nature computational science

Article

https://doi.org/10.1038/s43588-025-00893-8

Discovering network dynamics with neural symbolic regression

Received: 31 January 2025

Accepted: 19 September 2025

Published online: 23 October 2025

Check for updates

Zihan Yu [1,2], Jingtao Ding [1,2] & Yong Li [1]

Network dynamics are fundamental to analyzing the properties of high-dimensional complex systems and understanding their behavior. Despite the accumulation of observational data across many domains, mathematical models exist in only a few areas with clear underlying principles. Here we show that a neural symbolic regression approach can bridge this gap by automatically deriving formulas from data. Our method reduces searches on high-dimensional networks to equivalent one-dimensional systems and uses pretrained neural networks to guide accurate formula discovery. Applied to ten benchmark systems, it recovers the correct forms and parameters of underlying dynamics. In two empirical natural systems, it corrects existing models of gene regulation and microbial communities, reducing prediction error by 59.98% and 55.94%, respectively. In epidemic transmission across human mobility networks of various scales, it discovers dynamics that exhibit the same power-law distribution of node correlations across scales and reveal country-level differences in intervention effects. These results demonstrate that machine-driven discovery of network dynamics can enhance understandings of complex systems and advance the development of complexity science.

In complex systems, local interactions among many components give rise to emergent global behaviors that cannot be explained by individual components alone[1]. Understanding these phenomena requires network dynamics that describe the state changes of network nodes through their interactions using mathematical formulas[1–5] Network dynamics provide the theoretical foundation for analyzing complex system properties such as controllability[2], resilience[3], stability[4] and perturbation responses[5] across diverse systems, from neural[6] and ecological[7] to social[8] networks. Traditionally, the development of network dynamics has relied on expert intuition and experience, limiting mathematical formulations to a few well-studied domains such as biology and ecology, while most systems lacking first-principles knowledge remained without suitable models[9]. Fortunately, the development of deep learning and neuro-symbolic artificial intelligence has motivated a data-driven paradigm aimed at discovering how components regulate themselves and interact with others, based on observational data of network structures and node activities.

The process of finding the formula that best fits the given data is known as symbolic regression[10–14]. It derives functional relationships by searching combinations of mathematical operators (for example, +, × and sin) that best fit data and has been widely used to discover governing functions for nonnetworked single-body or few-body systems[13–16]. In complex networked systems, however, the search space grows super-exponentially with network dimensionality n. As n increases, the number of variables and the formula length scale as nd and nl, where d is the number of features per node and l is the interaction law length. This yields a search space of all possible variable permutations $O((nd)^{nl})$, which far exceeds existing methods typically designed for nonnetworked systems with n equals 1 or a very small constant[17–23]. In addition to symbolic regression, sparse identification of nonlinear dynamics[24] (SINDy) has been applied to the identification of network dynamics[25]. Despite its computational efficiency, SINDy depends on predefined function libraries, which limit its ability to discover previously unseen dynamics in systems

[1]Department of Electronic Engineering, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China. [2]These authors contributed equally: Zihan Yu, Jingtao Ding. e-mail: dingjt15@tsinghua.org.cn; liyong07@tsinghua.edu.cn

https://doi.org/10.1038/s43588-025-00893-8

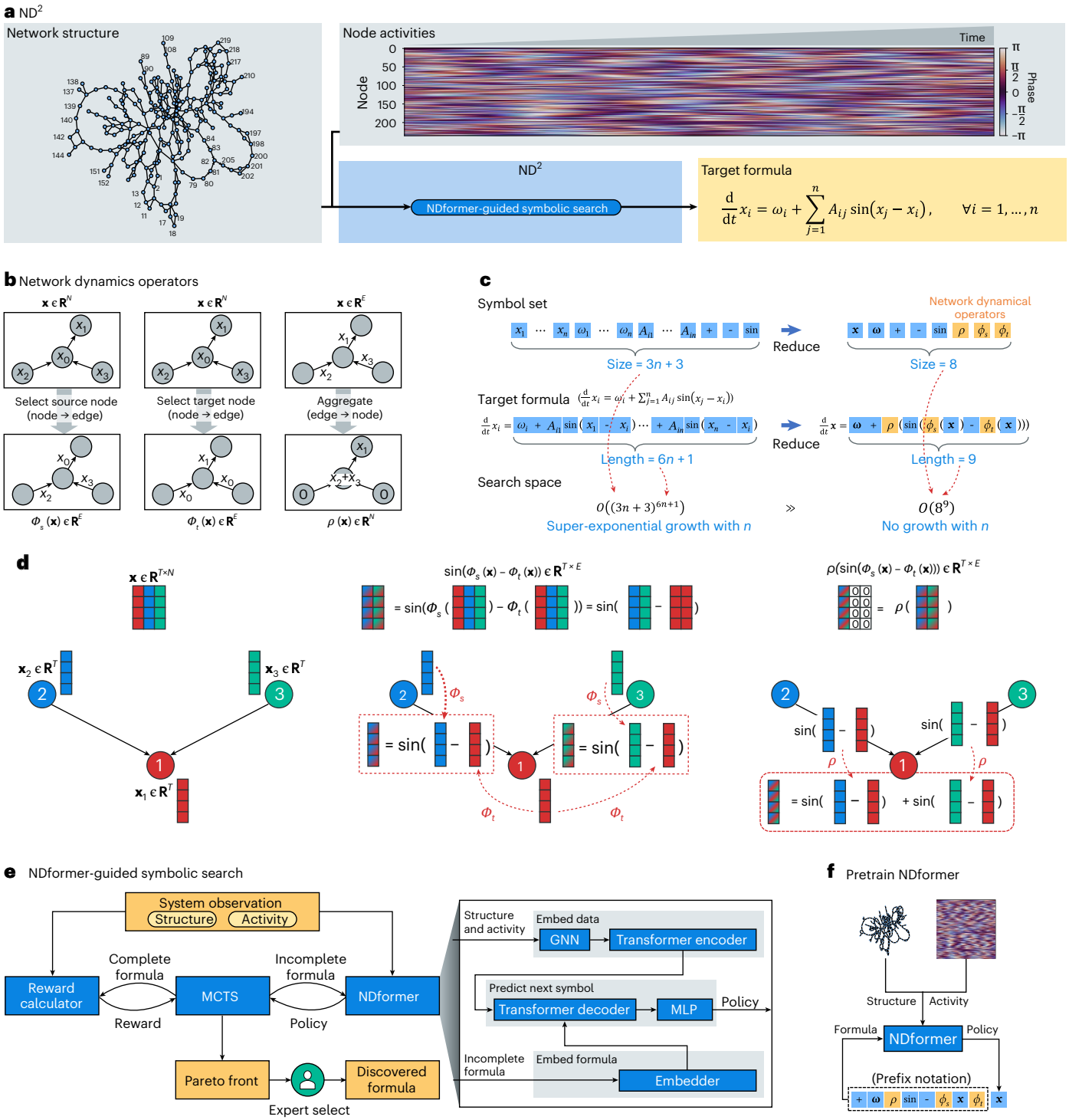

Fig. 1 | ND². **a**, Given the network structure and node activities, the ND² approach discovers the target formula. The example here is Kuramoto dynamics on an empirical network (the power grid network of Northern Europe) with 236 nodes and 320 undirected edges. **b**–**d**, Demonstration of the network dynamics operators, including definitions of the three proposed operators (**b**), an example of these operators reducing the search space of Kuramoto dynamics (**c**) and a three-node network example to illustrate the information path from nodes to edges and finally aggregate back to nodes (**d**). **e**,**f**, The NDformer-guided symbolic search algorithm and NDformer's architecture (**e**), as well as NDformer's pretraining process (**f**).

without prior knowledge. To the best of our knowledge, there is no method to implement symbolic regression in complex network systems so far.

In this article, we propose Neural Discovery of Network Dynamics (ND²), a deep learning method that automatically discovers network dynamics formulas through symbolic regression. We design a set of network dynamical operators that transform symbolic search on high-dimensional networks into an equivalent one-dimensional problem, alongside an NDformer-guided symbolic search algorithm for efficient formula discovery. These operators express network dynamics formulas independently of network size (Fig. 1b), reducing the original search space from the super-exponential $O((nd)^{nl})$ to $O(d^l)$ regardless of n (Fig. 1c). The search algorithm combines the advantages of neural network and symbolic search methods (Fig. 1e) by leveraging

NDformer, a neural network that incorporates graph neural networks (GNNs)[26] and transformers[27] to capture network dynamics. NDformer is pretrained to predict symbols in formulas given network structure and node activity data (Fig. 1f), and guides a Monte Carlo tree search (MCTS)[28] module to efficiently explore the search space and discover accurate, concise formulas.

We applied ND^2 to ten synthetic systems with ground-truth dynamics, two real-world nature systems with established dynamics, and a social system without existing dynamics. From synthetic datasets of well-known systems, ND^2 exactly recovered ground-truth formulas from node activities and networks, with NDformer accelerating the search by three orders of magnitude. In two natural systems with existing formulas, including a gene regulatory network and a microbial community, ND^2 discovered formulas from empirical data that corrected existing models, reducing errors by 59.98% and 55.94%. The corrected dynamics reveal higher-order interactions in gene regulatory networks and sensitivities inversely related to population size in microbial communities, both absent in existing models. For epidemic transmission, a complex social system without established dynamics, ND^2 discovered network dynamics formulas across regions of various spatial scales. All discovered dynamics exhibit the same power-law distribution of node correlations, revealing a common pattern across different spatial scales. Comparative analysis between the USA and China reveals differences in nonpharmaceutical intervention effects, demonstrating our method's utility in understanding complex systems. In summary, ND^2 provides a powerful tool for automatically discovering network dynamics formulas, bridging the gap between accumulated observational data and theoretical understanding in complexity science and fields such as ecology, molecular biology and neuroscience.

Result

ND^2 framework for discovering network dynamics

Many complex systems, from gene regulatory networks[29] to epidemic spreading[30], can be described as dynamic processes where the states of N nodes, $\mathbf{x} = [x_1, \cdots, x_N]^T$, evolve according to interactions encoded in a network structure A_{ij}, following the dynamics $\mathbf{f} = [f_1, \cdots, f_N]^T$. A typical example is the pairwise model[5,25,31]:

$$\frac{\mathrm{d}}{\mathrm{d}t}x_i = f_i(\mathbf{x}, A) = W(x_i) + \sum_{j=1}^{N} A_{ij} Q(x_i, x_j), \tag{1}$$

where the two terms on the right-hand side describe the self-dynamics and interaction dynamics of node i with its neighbors, respectively. The nonlinear functions $W(x_i)$ and $Q(x_i, x_j)$ encode the system's governing laws, and the connectivity matrix $A_{ij} \in \{0, 1\}$ (or $A_{ij} \in \mathbb{R}$ for weighted networks) captures the (weighted) interactions between nodes. To discover $\mathbf{f}$ from node activities $\mathbf{x}(t)$ and network structure A_{ij}, we introduce ND^2, a neural symbolic regression method that reduces the search space with network dynamical operators and enables efficient, robust discovery with a pretrained NDformer.

