


Machine learning for the identification of phase transitions in interacting agent-based systems: A Desai-Zwanzig example

Nikolaos Evangelou ^{1,2} Dimitris G. Giovanis,^{3,4} George A. Kevrekidis,²

Grigorios A. Pavliotis,⁵ and Ioannis G. Kevrekidis^{1,2,*}

¹*Department of Chemical and Biomolecular Engineering,*

Johns Hopkins University, 3400 North Charles Street, Baltimore, Maryland 21218, USA

²*Department of Applied Mathematics and Statistics, Johns Hopkins University, 3400 North Charles Street, Baltimore, Maryland 21218, USA*

³*Department of Civil and Systems Engineering, Johns Hopkins University, 3400 North Charles Street, Baltimore, Maryland 21218, USA*

⁴*Hopkins Extreme Materials Institute, Johns Hopkins University, 3400 North Charles Street, Baltimore, Maryland 21218, USA*

⁵*Department of Mathematics, Imperial College London, London SW7 2AZ, United Kingdom*



(Received 1 November 2023; revised 29 April 2024; accepted 17 June 2024; published 15 July 2024)

Deriving closed-form analytical expressions for reduced-order models, and judiciously choosing the closures leading to them, has long been the strategy of choice for studying phase- and noise-induced transitions for agent-based models (ABMs). In this paper, we propose a data-driven framework that pinpoints phase transitions for an ABM—the Desai-Zwanzig model—in its mean-field limit, using a smaller number of variables than traditional closed-form models. To this end, we use the manifold learning algorithm Diffusion Maps to identify a parsimonious set of data-driven latent variables, and we show that they are in one-to-one correspondence with the expected theoretical order parameter of the ABM. We then utilize a deep learning framework to obtain a conformal reparametrization of the data-driven coordinates that facilitates, in our example, the identification of a single parameter-dependent ordinary differential equation (ODE) in these coordinates. We identify this ODE through a residual neural network inspired by a numerical integration scheme (forward Euler). We then use the identified ODE—enabled through an odd symmetry transformation—to construct the bifurcation diagram exhibiting the phase transition.

DOI: [10.1103/PhysRevE.110.014121](https://doi.org/10.1103/PhysRevE.110.014121)

I. INTRODUCTION

Complex dynamic phenomena are ubiquitous in natural sciences, social sciences, and engineering [1–4]. In many cases, their study has been performed using agent-based models (ABMs), also known as interacting particle systems (IPSS). The dynamics of those models in the thermodynamic (mean-field) limit undergo phase transitions—bifurcations in nonlinear dynamics terminology [5,6]. The exploration of the dynamics of these systems typically involves extensive numerical simulation scenarios for a very large number, N , of interacting agents that can be truly challenging and impedes the widespread utilization of these models. Therefore, coarse-graining methodologies become necessary in order to reduce this complexity. Reducing the dimensionality of an ABM can be achieved by defining collective variables that are capable of accurately describing the full dynamics of such large systems in terms of a relatively small number of observables [7–9], or discovering those by data mining [10–12].

In our previous work [9], we developed and successfully tested a model reduction approach based on the cumulants of the single-agent probability distribution for such large ABM systems. The proposed coarse-graining framework was based on an analytical closure methodology of the infinite hierarchy of equations for the moments or, equivalently, cumulants of

the probability distribution of the infinite-dimensional system. The basic steps for building a reduced-order Desai-Zwanzig (DZ) model are as follows: (i) Consider the mean-field ansatz by writing the N -particle distribution function, the solution of the N -particle Fokker-Planck equation, as the product of one-particle distribution functions $\rho(x, t)$ [13]. (ii) Represent $\rho(x, t)$ either in terms of its moments (DZ [14]) or in terms of its Fourier coefficients (Smoluchowski/noisy Kuramoto model [15]), thus obtaining an infinite system of ordinary differential equations (ODEs) that is *exactly* equivalent to the McKean-Vlasov partial differential equation (PDE). Truncating the equations for the moments (or the Fourier coefficients), while choosing an appropriate closure scheme, gives the reduced-order model described and analyzed in Ref. [9]. However, selecting the correct closure scheme for the truncated moments' system is the *trickiest* part of the coarse-graining procedure.

This analytical approximation requires choosing the “correct” observable(s), the right level of observation/analytical closure, and—importantly—the level at which we attempt the closure. To overcome these constraints, this paper proposes a data-driven framework for studying phase transitions of ABMs by (i) discovering the coarse observables in a data-driven fashion, (ii) determining the level of closure, (iii) identifying the reduced dynamics directly from data, and (iv) utilizing the data-driven reduced model and applying an odd symmetry transformation to it for the construction of the bifurcation diagram. To this end, we

*Contact author: yannisk@jhu.edu

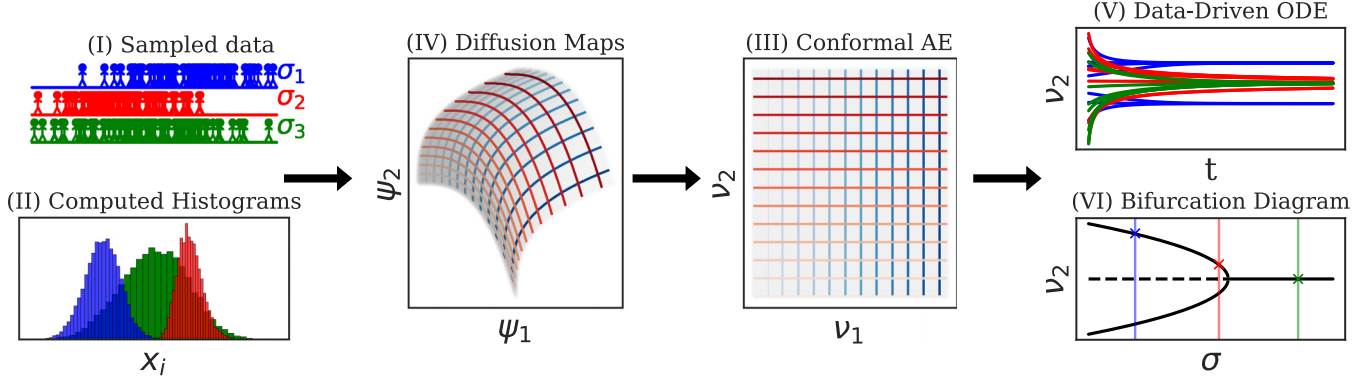


FIG. 1. Schematic of the overall workflow. (I) Sample data from the agent-based model (ABM) across multiple initial conditions and parameter values. (II) Compute histograms for each snapshot of the ABM. (III) Apply the diffusion maps algorithm on the computed histograms to discover a reduced latent embedding. (IV) Use a conformal autoencoder (AE) to find a conformal reparametrization of the latent space. (V) Identify a data-driven ODE in terms of the latent coordinate v_2 of the AE. (VI) Construct the bifurcation diagram (enabled via a symmetry transformation) in terms of the latent coordinate v_2 .

employ Diffusion Maps [16], a manifold learning algorithm, conformal autoencoders [17], and residual neural networks inspired by numerical integrators of ordinary differential equations [12,18–21]. This allows us to circumvent the difficulties arising from the choice of an analytical model and the selection of its closure. The main steps of our proposed data-driven workflow are illustrated in Fig. 1.

As an example, we apply our data-driven framework to the generalization of the DZ model with multiplicative noise considered in [9]; this very well-studied ABM features a second-order (continuous) phase transition, which enables us to compare the performance of our data-driven approach with well-known analytical and computational results. We emphasize, however, that our method is quite general, since it does not depend on the detailed features of the interaction, and it can be applied to many different ABMs that exhibit phase transitions in the thermodynamic limit.

II. THE AGENT-BASED MODEL

We consider a model describing an ensemble of N identical interacting agents subject to multiplicative noise. For this model, the agents are coupled via a mean reverting force, and a second-order phase transition exists at a critical temperature that can be calculated analytically ([9], Eq. (A12)). This transition appears as a symmetric pitchfork bifurcation of the mean-field dynamics. The dynamics of each agent, x_i , are described by a stochastic differential equation (SDE) of the form

$$dx_i = \left[-x_i^3 + (\alpha + \nu\sigma_m^2)x_i - \theta(x_i - \bar{x}) \right] dt + \sqrt{\sigma^2 + \sigma_m^2 x_i^2} dW_i, \quad (1)$$

where σ is the bifurcation parameter, θ denotes the interaction strength, α is a parameter characterizing the amplitude of the multiplicative noise, ν corresponds to different mathematical prescriptions of the SDEs (Itô, Stratonovich, etc.), and σ_m is a rectifying parameter: when $\sigma_m \neq 0$ the phase transition is pushed to higher values of σ . The agents are coupled through \bar{x} , which denotes the center of mass of the system (equal to the first moment M_1). Furthermore, dW_i , $i = 1, \dots, N$ denotes independent Brownian motions. The values of the parameters

were set to $\alpha = 1$, $\theta = 4$, $\sigma_m = 0.8$, $\nu = \frac{1}{2}$, and $N = 12\,000$ as in [9].

