Physics-informed learning of governing equations from scarce data

Zhao Chen, Yang Liu, Hao Sun

Nature Communications

https://doi.org/10.1038/s41467-021-26434-1

Overview

- Complex dynamical systems modeling
- ***Unclear or partially unknown*** ODEs, PDEs
- Increasingly ***rich*** observational data
  - E.g. evolution of sea ice from observational data (e.g., satellite remote sensing images)
- More ***interpretable*** and more ***accurate***

Contributions

- (1) A ***"root-branch"*** network, constrained by unified underlying physics, that is capable of dealing with a small number of multi-datasets coming from ***different initial/boundary conditions***
- (2) A simple, yet effective, ***alternating direction training strategy*** for optimization of heterogeneous parameters, i.e., DNN trainable parameters and sparse PDE coefficients.

PDE Discovery from Spatiotemporal data

- Objective: Find the closed form of $\mathcal{F}[\cdot]$

$$\mathbf{u}_t + \mathcal{F}\left[\mathbf{u}, \mathbf{u}^2, \ldots, \nabla_{\mathbf{x}}\mathbf{u}, \nabla_{\mathbf{x}}^2\mathbf{u}, \nabla_{\mathbf{x}}\mathbf{u} \cdot \mathbf{u}, \ldots; \boldsymbol{\lambda}\right] = \mathbf{p}$$

- Approximation: Linear Combination

$$\bar{\boldsymbol{\phi}} = \{1, \mathbf{u}, \mathbf{u}^2, \ldots, \mathbf{u}_x, \mathbf{u}_y, \ldots, \mathbf{u}^3 \odot \mathbf{u}_{xy}, \ldots, \sin(\mathbf{u}), \ldots\},$$

$$\mathbf{u}_t = \boldsymbol{\phi}\boldsymbol{\Lambda}$$

- Parsimonious closed form: sparse coefficient matrix

Previous Works

- (Bongard and Lipson, 2007) and (Schmidt and Lipson, 2009)
  - Symbolic regression and genetic programming (***Pioneering work***)
- Sparse identification of nonlinear dynamics (SINDy) (Brunton et al., 2016)

$$\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi} + \eta\mathbf{Z},$$

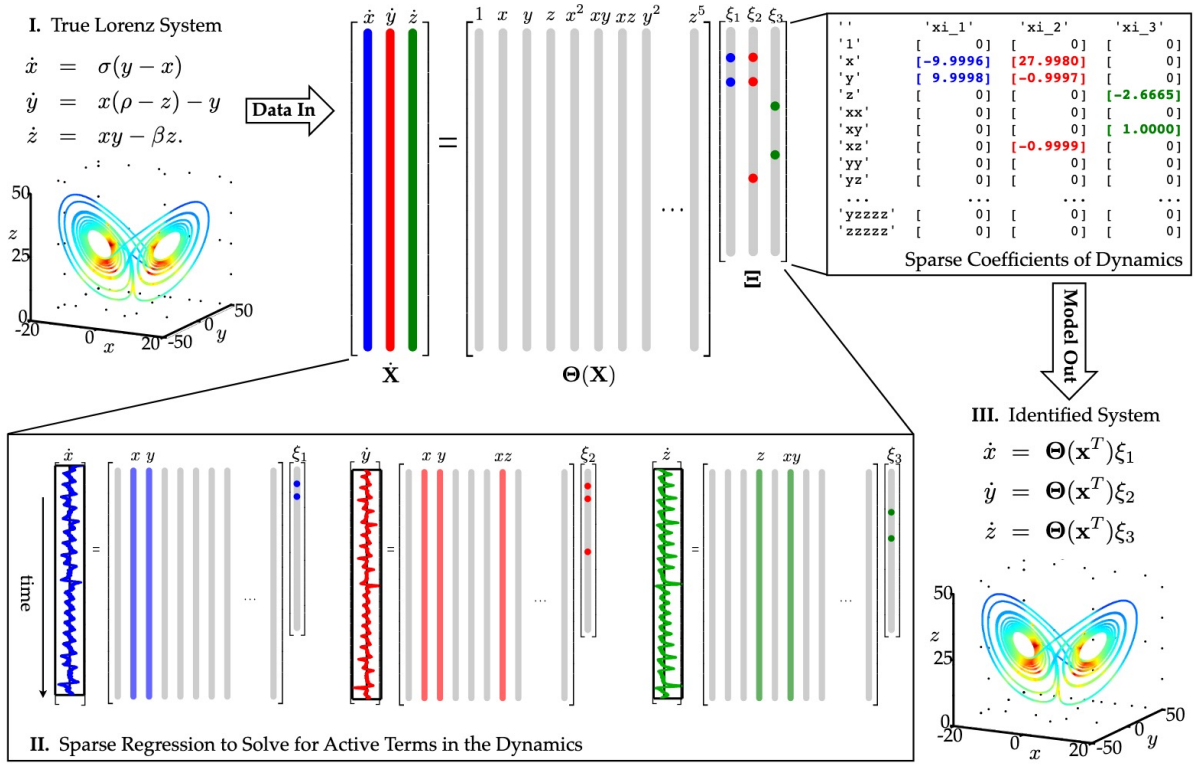
$$\mathbf{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & | & | & | & | & | & | & | \\ \mathbf{1} & \mathbf{X} & \mathbf{X}^{P_2} & \mathbf{X}^{P_3} & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \sin(2\mathbf{X}) & \cos(2\mathbf{X}) & \cdots \\ | & | & | & | & | & | & | & | & | \end{bmatrix}.$$

$$\boldsymbol{\xi} = \underset{\boldsymbol{\xi}'}{\operatorname{argmin}} \|\mathbf{\Theta}\boldsymbol{\xi}' - \mathbf{y}\|_2 + \lambda\|\boldsymbol{\xi}'\|_1.$$

Code 1: Sparse representation algorithm in Matlab.

```
%% compute Sparse regression: sequential least squares
Xi = Theta\dXdt;  % initial guess: Least-squares

% lambda is our sparsification knob.
for k=1:10
    smallinds = (abs(Xi)<lambda);   % find small coefficients
    Xi(smallinds)=0;                % and threshold
    for ind = 1:n                   % n is state dimension
        biginds = ~smallinds(:,ind);
        % Regress dynamics onto remaining terms to find sparse Xi
        Xi(biginds,ind) = Theta(:,biginds)\dXdt(:,ind);
    end
end
```



Previous Works

- Physics-informed neural networks (PINN)

Note: Explicit PDE form required

$$f := u_t + \mathcal{N}[u],$$

$$MSE = MSE_u + MSE_f,$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Automatic Differentiation

Burgers' equation:

$$\begin{aligned} &u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\ &u(0, x) = -\sin(\pi x), \\ &u(t, -1) = u(t, 1) = 0. \end{aligned} \tag{3}$$

Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx},$$

```
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u
```