We design three network dynamical operators: source (ϕ_s), target (ϕ_t) and aggregation (ρ) inspired by message-passing in graph networks[32]. Rather than treating each node state x_i separately, these operators act on the full state vector $\mathbf{x} \in \mathbb{R}^N$. As shown in Fig. 1b, ϕ_s and ϕ_t map node-level variables to edge-level variable $\mathbf{y} \in \mathbb{R}^E$ (E denotes the number of edges) by selecting the source and target nodes of each edges, while ρ aggregates edge-level variables back to nodes by summing incoming edges (formal definitions are given in Methods). Figure 1d illustrates a three-node example where node activities $x_i(t)$ are vectorized into $\mathbf{x}(t)$ (left). Operators ϕ_s and ϕ_t extract source and target node variables for each edge, then apply element-wise transformations using sine and subtraction functions (middle). Operator ρ aggregates edge states back to nodes, with zero values for nodes 2 and 3 having no incoming edges (right). As shown in Fig. 1c, these operators enable the expression of network dynamics regardless of the network dimensionality n, reducing the search space of a high-dimensional network to an equivalent one-dimensional system.

Despite the reduced space, it remains too large for existing symbolic regression algorithms. Therefore, we propose an NDformer-guided symbolic regression algorithm combining neural guidance with symbolic search (Fig. 1e). NDformer incorporates a GNN[26] and transformer[27] to encode network structure and node activity data, estimating probability distributions over symbols for formula construction (see 'Design of NDformer' in Methods). MCTS selects symbols, guided by these probabilities, to construct candidate formulas, which are then evaluated by a reward calculator according to their accuracy and simplicity. The resulting rewards further guide MCTS in generating better candidates, achieving higher accuracy with shorter formula lengths (see 'NDformer-guided symbolic search' in Methods). Compared with classic MCTS methods[23], ND^2 leverages NDformer to capture underlying patterns in the data and guide MCTS to prioritize promising symbol combinations, thereby accelerating the search process. Unlike existing tabular-based methods that cannot capture node interactions[20,21], ND^2 is specifically designed for network dynamics through its architecture and pretraining scheme, enabling efficient guidance of network dynamics discovery (detailed discussions in Supplementary Section 4.5). As shown in Fig. 1f, NDformer is pretrained on a large-scale, universal dataset of one million samples comprising randomly generated network dynamics formulas, synthetic network structures and corresponding node activity data. To ensure diversity, we generate Erdős–Rényi, Watts–Strogatz, Barabási–Albert and complete graph networks with varying sizes and degrees. Node activities are drawn from a Gaussian mixture model to embed low-dimensional attractors commonly seen in real systems. During pretraining, NDformer learns to predict the next formula symbol from preceding symbols, using network structure and node activities as contextual input (see 'Pretraining NDformer' in Methods). The pretrained NDformer then empowers symbolic search algorithms to efficiently discover network dynamics formulas in the reduced search space expressed by network dynamics operators.

Recovering ground-truth network dynamics of simulated systems with efficiency

To validate the capability of ND^2 in discovering accurate governing laws from data, we simulate ten representative model systems spanning well-known network dynamics, including Kuramoto[25], coupled Rössler oscillator[25], homogeneous coupled Rössler[25], FitzHugh–Nagumo[25], Wilson–Cowan[31], gene regulatory[25], Michaelis–Menten[31], Lotka–Volterra[31], mutualistic population[31] and susceptible-infected-susceptible[31] models (detailed in Supplementary Section 4.2).

Figure 2a shows node activities simulated with coupled Rössler oscillator dynamics on an Erdős–Rényi network, where each node has three state variables (x_i, y_i and z_i), initialized with random conditions and intrinsic frequencies ω_i, and evolves under neighbor influence. ND^2 accurately recovers the governing formulas for all three states, capturing both functional forms and parameters, and reproduces derivatives and long-term trajectories (Fig. 2b–d). Beyond this example, ND^2 also identifies the correct dynamics across nine additional systems on both synthetic and empirical networks (Supplementary Section 4.4).

We compare ND^2 with two state-of-the-art methods, physics-incorporated graph network (PIGN)[33] and two-phase inference[25] (see experimental setting in Supplementary Section 4.3). As shown in Fig. 2e, ND^2 outperforms both in recovering formulas and predicting derivatives. The two-phase method, based on sparse identification of nonlinear dynamics (SINDy)[24], is limited to fitting dynamics within a predefined library. A smaller library may lack expressiveness, while a larger library requires sufficient data to avoid underdetermination, often failing to capture the correct formula when prior knowledge is insufficient. By contrast, ND^2 uses symbolic regression to discover formulas of arbitrary form, enabling accurate recovery of network

https://doi.org/10.1038/s43588-025-00893-8

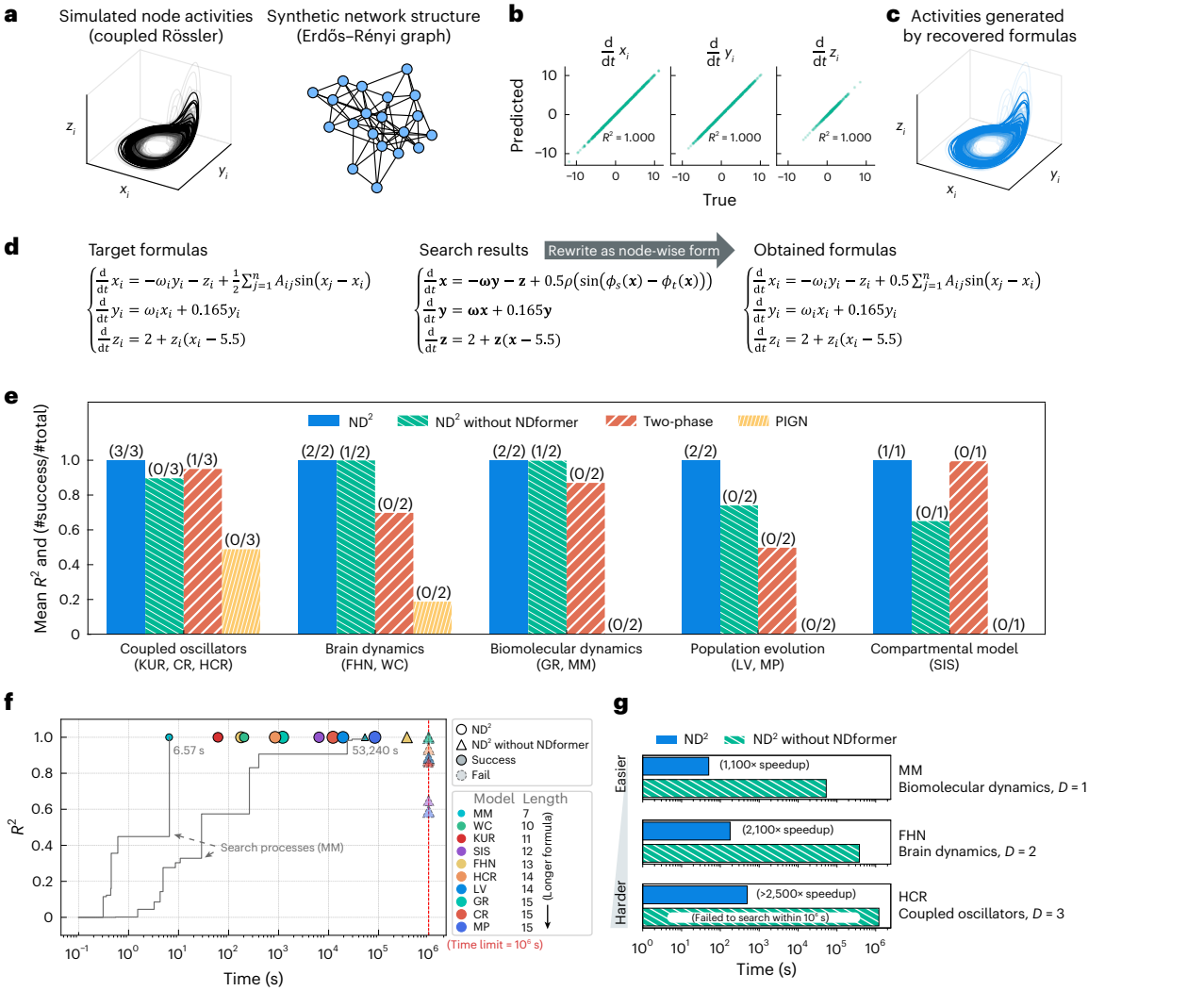

Fig. 2 | Synthetic experiment. a–d, Recovery results of coupled Rössler dynamics simulated on a synthetic Erdős–Rényi network (**a**), including the predicted time derivatives (**b**) and generated long-term activities (**c**) of the recovered formulas (**d**). **e**, Experiment results of our proposed ND^2 method, modified ND^2 with NDformer removed, two-phase inference and PIGN on five categories of dynamics, including Kuramoto (KUR), coupled Rössler (CR) oscillator, homogeneous coupled Rössler (HCR), FitzHugh–Nagumo (FHN), Wilson–Cowan (WC), gene regulatory (GR), Michaelis–Menten (MM), Lotka–Volterra (LV), mutualistic population (MP) and susceptible-infected-susceptible (SIS). The numbers above the bars indicate the number of successfully recovered formulas (that is, find correct or mathematically equivalent formula form) out of the total, while the height of the bars represents the average R^2 of results. Our method successfully discovered all ten models with R^2 values of 1, suggesting its ability to recover network dynamics formulas. **f**, The synthetic experiment of recovering dynamics on ten simulated systems using ND^2 with and without NDformer. The search time and R^2 of discovered formulas are depicted, with markers' shapes representing whether NDformer is used and sizes representing the formula length. ND^2 successfully recovers all ten dynamics. The gray lines show the search processes for the Michaelis–Menten (MM) dynamics, where the NDformer-guided search costs 99.91% less time. **g**, Search time to recover three models with and without NDformer, where the guidance of NDformer speeds up the search by three orders of magnitude.

dynamics. We provide further discussion of experimental results as well as more baseline methods in Supplementary Section 4.4.

We further tested ND^2 under unfavorable conditions using the same settings, finding it can recover governing laws with observational noise signal-to-noise ratio as low as −10 to approximately −5 dB, dynamical noise signal-to-noise ratio of 25 to approximately 35 dB and up to ~45–50% missing or spurious edges (Supplementary Section 6.2). Remarkably, without relying on prior knowledge, ND^2 achieves performance comparable to the two-phase method with strong prior knowledge. ND^2 also succeeds in more challenging scenarios. When the network structure is unknown, using only node activity as input, it recovers Kuramoto dynamics and the underlying 236-node network by treating edge presence as optimizable parameters (see 'Discover dynamics beyond pairwise models' in Methods). It also handles networks with unknown edge weights, heterogeneous networks where different node classes follow distinct dynamics, and community-structured networks unseen during NDformer pretraining, demonstrating strong generalization (Supplementary Section 6.5).

