Does In-Context Operator Learning Generalize to Domain-Shifted Settings?

Jerry W. Liu[1]*, N. Benjamin Erichson[2,3], Kush Bhatia[5], Michael W. Mahoney[2,3,4], and Christopher Ré[5]

[1]Institute of Computational and Mathematical Engineering, Stanford University
[2]International Computer Science Institute
[3]Lawrence Berkeley National Laboratory
[4]Department of Statistics, University of California at Berkeley
[5]Department of Computer Science, Stanford University

Abstract

Neural network-based approaches for learning differential equations (DEs) have demonstrated generalization capabilities within a DE *solution* or *operator instance*. However, because standard techniques can only represent the solution function or operator for a single system at a time, the broader notion of generalization *across classes of DEs* has so far gone unexplored. In this work, we investigate whether commonalities across DE classes can be leveraged to transfer knowledge about solving one DE towards solving another DE—without updating any model parameters. To this end, we leverage the recently-proposed in-context operator learning (ICOL) framework, which trains a model to identify in-distribution operators, given a small number of input-output pairs as examples. Our implementation is motivated by pseudospectral methods, a class of numerical solvers that can be systematically applied to a range of DEs. For a natural distribution of 1D linear ordinary differential equations (ODEs), we identify a connection between operator learning and in-context linear regression. Applying recent results demonstrating the capabilities of Transformers to in-context learn linear functions, our reduction to least squares helps to explain why Transformers can be expected to solve ODEs in-context. Empirically, we demonstrate that ICOL is robust to a range of distribution shifts, including observational noise, domain-shifted inputs, varying boundary conditions, and surprisingly, even operators from functional forms unseen during training.

1 Introduction

Neural network-based techniques, including physics-informed neural networks [4, 8, 14, 15, 16] and neural operators [2, 7, 10, 11, 12, 17], provide a flexible framework for solving a variety of differential equation (DE) problems [3]. However, one property we might want from machine learning approaches for DEs is an ability to generalize to wider settings at inference time (without costly parameter updates). Specifically, there are three broad notions of generalization that are desirable:

- within a DE *solution instance*: given a fixed set of DE parameters and inputs, can a learned method generalize to points in the domain previously unseen during training?
- within a DE *operator instance*: given a fixed set of DE parameters, can a learned method generalize to a new distribution of inputs?
- within and/or across *classes of DE operators*: can a learned method generalize to a new distribution of DE parameters, or even to DEs with unseen functional forms?

*Corresponding author: jwl50@stanford.edu

DLDE-III Workshop in the 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Most of the existing neural network-based approaches are able to handle generalization within DE solution and operator instances. However, since these methods can only represent the solution or solution operator for a single system at a time, none of them address the broadest and most challenging sense of generalization: across classes of DE operators.

In this work, we are interested in understanding (i) whether it is possible to exploit commonalities between different classes of DEs to transfer knowledge about solving one DE toward solving another DE; and (ii) what kinds of generalization can be expected within/across DE classes. To this end, we leverage the in-context operator learning framework [20], which trains a model to identify in-distribution operators, given a small number of input-output pairs as examples. Our implementation is motivated by spectral collocation (pseudospectral) methods [13], a class of numerical solvers that can be systematically applied to a range of DEs, often to high accuracy [6, 19]. From the machine learning literature, we exploit recent work [1, 5, 21] that shows Transformers are able to solve *in-context* least squares problems. That is, they are able to identify instances of least squares problems *at inference time* given input-output pairs, and produce corresponding outputs given new inputs.

As a proof-of-concept, we focus on a simple distribution of 1D linear ODEs, for which results from approximation theory create a natural connection between pseudospectral methods and in-context linear regression. Empirically, we investigate the generalization capacity of our in-context operator learner both *within* a distribution of DEs, with out-of-distribution (OOD) queries and examples, and *across* classes of ODEs. We find that our method can generalize to a variety of in-class domain-shifts, including OOD boundary conditions, noisy in-context examples, and OOD operator parameters. Interestingly, we show that our model, trained on 1st-order linear ODEs, also generalizes to a distribution of 1st-order ODEs with non-linear solution-forcing interaction and to 2nd-order linear ODEs, demonstrating that generalization across DEs of different functional forms is indeed possible. Our results are a first step towards a better understanding of the limits of generalization for DEs.

