# Metalearning Generalizable Dynamics from Trajectories

Qiaofeng Li, Tianyi Wang, Vwani Roychowdhury, and M. K. Jawed

Phys. Rev. Lett. **131**, 067301 — Published 10 August 2023

DOI: 10.1103/PhysRevLett.131.067301

# Meta-learning generalizable dynamics from trajectories

Qiaofeng Li[1,2,3], Tianyi Wang[2], Vwani Roychowdhury[2,*], and M.K. Jawed[1,*]
[1] *Dept. of Mechanical and Aerospace Engineering, University of California, Los Angeles, CA 90095, USA*
[2] *Dept. of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095, USA*
[3] *Dept. of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

We present the interpretable meta neural ordinary differential equation (iMODE) method to rapidly learn generalizable (i.e. not parameter-specific) dynamics from trajectories of multiple dynamical systems that vary in their physical parameters. The iMODE method learns meta-knowledge, the functional variations of the force field of dynamical system instances without knowing the physical parameters, by adopting a bi-level optimization framework: an outer level capturing the common force field form among studied dynamical system instances and an inner level adapting to individual system instances. *A priori* physical knowledge can be conveniently embedded in the neural network architecture as inductive bias, such as conservative force field and Euclidean symmetry. With the learned meta-knowledge, iMODE can model an unseen system within seconds, and inversely reveal knowledge on the physical parameters of a system, or as a *Neural Gauge* to "measure" the physical parameters of an unseen system with observed trajectories. iMODE can be generally applied to a dynamical system of an arbitrary type or number of physical parameters and is validated on bistable, double pendulum, Van der Pol, Slinky, and reaction-diffusion systems.

Building predictive models of dynamical systems is a central challenge across diverse disciplines of science and engineering. Traditionally, this has been achieved by first manually deriving the governing equations with carefully chosen state variables and then fitting the undetermined physical parameters using observed data, e.g., [1–3]. In order to avoid the painstaking formulation of analytical equations, researchers have recently leveraged advances in machine learning and the data-fitting power of neural networks (NNs) to make the modeling process both automatic and more expressive [4]. This is achieved by either adopting the conventional physics-based approach as a starting point and then replacing various components with data-driven modules [5, 6], or directly learning discrete dynamics using autoregressive models from high-dimensional observations [7–9]. These works, while promising, need to fit dedicated models separately for different system instances with different parameters, which limits a model's applicability to one specific instance.

In this letter, our goal is to learn meta-knowledge, the form of dynamics that is unrestricted to specific physical parameters or initial/boundary conditions, on dynamical systems to reveal physical insights [10–12] and to significantly improve the generalization ability of data-driven models. Specifically, we learn the shared dynamics form from the trajectories generated by a series of dynamical system instances in spite of their diversified behaviors in data, *without knowing the system parameters*. This separates our work from Refs. [13, 14] and Neural Operators [15–18], in which true parameters should be provided. This goal aligns with that of multi-task meta-learning [19], which aims to leverage the similarities between different tasks to enable better generalization and efficient adaptation to unseen tasks.

We propose an efficient and interpretable method to model a family of dynamical systems using their observed trajectories, by combining gradient-based meta-learning (GBML) [20–24] with neural ordinary differential equations (NODE) [6, 25, 26]. In recognizing that the systems have shared dynamics form and varying physical parameters, we separate the model parameters into two parts: the *shared parameters* that capture the shared form of dynamics, i.e. the meta-knowledge, and the *adaptation parameters* that account for variations across system instances. The method generalizes well on unseen systems from the same family, and the adaptation parameters show good interpretability. The intrinsic dimension of the varying system parameters can be estimated by analyzing the adaptation parameters. Given ground truth of the system parameters, simple correspondence can be established between the adaptation parameters and actual physical parameters through diffeomorphism, which can be utilized as a "*Neural Gauge*" to measure properties of new systems through observed trajectories. We name our method interpretable meta neural ODE (iMODE).

