ICML
International Conference
On Machine Learning

Closed-form Solutions: A New Perspective on Solving Differential Equations

Shu Wei[12], Yanjie Li[1], Lina Yu[12], Weijun Li[134], Min Wu[1], Linjun Sun[1], Jingyi Liu[1], Hong Qin[13], Yusong Deng[12], Jufeng Han[13],Yan Pang[14]

[1]AnnLab, Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China
[2]College of Materials Science and Opto Electronic Technology, University of Chinese Academy of Sci ences, Beijing, China
[3]School of Integrated Circuits, University of Chinese Academy of Sciences, Beijing, China
[4]School of Industry-education Integration, University of Chinese Academy of Sciences, Beijing, China.
Correspondence to: Lina Yu<yulina@semi.ac.cn>, Min Wu <wumin@semi.ac.cn>

Preliminaries: Closed-form Solutions to DEs

Example: Heat Equation

Governing Equation:

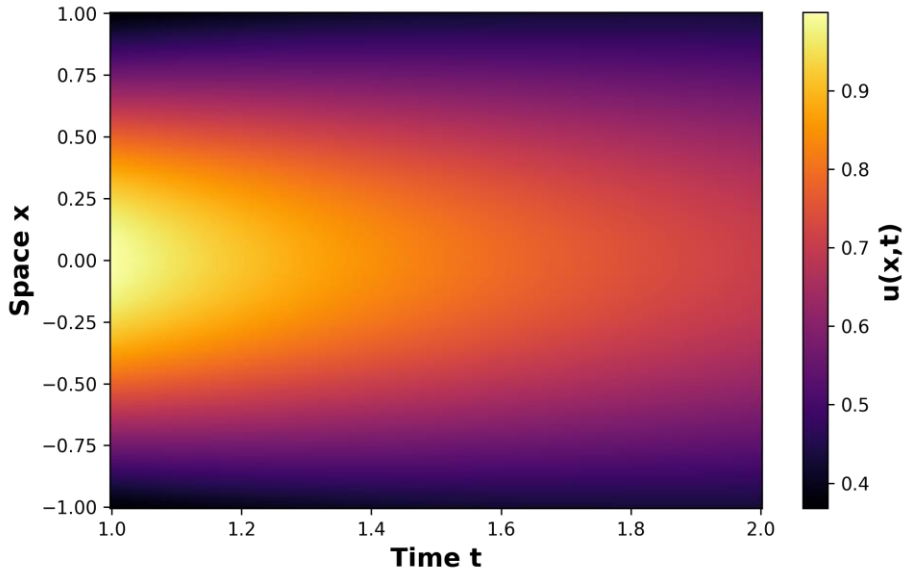
$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{4\partial x^2} = 0, x \in [-1,1], t \in [1,2]$$

Boundary Conditions:

$$u(-1,t) = u(1,t) = \frac{1}{\sqrt{t}} e^{-\frac{1}{t}}$$

Initial Condition:

$$u(x,1) = e^{-x^2}$$

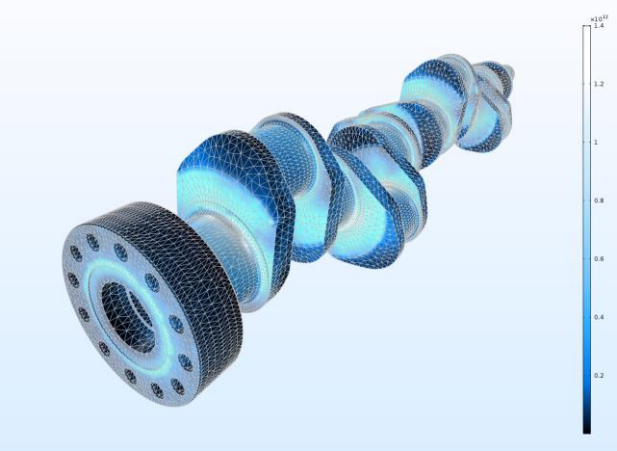


Closed-form solution:

$$u(x,t) = \frac{1}{\sqrt{t}} e^{-\frac{x^2}{t}}$$

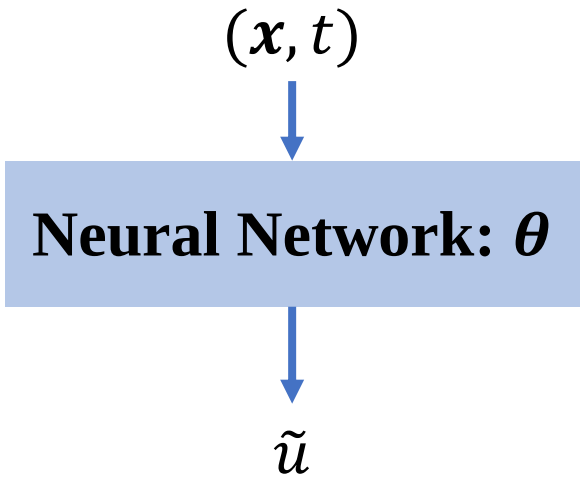
Goal: To Find a **Closed-form solution** $\hat{u}(\boldsymbol{x})$ to a differential equation (DE), expressed as a finite combination of known functions, such as elementary functions (e.g., polynomials, exponentials, trigonometric functions).

Reviews: Numerical Methods



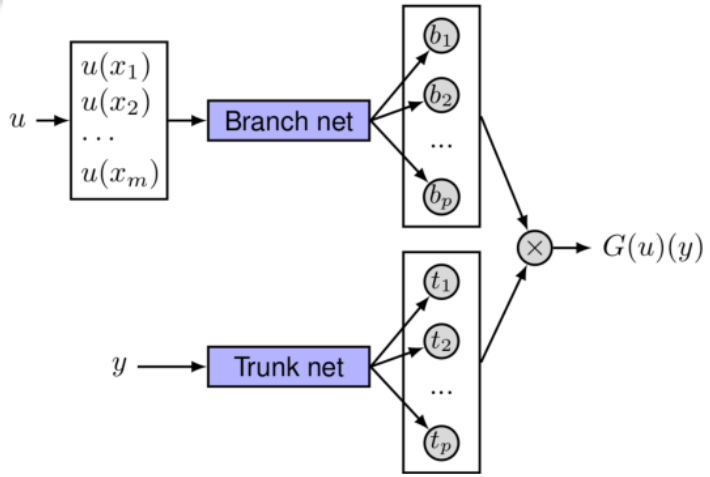
Traditional numerical methods (e.g., FEM)

The numerical accuracy and convergence depend on meshing



Physics-informed Neural Networks

The accuracy and convergence of the solution are limited.



Neural Operators

Requiring a large amount of labeled data.

Weaknesses of All Numerical Methods

W1. Limited accuracy and stability.

W2. Low interpretability.

Physics, PDEs, and Numerical Modeling, **COMSOL Multiphysics Cyclopedia 2019**

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equation, **Journal of Computational Physics 2019**

Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, **Nature machine intelligence 2021**

Reviews: Symbolic Methods

Expression Structure Search

Representative algorithms:

Genetic Programming[1], Colony Programming[2], etc.

Limitations:

- Extremely time-consuming
- Poor scalability to high-dimensional PDEs
- Tend to produce overly complex expressions

Symbolically Structured Neural Networks

Representative algorithms:

FEX[3], MSFL[4], etc.

Limitations:

- Low accuracy and poor convergence
- Limited interpretability
- Hard to guarantee exact solution form

The core challenge lies in balancing three critical aspects:

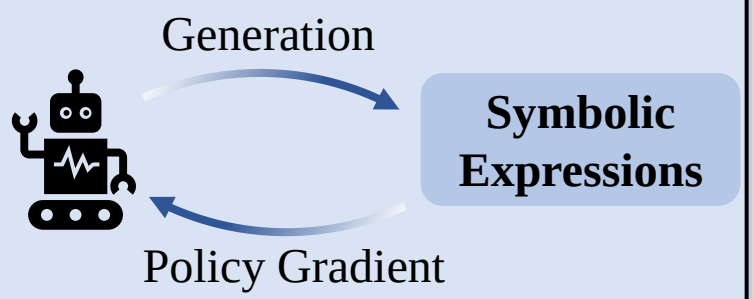
A1. Maintaining strict adherence to physical laws.
A2. Ensuring computational efficiency
A3. Preserving human-interpretable symbolic forms.