2 Hierarchy of Generalization for Differential Equations

Here, we describe the levels of generalization we would ideally want from our neural network-based approaches to DE problems to satisfy *at inference time*. For the sake of concreteness, we describe them for a simple distribution of parameterized ODEs:

$$\mathcal{L}_\alpha(c, u_0) = u \mid \begin{cases} u'(t) = \alpha c(t) \\ u(0) = u_0 \end{cases} \quad \text{on } \Omega = [-1, 1], \tag{1}$$

with parameter α, forcing function c, initial condition u_0, and solution u.

- **Generalization within a DE solution instance.** We define a DE *solution instance*, u, by fixing the parameter α and inputs (c, u_0). In this setting, we ask whether a method trained on a subset of the domain Ω^{train}, say $u(t \in [0, 1])$, can generalize to an unseen subset Ω^{test}, say $u(t \in [-1, 0])$.
- **Generalization within a DE operator instance.** We define a DE *operator instance*, $\mathcal{L}_\alpha$, by fixing the parameter α but allowing the inputs (c, u_0) to be drawn from a distribution. In this setting, we ask whether a method trained on inputs from D^{train}_{inputs}, say $c = \exp(-\beta t), \{\beta, u_0\} \in [-1, 1]$ can generalize to a new query distribution $(c, u_0)^q \sim D^{test}_{inputs}$, say $\{\beta, u_0\} \in [-2, 2]$.
- **Generalization within a class of DE operators.** We define a *class* of DE operators by allowing the parameter α to also be drawn from a distribution. In this setting, we ask whether a method trained on an operator distribution D^{train}_{op}, say $\alpha \in [0, 1]$, can generalize to a new test operator distribution D^{test}_{op}, say $\alpha \in [2, 4]$.
- **Generalization across classes of DE operators.** Now we consider two different classes of DE operators with two different functional forms, e.g., $\mathcal{L}_\alpha$ (above) and

$$\mathcal{L}^2_\beta(c, u_0) = u \mid \begin{cases} u'(t) = \beta c(t) u(t) \\ u(0) = u_0 \end{cases} \quad \text{on } \Omega = [-1, 1]. \tag{2}$$

  We ask whether a method trained on $\{\mathcal{L}_\alpha\}$ can generalize to $\{\mathcal{L}^2_\beta\}$.

The different types of generalization depend on the the particular model and training setup. Table 1 provides a summary. Standard neural network-based approaches represent a single system with a

Table 1: Hierarchical notions of generalization. In-context operator learning, where a single model is trained on multiple distributions of operators, allows transfer to occur within/across classes of DEs.

One network trained to represent:	Unseen points in domain	OOD inputs	OOD examples	OOD operators
One solution instance [4, 14, 15, 16]	✓	✗	✗	✗
One operator instance [2, 7, 10, 11, 12]	✓	✓	✗	✗
Multiple distributions (ours, [20])	✓	✓	✓	✓

single network, and thus they have limited generalization capabilities. In particular, generalization within/across a *class* of DE operators at inference time (without any parameter updates) requires a network that solves the *meta-level* problem of representing a distribution of operators at once.

3 Connecting Pseudospectral Methods and In-Context Least Squares

Here, we describe the in-context operator learning (ICOL) method we use to meta-learn a distribution of DEs. Using pseudospectral methods, a general class of numerical solvers with excellent theoretical guarantees, we reduce solving a class of 1D linear ODEs to solving linear systems. Motivated by this result and by recent works showing the capacity for Transformers to learn linear functions *in-context* (i.e., during inference time), we use a similar argument from approximation theory to reduce ICOL to in-context least squares.

Reducing linear ODEs to linear systems We focus on an idealized class of operators, namely 1D ODEs which are simultaneously linear in the forcing term and the solution:

$$\mathscr{L}_{linear} := \left\{ \mathcal{L}_\phi := \sum_{k=0}^{m} \alpha_k(t) \frac{\partial^{(k)}}{\partial t^{(k)}} u(t) + \beta_k(t) \frac{\partial^{(k)}}{\partial t^{(k)}} c(t) = b(t), \quad t \in [-1, 1] \right\}_\phi , \tag{3}$$

with parameters $\phi = \{\alpha_0, \ldots, \alpha_m, \beta_0, \ldots, \beta_m, b\}$, forcing function c, and solution u.