For this model, it was demonstrated in [9] that, away from the phase transition at $\sigma \cong 1.890$, the first moment, i.e., the order parameter of the system, is sufficient for an accurate description of the mean-field dynamics. As we get closer to the phase transition/the bifurcation, more moments need to be taken into account in order to accurately locate the bifurcation. More specifically, it was numerically demonstrated that at least a four-moment truncation is necessary in order to accurately predict the bifurcation/phase transition. In this work, by observing the eigenvalues of the Jacobian at the steady state (based on the moments equation proposed in [9]), we confirm that a clear separation of timescales prevails in the neighborhood of the bifurcation, and that the long-term dynamics live on a one-dimensional manifold. Our goal here, discussed in the next section, is to find a data-driven parametrization of this one-dimensional manifold, and identify the dynamics on it.

III. NUMERICAL RESULTS

A. Latent data-driven coordinates

We start by sampling data on an equidistant grid of 18 distinct values of the parameter σ in the interval $\sigma \in [0.5, 2.2]$. A more detailed description of the sampling strategy is discussed in Appendix A 1.

Given the sampled data from the ABM, we compute the first four moments M_1, M_2, M_3, M_4 as $M_k = \frac{1}{N} \sum_{i=1}^N x_i^k(t)$, where x_i^k denotes the i th agent at a fixed time raised in the power k , and M_k is the k th moment of the population from the sampled ABM data. The computation of the moments serves as a comparison between our data-driven framework and the one proposed in [9] where an analytical system of ODEs based on the four moments is proposed. The choice of four moment is also supported by studying the timescale separation of the analytical reduced model proposed in [9] between the four-moment truncation and higher-order moment truncations of the dynamics; see Appendix 5. In Fig. 2(a) we plot the first two moments M_1 and M_2 colored with the parameter σ . Based on this, we argue that the data

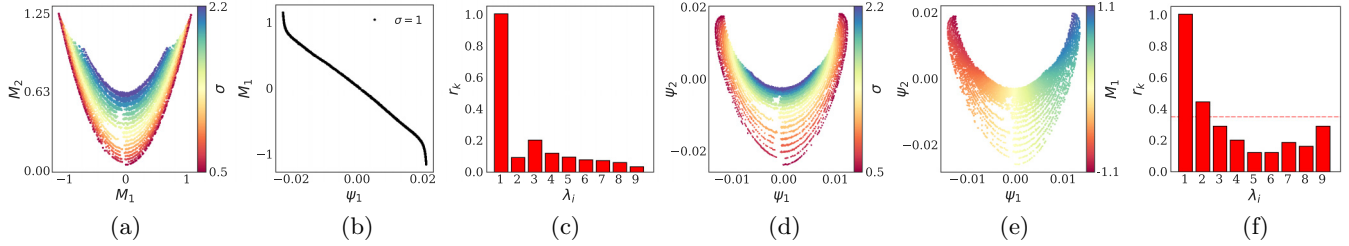


FIG. 2. (a) The first two moments (M_1, M_2) estimated from the ABM simulations colored with the parameter value σ at which they were obtained. (b),(c) Diffusion maps on histograms for a single value of the parameter σ : (b) one-to-one relation between leading histogram moment M_1 and leading diffusion maps coordinate ψ_1 ; (c) shows the residual (r_k) based on the local linear regression algorithm indicating that ψ_1 might be enough to parametrize the data (larger value of r_k). [(d)–(f)] Diffusion maps on collected histograms from the ABM simulation across multiple values of the parameter σ : (d),(e) show the nonharmonic diffusion map coordinates (ψ_1, ψ_2) colored with σ and M_1 , respectively; (f) shows the residual (r_k) based on the local linear regression algorithm indicating that ψ_1 and ψ_2 might be enough to parametrize the data (larger values of r_k).

are at least two-dimensional, and that the parameter σ appears to correlate with the moments M_1, M_2 .

We then proceed by computing Diffusion Maps (see Appendix 2 for a description of the algorithm) using data from the sampled ABM simulation. The Diffusion Maps algorithm was computed using (a) histograms obtained for a single value of the parameter σ , (b) histograms obtained across our grid of 18 distinct σ values, and (c) the four computed moments from data sampled across our 18 values of σ ; those computations are reported in Appendix 2c. We would like to reiterate that the diffusion maps computations on the moments are not necessary for the overall framework they serve as a comparison of our approach and the one by [9].

Computing Diffusion Maps (based on histograms) for single values of σ (far from the phase transition) gave a one-dimensional manifold parametrized by ψ_1 . In Fig. 2(b) we plot the leading Diffusion Maps coordinate ψ_1 against the first moment M_1 , and we demonstrate that they are one-to-one. This suggests that Diffusion Maps is capable of discovering a coordinate that is one-to-one with the known order parameter M_1 for this model [9,14,22]. The claim that the manifold is one-dimensional in this case is also supported from the results of the local-linear algorithm shown in Fig. 2(c); the residual r_k of the first eigenvector ψ_1 is larger than that of the remaining eigenvectors. In Appendix 2c we illustrate the relation of the Diffusion Maps coordinate with the subsequent moments M_2, M_3, M_4 .

We proceed with the Diffusion Maps computation on data sampled over multiple parameter values of σ . Details for the Diffusion Maps algorithm applied on histograms are provided in Appendix 2b. Here in Figs. 2(d) and 2(e) we show the two nonharmonic eigenvectors that parametrize different eigendirections ψ_1 and ψ_2 colored with the parameter σ and the first moment M_1 . The selection of the nonharmonic eigenvectors was made by using the local linear regression algorithm proposed in [23] and implemented by the *datafold* Python package [24]. The residual r_k of the first two eigenvectors, shown in Fig. 2(f), is greater than the residual of the remaining eigenvectors. This suggests that the first two eigenvectors (ψ_1, ψ_2) are nonharmonic, while the remaining ones are harmonic (functions) of the first two.

The Diffusion Maps results for the case of using collected data across multiple parameter values, shown in

Figs. 2(c), 2(d), and 2(e), suggest that the mean-field dynamics of the ABM live on a two-dimensional manifold. However, the parameter σ appears to correlate with the behavior of the moments observed at that σ value. This suggests that even though the data can be parametrized by two coordinates ψ_1, ψ_2 , one might be able to disentangle this coupled two-dimensional description into a factorized (a single state) \times (a parameter) description. In the next section, we illustrate how a data-driven reparametrization of the Diffusion Maps coordinates can be achieved that will allow us to disentangle the effect of the parameter σ from that of the latent variable.

B. Y-shaped conformal autoencoder

In this section, we illustrate the use of the Y-shaped conformal autoencoder, originally introduced in our previous work [17] as a means of disentangling the effect of different parameter combinations on model outputs in the context of parameter nonidentifiability. The Y-shaped conformal autoencoder seeks a reparametrization of the Diffusion Maps coordinates in which it disentangles the effect of the parameter σ from that of the latent coordinates of the autoencoder. A more detailed description of the overall framework, the different loss function components of the network, and the training procedure we followed are discussed in Appendix 3. A schematic of the Y-shaped conformal autoencoder is shown in Fig. 3(a). The coordinates ψ_1 and ψ_2 , obtained by computing the Diffusion Maps algorithm on estimated histograms of the ABM, were the ones used as input to the Y-shaped conformal autoencoder.

The bottleneck latent variables v_1, v_2 are shown in Figs. 3(b) and 3(c) colored with the parameter σ and the first moment M_1 (that is the known coarse variable for this model [9]). We see that the parameter σ appears to be varying only along v_1 compared to Fig. 2(d) in which σ varied across both ψ_1, ψ_2 . Similarly, M_1 varies only along v_2 . The latent variable v_1 shows a strong correlation with the parameter σ , as depicted by Fig. 3(d), indicating that they are effectively one-to-one. The conformality efficiency of the autoencoder becomes visually pronounced in Fig. 3(e) where level sets of v_1 and v_2 are plotted in the Diffusion Maps coordinates. Furthermore, the average value of the cosine of the angle,