To assess NDformer's acceleration capability, we implemented a version of ND^2 without it, finding that the search rarely recovers target formulas in a reasonable time (Fig. 2f). NDformer accelerates symbolic regression by three orders of magnitude (Fig. 2g), thanks to its pretraining process, in which it learns to predict formula symbols from the network structure and node activities $\mathbf{x}(t)$ across one million diverse dynamical systems (see 'Pretraining NDformer' in Methods). During pretraining, node states are sampled independently, $x_i(t_n) \overset{\text{i.i.d.}}{\sim} p_\theta$ (where ~ denotes 'distributed as'), avoiding numerical instability from iterating $x_i(t_n) = x_i(t_{n-1}) + f_i(\mathbf{x}(t_{n-1}), A)\Delta t$ (Methods). To capture the low-dimensional manifolds typical in dynamical systems, such as fixed points, limit cycles and other attractors[34], p_θ is modeled as a Gaussian mixture. Compared with uniform distribution, this design yields a 58-fold speedup in recovering FitzHugh–Nagumo

https://doi.org/10.1038/s43588-025-00893-8

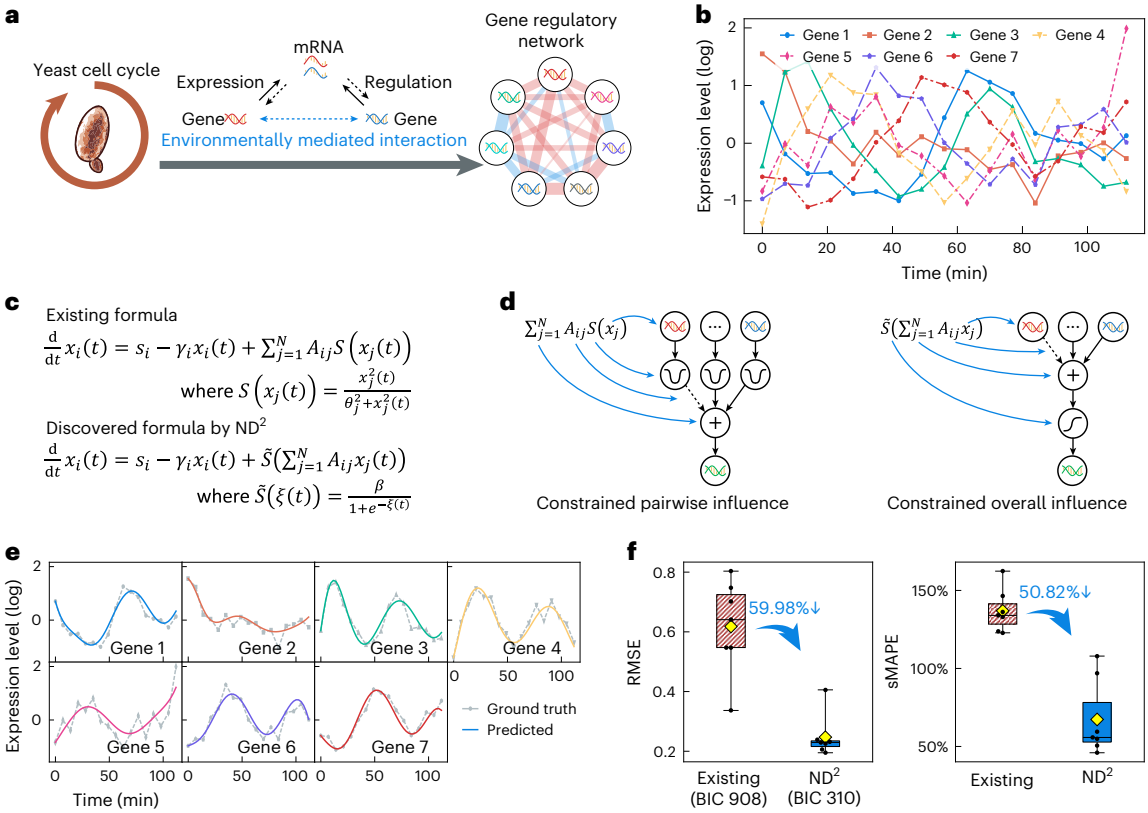

Fig. 3 | Gene regulatory dynamics. a, In the division cycle of yeast cells, the environmentally mediated regulation among genes can be described by a fully connected network. **b**, Gene expression levels (measured by the logarithm of the amount of expressed mRNA) served as the node activities. **c**,**d**, The comparison of the existing formula and the corrected formula (**c**) in terms of the order of action of aggregation and nonlinear operators (**d**). **e**,**f**, The generated node activities fit the ground truth well (**e**), decreasing RMSE by 59.68% and sMAPE by 50.82%. **f**, Box plots show the median (center line), 25th–75th percentiles (box), minimum/maximum (whiskers) and mean values (yellow diamond), and individual data points are overlaid as dots ($n = 7$).

dynamics, which has a one-dimensional limit cycle in the state space (Supplementary Fig. 10).

Correcting existing network dynamics in empirical systems to reveal scientific insights

Proper cellular function relies on timely, context-specific gene expression. Genes regulate cellular processes through their expression, which is in turn shaped by these processes, forming a complex self-regulating system. Understanding how genes promote or inhibit each other's expression is a central challenge in molecular biology. Unlike typical networks with direct interactions, genes influence others indirectly via environmentally mediated mRNA expression (Fig. 3a). Traditionally, gene dynamics are modeled using the Hill equation[3], $\frac{d}{dt}x_i(t) = s_i - \gamma_i x_i(t) + \sum_j A_{ij} S(x_j(t))$, where s_i and γ_i are basal synthesis and degradation rates, A_{ij} denotes the pairwise interaction strengths and $S(x) = x^2/(x^2 + \theta_j^2)$ is a bounded function ($|S(x)| \le 1$) with parameters θ_j. Originally derived from two-component systems and extended to multigene networks via additive assumptions, this mathematical model cannot accurately capture the true dynamics of gene regulation. Despite growing observational data, more accurate models remain lacking, hindering research on key problems that rely on dynamic models, such as network inference and stability analysis. Here, we apply ND^2 to empirical gene expression data to address this limitation.

We use yeast cell division cycle data, with gene expression levels measured every 7 min over 119 min (ref. 35) (approximately two cell cycles; details in Supplementary Section 5.1.1). Genes were grouped by biological function, with synchronized activity within each group, which we treated as distinct components (Fig. 3b). Because the underlying network structure is unobservable, we assume a fully connected network and let our method fit the connection weights A_{ij}. ND^2 discovered a corrected formula

$$\frac{d}{dt}x_i(t) = s_i - \gamma_i x_i(t) + \beta\tilde{S}\left(\sum_{j=1}^{n} A_{ij} x_j(t)\right), \qquad (2)$$

where s_i, γ_i, β and A_{ij} are parameters, and $\tilde{S}(x) = (1 + \exp(-x))^{-1}$ is the logistic function (Fig. 3c; details in Supplementary Section 5.1.2). Like $S(x)$, $\tilde{S}(x)$ is bounded ($|\tilde{S}(x)| \le 1$), but it acts on the sum of neighbors rather than individually (Fig. 3d). Consequently, $\frac{\partial \dot{x}_i}{\partial x_j} = \beta A_{ij}\tilde{S}(\sum_k A_{ik}x_k)(1 - \tilde{S}(\sum_k A_{ik}x_k))$ depends on nodes beyond i and j, reflecting a higher-order effect where one gene's influence on another can be modulated by additional genes. This aligns with higher-order interactions widely observed in many natural systems with environmentally mediated interactions[36] and other advanced gene regulatory models (Supplementary Section 5.1.4). As shown in Fig. 3e,f, the corrected formula accurately predicts gene expression levels and reproduces the oscillatory dynamics observed in real data. It reduces the root mean squared error (RMSE; see 'Discover dynamics from empirical data' in Methods) by 59.98% and the symmetric mean absolute percentage error (sMAPE) by 50.82%, and lowers the Bayesian information criterion[37] (BIC) from 908 to 310, indicating both improved accuracy and a better functional form rather than increased model complexity. It also outperforms other advanced gene regulatory models and GNNs (Supplementary Section 5.1.4) and demonstrates

https://doi.org/10.1038/s43588-025-00893-8

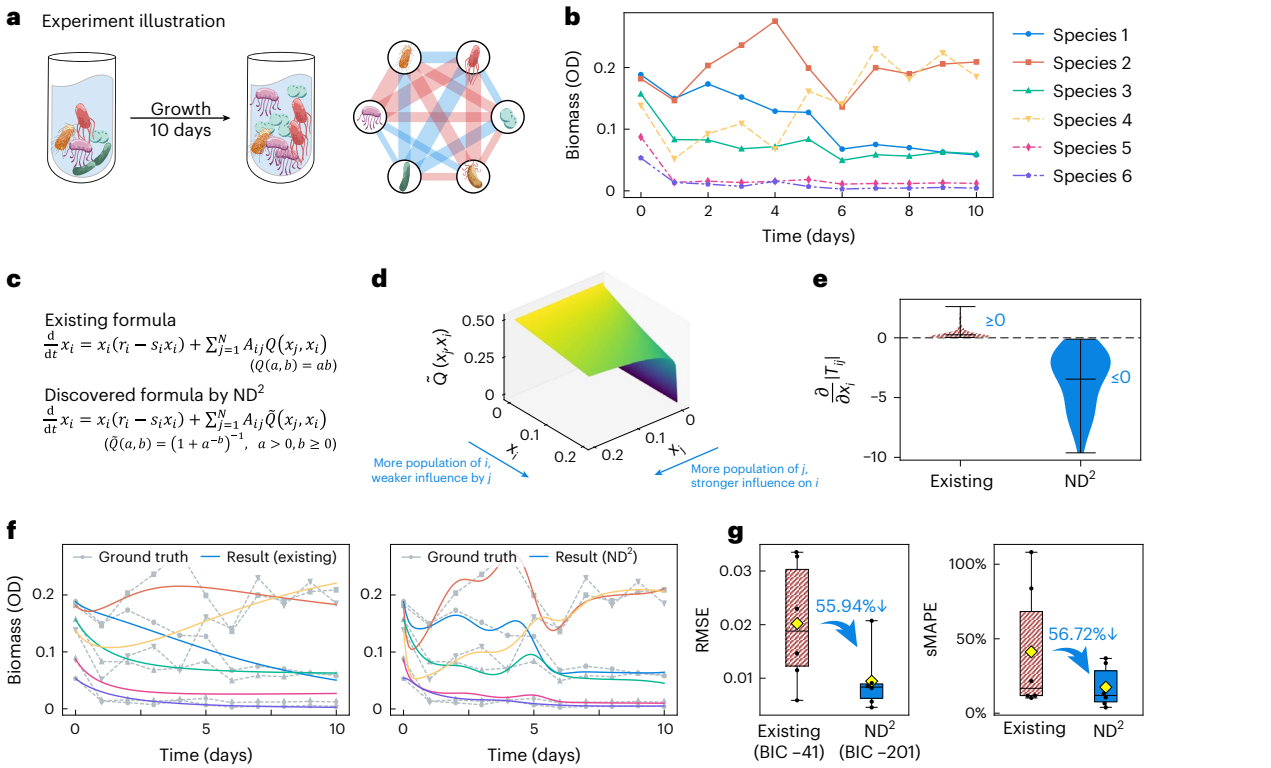

Fig. 4 | Microbial community dynamics. a,**b**, Six bacterial species grow together for 10 days, resulting in a network dynamical system with a fully connected network structure (**a**). The biomass of each species was measured in optical density (OD) daily as node activities (**b**). **c**–**e**, A comparison of the existing formula and the corrected formula (**c**). The corrected result introduces a different interaction term (**d**; with isosurface color indicating $\tilde{Q}$ values), where the influence strength of species j on i decreases as i's population grows (**e**). The center line and whiskers in **e** show the mean and minimum/maximum values of 30 edges in 10 days ($n = 300$). **f**,**g**, The node activities generated by the corrected formula fit the empirical data better than the existing formula (**f**), decreasing the RMSE by 55.94% and sMAPE by 56.72% (**g**). Box plots in **g** show the median (center line), 25th–75th percentiles (box), minimum/maximum (whiskers) and mean values (yellow diamond), and individual data points are overlaid as dots ($n = 6$).

generalizability across gene networks with varying numbers of genes (Supplementary Section 5.1.5).