Correspondingly, the *physics informed neural network* $f(t, x)$ takes the form

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

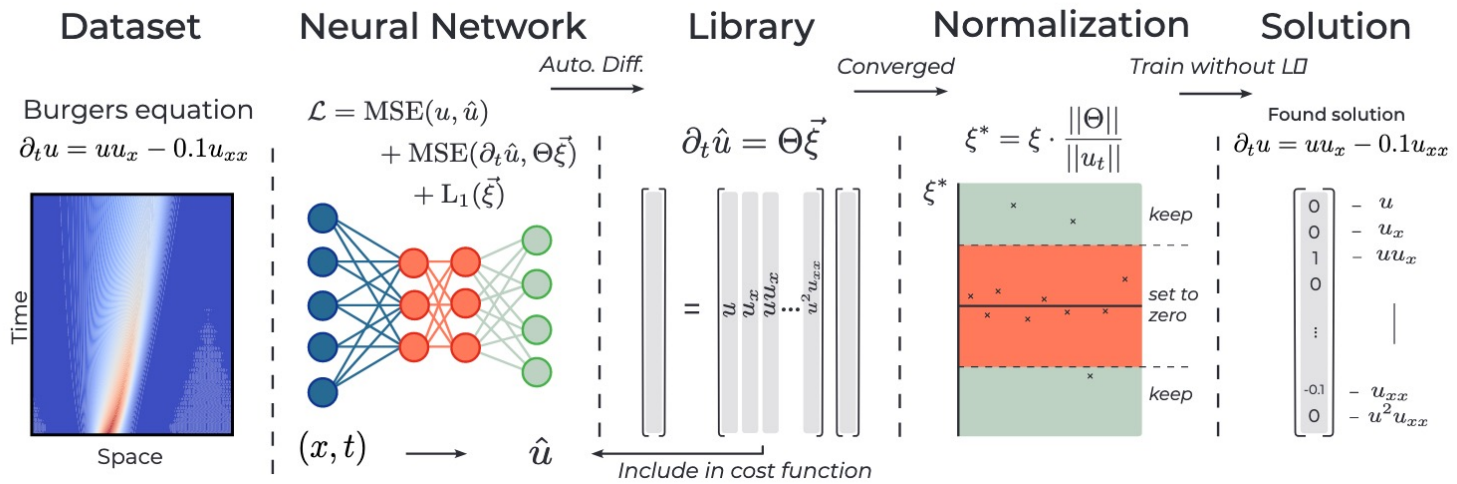
Previous Works

- PDE discovery
  - Data-driven discovery of PDEs in complex datasets (Berg and Nyström, 2019)

$$\mathbf{p}^* = \min_{\mathbf{p}} \frac{1}{2}||u(x,t) - \hat{u}(\mathbf{x},t;\mathbf{p})||^2 + \frac{\alpha_p}{2}||\mathbf{p}||^2,$$

$$\mathbf{q}^* = \min_{\mathbf{q}} \frac{1}{2}||\hat{u}_t - \hat{L}(\hat{u}, \partial\hat{u}, \ldots, \partial^m\hat{u};\mathbf{q})||^2 + \frac{\alpha_q}{2}||\mathbf{q}||_1^2.$$

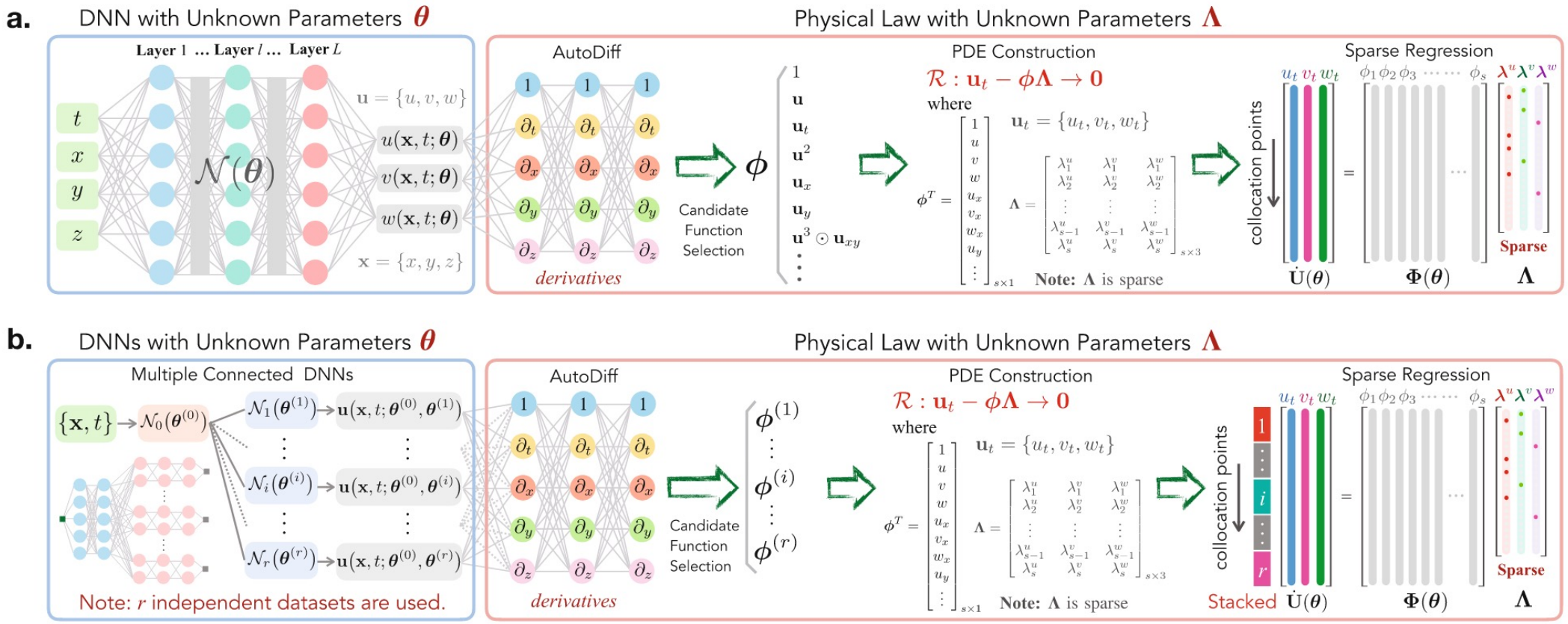
  - Deepmod: Deep learning for model discovery in noisy data (Both et al., 2020)



$$\partial_t u = \Theta\xi,$$

$$\Theta = \begin{bmatrix} 1 & u(\{x,t\}_0) & u_x(\{x,t\}_0) & \cdots & u^2u_{xx}(\{x,t\}_0) \\ 1 & u(\{x,t\}_1) & u_x(\{x,t\}_1) & \cdots & u^2u_{xx}(\{x,t\}_1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u(\{x,t\}_N) & u_x(\{x,t\}_N) & \cdots & u^2u_{xx}(\{x,t\}_N) \end{bmatrix}$$

Method



c.

Data Loss: $\mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) = \frac{1}{N_m}\|\mathbf{u}^\theta - \mathbf{u}^m\|_2^2$ (measurement)

$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_u, \mathcal{D}_c) = \mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) + \alpha\mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_c) + \beta\|\boldsymbol{\Lambda}\|_0$ (total loss; data loss; physics loss; regularization)

Residual Loss: $\mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_c) = \frac{1}{N_c}\|\dot{\mathbf{U}(\boldsymbol{\theta})} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Lambda}\|$ (collocation points)

Solution by **ADO**: $\hat{\boldsymbol{\Lambda}}_{k+1} := \arg\min_{\boldsymbol{\Lambda}} \left[\|\dot{\mathbf{U}}(\hat{\boldsymbol{\theta}}_k) - \boldsymbol{\Phi}(\hat{\boldsymbol{\theta}}_k)\boldsymbol{\Lambda}\|_2^2 + \beta\|\boldsymbol{\Lambda}\|_0\right]$ by STRidge

$\hat{\boldsymbol{\theta}}_{k+1} := \arg\min_{\boldsymbol{\theta}} \left[\mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) + \alpha\mathcal{L}_p(\boldsymbol{\theta}, \hat{\boldsymbol{\Lambda}}_{k+1}; \mathcal{D}_c)\right]$ by DNN training

Neural Networks

- Root Branch Structures
  - Root: shared networks
  - Branch: different I/BCs

Example:

```
layers_s = [2, 20, 20, 20, 20] # root NN
layers_i = [20, 30, 30, 30, 30, 1] # branch NNs
```

Optimization

- Total Loss(data loss + physics loss + reg term):

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_u, \mathcal{D}_c) = \mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) + \alpha \mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_c) + \beta \parallel \boldsymbol{\Lambda} \parallel_0$$

$$\mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) = \frac{1}{N_m} \left\| \mathbf{u}^{\theta} - \mathbf{u}^{m} \right\|_2^2$$