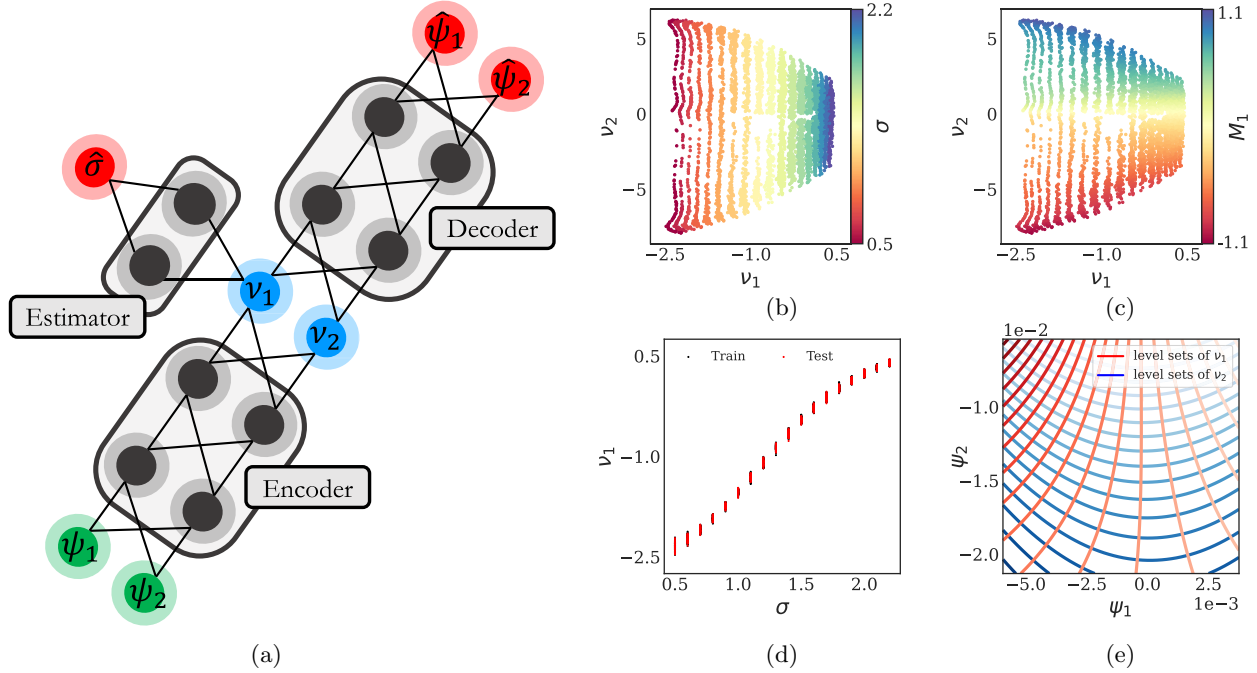


FIG. 3. (a) A schematic of the Y-shaped conformal autoencoder. The inputs to the network (ψ_1 and ψ_2) are shown as green nodes. The outputs of the autoencoder ($\hat{\psi}_1$ and $\hat{\psi}_2$) and the Estimator are shown as red nodes. The latent variables (v_1 and v_2) are shown as light blue nodes. (b),(c) The obtained latent coordinates v_1, v_2 colored with σ and M_1 , respectively. We can see that the σ appears to vary only across v_1 , compared to Fig. 2(c), in which σ varied across both ψ_1, ψ_2 . Similarly, M_1 appears to vary only across v_2 . (d) The parameter σ is plotted against the latent coordinate v_1 indicating a strong dependence. (e) Contour lines representing level sets of v_1 and v_2 are plotted in the Diffusion Maps (ψ_1, ψ_2) providing a visual illustration of the obtained conformality.

$\cos \theta = \frac{\nabla v_1 \cdot \nabla v_2}{\|\nabla v_1\| \|\nabla v_2\|}$, calculated over the test set, was found to be 5×10^{-4} .

This reparametrization of the latent coordinates is crucial since, as we show in the next section, it allows us to identify a single-state dimension ODE depending on σ (rather than a needlessly two-dimensional ODE).

In Appendix 3 we provide additional results obtained for the Y-shaped autoencoder illustrating the reconstruction of its input data (ψ_1, ψ_2). Remarkably, in addition, the Y-shaped autoencoder allows us to estimate σ from previously unseen histogram observation through v_1 .

C. Identifying parameter-dependent ODEs

Using the latent variable v_2 obtained from the Y-shaped conformal autoencoder as the state variable, we now identify a σ -dependent ODE; remember that v_2 is conformal to v_1 (and therefore to σ). The identification of the parameter-dependent ODE was achieved through a residual neural network inspired by the forward Euler numerical integration scheme. A schematic of the neural network is shown in Fig. 4(a). The inputs to this network are the state variable at time t , $v_2(t)$, and the parameter σ , and the output is the state variable evolved to time $t + h$, $v_2(t + h)$. We provide a more detailed description of the forward Euler network and the constructed loss function scheme in Appendix 4.

We test the ability of the ODE to produce accurate paths (trajectories), in v_2 space, for seven unseen values of the parameter $\sigma = \{0.57, 0.85, 1.11, 1.75, 1.9, 2.06, 2.25\}$; see Appendix 1 for more details. In Fig. 4(b) we illustrate

the predicted values $\hat{v}_2(t + h)$ from the Euler neural network against the ground truth (projected trajectories in v_2 generated by the ABM) for all seven unseen values of the parameter σ . The estimated MSE in this case was $\text{MSE} = 6 \times 10^{-4}$. In addition, in Fig. 4(c) we provide a visual comparison between trajectories simulated by the Euler neural network and by the ABM model (projected in v_2) for the test parameter values $\sigma = \{1.11, 1.75, 1.9\}$. The paths generated by the ABM and those generated by the neural network-identified ODE show excellent visual agreement across different values of σ and different initial conditions. This suggests that the identified right-hand side provides a successful approximation of the ground-truth dynamics.

D. Bifurcation diagram: Phase transition

In this section, we now proceed to test whether the parameter-dependent ODE identified in a data-driven way through our neural network can qualitatively and quantitatively capture the phase transition. To assess the robustness of our approach, we also estimate the error associated with the phase transition by training 5000 neural networks in parallel, using different splits of the data for training, validation, and testing. All hyperparameters were kept the same across models during this computation. For each of the trained networks given the identified right-hand side, we compute the steady states across a range of values of the parameter σ and construct the bifurcation diagram. For values of $\sigma \in [0.5, 2.2]$ a representative constructed bifurcation diagram is shown in Fig. 5(a). The bifurcation diagram shows that the

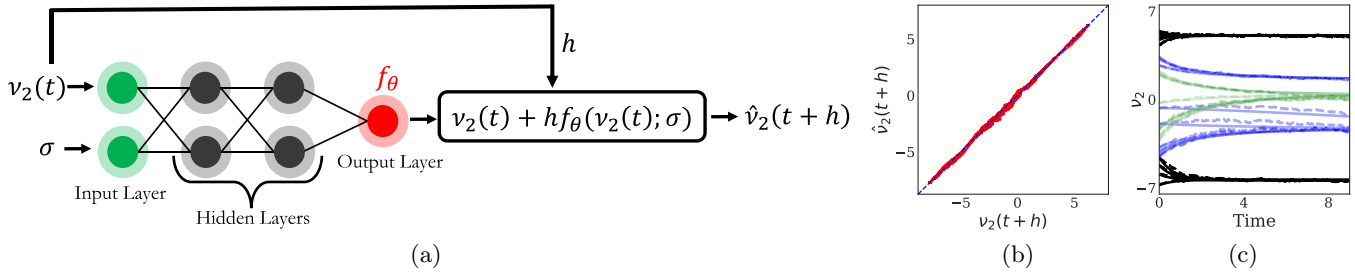


FIG. 4. (a) A schematic of the forward Euler residual neural network. The state variable $v_2(t)$, the parameter σ , and the time step h are inputs to the neural network that estimates the right-hand side f_θ of the ODE. The right-hand side is then used to estimate the state variable $\hat{v}_2(t+h)$ by using a forward Euler step. (b) The predicted value for $\hat{v}_2(t+h)$ estimated from the Euler neural network is plotted against the true value (projected ABM trajectories in v_2 space) for values of the parameter $\sigma = \{0.57, 0.85, 1.11, 1.75, 1.9, 2.06, 2.25\}$ not included in the training set. (c) For three values of the parameter not included in the training set, we contrast generated paths from the ODE (solid lines) with paths simulated by the ABM (dashed lines) embedded in v_2 . The colors black, blue (gray), and green (light-gray) correspond to $\sigma = 1.11, 1.75$, and 2.25 , respectively.

model identifies the existence of three steady states—two stable and one unstable—for $\sigma < 1.84$. For $\sigma \geq 1.84$ a unique stable steady state exists. However, the computed bifurcation diagram clearly possesses the flip symmetry of the generic pitchfork bifurcation. We have identified a *perturbed* pitchfork, in which the upper branch remains permanently stable and the lower branch exhibits a turning point where a stable and an unstable component collide. The consistency of identifying a perturbed pitchfork was maintained across the different models we trained.