Microbial communities are another important natural system, where species population growth is often modeled by the Lotka–Volterra equation[38]: $\frac{\mathrm{d}}{\mathrm{d}t}x_i(t) = (r_i - s_i x_i(t))x_i(t) + \sum_{j=1}^{n} A_{ij}Q(x_j(t), x_i(t))$, where r_i and s_i are growth and self-regulation rates, A_{ij} is the interaction between species and $Q(x, y) = xy$ is the interaction term. We analyze empirical data (Fig. 4a,b) of $n = 6$ bacterial species growing over 10 days[39], using a fully connected graph with optimizable edge weights (Supplementary Section 5.2.1). Applying ND², we obtain a corrected formula

$$\frac{\mathrm{d}}{\mathrm{d}t}x_i(t) = (r_i - s_i x_i(t))x_i(t) + \sum_{j=1}^{n} A_{ij}\tilde{Q}(x_j(t), x_i(t)), \tag{3}$$

where r_i, s_i and A_{ij} are parameters, and $\tilde{Q}(x, y) = (1 + x^{-y})^{-1}$ is the interaction term (Fig. 4c; details in Supplementary Section 5.2.2), which saturates with x_j (Fig. 4d), consistent with existing extensions to Lotka–Volterra (Supplementary Section 5.2.4). However, $\tilde{Q}(x_i, x_j)$ decreases with x_i, revealing a key difference in sensitivity. We consider $T_{ij} \triangleq A_{ij}Q(x_j, x_i)$, which measures how species j affects i's growth. As shown in Fig. 4e, in the Lotka–Volterra model, $\frac{\partial}{\partial x_i}|T_{ij}| \geq 0$, meaning species become more sensitive as their populations grow. In our corrected dynamics, however, $\frac{\partial}{\partial x_i}|T_{ij}| \leq 0$, indicating larger populations are less influenced by others (details in Supplementary Section 5.2.3). This challenges assumptions underlying the Lotka–Volterra model. In Fig. 4f,g, we compare model accuracy, finding our formula reduces RMSE by 55.94%, sMAPE by 56.72% and BIC from −41 to −201. It also outperforms other advanced ecological models and GNNs (Supplementary Section 5.2.4), and tests on larger communities (12, 24 and 48 species) and varying nutrient concentrations confirm its generalizability (Supplementary Section 5.2.5). Together with results from gene networks, these findings demonstrate that ND² can correct existing network dynamics formulas, not only achieving higher accuracy but also providing scientific insights beyond current understanding.

Discovering epidemic transmission dynamics with interpretability

Epidemics spreading across different regions of the world have a profound societal impact, disrupting economies, hindering education and claiming countless lives. Understanding its transmission via human mobility is crucial, yet precise dynamics formulas remain elusive due to system complexity and unknown underlying principles. Epidemic dynamics vary across scales: city-level spread is driven by daily interactions, whereas global spread depends on policies, socioeconomic and environmental factors. Differences in nonpharmaceutical interventions further complicate transmission[40], rendering traditional metapopulation models inadequate. Here, we apply ND² to discover transmission dynamics at multiple spatial scales worldwide.

As shown in Fig. 5a, we study coronavirus disease 2019 (COVID-19) transmission across seven regions at four spatial scales (city, state, country and global), where weekly confirmed cases $x_i(t)$ over 163 weeks serve as node activities, and traffic flows M_{ij} define the network structure (Supplementary Section 5.3.1). ND² discovers transmission dynamics in each region, which, despite regional differences, share a common structure as listed in Fig. 5b, that is, a self-evolutionary term $W(x_i)$, an interaction term $Q(x_j, x_i)$ and a function Φ acting on the total interaction $S_i = \sum_{j=1}^{n} A_{ij}Q(x_j, x_i)$:

https://doi.org/10.1038/s43588-025-00893-8

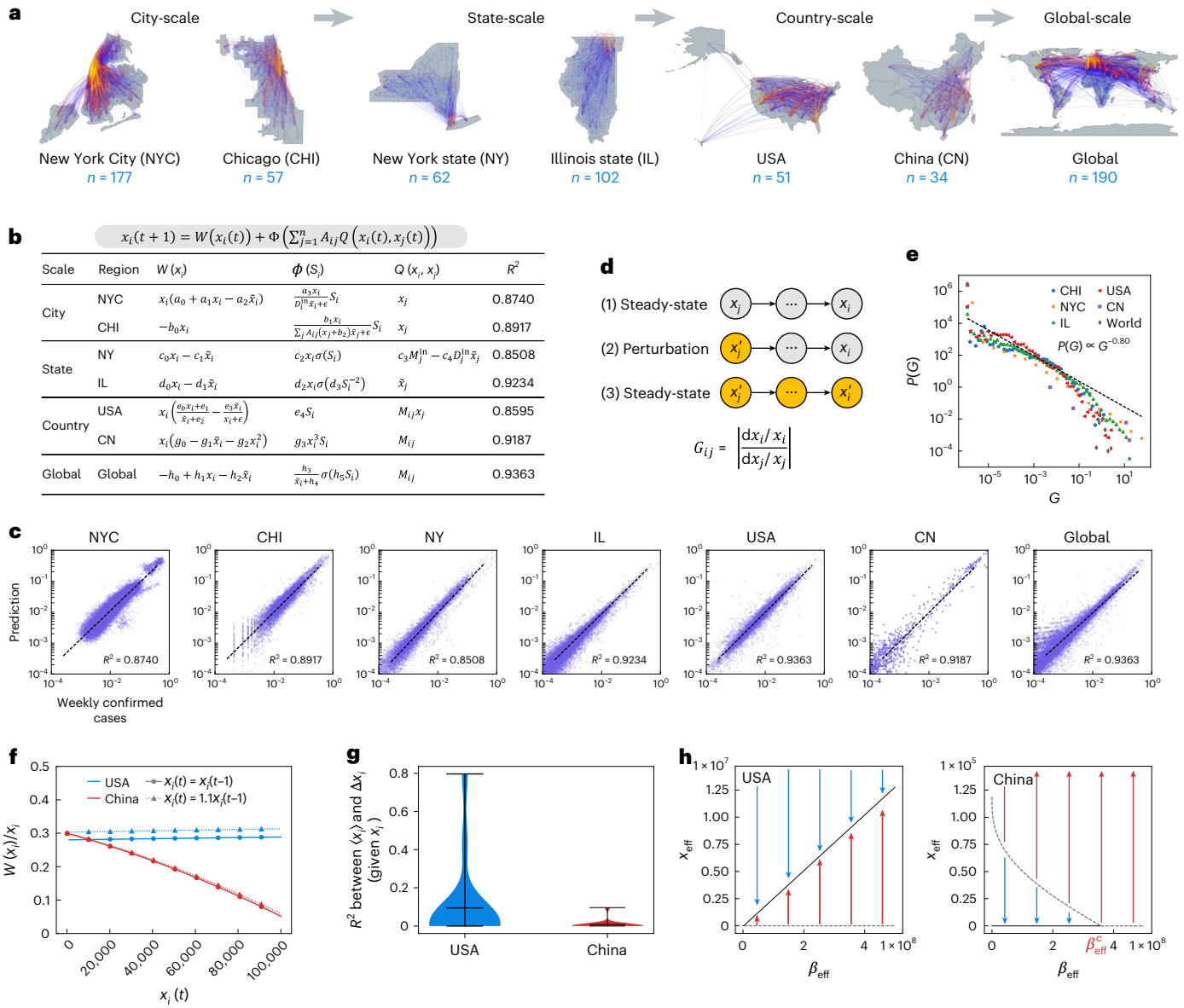

Fig. 5 | Epidemic transmission dynamics. a, The dataset consists of seven different regions at four spatial scales (city, state, country and global) worldwide, where the color of curves represents the traffic flow between different areas. **b**, Discovered epidemic dynamics in all seven regions, where $x_i(t)$ denotes the confirmed cases in the ith area at the tth week, $\tilde{x}_i(t) = x_i(t) + x_i(t-1)$ denotes the confirmed cases in history 14 days, A_{ij} denotes the network structure, $D_i^{\text{in}} = \sum_j^n A_{ij}$ denotes the in-degree of i, M_{ij} denotes the flow of people from j to i, $M_i^{\text{in}} = \sum_j M_{ij}$ denotes the total-in flow of i, $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function, ϵ is a small nonzero constant, and a_i - h_i are parameters. **c**, Discovered dynamics fit well with the ground-truth results. **d**,**e**, Perturbation experiments (**d**) show the same power-law distribution properties for all discovered dynamics $P(G) \propto G^{-0.80}$ (**e**). **f**–**h**, Comparison of the discovered COVID-19 dynamics in the USA and China in terms of self-evolution dynamics (**f**), infection dynamics (**g**) and steady-state analysis (**h**). The center line and whiskers in **g** mark the mean and minimum/maximum values ($n = 15$).

$$x_i(t+1) = W(x_i(t)) + \Phi\left(\sum_{j=1}^{n} A_{ij} Q(x_j(t), x_i(t))\right), \tag{4}$$

where $A_{ij} = \mathbb{I}_{(M_{ij} > 0)}$(details in Supplementary Section 5.3.2). To validate the discovered dynamics, Fig. 5c compares true and predicted weekly cases, showing $R^2 > 0.85$ across all regions. This accuracy also generalizes to unseen periods, confirmed by training–test splits along the time dimension (Supplementary Section 5.3.7). The physical meaning of these formulas can be interpreted item by item. For example, in the New York City region, the discovered formula $x_i(t+1) = x_i(t) \times (a_0 + a_1 x_i(t) - a_2 \tilde{x}_i(t) + a_3 \frac{\sum_{j=1}^n A_{ij} x_j(t)}{D_i^{\text{in}} \tilde{x}_i(t) + \epsilon})$ estimates $x_i(t+1)$ as $x_i(t)$ multiplied by a four-term correction factor. The first three terms adjust the basic correction rate a_0 using infections in the past 7 and 14 days, while the last term characterizes imported infections, moderated by the in-degree D_i^{in} and historical 14-day cases $\tilde{x}_i(t)$ in the denominator, possibly reflecting interventions such as lockdowns or school closures in areas with high traffic and recent outbreaks. Full explanations are provided in Supplementary Section 5.3.2, which show consistency with existing metapopulation models and offer additional insights (Supplementary Section 5.3.3). The discovered dynamics also enable analysis of pairwise node correlations, a common tool in complex networked systems[5]. As shown in Fig. 5d, a perturbation dx_j on node j propagates to node i, giving $G_{ij} = |\frac{dx_i/x_i}{dx_j/x_j}|$. Across all seven regions, G follows the same power law $P(G) \propto G^{-0.80}$ (Fig. 5e and Supplementary Section 5.3.4), despite differences in formula forms. This discovery

reveals a common pattern in system responses to perturbation across different countries and spatial scales worldwide, clarifying transmission characteristics of epidemic spreading and supporting the validity and universality of the discovered dynamics.