In a general autonomous second-order system, the state of the system $\mathbf{y}$ contains the position (generalized coordinates) $\mathbf{x}$ and the velocity $\dot{\mathbf{x}}$. The dynamics of the second-order system is expressed equivalently as a first-order system

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{M}^{-1}\mathbf{F}_{\boldsymbol{\phi}}(\mathbf{y}) \end{bmatrix}, \text{ where } \mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix} \tag{1}$$

where $\mathbf{F}_{\boldsymbol{\phi}}$ is the force vector containing all the internal and external forces, and $\mathbf{M}$ is the mass matrix. With a set of physical parameters $\boldsymbol{\phi}$, the force function $\mathbf{F}(\cdot)$ dictates the dynamics of the system, which determines a unique trajectory $\mathbf{y}(t)$ given an initial condition $\mathbf{y}(t_0)$. In the remainder of the letter, without loss of generality, mass is normalized to an identity matrix, i.e., $\mathbf{M} = \mathbf{I}$. It should be noted that, with this general formulation, the proposed method is applicable to systems other than second-order (e.g. first-order diffusion-reaction systems. See Supplemental Material (SM) [27] S6) and can be

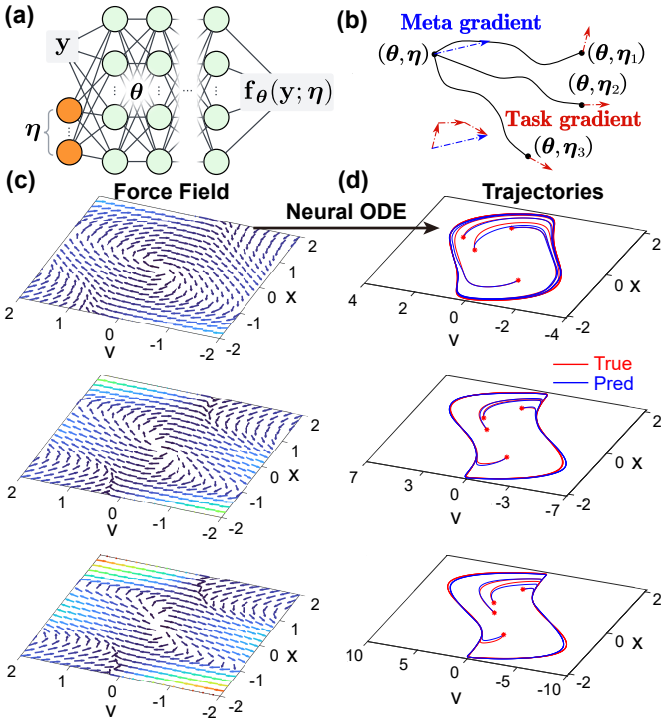


Figure 1. (color online). (a) In iMODE, a neural module $\mathbf{F}_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ takes the concatenation of system state $\mathbf{y}$ and the adaptation parameters $\boldsymbol{\eta}$ and generates the estimated force as output. (b) The bi-level iteration process in the iMODE method. The NN weights $\boldsymbol{\theta}$ are shared across system instances while $\boldsymbol{\eta}$ is adapted for each instance. The meta gradient w.r.t. $\boldsymbol{\theta}$ aggregates the gradients evaluated with instance-adapted $\boldsymbol{\eta}$. (c) Examples of estimated force field $\mathbf{f}_{\boldsymbol{\theta}}(\cdot;\boldsymbol{\eta})$ for Van der Pol system instances that differ in their $\epsilon$ parameter (in ascending order from top to bottom). The estimation quality is further evaluated through the trajectories generated by the fields as shown in (d). (d) The estimated force field can be used to predict system trajectories for unseen initial conditions through integration (Eq. (2)). The signature limit cycles of Van der Pol systems are faithfully reproduced.

easily extended to non-autonomous systems (the forcing term should be provided).

Trajectories are collected from multiple system instances into a dataset $\mathcal{D}$. Consider $N_\mathrm{s}$ instances that share the dynamics form $\mathbf{F}_{\boldsymbol{\phi}}(\cdot)$, but have distinct physical parameters, $\{\boldsymbol{\phi}_1,\ldots,\boldsymbol{\phi}_{N_\mathrm{s}}\}$ respectively. From each system instance, $N_\mathrm{tr}$ trajectories are observed, each containing observations across $T$ time steps. In summary, $\mathcal{D} = \left\{\{\mathbf{y}_{i,j}(t_k)\}_{k=0}^{T} | i=1,\ldots,N_\mathrm{s},\ j=1,\ldots,N_\mathrm{tr}\right\}$. The data-driven model is trained on $\mathcal{D}$, knowing which trajectories are from the same system instance (i.e. given both the index $i$ and $j$ of trajectories), but is not given the knowledge of $\{\boldsymbol{\phi}_i\}_{i=1}^{N_\mathrm{s}}$. Take the pendulum system as an example. An instance is a pendulum with a specific arm length (since the inertia is normalized), therefore $\boldsymbol{\phi}$ includes only the arm length. A trajectory contains the location and speed of the pendulum during a time period.