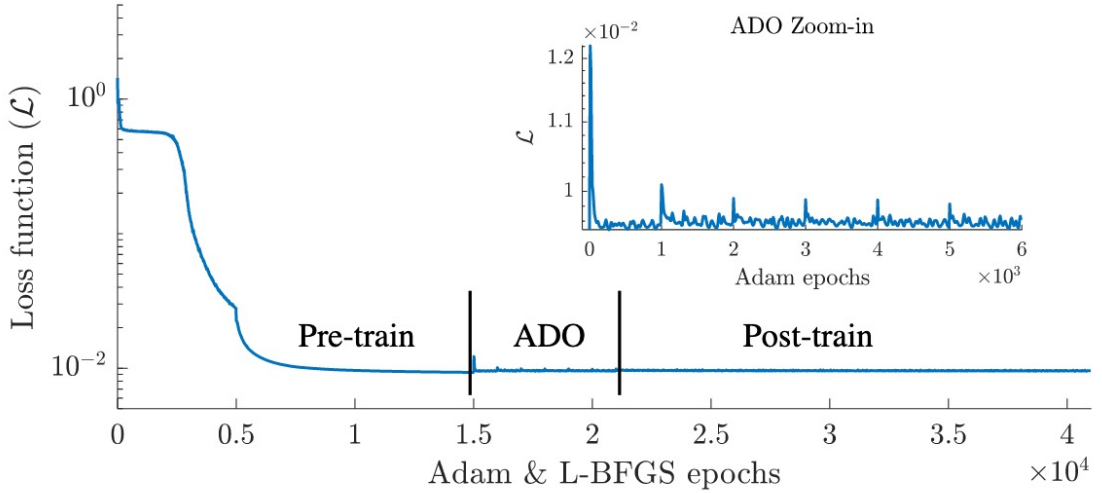
$$\mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_c) = \frac{1}{N_c} \left\| \dot{\mathbf{U}}(\boldsymbol{\theta}) - \boldsymbol{\Phi}(\boldsymbol{\theta}) \boldsymbol{\Lambda} \right\|_2^2$$

- Pre-training (Adam + L-BFGS)

$$\{\boldsymbol{\theta}^{\star}, \boldsymbol{\Lambda}^{\star}\} = \arg\min_{\{\boldsymbol{\theta}, \boldsymbol{\Lambda}\}} \{\mathcal{L}_d(\boldsymbol{\theta}; \mathscr{D}_u) + \alpha \mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathscr{D}_c) + \gamma \|\boldsymbol{\Lambda}\|_1\}$$

- Alternating Direction Optimization(ADO)
- Post-training (fine-tuning)

$$\{\boldsymbol{\theta}^{\star}, \boldsymbol{\Lambda}^{\star}\} = \arg\min_{\{\boldsymbol{\theta}, \boldsymbol{\Lambda}\}} \{\mathcal{L}_d(\boldsymbol{\theta}; \mathscr{D}_u) + \alpha \mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathscr{D}_c)\}$$



Alternating Direction Optimization

$$\Lambda^{\star}_{k+1} := \arg\min_{\Lambda}[\| \dot{\mathbf{U}}(\theta^{\star}_k) - \Phi(\theta^{\star}_k)\Lambda\|_2^2 + \beta\|\Lambda\|_0]$$

$$\theta^{\star}_{k+1} := \arg\min_{\theta}[\mathcal{L}_d(\theta; \mathscr{D}_u) + \alpha\mathcal{L}_p(\theta, \Lambda^{\star}_{k+1}; \mathscr{D}_c)]$$

Algorithm 1: STRidge($\Theta, \mathbf{U}_t, \lambda, tol, \text{iters}$)

$\hat{\xi} = arg\,min_{\xi}\|\Theta\xi - \mathbf{U}_t\|_2^2 + \lambda\|\xi\|_2^2$ # ridge regression
bigcoeffs $= \{j : |\hat{\xi}_j| \geq tol\}$ # select large coefficients
$\hat{\xi}[\sim \text{bigcoeffs}] = 0$ # apply hard threshold
$\hat{\xi}[\text{bigcoeffs}] = \text{STRidge}(\Theta[:, \text{bigcoeffs}], \mathbf{U}_t, tol, \text{iters} - 1)$
recursive call with fewer coefficients
return $\hat{\xi}$

Theorem 1. Let $\Theta^{\star} = \{\theta^{\star}, \Lambda^{\star}\}$ be a local minimizer of the total loss function $\mathcal{L}(\theta, \Lambda; \mathcal{D}_u, \mathcal{D}_c) : \mathbb{R}^{\eta} \mapsto \mathbb{R}$ and let $\mathcal{L}$ be strictly convex in a neighborhood $\mathfrak{N}(\Theta^{\star}, \delta)$, where η denotes the number of trainable parameters. We choose $\mathbf{0} < \epsilon \leq \delta$ so that $\mathcal{L}$ is strictly convex on $\mathfrak{N}(\Theta^{\star}, \epsilon)$. If $\mathbf{y} = \{\theta, \Lambda^{\star}\} \in \mathfrak{N}(\Theta^{\star}, \epsilon)$ and θ^{*} locally minimizes $\mathcal{L}(\theta, \Lambda^{\star}; \mathcal{D}_u, \mathcal{D}_c)$, then θ^{*} is the unique global minimizer. This is also applicable to Λ^{*}. For any admissible initial solution $\Theta_0 \in \mathfrak{N}(\Theta^{\star}, \epsilon)$, the corresponding ADO iteration sequence converges to $\Theta^{\star} q$-linearly in theory. The actual convergence rate depends on the error propagation in each ADO iteration.

Suppose that the sequence (x_k) converges to the number L. The sequence is said to *converge Q-linearly to* L if there exists a number $\mu \in (0, 1)$ such that

$$\lim_{k \to \infty} \frac{|x_{k+1} - L|}{|x_k - L|} = \mu.$$

The number μ is called the *rate of convergence*.[3]

Alternating Direction Optimization

Algorithm 1 The proposed ADO for network training: $[\boldsymbol{\theta}_{\text{best}}, \boldsymbol{\Lambda}_{\text{best}}] = \texttt{ADO}\left(\mathcal{D}_u, \mathcal{D}_c, \alpha, \gamma, \Delta\delta, n_{\max}, n_{\text{str}}\right)$

Input: Measurement data $\mathcal{D}_u$, collocation points $\mathcal{D}_c = \{\mathbf{x}_i, t_i\}_{i=1,2,\ldots,N_c}$, relative weighting of loss functions α and γ, threshold tolerance increment $\Delta\delta$ for STRidge, maximum number of ADO iterations $n_{\max}$, and maximum number of STRidge cycles n_{str}.
we take a 2D system in a 2D domain as an example: $\mathbf{u} = \{u, v\}$ and $\mathbf{x} = \{x, y\}$
Split measurement data $\mathcal{D}_u$ into training-validation sets ($n_{\text{tr}}/n_{\text{va}} = 80/20$): $\mathcal{D}_u^{\text{tr}} \in \mathbb{R}^{n_{\text{tr}} \times 2}$ and $\mathcal{D}_u^{\text{va}} \in \mathbb{R}^{n_{\text{va}} \times 2}$. # $N_m = n_{\text{tr}} + n_{\text{va}}$
Split collocation points $\mathcal{D}_c$ into training-validation sets ($m_{\text{tr}}/m_{\text{va}} = 80/20$): $\mathcal{D}_c^{\text{tr}} \in \mathbb{R}^{m_{\text{tr}} \times 3}$ and $\mathcal{D}_c^{\text{va}} \in \mathbb{R}^{m_{\text{va}} \times 3}$. # $N_c = m_{\text{tr}} + m_{\text{va}}$
Initialize the *Tensor Graph* for the entire network.
Pre-train the network via combined Adam and L-BFGS with $\{\mathcal{D}_u^{\text{tr}}, \mathcal{D}_c^{\text{tr}}\}$, and validate the trained model with $\{\mathcal{D}_u^{\text{va}}, \mathcal{D}_c^{\text{va}}\}$, namely,