By a standard result from approximation theory, if we sample a smooth-enough function f on $[-1, 1]$ on the Chebyshev-Gauss-Lobatto nodes, the derivatives of the polynomial interpolant well-approximate the derivatives $f^{(k)}$ evaluated on the Chebyshev nodes, and they can be written as a *linear combination* of the values of f. (See Appendix A for details.) This argument reduces our linear ODEs to $(N+1)$-dimensional linear systems, where functions are sampled on the Chebyshev nodes and the design matrix is specified by the ODE and its parameters. (See Appendix B.1 for details.)

In-context learning for operators. The connection to linear systems motivates us to meta-learn distributions of ODEs using the recently-proposed ICOL paradigm [20]. Using ICOL, the challenge of meta-learning can be reduced to supervised learning on sequences.

Specifically, during training, we sample linear ODE operators from a distribution $D_{op}^{train} \subset \mathscr{L}_{linear}$, where $\mathcal{L}_\phi \in D_{op}^{train}$ maps forcing functions and boundary/initial conditions (BICs) (c, u_0) to the solution sampled on a Chebyshev node, $u(t_j) \in \mathbb{R}$. For each operator $\mathcal{L}_\phi$, we sample K forcing functions and BICs from a training distribution D_{inputs}^{train}. As in the reduction, we represent functions $c : [-1, 1] \to \mathbb{R}$ as $(N+1)$-dimensional vectors $\vec{c} = \{c(t_j)\}_{j=0}^N$, sampled on Chebyshev nodes. We construct a training prompt P as follows:

$$P = \left((\vec{c}, u_0, 1)^{(1)}, u^{(1)}(t_j), (\vec{c}, u_0, 1)^{(2)}, u^{(2)}(t_j), \ldots, (\vec{c}, u_0, 1)^{(K)}, u^{(K)}(t_j) \right).$$

We supervise a large sequence model T_θ to predict the output $u^q(t_j) = u^{(k+1)}(t_j)$, given the first k in-context examples from our prompt and the query $(c, u_0)^{(k+1)}$. The training objective is to minimize the expected mean squared error over our prompt distribution:

$$\min \ \mathbb{E}_P \left[\frac{1}{K} \sum_{k=0}^{K-1} ||T_\theta(P^k) - u^{(k+1)}(t_j)|| \right], \tag{4}$$

with $P^k = \left((\vec{c}, u_0, 1)^{(1)}, u^{(1)}(t_j), \ldots, (\vec{c}, u_0, 1)^{(k)}, u^{(k)}(t_j), (\vec{c}, u_0, 1)^{(k+1)} \right)$.

Reducing ICOL to in-context linear regression By the same approximation theory argument as the linear ODEs reduction, the ICOL task from Equation 4 can be reduced (to high precision) to an in-context linear regression problem. (See Appendix B.2 for details.)

This equivalence between ICOL and in-context linear regression helps explain Transformers' generalization capabilities for operator learning from the lens of recent theoretical results [1, 21].

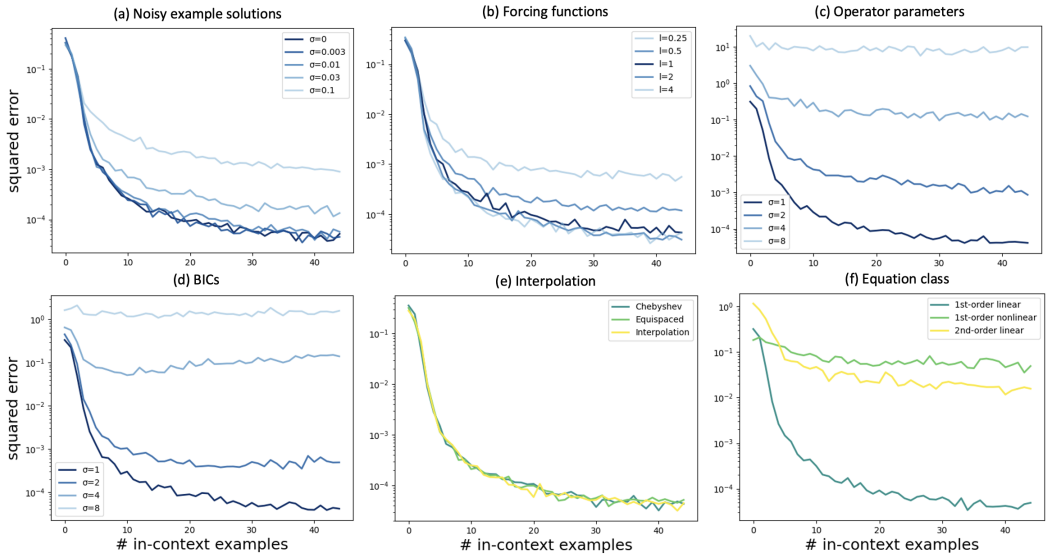

Figure 1: Generalization within (a-d) and across (e-f) operator classes.

