Advances in scientific discovery automation

Thesis by
Théo Bourdais

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy



CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2027
Defended August 31st, 2026

© 2027

Théo Bourdais
ORCID: 0009-0007-9439-2349

All rights reserved

ACKNOWLEDGEMENTS

I would like to thank my advisor Houman Owhadi for his mentorship throughout my PhD. I truly enjoyed working with someone whose scientific vision, mathematical skills, and generosity I admire. I am grateful for the many opportunities Houman provided me, from giving me the freedom to explore ideas and develop my voice as a researcher, to funding conference travel, introducing collaborators, and welcoming me into his group. I would like to thank Tuhin Sahai, whom I also consider a mentor today. His insights have helped form my understanding of the research process and how to make an impact. I enjoyed the hours spent trying to launch code on his numerous clusters, and learned from his ability to always find exactly the right problem to solve. I am grateful to both of them for making me a part of their algorithm-discovery research; this has been my most exciting endeavor to date, and I look forward to pursuing the work.

I thank my collaborators, including Xiajin Yang, Ricardo Baptista, Zongren Zou, Bamdad Hosseini, Aras Bacho, Aleksei Sorokin, Yitzchak Shmalo, Leonid Berlyand, Michael Tyrrell, Markos Katsoulakis, Zakaria Baba and many others. I would like to give special thanks to Nicolas Rouquette for pushing my work into the real and messy world, and for the unique experience at JPL. I am grateful to Venkat Chandrasekaran and Eric Mazumdar for serving on my committee and for their help. Finally, I would like to thank Olivier Le Maître and Pietro Congedo for giving me my first taste of research at Polytechnique, and for their continued help since; and Meelis Lootus for his guidance and for helping me come to Caltech.

I would like to thank the friends I made in Pasadena: Erik Jahn for sharing an office and many conversations, scientific or otherwise; Edoardo Calvello for making life fun; Matthieu Darcy for the debates; and Alessio, Alessandro, Elie, Isabel, Juliette, Ludovico, Pau, Solene A., and Thomas for the great memories. Another *merci* to Baptiste, Delphine, and Victor for their unwavering belief that I will achieve anything I attempt and then some; to Alex and Emeline for their guest rooms; and to Alec, Arthur, Constantin, Lily, and Tanguy. I would not have come this far without the support of my family, and I am deeply thankful for them all, especially my mother, father, and sister. Despite being far from home, I never felt homesick.

This adventure would never have been possible without Solene, who came with me to build a life in America. I dedicate this thesis to her and thank her for these four years.

ABSTRACT

Much of machine learning for scientific computing approximates a function specified in advance. This thesis investigates how dedicated learning systems can address tasks of increasing complexity and thereby automate larger parts of scientific discovery. Each method exploits structure in the learning method or in the scientific problem to constrain the search and incorporate mathematical knowledge.

In *Function Learning*, the target task is prescribed; we develop minimal-variance aggregation of black-box predictions, operator learning for partial differential equations at machine precision, and pruning of deep networks guided by random matrix theory.

In *Structure Discovery*, the relationships among variables must be recovered from data; we identify functional and stochastic dependency graphs for digital twins with up to one thousand variables while treating randomness, time, and control explicitly.

In *Algorithm Discovery*, we design algorithms by combining computational primitives through a variant of AlphaZero and expand the primitive set through Algorithmic Byte Pair Encoding. The resulting algorithms adapt during execution and outperform state-of-the-art methods on the strongly NP-hard Quadratic Assignment Problem; the framework also produces a new formulation of Grover's algorithm using half as many gates as the standard construction.

PUBLISHED CONTENT AND CONTRIBUTIONS

Zongren Zou, Théo Bourdais, Ricardo Baptista, and Houman Owhadi. Graphical conditional generative modeling for digital twin modeling. *arXiv preprint*, 2026. URL https://arxiv.org/abs/2606.16219.
TB co-developed the structure-discovery and conditional generative framework and co-wrote the manuscript.

Théo Bourdais, Abeynaya Gnanasekaran, Houman Owhadi, and Tuhin Sahai. Discovering algorithms with computational language processing. *Science Advances*, 12(37):eaea4216, 2026. doi: 10.1126/sciadv.aea4216. URL https://doi.org/10.1126/sciadv.aea4216.
TB co-designed the CLP framework, ran the main experiments, and co-wrote the manuscript.

Aras Bacho, Aleksei G. Sorokin, Xianjin Yang, Théo Bourdais, Edoardo Calvello, Matthieu Darcy, Alexander W. Hsu, Bamdad Hosseini, and Houman Owhadi. Operator learning at machine precision. *Journal of Computational Physics*, 566:115240, 2026. doi: 10.1016/j.jcp.2026.115240. URL https://doi.org/10.1016/j.jcp.2026.115240.
TB contributed to method development and manuscript writing and editing.

Leonid Berlyand, Théo Bourdais, Houman Owhadi, and Yitzchak Shmalo. Pruning deep neural networks via a combination of the Marchenko–Pastur distribution and regularization. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2503.01922.
TB co-developed the pruning methodology and theory, contributed to experiments, and co-wrote the manuscript.

Théo Bourdais and Houman Owhadi. Minimal variance model aggregation: A principled, non-intrusive, and versatile integration of black box models. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=grM2Yv49cI.
TB was lead author: developed the aggregation framework and theory, ran the experiments, and wrote the manuscript.

Théo Bourdais, Pau Batlle, Xianjin Yang, Ricardo Baptista, Nicolas Rouquette, and Houman Owhadi. Codiscovering graphical structure and functional relationships within data: A Gaussian process framework for connecting the dots. *Proceedings of the National Academy of Sciences*, 121(32):e2403449121, 2024. doi: 10.1073/pnas.2403449121. URL https://doi.org/10.1073/pnas.2403449121.
TB was lead author: co-developed the CHD methodology and software, ran experiments, and co-wrote the manuscript.

Houman Owhadi, Nicolas F. Rouquette, Pau Batlle, Théo Bourdais, Xianjin Yang, and Ricardo Baptista. Systems and methods for co-discovering graphical structure and functional relationships within data. US Patent Application 18/962,595, 2025. URL https://patents.google.com/patent/US20250173388A1/en.
Patent covering the graphical structure-discovery method included in this thesis. TB contributed to the underlying methodology.

TABLE OF CONTENTS

Acknowledgements . . . iii
Abstract . . . iv
Published Content and Contributions . . . v
Table of Contents . . . vi
List of Illustrations . . . xi
List of Tables . . . xix
Chapter I: Introduction . . . 1
1.1 Machine Learning in Scientific Computing . . . 1
1.1.1 Contributions in Function Learning . . . 2
1.2 Toward Automating Scientific Discovery . . . 2
1.2.1 Discovering functional dependencies from data . . . 3
1.2.2 Discovering stochastic dependencies from data . . . 4
1.2.3 Discovering algorithms using computational language processing . . . 4
1.3 LLMs and Scientific Discovery . . . 5
1.4 Extracting Structure from Problems . . . 6
1.5 Outline . . . 7

I Function Learning **8**
Chapter II: Minimal Variance Model Aggregation . . . 9
2.1 Introduction . . . 9
2.2 The minimal error aggregation . . . 11
2.2.1 Best linear aggregate when model correlations are known . . . 11
2.2.2 Data-driven aggregation . . . 13
2.3 Aggregation through error estimation: the Minimal Variance Aggregate (MVA) . . . 14
2.3.1 Modelling the error . . . 14
2.3.2 A theorem on the superiority of MEVA over MEEA . . . 15
2.3.3 Computing the aggregation: the Minimal Empirical Variance Aggregate (MEVA) . . . 16
2.4 Experiments . . . 17
2.4.1 Sanity check: aggregation on the Boston housing dataset . . . 17
2.4.2 Aggregation of PDE solvers . . . 18
2.4.3 The Laplace equation . . . 21
2.4.4 Burgers' equation . . . 22
2.5 Discussion . . . 23
Chapter III: Operator Learning at Machine Precision . . . 25
3.1 Introduction . . . 25

3.1.1 Summary of contributions . . . 26
3.1.2 Brief review of the relevant literature . . . 28
3.1.3 Outline . . . 30
3.2 Operator Learning . . . 31
3.2.1 The operator learning problem . . . 31
3.2.2 The Newton–Kantorovich method . . . 32
3.2.3 The Newton–Kantorovich method with Tikhonov regularization . . . 32
3.2.4 CHONKNORIS . . . 34
3.2.5 FONKNORIS . . . 35
3.3 Numerical Experiments . . . 38
3.3.1 Benchmarking Summary . . . 39
3.3.2 Forward Problems . . . 40
3.3.3 Foundation Model - FONKNORIS . . . 45
3.3.4 Inverse Problems . . . 47
3.4 Theoretical results . . . 53
3.4.1 Analytic Setting . . . 53
3.5 Conclusion and Outlook . . . 57
3.6 Code and data availability . . . 58
3.7 Acknowledgments. . . . 58
Chapter IV: Pruning Deep Neural Networks via Marchenko–Pastur Theory and Regularization . . . 60
4.1 Introduction . . . 60
4.2 Randomness in Deep Neural Networks . . . 62
4.2.1 Introduction to Deep Neural Networks . . . 62
4.2.2 The MP Distribution in Machine Learning contexts . . . 64
4.2.3 Reduction of randomness in DNN weights via MP Distribution 65
4.3 Pruning Visual Transformers (ViTs) . . . 66
4.3.1 Pruning strategy based on Random Matrix Theory . . . 66
4.3.2 Pruning procedure overview . . . 69
4.3.3 Results for the visual transformer Vit_Base_Patch16_224 Model . . . 70
4.3.4 Pruning the visual transformer Vit_large_Patch16_224 Model 70
4.3.5 Pruning results for ViT-B16/224 and ViT-L16/224 on ImageNet . . . 71
4.3.6 RMT-based pruning of DeiT . . . 71
4.4 Main theoretical results . . . 72
4.4.1 Assumptions for main theorems: Gaussian case . . . 72
4.4.2 A theorem on the asymptotic magnitude of the noise . . . 74
4.4.3 A theorem on how pruning randomness reduces loss . . . 76
4.4.4 Key lemmas: how removing randomness from DNN weight layers affects the output, loss and accuracy of a DNN . . . 78
4.5 Generalized Main Theoretical Results . . . 78
4.5.1 Assumptions for Generalized Main Theorems . . . 78
4.5.2 Main Theoretical results for RMT-based decrease in loss . . 81

4.5.3 Key Lemmas: how removing randomness from DNN weight layers affects the output, loss and accuracy of a DNN 83

II Structure Discovery **85**

Chapter V: Co-discovering Graphical Structure and Functional Relationships Within Data 86
5.1 Introduction 86
5.1.1 The three levels of complexity of function approximation. . . 86
5.1.2 Generalizing Gaussian Process methods. 87
5.1.3 The scope of Type 1, 2 and 3 problems. 87
5.2 Overview of the proposed approach for Type 3 problems. 89
5.2.1 A signal-to-noise ratio to decide whether or not a node has ancestors. 89
5.2.2 Selecting the signal kernel K_s 90
5.2.3 Pruning ancestors based on signal-to-noise ratio. 90
5.2.4 Computational complexity 91
5.3 Results 92
5.3.1 The Fermi-Pasta-Ulam-Tsingou system 92
5.3.2 The Google Covid 19 open data. 93
5.3.3 Chemical reaction network. 94
5.3.4 Algebraic equations. 95
5.3.5 Cell signaling network 95
5.3.6 A large-scale chemical reaction network: the BCR reaction benchmark 97
5.4 Discussions 99
5.4.1 Limitations 99
5.4.2 Conclusions 100
5.4.3 Data availability 100
5.4.4 Code availability 101
5.4.5 Online content 101
5.4.6 Acknowledgements 101

Chapter VI: Graphical Conditional Generative Modeling for Digital Twin Modeling 102
6.1 Introduction 102
6.2 Results 105
6.2.1 Problem formulation 105
6.2.2 An illustrative example 106
6.2.3 Conditional-law modeling recovers stochastic graph structure for improved accuracy 107
6.2.4 Sparse state-control structure preserves downstream control performance. 108
6.2.5 Discovering state and control dependence in the Lunar Lander control problem. 109

6.2.6 Delayed distributional dependence changes across economic regimes . . . 110
6.3 Discussion . . . 111
6.4 Methods . . . 112
6.4.1 Distillation and stochastic ancestor discovery . . . 113
6.4.2 Digital twin modeling and treatment of dynamical systems . 115

III Algorithm Discovery 117
Chapter VII: Discovering Algorithms with Computational Language Processing 118
7.1 Introduction . . . 118
7.2 Results . . . 119
7.2.1 Results on the Quadratic Assignment Problem (QAP). . . . 119
7.2.2 Results on the Quantum Search Problem. . . . 122
7.2.3 Results on the Quantum Approximate Optimization Algorithm (QAOA). . . . 123
7.3 Discussion . . . 124
7.4 Materials and methods . . . 124
7.4.1 Computational Language Processing. . . . 124
7.4.2 Ensemble Monte Carlo tree search with reinforcement learning. . . . 125
7.4.3 Algorithmic byte-pair encoding and learning higher-complexity tokens. . . . 126
7.4.4 Primitives, tokens and Algorithmic Byte-Pair Encoding for the QAP. . . . 128
7.5 Acknowledgments . . . 129
7.6 Ensemble Monte Carlo tree search (MCTS) and reinforcement learning (RL) for algorithm discovery . . . 129
7.7 CLP applied to the QAP . . . 131
7.7.1 Discovering new strategies with low-level primitives and CLP 131
7.7.2 Reaching state-of-the-art performance using high-level primitives and CLP . . . 135
7.7.3 Using CLP to manually create algorithms . . . 137
7.8 Discovering a new quantum circuit for Grover's algorithm with lower depth using CLP . . . 138
7.9 CLP-based QAOA results . . . 140

Bibliography . . . 144

IV Appendices 182
Appendix A: Appendix to Chapter 2: Minimal Variance Model Aggregation . 183
A.1 Theorem on the stability of the MEVA . . . 183
A.2 Aggregating PDE solvers in the Gaussian setting . . . 190
A.3 Another Pathological example: Unknown region . . . 192

A.4 Non-independent model errors . . . 193
A.5 An alternative, direct Minimal Empirical Variance Aggregation loss . 195
A.6 Adding a bias correction . . . 195
A.7 Aggregation using vector-valued Gaussian processes and matrix-valued kernels . . . 196
A.8 Minimizing the loss in section A.7 . . . 197
Appendix B: Appendix to Chapter 3: Operator Learning at Machine Precision 202
B.1 Additional Benchmarks . . . 202
B.2 Nested Kriging . . . 202
B.3 Choice of ordering in the Cholesky Factorization . . . 206
B.4 Discretizations and Cholesky factorization . . . 208
B.5 Choice of the step size α and relaxation λ . . . 208
B.6 Approximation of the Fréchet derivative . . . 209
B.7 Connection to Attention Mechanism . . . 211
B.8 Proofs . . . 212
Appendix C: Appendix to Chapter 4: Network Pruning via Marchenko–Pastur Theory . . . 217
C.1 Other Numerical Results . . . 217
C.2 A simple illustration of the relationship between RMT-based pruning and regularization . . . 220
C.3 Some known results on perturbation of matrices . . . 240
C.4 Algorithms for RMT-Based pruning of DNN . . . 246
Appendix D: Supplementary Information for Chapter 5: Computational Hypergraph Discovery . . . 253
D.1 Additional details on our proposed approach. . . . 253
D.2 Algorithm Overview for Type 3 problems: An Informal Summary . . 256
D.3 Type 2 problems: Formal description and GP-based Computational Graph Completion . . . 258
D.4 Hardness and well-posed formulation of Type 3 problems. . . . 260
D.5 A Gaussian Process method for Type 3 problems . . . 264
D.6 Algorithm pseudocode. . . . 276
D.7 Analysis of the signal-to-noise ratio test. . . . 278
D.8 Supplementary information on examples. . . . 283
Appendix E: Appendix to Chapter 6: Graphical Conditional Generative Modeling . . . 288
E.1 Conditional generative modeling via flow matching . . . 288
E.2 Algorithm for stochastic ancestor discovery . . . 288
E.3 Additional details . . . 289
E.4 Additional examples . . . 305
Appendix F: Supplementary Materials for Chapter 7: QAP Background and Primitives . . . 314
F.1 Why is the QAP hard? . . . 314

LIST OF ILLUSTRATIONS

Number *Page*

2.1 Pathological example (sec. 2.2.2) 14

2.2 Comparisons of the performance of the different models on the Boston housing dataset. Red bars are the performance of models trained using the training set and used in the aggregation. Blue bars show models trained on the training and validation set (*train+val*) to get a fair comparison with the aggregations. The aggregation does not use the models trained on *train+val*. 19

2.3 log MSE of the different methods for: (2.3a) the Laplace equation; (2.3b) Burgers' equation. Samples are sorted by the error of the aggregate. 21