To make sure we recover a generically symmetric pitchfork bifurcation diagram, a simple transformation suffices: given the identified right-hand side $f_\theta(v_2(t); \sigma)$ of the neural network, we compute

$$g(v_2; \sigma) = \frac{f_\theta(v_2(t); \sigma) - f_\theta(-v_2(t); \sigma)}{2}. \quad (2)$$

The vector field $g(v_2; \sigma)$ is then used to construct the bifurcation diagram, shown in Fig. 5(b) correctly capturing the symmetry. Note that the computation of Eq. (2) does not require a second neural network but only evaluations of the trained Euler neural network discussed in the previous section. The average critical parameter of the identified ODE

(based on the 5000 trained neural networks) was estimated at $\sigma^* = 1.837$ and the standard deviation was 0.026; the true critical transition has been computed to be at $\sigma^* = 1.890$ [9]; see also Ref. [14], Eq. 3.52, for the case of additive noise. This suggests that the identified ODE provides a qualitatively correct and arguably quantitatively accurate approximation of the phase transition. This small discrepancy can likely be attributed to the limited range of σ values used in our training data. With only 18 distinct values for σ (details on sampling in Appendix 1), the closest ones to the true bifurcation point are $\sigma = 1.80$ and 1.90 . Using a finer grid of parameter values, we believe it would have led to even smaller discrepancy.

IV. SUMMARY AND CONCLUSIONS

We presented a data-driven framework for identifying qualitatively and approximating quantitatively phase transitions of interacting agent systems through an interplay between direct simulations and machine-learning-assisted coarse-grained system identification, and bifurcation analysis. We demonstrated that our framework is capable of identifying the phase transition of the DZ system with a single interpretable data-driven variable, in contrast to the derived closed-form model proposed in [9] where a system of four approximate ODEs was required for the task. To discover the effective coarse (collective) variables that parametrize collected data from the ABM, the Diffusion Maps manifold learning algorithm [16] was used. We showed that, for a constant parameter value ($\sigma = 1$), away from the phase transition, the Diffusion Maps algorithm discovers a one-dimensional manifold parametrized by the data-driven coordinate ψ_1 , and that ψ_1 is one-to-one with the theoretical order parameter M_1 . We then illustrated that, while data collected over a range of parameter values (bracketing the critical value) lie on a two-dimensional manifold, this manifold can be “disentangled” (factored) into a one-dimensional state-variability manifold “crossing” a one-dimensional parameter variability manifold. To disentangle the variability due to the parameter σ from the state variability, in terms of the latent Diffusion Maps coordinates (ψ_1, ψ_2) , we introduced a Y-shaped conformal autoencoder, initially proposed in [17] for addressing parameter nonidentifiability. We illustrated that the autoencoder’s

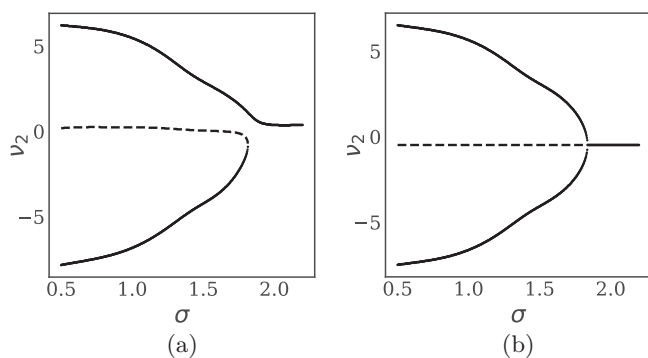


FIG. 5. (a) A representative bifurcation diagram constructed by the identified right-hand side of the Euler neural network suggests a “perturbed” pitchfork. (b) A representative bifurcation diagram after applying Eq. (2) to the identified right-hand side shows a symmetric pitchfork.

latent variables (ν_1, ν_2) form a conformal set of coordinates that ultimately allow us to learn a single effective nonlinear ODE in terms of a single state variable ν_2 and depending on a single parameter σ (the latter being one-to-one with ν_1). This effective nonlinear ODE was identified via a residual neural network, templated on the forward Euler numerical integrator. We compared generated paths of the identified ODE for test values of the parameter σ to paths generated with the full ABM embedded in the latent coordinate ν_2 , and we demonstrated good agreement between the two. To construct the bifurcation diagram from the identified right-hand side of the ODE, we used an odd symmetry transformation of the right-hand side. This transformation provides a symmetric vector field and captures the pitchfork bifurcation that denotes the phase transition. Imposing known physical symmetries in the data-driven models in order to enhance accuracy and improve generalizability is currently an active area of research [25–33].

Our proposed framework can be extended to a broad class of complex, multiscale dynamical systems for which a fine scale atomistic/stochastic mode exists, but for which accurate closed-form macroscopic equations at the coarse-grained level are not explicitly known. We are particularly interested in models from the social sciences, where no *physics-informed* natural choice for the order parameter(s) exists. As an example, we mention the noisy Hegselmann-Krause model for opinion dynamics [34], for which it has been rigorously proved that a discontinuous disorder (no consensus)/order (consensus) phase transition exists (Ref. [6], Prop. 6.2). It would be interesting to identify the phenomenological order parameter introduced in [34] using our approach. Other examples for which our approach is expected to be applicable are the Keller-Segel model for chemotaxis (see the work in [35–38]), the Fitzhugh-Nagumo model from the lattice Boltzmann method in [39,40], the mean-field Fitzhugh-Nagumo model for neurons (Ref. [41], Sec. 5.6), and the Dean-Kawasaki stochastic PDE [3,42], which takes into account finite-particle effects and fluctuations.

The code used to generate the results for this paper are available as a public repository [43].

ACKNOWLEDGMENTS

I.G.K. acknowledges partial support from the U.S. AFOSR FA9550-21-0317 and the U.S. Department of Energy SA22-0052-S001. G.A.P. is partially supported by the Frontier Research Advanced Investigator Grant ERC Grant No. Machine-aided general framework for fluctuating dynamic density functional theory. The authors are grateful to N. Zagli and A. Zanoni for useful discussions for making available the code from [9] and [41], respectively. Many thanks to T. Gaskin for helping set up the collaboration with Imperial College London. He acknowledges the hospitality of the Johns Hopkins group.

APPENDIX

1. Data collection

Given the ABM model described in Sec. II in the main text, we generated trajectories at 18 equidistant values of $\sigma \in$

[0.5, 2.2] [9]. For each value of σ we sampled 100 trajectories for different initial conditions (agents' distribution) drawn from the Pearson distribution (implemented in MATLAB) with prescribed mean, standard deviation, skewness, and kurtosis. The values for the mean, standard deviation, skewness, and kurtosis were chosen randomly from a prescribed grid of equidistant points: The mean was chosen from the range $[-2.0, 2.0]$, with increments of 0.2. The standard deviation was selected from the range $[0.0, 2.0]$, with increments of 0.1. The skewness was chosen from the range $[-2.0, 2.0]$, with increments of 0.2, and the kurtosis was selected from the range $[0.0, 15]$, with increments of 0.72. This scheme ensured a dense sampling both in parameter and state space.

The integration time of the ABM, containing $N = 12\,000$ agents, was set to $t_f = 10$ with a time step $dt = 0.005$. Data were collected every five snapshots, which led to a total of 400 snapshots per trajectory. Therefore, for a single value of the parameter, the total number of initial data is $n = 40\,000$ (400×100) while for multiple values of the parameter $n = 720\,000$ ($400 \times 100 \times 18$). Note that a number of trajectories explodes and thus they are omitted from any further computations.

An additional preprocessing step was applied to the data before the Diffusion Maps computation that includes discarding a short transient t_{cut} for each trajectory. This ensures that the fast transients have decayed and the collected data contain only the long-term dynamics (that live on the slow manifold). When dealing with multiple parameter values we chose $t_{\text{cut}} = 1$, while for a single value of the parameter $\sigma = 1$ we chose $t_{\text{cut}} = 0.5$ to make sure that enough transients are close to the unstable steady state, otherwise the manifold would appear as two clusters. As a test set, we sampled nine trajectories for each of the seven values of $\sigma = \{0.57, 0.85, 1.11, 1.75, 1.9, 2.06, 2.25\}$ not included in the training set. The selection of these test values was made to validate the predictions of our Euler neural network for dynamics before and after the bifurcation.

2. Diffusion Maps

The Diffusion Maps algorithm, introduced by Coifman and Lafon [16], can be used to discover a low-dimensional parametrization of high-dimensional data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ with each $\mathbf{x}_i \in \mathbb{R}^m$. Diffusion maps constructs a weighted graph $\mathbf{K} \in \mathbb{R}^{n \times n}$ between the sampled data points by using a kernel function. A common choice, also used in our case, is the Gaussian kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\varepsilon}\right), \quad (\text{A1})$$

where ε is a positive hyperparameter that controls the rate of the kernel's decay. The metric $\|\cdot\|$ in our case was chosen as the ℓ^2 norm, but different metrics are also possible.

To discover a low-dimensional manifold, regardless of the sampling density, the following normalization is required:

$$\tilde{\mathbf{K}} = \mathbf{P}^{-1} \mathbf{K} \mathbf{P}^{-1}, \quad (\text{A2})$$

where $P_{ii} = \sum_{j=1}^n K_{jj}$. A second normalization of $\tilde{\mathbf{K}}$ recovers a row-stochastic, Markovian matrix

$$\mathbf{M} = \mathbf{D}^{-1} \tilde{\mathbf{K}}, \quad (\text{A3})$$

where \mathbf{D} is a diagonal matrix defined as $D_{ii} = \sum_{j=1}^n \tilde{K}_{ij}$.