In our framework, a neural network $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{y};\boldsymbol{\eta})$ (Fig. 1(a). See SM [27] S7 for detailed description) replaces $\mathbf{F}_{\boldsymbol{\phi}}(\mathbf{y})$ in Eq. (1) to approximate the observed trajectories, where $\boldsymbol{\eta}$ is adapted to each system instance such that with a certain $\boldsymbol{\eta}_i$, $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{y};\boldsymbol{\eta}_i)$ approximates the force function of the $i$th system instance $\mathbf{F}_{\boldsymbol{\phi}_i}(\mathbf{y})$. After training, $\boldsymbol{\eta}$ becomes a proxy for the physical parameters $\boldsymbol{\phi}$. $\boldsymbol{\theta}$ is the model parameters that capture the functional form of dynamics shared across system instances. The predicted trajectory starting from an initial condition $\mathbf{y}_0$ is given by integration (the 5th-order Dormand-Prince-Shampine solver is used throughout this letter to compute integrals)

$$\widehat{\mathbf{y}}(t,\mathbf{y}_0,\boldsymbol{\theta},\boldsymbol{\eta}) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}_{\boldsymbol{\theta}}\left(\widehat{\mathbf{y}}(\tau);\boldsymbol{\eta}\right) \mathrm{d}\,\tau \tag{2}$$

For brevity, we denote the trajectory $\mathbf{y}_{i,j}(t)$ as $\mathbf{y}_{i,j}$, the corresponding prediction $\widehat{\mathbf{y}}(t,\mathbf{y}_{i,j}(t_0),\boldsymbol{\theta},\boldsymbol{\eta})$ as $\widehat{\mathbf{y}}_{i,j}(\boldsymbol{\theta},\boldsymbol{\eta})$, and use $\|\,\mathbf{y}_{i,j} - \widehat{\mathbf{y}}_{i,j}(\boldsymbol{\theta},\boldsymbol{\eta})\,\|^2$ to denote $\sum_{k=0}^{T}\left(\mathbf{y}_{i,j}(t_k) - \widehat{\mathbf{y}}(t_k,\mathbf{y}_{i,j}(t_0),\boldsymbol{\theta},\boldsymbol{\eta})\right)^2$, the squared difference between $\mathbf{y}_{i,j}$ and $\widehat{\mathbf{y}}_{i,j}(\boldsymbol{\theta},\boldsymbol{\eta})$ across all time steps.

The goal of the modeling is formulated as a bi-level optimization (Fig. 1(b)),

$$\textbf{outer:}\quad \min_{\boldsymbol{\theta}}\ \widetilde{\mathcal{L}}(\boldsymbol{\theta}) = \frac{1}{N_\mathrm{s}}\sum_{i=1}^{N_\mathrm{s}} \mathcal{L}_i(\boldsymbol{\theta},\boldsymbol{\eta}_i^{(m)}),\ \text{where} \tag{3}$$

$$\mathcal{L}_i(\boldsymbol{\theta},\boldsymbol{\zeta}) = \frac{1}{N_\mathrm{tr}T}\sum_{j=1}^{N_\mathrm{tr}} \|\,\mathbf{y}_{i,j} - \widehat{\mathbf{y}}_{i,j}(\boldsymbol{\theta},\boldsymbol{\zeta})\,\|^2, \tag{4}$$

$$\textbf{inner:}\quad \boldsymbol{\eta}_i^{(l+1)} = \boldsymbol{\eta}_i^{(l)} - \alpha\nabla_{\boldsymbol{\eta}}\mathcal{L}_i(\boldsymbol{\theta},\boldsymbol{\eta}_i^{(l)}),\ \ \boldsymbol{\eta}_i^{(0)} = \boldsymbol{\eta} \tag{5}$$