4 Experimental Results

Here, we train an in-context operator learner using a Transformer backbone on a distribution of 1st-order, 1D linear ODEs, a subset of the simple class of operators (Equation 3). (See Appendix C.1 for details.)

Generalization. We focus on evaluating the two broadest and most challenging forms of out-of-distribution (OOD) generalization: *within an operator class* and *across operator classes*.

- **Within operator class.** We evaluate various forms of domain shifts affecting the input-output distributions of the in-context examples and queries, including observational noise in examples, OOD forcing functions, and OOD boundary conditions. (See Appendix C.1 for details.) On the operator level, we evaluate in-class OOD operator parameters.
- **Across operator classes.** We investigate two instances: (i) interpolation, where the operators evaluated at inference are of the form $\mathcal{L}_\phi \in D_{op}^{train}$ mapping $(c, u_0) \to u(t)$ for *general* $t \in [-1, 1]$ (recall the network is only trained on operators mapping to Chebyshev nodes); and (ii) unseen functional forms, where we evaluate on distributions of 1st-order ODEs with *non-linear* solution-forcing interaction terms and on *2nd-order* linear ODEs.

Results and discussion. Figure 1 shows that our method is able to handle the broadest senses of generalization. Within our DE class, we find the model is robust to various domain shifts on the in-context examples, queries, and operator parameters. Further, performance decays smoothly as extent of domain shift increases. We find that generalization to unseen boundary/initial condition distributions is the most challenging task for the within-class setting.

Most interestingly, we demonstrate positive results for generalization *across* classes of DEs. For the interpolation task, we surprisingly observe almost no generalization gap. This can be explained by a connection to approximation theory. (See Appendix B.3 for details.) We also observe surprising generalization across qualitatively different DE operators, i.e., from 1st- to 2nd-order and from linear (simultaneously in forcing and solution terms) to non-linear (solution-forcing interactions). Finally, we compare the Transformer model's performance to least squares. (See Appendix C.2 for details.) Interestingly, we observe that the model's accuracy saturates orders of magnitude above the performance of the least squares solver.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); US DEVCOM ARL under No. W911NF-21-2-0251 (Interactive Human-AI Teaming); ONR under No. N000141712266 (Unifying Weak Supervision); ONR N00014-20-1-2480: Understanding and Applying Non-Euclidean Geometry in Machine Learning; N000142012275 (NEPTUNE); NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-GCP Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Facebook, Google, and VMWare. J.W.L would like to acknowledge the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0023112. M.W.M. would like to acknowledge the DOE, NSF, and ONR for providing partial support of this work. N.B.E. would like to acknowledge support from NSF (DMS- 2319621), DOE (AC02-05CH11231), and NERSC (DE-AC02-05CH11231). Our conclusions do not necessarily reflect the position or the policy of our sponsors, and no official endorsement should be inferred.

References

[1] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023.

[2] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

[3] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52:477–508, 2020.

[4] Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7), 2022.

[5] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

[6] David Gottlieb and Steven A Orszag. *Numerical analysis of spectral methods: theory and applications*. SIAM, 1977.

[7] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

[8] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. Technical Report Preprint: arXiv:2109.01050, 2021.

[9] A. S. Krishnapriyan, A. F. Queiruga, N. B. Erichson, and M. W. Mahoney. Learning continuous models for continuous physics. Technical Report Preprint: arXiv:2202.08494, 2022.

[10] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020.

[11] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[12] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[13] Steven A Orszag. Comparison of pseudospectral and spectral approximation. *Studies in Applied Mathematics*, 51(3):253–259, 1972.

[14] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[15] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[16] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[17] Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W. Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *arXiv preprint arXiv:2306.00258*, 2023.

[18] Lloyd N Trefethen. *Spectral methods in MATLAB*. SIAM, 2000.

[19] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[20] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning for differential equation problems. *arXiv preprint arXiv:2304.07993*, 2023.