1 Genetic programming based symbolic regression for analytical solutions to differential equations, **arXiv:2302.03175, 2023**
2 Solving differential equations with ant colony programming**. Applied Mathematical Modelling 2019**
3 Finite expression method for solving high-dimensional partial differential equations. **arXiv:2206.10121, 2022**.
4 Symbolically Solving Partial Differential Equations using Deep Learning, **arXiv:2011.06673, 2020**

Motivations

Risk-Seeking Reinforcement Learning

- Optimize symbolic expressions using policy gradient.
- The policy gradient depends on the reward of the top-ϵ expressions.



Parametric expressions

- The symbolic solution is represented as a parametric expression with respect to a single variable.
- The complete solution is constructed recursively from these parametric expressions.

$$u(x,t) = e^{-\frac{x^2}{t}}/\sqrt{t}$$

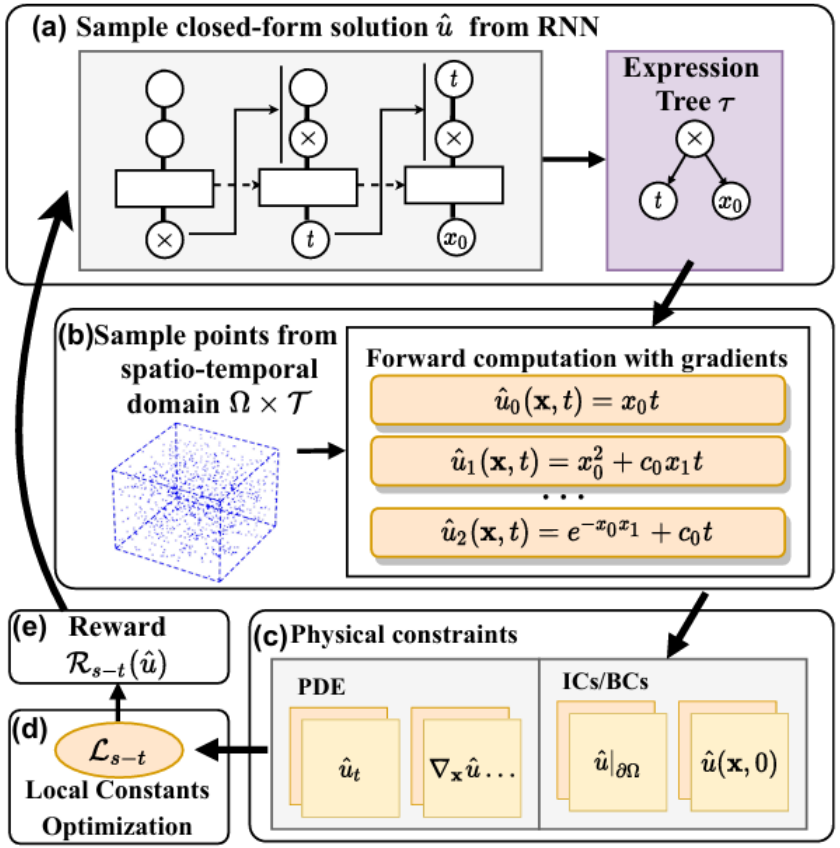
$$\tilde{u}_0(x) = \alpha_1^0 e^{\alpha_1^1 x^2}$$

$$\tilde{u}_1^0(t) = \alpha_1^0 = \frac{1}{\sqrt{t}} \qquad \tilde{u}_1^1(t) = \alpha_1^1 = \frac{-1}{t}$$

Contributions

1) We propose **SSDE**, a reinforcement learning-based method for closed-form solutions' structure search.
2) We introduce the **RSCO** algorithm for efficient constant optimization.
3) We develop a **recursive** exploration strategy for solving high-dimensional PDEs **dimension by dimension.**

Proposed Method



Algorithm Overview

(a) The RNN generates skeletons for candidate solutions.

(b) Sampled points are fed into the candidate skeletons to construct computational graphs with constants c_i as parameters.

(c) Physical constraints are built via automatic differentiation.