where the inner-level involves an $m$-step gradient descent adapting $\boldsymbol{\eta}$ for each instance, while the outer-level finds the optimal initialization for $\boldsymbol{\theta}$. $\alpha$ is the inner-level stepsize and $\boldsymbol{\eta}_i^{(m)}$ is the adaptation parameters for the $i$th system instance after $m$ steps of adaptation. For short, we denote such $i$th adaptation result as $\boldsymbol{\eta}_i$. Note that $\boldsymbol{\eta}_i$ depends on both $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ as shown in Eq. (5). To avoid higher-order derivatives, we simplify such dependency following the first-order Model Agnostic Meta-Learning (first-order MAML) [20] and use the outer-level step as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\beta}{N_\mathrm{s}}\sum_{i}\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta},\boldsymbol{\eta}_i),\ \ (\text{assuming that } \frac{\partial\boldsymbol{\eta}_i}{\partial\boldsymbol{\theta}} = \mathbf{0}) \tag{6}$$

where $\beta$ is the outer-level stepsize. At both the inner-level and outer-level, the gradient calculation for functions involving integrals is enabled by NODE [6, 25, 26].

As shown in Fig. 1(c), $\mathbf{f}_{\boldsymbol{\theta}}(\cdot;\boldsymbol{\eta})$ specifies a force field that morphs as $\boldsymbol{\eta}$ changes. Note that $m$ is normally quite small (e.g. 5), so given trajectories of a previously unseen system, $\boldsymbol{\eta}$ can be efficiently updated with few gradient steps, adapting the NN to specify a force field explaining behaviors of the new system, which is one order-of-magnitude faster compared to training from scratch (Fig. 3(a)). Trajectories with arbitrary initial conditions can be inferred

based on the force field (Fig. 1(d)). Generally speaking, there is no restriction to the dimension of $\boldsymbol{\phi}$, i.e. the number of physical parameters of the system. However, as it increases, more system instances (larger training dataset) and hence longer training time of iMODE is expected.

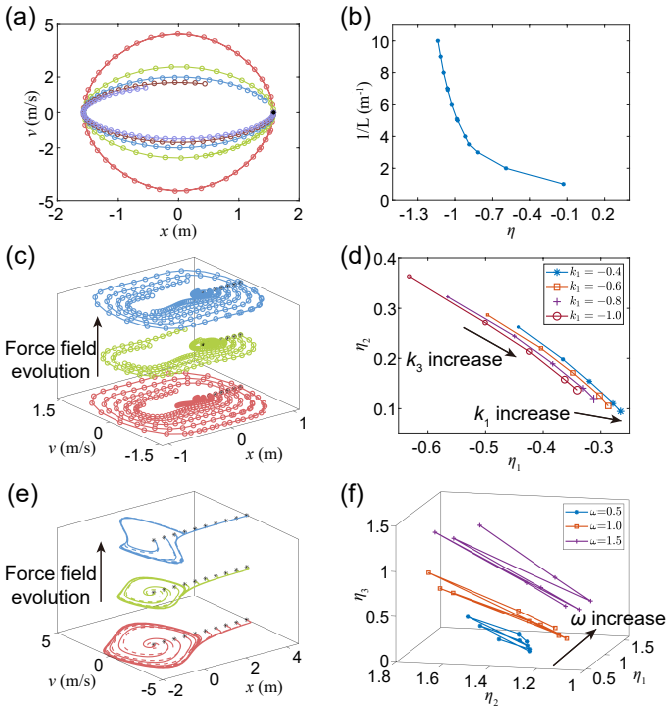


Figure 2. (color online). (a) The meta-learning results for the pendulum. The iMODE trajectory prediction (circles) with different arm lengths (different colors) match those of the ground truth (solid lines). (b) The learned $\boldsymbol{\eta}$ is in good correlation with the effective stiffness of different pendulums ($1/L$). (c) The predicted trajectories (circles) match those of ground truth (solid lines) with different initial conditions (black stars) and different system parameters (different colors) for the bistable system. (d) Two principal axes can be identified from the latent space of the learned $\boldsymbol{\eta}$, each regarding the variation of one physical parameter. (e) Similar to (c) but for the Van der Pol system. (f) The principal axis regarding to the variation of $\omega$ for the Van der Pol system.

First we validate the modeling capability of the iMODE algorithm on 3 cases: oscillating pendulum, bistable oscillator, and Van der Pol system (see SM [27] S1 for detailed description). The oscillating pendulum has 1 physical parameter, i.e. the arm length (rotational inertia normalized). Fig. 2(a) shows that the predicted trajectories using task-adapted NNs match the ground truth of each system. Fig. 2(b) shows that the learned $\boldsymbol{\eta}$ correlates well with the effective stiffness of the pendulum, i.e. $1/L$. Effectively $\boldsymbol{\eta}$ acts as a proxy of the true arm length and can be used to infer such parameters of unseen systems.