3.1 CHONKNORIS. (a) An initial guess v_0 for the true solution $v = \mathcal{G}(u)$ is iteratively refined by adding a correction term. (b) Each iteration consists of two steps: First, compute the correction term $r = \left[\frac{\delta \mathcal{F}}{\delta v}\right]^* \mathcal{F}$ and then $\delta v = -\left(\widehat{\mathcal{R}}\widehat{\mathcal{R}}^* r\right)(u, v_k)$, where $\widehat{\mathcal{R}}$ is a learned surrogate for the Cholesky factors of $(\frac{\delta \mathcal{F}}{\delta v}\frac{\delta \mathcal{F}}{\delta v}^T + \lambda I)^{-1}$, $\mathcal{F}$ is the forward map with its Fréchet derivative $\frac{\delta \mathcal{F}}{\delta v}$, v_k is the current approximation of the desired function $v = G(u)$, and u is the input for which we seek the solution. Next, update the current approximation via $v_{k+1} = v_k + \delta v$. 37

3.2 Forward problems. (a) Results for the nonlinear elliptic PDE problem. Quantiles of $10\% - 90\%$ are shown across test realizations. Our CHONKNORIS method is able to achieve machine precision accuracy in around 10 iterations. (b) Results for Burgers' equation. CHONKNORIS was able to achieve machine precision error in recovering the discretized solution which contained shocks. (c) Results for the Darcy flow PDE: (c1) shows that more challenging realizations require more CHONKNORIS iterations. (c4) shows a single realization of the random coefficient with the corresponding solution in (c2). (c3) shows the fixed forcing term. 44

3.3 Quantiles of predictions of FONKNORIS for 100 realizations of initial conditions, external forces, and conductivities using a mixture of experts consisting of GPs for nonlinear elliptic, nonlinear darcy flow, and Burgers' equation and testing it for withheld Sine–Gordon and Klein–Gordon equations. 45
3.4 Inverse Problems. (a) Results for the seismic imaging problem, showing the iterative solutions and the relative L^2 error across CHONKNORIS iteration. (b) For the seismic imaging problem, evolution of the adaptive regularization term in the Newton–Kantorovich iterations for different resolutions, and comparison of the relative L^2 error between the Newton–Kantorovich method and our CHONKNORIS method. (c) Results for the Calderón problem. (d) Results for the inverse wave scattering problem. 52
4.1 The top 1 accuracy and top-5 accuracy of ViT-base vs. percentage of parameters kept for pruning through RMT-based sparsification and no fine-tuning. The accuracy of the DNN is given as a percentage above the data points. 70
4.2 The top 1 accuracy and top-5 accuracy of ViT-large model vs. percentage of parameters kept for pruning through RMT-based sparsification and no fine-tuning. The accuracy of the DNN is given as a percentage above the data points. 71
4.3 Comparison of pruning results between our method and CP-ViT [316] 72
5.1 The three levels of complexity of function approximation. 87
5.2 Ancestors identification in Type 3 problem. 88
5.3 (a-d) The Fermi-Pasta-Ulam-Tsingou system. (e-k) The Google Covid 19 open data. 92
5.4 (a-c) Chemical reaction network. (d-g) Algebraic equations. (h-j) Cell signaling network. 94
5.5 (a) Cell signaling network comparisons. (b-h) The BCR reaction benchmark. 97
6.1 (a): Daily gold returns (next day) and silver returns (data from Google Finance) to illustrate stochastic relation in financial data. (b): Equation discovery with missing variables to illustrate the ancestor hidden in the distribution rather than the conditional mean. (c): A schematic of the proposed two-stage framework for digital twin modeling. . . . 104

6.2 Stochastic Lorenz-63 system: (a)–(b) pruning process over ten independent runs, (c) recovered computational graph, and (d) generalization of the full and pruned models under parameter-induced distribution shift by training at $\rho = 28$ and testing at different values of ρ_{test}. 106
6.3 Control examples. (a)–(b) Missing-variable pendulum: memory discovery and policy transfer. (c)–(d) Stochastic heat equation: boundary discovery and MPC performance. 108
6.4 Robotics control examples and comparisons. (a)–(b) Lunar Lander: discovered thrust-stage dependence structure and policy-transfer performance. (c)–(d) 7-DOF reacher: discovered pruned variables and PETS performance. 109
6.5 Delayed dependence analysis in economic markets. (a) Aligned WTI crude oil price and GPR index data. (b)–(e) First-iteration relative contribution scores of delayed GPR variables across time periods; the highlighted bar denotes the variable pruned last. 110
7.1 **Overview of Algorithm Tokens and MCTS+RL Search Tree:** (**A**) We conceptualize an algorithm as a chain of elementary computational steps (primitives/tokens), chained with a precise grammar/syntax. (**B**) Starting with elementary tokens, we discover new tokens and algorithms with ensemble Monte Carlo tree search (MCTS) + reinforcement learning (RL) as a babbling mechanism to search over the tree of all possible strings of tokens. Edges represent primitives/tokens and nodes the ensemble of computational state variables sampled from the string of (possibly stochastic) tokens leading that node. 119
7.2 **Optimality gap for different methods:** the figure displays median, 10th and 90th quantiles of optimality gap. Computational Language Processing (CLP) beats all other methods, including Simulated Annealing (SA) and Branch and Bound (B&B), on all instances, and achieves global optimum on all tested Composite Quadratic Assignment Problem (CQAP) instances 121
7.3 **Comparison of Grover's algorithm implementations**: standard (left, U_D is the standard diffusion operator) and optimized (right, with simplified diffusion operator U'_D). 123

7.4 **Navigating the complexity of algorithmic tokens, complexification and subdivision using Monte Carlo Tree Search (MCTS) + Reinforcement Learning (RL)**: (**A**) Algorithmic Byte-Pair encoding to increase vocabulary by creating new tokens from old ones (**B**) Low-level primitives (**C**) Fast Approximate Quadratic assignment problem (FAQ) algorithm rediscovery as a chain of three primitives (gradient, linear sum assignment, convex combination of doubly stochastic relaxation). See supplementary Sec. F.1.3 for low-level primitive definitions. 126

7.5 **Emergence of dominant strategies as difficulty increases**: (left-axis) Frequencies of occurrence of dominant strategy and finding the global optimum. (right-axis) Average optimality ratio, $(\mathcal{L} - \min \mathcal{L})/\min \mathcal{L}$. (x-axis) A-BPE rounds occurring every 10 iterations are marked by a vertical dotted line. CLP discovers increasingly more complex and effective algorithms, with identifiable dominant strategies and instance-specific adaptations, to reach a high success rate. Strategies include Frank-Wolfe (FW), 2-city flip (2FLIP), 2-city optimal (2OPT), gradient (GRAD), Linear Sum Assignment (LSA). Details in 7.7.1 127

7.6 **Optimal step sizes for the Fast Approximate Quadratic assignment problem (FAQ) algorithm generated by our Computational Language Processing (CLP) approach compared to the standard step size schedule implemented in `SciPy`**. CLP discovers a cyclic rate (blue) with interleaving of Linear Sum assignments (LS) and Linear Sum assignments on the Gradient (LSG), and reaches global optimum unlike the standard line search-based approach used in `SciPy` (red) 135

7.7 **Comparison of workflows for generating quantum circuits for Quantum Approximate Optimization Algorithm (QAOA):** (**A**) ADAPT-QAOA, (**B**) our Computational Language Processing (CLP)-based approach. 143

A.1 (a) Real solution of the PDE (b) One of the models aggregated (c) Uniform average of all models (d) Proposed aggregate (2.3) 192

A.2 Pathological example (sec. 2.2.2) 193

A.3 A pair (f, u) s.t. $-\Delta u = f$ and $u(\partial\Omega) = 0$, sampled using (2.19) . . . 198

A.4 Example of aggregation for the Laplace equation (section 2.4.3). Line 1: Outputs of the different models for a given input f. Line 2: Prediction errors for each model. Line 3: Values of α. Line 4: Aggregate and its error 200
A.5 Example of aggregation for Burgers' equation (section 2.4.4). Line 1: Outputs of the different models for a given input f. Line 2: Prediction errors for each model. Line 3: Corresponding values of α. Line 4: Aggregate and its error. Note that the explicit, implicit, and Lax-Wendroff methods all diverged, while the spectral method exhibited spurious oscillations. In contrast, the aggregated result avoids these issues. 201
B.1 Comparison of L_2 relative errors versus inference time for the seismic imaging full waveform inversion problem in Section 3.3.4.3. 203
B.2 Given the relaxation λ_n and step size α_n at iteration n, the next parameters $(\lambda_{n+1}, \alpha_{n+1})$ are chosen from the above 9 options to minimize the loss. The deflation factors $1/2$ and inflation factors 2 can be replaced by more general factors, see the discussion in Appendix B.5. 210
C.1 DNN performance vs. sparsification 224
C.2 Accuracy and loss as we add noise to the weight layers of the DNN. . 225
C.3 Example of data generated from the problem statement on one axis. The blue line is the function y_0, and the blue dots are the data points. 226
C.4 Cumulative distribution of singular values of the optimal weight matrix 226
C.5 Cumulative distribution of singular values of the L^2 regularised weight matrix 227
C.6 Cumulative distribution of singular values of the L^1 regularised weight matrix 227
C.7 Cumulative distribution of singular values of the weight matrix prior to pruning 228
C.8 Plot of $a(N)$ vs. the width N of the middle layer $\mathbf{W_2}$. This plot shows how the complexity-related metric $a(N)$ behaves as the width N is increased, reflecting changes in the norms of the weight matrices and their dependence on N. 235
C.9 Plot of test accuracy vs. the width N of the middle layer $\mathbf{W_2}$. This plot demonstrates how the performance of the DNN changes with increasing N, showing the relationship between network width and generalization ability on the Fashion MNIST dataset. 236

C.10 Plot of $b(N)$ vs. the width N of the middle layer $\mathbf{W_2}$. This plot shows how the metric $b(N)$ behaves as the size of the middle layer increases, incorporating a stronger dependence on N^2. 237
C.11 Plot of test accuracy vs. the width N of the middle layer $\mathbf{W_2}$. This plot demonstrates how the network's accuracy changes as N increases, with weight matrices initialized using $1/N^2$. 237
C.12 In blue we have the ESD of X, in red the Marchenko-Pastur distribution which best fits the ESD based on the BEMA algorithm. 247
C.13 How λ_+ depends on the hyperparameters α and β. 247
C.14 The ESD of X_l and its best fit MP distribution 250
C.15 Original DNN with two hidden layers, each with 10 nodes (total 100 parameters), is transformed into a new DNN with three hidden layers. The first layer has 10 nodes, the second layer has 2 nodes (keeping only two singular values in the SVD), and the third layer has 10 nodes, resulting in a total of 20 parameters. 250
C.16 Best fit MP distribution (on the left) for $\alpha = .25$ and $\beta = .8$ for a layer of the base ViT model. Here the MP fit error is .74 while the percentage of singular values smaller than $\sqrt{\lambda_+}$ is 73%. For more on the right figure, see Section C.2. 251
C.17 Best fit MP distribution (on the left) for $\alpha = .25$ and $\beta = .8$ for a layer of the base ViT model. Here the MP fit error is .01 while the percentage of singular values smaller than $\sqrt{\lambda_+}$ is 99.73%. For more on the right figure, see Section C.2. 252
C.18 In the left figure, we have α vs LRM and on the right we have α vs MP fit metric. This is for the base ViT model studied in 4.1. 252
D.1 Histogram of the eigenvalues of D_γ=(D.2) for $\gamma = 10^{-2}$ (good choice) and $\gamma = 10^{-6}$ (bad choice). 254
D.2 Formal description of Type 2 problems. 257
D.3 (a) Electric circuit. (b) Resistance, capacitance, and inductances are nonlinear functions of currents and voltages (c) Measurements. (d) Kirchhoff's circuit laws. (e) The computational graph with unknown functions represented as red edges. (f) Recovered functions. 259
D.4 Computational Hypergraph Discovery with three variables 261
D.5 The structure of the hypergraph is identifiable in (a), (b), and non-identifiable in (c). The relationship between variables is implicit in (d). 261

D.6 (a) CHD formulation as a manifold discovery problem and hypergraph representation (b) The hypergraph representation of an affine manifold is equivalent to its Row Echelon Form Reduction. 263
D.7 Feature map generalization 265
D.8 Iterating by removing the least active modes from the signal 274
D.9 Computing the ancestors of the variable $\dot{x}_0$ in the Fermi-Pasta-Ulam-Tsingou problem. (a) Noise-to-Signal Ratio, denoted as $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$, with respect to the number of proposed ancestors, represented by q. Additionally, we include a visualization of the quantiles derived from the Z-test, as described in Section D.7.3. Notably, when there is no signal present, the noise-to-signal ratio is expected to fall within the shaded area with a probability of 0.9. (b) Increments in the Noise-to-Signal Ratio, defined as $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q) - \frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q-1)$, as a function of the number of ancestors, denoted as q. The horizontal axis represents the number of proposed ancestors for $\dot{x}_0$. Determining an appropriate stopping point based solely on absolute noise-to-signal ratio levels can be challenging. In contrast, the increments in the noise-to-signal ratio clearly exhibit a discernible maximum, offering a practical point for decision-making. 275
E.1 Recovered computational graph of the multiscale Lorenz 96 system with missing fast variables: dashed lines represent that Δx_j having x_j as an ancestor is given as a prior knowledge to the method. 291
E.2 Lunar Lander control problem: Controlled flight paths from 1,000 episodes using the SAC agent trained in the surrogate environment with the pruned ancestor sets. 299
E.3 The 7-DOF reacher problem. (a) A schematic illustration of a robotic arm. (b) Performance comparison of the model-based RL problem between probabilistic and deterministic dynamics models, adapted from [66]. Higher rewards indicate better performance. 301
E.4 Stochastic KO system: (a) pruning process and (b) recovered computational graph. 307
E.5 The algebraic system with hidden conditional dependence: (a) conditional generative modeling of $y_2 \mid x_1, x_2, x_3$ (without pruning) and $y_2 \mid x_3$ (with pruning) and (b) pruning process. 307

E.6 The coarse-grained transition dynamics with missing variables: (a) pruning process, (b) discovered graph, (c) generated samples of x at different times from the learned conditional law, and (d) generated trajectories from the learned conditioned law. 310
E.7 The stochastic chemical system: (a) pruning process and (b) discovered graph. 310

LIST OF TABLES

Number *Page*

2.1 Result of Laplace equation experiment (sec. 2.4.3) 23

2.2 Result of Burgers' equation experiment (sec. 2.4.4) 24

3.1 Summary of median relative L^2 losses over multiple forward and inverse problem benchmarks. 40

3.2 Forward and inverse problem parameters. u distrib. is the distribution of random coefficients. FD grid contains the regular grid sizes for the finite difference scheme. $(R_{\text{train}}, R_{\text{val}})$ are the number of training and validation realizations respectively. n_{warm} denotes the number of Newton–Kantorovich iterations used to generate training data along the flow. λ_{flow} is the relaxation used in the NK method, while λ_{train} is the relaxation shown to the Hessian prediction model. CHONKNORIS training is performed with $\Lambda_{\text{train}} = \{\lambda_{\text{flow}}\}$ in the context of Algorithm 1. GP-kernel denotes a Gaussian process with the given kernel, with the invLaplacian kernel given by $5(-\Delta+1/100)^{-2}$ where $-\Delta$ denotes the Laplacian. MLP-nonlinearity is a multi-layer perceptron neural network architecture where the tuple specifies layer sizes, including input and output layers, and the given nonlinearity is applied to all hidden layers. Note that for Burgers' equation, $n_{\text{warm}} = 5$ NK steps were run per time step. 40

3.3 FONKNORIS coefficients. 45

4.1 Comparison with different ViT pruning methods on the ImageNet dataset. The accuracy without finetuning was not mentioned in VTP, PoWER, and HVT but were replicated in [316]. We fine-tune for only 7 epochs on the ImageNet training set. We see that the larger the DNN, the better our method performs, which makes sense from an RMT perspective. This is also reassuring, given that modern AI models are often much larger than these models. 73

7.1 **Comparison of Grover Optimization Strategies** 140

7.2 **Performance Comparison**: CLP-QAOA vs. ADAPT-QAOA and % improvement 142

B.1 Inference time (seconds) across multiple forward and inverse problem benchmarks. Problems with an * were run on a single GPU (NVIDIA A100), with other problems run on the CPUs (Intel(R) Xeon(R) Platinum 8380 CPUs @ 2.30GHz). 203
B.2 Seismic imaging full waveform inversion (FWI) dataset generation and model fitting times at different resolutions. We record the time per Newton–Kantorovich (NK) iteration, the median number of iterations required for convergence, the total NK dataset generation time for a single realization, and the time required to fit a small version of the CHONKNORIS-GP model (including PyTorch compilation time). All times are in seconds. We emphasize the dataset generation times use the NK methods with the full Cholesky decomposition. Compared to the small CHONKNORIS-GP models fit here, the full CHONKNORIS-GP models we deploy in inference use between two and three times as much data and between five and ten times as many GP optimization iterations depending on the resolution. All timings in this table were carried out on a single NVIDIA A100 GPU. 203
C.1 Performance of fully connected DNN at final epoch for various training strategies. 219
C.2 Performance of fully connected DNN at final epoch for various training strategies. 219
C.3 hyperparameter values 220
C.4 Mean Squared Errors of Different Models for the simple regression task 228
E.1 Approximation quality of the GP surrogate for the generative model of $\Delta x_{1,n} := x_{1,n+1} - x_{1,n}$. *Top row*: the regression mean-squared error (MSE) on validation distilled samples generated from the learned conditional sampler. *Bottom row*: Wasserstein-1 distance to the exact data-generating process, evaluated on 1,000 test inputs with 1,000 surrogate samples drawn per test point. For reference, the Wasserstein-1 distance of the trained flow matching model to the exact data-generating process is 3.42e-2±2.55e-2. 291
E.2 The W_1 distance in average between the learned generative update model and the exact conditional distribution. We randomly select $1,000$ test data points. For each one of them, W_1-distance is estimated based on $10,000$ samples. 307

F.1 **Relative gap** $((\mathcal{L} - \min \mathcal{L})/\min \mathcal{L}$**, in %) on QAPLIB Benchmark Instances** against Gurobi, Branch and Bound (BB) and Simulated Annealing (SA) 318

Chapter 1

INTRODUCTION

Scientific computing combines mathematical modeling with computational methods to tackle problems in science and engineering, thus supporting scientific discovery. Here, we refer to scientific discovery as producing new relationships, models, or procedures from observations and prior knowledge. Much of machine learning in science addresses the narrower task of approximating a well-identified function from data under a predefined success criterion. Yet machine learning can also determine what to model or how to solve a problem, allowing it to address more open-ended parts of the scientific process.