(d) Constants are optimized to minimize L_{s-t} .

(e) The evaluator computes rewards based on physical constraints to train RNN.

The process iterates until a valid solution is found.

Physics-regularized Loss:

$$\mathcal{L}_{\text{s-t}} = \lambda_0 \text{MSE}_{\mathcal{F}} + \lambda_1 \text{MSE}_{\mathcal{B}} + \lambda_2 \text{MSE}_{\mathcal{I}}, \text{MSE}_{\mathcal{F}} = \frac{1}{N_{\mathcal{F}}} \sum_i^{N_{\mathcal{F}}} |\mathcal{F}(x_f^i, t_f^i)|^2$$

$$\text{MSE}_{\mathcal{B}} = \frac{1}{N_{\mathcal{B}}} \sum_i^{N_{\mathcal{B}}} |\hat{u}(x_b^i, t_b^i) - u_b^i|^2, \text{MSE}_{\mathcal{I}} = \frac{1}{N_{\mathcal{I}}} \sum_i^{N_{\mathcal{I}}} |\hat{u}(x_0^i, 0) - u_0^i|^2$$

Risk-Seeking Policy Gradient:

$$\nabla_\theta J_{\text{risk}}(\theta, \epsilon) = \mathbb{E}_{\hat{u} \sim p(\hat{u}|\theta)} [(\mathcal{R}(\hat{u}) - \mathcal{R}_\epsilon(\theta)) \cdot \nabla_\theta \log p(\hat{u} \mid \theta) \mid \mathcal{R}(\hat{u}) \geq \mathcal{R}_\epsilon(\theta)]$$

Reward: $$\mathcal{R}_{\text{s-t}}(\hat{u}) = \frac{1}{1 + \text{Average}(\sqrt{\text{MSE}_{\mathcal{F}}} + \sqrt{\text{MSE}_{\mathcal{B}}} + \sqrt{\text{MSE}_{\mathcal{I}}})}$$

Closed-form Solutions: A New Perspective on Solving Differential Equations. ICML 2025

Proposed Method

Risk-Seeking Constant Optimization

Coarse optimization by minimizing

$$\mathcal{L}'_{s-t} = \lambda_1 \mathrm{MSE}_{\mathcal{B}} + \lambda_2 \mathrm{MSE}_{\mathcal{I}}$$

→ **Select top-ϵ expressions by reward and refine by minimizing** $\mathcal{L}_{s-t}$

Recursion-Based Exploration

Parametric expression

Solution $\tilde{u}_k(x_i, \vec{\alpha}_k(\boldsymbol{x}_{-i}))$ is only respect to observed variable x_i

Recursive call

The parameter $\vec{\alpha}_k$ are viewed as new parametric expressions.

Recursive case

Get Closed-form Solution $\tilde{u}_k(x_i, \vec{\alpha}_k)$ by RL agent.

Base case

Terminates once the last variable is observed.

Distinguish between fixed constants and $\vec{\alpha}$ parameters:

Parameters are optimized as vectors over the unobserved space. Low-variance parameters are identified as constants. When only one unobserved dimension remains, the parameters are directly optimized as constants.

Experimental Results

Benchmark

- Time-dependent PDE systems
- Time-independent PDE systems
- Linear PDEs
- Nonlinear PDEs