The entries of matrix \mathbf{M} can be seen as probabilities of jumping from one point to the other. The eigendecomposition of \mathbf{M} ,

$$\mathbf{M}\phi_i = \lambda_i\phi_i, \quad (\text{A4})$$

provides a set of eigenvectors ϕ_i and corresponding eigenvalues λ_i . To obtain a more parsimonious representation of the original data set \mathbf{X} , proper selection of the eigenvectors is needed. If the *intrinsic* dimension of the data is small, this selection can be achieved by visual inspection of the nonharmonic eigenvectors (eigenvectors that span independent directions) [44]. Alternatively, the local-linear regression algorithm proposed by Dsilva *et al.* [23] can be used for selecting the nonharmonic eigenvectors. If the number of independent nonharmonic eigenvectors is smaller than the dimension m of the data \mathbf{X} , then dimensionality reduction has been achieved. In our work, the Python library *datafold* [24] was used for the Diffusion Maps and the local-linear regression algorithms.

a. Nyström extension

The Nyström extension formula provides a numerical approximation of eigenfunctions of the form [45]

$$\int_a^b M(x_j, x_i)\phi_i(x_i)dx_i = \lambda\phi(x_j). \quad (\text{A5})$$

In our work, the Nyström extension is utilized when new *out-of-sample* data points are given, e.g., $\mathbf{x}_{\text{new}} \notin \mathbf{X}$. To generate the Diffusion Maps coordinates ϕ_{new} , the Nyström extension uses an interpolation scheme based on the kernel computations and normalizations applied during the dimensionality reduction step, discussed in the previous section. The Nyström extension formula reads

$$\phi_i(\mathbf{x}_{\text{new}}) = \frac{1}{\lambda_i} \sum_{j=1}^n \tilde{\mathbf{M}}(\mathbf{x}_{\text{new}}, \mathbf{x}_j)\phi_i(\mathbf{x}_j), \quad (\text{A6})$$

where $\phi_i(\mathbf{x}_{\text{new}})$ denotes the estimated i th eigenvector for the data point ϕ_{new} , λ_i denotes the corresponding eigenvalue, $\phi_i(\mathbf{x}_j)$ denotes the j th component of the i th eigenvector, and $\tilde{\mathbf{M}}(\cdot, \cdot)$ denotes the kernel function used to determine the similarity of \mathbf{x}_{new} to all the points in \mathbf{X} .

b. Diffusion Maps on ABM data

In this section, we provide details on how the Diffusion Maps algorithm was computed on histograms and moments from the ABM. In the first case, for each snapshot of the ABM we constructed a histogram as an approximation of the agents' density. Each histogram contains 40 equidistant bins defined in the range $[-4, 4]$. This range ensures that all agents in the collected (training) data lie in between. Note that the method is insensitive to the selected number of bins. To reduce the computational cost of the Diffusion Maps, we subsampled the training data uniformly [24], which resulted in about 3500 data points when Diffusion Maps applied for a single parameter and about 19 000 data points for multiple values of the parameter. The hyperparameter ε was selected as the square of the median of the pairwise distances multiplied by a constant

c , where $c = 0.03$ for a single value of the parameter σ and $c = 20$ for multiple values of the parameter σ . In the second case, Diffusion Maps was computed on the sampled moments, M_1, M_2, M_3, M_4 . Again, to alleviate the computational cost, we subsampled the data [24], which results in $N \sim 11\,000$. The hyperparameter ε was also selected for this case by computing the median of the pairwise distances multiplied with $c = 10$.

c. Diffusion Maps: Additional results

In this section, we provide additional results for the Diffusion Maps computation for $\sigma = 1$ on the histograms and the Diffusion Maps computations on the moments M_1, M_2, M_3, M_4 . We showed in the main text that the Diffusion Maps coordinate ψ_1 is one-to-one with the first moment M_1 . Here we show that M_3 is also one-to-one with ψ_1 [Fig. 6(a)] and that the even moments M_2 and M_4 can be seen as functions of ψ_1 [Figs. 6(b) and 6(c)].

To strengthen the argument that the manifold in this case is one-dimensional, we show in Fig. 6(d) the eigenvectors $\psi_2 - \psi_9$ plotted against the first nontrivial eigenvector ψ_1 . From Fig. 6(d) it appears that the eigenvectors $\psi_2 - \psi_9$ are harmonics of ψ_1 . This suggests that the manifold for a fixed value of σ is one-dimensional.

The Diffusion Maps algorithm applied using the computed first four moments reveals a two-dimensional manifold embedded in four dimensions. The first two eigenvectors in this case, ϕ_1 and ϕ_2 , shown in Figs. 7(a) and 7(b), are the nonharmonic eigenvectors that parametrize the data. This is corroborated by the larger residuals r_k of the first two eigenvectors shown in Fig. 7(c). Also in this case, the parameter σ and the first moment M_1 appear visually as functions of the Diffusion Maps coordinates [Figs. 7(a) and 7(b)]. This pair of coordinates, obtained by Diffusion Maps on moments, could have been used for the CAE's training instead of the Diffusion Maps coordinates (ψ_1, ψ_2) obtained from the computations on the histograms, but we omit this for brevity.

3. Y-shaped conformal autoencoder

The Y-shaped conformal autoencoder was initially presented in [17]. In this work, the Y-shaped conformal autoencoder consists of the three connected (sub)networks

$$\text{Encoder: } (\psi_1, \psi_2) \mapsto (v_1, v_2), \quad (\text{A7})$$

$$\text{Decoder: } (v_1, v_2) \mapsto (\hat{\psi}_1, \hat{\psi}_2), \quad (\text{A8})$$

$$\text{Estimator: } v_1 \mapsto \hat{\sigma}. \quad (\text{A9})$$

The Encoder receives as inputs the two Diffusion Maps coordinates ψ_1, ψ_2 and maps them to the latent variables v_1, v_2 . The Decoder aims to reconstruct the Diffusion Maps coordinates from the latent variables v_1, v_2 . The Estimator has as input the latent coordinate v_1 and aims to learn a map from v_1 to the parameter σ .

The loss function used to train the Y-shaped conformal autoencoder consists of three parts: (a) The loss function of the Encoder-Decoder (autoencoder) \mathcal{L}_{ae} that aims to reconstruct the input itself, (b) the loss function of the Estimator, \mathcal{L}_{est} , that aims to reproduce the parameter σ given v_1 , and (c) the