To demonstrate ND^2's value in helping understand system properties, we compare the discovered dynamics in the USA and China. Both countries' $W(x_i)$ is $x_i(t)$ multiplied by a correction factor, which exhibits self-inhibition in China as $x_i(t)$ grows, but remains constant in the USA (Fig. 5f). Interaction dynamics $Q(x_j, x_i)$ also differ. In the USA, it depends on neighboring infections $\langle x_i \rangle := \sum_j M_{ij} x_j$, whereas in China it is independent of x_j, indicating negligible interaction effects. This is confirmed in Fig. 5g, where the squared Pearson correlation between $\langle x_i \rangle$ and $\Delta x_i = x_i(t+1) - x_i(t)$ (details in Supplementary Section 5.3.5) is high in the USA (up to 0.8) but much lower in China (less than 0.1). These differences probably reflect China's aggressive interventions, including lockdowns, contact tracing and targeted measures to limit transmission in highly infected areas and minimize cross-regional infections. Finally, we analyze steady-state properties using Gao et al.'s method[3], which derives the dynamics of average infections $x_{\text{eff}}(t) = \sum_i w_i x_i(t)$ under given average cross-regional traffic $\beta_{\text{eff}} = \sum_i w_i M_i^{\text{in}}$, where $w_i = M_i^{\text{out}} / \sum_j M_j^{\text{out}}$ and $M_i^{\text{in}} = \sum_j M_{ij}$, $M_i^{\text{out}} = \sum_j M_{ji}$ are total-in and total-out flows (Supplementary Section 5.3.6). As shown in Fig. 5h, China's phase diagram has a tipping point β_{eff}^c: when $\beta_{\text{eff}} < \beta_{\text{eff}}^c$, x_{eff} stabilizes at 0, corresponding to a controlled epidemic state. When $\beta_{\text{eff}} > \beta_{\text{eff}}^c$, x_{eff} diverges, indicating a severe outbreak if cross-provincial traffic exceeds this threshold. In the USA, however, the stable state varies linearly with β_{eff} and remains finite, so interventions on traffic only linearly affect infections with smaller impacts than in China. These findings are consistent with the different effects of nonpharmacological interventions between these two countries and demonstrate ND^2's ability to reveal system properties.

Discussion

Symbolic regression, which discovers formulas revealing underlying patterns in data, is a crucial direction for automated scientific discovery[9]. While widely applied in low-dimensional dynamical[13,14,16] and nondynamical systems[12,15], it has been considered ineffective for high-dimensional networks[9]. Here, by designing a neural symbolic regression method, we show that symbolic searches on high-dimensional networks can be reduced to searches on equivalent one-dimensional systems. Our work represents a theoretical breakthrough that expands the scope of symbolic regression to complex networked systems, thereby remarkably enhancing its applicability and potential.

In addition to symbolic regression, Gao and Yan proposed a SINDy-based method for network dynamics identification[25], which represents system behavior as a sparse linear combination of functions from a predefined library. Their approach has also been applied to stochastic complex systems[41]. While efficient for identifying dominant terms that govern system dynamics[24], its expressiveness is limited by the library, whose size is itself constrained by the available data, restricting its applicability to unknown systems. By contrast, our method is more expressive and automated, capable of discovering formulas of any form without prior knowledge, enabling the discovery of network dynamics in complex, unseen systems.

ND^2 advances traditional human-driven discovery to machine-driven discovery of network dynamics, particularly valuable for complex systems such as transportation[42], disaster management[30] and economics[43], where empirical complexity and limited knowledge constrain traditional human-driven approaches. The dynamics discovered can provide insights into these systems, including inferring node correlations[5], revealing resilience properties and phase transitions[3], as demonstrated in our analysis of COVID-19 transmission dynamics. ND^2 can also advance the research of complexity science[3–5] by discovering diverse network dynamics formulas exhibiting richer behaviors across a wide range of complex systems.

Despite its strong performance, expanding ND^2 to scenarios with higher-order interactions, nonadditive aggregations and unknown network structures requires further study. Higher-order interactions, observed in brain cortex dynamics[44], species interactions[45] and social conventions[46], can be modeled by extending network dynamics operators to hyperedges (see 'Discover dynamics beyond pairwise models' in Methods). However, adapting NDformer for higher-order dynamics is nontrivial. First, the diversity of higher-order formulas makes random formulas inadequate for pretraining, necessitating more realistic formulas generated using large language models[47]. Second, integrating NDformer with hypergraph representation learning is required to capture latent features of higher-order interactions. Finally, given the difficulty of acquiring hypergraph structures, ND^2 could synergize with hypergraph inference methods[48] to jointly discover hypergraph structures and higher-order dynamics. Aggregation operators could also be extended to maximum or product to capture nonadditive behaviors, with corresponding adaptations for NDformer (Methods). When network structure is unobservable, ND^2 can recover dynamics and structures for networks with hundreds of nodes by optimizing edge existence (Methods), but this approach scales poorly. Linear regression requires more time steps than nodes, and simply increasing time steps slows computation for large networks. Future improvements may involve sparse regression or integrating network inference methods.

Our method selects formulas that best balance accuracy and complexity (Pareto front). However, context-based criteria, such as consistency with existing theory or physical intuition, may be preferable[9], particularly when data are insufficient or noisy. In such cases, simple functions can match the accuracy of the ground-truth formula (if any), making them nonidentifiable[49] (detailed discussions in Supplementary Section 2.4). At this stage, we introduce human intelligence to evaluate and select from the discovered Pareto front. Integrating large language models into symbolic regression[22] offers a promising approach, which can incorporate domain knowledge to construct or assess formulas in line with human preference.

Methods

In this section, we provide a detailed description of our ND^2 method, including the formal definition of network dynamical operators (part 1), the symbolic search method guided by NDformer (part 2), the NDformer's model architecture (part 3) and pretraining process (part 4), and the method for discovering formulas from empirical data (part 5).

Network dynamical operators

To reduce the search space that grows super-exponentially with the number of system variables, we treat all nodes' scalar states or edges' scalar weights as a whole 'vectorized' variable. The vectorized variables are thus categorized into two types: node-level variables $\mathbf{v} \in \mathbb{R}^N$ (for example, nodes' states and in-degrees) and edge-level variables $\mathbf{e} \in \mathbb{R}^E$ (for example, edges' weights and state difference of neighboring nodes), where N and E denote the number of nodes and edges in the network, respectively. The proposed three network dynamical operators, inspired by the node update blocks and edge update blocks in graph networks[32], can map variables between node-level and edge-level in the following ways:

$$\rho : \mathbb{R}^E \to \mathbb{R}^N, \quad \mathbf{v} = \rho(\mathbf{e}) \iff v_i = \sum_{j=1}^{N} A_{ij} e_k \quad (\forall i = 1 \cdots N), \tag{5}$$

$$\phi_s : \mathbb{R}^N \to \mathbb{R}^E, \quad \mathbf{e} = \phi_s(\mathbf{v}) \iff e_k = v_j \quad (\forall k = 1 \cdots E), \tag{6}$$

$$\phi_t : \mathbb{R}^N \to \mathbb{R}^E, \quad \mathbf{e} = \phi_t(\mathbf{v}) \iff e_k = v_i \quad (\forall k = 1 \cdots E), \tag{7}$$

where the kth edge connects from the jth node to the ith node, and A represents the adjacency matrix of the network. Intuitively, the

ρ operator produces a node-level variable by aggregating each node's incoming edges, while the ϕ_s/ϕ_t operator produces an edge-level variable by picking each edge's source/target nodes (Fig. 1b). To integrate the vectorized variables and proposed network dynamical operators with other mathematical operators, we further define that the vectorized variables of the same category can directly operate with each other in an element-wise manner (for example, $\mathbf{v}_1 + \mathbf{v}_2$, $\mathbf{e}_1 \times \mathbf{e}_2$, and $\sin(\mathbf{v})$) and produce a new vectorized variable of the same category. Conversely, variables of different categories cannot operate with each other before they are mapped to the same category using network dynamical operators (for example, $\mathbf{v} + \rho(\mathbf{e})$ and $\mathbf{e} \times \phi_s(\mathbf{v})$). Figure 1d illustrates a three-node network example that demonstrates how the proposed network dynamics operators can be combined to represent the interaction term in Kuramoto dynamics, that is, $\sum_{j=1}^{N} A_{ij} \sin(x_i - x_j)$. The left panel shows the states x_i of each node, which can be stacked into a vectorized node-level variable $\mathbf{x} = [x_1, x_2, x_3]$. The middle panel demonstrates how the operators ϕ_s and ϕ_t map the node-level variable $\mathbf{x}$ to the edge level by selecting the source and target nodes of each edge. The resulting two edge-level variables, $\phi_s(\mathbf{x})$ and $\phi_t(\mathbf{x})$, are then element-wise transformed through subtraction and the sine function to compute the sine of the phase differences across edges, that is, $\sin(x_2 - x_1)$ and $\sin(x_3 - x_1)$. The right panel illustrates how these sinusoidal phase differences on the edges are aggregated to the target nodes, resulting in a node-level variable that captures neighbor interaction. For node 1, the aggregation yields the sum of sinusoidal phase differences from its incoming edges, which corresponds exactly to the interaction term of Kuramoto dynamics at that node. For nodes 2 and 3, the aggregation yields zero, as they have no incoming edges, which also aligns with the Kuramoto interaction term at their locations.

NDformer-guided symbolic search

In this work, we propose a neural-symbolic algorithm to search for the target formulas, where the symbolic part, MCTS[28], searches for formulas under the guidance of the neural part, an NDformer. Although previous work has demonstrated that MCTS can efficiently perform symbolic searches on low-dimensional systems[23], symbolic search on high-dimensional networked systems—despite our efforts to reduce the search space by introducing network dynamical operators—remains too large for the pure MCTS algorithm. To address this, we introduce an NDformer-guided MCTS algorithm, which includes a symbolic part for searching and a neural part for guiding the search. The neural part, NDformer, learns to capture latent features of the system's underlying dynamics and estimates the probability distribution over each symbol used to construct the formula. The symbolic part, MCTS, selects symbols according to the probabilities predicted by NDformer to construct candidate formulas. For each candidate formula, a reward calculator fits undetermined coefficients (if any) to the data using the Broyden–Fletcher–Goldfarb–Shanno algorithm[50] and returns a reward that evaluates the accuracy and simplicity. Formulas that better fit the data and have shorter lengths receive higher rewards, guiding MCTS to construct better candidate formulas.