In this thesis, we develop machine learning methods for scientific computing that automate tasks of increasing complexity: Part I improves the learning of prescribed functions and operators, Part II automates the identification of functional and stochastic relationships between variables, and Part III automates the design of algorithms. Each stage expands the computational toolbox by asking the learner to search over a richer object: first a function, then a structure, and finally an algorithm. Throughout this work, structure in the learning method or in the scientific problem restricts the search space and allows mathematical knowledge to be incorporated directly.

Large Language Models (LLMs), which rose to prominence over the course of this thesis (2022–2026), provide a contrasting route based on general-purpose models rather than dedicated systems. Their rapid progress makes this alternative essential to consider in a thesis on automating scientific discovery. We therefore first examine the role of machine learning in scientific computing and its progression toward more open-ended tasks, then assess the LLM-based approach, and finally return to the problem structure that connects the contributions of this thesis.

1.1 Machine Learning in Scientific Computing

Using data to support research has long been a staple of the scientific method. Kepler's first law, formulated by fitting an ellipse to Tycho Brahe's observations of Mars, illustrates how approximating a function from data can contribute to discovery [159]. Modern machine learning scales this pattern: given suitable data,

objectives, and computational resources, learned systems can outperform manually designed pipelines on difficult tasks. Examples include computer vision [6, 164], speech recognition and machine translation [179], and strategic game playing, as exemplified by AlphaGo [308].

Machine learning now contributes across the sciences, an impact recognized by the 2024 Nobel Prizes in Physics and Chemistry [229, 230]. Many of these applications learn prescribed mappings, including PDE solution operators [56, 173], protein sequence-to-structure maps with AlphaFold [154], weather forecasts with FourCastNet [177], and predictive models from biological and medical data [15, 206, 325, 351, 379].

1.1.1 Contributions in Function Learning

Part I (*Function Learning*) focuses on this setting, using primarily Gaussian processes and neural networks [179, 273]. Chapter 2 develops a robust method for combining predictions from multiple black-box models without assumptions on the models. We train an aggregation model to predict the error each model will make on an input, and combine the models' predictions with a weighted average inversely proportional to the predicted errors. We compare to the straightforward aggregation that minimizes error and show that it suffers from catastrophic failures in adversarial cases, and prove that our method enjoys a faster convergence rate for fixed inputs. Finally, in experiments on PDEs, our method combines traditional numerical solvers and machine learning methods and reaches a best-of-both-worlds regime: it outperforms all aggregated models and minimal-error aggregation by an order of magnitude in mean squared error.

Chapter 3 introduces an operator-learning method that reaches machine precision on PDE problems. Chapter 4 uses random matrix theory to identify and prune uninformative components of neural-network weights. Together, these contributions improve the robustness, accuracy, or efficiency of learning while the target task remains fixed. By contrast, governing-equation discovery requires the learner to identify the equation itself, whether through sparse regression, as in SINDy [47], or through kernel methods that jointly learn equations and their solution operators [152].

1.2 Toward Automating Scientific Discovery

Automating scientific discovery is challenging: the problems are domain-specific, often poorly specified, and located at the boundary of existing knowledge; represen-

tative data may be scarce, past performance may be a weak guide, and success may be difficult to define in advance [102, 166, 211, 248, 380]. These difficulties make the division of work between the researcher and the system central. Human-factors research treats automation as a division of functions between humans and machines [276, 305]. Under this definition, Parts II and III automate the recovery and analysis of relationships and the search for algorithms, respectively, while the rest of scientific discovery remains under human direction. Relative to Function Learning, these parts transfer more pattern recognition and trial and error from the researcher to the system.

1.2.1 Discovering functional dependencies from data

Recovering structure from data is already central to causal discovery [259, 319]. Part II (*Structure Discovery*) considers functional and stochastic dependencies rather than causal relationships. Digital-twin modeling motivated this work, as did the operator-learning contributions in Part I and the author's supply-chain emulator [376]. High-fidelity models are often used as digital twins, but they can become too complex to build, validate, interpret, or stress-test in real-world systems. Here, our objective is to construct from data an interpretable reduced model that retains the main behavior of the full system, while reducing the dimension of the problem and removing spurious correlations between variables.

Chapter 5 recovers the sparsest graph of functional dependency between variables that accurately represents the data. Each variable is modeled as a function of the others, and we use Gaussian processes (GPs) to approximate that function. We then exploit the variance-decomposition property of GPs [243] to measure nonlinear sensitivity and prune insignificant variables. The method integrates regularization, kernel selection, and ancestor pruning; while certain components are already known, their orchestration into a unified system unlocks the new capability of automatically discovering structure from data.

The resulting model supports prediction, control, anomaly detection, and uncertainty quantification. When there is a strong prior and a relationship belongs to the class represented by the GP's kernel, the method can recover its equation; otherwise, it provides an approximation. In this restricted sense, the procedure automates part of the inference illustrated by Kepler's construction of an orbital law from observations. In our experiments, linear and quadratic kernels recovered exact equations for algebraic equations and chemical reactions. Under uninformative

priors, we recover known dependencies for the Fermi–Pasta–Ulam–Tsingou problem and produce consistent results with real-world COVID and gene-activation data. Although we demonstrated scaling to 1,000 variables, practical digital twins may be orders of magnitude larger and operate in uncertain environments, aspects this first work did not fully address.

1.2.2 Discovering stochastic dependencies from data

Chapter 6 addresses these limitations by identifying a reduced set of variables of interest to track and treating missing variables, environmental perturbations, and memory effects as sources of randomness. The method recovers the transition law of the variables of interest as a sparse graph of conditional probability distributions, which enables the recovery of dependencies hidden in variance, tail behavior, or other features invisible to conditional-mean modeling. This recovery proceeds in two steps: we first use stochastic interpolants [5], a flexible class of conditional generative models, to approximate the full conditional law of the target variables. We then build a Gaussian process surrogate of the generative model, which allows us to leverage the nonlinear sensitivity analysis described in Section 1.2.1 to identify which variables influence the conditional distribution.

Using the reduced graph, we demonstrate accurate recovery of known dependencies in SDEs and PDE control problems, discover memory variables in partially observed dynamics, and produce sparse digital twins whose downstream control performance matches full-variable models in pendulum, heat-equation, Lunar Lander, and robotic-arm benchmarks.

Together, the two chapters of Part II automate part of the analysis that turns observations into functional or stochastic relationships.

1.2.3 Discovering algorithms using computational language processing

Automated search has also been used to design computational systems, notably neural-network architectures [29]. Part III (*Algorithm Discovery*) moves from discovering relationships to searching directly for algorithms. Algorithms consist of sequences of computational steps and often share common subprocedures. We represent these subprocedures as predefined operations, called primitives, and search over their combinations. We use a variant of AlphaZero [308] to automate the trial and error of constructing and evaluating candidate algorithms. Because of the similarity to producing sequences of tokens in Natural Language Processing (NLP), we coin this framework Computational Language Processing (CLP).

The researcher still frames the problem and designs the initial primitives, while the system uses computation and reinforcement learning to combine them. The abstraction level of the primitives is therefore crucial: overly granular primitives make the search space too large, whereas coarse primitives limit its expressivity. We address this tradeoff with a procedure based on Byte Pair Encoding [109] that generates new primitives from existing ones. This procedure explores a hierarchy of complexity adapted to the problem. The system can also generate algorithms during execution, thereby adapting them to individual problem instances.

We test the framework on the Quadratic Assignment Problem (QAP), a strongly NP-hard combinatorial optimization problem, and obtain state-of-the-art results. Because the action space consists of algorithmic operations, it does not grow exponentially with problem size, a pathology common for reinforcement learning methods in combinatorial optimization. The same framework also improves upon ADAPT-QAOA for the Quantum Approximate Optimization Algorithm (QAOA). Finally, the framework discovers a new formulation of Grover's well-known algorithm [126] that uses half as many gates as the standard construction. These results show that substantial parts of algorithm design can be automated while the researcher retains responsibility for defining the problem and the initial computational vocabulary.

1.3 LLMs and Scientific Discovery

LLMs provide another approach to automating scientific discovery. ChatGPT was released two months after the start of this thesis [352]; since then, LLMs have progressed from struggling with basic arithmetic to contributing to the solution of major open problems in mathematics [149, 289]. This rapid progress makes it necessary to consider whether this approach can automate scientific discovery and how it differs from the work presented here.

Three observations support this possibility. First, generative pretrained models showed that language models could act as general conversational agents [45, 232]. Second, grokking showed that prolonged optimization could produce an abrupt transition from memorization to generalization [264]. Third, empirical scaling laws showed predictable improvements as model size, dataset size, and training compute increase [143]. These observations suggest that scaling existing architectures can produce further gains and possibly new capabilities. Reinforcement-learning-based fine-tuning, including reasoning-oriented training, has extended LLM capabilities in

mathematics and coding [80, 128, 232, 358], bringing LLM performance close to the upper tail of human performance on some tasks [16, 129, 329]. Researchers and AI companies now pursue scientific discovery directly with these systems [11, 79, 231]. Recent examples include AlphaEvolve for algorithm design [228] and language-model-based proof formalization [262].

Nevertheless, this approach has important limitations. First, some researchers argue that systems trained predominantly on language have an incomplete physical understanding and that embodied or visual experience is necessary to construct true world models [14, 61, 77, 354]. This concern is supported by the gap between strong results on knowledge-intensive benchmarks and weaker results on novel or visual tasks and long-horizon interactions [106, 160, 311, 357]. A well-known example is Anthropic's Claude spending many hours stuck in the 1996 game *Pokémon Red* under a minimal agent harness [233].

Second, it is unclear how far generalization and grokking can advance scientific discovery [160]. LLMs can manipulate existing theories and solve difficult problems when the task is well-defined. Performance on these tasks does not establish whether they can formulate new conceptual frameworks outside the structure of existing knowledge, or achieve human-level scientific intelligence on open-ended problems.

Finally, these systems consume substantial resources without necessarily providing a corresponding computational advantage on the underlying scientific problem [53, 84, 148, 203, 288, 335].

1.4 Extracting Structure from Problems

LLMs can generate arbitrary text, making the search space for scientific discovery immense. They constrain this search using patterns learned from existing text, which biases their output toward familiar patterns and can reduce its diversity after alignment [370]. The work presented in this thesis follows a different approach inherited from scientific computing: each method uses structure in its learning method or in the scientific problem to restrict what must be learned.

We leverage two types of structure in this dissertation: the structure of the learning method and that of the problem itself. We exploit the first type in Chapter 2 by formulating model combination statistically and proving a convergence result that justifies the method over more direct alternatives. Chapter 4 uses random matrix theory (RMT) to identify properties of the large weight matrices that arise during the training of transformers. Part II uses the variance-decomposition property of Gaus-

sian processes to perform nonlinear sensitivity analysis and recover dependencies among variables.

We exploit the second type in Chapter 3 by constraining operator learning with the Newton–Kantorovich algorithm; the resulting model reproduces the output of this fast-converging method to machine precision. Part III incorporates known QAP operations, including convex and orthogonal relaxations and local searches, as primitives that can be combined into new algorithms. Algorithmic Byte Pair Encoding further exploits repeated subprocedures to explore the hierarchy of complexity within algorithms.

Across these contributions, mathematical structure makes the assumptions explicit, restricts the search to the problem at hand, and incorporates mathematical knowledge directly into the procedure. It also separates what is known from what must be discovered, controlling which prior knowledge enters the search, supporting mathematical guarantees, and allowing new solutions to emerge.

1.5 Outline

This thesis is organized in three parts of increasing scope and autonomy. Part I improves function learning on well-defined computational tasks, Part II discovers functional and stochastic relationships between variables from data, and Part III automates the design of algorithms.

Part I

Function Learning

Chapter 2

MINIMAL VARIANCE MODEL AGGREGATION

Théo Bourdais and Houman Owhadi. Minimal variance model aggregation: A principled, non-intrusive, and versatile integration of black box models. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=grM2Yv49cI.
TB was lead author: developed the aggregation framework and theory, ran the experiments, and wrote the manuscript.

2.1 Introduction

Many challenges in scientific computing and Machine Learning (ML) involve approximating functions using models. These models can vary significantly, ranging from numerical solvers that integrate differential equations to various ML methods and other approximation techniques. Their performance can differ widely across different benchmarks, as seen in the Imagenet leaderboards [282]. In some cases, no single model outperforms others across all scenarios; each has its own strengths and weaknesses. For instance, the Intergovernmental Panel on Climate Change (IPCC) report on model evaluation [104] highlights numerous evaluation metrics, showing that different models excel in different areas. When multiple models are available to predict the same quantity of interest, a key challenge arises: how can their predictions be combined most effectively? Ensembling ML models is a well-established practice with many successful applications, such as bagging, boosting, and Mixture of Experts (MoE) [369]. Gradient boosting, in particular, is widely regarded for its effectiveness, combining simplicity and accuracy in data analysis [32]. Most ensembling methods focus on training strategies that build and combine multiple models [369]. However, methods for aggregating predictions from existing models are less developed and broadly used, and practitioners often rely on simplistic techniques like averaging [92]. This highlights the need for a simple and general framework to aggregate predictions and better exploit diverse models' unique strengths.

Related work Ensemble learning methods combine multiple learnable models to improve accuracy. While these works are influential training paradigms, our approach assumes the models are not trainable, and focuses solely on their aggregation, yielding a different task. We emphasize that our work is not comparable to popular

ensembles such as Gradient Boosting [295] as it combines black box models.
Extensive research has been made on model averaging, especially in cases where the models are assumed to be linear [139, 188, 374]. Combination strategies with constant factors include Pawlikowski and Chorowska [254], Viana et al. [342]. Other studies have explored the pointwise linear aggregation of surrogate models, with various heuristics using weighted averages. Zerpa et al. [367] uses the variance predicted by Gaussian processes, Lee and Choi [180], Liang et al. [187], Ye et al. [363] try to predict the cross-validation error at each point and use it to combine models while Liu et al. [193] trains an aggregation to minimize the Mean Squared Error of the aggregate. Crucially, these methods are heuristics, which do not justify why the aggregation should be a convex combination, or why it should be weighted using cross-validation. Secondly, they do not formulate the aggregation as a learning problem and instead use specific approaches, such as K-nearest neighbours or interpolation through weighted averages. This limits the scope of the aggregation, in particular, these studies only aggregate one-dimensional regression, and prevent the methods from benefiting from advances in other areas of Machine Learning.
A growing trend in scientific computing involves applying statistical inference and machine learning (ML) methods [28, 41, 56, 271]. These problems are often framed as operator learning tasks, which can be seen as infinite-dimensional regression problems. In this work, we adopt a statistical/ML framework to unify and aggregate diverse methods, even those not inherently inference or ML-based. This approach is particularly well-suited to structured problems with inherent design constraints, where heterogeneous models naturally arise. Such contexts often involve a combination of non-trainable numerical solvers, legacy simulation codes, and specialized ML models with distinct trade-offs, which are challenging to integrate or retrain using standard ensemble techniques. Our work focuses on addressing these challenges by reconciling and leveraging the strengths of pre-trained or specialized models in constraint-heavy, structured problems.