Name	PDE	Deterministic Conditions
Poisson2D	$\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}=30x_1^2-7.8x_1+1, \mathbf{x}\in[-1,1]^2$	$u(\mathbf{x})=2.5x_1^4-1.3x_1^3+0.5x_2^2-1.7x_2, \mathbf{x}\in\partial[-1,1]^2$
Poisson3D	$\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\frac{\partial^2 u}{\partial x_3^2}=30x_1^2-7.8x_2+1, \mathbf{x}\in[-1,1]^3$	$u(\mathbf{x})=2.5x_1^4-1.3x_2^3+0.5x_3^2, \mathbf{x}\in\partial[-1,1]^3$
Heat2D	$\frac{\partial u}{\partial t}-(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2})=-30x_1^2+7.8x_2+t, \mathbf{x}\in[-1,1]^2$	$u(\mathbf{x},t)=2.5x_1^4-1.3x_2^3+0.5t^2, \mathbf{x}\in\partial[-1,1]^2, t\in[0,1]$ $u(\mathbf{x},0)=2.5x_1^4-1.3x_2^3, \mathbf{x}\in[-1,1]^2$
Heat3D	$\frac{\partial u}{\partial t}-(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\frac{\partial^2 u}{\partial x_3^2})=-30x_1^2+7.8x_2-2.7$, $\mathbf{x}\in[-1,1]^3$	$u(\mathbf{x},t)=2.5x_1^4-1.3x_2^3+0.5x_3^2-1.7t, \mathbf{x}\in\partial[-1,1]^3, t\in[0,1]$ $u(\mathbf{x},0)=2.5x_1^4-1.3x_2^3+0.5x_3^2, \mathbf{x}\in[-1,1]^3$
Wave2D	$\frac{\partial^2 u}{\partial t^2}-(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2})=-u-u^3+((0.25-4x_1^2)\cdot e^{x_1^2-0.5t}+e^{3x_1^2-1.5t}\sin(x_2)^2)\sin(x_2), \mathbf{x}\in[-1,1]^2$	$u(\mathbf{x},t)=\exp(x_1^2)\sin(x_2)e^{-0.5t}, \mathbf{x}\in\partial[-1,1]^2, t\in[0,1]$ $u(\mathbf{x},0)=\exp(x_1^2)\sin(x_2), \mathbf{x}\in[-1,1]^2$
Wave3D	$\frac{\partial^2 u}{\partial t^2}-(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\frac{\partial^2 u}{\partial x_3^2})=u^2-((4x_1^2+4x_3^2+2.75)\cdot e^{x_1^2+x_3^2-0.5t}+e^{2x_1^2+2x_3^2-t}\cos(x_2))\cos(x_2), \mathbf{x}\in[-1,1]^3$	$u(\mathbf{x},t)=\exp(x_1^2+x_3^2)\cos(x_2)e^{-0.5t}, \mathbf{x}\in\partial[-1,1]^3, t\in[0,1]$ $u(\mathbf{x},0)=\exp(x_1^2+x_3^2)\cos(x_2), \mathbf{x}\in[-1,1]^3$

Name	Identified Solution $\hat{u}$
Poisson2D	$x_1^2(2.5000x_1^2-1.3000x_1)+0.4999x_2(x_2-1.4002)-x_2$
Poisson3D	$2.5000x_1^4-1.3000x_2^3+0.4999x_3^2+2.3763e-5$
Heat2D	$0.5000t^2+\log(e^{2.5000x_1^4-x_2^2(1.3000x_2-8.1811e-7)})$
Heat3D	$-1.7003t+2.5000x_1^4-1.3000x_2^3+0.4999x_3^2+0.0002$
Wave2D	$\exp(-0.5000t+x_1^2)\sin(x_2)$
Wave3D	$\exp(-0.5000t+1.000x_1^2+x_3^2)\cos(x_2)^{1.000}$

Results of SSDE

- Successfully identified the correct symbolic skeletons for all test cases
- Discovered solutions span from simple polynomials to complex nested nonlinear expressions