The bistable system has a potential energy function controlled by 2 parameters $k_1$ and $k_3$. Its potential energy has two local minima, or potential wells. When the initial conditions vary, the bistable system can oscillate intra-well or inter-well. Fig. 2(c) shows that the task adapted trajectories ($m = 5$) match the ground truth well. Fig. 2(d) shows that the identified $\boldsymbol{\eta} \in \mathbb{R}^2$ has two principal axes, along which $k_1$ and $k_3$ increases. As mentioned, $\boldsymbol{\eta}$ is effectively a proxy for $k_1$ and $k_3$. Later we will show that the mapping from $\boldsymbol{\eta}$ to $\boldsymbol{\phi} = [k_1, k_3]$ can be constructed as a diffeomorphism with NODE.

The Van der Pol system has 3 physical parameters $\boldsymbol{\phi} = [\epsilon, \delta, \omega]$. It exhibits limit cycles due to the negative damping for small oscillation amplitudes. Fig. 2(e) shows that the evolution of limit cycles due to the change of physical parameters is well predicted. Three principal axes can be found for the identified $\boldsymbol{\eta}$. The one for $\omega$ is shown in Fig. 2(f) (see SM [27] S1 for the other two). Again, the mapping from $\boldsymbol{\eta}$ to $\boldsymbol{\phi} = [\epsilon, \delta, \omega]$ can be constructed as a diffeomorphism.

The fast adaptation of iMODE is demonstrated with the bistable systems in Fig. 3(a). The iMODE is able to adjust the adaptation parameters in 5 steps to learn the dynamics of unseen system instances. Training the same network from scratch (random initialization) on the same test dataset requires much more epochs to achieve a comparable accuracy. When evaluated on trajectories with unseen initial conditions (of distinct conserved energies), the performance of iMODE-adapted models outperforms that of the model trained from scratch by several orders of magnitude, showing superior generalization ability with limited data (see SM [27] S3 for a more disparate comparison when data is scarce).

Second, we demonstrate the combination of the iMODE algorithm with certain physics priors for efficient modeling of more complicated systems. Since iMODE does not assume specific architecture of $\mathbf{f}_{\boldsymbol{\theta}}$, a wide range of neural network architectures can be adopted to embed appropriate inductive biases. For example, in bistable and the following wall bouncing and Slinky systems, the assumption of conservative force is introduced, where the system dynamics is determined by a potential energy function. Accordingly we take a specific form for the neural force estimator $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}; \boldsymbol{\eta}) = \partial E_{\boldsymbol{\theta}}(\mathbf{x}; \boldsymbol{\eta})/\partial \mathbf{x}$. That is, the NN first outputs an energy field and then induces the force field from the energy field (using auto-differentiation [28]). In this way, iMODE enables the fast adaptation of not only the force field but also the potential energy field for the parametric systems. The learned potential energy functions are shown in Fig. 3(b). The wall bouncing system has a potential energy well that is not a linear function of the well's (half-)width $w$ or the particle position $x$ (see SM [27] S2). However, iMODE is still able to approximate the discontinuous energy function. $\boldsymbol{\eta}$ correlates well with the true width $w$, i.e., we can control the width of the potential energy well by tuning $\boldsymbol{\eta}$ (see SM [27] S2).

The intrinsic dimension $d_{\boldsymbol{\phi}}$ of the physical parameters $\boldsymbol{\phi}$ can be estimated by applying Principal Component Analysis (PCA) to the collection of the $\boldsymbol{\eta}$ vectors, each