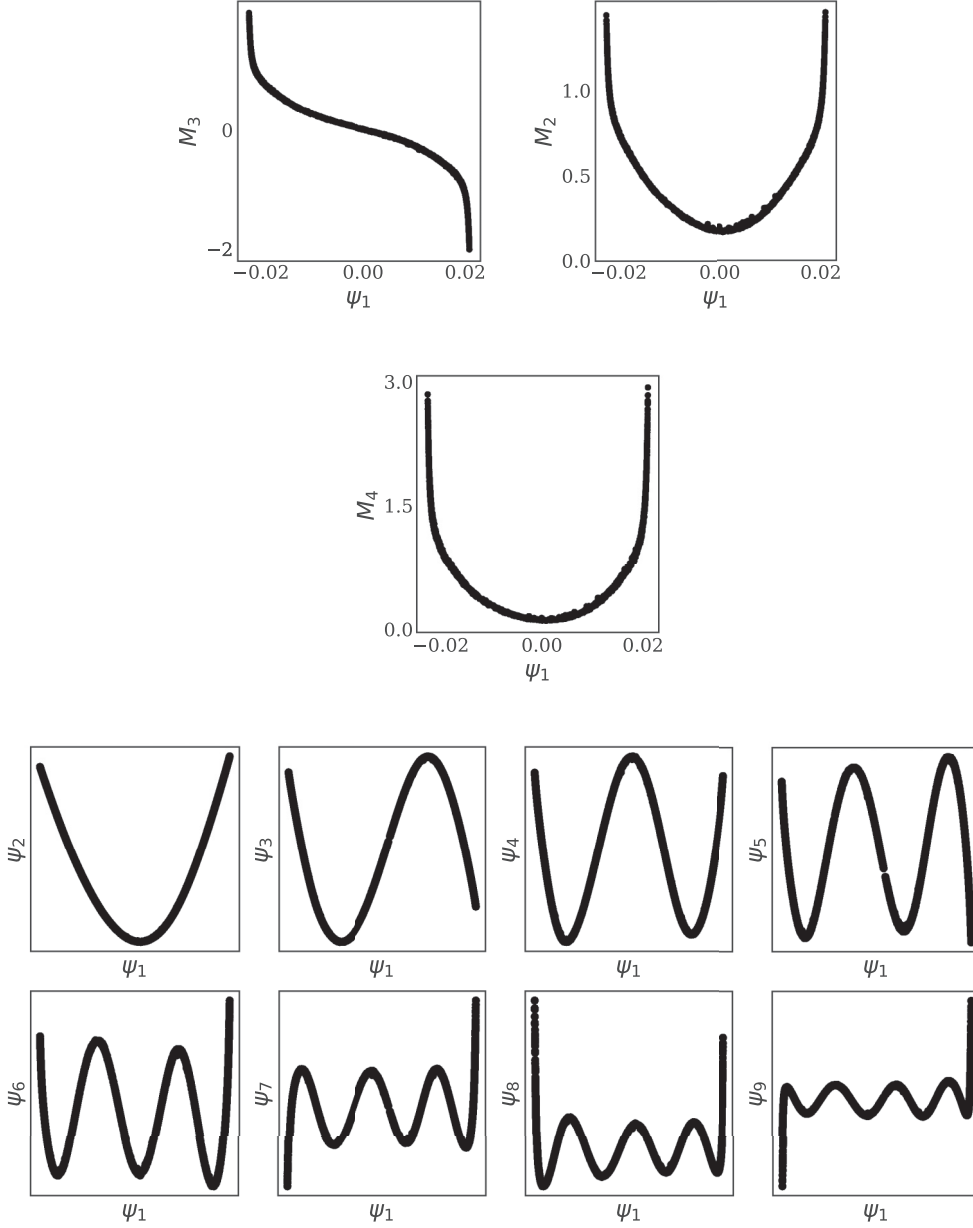


FIG. 6. [(a)–(c)] The Diffusion Maps coordinate ψ_1 is plotted against the computed moments M_2 , M_3 , and M_4 , respectively. (d) The eigenvector ψ_1 is plotted against the eigenvectors $\psi_2 - \psi_9$. This supports our argument that the eigenvectors $\psi_2 - \psi_9$ are harmonics of ψ_1 .

loss function for imposing the conformality constraint, \mathcal{L}_{con} , between v_1 and v_2 ,

$$\langle \nabla v_1, \nabla v_2 \rangle = 0, \quad (\text{A10})$$

where the gradient ∇ is in terms of the Diffusion Maps coordinates (ψ_1, ψ_2) of the input and $\langle \cdot, \cdot \rangle$ denotes the inner product between the two vectors. The gradients were computed by using the automatic differentiation of Pytorch [46]. In practice, instead of using Eq. (A10), one can minimize the angle between the vectors,

$$\cos\theta = \frac{\nabla v_1 \cdot \nabla v_2}{\|\nabla v_1\| \|\nabla v_2\|}, \quad (\text{A11})$$

which stabilizes the training of the network. We describe details for training the network, specifics about the architecture used, and the choice of hyperparameters in the next section.

a. Hyperparameter selection and training procedure

The implementation of the Y-shaped conformal autoencoder was done with the Pytorch Python library [46].

Each (sub)network (Encoder, Decoder, Estimator) in the architecture of the Y-shaped conformal autoencoder consists of five fully connected layers. The first four hidden layers have 20 neurons and $\tanh(t)$ activation functions, and the fifth has no activation function (linear activation) and its size depends on the size of the desired output. The ADAM optimizer was chosen for training the overall network. We chose minibatches

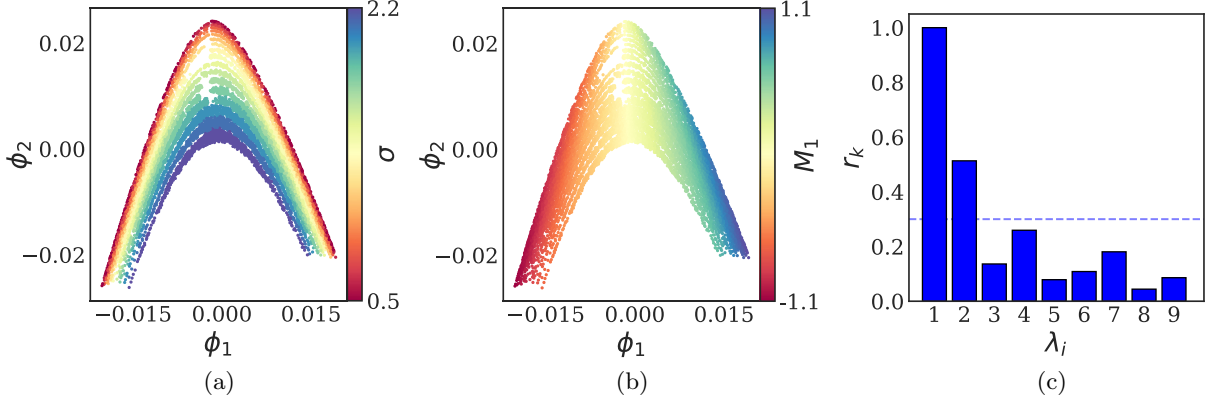


FIG. 7. (a),(b) The nonharmonic coordinates ϕ_1 and ϕ_2 colored with σ and M_1 , respectively. (c) The residual r_k indicates ϕ_1 and ϕ_2 are the two nonharmonic coordinates.

of size 32 to train the network. The learning rate was selected as $\eta = 0.001$ and 500 epochs. Training the network for a larger number of epochs leads to overfitting.

To train the network and test its generalization capability, we used about 19000 data points. We split the data into train|test|validation as 80:10:10. We then rescaled the training data by using the *MinMaxScaler* Python preprocessing scheme from *sklearn*. We applied the same transformation for the validation and test sets. During the training of the network, we used only the training set to perform backpropagation and the validation set to get insight into the model's performance. The network did not *see* the test set during training.

The optimization process we performed was *heuristic*: for a fixed minibatch, two updates (backpropagation steps) were performed. The first step updates the weights of the Encoder-Decoder and the second step the weights of the Estimator-Encoder. Altering this training protocol is possible. In Algorithm 1 below we provide a more detailed description of the network's training.

Upon training of the neural network, the estimated MSE for the autoencoder's reconstruction was $\mathcal{L}_{\text{ae}} = 1.36 \times 10^{-8}$ on the train set and $\mathcal{L}_{\text{ae}} = 1.37 \times 10^{-8}$ on the test set. The MSE for the Estimator was $\mathcal{L}_{\text{est}} = 1.2 \times 10^{-3}$ for the train set and $\mathcal{L}_{\text{est}} = 1.2 \times 10^{-3}$ for the test. The average value of the $\cos\theta$ on the train set was $\mathcal{L}_{\text{con}} = 2.72 \times 10^{-5}$ and on the test set was $\mathcal{L}_{\text{con}} = 2.63 \times 10^{-5}$.

b. Y-shaped conformal autoencoder: Additional results

In this section, we provide additional results regarding the Y-shaped conformal autoencoder described in the main text. The ability of the Estimator to predict the parameter σ from v_1 is shown in Fig. 8(a) for train (black dots) and test (red dots) points. We also illustrate, in Figs. 8(b) and 8(c), the reconstruction of the autoencoder for train and test points.

4. Forward Euler neural network

In this section, we describe how we identified the right-hand side of an ordinary differential equation directly from data. Let $v_2(t)$ be a state variable whose dynamics are governed by a σ -dependent ODE given by the general form

$$\dot{v}_2(t) = f(v_2(t); \sigma). \quad (\text{A12})$$

Our goal is to construct a neural network architecture inspired by numerical integrators of ODEs to estimate the right-hand side $f(v_2(t); \sigma)$. To this end, we constructed a forward Euler residual neural network depicted in Fig. 4(a). To train this network, we do not require long trajectories but only snapshots of the form $\mathcal{D} = \{v_2(t+h), v_2(t), \sigma, h\}$, where $v_2(t)$ is the state variable at time t , and $v_2(t+h)$ is the state variable after a small time step h . Given sampled data in the form of \mathcal{D} , we wish to approximate f by using a neural network, with weights denoted as θ .

To formulate the loss used to train the network f_θ , we remind the reader that the forward Euler approximates the

ALGORITHM 1. The algorithm illustrates a full iteration during training of the Y-shaped conformal autoencoder. We set the scale parameter to $\alpha = 10$. The learning rate is denoted as η .

Input: Diffusion Maps coordinates ψ_1, ψ_2 and parameter values σ .

Output: The weights of (i) Encoder (θ_{encoder}), (ii) Decoder (θ_{decoder}), (iii) Estimator ($\theta_{\text{estimator}}$).