[21] Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

A Approximation, interpolation, and the pseudospectral method

Here, we provide results about the accuracy of polynomial interpolation in approximating a smooth function and its derivatives, when samples are taken on a specific distribution of nodes.

Definition 1. *The **Chebyshev-Gauss-Lobatto** points are defined as* $\{\cos\left(\frac{\pi j}{N}\right)\}_{j=0}^{N}$.

Definition 2. *Given* $N+1$ *distinct interpolation nodes* x_j, $j \in \{0,\dots,N\}$, *together with corresponding values* f_j, *the **Lagrange interpolating polynomial** is the unique degree* $\leq n$ *polynomial* p *that passes through the points* $\{(x_j, f_j)\}_{j=0}^{N}$*:*

$$p(x) = \sum_{j=0}^{N} f_j \ell_j(x), \quad \ell_j(x) = \frac{\prod_{k=0,k\neq j}^{n}(x - x_k)}{\prod_{k=0,k\neq j}^{N}(x_j - x_k)}.$$

Note that for any $x \in [-1,1]$, $p(x)$ is a linear combination of the f_j's. Similarly, we can evaluate the derivative of the Lagrange interpolating polynomial, $p'(x)$, at any $x \in [-1,1]$ as a linear combination of the f_j's. In particular, there exists a matrix that calculates $\{p'(x_j)\}_{j=0}^{N}$ when the x_j's are the Chebyshev-Gauss-Lobatto points.

Definition 3. *The* $(N+1)\times(N+1)$ *Chebyshev spectral differentiation matrix* $\boldsymbol{D}_N$ *is the unique matrix such that*

$$\boldsymbol{D}_N \begin{bmatrix} f_0 \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} p'(x_0) \\ \vdots \\ p'(x_N) \end{bmatrix}$$

$\boldsymbol{D}_N$ *has entries*

$$(\boldsymbol{D}_N)_{00} = \frac{2N^2+1}{6}, \quad (\boldsymbol{D}_N)_{NN} = -\frac{2N^2+1}{6}$$

$$(\boldsymbol{D}_N)_{jj} = -\frac{x_j}{2(1-x_j^2)}, \quad j \in \{1,\dots,N-1\}$$

$$(\boldsymbol{D}_N)_{ij} = \frac{c_i}{c_j}\frac{(-1)^{i+j}}{(x_i - x_j)}, \quad i \neq j, \quad i,j \in \{1,\dots,N-1\}$$

where

$$c_i = \begin{cases} 2 & i = 0 \text{ or } N \\ 1 & \text{else} \end{cases}.$$

For smooth-enough functions sampled on the Chebyshev-Gauss-Lobatto points, the derivatives of the Lagrange interpolant well-approximate the function's derivatives.

Theorem 1 (Accuracy of Chebyshev spectral differentiation, [18])**.** *Suppose* u *is analytic on and inside the ellipse with foci* ± 1 *on which the Chebyshev potential takes the value* ϕ_f*, that is, the ellipse whose semi-major and semi-minor axis lengths sum to* $K = \exp(\phi_f + \log 2)$*. Let* w *be the* ν*th Chebyshev spectral derivative of* u*. Then*

$$|w_j - u^{(v)}(x_j)| = O(\exp(-N(\phi_f + \log 2))) = O(K^{-N}).$$

B Pseudospectral reductions

B.1 Reducing ODEs to linear systems

Let's consider the class of 1D ODEs of order at most m which are linear in solution u and forcing term c simultaneously, defined on the domain $t \in [-1, 1]$:

$$\begin{cases} \sum_{k=0}^{m} \alpha_k(t) \frac{d^{(k)}}{dt^{(k)}} u(t) + \beta_k(t) \frac{d^{(k)}}{dt^{(k)}} c(t) = b(t), \quad t \in [-1, 1] \\ u(0) = u_0. \end{cases} \tag{5}$$

Sampling and interpolating on N Chebyshev nodes, this ODE reduces (to high precision) to an $N \times N$ linear system:

$$\begin{cases} \sum_{k=0}^{m} \vec{\alpha_k} \odot \boldsymbol{D}^k \vec{u} + \vec{\beta_k} \odot \boldsymbol{D}^k \vec{c} = \vec{b} \\ u(t=0) = u_0. \end{cases} \tag{6}$$