Contributions We introduce Minimal Empirical Variance Aggregation (MEVA), a versatile method for aggregating predictions from diverse models, treating them as black-box input-output functions (deterministic or stochastic). MEVA provides a robust, non-intrusive framework for structured problems, particularly in scientific computing and other domains where traditional methods often fall short. While the approach is broadly applicable, its primary strength lies in addressing challenges in scientific computing and other structured domains. Key contributions include:

- **Justification and formulation**: Contrary to previous heuristics, we present a probabilistic model for aggregation, grounded in clear assumptions that justify using convex combinations and approximations of cross-validation error (Sections 2.2, 2.3). Future works may build upon this model and adapt it to constraints specific to the application at hand. We also formulate MEVA as a learning problem (sec. 2.3), making an extension to other settings such as graphs or computer vision simple using popular learning techniques of these fields.

- **Ruling out MEEA** We assess Minimal Empirical Error Aggregation (MEEA), which minimizes aggregate empirical error directly, and identify its limitations via a pathological example (Section 2.2.2). MEVA, introduced in Section 2.3, addresses these challenges and consistently outperforms MEEA in experiments. We rigorously prove that MEVA converges faster than MEEA, making it more robust in scarce data regime.

- **Extension to operator learning** Our approach extends model aggregation beyond machine learning to include scientific computing, enabling the integration of diverse methods for estimating shared target quantities. We demonstrate this by expanding from one-dimensional problems to operator learning using established techniques (Section 2.4.2). To our knowledge, this is the first work to aggregate outputs across machine learning and classical PDE-solving methods.

The remainder of the paper is structured as follows: Section 2.2 discusses pointwise aggregation and MEEA's limitations. Section 2.3 introduces the probabilistic foundation of MEVA, a theorem proving its superiority, and its computation. Section 2.4 validates MEVA with two experiments: a sanity check in data science and examples from Scientific Machine Learning (SciML) involving PDE solving.

2.2 The minimal error aggregation

2.2.1 Best linear aggregate when model correlations are known

An effective way to derive an aggregation method is to define it as optimal with respect to a specific loss function. If the target function $Y(x)$ and the models $M_i(x)$ are viewed as random variables on the same probability space, a commonly used loss function is the Mean Square Error (MSE). In this context, an aggregated model $M_A(x)$ can be defined as any measurable function of the models that minimizes the

expected squared error:

$$M_A(x) := \arg\min_{f \text{ measurable}} \mathbb{E}\left[(Y(x) - f(M_1(x), .., M_n(x))^2\right] \\ = \mathbb{E}\left[Y(x) \,|M_1(x), .., M_n(x)\right] , \quad (2.1)$$

where the second equality follows from the L_2 characterization of conditional expectation. While computing the conditional expectation can be intractable [2], this computation reduces to solving a linear system in the Gaussian case. Specifically, if the vector $(Y(x), M_1(x), \ldots, M_n(x))$ is Gaussian with mean 0 (see remark below), then the conditional expectation in the equation above is a linear combination of the models' outputs. Rullière et al. [281] uses this assumption to aggregate Gaussian processes. Bajgiran and Owhadi [18] demonstrates that any consistent and rational aggregation of models must be a weighted average that is point-wise linear. Finally, many ensembling methods (random forest [43], MoE [365]) also use a linear combination of models. Motivated by this, we restrict our aggregation approach to a pointwise linear combination of the models:

$$M_A(x) = \alpha^*(x)^T M(x) \quad (2.2)$$

With this simplification, aggregation can be achieved by determining the Minimal Error Aggregation (MEA) $\alpha^*(x)$ such that:

$$\alpha^*(x) = \arg\min_{\alpha \in \mathbb{R}^n} \mathbb{E}\left[(Y(x) - \alpha^T M(x))^2\right] \\ = \mathbb{E}\left[M(x)M(x)^T\right]^{-1} \mathbb{E}\left[M(x)Y(x)\right] \quad (2.3)$$

By definition, the MEA is the best possible point-wise linear aggregation. This equation reveals that the MEA requires knowledge of both the correlation matrix of the models, $\mathbb{E}\left[M(x)M(x)^T\right]$, as well as the correlation between the models and the target $\mathbb{E}\left[Y(x)M(x)\right]$. We demonstrate in appendix A.2.2 that the aggregated model can achieve exceptional performance when these correlations are perfectly known.

Remark In the general case with non-zero mean, the aggregation of a Gaussian vector would be affine. Since we do not assume anything on the models, affine coefficients can be recovered by adding an extra constant model, i.e., by a linear aggregation of $\tilde{M}(x) = (M(x), 1)$. In section 2.2.2, we will see that this can lead to catastrophic failure for empirical error minimization. Appendix A.6 shows that empirical variance minimization provides a more robust approach for estimating bias.

2.2.2 Data-driven aggregation

We will now consider the situation where the quantities $\mathbb{E}[Y(x)M(x)]$ and $\mathbb{E}\left[M(x)M(x)^T\right]$ required for the best linear aggregate (2.3) may be ill-defined (e.g., because the underlying models are not stochastic) or difficult to estimate. In this situation, a natural alternative approach is to estimate the aggregation coefficients α^* appearing in (2.2) directly from data by a minimizer $\hat{\alpha}$ of a (possibly regularized) empirical version of the loss (2.3), as seen in [193]:

$$\hat{\alpha}_E = \arg\min_{a\in\mathcal{H}} \sum_{i=1}^{N} \left|Y^i - a(X^i)^T M(X^i)\right|^2 + \lambda\|a\|_{\mathcal{H}}^2 . \tag{2.4}$$

where the $(X^i, Y^i := Y(X^i))$ are N data points; $\mathcal{H}$ is a set of functions used for the approximation; $\lambda \geq 0$ and $\|a\|_{\mathcal{H}}^2$ is a regularizing norm. We call this aggregation Minimal Empirical Error Aggregation (MEEA). While this transition from expected loss to empirical loss may seem well-founded, it can introduce a significant loss of information, which yields to the aggregation completely failing, as showcased in the following sections.

Pathological example: A dubious trend In this example, we introduce a basic aggregation with linear coefficients, where the MEEA fails. This, along with appendix A.3 and A.7, gives intuition as to why MEEA is not the appropriate strategy. Figure 2.1 represents the results of this experiment. We pick Y, a good model M_G and bad models M_B s.t.:

$$\begin{cases} Y(x) &= 2x + \cos(3\pi x) \\ M_G(x) &= Y(x) + \epsilon(x) \qquad \text{where } \epsilon(x) \sim \mathcal{N}(0, 0.2) \\ M_B(x) &= 1 \end{cases} \tag{2.5}$$

We also choose a linear aggregation, which means that we will find a_G, a_B, b_G, b_B s.t. $\hat{\alpha}(x) = (a_G x + b_G, a_B x + b_B) \in \mathbb{R}^2$. Finally, we pick some trick data $X = (0.8 - 2/3 * 4, 0.8 - 2/3 * 3, .., 0.8, -0.8, -0.8 + 2/3 * 1, .., -0.8 + 2/3 * 4)$ so that the $Y(X^i) = Y^i$ form a line. We may see that M_G has a lower error than M_B on each X^i by evaluating the models at these data points. Thus, we expect an aggregation method to use the good model primarily in the aggregation. However, the opposite behaviour is observed. Minimizing the loss (2.4) with $\lambda = 0$, we obtain that the coefficient on the good model is 0, so M_G is completely ignored, which is counterintuitive. The aggregate uses the bad model, M_B, to directly estimate Y by linear regression. This shows that the MEEA simply employs models as features to interpolate the data points instead of weighting each model according to its estimated accuracy.

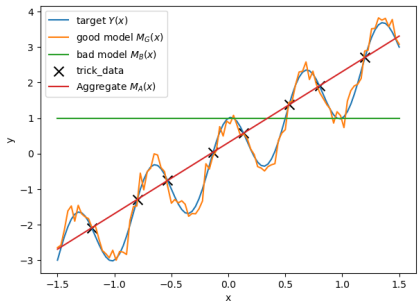


Figure 2.1: Pathological example (sec. 2.2.2)

2.3 Aggregation through error estimation: the Minimal Variance Aggregate (MVA)

2.3.1 Modelling the error

By placing no assumption on our model, MEEA is vulnerable to pathological examples, and in our experiments does not generalize well. In particular, there is nothing representing the fact that models are approximations of the target. To encode this, we add the assumption that the models, represented in the vector M, are unbiased estimators of Y. Thus, the errors of each model are interpreted as follows:

$$M(x) = \begin{pmatrix} Y(x) \\ \vdots \\ Y(x) \end{pmatrix} + Z(x) \text{ where } \begin{cases} \mathbb{E}[Z(x)] = 0 \\ \text{Cov}[Z(x)] = A(x) \end{cases} \tag{2.6}$$

Under this assumption, any linear combination with weights $\alpha_i(x)$ s.t. $\sum_{i=1}^{n} \alpha_i(x) = 1$ is unbiased. Thus, we define the Minimal Variance Aggregate (MVA) as the Best Linear Unbiased Estimate (BLUE) of $Y(x)$:

$$M_A(x) = \alpha_V(x)^T M(x) := \frac{\mathbb{1}^T A(x)^{-1} M(x)}{\mathbb{1}^T A(x)^{-1} \mathbb{1}} \tag{2.7}$$

where $\mathbb{1} := (1, .., 1)^T$ is the all ones vector. Note that

$$\alpha_V(x) = \arg\min_{\mathbb{1}^T a = 1} \mathbb{E}\left[(Y(x) - a^T M(x))^2\right],$$

regardless of whether $\mathbb{E}[Z(x)] = 0$. In previous heuristics [180, 187, 193, 342], $\sum_{i=1}^{n} \alpha_i(x) = 1$ is assumed, while we justify it as the set of unbiased linear aggregations. This leads to a natural generalization for biased models in appendix A.6, while in practice bias correction did not improve results. The following section proves that despite MVA being inferior to MEA in MSE, MVA can be more robustly estimated, making MEVA better performing than MEEA.

2.3.2 A theorem on the superiority of MEVA over MEEA

To investigate the theoretical properties of MEVA and MEEA, we fix x and focus on estimating α^* and α_V. Details and proofs are given in appendix A.1. Let Y be a random real-valued variable and the random vector of models $M \in \mathbb{R}^n$. Suppose we have i.i.d observations $(M^1, Y^1), \ldots, (M^N, Y^N)$, collected in the matrices $\mathcal{M} \in \mathbb{R}^{N\times n}$ and $\mathcal{Y} \in \mathbb{R}^N$. Define $M = Y\mathbb{1} + \epsilon Z$, where Z is the scaled error, with ϵ chosen so that $\mathbb{E}\left[ZZ^T\right] := A$ has Frobenius norm 1 ; and $b := \mathbb{E}\left[YZ\right]$. We find that the MEA α^* defined in equation (2.3) and the MVA α_V (equation (2.7)) satisfy:

$$\alpha^* = \lambda\alpha_V + (1-\lambda)\alpha_R \text{ where } \begin{cases} \lambda &= \frac{s(E[Y^2]-u)}{s(E[Y^2]-u)+(\epsilon+t)^2} \\ \alpha_R &= \frac{1}{\epsilon+t}A^{-1}b \end{cases} \tag{2.8}$$

with $s = \mathbb{1}^T A^{-1}\mathbb{1}$, $t = b^T A^{-1}\mathbb{1}$ and $u = b^T A^{-1} b$ assumed to be non zero quantities. Note that $\lambda \in [0,1]$, making α^* a convex combination of α_V and α_R. For the empirical variants of these aggregations, we use $\hat{A} = \frac{1}{\epsilon^2 N}(\mathcal{M}^T - \mathbb{1}\mathcal{Y}^T)(\mathcal{M} - \mathcal{Y}\mathbb{1}^T)$. Using the central limit theorem on $\hat{A} - A := \frac{1}{\sqrt{N}}\delta A$, we find that the scaled error δA converges to a Gaussian Matrix of finite covariance. Similarly, $\mathbb{E}\left[Y^2\right]$ and b are estimated with estimators converging at $\frac{1}{\sqrt{N}}$ rates, with δV and δb their scaled error. Write $\hat{\alpha}_E := (\mathcal{M}\mathcal{M}^T)^{-1}\mathcal{M}^T\mathcal{Y}$ for the empirical MEA, $\hat{\alpha}_V = (\hat{A}^{-1}\mathbb{1})/(\mathbb{1}^T\hat{A}^{-1}\mathbb{1})$ the empirical MVA, and $\mathcal{L}(\alpha) = \mathbb{E}\left[(Y - \alpha^T M)^2\right]$ the Mean Squared Error. We prove the following theorem:

Theorem 2.3.1. *Assume $s, u, t \neq 0$, then there exist two sequences of random variables, K_N^E and K_N^V, each of which converges in distribution to a distinct finite random variable, such that*

$$\mathcal{L}(\hat{\alpha}_V) = \frac{1}{\lambda}\mathcal{L}(\alpha^*) + \frac{1}{N}K_N^V \tag{2.9}$$

$$\mathcal{L}(\hat{\alpha}_E) = \mathcal{L}(\alpha^*) + \frac{1}{\sqrt{N}}K_N^E \tag{2.10}$$

MEVA incurs a constant error relative to MEEA, given by $\frac{1}{\lambda}$. However, MEVA is less sensitive to perturbations, causing its error to converge more rapidly to its limiting value than MEEA. Thus, in the model aggregation setting where λ is close to 1[1], the smaller difference in limiting loss values is outweighed by MEVA's faster convergence making it outperform MEEA

While this theorem applies to a fixed x and N observations, we are interested in the case where these N data points are used to extrapolate across all x. In such cases,

[1] See appendix A.1 for more details

model aggregation inherently operates in the scarce data regime where $\hat{A} - A$ is large. This amplifies the behavior described in the theorem, further emphasizing the practical advantages of MEVA.

2.3.3 Computing the aggregation: the Minimal Empirical Variance Aggregate (MEVA)

To compute our empirical aggregation $\hat{\alpha}_V(x)$, we must compute the covariance matrix of the error for all $x \in \mathcal{X}$ from the data. This poses several challenges when $A(x)$ is unknown. First, we must approximate a symmetric matrix everywhere on the domain, needing $n(n+1)/2$ coefficients to aggregate n models. Moreover, we must ensure this matrix is always positive-definite. To simplify the approximation and to easily enforce the positive-definite condition, we suppose that for all x, $A(x)$ can be diagonalized using a fixed eigenbasis P. Then, its positive eigenvalues are approximated by $e^{\lambda_i(x)}$, where the λ_i are functions to be regressed from the data. Thus, we approximate $A(x)$ with $\hat{A}(x) = P^T Diag(e^{\lambda_1(x)}, .., e^{\lambda_n(x)})P$. This formulation has the advantage of symmetrizing the covariance and precision matrix estimation, as the log-eigenvalues of both matrices are opposite. This is desirable because while we are interested in the precision matrix, the covariance matrix is easier to estimate. As a further simplifying assumption, we will take $P = I_n$, which is equivalent to assuming that models have independent errors. Appendix A.4 presents the general case $P \neq I_n$, which in our experiments did not improve performance. Using this assumption on $\hat{A}(x)$ in (2.7), we get:

$$\begin{cases} \hat{A}(x) &= Diag(e^{\lambda_1(x)}, .., e^{\lambda_n(x)}) \\ \hat{\alpha}_V(x) &= Softmax(-\lambda(x)) \end{cases} \tag{2.11}$$

where λ_i are regressors $Softmax(-\lambda(x))_i = \frac{e^{-\lambda_i(x)}}{\sum_{k=1}^{n} e^{-\lambda_k(x)}}$. Notice that using our model of the error (2.6), we have re-discovered the popular softmax, used in particular in Mixture-of-Experts (MoE). All that remains for our aggregation to be complete is to train the regressors λ_i. To do so, we will rely on covariance matrix estimation. Let $Z^1, ., Z^N$ be i.i.d. samples of the random variable $Z \in \mathbb{R}^n$ such that $\mathbb{E}[Z] = 0$. An unbiased estimate of $Cov[Z]$ is

$$\hat{\text{Cov}}\, Z = \frac{1}{N}\sum_{i=1}^{N} Z^i (Z^i)^T = \arg\min_{\Sigma \in \mathbb{R}^{n\times n}} \sum_{i=1}^{N} \|\Sigma - Z^i (Z^i)^T\|_F^2 \tag{2.12}$$

where $\|\cdot\|_F$ is the Frobenius norm. If we further assume that $\hat{\text{Cov}}\, Z$ must be diagonal (as we do with $\hat{A}(x)$), then its coefficients are such that $(\hat{\text{Cov}}\, Z)_{kk} =$

$\arg\min_{c\in\mathbb{R}} \sum_{i=1}^{N} \|c - (Z_k^i)^2\|_2^2$. We will now regularize this arg min formulation to approximate the model errors covariance matrix for all values of x. Writing e^i for the vector with entries defined as the sample errors $(e^i)_k = M_k(X^i) - Y(X^i)$, we identify the functions λ_i as

$$\lambda = \arg\min_{l\in\mathcal{H}} \sum_{i=1}^{N} \sum_{k=1}^{n} \left(e^{l_k(X^i)} - (e^i)_k^2\right)^2 + a\|l\|_{\mathcal{H}}^2 \tag{2.13}$$

Here, we denote the entries of the vector-valued function l by l_k, and $\mathcal{H}$ represents the space of vector-valued functions selected to approximate the log-eigenvalues. This space could be a RKHS associated with a particular kernel, a neural network with a specific architecture, or any other normed function space. The notation $\|\cdot\|_{\mathcal{H}}$ refers to the regularization norm intrinsic to that space, such as the RKHS norm or the L_2-norm of the weights and biases in a neural network.