$$\{\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0\} = \arg\min_{\{\boldsymbol{\theta}, \boldsymbol{\Lambda}\}} \left\{\mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) + \alpha \mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathcal{D}_c) + \gamma \|\boldsymbol{\Lambda}\|_1\right\}. \quad \text{\# pre-train the network; } \hat{\boldsymbol{\Lambda}}_0 = \{\hat{\boldsymbol{\lambda}}_0^u, \hat{\boldsymbol{\lambda}}_0^v\}$$

for $k = 1, 2, \ldots, n_{\max}$ **do**
  Assemble the system states over the collocation points $\mathcal{D}_c^{\text{tr}}$ and $\mathcal{D}_c^{\text{va}}$:

$$\dot{\mathbf{U}}_u^{\text{tr}} = \bigcup_{i=1}^{N_c^{\text{tr}}} u_t\left(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{tr}}, t_i^{\text{tr}}\right) \quad \text{and} \quad \dot{\mathbf{U}}_u^{\text{va}} = \bigcup_{i=1}^{N_c^{\text{tr}}} u_t\left(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{va}}, t_i^{\text{va}}\right)$$
$$\dot{\mathbf{U}}_v^{\text{tr}} = \bigcup_{i=1}^{N_c^{\text{va}}} v_t\left(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{tr}}, t_i^{\text{tr}}\right) \quad \text{and} \quad \dot{\mathbf{U}}_v^{\text{va}} = \bigcup_{i=1}^{N_c^{\text{va}}} v_t\left(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{va}}, t_i^{\text{va}}\right).$$

  Assemble the candidate library matrices over the collocation points $\mathcal{D}_c$, $\mathcal{D}_c^{\text{tr}}$ and $\mathcal{D}_c^{\text{va}}$:

$$\tilde{\boldsymbol{\Phi}} = \bigcup_{i=1}^{N_c} \boldsymbol{\phi}\left(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i, t_i\right), \quad \tilde{\boldsymbol{\Phi}}^{\text{tr}} = \bigcup_{i=1}^{N_c^{\text{tr}}} \boldsymbol{\phi}\left(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{tr}}, t_i^{\text{tr}}\right) \quad \text{and} \quad \tilde{\boldsymbol{\Phi}}^{\text{va}} = \bigcup_{i=1}^{N_c^{\text{va}}} \boldsymbol{\phi}\left(\hat{\boldsymbol{\theta}}_{k-1}; \mathbf{x}_i^{\text{va}}, t_i^{\text{va}}\right).$$

  Normalize candidate library matrices $\tilde{\boldsymbol{\Phi}}$, $\tilde{\boldsymbol{\Phi}}^{\text{tr}}$ and $\tilde{\boldsymbol{\Phi}}^{\text{va}}$ column-wisely ($j = 1, \ldots, s$) to improve matrix condition:

$$\boldsymbol{\Phi}_{:,j} = \tilde{\boldsymbol{\Phi}}_{:,j} \Big/ \left\|\tilde{\boldsymbol{\Phi}}_{:,j}\right\|_2, \quad \boldsymbol{\Phi}_{:,j}^{\text{tr}} = \tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}} \Big/ \left\|\tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}}\right\|_2 \quad \text{and} \quad \boldsymbol{\Phi}_{:,j}^{\text{va}} = \tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}} \Big/ \left\|\tilde{\boldsymbol{\Phi}}_{:,j}^{\text{tr}}\right\|_2.$$

  Determine ℓ_0 regularization parameters $\beta^u = \kappa \mathcal{L}_p^u\left(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\lambda}}_0^u; \mathcal{D}_c^{va}\right)$ and $\beta^v = \kappa \mathcal{L}_p^v\left(\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Lambda}}_0^v; \mathcal{D}_c^{va}\right)$. # κ can be determined via a Pareto front analysis, e.g., $\kappa = 1$.
  Initialize the error indices:

$$\hat{\epsilon}^u = \mathcal{L}_p^u\left(\hat{\boldsymbol{\theta}}_{k-1}, \hat{\boldsymbol{\lambda}}_{k-1}^u; \mathcal{D}_c^{va}\right) + \beta^u \left\|\hat{\boldsymbol{\lambda}}_{k-1}^u\right\|_0 \text{ and } \hat{\epsilon}^v = \mathcal{L}_p^v\left(\hat{\boldsymbol{\theta}}_{k-1}, \hat{\boldsymbol{\lambda}}_{k-1}^v; \mathcal{D}_c^{va}\right) + \beta^v \left\|\hat{\boldsymbol{\lambda}}_{k-1}^v\right\|_0.$$

  Set the initial threshold tolerance $\delta_1 = \Delta\delta$.
  for $iter = 1, 2, \ldots, n_{\text{str}}$ **do**
    Run STRidge as shown in Algorithm 2 to determine:

$$\tilde{\boldsymbol{\lambda}}^u = \texttt{STRidge}\left(\dot{\mathbf{U}}_u^{\text{tr}}, \boldsymbol{\Phi}^{\text{tr}}, \delta_{iter}\right) \quad \text{and} \quad \tilde{\boldsymbol{\lambda}}^v = \texttt{STRidge}\left(\dot{\mathbf{U}}_v^{\text{tr}}, \boldsymbol{\Phi}^{\text{tr}}, \delta_{iter}\right).$$

    Update the error indices:

$$\epsilon^u = \mathcal{L}_p^u\left(\hat{\boldsymbol{\theta}}_{k-1}, \tilde{\boldsymbol{\lambda}}^u; \mathcal{D}_c^{va}\right) + \beta^u \left\|\tilde{\boldsymbol{\lambda}}^u\right\|_0 \text{ and } \epsilon^v = \mathcal{L}_p^v\left(\hat{\boldsymbol{\theta}}_{k-1}, \tilde{\boldsymbol{\lambda}}^v; \mathcal{D}_c^{va}\right) + \beta^v \left\|\tilde{\boldsymbol{\lambda}}^v\right\|_0.$$

    if $\epsilon^u \leqslant \hat{\epsilon}^u$ or $\epsilon^v \leqslant \hat{\epsilon}^v$ (run in parallel) **then**
      Increase threshold tolerance with increment: $\delta_{iter+1} = \delta_{iter} + \Delta\delta$.
    else
      Decrease threshold tolerance increment $\Delta\delta = \Delta\delta/1.618$.
      Update threshold tolerance with the new increment $\delta_{iter+1} = \max\{\delta_{iter} - 2\Delta\delta, 0\} + \Delta\delta$.
    end if
  end for
  Return and re-scale the current best solution from STRidge cycles: $\hat{\boldsymbol{\Lambda}}_k = \left\{\tilde{\boldsymbol{\lambda}}^u, \tilde{\boldsymbol{\lambda}}^v\right\}$. # re-scaling due to normalization of $\boldsymbol{\Phi}$
  Train the DNN via combined Adam and L-BFGS with $\{\mathcal{D}_u^{\text{tr}}, \mathcal{D}_c^{\text{tr}}\}$, and validate the trained model with $\{\mathcal{D}_u^{\text{va}}, \mathcal{D}_c^{\text{va}}\}$, namely,

$$\hat{\boldsymbol{\theta}}_k = \arg\min_{\boldsymbol{\theta}} \left\{\mathcal{L}_d(\boldsymbol{\theta}; \mathcal{D}_u) + \alpha \mathcal{L}_p(\boldsymbol{\theta}, \hat{\boldsymbol{\Lambda}}_k; \mathcal{D}_c)\right\}. \quad \text{\# train DNN given } \hat{\boldsymbol{\Lambda}}_k \text{ as known}$$

end for
Output: the best solution $\boldsymbol{\theta}_{\text{best}} = \hat{\boldsymbol{\theta}}_{n_{\max}}$ and $\boldsymbol{\Lambda}_{\text{best}} = \hat{\boldsymbol{\Lambda}}_{n_{\max}}$