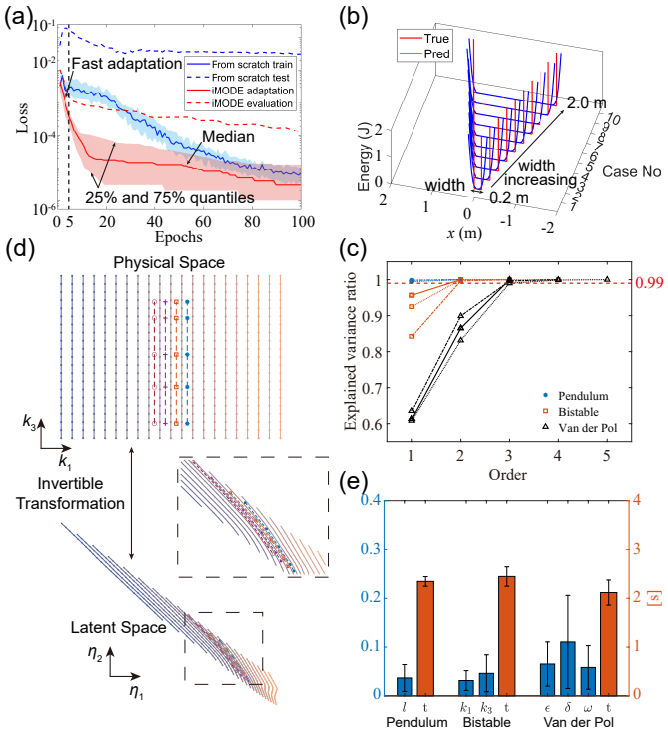


Figure 3. (color online). (a) Comparison of iMODE test adaptation v.s. training from scratch on (50) unseen bistable system instances with randomly chosen physical parameters. iMODE demonstrates fast adaptation and good generalization within the first 5 adaptation steps. (b) The true and learned potential energy functions for the wall bouncing system. The width of the potential well increases as the adaptation parameter increases. (c) The number of top PCA components that preserve a significant portion ($> 99\%$) of the variance gives a good estimation of the dimension of true physical parameters. (d) The diffeomorphism constructed by NODE for the bistable system. It shows how a grid in the physical space is continuously deformed into the latent space of adaptation parameters. (e) The mean error and computation time of *Neural Gauge* for 100 systems with randomly generated unseen parameters.

adapted to one of the system instances. Using an "elbow" method on the cumulative explained variance ratio curve of the PCA result, the number of the principal components that explain the most of the variance has a good correspondence with $d_\phi$, as long as $d_{\boldsymbol{\eta}} \geq d_{\boldsymbol{\phi}}$, where $d_{\boldsymbol{\eta}}$ is the dimension chosen for $\boldsymbol{\eta}$. The PCA results on the pendulum, bistable system, and Van der Pol system are shown in Fig. 3(c) (see SM [27] S4 for the results of other systems). Taking the Van der Pol system as an example, $d_{\boldsymbol{\eta}}$ is respectively 3, 4 or 5 for the three curves with triangle markers. In all three cases, the first three principal components explain more than 99% of the variance, and the "elbow" appears at 3, which corresponds well with the fact that $d_{\boldsymbol{\phi}} = 3$ for the Van der Pol system.

*Neural Gauge*: Without labels for the physical parameters, iMODE develops a latent space of adaptation parameters accounting for the variations in dynamics among system instances. Given the physical parameter labels of the system instances in the training data, a mapping between the space of the physical parameters and the latent space can be established so that the corresponding physical parameters can be estimated given any point in the latent space. iMODE therefore can be exploited as a "Neural Gauge" to identify the physical parameters of unseen system instances, and the establishment of such mappings can be seen as a calibration process. We propose to construct such mappings as diffeomorphism, which can be learned with a neural ODE $\mathrm{d}\mathbf{z}(t)/\mathrm{d}t = \mathbf{g}_{\boldsymbol{\xi}}(\mathbf{z})$, such that starting from a given point in the latent space, $\mathbf{z}(0) = \boldsymbol{\eta}_i$, the state $\mathbf{z}$ at $t = 1$ gives the corresponding physical parameters, $\mathbf{z}(1) = \boldsymbol{\phi}_i$, $i = 1, \ldots, N_s$. For simplicity, the dimension of the latent space and that of the physical parameter space are assumed to match ($d_{\boldsymbol{\eta}} = d_{\boldsymbol{\phi}}$); see SM [27] S5 for more general treatment. $\mathbf{g}_{\boldsymbol{\xi}}$ is a NN whose weights are optimized by $\boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} \sum_i \|\mathbf{z}_i(1) - \boldsymbol{\phi}_i\|_2^2$.