Solving for $\vec{u}$ in the homogeneous case first:

$$\left(\sum_{k=0}^{m} \operatorname{diag}(\vec{\alpha_k}) \boldsymbol{D}^k \right) \vec{u} = \left(\sum_{k=0}^{m} - \operatorname{diag}(\vec{\beta_k}) \boldsymbol{D}^k \right) \vec{c} + \vec{b},$$

and thus

$$\begin{aligned} \vec{u} &= \left(\sum_{k=0}^{m} \operatorname{diag}(\vec{\alpha_k}) \boldsymbol{D}^k \right)^{-1} \left(\sum_{k=0}^{m} - \operatorname{diag}(\vec{\beta_k}) \boldsymbol{D}^k \right) \vec{c} + \left(\sum_{k=0}^{m} \operatorname{diag}(\vec{\alpha_k}) \boldsymbol{D}^k \right)^{-1} \vec{b} \\ &:= \boldsymbol{W} \vec{c} + \vec{b}^*. \end{aligned}$$

In particular, we can approximate the operator that takes c and outputs $u(t_j)$, where t_j is the jth Chebyshev node, by the linear function

$$u(t_j) = \vec{w_j}^T \vec{c} + \vec{b}^*[j], \tag{7}$$

where $\vec{w_j}$ is the jth row of $\boldsymbol{W}$ and $\vec{b}^*[j]$ is the jth entry of $\vec{b}^*$.

The initial condition $u(0) = u_0$ is enforced by adding an extra term to the linear system in Equation 6. For example, for odd N, $t = 0$ is the $\lfloor \frac{N}{2} \rfloor$th Chebyshev node, so the initial condition can be enforced exactly and the remaining $N-1$ entries of $\vec{u}$ can be obtained by solving the resulting $(N-1) \times (N-1)$ linear system.

B.2 Reducing in-context operator learning to least squares

For the in-context learning problem, we are given pairs of input-output examples for an unknown operator:

$$\{(\vec{c}_1, u_1(t_j)), \ldots, (\vec{c}_n, u_n(t_j))\},$$

for some Chebyshev node t_j. Assuming the underlying differential operator comes from the linear class (Equation 3), we can apply the pseudospectral reduction from Appendix B.1 to reduce each ODE solve to a shared linear function. Then the ICL problem reduces to solving the linear system

$$\begin{bmatrix} (\vec{c}^{(1)}, u_0^{(1)})^T \\ \vdots \\ (\vec{c}^{(k)}, u_0^{(k)})^T \end{bmatrix} \vec{w_j} + \vec{b}^*[j] \vec{1} = \begin{bmatrix} u^{(1)}(t_j) \\ \vdots \\ u^{(k)}(t_j) \end{bmatrix}.$$

B.3 Generalization to interpolated operators

For arbitrary $t \in [-1, 1]$ that are not Chebyshev nodes, note that we can still write $u(t)$ (to high precision) as a linear combination of the $u(t_j)$'s by Lagrange interpolation (Appendix A). Since each $u(t_j)$ is obtainable as the solution to an approximate in-context least squares problem, we expect to represent the map $(c, u_0) \to u(t)$ as an approximate least squares problem for *any* t.

C Experiments

C.1 Experimental details

Setup. Following recent work [5], we use a decoder-only GPT2 Transformer as our sequence model with 12 layers, 8 heads, and 256-dimensional embeddings (10M parameters). Inputs and outputs are represented as 40-dimensional vectors (padded with zeros as necessary). Linear layers are used to map between the input/output dimension and the Transformer embedding space. We train the Transformer according to the objective (4). We use a batch size of 64 and train for 500k steps. The Adam optimizer is used with a learning rate of 10^{-4}.

Data. Our data is a distribution of 1D linear ODEs, a subset of $\mathscr{L}_{linear}$ (3):

$$u'(t) = \alpha_1 c(t) + \alpha_2 u(t) + \alpha_3, \quad t \in [-1, 1], \tag{8}$$

with parameters $\alpha_1 \sim Unif[0.5, 1.5]$, $\alpha_2 \sim Unif[-1, 1]$, $\alpha_3 \sim Unif[-1, 1]$. The initial conditions are sampled as $u(0) = u_0 \sim Unif[-1, 1]$; and c is drawn from a Gaussian process with RBF kernel, $K(x, x') = \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$, with length-scale parameter $\ell = 1$.