Algorithm 2 Sequential threshold ridge regression (STRidge): $\hat{\boldsymbol{\lambda}} = \texttt{STRidge}\left(\dot{\mathbf{U}}, \boldsymbol{\Phi}, \delta\right)$

Input: Time derivative vector $\dot{\mathbf{U}}$, candidate function library matrix $\boldsymbol{\Phi}$, and threshold tolerance δ.
Inherit coefficients $\hat{\boldsymbol{\lambda}}$ from the DNN pre-training or the previous update.
repeat
  Determine indices of coefficients in $\hat{\boldsymbol{\lambda}}$ falling below or above the sparsity threshold δ:

$$\mathcal{I} = \{i \in \mathcal{I} : |\hat{\lambda}_i| < \delta\} \text{ and } \mathcal{J} = \{j \in \mathcal{J} : |\hat{\lambda}_j| \geqslant \delta\}.$$

  Enforce sparsity to small values by setting them to zero: $\hat{\boldsymbol{\lambda}}_{\mathcal{I}} = \mathbf{0}$.
  Update remaining non-zero values with ridge regression:

$$\hat{\boldsymbol{\lambda}}_{\mathcal{J}} = \arg\min_{\boldsymbol{\lambda}_{\mathcal{J}}} \left\{\left\|\boldsymbol{\Phi}_{\mathcal{J}} \boldsymbol{\lambda}_{\mathcal{J}} - \dot{\mathbf{U}}\right\|_2^2 + 1 \times 10^{-5} \left\|\boldsymbol{\lambda}_{\mathcal{J}}\right\|_2^2\right\}. \quad \text{\# the parameter } 1 \times 10^{-5} \text{ is small and tunable}$$

until maximum number of iterations reached.
Output: The best solution $\hat{\boldsymbol{\lambda}} = \hat{\boldsymbol{\lambda}}_{\mathcal{I}} \cup \hat{\boldsymbol{\lambda}}_{\mathcal{J}}$

Coefficients Error

Table 1 Summary of the PINN-SR discovery results in the context of accuracy for a range of canonical models.

PDE name	Err. (N-0%)	Err. (N-1%)	Err. (N-10%)	Description of data discretization
Burgers'	0.01 ± 0.01%	0.19 ± 0.11%	0.88 ± 0.03%	$x \in [-8, 8]_{\tilde{n}=256}$, $t \in [0, 10]_{\tilde{n}=101}$, sub. 3.19%
KS	0.07 ± 0.01%	0.61 ± 0.04%	0.94 ± 0.05%	$x \in [0, 100]_{\tilde{n}=1024}$, $t \in [0, 100]_{\tilde{n}=251}$, sub. 12.6%%
Schrödinger	0.09 ± 0.04%	0.65 ± 0.29%	0.08 ± 0.03%	$x \in [-4.5, 4.5]_{\tilde{n}=512}$, $t \in [0, \pi]_{\tilde{n}=501}$, sub. 37.5%
NS	0.66 ± 0.72%	0.86 ± 0.63%	1.22 ± 0.69%	$x \in [0, 9]_{\tilde{n}=449}$, $y \in [-2, 2]_{\tilde{n}=199}$, $t \in [0, 30]_{\tilde{n}=151}$, sub. 0.22%
λ-ω RD	0.07 ± 0.08%	0.25 ± 0.30%	1.84 ± 1.48%	$x, y \in [-10, 10]_{\tilde{n}=256}$, $t \in [0, 10]_{\tilde{n}=201}$, sub. 0.29%

The error is defined as the average relative error of the identified non-zero coefficients w.r.t. the ground truth. The percentage values in the parentheses denote the noise levels (e.g., noise free 0%, 1% and 10%) and the subscript $\tilde{n}$ represents the number of discretization. Our method is also compared with SINDy (the PDE-FIND approach presented in ref. [6]) as illustrated in Supplementary Table 1. It is noted that much less measurement data polluted with a higher level of noise are used in our discovery. Gaussian white noise is added to the synthetic response with the noise level defined as the root-mean-square ratio between the noise and the exact solution.

Compared with SINDy

PDE name	Method	Error (noise 0%)	Error (noise 1%)	Error (noise 10%)	# of Measurement points
Burgers'	PINN-SR	0.01±0.01%	0.19±0.11%	0.88 ± 0.03%	∼1k
	PDE-FIND	Fail	Fail	Fail	∼1k
		0.15±0.06%	0.80±0.60%	Fail	∼26k
KS	PINN-SR	0.07±0.01%	0.61±0.04%	0.94 ± 0.05%	∼32k
	PDE-FIND	35.75±16.30%	Fail	Fail	∼32k
		1.30±1.30%	52.00±1.40%	Fail	∼257k
Schrödinger	PINN-SR	0.09±0.04%	0.65±0.29%	0.08±0.03%	∼96k
	PDE-FIND	Fail	Fail	Fail	∼96K
		0.05±0.01%	3.00±1.00%	Fail	∼257k
NS	PINN-SR	0.66±0.72%	0.86±0.63%	1.22±0.69%	∼30k
	PDE-FIND	Fail	Fail	Fail	∼30K
		1.00±0.20%	7.00±6.00%	Fail	∼300k
λ-ω RD	PINN-SR	0.07±0.08%	0.25±0.30%	1.84 ± 1.48%	∼37.5k
	PDE-FIND	Fail	Fail	Fail	∼37.5k
		0.02±0.02%	Fail	Fail	∼150k

Burgers' Equation

$$u_t = -uu_x + \nu u_{xx}$$

Measured Grid: 256 x 101
Training(80%) + Validation(20%): 10 x 101
Collocation Points: $5x10^4$
Polynomial Terms: (u, u^2, u^3)
Derivatives: (u_x, u_{xx}, u_{xxx})
Zero Initiated Coefficient Matrix
Networks: 8 hidden layers of width 20