2.4 Experiments

2.4.1 Sanity check: aggregation on the Boston housing dataset

The Boston housing dataset [142] is a popular benchmarking dataset for regression. It contains $N = 506$ samples of 14 variables. We use this dataset as a sanity check that our approach can improve upon the predictions of the aggregated models, while our focus is on the PDE examples below. We split this dataset into train, validation, and test sets, and average results over many splits. To evaluate our method, we train several standard ML methods on the train set (e.g. linear regression, gradient boosting, etc.). We then train our aggregation on the validation set before testing on the test set. To get a fair comparison, we also trained the ML methods on the train and validation sets combined so that they see the same amount of data as the aggregate. We first implement our aggregation as described in section 2.3.3, using a kernel method with kernel κ. This is equivalent to minimizing (2.13) over $\lambda_i \in \mathcal{H}_\kappa$ where $\mathcal{H}_\kappa$ is the RKHS defined by the kernel κ. After experimenting with many kernels, the best we found uses the Matérn kernel $\kappa_{\text{Matérn}}(u, v, \rho) = \left(1 + \frac{\sqrt{3}\|u-v\|}{\rho}\right) \exp\left(-\frac{\sqrt{3}\|u-v\|}{\rho}\right)$, and is defined as

$$\kappa(x, y) = \kappa_{\text{Matérn}}(M(x), M(y), \rho) \tag{2.14}$$

with adequate choice[2] of ρ. This formulation uses the prediction $M(x)$ to output the aggregation coefficients $\alpha(x)$.

The results of this experiment are shown in figure 2.2. In this situation, the mean

[2]See details in the paper's repository

performs well. The aggregate, denoted *MEVA with kernel*, performs better than all individual models being aggregated, their average and any ML model when fed the same data. This is a success for the MEVA, which is difficult to obtain with MEEA. The aggregation improves upon the performance of the best aggregated model, Gradient Boosting, by $r_{\text{train}} = 1 - \frac{MSE(\text{Aggregate})}{MSE(\text{Gradient Boosting})} = 7\%$, and the best model overall by $r_{\text{all}} = 2\%$.

We conduct a second experiment to demonstrate the effectiveness of our proposed method compared to Minimal Error Aggregation, defined in (2.4). In this experiment, we use a small, fully connected Deep Neural Network (DNN)[3] and train it using two different loss functions: the MEA described in section 2.2, and a Minimal Variance Aggregate (MVA) using an end-to-end loss especially useful for neural networks, described in appendix A.5. The results, presented in figure 2.2, highlight the impact of the chosen loss function. Since the only difference between the two approaches is the loss function, the results clearly illustrate the advantages of variance minimization. Specifically, the minimal error aggregation (denoted *MEEA with DNN* in the figure) performs significantly worse compared to our minimal variance aggregation (denoted *MEVA with DNN* in the figure). It also performs worse than the two best models aggregated.

2.4.2 Aggregation of PDE solvers

This section introduces the approach for combining Partial Differential Equation (PDE) solvers through model aggregation. By incorporating classical PDE solvers into the aggregation framework, we lower the accuracy threshold necessary for the aggregated solution to outperform individual models. Notably, this threshold is significantly below the typical 1% relative error benchmark targeted by most ML-based PDE solvers. We first formulate the aggregation problem in the operator learning context, followed by two experimental case studies demonstrating the effectiveness of this approach.

2.4.2.1 Operator Learning in the Aggregation Context

Consider the Laplace equation with zero Dirichlet boundary conditions on the domain $\Omega = [0, 1]^2$ as an illustrative example of a PDE:

$$\begin{cases} -\Delta u^\dagger(\omega) = f(\omega) & \text{for} \quad \omega \in \Omega \\ u^\dagger(\omega) = 0 & \text{for} \quad \omega \in \partial\Omega \end{cases} \tag{2.15}$$

[3] See details in the paper's repository

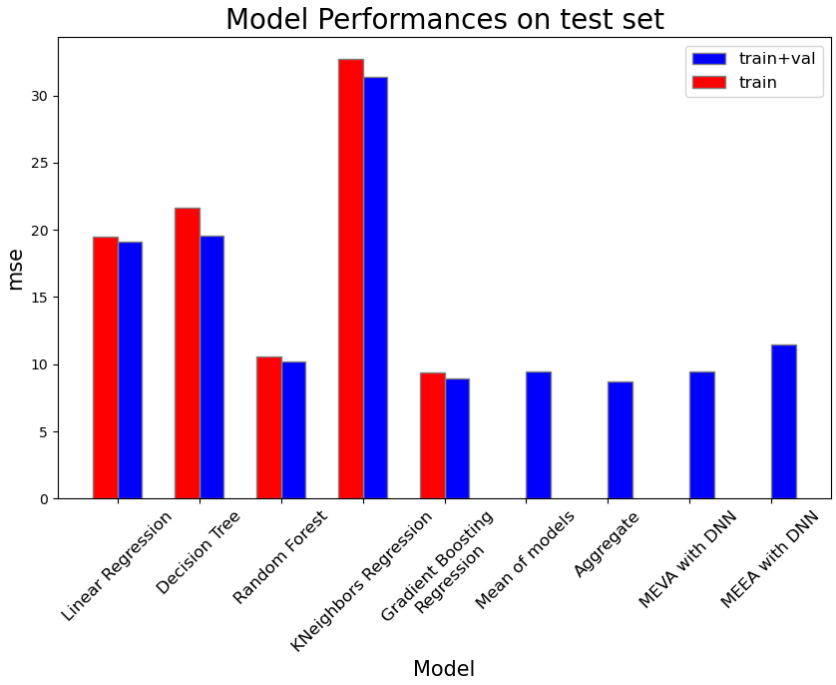


Figure 2.2: Comparisons of the performance of the different models on the Boston housing dataset. Red bars are the performance of models trained using the training set and used in the aggregation. Blue bars show models trained on the training and validation set (*train+val*) to get a fair comparison with the aggregations. The aggregation does not use the models trained on *train+val*.

where $f \in \mathcal{F} \subset L^2(\Omega)$ and $u^\dagger \in \mathcal{Y} \subset H^2(\Omega) \cap H_0^1(\Omega)$ is the solution. The solution operator maps the source term f to the solution $u^\dagger$. In this context, the solution operator is defined as $S : f \in \mathcal{F} \mapsto u^\dagger \in \mathcal{Y}$. This operator is often approximated using a PDE solver, which can be based on methods like finite element analysis [24] or spectral methods [105]. Alternatively, the operator can be learned directly from pairs of input/output solutions, a supervised learning task known as operator learning [173]. Since PDE solvers and machine learning methods approximate operators, we can naturally frame our aggregation task as an aggregation of multiple operators (or PDE solvers), denoted $M_1, \ldots, M_n$. The aggregated operator M_A is then expressed using α, an operator that maps $\mathcal{F}$ to $L^2(\Omega, \mathbb{R}^n)$, so that:

$$M_A(f) = \sum_{i=1}^{n} \alpha_i(f) M_i(f) \in \mathcal{F} \tag{2.16}$$

Remark: In the operator context, one may consider many aggregations that are pointwise linear with respect to the models, for instance, the aggregation of Fourier coefficients, $\tilde{M}_A(f) = \mathcal{T}^{-1}\left(\sum_{i=1}^{n} \tilde{\alpha}_i(f)\mathcal{T}(M_i(f))\right)$, where $\mathcal{T}$ represents the Fourier

transform.

2.4.2.2 The operator aggregation loss

To get the loss for our problem, we will first state the general loss in the operator learning setting and then modify it for increased performance. Since the aggregation loss (2.13) is defined for real-valued objectives with an abstract input $x \in \mathcal{X}$, we must specify $\mathcal{X}$ in the context of operator learning. We use $\mathcal{X} = \{(\omega, f) \in \Omega \times \mathcal{F}\}$, which means $x = (\omega, f)$ is a choice of point in space and source term. We define $Y(x) := S(f)(\omega) = u^{\dagger}(\omega) \in \mathbb{R}$. Given a grid $(\omega^1, ...\omega^{N_1}) \in \Omega$ and a set of input functions $(f^1, .., f^{N_2})$, we get the $N_1 \times N_2$ datapoints $X^{i,j} := (\omega^i, f^j)$ for our aggregation. These pairs of datapoints $(X^{i,j}, Y^{i,j} = S(f^j)(\omega^i))$ can then be used in (2.13).

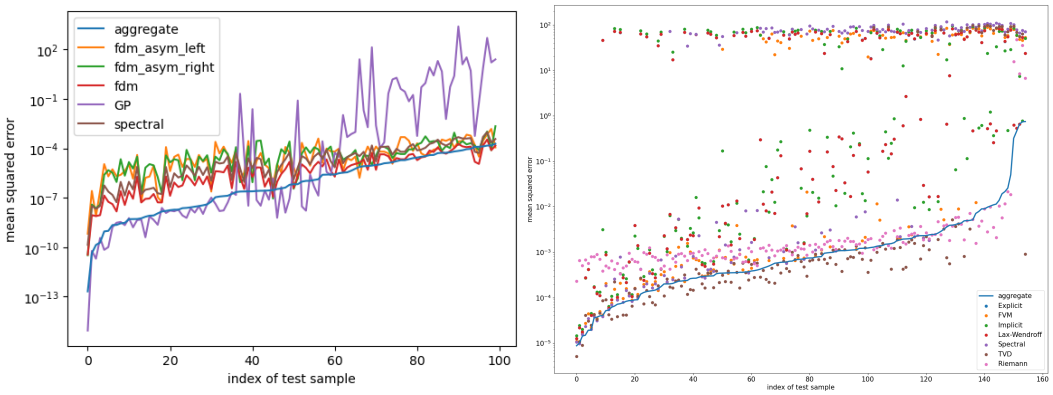
When solving PDEs, the focus is often on the order of magnitude of the error, i.e., its logarithm. Therefore, we may adapt the loss function (2.13) to account for this. Specifically, observe that if the regularization parameter a is small, then functions λ obtained from regressors that can interpolate arbitrary data (e.g., GPs with universal kernels) may result in a near-zero loss and $e^{\lambda_k(X^{i,j})} \to (Y(X^{i,j}) - M_k(X^{i,j}))^2 := e_k^{i,j}$ as $a \to 0$. Thus, we can linearize the exponential around $\log[e_k^{i,j}]$ to see that:

$$(e^{\lambda_k(X^{i,j})} - e_k^{i,j})^2 \approx \left(l_k(X^{i,j}) - \log[e_k^{i,j}]\right)^2 (e_k^{i,j})^4 \tag{2.17}$$

This linearization shows that samples where the models perform well are down-weighted, and the aggregation would focus on instances where they perform poorly. Instead, we use the following more sensitive loss (2.18) where the down-weighting is removed, which, in practice, gives good results on these problems:

$$\lambda_s = \arg\min_{l \in \mathcal{H}} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1}^{n} \Big(l_k(X^{i,j}) - \log\left[(Y(X^{i,j}) - M_k(X^{i,j}))^2\right]\Big)^2 + a\|l\|_{\mathcal{H}}^2 \tag{2.18}$$

We also recall the process for obtaining α and the aggregate M_A from λ_s is given in equations 2.11 and 2.16. In both examples, we use the Fourier Neural Operator (FNO) [185] to learn the operator l. We also explored a kernel-based approach for operator learning, as detailed in Batlle et al. [28]. Although operator learning using a Matérn kernel yielded good results, FNO offered better performance with minimal hyperparameter tuning for the examples discussed in this section.



(a) Laplace equation (b) Burgers' equation

Figure 2.3: log MSE of the different methods for: (2.3a) the Laplace equation; (2.3b) Burgers' equation. Samples are sorted by the error of the aggregate.

2.4.3 The Laplace equation

Our first example is the Laplace equation (2.15). We use six simple solvers as models: a finite difference method, two finite differences with an inhomogeneous grid (one denser for $x < 0.4$, the other for $x > 0.4$), a spectral method, and a GP solver. Our data is generated by sampling[4] random parameters f_{max}, μ_0, μ_1, R for functions of the form

$$u(x,y) = -\sin(\pi x)\cdot\sin(\pi y) \cdot \sin\left(f_{max}\cdot\exp\left(-\begin{bmatrix}x-\mu_0\\ y-\mu_1\end{bmatrix}^T\cdot R\cdot\begin{bmatrix}x-\mu_0\\ y-\mu_1\end{bmatrix}\right)\right) \quad (2.19)$$

Six hundred pairs of functions $(f_i := \Delta u_i, u_i)$ are sampled, of which 500 are used for training and 100 for testing. Figure A.3 shows an example of such a pair. These functions and the models are evaluated on a grid of 100×100. The aggregation is performed by FNO, which takes input f and the models' outputs and outputs the aggregation coefficients' logits λ. When training a FNO with parameters $\theta \in \Theta \subset \mathbb{R}^{N_\Theta}$ and loss (2.18), we find the minimizer of (2.18) with $\mathcal{H} = \{f \mapsto FNO(\theta)(f, M_1(f), .., M_n(f)) \text{ for } \theta \in \Theta\}$ with $\|FNO(\theta)\|_{\mathcal{H}} = \|\theta\|_2$.

Apart from the GP solver, all other methods perform similarly, with the finite difference method being the best overall. The GP solver, implemented with a fixed length scale, has two regimes. When the function u_i is not evolving too fast (i.e. when f_{max} is not too large), the kernel is well specified, and the GP solver obtains the best performance. On other occasions where f_{max} is larger, the method fails

[4] See details in the paper's repository

and has a substantial error. Thus, the aggregation challenge here is to use the GP when performing well but ignore it when it fails. The result of this experiment is shown in figure 2.3a. Because of the significant error in the failure case, we observe that the mean is a poor aggregation that performs worse than all models aggregated. Our aggregate is consistently among the best performers on each sample and the best-performing model for many samples. On average, our aggregate performs one order of magnitude better than the models aggregated, as shown in table 2.1. An example of model outputs, errors, α, and final aggregation is shown in figure A.4. As a comparison, we also train FNO using a direct MSE loss of the aggregation, akin to (2.4). We observe that this method fails as the aggregation converges to a constant, i.e., $\alpha(f, \omega) \approx (1, 0, 0, 0, 0, 0)\ \forall\ f, \omega$, picking the best-performing model. We believe that this convergence to a fixed value, without combining the different models to get a better one as observed in our method, is due to the absence of the log term that we added in the sharp loss (2.18). As we observed in section 2.4.2.2, without this log it is difficult for a loss to differentiate between 10^{-3} and 10^{-9} error. Furthermore, while learning the log-error instead of the error is a simple modification from (2.13) to (2.18), it is difficult to see where this log would be added in a loss like (2.4).

2.4.4 Burgers' equation

We now turn to a non-linear PDE, Burgers' equation with periodic boundary condition on $\Omega = [0, 1]^2$:

$$\begin{cases} \partial_t u(t,x) + \partial_x(\frac{1}{2}u^2(t,x)) = \nu \partial_{xx} u(t,x) \\ u(0,x) = f(x) \\ u(t,0) = u(t,1) \end{cases} \tag{2.20}$$

We choose $\nu = 2.10^{-3}$ and pick initial conditions f_i as samples from the Gaussian process $\mathcal{N}(0, K_{\exp\sin^2})$ where $K_{\exp\sin^2}(x, y) = \exp\left(-\frac{2\sin^2(\pi|x-y|)}{l^2}\right)$ and $l = 1.5$. With this set of parameters, the initial conditions are infinitely differentiable periodic functions that often, but not always, form a shock within the time frame studied ($t \in [0, 1]$). We use seven different solvers with the same fixed discretization grid. While the first five methods implement the viscid version of Burgers' equation directly, for instance, using an explicit scheme or a finite volume method, the penultimate uses a flux limiter. The last one solves the inviscid Burgers' equation. With this variety of solvers, we have methods that are accurate in smooth cases when there is no large shock, an intermediate method that may be more robust to shocks, and a method that will never diverge even in the presence of shocks, but

will be less accurate in general because it solves a slightly different equation. A good aggregation must be able to recognize artefacts and divergences that may arise around shocks but use the more precise, although more brittle, solvers in cases where they are accurate. To perform the aggregation, similarly to the previous section, we use FNO, to which we feed the models' prediction as a stack of 2-dimensional arrays (one for time and one for space). Specifically, we minimize loss (2.18) with $\lambda(f) = FNO(\theta)(M_1(f), .., M_n(f))$ for some θ.