Figure 3(d) shows the learned diffeomorphism for the bistable system. The diffeomorphism establishes a bijection between the physical space and the latent space so that a grid in the physical parameter space can be continuously transformed into the adaptation parameter space (see SM [27] S9). The visualization highlights the advantages of diffeomorphism mapping: (1) The transformation is smooth so that the local geometric structure is preserved; (2) Invertible transformation allows a better interpretation of the latent space compared to degenerating ones.

After constructing the diffeomorphism, we test the physical parameter identification performance on 100 randomly selected unseen instances (with random physical parameters). The identification error and time cost are shown in Fig. 3(e) for pendulum, bistable, and Van der Pol systems. The end-to-end identification starting from data-feeding normally takes around 2 seconds (see SM [27] S8 for details).

*Complex systems*: We further demonstrate that iMODE applies to complex systems with two examples: a 40-cycle Slinky (Fig. 4) and a reaction-diffusion system described by the Kolmogorov-Petrovsky-Piskunov (KPP) equation. In the Slinky case, we embed Euclidean invariance for the energy field and induce equivariance for the force field. iMODE is able to learn from 4 Slinky cases (of Young's modulus 50, 60, 70, and 80 GPa, dropping under gravity from a horizontal initial configuration with both ends fixed) and then quickly generalize (with 2 adaptation steps) to an unseen Slinky (of Young's modulus 56 GPa) under unseen initial and boundary conditions. In the KPP equation case, iMODE is able to learn the reaction term with different reaction strength coefficients in 5 adaptation steps under Neumann boundary conditions and directly generalize to unseen Dirichlet boundary conditions. Refer to SM [27] S6 for details.

We have presented the iMODE method, i.e., interpretable meta NODE. As a major difference from existing

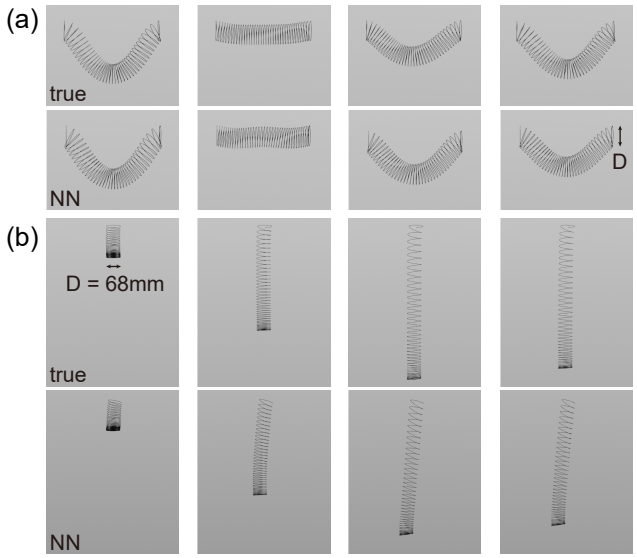


Figure 4. (color online). (a) The testing performance of the iMODE model on an unseen Slinky (of an unseen Young's modulus) with the same boundary condition as the training dataset. The top row is ground truth and the bottom row is the iMODE model prediction at 0.28, 0.47, 0.65, 0.83 s (left to right). The mean squared error of the 3D Slinky reconstruction over the entire trajectory is $8.0 \times 10^{-4}\mathrm{m}^2$. (b) The testing performance of the iMODE model on unseen initial and boundary conditions. The top row is ground truth and the bottom row is the iMODE model prediction at 0.15, 0.32, 0.48, 0.65 s (left to right). The mean squared error of the 3D Slinky reconstruction over the entire trajectory is $12.6 \times 10^{-4}\mathrm{m}^2$.

NN-based methods, iMODE learns meta-knowledge on a family of dynamical systems, specifically the functional variation of the derivative (force) field. It constructs a parametrized functional form of the derivative field with a shared NN across system instances and latent adaptation parameters adapted for different instances. The NN and adaptation parameters are learned from the difference between the ground truth and the solution calculated by an appropriate ODE solver. We have validated with various examples the generalizability, interpretability, and fast adaptation ability of the iMODE method. iMODE could open new possibilities for numerous potential applications. Two examples are autonomous modeling of dynamical systems where the underlying physics is difficult to express as an explicit function of controllable experiment parameters: (1) the force-deformation constitutive relation of cells as a function of ion concentrations in the culture medium, and (2) agile maneuvering of dynamical systems where the timely knowledge on the interaction between the dynamical system and its external environment is required, such as robotic control in a rapidly changing and partially understood environment.