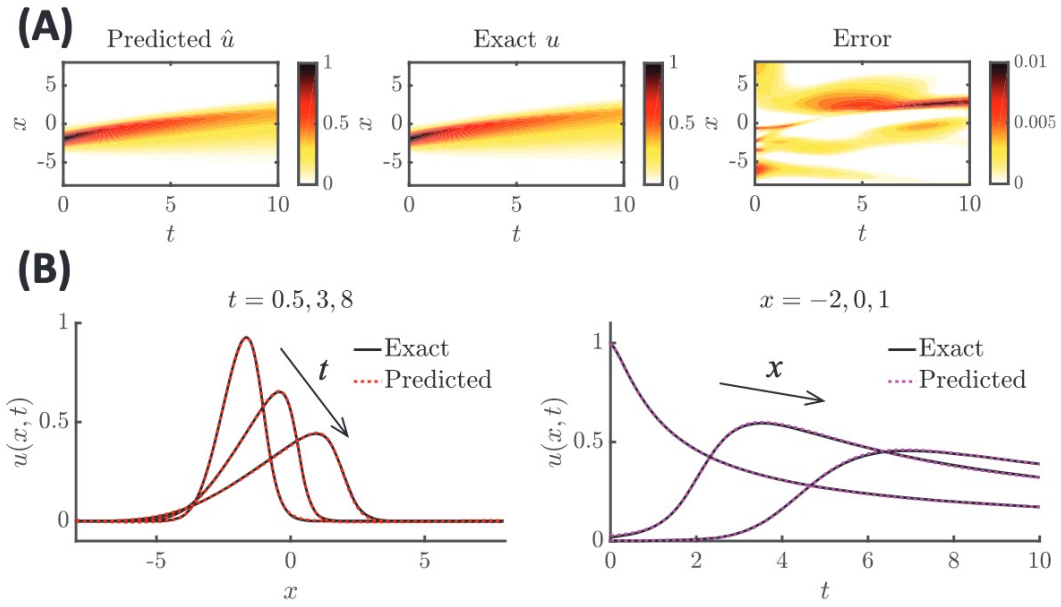
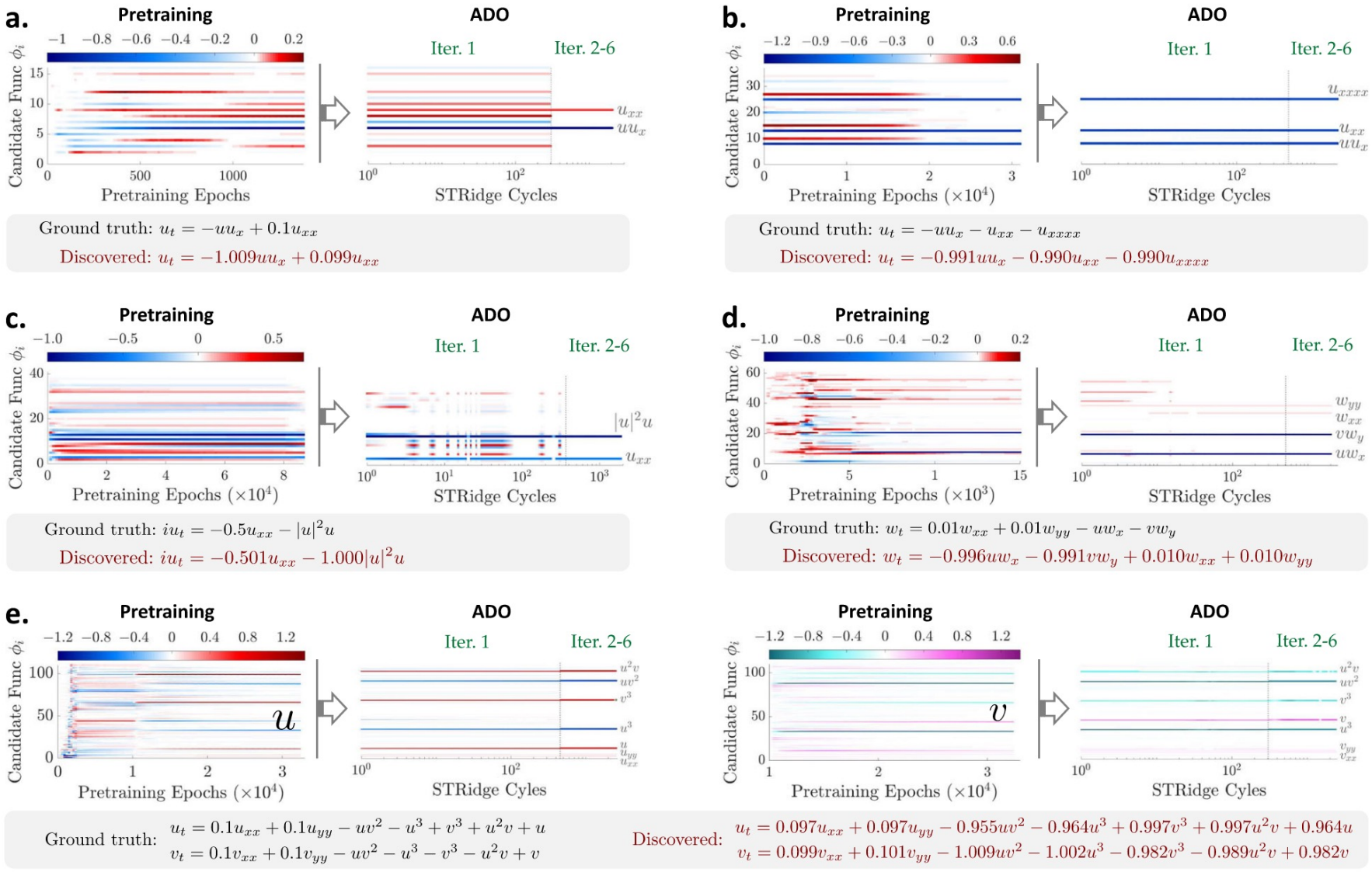


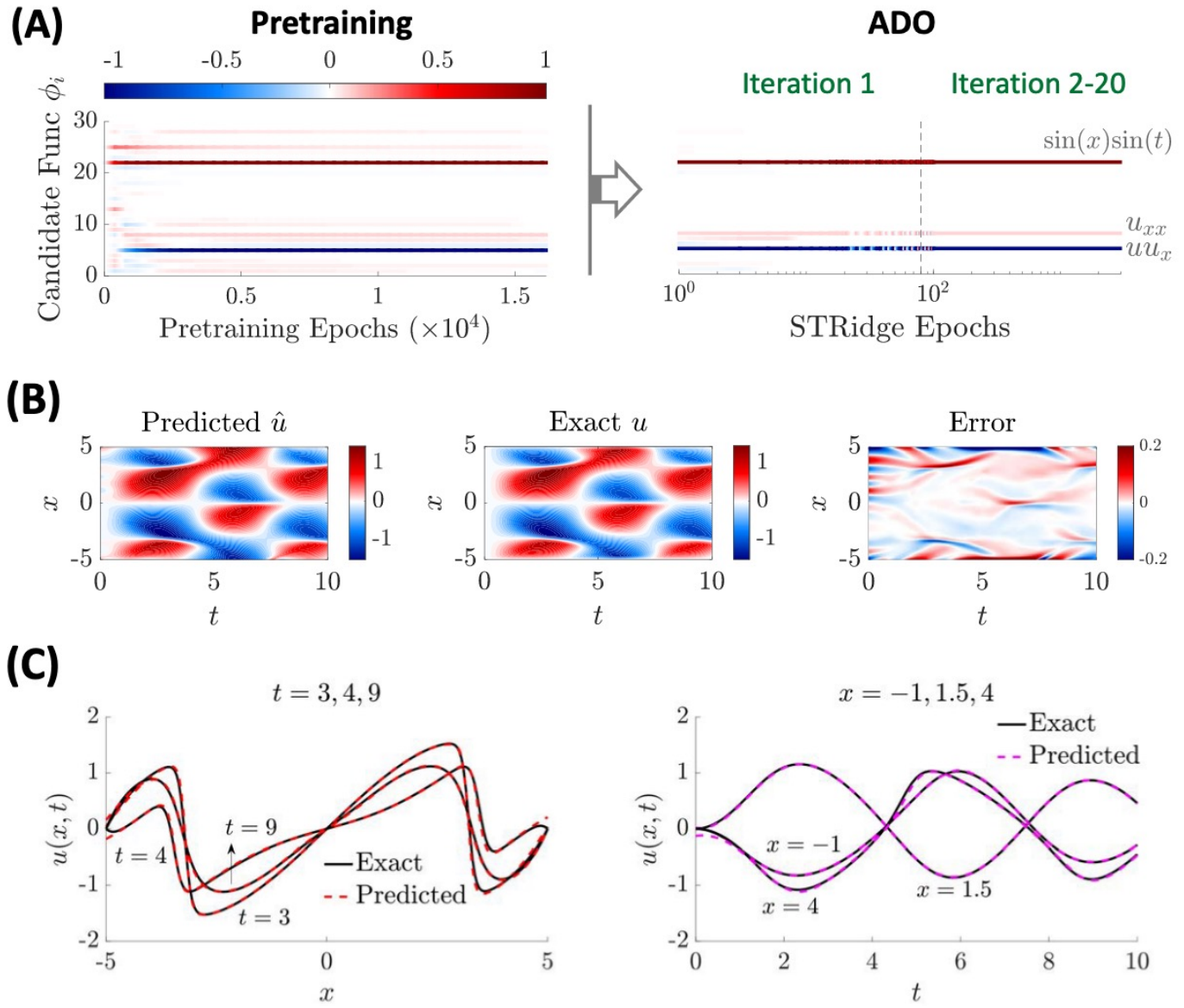
Figure 1: Discovery of Burgers' equation for data with 10% noise: (A) The predicted response in comparison with the exact solution with the prediction error. (B) Comparison of spatial and temporal snapshots between the predicted and the exact solutions. The relative full-field ℓ_2 error of the prediction is 1.32%.

