Diff-SINDy: A Differentiable Framework for Discovering Partial Differential Equations from Data

Eddie Aljamal, Thomas M. Baer, Devon Loomis

Department of Physics, University of Michigan

Motivation

Discovering the governing equations of complex partial differential equations (PDEs) from data remains a challenging problem.

Traditional approaches such as SINDy [1] and PINN-SR [2] rely on fixed libraries of candidate functions, limiting flexibility.

Evo-SINDy [3] relaxes this constraint by introducing the derivative as a unary operator but remains non-differentiable in training and depends on discrete derivative estimates that are sensitive to noise.

Feature	SINDy	PINN-SR	Evo-SINDy	**Diff-SINDy**
No Functional Library	✗	✗	✓	✓
Differentiability	✗	✓	✗	✓
AD Derivatives	✗	✓	✗	✓
Sparsity	✓	✓	✓	✓
No Pre-training	✓	✗	✗	✓

Diff-SINDy is a differentiable framework for data-driven PDE discovery:

- Separating fully-differentiable training and Symbolic Regression (SR) inference stages
- Using a flexible derivative library rather than a functional library
- Observed solutions are approximated using a deep neural network (DNN) and derivatives calculated using AD

Problem Statement

Consider PDEs linear in their derivatives with spatiotemporal samples (x,t) and partially observed solution data, $u(x,t)$,

$$u_t = f_0 + f_1\,u_x + f_2\,u_{xx} + f_3\,u_{tt} + \cdots + f_j\,u_{xxx} + \ldots \qquad u_{tt} = g_0 + g_1\,u_x + g_2\,u_t + g_3\,u_{xx} + \cdots + g_j\,u_{xxx} + \ldots$$

Many physical systems of interest:

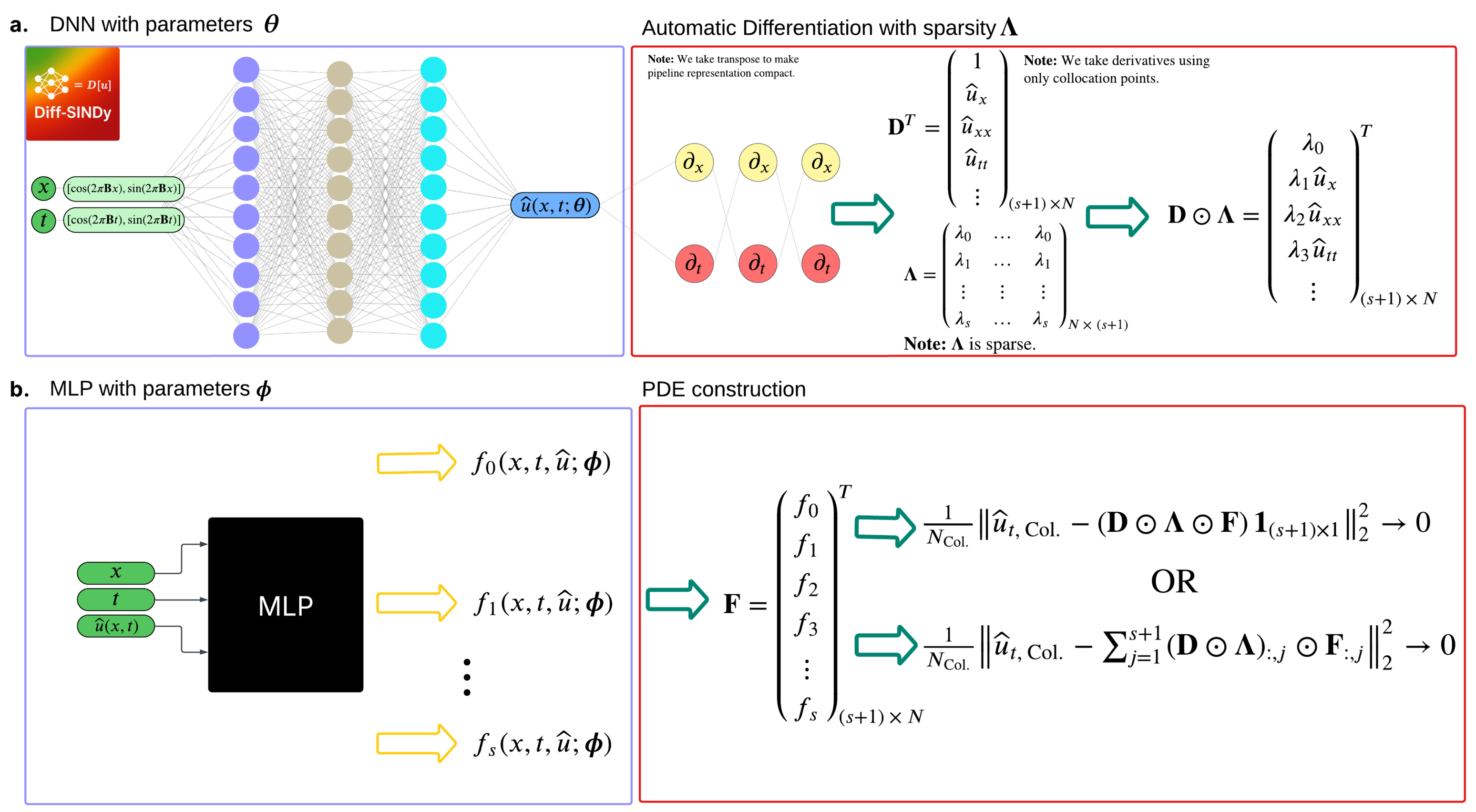
Heat: $u_t = \nu u_{xx}$ — Wave: $u_{tt} = c^2 u_{xx}$