The results are displayed in figure 2.3b. Our method successfully leverages the predictions of the different models to give a robust and accurate prediction. The behaviors of our aggregation can be separated into a few cases. On the left of the graph, we find easy initial conditions that all models can approximate correctly. We may notice the Riemann method, which solves the inviscid Burgers' equation, has lower accuracy. Thus, on this easy half of the graph, the aggregation mainly relies on the TVD solver, the most precise, to achieve excellent accuracy. In the right half, we find the more complex cases where the aggregation has an accuracy closer to the more robust Riemann solver. In this half, there are cases where the TVD method fails and diverges and is avoided by our aggregation. The process can identify each solver's strengths and weaknesses and obtain a better average error, as shown in table 2.2. An example of model outputs, errors, α, and final aggregation is shown in figure A.5.

Method	Geometric Mean of MSE ($\log_{10}$)
Aggregate	**-6.282**
FDM	-5.523
Spectral	-4.988
GP	-4.739
FDM (Asymmetric Right)	-4.685
FDM (Asymmetric Left)	-4.699

Table 2.1: Result of Laplace equation experiment (sec. 2.4.3)

2.5 Discussion

We introduced a general data-driven framework for aggregating models with minimal assumptions. As demonstrated in the pathological example Section 2.2.2 and the theorem 2.3.2, directly training the aggregate model is unstable and may result in an aggregator that fails to improve upon the performance of individual models. To address this, we introduced a simple assumption of unbiased models in (2.6) and reformulated the aggregation task as a variance minimization problem (Section

Method	Geometric Mean of MSE ($\log_{10}$)
Aggregate	**-3.106**
Riemann	-2.734
TVD	-2.568
FVM	-1.228
Spectral	-0.625
Implicit	-0.488
Explicit	-0.455
Lax-Wendroff	-0.455

Table 2.2: Result of Burgers' equation experiment (sec. 2.4.4)

2.3). In one data science problem (Section 2.4.1) and two PDE-solving examples (Sections 2.4.3 and 2.4.4), our loss function consistently outperformed the minimal error aggregate and led to an aggregation that surpassed the performance of the individual models. These latter two examples also illustrate the flexibility of our method, which can aggregate not only machine learning models but also other types of models, such as PDE solvers, in a data-driven manner. Notably, we used FNO to get an aggregate performance much higher than it could get when trying to solve the PDE directly.

Our method relies on the availability of unseen data to estimate the error of different models. Therefore, as seen in the Boston Housing example (section 2.4.1), splitting the data further to create an unseen validation set in scenarios where data is limited may significantly reduce the amount of training data available for the models, potentially impairing overall performance. Additionally, in challenging aggregation tasks, there is no guarantee that the aggregate will consistently outperform the individual models or simpler methods, such as averaging, despite the higher computational cost.

MEVA's flexibility makes it well-suited for integrating diverse methods, particularly in structured contexts like scientific computing, where non-intrusive solutions are often necessary. By requiring minimal assumptions about the models or data, MEVA can effectively aggregate legacy solvers that are difficult to modify, physics-based models built on differing assumptions, or models with incompatible software frameworks.

Chapter 3

OPERATOR LEARNING AT MACHINE PRECISION

Aras Bacho, Aleksei G. Sorokin, Xianjin Yang, Théo Bourdais, Edoardo Calvello, Matthieu Darcy, Alexander W. Hsu, Bamdad Hosseini, and Houman Owhadi. Operator learning at machine precision. *Journal of Computational Physics*, 566: 115240, 2026. doi: 10.1016/j.jcp.2026.115240. URL https://doi.org/10.1016/j.jcp.2026.115240.
TB contributed to method development and manuscript writing and editing.

3.1 Introduction

Operator learning [27, 40, 172] extends the idea of approximating finite-dimensional objects, such as functions, with neural networks to infinite-dimensional mappings, namely operators $\mathcal{G} : \mathcal{U} \to \mathcal{V}$ between Banach spaces $\mathcal{U}$ and $\mathcal{V}$. Often, $\mathcal{G}$ may be implicitly defined by another operator $\mathcal{F}$ such that $\mathcal{F}(u, \mathcal{G}(u)) = 0$. In the PDE setting, a central example is the solution operator $\mathcal{G}$ which maps input data in $\mathcal{U}$ such as initial conditions, boundary conditions, source terms, or other PDE parameters to outputs in $\mathcal{V}$, such as the solution of the PDE itself or other quantities of interest. The main motivation for using operator learning methods is that, once the surrogate operator $\mathcal{S}_\theta$ has been trained on a family of PDE instances, it can predict the solution for a new unseen input in a single forward pass. In contrast to traditional numerical solvers, this avoids solving the PDE from scratch for every new query. As a result, operator learning is particularly attractive in many-query settings, such as reduced-order modeling, real-time prediction, optimization, uncertainty quantification, and digital-twin applications. Its main advantages over classical numerical methods are the potentially much faster inference after training, the ability to amortize computational cost over many evaluations, and the flexibility to handle parametric families of PDEs within a unified framework. Depending on the architecture, operator-learning methods may also exhibit good transfer across discretizations or resolutions. At the same time, these methods also have limitations. Their accuracy may still be below that of high-fidelity numerical solvers, especially in applications requiring strict reliability or very high precision. Training can require large datasets and substantial computational resources. Moreover, performance may degrade outside the training distribution, and challenging high-dimensional, multiscale, or strongly nonlinear problems remain difficult. Finally, compared with

traditional numerical methods, operator-learning approaches often provide weaker guarantees in terms of error control, stability, and interpretability. Operator learning has attracted significant attention in scientific computing and has become one of the core problems of physics-informed machine learning (PIML) [156] for learning equations [47, 151, 279], and solving Partial Differential Equations (PDEs) and inverse problems [23, 55, 72, 219]. Among the most well-known operator learning models are Deep Operator Nets (DeepONet) [202] and Fourier Neural Operators (FNO) [184]. While differing in their parameterizations, operator learning methods are typically designed to learn discretized approximations of operators between function spaces from empirical data.

We consider two main categories of operator learning problems:

- **Category 1**: The target operator is available only through input-output pairs, and the governing equations or boundary conditions are unknown or incomplete. This often necessitates a purely data-driven approach.
- **Category 2**: The target operator is known to satisfy a set of constraints, typically explicitly or implicitly in the form of a PDE with appropriate boundary conditions.

In the context of **Category 2**, physics-informed operator learning methods [122] are often used to encode the constraints into an accelerated emulator for the PDE solution. While these approaches can provide substantial acceleration, to the best of our knowledge existing methods cannot match the accuracy of the numerical solvers they emulate.

3.1.1 Summary of contributions

In this article we focus on operator learning problems under **Category 2** and make six key contributions:

- **Introduction of CHONKNORIS:** We propose a novel neural operator learning method inspired by the Newton–Kantorovich method [263]. The CHONKNORIS model explicitly learns the dependence of the Cholesky factor $\mathcal{R}$ of the Gauss-Newton Hessian matrix $(\lambda I + \mathcal{F}'(u,v)^*\mathcal{F}(u,v))^{-1}$ on the approximate PDE solution v and the random coefficients u. Integrating CHONKNORIS Cholesky factor predictions $\hat{\mathcal{R}}$ into a quasi-Newton iterative algorithm enables us to emulate PDE solvers to machine precision error.

- **Generalization via FONKNORIS:** Building upon CHONKNORIS, we introduce FONKNORIS, a foundation modeling framework which learns the dependence of the Cholesky factors $\mathcal{R}$ on the approximate PDE solution v and a set of coefficients functions u common to a large class of PDEs. The enhanced generalization of FONKNORIS enables machine precision recovery of PDEs not seen during training.

- **Enhanced Accuracy and Interpretability:** By explicitly embedding the underlying physics into a kernel interpolation model or a neural network architecture resembling ResNet / Transformer layers, CHONKNORIS significantly surpasses the accuracy limitations of existing operator learning frameworks while providing improved interpretability through the physically meaningful structure of each iteration.

- **Theoretical Guarantees.** We establish an inexact Newton–Kantorovich analysis for the *learned* Tikhonov inverse approximate Hessian, yielding rigorous Kantorovich-style convergence guarantees. The theoretical result is instantiated on a nonlinear elliptic PDE, where we explicitly compute the design error for a kernel-based approximation of the learned Cholesky factors.

- **Comprehensive Experimental Validation:** Extensive numerical experiments demonstrate that CHONKNORIS and FONKNORIS are capable of machine precision recovery across a wide variety of forward and inverse nonlinear PDE problems. These include a nonlinear elliptic equation, the Burgers' equation, a nonlinear Darcy flow equation, Calderón's inverse problem, an inverse wave scattering problem, a seismic imaging full waveform inversion problem, and a FONKNORIS generalization to the Klein–Gordon and Sine–Gordon equations.

- **Benchmarking against operator learners.** Compared to strong baselines (kernel methods, Fourier Neural Operators, and Transformer-based Neural Operators), our approach lowers typical relative errors from around 10^{-3} or 10^{-2} to near 10^{-16} (machine precision) on benchmark forward and inverse problems.

Finally we highlight the practical and scientific use cases of CHONKNORIS and FONKNORIS algorithms:

- **Known physics.** When the residual map $\mathcal{F}(u, v) = 0$ is specified (forward or inverse settings) and high accuracy is required, the solver-emulating updates attain machine precision accuracy in practice, substantially exceeding typical operator-learning baselines.

- **Reliability.** CHONKNORIS is equipped with theoretical guarantees (e.g., convergence and stability under standard assumptions), yielding reproducible and dependable results.

- **Controllable accuracy and cost.** Accuracy is governed by the iteration budget; per-iteration complexity is dominated by Jacobian actions in the form of two triangular matrix–vector products.

- **Cross-PDE transfer.** Our foundation model, FONKNORIS, learns a single mapping from operator coefficients to Cholesky factors, enabling transfer across PDE families and generalization to previously unseen equations without retraining.

3.1.2 Brief review of the relevant literature

Methods for operator learning can be broadly categorized into three main groups: (1) Artificial neural network (ANN)-based approaches [171, 184, 202], (2) kernel-based methods [27, 155, 225], and (3) hybrid approaches [221, 240, 242]. Kernel-based methods are well-established, offering strong theoretical foundations and convergence guarantees. Theoretical support for ANN-based methods has been developed more recently [78, 172, 217, 274]. ANN methods have become popular as they can benefit from advanced hardware and software ecosystems, enabling efficient scaling to large datasets. When $\mathcal{G}$ corresponds to the solution operator of a PDE, learning $\mathcal{G}$ through evaluations of $\mathcal{F}$ is referred to as physics-informed operator learning [122, 186, 347]. These methods typically incorporate a loss term to enforce the consistency of the learned operator with the underlying PDE. As mentioned earlier, a common issue with operator learning methods is their limitations in terms of accuracy. For example, in tasks such as mapping diffusion coefficients to solutions in second order elliptic PDEs a relative L^2 error of around 0.1% is achieved on benchmark data sets, even in the physics-informed setting [144, 165, 186]. Adjacent to the above works, the recent papers [151, 198] explored operator learning via equation learning where $\mathcal{F}$ is learned from data and then inverted numerically to estimate $\mathcal{G}$ and achieved significant improvements in accuracy and data efficiency. However, these methods are not true emulators since every evaluation of the learned

operator requires the numerical solution of a nonlinear PDE. In parallel, there is a growing interest in *foundation models* for operator learning, where large neural operators are pre-trained across families of PDEs and subsequently adapted to solve new unseen PDEs or tasks, see, e.g., [37, 145, 364].

These results raise a compelling question:

> *Can operator learning for PDE problems achieve machine precision or at least approach it, if we incorporate explicit knowledge that the underlying map $\mathcal{G}$ is fully determined by known constraints?*

The recent works [94, 115, 116, 140, 144] introduced operator learning techniques based on traditional numerical algorithms. In [144], the authors propose a multigrid operator for solving linear PDEs while [94, 140] introduce an operator learning algorithm that emulates the iterative updates of Newton's method for solving PDEs. In [115, 116], the authors introduce Fredholm neural networks, which solve Fredholm integral equations of the second kind by unrolling an iterative fixed-point scheme into a feed-forward architecture. Several PDEs, including the Helmholtz equation, admit such formulations. In [115], they report small interior errors and near machine precision accuracy on the boundary for selected benchmarks. By contrast, [94] proposes *RandONets*: shallow, one-hidden-layer operator networks that first embed inputs via random projections and then learn only the output weights with linear solvers. For certain linear operators with aligned data, this approach attains near machine precision. An abstract viewpoint of that is provided in [101], who introduce generalized neural operator networks via unrolled Banach fixed-point iterations and prove that, when the iterator map is emulated sufficiently accurate, the depth of the neural network yields an exponential convergence for nonlinear operator equations in Banach spaces. This perspective complements our method conceptually: both works leverage contractive iterative structure, but CHONKNORIS is tailored to PDE-constrained operator learning through learned approximations of the linearized Newton–Kantorovich operators rather than a general fixed-point construction, cf. fig. 3.1 and appendix B.7. Furthermore, our numerical experiments indicate that the required number of iterations varies substantially across instances, suggesting that a fixed depth chosen in advance is suboptimal. The works [140] are the closest method to ours. Methodologically, the method in [140] learns a neural operator that directly approximates the nonlinear Newton update map and obtains a

solution by iterating this learned map, guided by a Newton-informed loss. In contrast, our CHONKNORIS framework regresses Cholesky factors of the underlying linear elliptic operators and hard-wires a regularized Newton–Kantorovich residual iteration as the network architecture. The CHONKNORIS framework therefore does not approximate the full nonlinear Newton step, but only the linear elliptic operator entering the Newton–Kantorovich linearization. Under standard assumptions ensuring that the Newton–Kantorovich iteration based on the exact elliptic operator is contractive, and provided that the learned operator is sufficiently close to this exact operator so as to preserve contraction, increasing the iteration budget systematically reduces the error down to the accuracy level dictated by the underlying numerical discretization.

Gauss–Newton algorithms, along with other quasi-Newton algorithms as well as their function space extensions, the Newton–Kantorovich method, have been widely used for solving nonlinear PDEs [267, 366] and inverse problems [133, 134]. Such algorithms are also central to the design of recent RKHS methods for solving PDEs [55] as well as boosting Physics Informed Neural Net (PINN)-type methods [153]. The wide applicability and simple abstract formulation of quasi-Newton algorithms suggests that they can be emulated or approximated using Machine Learning (ML) models, further motivating our exposition in light of algorithm unrolling ideas in the design of ANN architectures [220, 362].

Finally, we note that operator learning is deeply related to older ideas in scientific computing and applied mathematics such as computer model emulation [158], operator compression [100], polynomial chaos expansions [356], and model order reduction[297] among others. For more historical remarks in this direction we refer the reader to the literature review in [27].

3.1.3 Outline

The remainder of this article is organized as follows. Section 3.2 details the operator learning problem (Section 3.2.1), the Newton–Kantorovich method (Section 3.2.2), the Newton–Kantorovich method with Tikhonov regularization (Section 3.2.3), the proposed CHONKNORIS method (Section 3.2.4), and the proposed FONKNORIS foundation model (Section 3.2.5). Section 3.3 details our numerical experiments including benchmarking CHONKNORIS against existing operator learning methods (Section 3.3.1), forward problem modeling with CHONKNORIS (Section 3.3.2), foundation modeling with FONKNORIS (Section 3.3.3), and inverse problem mod-

eling with CHONKNORIS (Section 3.3.4). Section 3.4 provides theoretical guarantees of the CHONKNORIS method. Finally, Section 3.5 gives a brief conclusion and outlook of future work.

3.2 Operator Learning

3.2.1 The operator learning problem

Let $\mathcal{U}, \mathcal{V}$, and $\mathcal{W}$ be separable Banach spaces, and denote with $\mathcal{L}(\mathcal{V}, \mathcal{W})$ the space of linear bounded operators from $\mathcal{V}$ to $\mathcal{W}$. We are interested in learning the operator $\mathcal{G} : \mathcal{U} \to \mathcal{V}$ that is implicitly defined by $\mathcal{F} : \mathcal{U} \times \mathcal{V} \to \mathcal{W}$:

$$\mathcal{G} : u \mapsto \mathcal{G}(u) := v \quad \text{such that} \quad \mathcal{F}(u, \mathcal{G}(u)) = 0 \quad \text{for all } u \in \mathcal{U}.$$

In this work, we assume to know and have full access to the operator $\mathcal{F}$, and to its Fréchet derivative $\frac{\delta \mathcal{F}}{\delta v} : \mathcal{U} \times \mathcal{V} \to \mathcal{L}(\mathcal{V}, \mathcal{W})$ as well as its adjoint $\frac{\delta \mathcal{F}}{\delta v}^* : \mathcal{U} \times \mathcal{V} \to \mathcal{L}(\mathcal{W}, \mathcal{V})$.