For $i = 1, 2, \dots, T$

1. Predict:

$$(v_1, v_2) = \text{Encoder}(\psi_1, \psi_2)$$

$$(\hat{\psi}_1, \hat{\psi}_2) = \text{Decoder}(v_1, v_2)$$

2. Compute Autoencoder (Encoder-Decoder) and Conformality

Losses:

$$\mathcal{L}_1 = \mathcal{L}_{\text{ae}} + \mathcal{L}_{\text{con}}$$

$$= \text{MSE}(\hat{\psi}, \psi) + \alpha \text{MSE}(\cos\theta, 0)$$

3. Backpropagation step—update weights (illustration with gradient descent):

$$\theta_{\text{encoder}}^- = \eta \theta_{\text{encoder}} \mathcal{L}_1$$

$$\theta_{\text{decoder}}^- = \eta \theta_{\text{decoder}} \mathcal{L}_1$$

4. Predict:

$$(v_1, v_2) = \text{Encoder}(\psi_1, \psi_2)$$

$$\hat{\sigma} = \text{Estimator}(v_1)$$

5. Compute Estimator Loss:

$$\mathcal{L}_{\text{est}} = \text{MSE}(\hat{\sigma}, \sigma)$$

6. Backpropagation step—update weights (illustration with gradient descent)

$$\theta_{\text{estimator}}^- = \eta \theta_{\text{estimator}} \mathcal{L}_{\text{est}}$$

$$\theta_{\text{encoder}}^- = \eta \theta_{\text{encoder}} \mathcal{L}_{\text{est}}$$

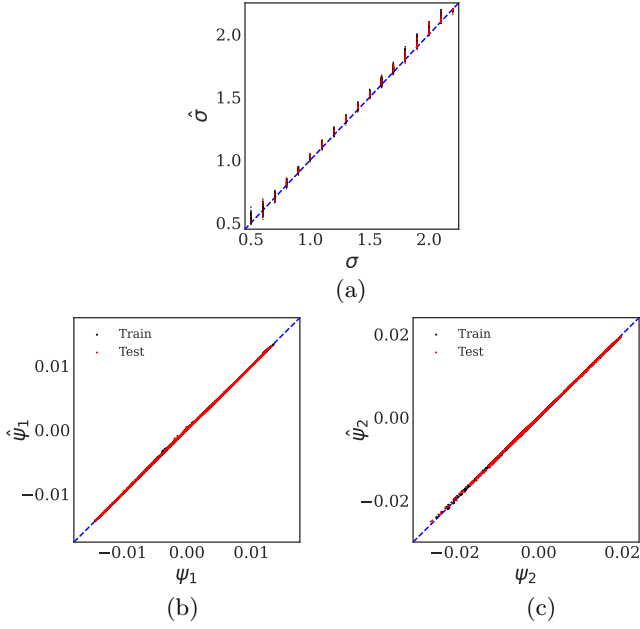


FIG. 8. (a) The true values of the parameter σ are plotted against that reconstructed by the Estimator for train (black) and test (red) points. The blue dashed line indicates $y = x$. (b),(c) The true values of the Diffusion Maps coordinates ψ_1 and ψ_2 are plotted against the reconstructed $\hat{\psi}_1$, $\hat{\psi}_2$ coordinates by the autoencoder.

evolution of an ODE by a small positive step h , as

$$v_2(t+h) = v_2(t) + hf(v_2; \sigma). \quad (\text{A13})$$

In our case, without having access to f but only data in the form \mathcal{D} we wish to approximate the right-hand side with the neural network f_θ . This is achieved by performing for each pair of inputs $(v_2(t), \sigma)$ one integration step of size h , estimating the evolved dynamics $\hat{v}_2(t+h)$ and minimizing the loss

$$\mathcal{L}(\theta|v_2(t), v_2(t+h), h, \sigma) = \|\hat{v}_2(t+h) - v_2(t+h)\|^2. \quad (\text{A14})$$

For our computations, the time step h was kept constant but the overall approach can be easily extended to handle also varying time steps h .

a. Hyperparameter selection and training procedure

The implementation of the forward Euler neural networks was done with the Tensorflow/Keras Python libraries [47].

To train the forward Euler neural network, we need to ensure our data are in the form of snapshots \mathcal{D} . To achieve that, we used the Nyström extension formula (see Appendix 2a) to all the available sampled trajectories and obtained the corresponding trajectories in $\psi_1 \psi_2$. We then evaluated the Encoder to get trajectories in terms of v_2 . These two steps provided us with about 600 000 snapshots. We then split the data into train|test|validation as 80: 10 :10. We then centered and whitened the data (based on the mean and variance of the training set) and applied the same transformation to the validation and test set.

The architecture consisted of two hidden layers with 10 neurons each. The first hidden layer had a $\tanh(t)$ activation

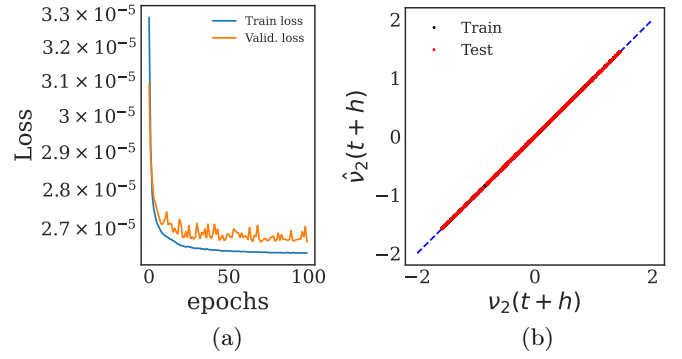


FIG. 9. (a) The learning curves for the train and validation sets. (b) The true values of the state variable $v_2(t+h)$ plotted against that predicted by the Euler neural network $\hat{v}_2(t+h)$ for the train (black points) and test (red points).

function and the second hidden layer had a linear activation function. We used ADAM to optimize this network. The mini-batch size was set to 32, the number of total epochs to 100, and the learning rate to $\eta = 0.001$. The learning curves for the training and validation are shown in Fig. 9(a). The ability of the network to fit the training set and generalize is shown in Fig. 9(b). Upon training of the network, the MSE on the train and test sets were 2.63×10^{-3} and 2.79×10^{-5} , respectively. This neural network was used for the computations reported in Sec. III C. We did not record the MSE for all the 5000 networks used to check the robustness of our approach in identifying the critical transition.

b. Euler neural network: Additional results

In this section, we provide some additional results for the forward Euler neural network. The learning curves for the training and validation are shown in Fig. 9(a). The ability of the network to fit the train set and generalize is shown in Fig. 9(b). Note that the normalized value of the state variable v_2 is shown in Fig. 4(a) and that the test points in this include values of the parameter σ that are also in the training set.

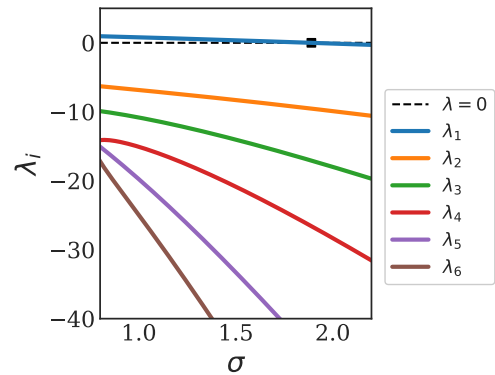


FIG. 10. The eigenvalues of the Jacobian computed across a range of parameter values σ for the moments' equations with six ODEs presented in [9]. The slowest eigenvalue (upper line) λ_1 crosses the real axis at $\sigma = 1.89$ where the bifurcation occurs. The black square indicates the value of σ where the (slowest) eigenvalue crosses zero.

5. Separation of timescales

In this section, we discuss the separation of timescales for the reduced order models based on the equations for the moments proposed in [9]. As shown in Fig. 10, the slowest

eigenvalue λ_1 is at least one order of magnitude smaller than the second slowest λ_2 . This suggests that the dynamics after a short transient are slaved in the direction of this slowest eigenvalue and are effectively one-dimensional.