The following research grants were gratefully acknowledged: NSF (CMMI-2053971) for Q.L., T.W., V.R., and M.K.J.; and NSF (CAREER-2047663, CMMI-2101751, and OAC-2209782) for M.K.J.

* V.R.: vwani@ee.ucla.edu, M.K.J.: khalidjm@seas.ucla.edu

[1] J. Sprakel, S. B. Lindström, T. E. Kodger, and D. A. Weitz, Stress enhancement in the delayed yielding of colloidal gels, Physical Review Letters **106**, 248303 (2011).
[2] M. K. Jawed, P. Dieleman, B. Audoly, and P. M. Reis, Untangling the mechanics and topology in the frictional response of long overhand elastic knots, Physical Review Letters **115**, 118302 (2015).
[3] R. Alert, A. Martínez-Calvo, and S. S. Datta, Cellular sensing governs the stability of chemotactic fronts, Physical Review Letters **128**, 148101 (2022).
[4] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learning, Nature Reviews Physics , 1–19 (2021).
[5] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, Journal of Machine Learning Research **19**, 1 (2018).
[6] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, Neural ordinary differential equations, Advances in Neural Information Processing Systems (2018).
[7] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences **113**, 3932 (2016).
[8] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, Data-driven discovery of coordinates and governing equations, Proceedings of the National Academy of Sciences **116**, 22445 (2019).
[9] B. Chen, K. Huang, S. Raghupathi, I. Chandratreya, Q. Du, and H. Lipson, Automated discovery of fundamental variables hidden in experimental data, Nature Computational Science **2**, 433 (2022).
[10] R. Iten, T. Metger, H. Wilming, L. d. Rio, and R. Renner, Discovering physical concepts with neural networks, Physical Review Letters **124**, 010508 (2020).
[11] Z. Liu and M. Tegmark, Machine learning conservation laws from trajectories, Physical Review Letters **126**, 180604 (2021).
[12] Z. Liu and M. Tegmark, Machine learning hidden symmetries, Physical Review Letters **128**, 180201 (2022).
[13] K. Lee and E. J. Parish, Parameterized neural ordinary differential equations: Applications to computational physics problems, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **477**, 20210162 (2021).
[14] S. Desai, M. Mattheakis, H. Joy, P. Protopapas, and S. J. Roberts, One-shot transfer learning of physics-informed neural networks, in *ICML 2022 2nd AI for Science Workshop* (2022).
[15] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, Neural operator: Graph kernel network for partial differential equations (2020), arXiv:2003.03485 [cs.LG].
[16] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, Fourier neural operator for parametric partial differential equations, in *International Conference on Learning Represen-*

*tations* (2020).

[17] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nature Machine Intelligence **3**, 218–229 (2021).

[18] S. Wang, H. Wang, and P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, Science Advances **7**, eabi8605 (2021).

[19] H. Wang, H. Zhao, and B. Li, Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation, in *Proceedings of the 38th International Conference on Machine Learning* (PMLR, 2021) pp. 10991–11002.

[20] C. Finn, P. Abbeel, and S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in *International conference on machine learning* (PMLR, 2017) pp. 1126–1135.

[21] A. Nichol, J. Achiam, and J. Schulman, On first-order meta-learning algorithms, CoRR **abs/1803.02999** (2018), 1803.02999.

[22] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, Online meta-learning, in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019) pp. 1920–1930.

[23] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, Meta-learning with implicit gradients, in *Advances in Neural Information Processing Systems*, Vol. 32 (Curran Associates, Inc., 2019).

[24] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, Rapid learning or feature reuse? Towards understanding the effectiveness of MAML, in *International Conference on Learning Representations* (2019).

[25] R. T. Q. Chen, B. Amos, and M. Nickel, Learning neural event functions for ordinary differential equations, International Conference on Learning Representations (2021).

[26] Q. Li, T. Wang, V. Roychowdhury, and M. Jawed, Rapidly encoding generalizable dynamics in a Euclidean symmetric neural network, Extreme Mechanics Letters , 101925 (2022).

[27] See supplemental material for details on iMODE training, wall bouncing system, double pendulum system, dimension determination of latent space, demonstration on complex systems, and movies on diffeomorphism, which includes Ref. [29].

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019) pp. 8024–8035.

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, Densely connected convolutional networks, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017) pp. 2261–2269.