ODEs are solved using the pseudospectral method on $N = 41$ Chebyshev nodes over $[-1, 1]$. To obtain solution values at other points in the domain, the Lagrange interpolating polynomial is used. Empirically, we find for smooth functions that this approach typically achieves an error of less than 10^{-10}.

Generalization settings.

- **Additive noise in inputs/outputs.** We evaluate the model's performance when the in-context example inputs and outputs are perturbed by Gaussian noise with different variances.
- **Out-of-distribution forcing functions.** We evaluate how well our model generalizes to different forcing function distributions. Specifically, we vary the length-scale parameter, ℓ, of our Gaussian process kernel, $K(x, x') = \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$, making the forcing functions "bumpier" or "smoother" as ℓ decreases and increases, respectively.
- **Out-of-distribution boundary/initial conditions.** We evaluate the model's robustness to a widened distribution of initial conditions $u(0) = u_0$. We parameterize the distribution width with a scalar σ_{IC}, where $u_0 \sim Unif[-\sigma_{IC}, \sigma_{IC}]$.
- **Out-of-distribution operator parameters.** We evaluate the model's robustness to inputs from OOD operators, of the same functional form as the training operators, but whose parameter distribution is widened. Specifically, we parameterize the width of the ODE operator distribution with a scalar σ_{op}, where $\alpha_1 \sim Unif[1 - 0.5\sigma_{op}, 1 + 0.5\sigma_{op}]$, and $\alpha_2, \alpha_3 \sim Unif[-\sigma_{op}, \sigma_{op}]$.
- **Interpolation.** On our original task, we train the model to predict the solution at a discrete number of domain points $u(t_i)$. Here, we sample the desired t uniformly on the domain $[-1, 1]$, and we observe that the model is able to generalize to continuously-sampled t (a problem similar to one considered previously [9]).
- **Unseen forms of operators.** We evaluate on two other simple 1D ODE distributions with qualitatively different characteristics: a 1st-order ODE *nonlinear* in forcing and solution terms

  $$u' = \alpha_1 cu + \alpha_2 u + \alpha_3$$

  and a *2nd-order* ODE linear in forcing and solution terms simultaneously:

  $$u'' = \alpha c.$$

C.2 Additional experiments

ICOL Transformer vs. least squares Here, we compare the performance of our Transformer-based in-context operator learner against a standard OLS solver for the 1st-order linear ODE class (Equation 8). See Figure 2. Surprisingly, we observe a huge performance gap between the Transformer and OLS as the number of in-context examples increases.

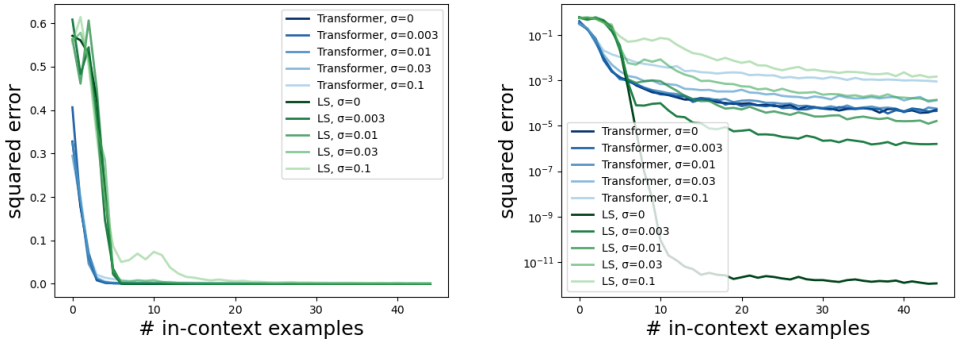

Figure 2: Transformer vs. OLS generalization given in-context example solutions with additive Gaussian noise of varying magnitudes, standard (left) vs. log (right) scaled.