Burgers': $u_t = -\alpha u u_x + \nu u_{xx}$ — Klein–Gordon: $u_{tt} = c^2 u_{xx} - m^2 u$

KdV: $u_t = -\alpha u u_x - \nu u_{xxx}$ — Sine–Gordon: $u_{tt} = c^2 u_{xx} - m^2 \sin u$

Diff-SINDy learns symbolic expressions for coefficients $f_j(x,t,u)$, $g_j(x,t,u)$, providing a closed-form parsimonious PDE that describes the data.

Model Architecture and Training



First, a DNN is trained to approximate the solution using an initial condition (IC), boundary condition (BC), and interior data (Data) losses.

$$\mathcal{L}_{\mathrm{IC}}(\theta;\ \overbrace{u_{\mathrm{IC}}}^{\text{measured}}\ ) = \frac{1}{N_{\mathrm{IC}}}\Big\|\overbrace{\widehat{u}_{\mathrm{IC}}}^{\text{predicted}} - u_{\mathrm{IC}}\Big\|_2^2;\ \mathcal{L}_{\mathrm{Data}}(\theta; u_{\mathrm{Data}}) = \frac{1}{N_{\mathrm{Data}}}\|\widehat{u}_{\mathrm{Data}} - u_{\mathrm{Data}}\|_2^2;\ \mathcal{L}_{\mathrm{BC}}(\theta; u_{\mathrm{BC}}) = \frac{1}{N_{\mathrm{BC}}}\|\widehat{u}_{\mathrm{BC}} - u_{\mathrm{BC}}\|_2^2$$

The solution network is warmed by minimizing the loss:

$$\mathcal{L}_{\mathrm{warm}} = \lambda_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}} + \lambda_{\mathrm{BC}}\mathcal{L}_{\mathrm{BC}} + \lambda_{\mathrm{Data}}\mathcal{L}_{\mathrm{Data}}$$

Then, the approximate solutions for the derivatives are calculated via AD. A sparse matrix Λ is used to select active and non-active derivatives.

The final component is the coefficient network, implemented with an MLP architecture in one of three forms: a single MLP for all coefficients, a multi-task MLP with a shared backbone and separate heads, or individual MLPs for each coefficient. It receives spatiotemporal coordinates and predicted solutions at the collocation points as input.