3.2.1.1 Nonlinear elliptic PDE as a running example

As a running example, we consider the following nonlinear elliptic equation supplemented with periodic boundary conditions. We will use this PDE throughout the paper to ground our ideas and provide intuition for our abstract derivations:

$$\begin{cases} -\Delta v(x) + \kappa v(x)^3 = u(x) & \text{for } x \in (0, 1) \\ v(0) = v(1) \end{cases}. \tag{3.1}$$

The corresponding differential operator and its derivative are

$$\mathcal{F}(u, v) = -\Delta v + \kappa v^3 - u \qquad \text{and} \qquad \left[\frac{\delta \mathcal{F}}{\delta v}(u, v)\right](h) = [-\Delta + 3\kappa v^2](h) \tag{3.2}$$

respectively. Our task is to learn an approximation of $\mathcal{G}$ such that for $v = \mathcal{G}(u)$ the above equation is satisfied in some suitable sense, e.g., in the classical or weak sense. In other words, we aim to learn the solution operator that maps the right hand side u to the solution v of (3.1).

A suitable analytical framework may set $\mathcal{U} = \mathcal{W} = \mathrm{C}_{per}(\mathbb{T})$, the space of periodic continuous functions on $\mathbb{T}$, and either $\mathcal{V} = \mathrm{C}^2_{per}(\mathbb{T})$, the space of periodic twice differentiable functions on Ω for classical solutions, or $\mathcal{V} := H^1_{\mathrm{per},0}(\mathbb{T}) = \left\{u \in H^1_{\mathrm{per}}(\mathbb{T}) : \int_0^1 u = 0\right\}$ for weak solutions where $W := V^* \simeq H^{-1}_{\mathrm{per},0}(\mathbb{T})$ with $\|u\|_V := \|u'\|_{L^2(0,1)}$.

3.2.2 The Newton–Kantorovich method

The Newton–Kantorovich (NK) method [263] is a generalization of Newton's method to Banach spaces. First, we outline the method with the nonlinear elliptic PDE defined in (3.1) and (3.2). To solve this equation, we start from an initial guess v_0, and iteratively perturb the approximation using successive linearizations around each estimate. Writing v_n for the current estimate and $v_{n+1} := v_n + \delta v_n$ for the next estimate, we have:

$$\begin{aligned} 0 &= -\Delta(v_n + \delta v_n) + \kappa(v_n + \delta v_n)^3 - u \\ &= \left[-\Delta(v_n) + \kappa(v_n)^3 - u\right] + (-\Delta\delta v_n + 3\kappa v_n^2 \delta v_n) + o(\delta v_n) \\ &= \mathcal{F}(u, v_n) + (-\Delta + 3\kappa v_n^2)\delta v_n + o(\delta v_n). \end{aligned}$$

Ignoring higher orders of approximation, this defines the update δv_n as the solution of a v_n-dependent linear PDE. Our critical insight is that learning to solve this linearized PDE is much simpler than directly solving the original nonlinear PDE. This methodology can be generalized to any operator equation $\mathcal{F}(u, v) = 0$: We have

$$0 = \mathcal{F}(u, v_n + \delta v_n) \approx \mathcal{F}(u, v_n) + \left[\frac{\delta \mathcal{F}}{\delta v}(u, v_n)\right](\delta v_n) \tag{3.3}$$

where again $\frac{\delta \mathcal{F}}{\delta v}$ is the Fréchet derivative of $\mathcal{F}$. The NK method proceeds to find δv_n by solving the linear problem in (3.3) giving

$$\delta v_n = -\left(\frac{\delta \mathcal{F}}{\delta v}(u, v_n)\right)^{-1} \mathcal{F}(u, v_n). \tag{3.4}$$

Therefore to emulate the NK flow, it is sufficient to learn the term $\left(\frac{\delta \mathcal{F}}{\delta v}(u, v_n)\right)^{-1}$.

3.2.3 The Newton–Kantorovich method with Tikhonov regularization

Limited data and the potential ill-conditioning of the operator $\frac{\delta \mathcal{F}}{\delta v}$ can lead to inaccuracies and instabilities if we try to directly use the iteration in (3.4). To address this issue, it is necessary to regularize the problem. We therefore consider a Tikhonov-regularized variant of the NK method by defining

$$\delta v_n = \arg\min_{\delta v \in \mathcal{V}} \left\{ \left\| \mathcal{F}(u, v_n) + \frac{\delta \mathcal{F}}{\delta v}(u, v_n)[\delta v] \right\|_{\mathcal{W}}^2 + \lambda \|\delta v\|_{\mathcal{V}}^2 \right\} \tag{3.5}$$

where $\lambda > 0$ is the regularization parameter that balances robustness and accuracy; larger values of λ slow down convergence and improve robustness while smaller values lead to faster algorithms that are less stable. Assuming that the underlying

spaces are Hilbert, the first-order optimality condition for (3.5) yields a (linear) normal equations that can be solved explicitly. This results in the following iterative scheme

$$\delta v_n = -\Big(\big(\tfrac{\delta \mathcal{F}}{\delta v}(u, v_n)\big)^* \tfrac{\delta \mathcal{F}}{\delta v}(u, v_n) + \lambda I \Big)^{-1} \big(\tfrac{\delta \mathcal{F}}{\delta v}(u, v_n)\big)^* \mathcal{F}(u, v_n), \tag{3.6}$$

$$v_{n+1} = v_n + \delta v_n, \tag{3.7}$$

where $\frac{\delta \mathcal{F}}{\delta v}^*$ denotes the Hilbert adjoint of the Fréchet derivative of $\mathcal{F}$. This is precisely the Hilbert-space analogue of the Levenberg–Marquardt step [226]. We note that this iterative scheme leads to a connection with the attention mechanism from transformers [51, 338] that we highlight in Appendix B.7, leaving further investigation to future work.

In Banach spaces, a (unique) minimizer of (3.5) still exists under standard assumptions (e.g., uniform convexity/smoothness and suitable conditions on $\mathcal{F}$). The corresponding optimality conditions replaces inner products by duality pairings and involves the duality mappings J_V and J_W, leading to a nonlinear equation of the form

$$\big(\tfrac{\delta \mathcal{F}}{\delta v}(u, v_n)\big)^* J_W\big(\mathcal{F}(u, v_n) + \tfrac{\delta \mathcal{F}}{\delta v}(u, v_n)\, \delta v_n\big) + \lambda\, J_V(\delta v_n) = 0,$$

which can be solved by standard monotone-operator or Newton-type methods. For clarity, we confine our analysis to the Hilbert-space setting.

Since we assumed that the adjoint operator $\frac{\delta \mathcal{F}}{\delta v}^*$ is known, the most significant computational burden in computing the update δv_n is, in general, to compute $Q(u, v_n) = \Big(\big(\tfrac{\delta \mathcal{F}}{\delta v}(u, v_n)\big)^* \tfrac{\delta \mathcal{F}}{\delta v}(u, v_n) + \lambda I \Big)^{-1}$. For that reason, we only learn the parametric operator $(u, v) \mapsto Q(u, v)$. This operator is symmetric, positive definite and linear, but it depends nonlinearly on both u and v_n. The next remark justifies learning Q, as it can be done efficiently.

Remark 3.2.1 (Reduction to parametric elliptic operator learning)**.** *Let us write $\mathcal{L}_+(\mathcal{V}, \mathcal{V})$ for the space of symmetric positive linear operators from $\mathcal{V}$ to $\mathcal{V}$. We notice that Q is indeed a mapping from $\mathcal{U} \times \mathcal{V}$ to $\mathcal{L}_+(\mathcal{V}, \mathcal{V})$. Then, assuming that the adjoint operator $\frac{\delta \mathcal{F}}{\delta v}^*$ is known, we can compute $Q \frac{\delta \mathcal{F}}{\delta v}^*$. The operator Q^{-1} is indeed an elliptic differential operator whose solution map Q can be learned efficiently [54, 291].*

The subsequent remark shows that our analogue of the Levenberg–Marquardt algorithm, as in the finite-dimensional case, interpolates between gradient descent and the Newton–Kantorovich method.

Remark 3.2.2 (Interpolation between Gradient Descent and NK)**.** *In the Hilbert space setting, we can formally write*

$$\left\| \mathcal{F}(u, v_n) + \left[\frac{\delta \mathcal{F}}{\delta v}(u, v_n) \right] (\delta v) \right\|_{\mathcal{W}}^2 = \|\mathcal{F}(u, v_n)\|_{\mathcal{W}}^2 + 2 \left\langle \delta v, \left[\frac{\delta \mathcal{F}}{\delta v}(u, v_n) \right]^* \mathcal{F}(u, v_n) \right\rangle_{\mathcal{V}} + O(\|\delta v\|_{\mathcal{V}}^2). \tag{3.8}$$

and see that δv_n given by (3.6) *approximates the NK increment* (3.4) *as $\lambda \downarrow 0$ while it is proportional to a gradient descent update for the left-hand side of* (3.8) *as $\lambda \uparrow \infty$. Therefore, the regularization parameter λ allows us to interpolate between the NK algorithm and gradient descent, enabling a controlled balance between convergence stability and update accuracy.*

3.2.4 CHONKNORIS

We are now ready to introduce CHONKNORIS based on the idea of using operator learning to efficiently learn the solution operator Q as mentioned in Remark 3.2.1. More precisely, CHONKNORIS uses the same update rule as in (3.7), but replaces the costly operator inversion in the definition of Q with a learned approximation. In practice, we work with an arbitrary discretization of input and output spaces $\mathcal{U}, \mathcal{V}$, and $\mathcal{W}$. This discretization can be derived from various numerical methods such as finite-elements, finite-differences, spectral methods, or any other discretization method. Thus, choosing an appropriate parametrization of our input and output functions, CHONKNORIS can be made discretization-invariant following the optimal recovery approach in [27] or the Fourier neural operator (FNO) framework in [184]. Consequently, $Q(u, v_n)$ is discretized as a positive definite matrix. We choose to approximate the Cholesky factors of Q to enforce positivity, which stabilizes learning, guarantees descent directions, reduces the number of learned parameters, and allows for efficient triangular solves. We train a surrogate operator $\hat{\mathcal{R}}$, parameterized for instance as a neural operator or a kernel-based model, such that

$$\hat{\mathcal{R}}(u, v)\hat{\mathcal{R}}(u, v)^T \approx Q(u, v) = \left[\left(\frac{\delta \mathcal{F}}{\delta v}(u, v) \right)^* \left(\frac{\delta \mathcal{F}}{\delta v}(u, v) \right) + \lambda I \right]^{-1} \tag{3.9}$$

$$\hat{\mathcal{R}}(u, v) \text{ is upper triangular.} \tag{3.10}$$

To that end, we use a traditional NK solver to generate training data for our method. Using input data $u^{(1)}, \ldots, u^{(M)} \sim \mu$ that are sampled from a probability measure μ supported on $\mathcal{U}$, we run the NK solver for $n_{warm} \in \mathbb{N}$ steps to generate flow data

$v_k^{(m)}$ where $m = 1, \ldots, M$ and $k = 0, \ldots n_{warm}$. This flow data lives along the *true* NK method trajectory to mitigate the curse of dimensionality and avoid generating data that is not seen in the NK iteration scheme. We can then compute the Cholesky factors of $Q(u^{(m)}, v_k^{(m)})$ and train a parameterized $\hat{\mathcal{R}}_\theta$ model to approximate them, e.g., by using the usual regularized mean squared error loss

$$\min_\theta \sum \left\| \hat{\mathcal{R}}_\theta(z, \lambda) - R \right\|_F^2.$$

Note that it is also possible to vary the Tikhonov regularization λ, introducing an additional dependency for $\hat{\mathcal{R}}$, i.e., $\hat{\mathcal{R}}(u, v, \lambda)$. This is particularly useful for ill-posed problems, such as inverse problems, to speed up the convergence and obtaining higher accuracy.

Once we have trained $\hat{\mathcal{R}}$, the CHONKNORIS approximation is $\hat{\mathcal{G}} : u \mapsto \hat{v}_N$, where

$$\hat{v}_{n+1} = \hat{v}_n - \alpha_n \hat{\mathcal{R}}(u, \hat{v}_n, \lambda_n) \hat{\mathcal{R}}(u, \hat{v}_n, \lambda_n)^T \left(\frac{\delta \mathcal{F}}{\delta v}(u, \hat{v}_n) \right)^* \mathcal{F}(u, \hat{v}_n), \quad n = 0, \ldots, N-1. \tag{3.11}$$

For a sufficiently large number or inference iterations N, we expect $\hat{\mathcal{G}}(u) = \hat{v}_N \approx \mathcal{G}(u)$. Here, α_n is a step size schedule and λ_n the Tikhonov regularization schedule. Both schedules may be chosen adaptively, e.g., using line-search or the more robust inflation/deflation scheme we detail in Appendix B.5. Thus, CHONKNORIS is an emulator of the *true* NK method. We summarize the resulting algorithm in Algorithm 1 and give a visual depiction of the post-training operator in Figure 3.1.

3.2.5 FONKNORIS

In this section, we introduce FONKNORIS, a foundational model variant of CHONKNORIS. We illustrate this method using our running example of a nonlinear elliptic PDE. First, we notice that in 1D the Fréchet derivative of the nonlinear elliptic PDE in (3.2), $\frac{\delta \mathcal{F}}{\delta v}(u, v) = (-\partial_{xx} + 3\kappa v^2)$, can be expressed as

$$\left[\frac{\delta \mathcal{F}}{\delta v}(u, v) \right] (h)(x) = \left[a(u, v)\partial_{xx} + b(u, v)\partial_x + c(u, v) \right] h(x), \tag{3.12}$$

for coefficient functions $a = -1$, $b = 0$, $c = 3\kappa v^2$. While CHONKNORIS learns to predict the Cholesky factors of the Tikhonov-regularized inverse of this operator as a function of v_n and u, i.e., $\hat{\mathcal{R}}(u, \hat{v}_n)$, FONKNORIS aims to learn the same Cholesky factors as a function of the coefficient functions a, b, and c, i.e., $\hat{\mathcal{R}}(a, b, c)$. This implies that, once the model $\hat{\mathcal{R}}(a, b, c)$ is trained on a sufficiently large and diverse dataset of (a, b, c) combinations, we are able to predict the Cholesky factors arising

Algorithm 1 CHONKNORIS/FONKNORIS

Inputs: training data size M; residual map $\mathcal{F}$; Jacobian $J(u,v) = \frac{\delta \mathcal{F}}{\delta v}(u,v)$; adjoint J^*; measure μ; NK warm-up steps n_{warm}; model $\hat{\mathcal{R}}_\theta$ (lower-triangular, diag > 0); flow relaxation λ_{flow}; training relaxation set λ_{train}

Mode: choose *parameterization* Φ and *Jacobian builder* BuildJ:

- CHONKNORIS: $\Phi(u,v) = (u,v)$, $\mathsf{BuildJ}(u,v) = J(u,v)$
- FONKNORIS: $\Phi(u,v) = (a(u,v), b(u,v), c(u,v))$, $\mathsf{BuildJ}(a,b,c) = a\,\partial_{xx} + b\,\partial_x + c$

Data (offline):
for $m = 1, \ldots, M$ **do**
  sample $u^{(m)} \sim \mu$
  initialize v_0
  for $i = 0, \ldots, n_{\text{warm}}$ **do** ▷ solver warmup
    $z_i \leftarrow \Phi(u^{(m)}, v_i)$ ▷ $z_i = (u^{(m)}, v_i)$ in CHONKNORIS, or $z_i = (a,b,c)$ in FONKNORIS
    $J_i \leftarrow \mathsf{BuildJ}(z_i)$
    $v_{i+1} \leftarrow v_i - \left(J_i^* J_i + \lambda_{\text{flow}} I\right)^{-1} J_i^* \mathcal{F}(u^{(m)}, v_i)$
    for $\lambda \in \lambda_{\text{train}}$ **do**
      $R = \left(\text{chol}_{\text{lower}}(J_i^* J_i + \lambda I)\right)^{-*}$ ▷ $RR^* = (J_i^* J_i + \lambda I)^{-1}$
      store training data $\left(z_i, \lambda, R\right)$
    end for
  end for
end for
Learn (offline): $\min_\theta \sum \left\| \hat{\mathcal{R}}_\theta(z,\lambda) - R \right\|_F^2$; enforce lower-triangular with positive diagonal.
Evaluate (online): given u, v_0, budget N; initial values $\alpha_{\text{predict}}/\lambda_{\text{train}}$.
for $n = 0, \ldots, N-1$ **do**
  $r \leftarrow \mathcal{F}(u, v_n)$
  $z_n \leftarrow \Phi(u, v_n)$
  $J_n \leftarrow \mathsf{BuildJ}(z_n)$
  $R \leftarrow \hat{\mathcal{R}}_\theta(z_n, \lambda_n)$
  $\delta v \leftarrow -\alpha_n R^* R J_n^* r$
  $v_{n+1} \leftarrow v_n + \delta v$
  choose α_n, λ_n by repeating the above steps until $\|\mathcal{F}(u, v_{n+1})\| \ll \|r\|$ ▷ e.g., using line-search
  Stop if $\|r\|$ and/or $\|\delta v\|$ below given tolerances
end for
Return $\hat{v} = v_N$