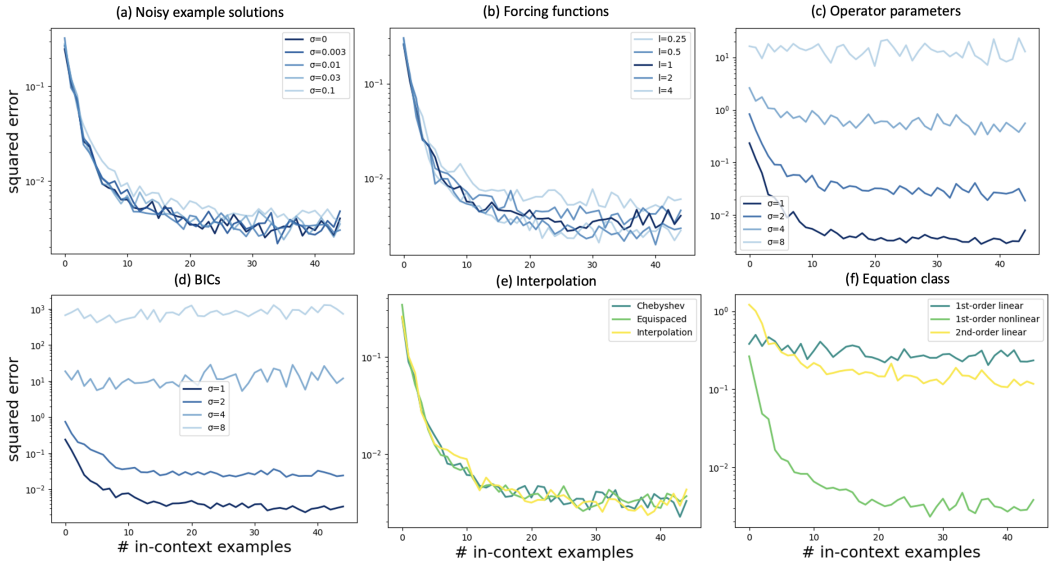

Figure 3: Generalization within (a-d) and across (e-f) operator classes for 1st-order ODE with nonlinear solution-forcing interaction term.

Generalization for other 1D ODEs Here, we evaluate the same within-class and across-class generalization tests for a 1st-order ODE *nonlinear* in forcing and solution terms

$$u' = \alpha_1 cu + \alpha_2 u + \alpha_3$$

and a *2nd-order* ODE linear in forcing and solution terms simultaneously:

$$u'' = \alpha c.$$

Interestingly, although both 1st- and 2nd-order linear ODEs show similar precision, down to 10^{-5} for in-distribution evaluation, the 1st-order non-linear ODEs saturate at a much higher error, around 10^{-2}. See Figure 3 and Figure 4 for a comparison of results.

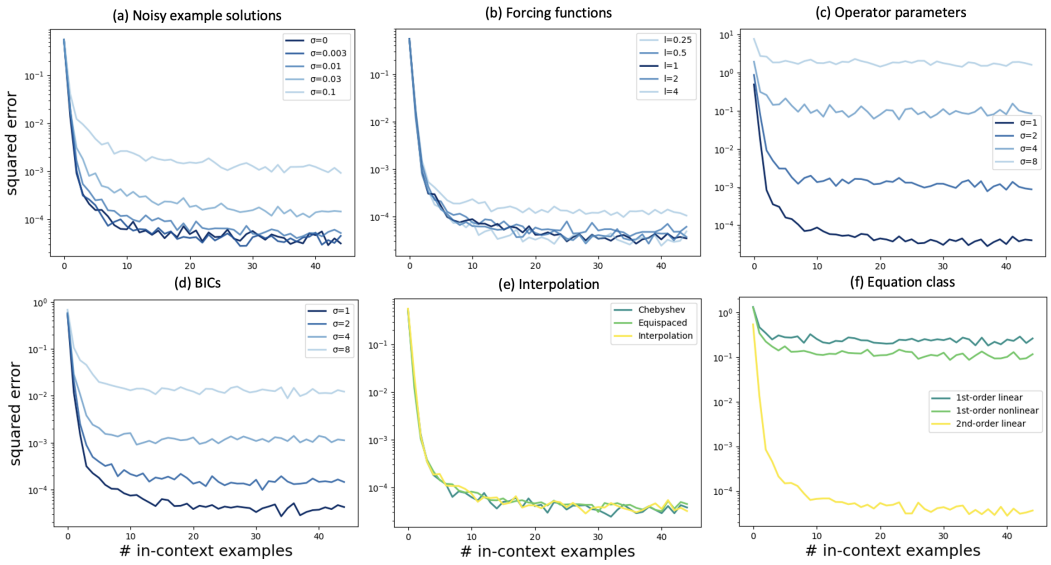

Figure 4: Generalization within (a-d) and across (e-f) operator classes for 2nd-order linear ODE.