Training proceeds in two steps:

1. The solution network is trained with sparsity and coefficient network(s) frozen. The PDE is gradually introduced (i.e. λ_{PDE} gradually increases to 1).

$$\mathcal{L}\,(\overbrace{\boldsymbol{\theta}}^{\text{unfrozen}};\ x, t, u, \overbrace{\boldsymbol{\phi}, \boldsymbol{\Lambda}}^{\text{frozen}}) = \lambda_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}} + \lambda_{\mathrm{BC}}\mathcal{L}_{\mathrm{BC}} + \lambda_{\mathrm{Data}}\mathcal{L}_{\mathrm{Data}} + \lambda_{\mathrm{PDE}}(\mathrm{epoch})\mathcal{L}_{\mathrm{PDE}}$$

2. The coefficient network with sparsity (STRidge) is trained with solution network frozen.

$$\mathcal{L}(\overbrace{\boldsymbol{\phi}, \boldsymbol{\Lambda}}^{\text{unfrozen}}; x_{\mathrm{Col.}}, t_{\mathrm{Col.}}, u_{\mathrm{Col.}}, \overbrace{\boldsymbol{\theta}}^{\text{frozen}}) = \mathcal{L}_{\mathrm{PDE}} + \beta\|\boldsymbol{\Lambda}\|_0$$

Inference

After sparsity is applied, the active coefficients are passed to PySR (Python Symbolic Regression) [4] which infers closed-form expressions by a combination of a genetic evolutionary search algorithm, symbolic manipulation, pruning, and heuristics.

The expressions from PySR are ranked by symbolic complexity. The *score* of the ith expression balances the trade off between its fit quality from the mean squared error (MSE) loss and its simplicity, quantified by its comparison to the next simplest $i-1$ expression.

Complexities ($\mathcal{C}$)	MSE Loss	Score
`[const, +, -, *]`: 1 `[/, sin, cos]`: 2 `[exp]`: 4	$\mathcal{L}_{SR,i}(u) = \big\|f_{SR,i} - f_i\big\|_2^2$	$-\dfrac{\log(\mathcal{L}_{SR,i} - \mathcal{L}_{SR,i-1})}{\mathcal{C}_i - \mathcal{C}_{i-1}}$.

Experiments

We trained the model using a 200-epoch warm start followed by 1,000 epochs in a two-stage procedure. Training required 30 minutes and inference 10 minutes on a single T4 GPU. For the Burgers', heat, and wave equations, the predicted terms differed from the true equations by 0.5%/6% (u_x/u_{xx}), 1%, and 0.3%, respectively.

	Burgers' equation	Heat equation	Wave equation
Exact	$u_t = -1uu_x + \frac{0.01}{\pi}u_{xx}$	$u_t = 0.005\,u_{xx}$	$u_{tt} = 1\,u_{xx}$
Diff-SINDy	$u_t = -0.995uu_x + 0.00298u_{xx}$	$u_t = 0.00494\,u_{xx}$	$u_{tt} = 0.997\,u_{xx}$

Diff-SINDy results for coefficients of u_x and u_{xx} of Burger's equation.

Complexity	Expression	MSE	Score
	$\alpha u = -1u$		
1	x	0.2787	0.000
3	$-0.995\,u$	3.451×10^{-4}	3.346
5	$-0.982\,u + 0.003$	3.392×10^{-4}	0.0099
7	$-0.958(-0.028\,x + u)$	1.835×10^{-4}	0.510
	$\nu = 0.01/\pi \approx 0.0318$		
1	0.00298	2.148×10^{-7}	0.000
5	$0.0009x + 0.003$	1.962×10^{-7}	0.242
7	$0.004\cos(x - 0.303)$	1.334×10^{-7}	0.191
8	$0.0005(x - 0.668)^3 + 0.003$	9.729×10^{-8}	0.316

References

[1] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems,"

[2] Z. Chen, Y. Liu, and H. Sun, "Physics-informed learning of governing equations from scarce data,"

[3] Y. Jiang and J. Sun, "Evo-sindy: Universal discovery of partial differential equations using cooperative evolutionary computation,"

[4] M. Cranmer, "Interpretable Machine Learning for Science with PySR and SymbolicRegression,"