MCTS maintains a search tree whose inner and terminal nodes represent incomplete formulas (for example, $\sin(\cdot), x + (\cdot)$) and complete formulas (for example, $\sin(x), x + y$), respectively. The complete formula can be evaluated by its length and goodness of fit to the observational data using a reward function proposed by Sun et al.[23]:

$$r(\mathrm{MSE}, c) = \frac{\eta^c}{1 + \mathrm{MSE}/\sigma_{\mathrm{out}}^2}, \tag{8}$$

where $\eta \leq 1$ is a hyperparameter, c is the length of the formula, σ_{out}^2 is the variance of the ground-truth outputs and MSE is the mean squared error between the formula's output and the ground truth. Therefore, a formula fitting the data more accurately with a more concise form will have a higher reward. For the incomplete formula, despite the inability to evaluate its goodness of fit, the NDformer can estimate the likelihood of different symbols filling into the incomplete part to fit the data, named the policy $\pi \in \mathbb{R}^S$ (S is the size of the symbol set as listed in Supplementary Section 2.1). The policies can indicate promising search directions at internal nodes where rewards cannot be calculated, thus guiding and accelerating the search process.

The MCTS algorithm iteratively adds notes into the search tree through a four-step cycle (illustrated in Supplementary Section 2.2):

(1) Selection. In this step, MCTS selects the most promising nodes to add to the search tree based on the recorded policies and rewards. Following Kamienny et al.[20], we evaluate the promising nodes using the predicted upper confidence bounds for trees (PUCT):

$$u(s,a) := q(s,a) + c_{\mathrm{PUCT}} \frac{\sqrt{1 + \sum_{a'} n(s,a')}}{1 + n(s,a)} \pi_s(a), \tag{9}$$

where $q(s,a)$ and $n(s,a)$ represent the recorded maximum reward and visit count of action a of node s, respectively (here 'action' means 'fill a symbol a into the incomplete part of s'), $\pi_s(a)$ is the ath value of the policy π_s, and c_{PUCT} is a hyperparameter that balances the exploration–exploitation trade-off. The first term of PUCT, $q(s,a)$, reflects the search history, while the second term, proportional to $\pi_s(a)$, reflects the NDformer's guidance. This guidance enables MCTS to make effective selections without exploring $q(s,a)$ for every action a, thereby accelerating the search process. Furthermore, to fully utilize NDformer's parallel processing capability in the expansion step, we adopt the beam search technique to select K nodes simultaneously, rather than selecting only one node like other works[20]. Specifically, we maintain two priority queues, $\mathcal{P} = \{(s_0, 0)\}$ and $\mathcal{Q} = \varnothing$, to store nodes s with the top-K 'route-averaged' PUCT values v_s and nodes to add into the search tree, respectively. Here, s_0 is the root node (for example, an empty formula) with a PUCT value of 0. We update them by

$$\mathcal{Q} \xleftarrow{\text{Enqueue}} \{(s', (1-\alpha)v_s + \alpha u(s,a)) \,|\forall (s, v_s) \in \mathcal{P}, \forall a, \text{ s.t. } s' \notin \text{ Search tree}\}, \tag{10}$$

$$\mathcal{P} = \text{Top-}K \text{ of } \{(s', (1-\alpha)v_s + \alpha u(s,a)) | \forall (s, v_s) \in \mathcal{P}, \forall a, \text{ s.t. } s' \in \text{ Search tree}\}, \tag{11}$$

until $|\mathcal{Q}| = K$, where $\alpha = d_s^{-1}$ is the inverse of the depth of node s in the search tree, action a acts on node s to produce a child node s', and v_s is the average PUCT value along the route from s_0 to s. The selected K nodes in $\mathcal{Q}$ are added to the search tree and used for subsequent steps.

(2) Expansion. For the added K nodes, MCTS initializes their $q(s,:)$ and $n(s,:)$ to zero and feeds them to the NDformer to obtain their policies π_s. The beam search technology adopted in the previous step enables NDformer to process $K > 1$ nodes at a time and thus fully utilize its parallel processing capabilities, which reduces the processing time and increases the overall search speed by a factor of 3–11 when $K = 10$ (Supplementary Section 2.3).

(3) Simulation. To evaluate the added nodes, MCTS estimates the maximum reward R_s MCTS estimates the maximum reward R_s of the terminal nodes in the subtree corresponding to each added node s. For each added node s, it samples M terminal nodes within the subtree of s following NDformer's policies and then calculates R_s as the maximum reward of these terminal nodes. It is worth noting that we use the maximum instead of the typically used average[20,23] because symbolic regression aims to find the optimal formula, rather than a series of formulas with the best average performance.

(4) Backpropagation. With estimated R_s of the added nodes, MCTS updates q and n along the routes from selected nodes to the root node as follows:

$$q(s,a) = \max\{q(s,a), R_s\}, \tag{12}$$

$$n(s,a) = n(s,a) + 1, \tag{13}$$

where a is the action of node s on the route.

MCTS repeats the four-step cycle until it meets the termination condition, such as finding a sufficiently accurate formula, reaching the time limit, or manual termination by the user. After termination, MCTS examines all terminal nodes in its search tree to identify the Pareto front. Specifically, it selects the optimal formulas of different lengths and then discards those with longer lengths but lower accuracy, thereby obtaining the best-fitting formulas under different length constraints.

Design of NDformer

We design the NDformer to capture the latent patterns from observed data and guide the search direction of MCTS. The NDformer takes the observed network structure and node activities, as well as incomplete formulas as input, and generates a probability distribution over the symbol set, named the policy, to evaluate the likelihood of each symbol filling in the incomplete part of the input formula to fit the observational data. It consists of three main components: (1) a network encoder, (2) a formula encoder and (3) a policy decoder (illustrated in Supplementary Section 3.1).

The network encoder embeds the network structure $A \in \mathbb{R}^{N\times N}$ and node activities $X \in \mathbb{R}^{T\times N}$ into context embeddings $H \in \mathbb{R}^{N_{\max}\times d_f}$, where N is the number of nodes, T is the number of time steps, d_f is the model dimension and $N_{\max}$ is a hyperparameter. Existing symbolic regression methods focused on nonnetwork systems $y = f(\mathbf{x})$ typically embed each time step's x–y pair $(\mathbf{x}(t), y(t))$ into a fixed-dimensional vector $\mathbf{v}(t) \in \mathbb{R}^{d_f}$, which represents a sample of the target formula $y(t) = f(\mathbf{x}(t))$ (ref. 19). In networked systems, however, although we can embed the x–y pair $(x_i(t), y_i(t))$ into a fixed-dimensional vector $\mathbf{v}_i(t) \in \mathbb{R}^{d_f}$ in the same way, it does not contain sufficient information to serve as a sample of the target formula $y_i(t) = f(\mathbf{x}(t); A, i)$, because this target formula depends on not only $x_i(t)$ but also its neighbors and the network structure A, which are not included in $\boldsymbol{\nu}_i(t)$. To address this issue, we introduce a GNN, where nodes share states with their neighbors through a message-passing mechanism:

$$\mathbf{v}_i \leftarrow \frac{1}{D_i^{\text{in}}} \sum_{j=1}^{N} A_{ij} M_0(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_k), \quad \forall i \in \{1 \cdots N\}, \tag{14}$$

$$\mathbf{e}_k \leftarrow M_1(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_k), \quad \forall i \in \{1 \cdots E\}, \tag{15}$$

where the kth edge links the jth node to the ith node, D_i^{in} is the in-degree of node i, $M_{0,1}$ are multilayer perceptrons (MLPs) with a hidden layer, $\mathbf{v}_i$ is the embedding of the ith node (we omit the time index t) and $\mathbf{e}_k \in \mathbb{R}^{d_f}$ is the embeddings of edge (initialized by the embedding of edge-level variables such as edge weights). After R rounds of updates, the obtained $\mathbf{v}_i$ and $\mathbf{e}_i$ contain enough information, including neighbor status information, to serve as a sample describing the target formula $y_i = f(\mathbf{x}; A, i)$. Subsequently, the updated node embeddings $\mathbf{v}_i(t)$ are encoded by a transformer encoder to produce the context embedding H. To avoid the $O(N^2)$ complexity of the self-attention mechanism in the transformer, we sample no more than $N_{\max}$ sample points from $\boldsymbol{\nu}_i(t)$. In addition, we omit the positional encoding here because the order of sample points is not important.

The input formula of length L is processed by the formula encoder, generating a formula embedding $F \in \mathbb{R}^{(L+4)\times d_f}$. Specifically, a formula can be seen as a formula tree, whose internal nodes are operators and leaf nodes are operands (that is, variables, undetermined parameters or constants). By performing a preorder traversal of the tree, we obtain a symbol sequence, named the formula's prefix notation. We add a pair of tokens (<SOS> and <EOS>) to indicate the start and end of the sequence, a token ([N] or [E]) to indicate the formula's type (node-level or edge-level), and a <QUERY_POLICY> token to specify the position in the output sequence from which to obtain the policy. Although the prefix notation already contains all the information from the original formula, we introduce two auxiliary embeddings to emphasize the tree structure of the formula as shown in Supplementary Fig. 4, including (1) the index of each node's parent node in the formula tree and (2) the type of each node's subtree (node-level or edge-level). These auxiliary embeddings, along with the positional encoding, are added to the prefix notation's embedding to obtain the final formula embedding.

Taking the context embedding and the formula embedding as input, the policy decoder generates an output sequence $E \in \mathbb{R}^{(L+4)\times d_f}$ with a transformer decoder. The embedding at the position specified by the <QUERY_POLICY> token is then fed into a MLP to generate the policy $\pi \in \mathbb{R}^S$, which is a probability distribution over the symbol set that evaluates how likely each symbol is to be used as the next symbol to complete the input formula and fit the input data.