Discovery of selected benchmark PDEs for sparsely sampled measurement data with 10% noise



Discovery of Burgers' equation and source term for measurement data with 10% noise

$\{\sin(t), \sin(x), \cos(t), \cos(x)\}$



Ground truth: $u_t + uu_x - 0.1u_{xx} = \sin(x)\sin(t)$

Discovered: $u_t + 1.002uu_x - 0.088u_{xx} = 0.995\sin(x)\sin(t)$

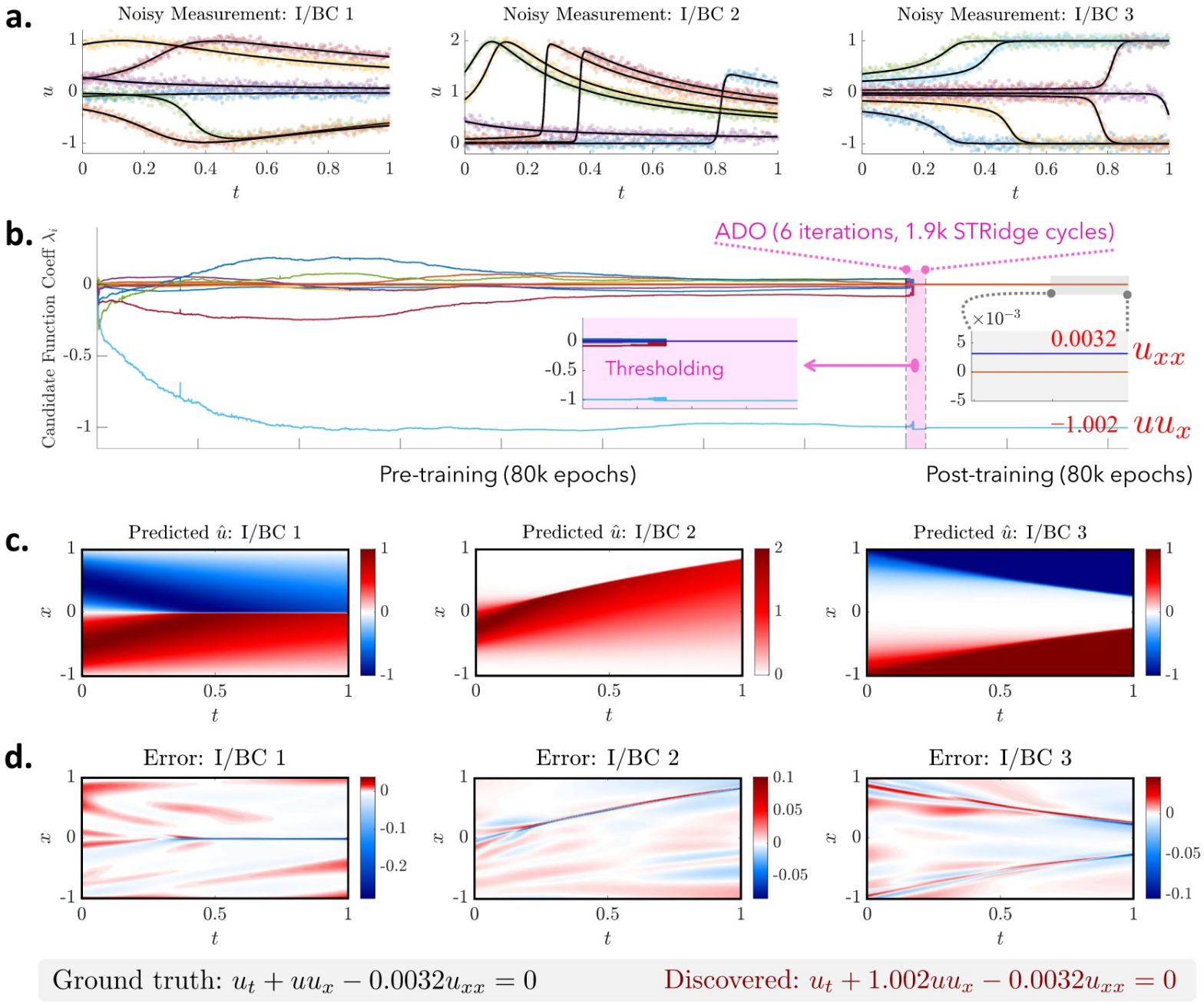
Discovered Burgers' equation with small viscosity based on datasets sampled under three I/BCs with 10% noise.

Networks:
20 x 20 x 20 x 20
30 x 30 x 30 x 30

$$\Omega \times [0, T] = [-1, 1]_{d=200} \times [0, 1]_{d=1000}.$$

30 x 500 (7.5%)
4.5 x 10^4

Small viscosity -> Sharp gradient -> Challenge DNN approximation



Discovered Fitzhugh–Nagumo equations based on data sampled under three initial conditions (ICs) with 10% noise.