Experimental Results

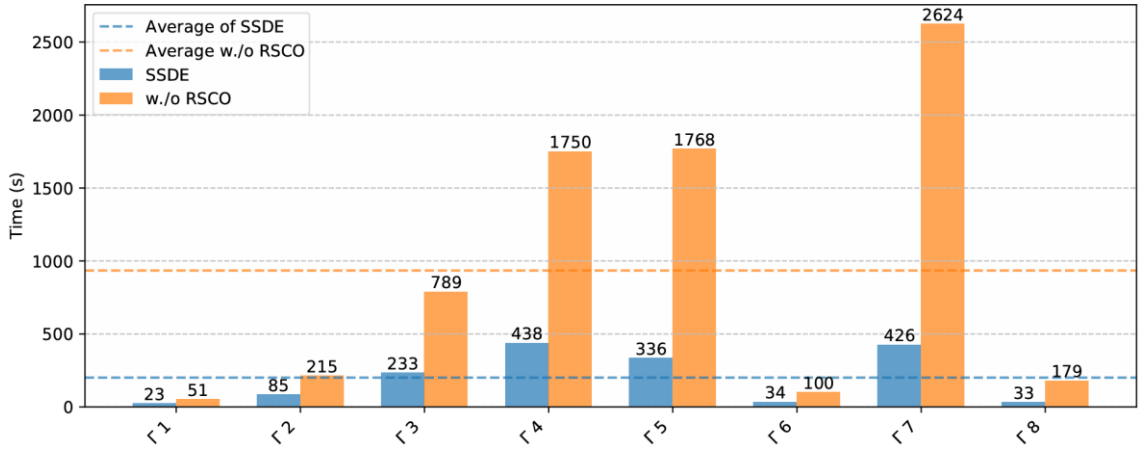
Compared with other mainstream methods

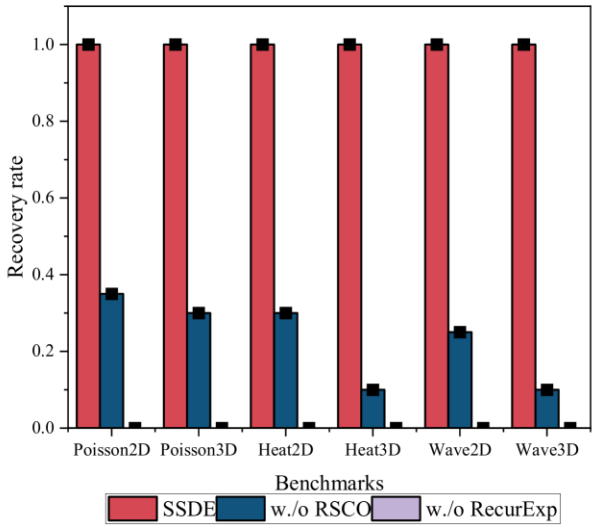
NAME	SSDE	MATHEMATICA	PR-GPSR	KAN	FEX	DSR	PINN
POISSON2D	**1.55E-05**	✗	1.49E-02	3.24E+00	2.00E-02	9.85E-02	6.24E-04
POISSON3D	**3.58E-06**	✗	2.01E-02	2.57E+00	9.13E-02	2.15E-01	3.93E-03
HEAT2D	**4.45E-06**	✗	4.57E-02	6.54E+00	4.70E-02	2.19E-01	6.98E-03
HEAT3D	**9.34E-06**	✗	1.39E+00	5.58E+00	3.63E-01	3.20E-01	1.15E-02
WAVE2D	**3.11E-05**	✗	1.80E-01	1.18E+00	2.34E-01	7.40E-02	1.34E-02
WAVE3D	**7.62E-06**	✗	4.56E-01	7.26E+00	3.97E-01	1.95E+00	3.69E-02

Ablation Studies

Left: Computational efficiency with RSCO

Right: Recovery rate without RSCO or Recursive Exploration





Conclusion

- We propose a reinforcement learning-based architecture for analytically solving differential equations using parametric expressions. The approach significantly enhances the ability to discover true solution structures in high-dimensional settings.
- We demonstrate that SSDE successfully solves all benchmark PDEs with exact symbolic skeletons and significantly outperforms existing mainstream methods.

Limitations

SSDE currently faces efficiency limitations when applied to systems of differential equations, due to their inherently coupled solution structures. Fully exploring the solution space requires multiple RNNs and novel techniques for evaluating reward-guided gradients. Addressing this challenge is a key direction for future work.

ICML International Conference On Machine Learning

AnnLab

Paper

Presenter: Shu Wei

Email: weishu@semi.ac.cn

References

Physics, PDEs, and Numerical Modeling, **COMSOL Multiphysics Cyclopedia 2019**

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equation, **Journal of Computational Physics 2019**

Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, **Nature machine intelligence 2021**

Genetic programming based symbolic regression for analytical solutions to differential equations, **arXiv:2302.03175, 2023**

Solving differential equations with ant colony programming**. Applied Mathematical Modelling 2019**

Finite expression method for solving high-dimensional partial differential equations. **arXiv:2206.10121, 2022**.

Symbolically Solving Partial Differential Equations using Deep Learning, **arXiv:2011.06673, 2020**