Pretraining NDformer

To gain the capability of guiding the searches across various systems, the NDformer is pretrained on a large-scale universal dataset in a self-supervised manner (Supplementary Section 3.2). The dataset comprises 1 million randomly generated network dynamics formulas $\mathbf{f}$, network structures A and node activities $X(t)$. To generate $\mathbf{f}$, we start with an initial incomplete formula (for example, an empty formula) and iteratively fill it with randomly chosen symbols until it becomes complete. In addition, we ensure that each part of the formula (that is, subtrees with more than one node) contains at least one variable, thus preventing the generation of trivial formulas that consist solely of constants without any variables (detailed in Supplementary Section 3.3.1). To generate the network structure A, we first randomly select a network type from Erdős–Rényi, Watts–Strogatz, Barabási–Albert and complete graphs, and then generate a corresponding random network structure with $N \sim \mathcal{U}\{10 \cdots 100\}$ nodes (detailed in Supplementary Section 3.3.2). To generate the node activities $X(t) = [\mathbf{x}_1(t), \cdots, \mathbf{x}_N(t)]$, instead of using the rollout approach

$$\mathbf{x}_i(t_{n+1}) = \mathbf{x}_i(t_n) + \mathbf{f}(X(t_n); A, i)\Delta t,$$

from a random initial value $\mathrm{X}(t_0)$, which can cause node states to diverge during the rollout, we sample the state of each node i at each timestep t_n independently and identically distributed from a D-dimensional distribution

$$\mathbf{x}_i(t_n) \overset{\text{i.i.d.}}{\sim} p_\theta(\mathbf{x}),$$

with random parameters θ. We then calculate the formula's output $\mathbf{f}(X(t); A, i)$ for each node i at each time step t to replace $\dot{\mathbf{x}}_i(t)$. Considering that real dynamical systems typically have attractors (for example, fixed points or limit cycles) in their state spaces[34], the observed node states are usually restricted to a low-dimensional manifold rather than uniformly distributed in the entire state space. Therefore, we use a Gaussian mixture model with a discrete latent variable and an L-dimensional continuous latent variable ($L < D$) to generate the distribution $p_\theta(\mathbf{x})$ with multiple centroids and shapes of low-dimensional manifolds (detailed in Supplementary Section 3.3.3). In this way, we can sample nodes' states on low-dimensional (that is, L-dimensional) manifolds within D-dimensional phase space, which are more reflective of real dynamical systems. This method substantially enhances the NDformer's guidance ability on dynamical systems. In the discovery of FitzHugh–Nagumo dynamics, which have a one-dimensional limit cycle in its state space, the NDformer trained with Gaussian mixture model sped up the search by around 60 times compared with using a uniform distribution $p_\theta(\mathbf{x}) = \mathcal{U}[-10, 10]^D$ (Supplementary Fig. 10).

To pretrain the NDformer, we decompose each generated formula into input–label pairs by sequentially removing the final symbol from

the formula's prefix notation. However, we found that, because a formula of length L can be decomposed into L input–label pairs, longer formulas generate more samples than shorter ones, leading NDformer to overfit them. To mitigate this issue, we sample no more than N_{exp} input–label pairs from the decomposed result of each formula, which prevents overfitting and improves the NDformer's performance by 10.87% (Supplementary Fig. 7). The decomposed inputs, network structure and node activities are fed into the NDformer to generate the policy, and its cross-entropy loss with the labels is then computed for backpropagation. However, as the GNN module within the NDformer can process only one network structure at a time, the batch size is restricted to 1, thus limiting the amount of data for each backpropagation step and leading to unstable training. To address this issue, we use a buffer to cache the context embeddings and formula embeddings. Once the buffer has stored more than N_{buf} tokens, the saved embeddings are used to train the model for one step. This design makes the training more stable and improves the NDformer's performance by 50.63% (Supplementary Fig. 8).

Discover dynamics from empirical data

When discovering dynamical formulas from empirical data, we perform forward differencing on the node activities of the empirical data to estimate their $\dot{\mathbf{x}}_i \approx \frac{\mathbf{x}_i(t+\Delta t)-\mathbf{x}_i(t)}{\Delta t}$, where Δt is the length of the time step. Then, using the estimated $\dot{\mathbf{x}}_i$ as the dependent variable, we adopt ND^2 to discover the formula $\mathbf{f}=[f_1,\cdots,f_D]$, such that $\dot{\mathbf{x}}_i \approx \mathbf{f}(X;A,i)$, where $X=[\mathbf{x}_1, \mathbf{x}_2,\cdots,\mathbf{x}_N]$. Each f_d is a symbol sequence constructed from the network dynamical symbol set, which consists of the proposed network dynamical operators, common mathematical operators, undetermined parameters, mathematical constants and vectorized variables (Supplementary Section 2.1).

In empirical systems, unknown edge weights and heterogeneous parameters may exist in the target formulas. To enable ND^2 to automatically discover these potential parameters, we introduce two additional types of 'vectorized' parameters to the symbol set, beyond the scalar parameters $C\in\mathbb{R}$, including (1) the edge-level parameters $\mathbf{C}_e\in\mathbb{R}^E$ to fit edge weights and (2) the node-level parameters $\mathbf{C}_n\in\mathbb{R}^N$ to fit heterogeneous parameters. For example, consider the Kuramoto model

$$\dot{x}_i=\omega_i+\sum_{j=1}^{N}K_{ij}\sin(x_j-x_i), \tag{16}$$

where the natural frequencies ω_i and coupling strengths K_{ij} are unknown. Based merely on the observed node activities $x_i(t)$, our method can discover

$$\dot{\mathbf{x}}=\mathbf{C}_v+\rho(\mathbf{C}_e\times\sin(\phi_s(\mathbf{x})-\phi_t(\mathbf{x}))), \tag{17}$$

where parameters $\mathbf{C}_v$ and $\mathbf{C}_e$ can then fit the $x_i(t)$ data using the reward calculator's nonlinear optimizer to restore the values of the unknown heterogeneous parameters (ω_i) and edge weights (K_{ij}). Our approach does not require that each node have the same parameters or that each edge have the same weights. The only isomorphism requirement is that the parameters of each node and the weights of its edges affect node activity in the same way (that is, following the same formula). Furthermore, when only a subset of nodes meets this isomorphism requirement, we can also find their dynamics formulas by adding masks in NDformer and the reward calculator to make them focus on these nodes (Supplementary Section 6.7).

To evaluate the discovered network dynamics formula $\mathbf{f}$, we compare the node activities generated by the discovered formula with the ground truth using the RMSE and sMAPE metrics. Specifically, the node activities are generated through

$$\hat{\mathbf{x}}_i(t+\Delta t)=\hat{\mathbf{x}}_i(t)+\mathbf{f}(\hat{X}(t);A,i)\Delta t, \tag{18}$$

where $\hat{X}(t)=[\hat{\mathbf{x}}_1(t),\cdots,\hat{\mathbf{x}}_N(t)]$ represents the generated node activities at time t, Δt denotes the length of the time step and $\hat{X}(t_0)=X(t_0)$ represents the ground-truth initial states. In experiments of microbial community dynamics and gene regulation dynamics, due to the large Δt in the original data, we use $\frac{1}{10}$ of Δt to generate continuous trajectories (Figs. 3e and 4f). In the experiment of epidemic spreading dynamics, because the output of the discovered formula $\mathbf{f}$ has physical meaning (that is, daily new cases), we additionally compare the correlation between the output of $\mathbf{f}$ and the ground truth $\dot{\mathbf{x}}_i$ using R^2. Moreover, to compare existing and our corrected dynamics formulas in gene regulation and microbial communities, we use the BIC[37], which evaluates models by balancing goodness of fit with model complexity. The definitions of RMSE, sMAPE, R^2 and BIC are provided in Supplementary Section 5.1.3.

In addition to the expert select step that involves human intervention, we introduce constraints to guide the search process toward interpretable results. Specifically, in both the synthetic and the empirical experiments, we limit the formula length to a maximum of 30 and restrict the number of parameters to no more than 5. In empirical experiments, we additionally require the unknown weight $\mathbf{C}_e$ to appear only as a multiplicative factor within the aggregate operator (that is, $\rho(\mathbf{C}_e\times\cdots)$) so that $\mathbf{C}_e$ can be interpreted as edge weights. Under these restrictions, the obtained formulas exhibit capabilities of interpretability. Therefore, we directly select the most accurate formulas from the Pareto frontier (listed in Supplementary Tables 7 and 9).

Discover dynamics beyond pairwise models

Pairwise models have a general form of

$$\dot{x}_i=W(x_i)+\sum_{j=1}^{N}A_{ij}Q(x_i,x_j).$$

By varying the forms of W and Q, it can effectively describe a broad range of systems[3,7,25,31]. However, they fall short in capturing more complex dynamics. Examples include systems with nonlinear aggregation[51], heterogeneous node dynamics[52], hidden edge weights[53], unknown network strucutre[31,54], higher-order interaction[44,45,55,56] and nonadditive aggregation[57]. Our core innovation—the network dynamics operators—enables the discovery of not only classical pairwise models but also more complex, higher-order models

Systems with nonlinear aggregation. When the interaction term influences the node dynamics through a nonlinear function Φ, that is, $\dot{x}_i=W(x_i)+\Phi(\sum_{j=1}^{N}A_{ij}Q(x_i,x_j))$, the system can exhibit multistable states or chaos behavior[51]. As demonstrated in Figs. 3c and 5b, our method is capable of discovering such dynamic models.

Systems on heterogeneous network. A heterogeneous network consists of multiple classes of nodes, each exhibiting distinct dynamics. Hong and Strogatz studied a Kuramoto system with conformist nodes that synchronize with their neighbors and contrarian nodes that oppose them[52]. As shown in the experiment in Supplementary Section 6.7, by applying masks to the NDformer and reward calculator to make them focus on one node class at a time, our method can recover the heterogeneous dynamics of each node class.

Systems with hidden edge weights. In many real-world cases, edges may have hidden weights that reflect varying levels of information transmission[53]. We introduce a solution that incorporates optimizable vectorized parameters within the symbol set to fit these edge weights. In Supplementary Section 6.8, we further demonstrate the effectiveness of this approach by successfully recovering both the Kuramoto dynamics and the hidden heterogeneous edge weights in a Kuramoto system.

Systems with unknown network structure. When the network structure of a system is unknown, discovering system dynamics from node activities alone can become very challenging[31,54]. To address this, as detailed in Supplementary Section 6.6, we extend the hidden edge-weight solution by treating A_{ij} as unknown weights on a fully connected network, solving A_{ij} via linear regression to improve accuracy and using BIC to promote sparsity in A_{ij}. As demonstrated in Supplementary Section 6.6, our method successfully recovers both the dynamics formula and the underlying network structure using only the node activities of the Kuramoto system in the synthetic experiment.

Systems with higher-order interaction. Real-world complex systems usually contain higher-order interactions[44,45,56], where nodes interact via hyperedges that connect multiple nodes[55]. As discussed in Supplementary Section 1.3, our core innovations, network dynamics operators, can be generalized to describe higher-order interactions. Specifically, considering that ϕ_s and ϕ_t map node-level variables to the edge level by selecting the first (source) and second (target) nodes on each edge, it is natural to generalize them to $\phi_1^{(n)}, \phi_2^{(n)}, \cdots, \phi_{n+1}^{(n)}$ that map node-level variables to the n-hyperedge level by selecting the 1st, 2nd, … $(n+1)$st nodes on each n-hyperedge. Similarly, ρ that aggregates information on edges to nodes can be generalized to $\rho^{(n)}$ that aggregates information on n-hyperedges to nodes.

Systems with nonadditive aggregation. Certain systems can have nonadditive aggregation mechanisms[57], where neighbor influence is combined via nonsummative operations such as product or maximization. As discussed in Supplementary Section 1.1.4, we can extend the aggregation operator ρ to $\rho_{\max}$ that aggregates using the maximum function, thereby enabling the discovery of such models.

Data availability

Source data are provided with this paper. All datasets, including the generated pretraining dataset, simulated model system activities, collected gene regulatory[35] and ecological data[39], epidemic transmission and corresponding human mobility data from publicly available sources, together with the scripts used for data generation and collection, are available via GitHub at https://github.com/tsinghua-fib-lab/ND2 and via Zenodo at https://doi.org/10.5281/zenodo.16995963 (ref. 58).