-
- [1] *Boston Mathematical Modeling of Collective Behavior in Socio-Economic and Life Sciences*, Modeling and Simulation in Science, Engineering and Technology, edited by G. Naldi, L. Pareschi, and G. Toscani (Birkhäuser Boston Ltd., Boston, MA, 2010).
- [2] L. Pareschi and G. Toscani, *Interacting Multiagent Systems: Kinetic Equations and Monte Carlo Methods* (Oxford University Press, Oxford, 2013).
- [3] L. Helfmann, J. Heitzig, P. Koltai, J. Kurths, and C. Schütte, Statistical analysis of tipping pathways in agent-based models, *Eur. Phys. J.: Spec. Top.* **230**, 3249 (2021).
- [4] L. Helfmann, N. D. Conrad, A. Djurdjevac, S. Winkelmann, and C. Schütte, From interacting agents to density-based modeling with stochastic PDEs, *Commun. Appl. Math. Comput. Sci.* **16**, 1 (2021).
- [5] L. Chayes and V. Panferov, The McKean-Vlasov equation in finite volume, *J. Stat. Phys.* **138**, 351 (2010).
- [6] J. A. Carrillo, R. S. Gvalani, G. A. Pavliotis, and A. Schlichting, Long-time behaviour and phase transitions for the McKean-Vlasov equation on the torus, *Arch. Ration. Mech. Anal.* **235**, 635 (2020).
- [7] T. Gross, C. J. D. D’Lima, and B. Blasius, Epidemic dynamics on an adaptive network, *Phys. Rev. Lett.* **96**, 208701 (2006).
- [8] S. Winkelmann, J. Zonker, C. Schütte, and N. D. Conrad, Mathematical modeling of spatio-temporal population dynamics and application to epidemic spreading, *Math. Biosci.* **336**, 108619 (2021).
- [9] N. Zagli, G. A. Pavliotis, V. Lucarini, and A. Alecio, Dimension reduction of noisy interacting systems, *Phys. Rev. Res.* **5**, 013078 (2023).
- [10] P. Liu, H. R. Safford, I. D. Couzin, and I. G. Kevrekidis, Coarse-grained variables for particle-based models: diffusion maps and animal swarming simulations, *Computat. Part. Mech.* **1**, 425 (2014).
- [11] P. Liu, C. Siettos, C. W. Gear, and I. Kevrekidis, Equation-free model reduction in agent-based computations: Coarse-grained bifurcation and variable-free rare event analysis, *Math. Modell. Natural Phenom.* **10**, 71 (2015).
- [12] G. Fabiani, N. Evangelou, T. Cui, J. M. Bello-Rivas, C. P. Martin-Linares, C. Siettos, and I. G. Kevrekidis, Task-oriented machine learning surrogates for tipping points of agent-based models, *Nat. Commun.* **15**, 4117 (2024).
- [13] N. Martzel and C. Aslangul, Mean-field treatment of the many-body Fokker-Planck equation, *J. Phys. A* **34**, 11225 (2001).
- [14] D. A. Dawson, Critical dynamics and fluctuations for a mean-field model of cooperative behavior, *J. Stat. Phys.* **31**, 29 (1983).
- [15] J. A. Acebrón, L. L. Bonilla, C. J. Pérez V., F. Ritort, and R. Spigler, The kuramoto model: A simple paradigm for synchronization phenomena, *Rev. Mod. Phys.* **77**, 137 (2005).
- [16] R. R. Coifman and S. Lafon, Diffusion maps, *Appl. Computat. Harmon. Anal.* **21**, 5 (2006).
- [17] N. Evangelou, N. J. Wichrowski, G. A. Kevrekidis, F. Dietrich, M. Kooshkbaghi, S. McFann, and I. G. Kevrekidis, On the parameter combinations that matter and on those that do not: Data-driven studies of parameter (non) identifiability, *Proc. Natl. Acad. Sci. (USA) nexus* **1**, pgac154 (2022).
- [18] R. Rico-Martínez, K. Krischer, I. Kevrekidis, M. Kube, and J. Hudson, Discrete-vs. continuous-time nonlinear signal processing of cu electrodisolution data, *Chem. Eng. Commun.* **118**, 25 (1992).
- [19] R. González-García, R. Rico-Martínez, and I. G. Kevrekidis, Identification of distributed parameter systems: A neural net based approach, *Comput. Chem. Eng.* **22**, S965 (1998).
- [20] F. Dietrich, A. Makeev, G. Kevrekidis, N. Evangelou, T. Bertalan, S. Reich, and I. G. Kevrekidis, Learning effective stochastic differential equations from microscopic simulations: Combining stochastic numerics and deep learning, [arXiv:2106.09004](https://arxiv.org/abs/2106.09004).
- [21] N. Evangelou, F. Dietrich, J. M. Bello-Rivas, A. J. Yeh, R. S. Hendley, M. A. Bevan, and I. G. Kevrekidis, Learning effective sdes from brownian dynamic simulations of colloidal particles, *Mol. Syst. Des. Eng.* **8**, 887 (2023).
- [22] C. Van den Broeck, J. M. R. Parrondo, J. Armero, and A. Hernández-Machado, Mean field model for spatially extended systems in the presence of multiplicative noise, *Phys. Rev. E* **49**, 2639 (1994).
- [23] C. J. Dsilva, R. Talmon, R. R. Coifman, and I. G. Kevrekidis, Parsimonious representation of nonlinear dynamical systems through manifold learning: A chemotaxis case study, *Appl. Computat. Harmon. Anal.* **44**, 759 (2018).
- [24] D. Lehmborg, F. Dietrich, G. Köster, and H.-J. Bungartz, Datafold: Data-driven models for point clouds and time series on manifolds, *J. Open Source Softw.* **5**, 2283 (2020).
- [25] P. J. Olver, Modern developments in the theory and applications of moving frames, *London Math. Soc. Impact150 Stories* **1**, 9 (2015).
- [26] M. Mattheakis, P. Protopapas, D. Sondak, M. Di Giovanni, and E. Kaxiras, Physical symmetries embedded in neural networks, [arXiv:1904.08991](https://arxiv.org/abs/1904.08991).
- [27] D. Yarotsky, Universal approximations of invariant maps by neural networks, *Construct. Approx.* **55**, 407 (2022).
- [28] B. Blum-Smith and S. Villar, Equivariant maps from invariant functions, [arXiv:2209.14991](https://arxiv.org/abs/2209.14991).
- [29] S. Villar, W. Yao, D. W. Hogg, B. Blum-Smith, and B. Dumitrascu, Dimensionless machine learning: Imposing exact units equivariance, *J. Machine Learning Res.* **24**, 32 (2023).
- [30] P. J. Olver, M. Sabzevari, and F. Valiquette, Normal forms, moving frames, and differential invariants for nondegenerate hypersurfaces in \mathbb{C}_2 , *J. Geom. Anal.* **33**, 192 (2023).
- [31] P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis, Sympnets: Intrinsic structure-preserving symplectic networks

- for identifying Hamiltonian systems, *Neural Netw.* **132**, 166 (2020).
- [32] F. Alet, D. Doblar, A. Zhou, J. Tenenbaum, K. Kawaguchi, and C. Finn, Noether networks: Meta-learning useful conserved quantities, *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, Vol. 34 (Curran Associates Inc., Red Hook, 2021), pp. 16384–16397.
- [33] J. W. Burby, Q. Tang, and R. Maulik, Computing Poincaré maps using physics-informed deep learning, Tech. Rep. [Los Alamos National Lab. (LANL), Los Alamos, NM, 2020].
- [34] C. Wang, Q. Li, W. E, and B. Chazelle, Noisy Hegselmann-Krause systems: phase transition and the $2R$ -conjecture, *J. Stat. Phys.* **166**, 1209 (2017).
- [35] S. Lee, Y. M. Psarellis, C. I. Siettos, and I. G. Kevrekidis, Learning black-and gray-box chemotactic pdes/closures from agent based monte carlo simulation data, *J. Math. Biol.* **87**, 15 (2023).
- [36] Y. M. Psarellis, S. Lee, T. Bhattacharjee, S. S. Datta, J. M. Bello-Rivas, and I. G. Kevrekidis, Data-driven discovery of chemotactic migration of bacteria via machine learning, [arXiv:2208.11853](https://arxiv.org/abs/2208.11853).
- [37] C. Siettos, Coarse-grained computational stability analysis and acceleration of the collective dynamics of a Monte Carlo simulation of bacterial locomotion, *Appl. Math. Comput.* **232**, 836 (2014).
- [38] S. Setayeshgar, C. W. Gear, H. G. Othmer, and I. G. Kevrekidis, Application of coarse integration to bacterial chemotaxis, *Multiscale Model. Simul.* **4**, 307 (2005).
- [39] A. Armaou, I. G. Kevrekidis, and C. Theodoropoulos, Equation-free gaptooth-based controller design for distributed complex/multiscale processes, *Comput. Chem. Eng.* **29**, 731 (2005).
- [40] E. Galaris, G. Fabiani, I. Gallos, I. Kevrekidis, and C. Siettos, Numerical bifurcation analysis of pdes from lattice Boltzmann model simulations: A parsimonious machine learning approach, *J. Sci. Comput.* **92**, 34 (2022).
- [41] G. A. Pavliotis and A. Zaroni, A method of moments estimator for interacting particle systems and their mean field limit, [arXiv:2212.00403](https://arxiv.org/abs/2212.00403) [math.NA].
- [42] F. Cornalba and J. Fischer, The dean-kawasaki equation and the structure of density fluctuations in systems of diffusing particles, [arXiv:2109.06500](https://arxiv.org/abs/2109.06500) [math.AP].
- [43] https://gitlab.com/nicolasevangelou/ml_phase_transition.
- [44] N. Evangelou, F. Dietrich, E. Chiavazzo, D. Lehmborg, M. Meila, and I. G. Kevrekidis, Double diffusion maps and their latent harmonics for scientific computations in latent space, *J. Comput. Phys.* **485**, 112072 (2023).
- [45] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, Spectral grouping using the nystrom method, *IEEE Trans. Pattern Anal. Machine Intell.* **26**, 214 (2004).
- [46] 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 *et al.*, Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems*, Vol. 32 (Curran Associates, 2019), pp. 8024–8035.
- [47] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from [tensorflow.org](https://www.tensorflow.org).