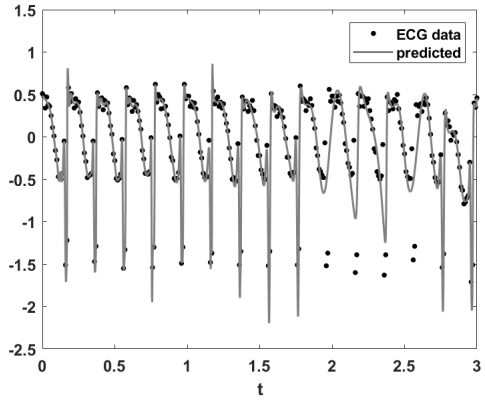


Figure 7: Numerical result of OPCNet on the ECG dataset.

prediction competition using time series, and there are various time series data in addition to hourly data. Since the hourly dataset is relatively oscillatory, we select the M4 hourly dataset as the second practical example. Figure 6 shows the result of the proposed model for the dataset. As shown in the figure, the dataset is highly oscillatory and the proposed model captures the oscillatory pattern of the data well. However, it can be seen that our model has some failure moments near some points at the peak of the oscillation.

5.3.3 ECG data

The heartbeat generates small electrical signals that regulate contraction and relaxation. These signals are amplified and recorded through electrodes attached to the arms, chest, and legs, and the resulting measurements are referred to as electrocardiogram (ECG) data. ECG waveforms arise from the cardiac contraction–relaxation cycle and exhibit a periodic pattern, which is widely used to assess and predict various heart diseases. ECG data typically have large amplitudes and, because they are measured through electrodes, are often susceptible to noise. In addition, ECG waveforms contain periodic downward deflections that drop significantly below the surrounding values. Figure 7 shows the result of applying OPCNet to ECG data, indicating that the model still struggles to fully capture such signals. In particular, OPCNet is able to reproduce some of the irregular downward segments but fails to track these patterns consistently over time.

6 Conclusion and future works

In this work, we provide a neural network-based model to fit highly oscillatory data. The proposed model consists of two subnetworks: FuncNet and SolNet. FuncNet is a network for learning the dynamics of the data with the assumption that the oscillatory data can be seen as the solution of the linear ODE with perturbations (3.1), and SolNet is devised to learn the solution of (3.1). With the properly defined loss function, the proposed model predicts the oscillatory patterns inherent in the data by communicating between the information from FuncNet and SolNet.

Although the results obtained from the proposed model are promising, there is a drawback that the way of selecting hyperparameters λ, μ in the loss function is ambiguous. We believe that theoretical support for selecting optimal values of hyperparameters is available in the near future. Due to limitations of time and resources, we leave mathematical analysis on the hyperparameters to future works.

Funding information

The work of Byungjoon Lee is partially supported by The Catholic University of Korea, Research Fund, 2026, and the National Research Foundation of Korea (NRF) grant funded by the Korean government (RS-2025-24523778). The work of Hyeong-Ohk Bae is partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2026-25475912). The work of SangUk Hwang is partially supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP-2026-RS-2024-00438207).

Acknowledgement

The authors are grateful to anonymous reviewers for their careful reading and valuable comments.

References

[1] FJ Agocs, WJ Handley, AN Lasenby, and MP Hobson. Efficient method for solving highly oscillatory ordinary differential equations with applications to physical systems. *Physical Review Research*, 2(1):013030, 2020.

[2] Weizhu Bao, Xuanchun Dong, and Xiaofei Zhao. Uniformly accurate multiscale time integrators for highly oscillatory second order differential equations. *J. Math. Study*, 47(2):111–150, 2014.

[3] Willem Bonnaffé, Ben C Sheldon, and Tim Coulson. Neural ordinary differential equations for ecological and evolutionary time-series analysis. *Methods in Ecology and Evolution*, 12(7):1301–1315, 2021.

[4] JE Bunder and Anthony J Roberts. Numerical integration of ordinary differential equations with rapidly oscillatory factors. *Journal of computational and applied mathematics*, 282:54–70, 2015.

[5] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[6] David Cohen, Tobias Jahnke, Katina Lorenz, and Christian Lubich. Numerical integrators for highly oscillatory hamiltonian systems: a review. *Analysis, modeling and simulation of multiscale problems*, pages 553–576, 2006.

[7] Marissa Condon, Alfredo Deano, and Arieh Iserles. On asymptotic-numerical solvers for differential equations with highly oscillatory forcing terms. Technical report, DAMTP Tech. Rep. 2009/NA05, 2009.

[8] Steven A Frank. Optimizing differential equations to fit data and predict outcomes. *Ecology and Evolution*, 13(3):e9895, 2023.

[9] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[10] Arieh Iserles. Think globally, act locally: solving highly-oscillatory ordinary differential equations. *Applied Numerical Mathematics*, 43(1-2):145–160, 2002.

[11] Marianna Khanamiryan. Quadrature methods for highly oscillatory linear and nonlinear systems of ordinary differential equations: Part i. *BIT Numerical Mathematics*, 48:743–761, 2008.

[12] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[13] Eugene Pleasants Odum, Gary W Barrett, et al. *Fundamentals of ecology*, volume 3. Saunders Philadelphia, 1971.

[14] Linda R Petzold. An efficient numerical method for highly oscillatory ordinary differential equations. *SIAM Journal on Numerical Analysis*, 18(3):455–479, 1981.

[15] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[16] Alireza Yazdani, Lu Lu, Maziar Raissi, and George Em Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS computational biology*, 16(11):e1007575, 2020.

[17] Xiaofei Zhao. Uniformly accurate multiscale time integrators for second order oscillatory differential equations with large initial data. *BIT Numerical Mathematics*, 57(3):649–683, 2017.