Code availability

All code used for data generation, model training and analysis in this study is available via GitHub at https://github.com/tsinghua-fib-lab/ND2 and via Zenodo at https://doi.org/10.5281/zenodo.16995963 (ref. 58) with an MIT license.

References

1. Mitchell, M. *Complexity: A Guided Tour* (Oxford Univ. Press, 2009).
2. Liu, Y.-Y., Slotine, J.-J. & Barabási, A.-L. Controllability of complex networks. *Nature* **473**, 167–173 (2011).
3. Gao, J., Barzel, B. & Barabási, A-L. Universal resilience patterns in complex networks. *Nature* **530**, 307–312 (2016).
4. Meena, C. et al. Emergent stability in complex network dynamics. *Nat. Phys.* **19**, 1033–1042 (2023).
5. Barzel, B. & Barabási, A.-L. Universality in network dynamics. *Nat. Phys.* **9**, 673–681 (2013).
6. FitzHugh, R. Impulses and physiological states in theoretical models of nerve membrane. *Biophys. J.* **1**, 445–466 (1961).
7. Harush, U. & Barzel, B. Dynamic patterns of information flow in complex networks. *Nat. Commun.* **8**, 2181 (2017).
8. Jiménez-Martínez, A. A model of belief influence in large social networks. *Econ. Theory* **59**, 21–59 (2015).
9. Camps-Valls, G. et al. Discovering causal relations and equations from data. *Phys. Rep.* **1044**, 1–68 (2023).
10. Guimerà, R. et al. A Bayesian machine scientist to aid in the solution of challenging scientific problems. *Sci. Adv.* **6**, eaav6971 (2020).
11. Reinbold, P. A. K., Kageorge, L. M., Schatz, M. F. & Grigoriev, R. O. Robust learning from noisy, incomplete, high-dimensional experimental data via physically constrained symbolic regression. *Nat. Commun.* **12**, 3219 (2021).
12. Weng, B. et al. Simple descriptor derived from symbolic regression accelerating the discovery of new perovskite catalysts. *Nat. Commun.* **11**, 3513 (2020).
13. Bongard, J. & Lipson, H. Automated reverse engineering of nonlinear dynamical systems. *Proc. Natl Acad. Sci. USA* **104**, 9943–9948 (2007).
14. Schmidt, M. & Lipson, H. Distilling free-form natural laws from experimental data. *Science* **324**, 81–85 (2009).
15. Liu, S., Li, Q., Shen, X., Sun, J. & Yang, Z. Automated discovery of symbolic laws governing skill acquisition from naturally occurring data. *Nat. Comput. Sci.* **4**, 334–345 (2024).
16. Quade, M., Abel, M., Shafi, K., Niven, R. & Noack, B. R. Prediction of dynamical systems by symbolic regression. *Phys. Rev. E* **94**, 012214 (2016).
17. Petersen, B. K. et al. Deep symbolic regression: recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations* (ICLR, 2021).
18. Udrescu, S.-M. & Tegmark, M. AI Feynman: a physics-inspired method for symbolic regression. *Sci. Adv.* **6**, eaay2631 (2020).
19. Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A. & Parascandolo, G. Neural symbolic regression that scales. In *Proc. 38th International Conference on Machine Learning* (eds Meila, M. & Zhang, T.) 936–945 (PMLR, 2021).
20. Kamienny, P.-A., Lample, G., Lamprier, S. & Virgolin, M. Deep generative symbolic regression with Monte-Carlo-tree-search. In *Proc. 40th International Conference on Machine Learning* (eds Krause, A. et al.) 15655–15668 (PMLR, 2023).
21. Shojaee, P., Meidani, K., Farimani, A. B. & Reddy, C. Transformer-based planning for symbolic regression. In *37th Conference on Neural Information Processing Systems* 45907–45919 (NeurIPS, 2023).
22. Shojaee, P., Meidani, K., Gupta, S., Farimani, A. B. & Reddy, C. K. LLM-SR: scientific equation discovery via programming with large language models. In *International Conference on Learning Representations* (ICLR, 2025).
23. Sun, F., Liu, Y., Wang, J.-X. & Sun, H. Symbolic physics learner: discovering governing equations via Monte Carlo tree search. In *International Conference on Learning Representations* (ICLR, 2023).
24. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl Acad. Sci. USA* **113**, 3932–3937 (2016).
25. Gao, T. & Yan, G. Autonomous inference of complex network dynamics from incomplete and noisy data. *Nat. Comput. Sci.* **2**, 160–168 (2022).
26. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **20**, 61–80 (2009).
27. Vaswani, A. et al. Attention is all you need. In *31st Conference on Neural Information Processing Systems* (eds Guyon, I. et al.) 5998–6008 (Curran Associates, 2017).
28. Metropolis, N. & Ulam, S. The Monte Carlo method. *J. Am. Stat. Assoc.* **44**, 335–341 (1949).
29. Hecker, M., Lambeck, S., Toepfer, S., Van Someren, E. & Guthke, R. Gene regulatory network inference: data integration in dynamic models—a review. *Biosystems* **96**, 86–103 (2009).

30. Pastor-Satorras, R., Castellano, C., Van Mieghem, P. & Vespignani, A. Epidemic processes in complex networks. *Rev. Modern Phys.* **87**, 925 (2015).
31. Prasse, B. & Van Mieghem, P. Predicting network dynamics without requiring the knowledge of the interaction graph. *Proc. Natl Acad. Sci. USA* **119**, e2205517119 (2022).
32. Battaglia, P. W. et al. Relational inductive biases, deep learning, and graph networks. Preprint at https://arxiv.org/abs/1806.01261 (2018).
33. Cranmer, M. et al. Discovering symbolic models from deep learning with inductive biases. In *Proc. 34th International Conference on Neural Information Processing System* 17429–17442 (NeurIPS, 2020).
34. Durstewitz, D., Koppe, G. & Thurm, M. I. Reconstructing computational system dynamics from neural data with recurrent neural networks. *Nat. Rev. Neurosci.* **24**, 693–710 (2023).
35. Luan, Y. & Li, H. Clustering of time-course gene expression data using a mixed-effects model with B-splines. *Bioinformatics* **19**, 474–482 (2003).
36. Kuzmin, E. et al. Systematic analysis of complex genetic interactions. *Science* **360**, eaao1729 (2018).
37. Schwarz, G. Estimating the dimension of a model. *Ann. Stat.* **6**, 461–464 (1978).
38. Coyte, K. Z., Schluter, J. & Foster, K. R. The ecology of the microbiome: networks, competition, and stability. *Science* **350**, 663–666 (2015).
39. Hu, J., Amor, D. R., Barbier, M., Bunin, G. & Gore, J. Emergent phases of ecological diversity and dynamics mapped in microcosms. *Science* **378**, 85–89 (2022).
40. Zhang, J. et al. The impact of relaxing interventions on human contact patterns and SARS-CoV-2 transmission in China. *Sci. Adv.* **7**, eabe2584 (2021).
41. Gao, T.-T., Barzel, B. & Yan, G. Learning interpretable dynamics of stochastic complex systems from experimental data. *Nat. Commun.* **15**, 6029 (2024).
42. Shepherd, S. P. A review of system dynamics models applied in transportation. *Transportmetrica B* **2**, 83–105 (2014).
43. Somin, S., Altshuler, Y., Gordon, G., Pentland, A. & Shmueli, E. Network dynamics of a financial ecosystem. *Sci. Rep.* **10**, 4587 (2020).
44. Giusti, C., Ghrist, R. & Bassett, D. S. Two's company, three (or more) is a simplex: algebraic-topological tools for understanding higher-order structure in neural data. *J. Comput. Neurosci.* **41**, 1–14 (2016).
45. Momeni, B., Xie, L. & Shou, W. Lotka–Volterra pairwise modeling fails to capture diverse pairwise microbial interactions. *eLife* **6**, e25051 (2017).
46. Iacopini, I., Petri, G., Baronchelli, A. & Barrat, A. Group interactions modulate critical mass dynamics in social convention. *Commun. Phys.* **5**, 64 (2022).
47. Shojaee, P. et al. LLM-SRBench: a new benchmark for scientific equation discovery with large language models. In *42nd International Conference on Machine Learning* (ICML, 2025).
48. Delabays, R., De Pasquale, G., Dörfler, F. & Zhang, Y. Hypergraph reconstruction from dynamics. *Nat. Commun.* **16**, 2691 (2025).
49. Fajardo-Fontiveros, O. et al. Fundamental limits to learning closed-form mathematical models from data. *Nat. Commun.* **14**, 1043 (2023).
50. Fletcher, R. *Practical Methods of Optimization* (Wiley, 2000).
51. Morrison, K., Degeratu, A., Itskov, V. & Curto, C. Diversity of emergent dynamics in competitive threshold-linear networks. *SIAM J. Appl. Dyn. Syst.* **23**, 855–884 (2024).
52. Hong, H. & Strogatz, S. H. Kuramoto model of coupled oscillators with positive and negative coupling parameters: an example of conformist and contrarian oscillators. *Phys. Rev. Lett.* **106**, 054102 (2011).
53. Rodriguez, M. G., Leskovec, J., Balduzzi, D. & Schölkopf, B. Uncovering the structure and temporal dynamics of information propagation. *Netw. Sci.* **2**, 26–65 (2014).
54. Martin, T., Ball, B. & Newman, M. E. J. Structural inference for uncertain networks. *Phys. Rev. E* **93**, 012306 (2016).
55. Battiston, F. et al. Networks beyond pairwise interactions: structure and dynamics. *Phys. Rep.* **874**, 1–92 (2020).
56. Centola, D., Becker, J., Brackbill, D. & Baronchelli, A. Experimental evidence for tipping points in social convention. *Science* **360**, 1116–1119 (2018).
57. Yang, J., Zhou, L., Wang, B. & Zheng, Y. Max-consensus of multi-agent systems in random networks. *J. Franklin Inst.* **361**, 106712 (2024).
58. Yu, Z., Ding, J. & Li, Y. Discover network dynamics with neural symbolic regression. Version v1.0. *Zenodo* https://doi.org/10.5281/zenodo.16995963 (2025).

Acknowledgements

This research is supported by the National Key Research and Development Program of China (grant no. 2024YFC3307603) and the National Natural Science Foundation of China (grant no. 62476152) and is sponsored by Tsinghua-Toyota Joint Research Institute Inter-disciplinary Program. The funders had no role in study design, data collection and analysis, decision to publish or preparation of the paper.

Author contributions

Y.L. and J.D. launched this project, provided the research outline and guided the overall research direction. J.D. provided key guidance in method design and experiment implementation. Z.Y. designed the method and conducted experiments and result analysis. All authors jointly participated in the discussions on the methods and writing of the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s43588-025-00893-8.

Correspondence and requests for materials should be addressed to Jingtao Ding or Yong Li.

Peer review information *Nature Computational Science* thanks Yuanzhao Zhang and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available. Primary Handling Editor: Jie Pan, in collaboration with the *Nature Computational Science* team.