Networks:
60 x 60
60 x 60 x 60

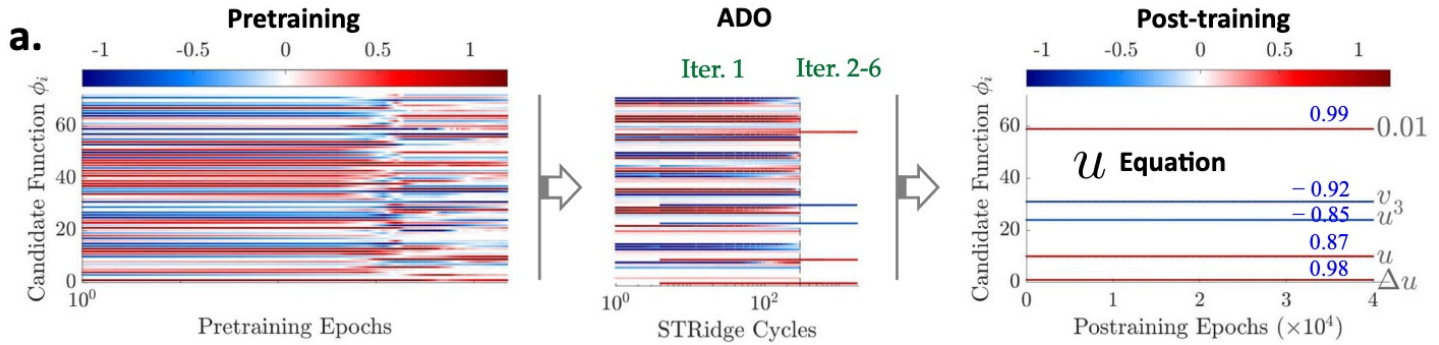
$$[0, 150] \times [0, 150]$$

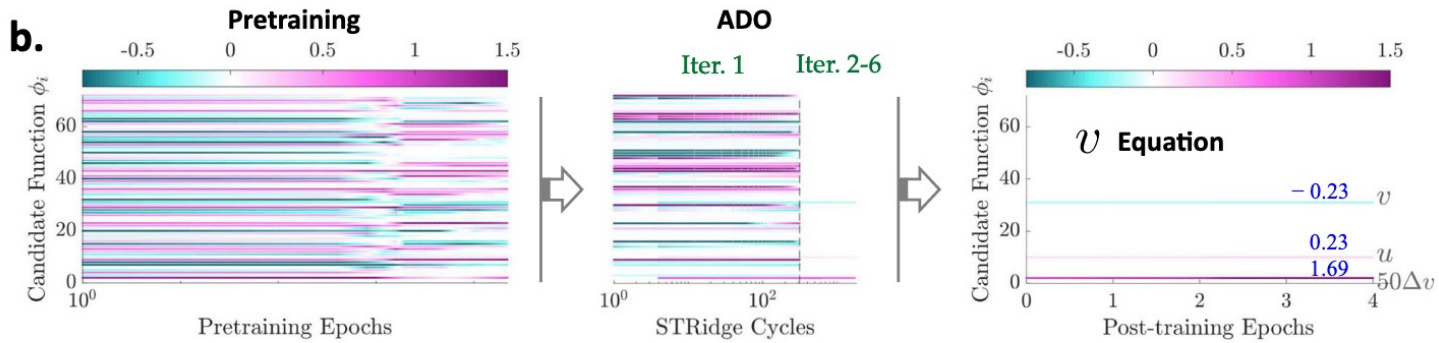
$$[0, T] = [0, 36]$$

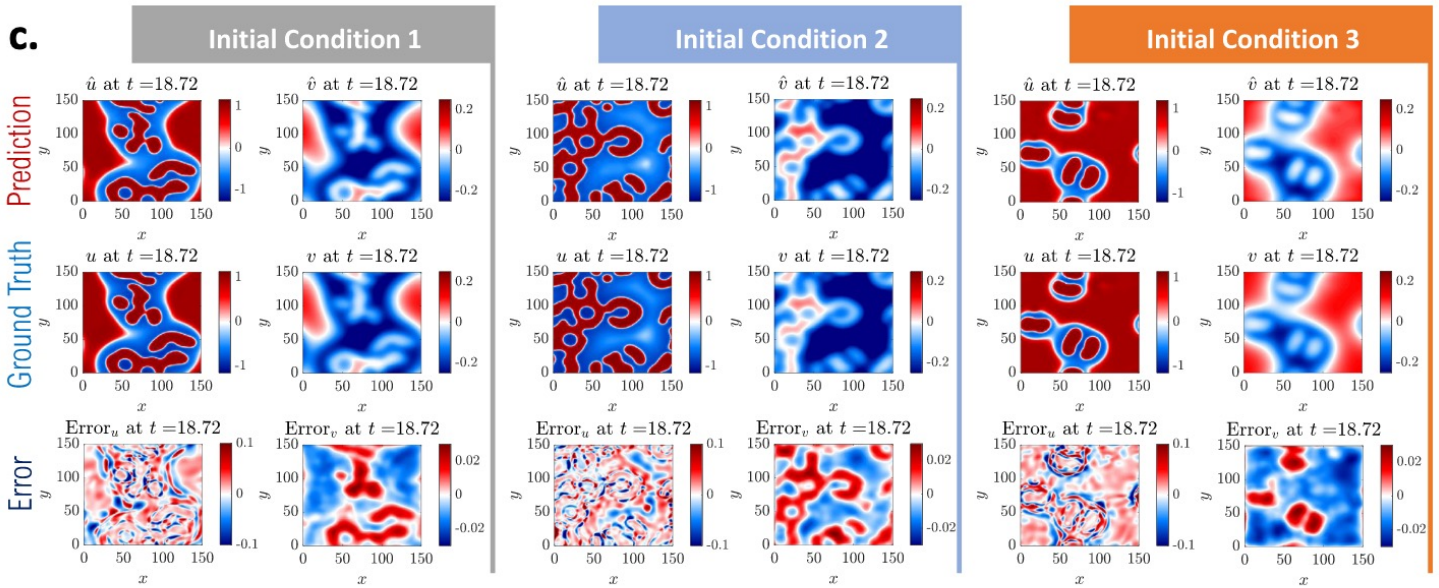
31 x 31
5 x 10^4

$$u_t = \gamma_u \Delta u + u - u^3 - v + \alpha$$

$$v_t = \gamma_v \Delta v + \beta(u - v).$$







Ground truth: $u_t = \Delta u + u - u^3 - v + 0.01$
$v_t = 100\Delta v + 0.25u - 0.25v$

Discovered: $u_t = 0.975\Delta u + 0.871u - 0.847u^3 - 0.924v + 0.010$
$v_t = 84.339\Delta v + 0.225u - 0.229v$

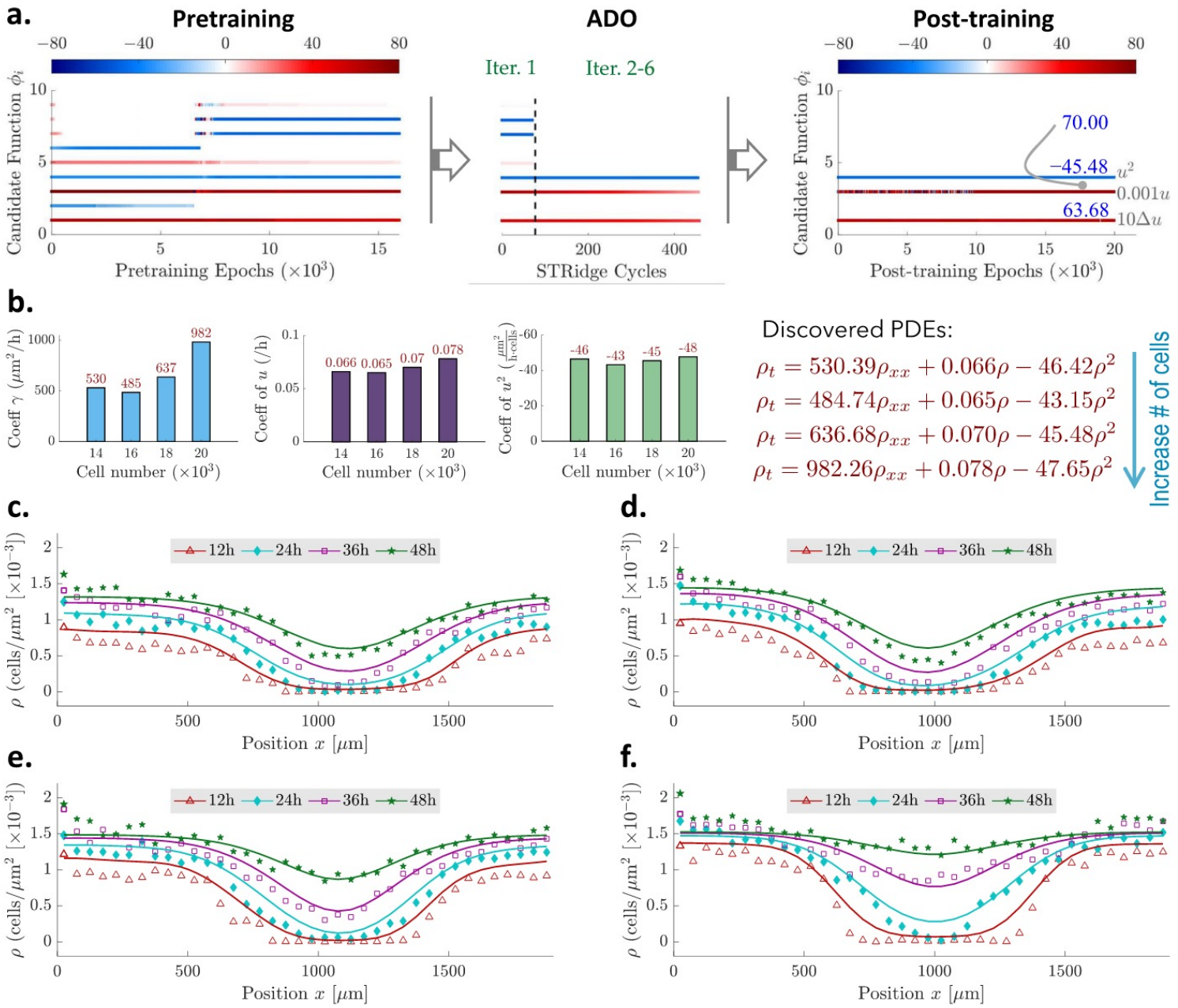
Discovery result for cell migration and proliferation.

14,000, 16,000, 18,000 to 20,000.

$$\rho_t = \gamma \rho_{xx} + \mathcal{F}(\rho),$$

$$\{1, \rho, \rho^2, \rho^3, \rho_x, \rho\rho_x, \rho^2\rho_x, \rho^3\rho_x\}$$

Fisher-Kolmogorov



Thanks!