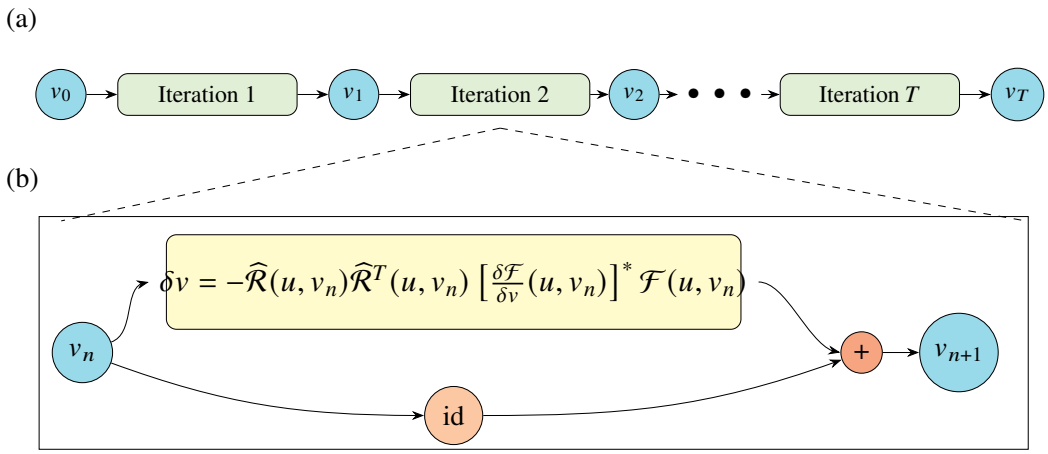


Figure 3.1: CHONKNORIS. (a) An initial guess v_0 for the true solution $v = \mathcal{G}(u)$ is iteratively refined by adding a correction term. (b) Each iteration consists of two steps: First, compute the correction term $r = \left[\frac{\delta \mathcal{F}}{\delta v}\right]^* \mathcal{F}$ and then $\delta v = -\left(\widehat{\mathcal{R}}\widehat{\mathcal{R}}^* r\right)(u, v_k)$, where $\widehat{\mathcal{R}}$ is a learned surrogate for the Cholesky factors of $(\frac{\delta \mathcal{F}}{\delta v}\frac{\delta \mathcal{F}}{\delta v}^T + \lambda I)^{-1}$, $\mathcal{F}$ is the forward map with its Fréchet derivative $\frac{\delta \mathcal{F}}{\delta v}$, v_k is the current approximation of the desired function $v = G(u)$, and u is the input for which we seek the solution. Next, update the current approximation via $v_{k+1} = v_k + \delta v$.

from any PDE whose Fréchet derivative is of the form (3.12), i.e., any local PDE of second order. We note that the coefficient functions a, b, and c depend on and are uniquely determined by the operator $\mathcal{F}$.

To this end, FONKNORIS seeks to train a data-driven surrogate operator $\hat{\mathcal{R}}$, such as a neural operator or a kernel-based model, where

$$\hat{\mathcal{R}}(a, b, c)\hat{\mathcal{R}}(a, b, c)^T \approx \left([a\partial_{xx} + b\partial_x + c]\,[a\partial_{xx} + b\partial_x + c]^* + \lambda I\right)^{-1},$$
$$\hat{\mathcal{R}}(a, b, c) \text{ is lower triangular.}$$

This model is *trained once, and generalizes to any equation* with a Jacobian of the form (3.12). Then, the FONKNORIS approximation is $\hat{\mathcal{G}} : u \mapsto \hat{v}_N$, where

$$\hat{v}_{n+1} = \hat{v}_n - \alpha_n \hat{\mathcal{R}}(a_n, b_n, c_n)\hat{\mathcal{R}}(a_n, b_n, c_n)^T\,[a_n\partial_{xx} + b_n\partial_x + c_n]^*\,\mathcal{F}(u, \hat{v}_n)$$
$$\text{where } a_n = a(u, v_n),\ b_n = b(u, v_n),\ c_n = c(u, v_n).$$

Again, for a sufficiently large number or iterations N, we expect $\hat{\mathcal{G}}(u) = \hat{v}_N \approx \mathcal{G}(u)$. Note that for FONKNORIS, we add an intermediate step that, given u and v_n, computes the coefficient functions $a_n = a(u, v_n)$, $b_n = b(u, v_n)$, $c_n = c(u, v_n)$ and passes them to the model $\hat{\mathcal{R}}$, hence we assume knowledge of these functions.

Furthermore, we note that this 1D example can be easily extended to any dimension and to any nonlinear operator. For local differential operators of order $k \in \mathbb{N}$, we

observe that

$$\frac{\partial \mathcal{F}}{\partial v}(u, v) = \sum_{|\alpha| \leq k} \beta^{\alpha}(u, v) D^{\alpha}$$

where $\alpha \in \mathbb{N}^k$ is a multi-index, D^{α} denotes a linear differential operator, and β^{α} denotes the coefficient functions. For nonlinear operators that are nonlocal, the Fréchet derivative also contains linear nonlocal operators. However, the principle remains the same: The Fréchet derivative can still be parameterized in terms of the coefficient functions. FONKNORIS can be regarded as a *foundation model* for local differential equations, as it is trained once on a diverse collection of PDEs through their coefficient functions β^{α} and generalizes to unseen equations without retraining. By learning a universal mapping from operator coefficients to inverse operators, FONKNORIS provides a reusable model that captures the shared structure underlying broad classes of PDEs. The FONKNORIS algorithm is summarized in Algorithm 1.

3.3 Numerical Experiments

In this section we present various numerical experiments that verify the ability of CHONKNORIS and FONKNORIS in emulating various PDE and inverse problem solution maps to machine precision. We note that the reported machine-precision errors should be interpreted with respect to the *discrete reference solutions* produced by the underlying finite-difference solver that we use for our examples, not with respect to the exact continuous PDE solution or inverse problems. Therefore, the truncation error of the finite-difference discretization is not included in the reported error; rather, our method is aimed to perfectly emulate the classical solver. In this sense, the small errors indicate that the proposed method can emulate the chosen discrete solver to near machine precision. In the experiments, the spatial discretization is based on central finite differences, while time integration is performed with an implicit time-stepping scheme. CHONKNORIS is then applied at each time step to reproduce the corresponding discrete update.

Furthermore, we note that for the forward problems, we mainly adopt a linearize-then-discretize perspective, whereas for the inverse problems we use a discretize-then-linearize perspective for practical reasons. For the forward problems, linearizing first is natural within the Newton–Kantorovich framework, since the method is formulated at the operator level in function space. This allows us to interpret each iteration as the solution of a linearized PDE operator and then discretize that linear

problem using a numerical method of choice. One advantage of this viewpoint is that it aligns with the operator-learning formulation and facilitates the use of discretization- or resolution-invariant architectures. For the inverse problems, by contrast, discretizing first is more convenient computationally, because the relevant parameter-to-observable map is implicit at the continuous level and is most naturally handled through its discrete realization. This makes it possible to compute the required derivatives and adjoints directly at the computational level, for example by automatic differentiation. There are also fundamental differences between the two viewpoints. In general, linearization and discretization do not commute exactly, so linearize-then-discretize and discretize-then-linearize may lead to different discrete operators. The former is often more natural for analysis and for preserving the continuous operator interpretation, whereas the latter is often preferred in practice because it is directly tied to the implemented numerical scheme.

3.3.1 Benchmarking Summary

To assess the performance of the CHONKNORIS method, we benchmark against the kernel/Gaussian process (GP) operator learning framework of [27], Fourier neural operators (FNOs) [184], and transformer neural operator (TNO) [51]. We deploy these three models as purely data-driven baselines which learn the operator $\mathcal{G}$ directly from input-output pairs (u, v). Note that v is given by the last iterate of the Newton–Kantorovich method, so the baseline methods do not have access to the intermediate steps of the solver. These baselines are not intended as a comprehensive benchmark; rather, they serve as a control to verify that conventional operator-learning frameworks do not attain machine precision on these problems. Furthermore, we note that the benchmark methods are trained in the low-data regime detailed in the following subsections, justifying the varying performance of the baselines. We also note that the vanilla FNO and vanilla TNO do not support the different input and output domains which arise in our benchmark inverse problems as detailed in Section 3.3.4; hence, we do not apply these benchmarks to the inverse problems we test here. Table 3.1 gives a performance comparison between these benchmark operator learning methods and our proposed CHONKNORIS approach. The details of the benchmark problems are outlined in the remainder of this section. Additional details on the GP baselines are given in Appendix B.1 along with an inference time comparison in Table B.1.

	GP	FNO	TNO	CHONKNORIS (ours)
Nonlinear elliptic	2.4e-5	1.3e-3	5.7e-3	8.9e-16
Burgers'	1.1e-1	6.0e-3	2.2e-2	5.1e-16
Nonlinear Darcy	1.8e-3	4.8e-3	3.9e-3	9.6e-16
Calderòn	1.6e-2	–	–	3.2e-15
Inverse wave scattering	1.5e-2	–	–	9.2e-13
Seismic imaging 5×5	2.2e-2	–	–	2.6e-14
Seismic imaging 7×7	4.1e-2	–	–	3.0e-12
Seismic imaging 10×10	5.0e-2	–	–	1.2e-03

Table 3.1: Summary of median relative L^2 losses over multiple forward and inverse problem benchmarks.

3.3.2 Forward Problems

Table 3.2 summarizes the forward problems we consider and their parameterizations. The following subsections provide additional details on each of these forward problems.

forward problem	Nonlinear Elliptic 1D	Burgers' 1D time-dependent	Darcy 2D
u distrib.	GP-periodic	sum of sines	GP-invLaplacian
FD grid	$N_x = 63$	$(N_t, N_x) = (151, 127)$	$(N_{x_1}, N_{x_2}) = (20, 20)$
$(R_{\text{train}}, R_{\text{val}})$	$(896, 128)$	$(448, 64)$	$(896, 128)$
$(n_{\text{warm}}, \lambda_{\text{flow}}, \lambda_{\text{train}})$	$(5, 0, 0)$	$(5, 0, 10^{-2})$	$(6, 0, 10^{-3})$
Hessian model	GP-Gaussian	MLP-Tanh $(127, 500, 1000, 8128)$	GP-Gaussian
Section	Section 3.3.2.1	Section 3.3.2.2	Section 3.3.2.3
inverse problem	Calderòn	Inverse Wave Scattering	Seismic Imaging FWI
u distrib.	GP-invLaplacian	GP-invLaplacian	OpenFWI dataset [82]
FD grid	$(N_{x_1}, N_{x_2}) = (9, 9)$	$(N_{x_1}, N_{x_2}) = (7, 7)$	$(N_{x_1}, N_{x_2}) \in \{(5, 5), (7, 7), (10, 10), (14, 14)\}$
$(R_{\text{train}}, R_{\text{val}})$	$(7500, 2500)$	$(9750, 250)$	$(800, 200)$
$(n_{\text{warm}}, \lambda_{\text{flow}}, \lambda_{\text{train}})$	$(0, 10^{-10}, 10^{-10})$	$(0, 10^{-4}, 10^{-4})$	(400, adaptive, adaptive)
Hessian model	GP–Gaussian	GP–Gaussian	GP–Gaussian
Section	Section 3.3.4.1	Section 3.3.4.2	Section 3.3.4.3

Table 3.2: Forward and inverse problem parameters. u distrib. is the distribution of random coefficients. FD grid contains the regular grid sizes for the finite difference scheme. $(R_{\text{train}}, R_{\text{val}})$ are the number of training and validation realizations respectively. n_{warm} denotes the number of Newton–Kantorovich iterations used to generate training data along the flow. λ_{flow} is the relaxation used in the NK method, while λ_{train} is the relaxation shown to the Hessian prediction model. CHONKNORIS training is performed with $\Lambda_{\text{train}} = \{\lambda_{\text{flow}}\}$ in the context of Algorithm 1. GP-kernel denotes a Gaussian process with the given kernel, with the invLaplacian kernel given by $5(-\Delta + 1/100)^{-2}$ where $-\Delta$ denotes the Laplacian. MLP-nonlinearity is a multi-layer perceptron neural network architecture where the tuple specifies layer sizes, including input and output layers, and the given nonlinearity is applied to all hidden layers. Note that for Burgers' equation, $n_{\text{warm}} = 5$ NK steps were run per time step.

3.3.2.1 Nonlinear Elliptic Equation

The first example is the 1D nonlinear elliptic PDE (3.1) with $\kappa = 50$ as introduced in Section 3.2.

The differential operator is given in (3.2) and the corresponding solution operator is given by $\mathcal{G}(u) = v$ mapping the right hand side to the solution of the boundary value problem Section 3.2. The random coefficient u is sampled from a zero-mean GP with a periodic kernel

$$K(x, x') = \exp(-2/\ell \sin^2(\pi/p(x - x')))$$

with period length $p = 1/2$ and lengthscale $\ell = 10$. Here we fit two operator learning models. The first is an end-to-end operator learning model whose prediction is used as an initial guess for the NK/CHONKNORIS method. The second is our CHONKNORIS predictor model for the Cholesky factor. For both models we use vector-valued GP regression with Matérn kernels with smoothness parameter $\nu = 5/2$ (which we abbreviate as $5/2$ Matérn) and squared exponential kernels, all with tuned lengthscales.

3.3.2.2 Burgers' Equation

The second example is the 1D time-dependent Burgers' equation, supplemented with periodic boundary conditions and a random initial condition, which may be written as

$$\begin{cases} \partial_t f = \nu \partial_{xx} f - f \partial_x f, & (x, t) \in \mathbb{T} \times [0, T], \\ f(0, t) = f(1, t), & t \in [0, T], \\ f(x, 0) = f_0(x), & x \in \mathbb{T} \end{cases} . \tag{3.13}$$

Here $\nu = 1/50$, f_0 is the initial condition and ∂_t, ∂_x, ∂_{xx} are partial derivatives. Here, we discretize the PDE in time and learn the solution operator of the time-discrete problem. More precisely, we discretize $[0, T]$ using the M-point uniform grid $\{t_i\}_{i=0}^{M-1} := \{i\Delta t\}_{i=0}^{M-1}$, $\Delta t = T/(M - 1)$, and apply an implicit Euler time discretization leading to

$$\frac{f^{i+1}(x) - f^i(x)}{\Delta t} = \nu \partial_{xx} f^{i+1}(x) - f^{i+1}(x) \partial_x f^{i+1}(x)$$

where $f^i(x) = f(x, t_i)$ and $\Delta t = T/(M - 1)$. Thus, the time marching discretization scheme defines the next time step f^{i+1} given f^i so that $\mathcal{F}(f^i, f^{i+1}) = 0$ with

$$\begin{aligned} \mathcal{F}(u, v) &= v - \Delta t \left(\nu \partial_{xx} v - v \partial_x v\right) - u, \\ \frac{\delta \mathcal{F}}{\delta v}(u, v)[h] &= h - \Delta t \left(\nu \partial_{xx} h - \partial_x v h - v \partial_x h\right). \end{aligned} \tag{3.14}$$

Notice that $\delta\mathcal{F}/\delta v$ does *not* depend on u. The resulting solution operator is given by $\mathcal{G}(u) = v$ mapping the previous time step $u = f^i$ to the next time step $v = f^{i+1}$.

Here, CHONKNORIS approximates the solution operator $\mathcal{G}$ which in turn is used iteratively to obtain the solution for all time steps. We take the random initial condition to be

$$f_0(x) = \sum_{k=1}^{3} a_k \sin(\pi k x), \qquad (a_1, \ldots, a_3) \sim \mathcal{N}(0, 1).$$

3.3.2.3 Nonlinear Darcy Flow

The third example is the 2D Darcy flow equation, supplemented with homogeneous Dirichlet boundary conditions, which may be written as

$$\begin{cases} -\nabla \cdot (e^u \nabla v) + v^3 = f, & x \in [0,1]^2 \\ v = 0, & x \in \partial[0,1]^2 \end{cases} \tag{3.15}$$

with forcing term f and conductivity e^u. Expanding $-\nabla\cdot(e^u\nabla v) = -e^u[\nabla u\cdot\nabla v+\Delta v]$, we have,

$$\begin{aligned} \mathcal{F}(u, v) &= -e^u[\nabla u \cdot \nabla v + \Delta v] + \kappa v^3 - f, \\ \left[\frac{\delta \mathcal{F}}{\delta v}(u, v)\right](h) &= -e^u[\nabla u \cdot \nabla h + \Delta h] + 3\kappa v^2 h. \end{aligned} \tag{3.16}$$

Here, the solution operator is given by $\mathcal{G}(u, f) = v$ which maps the forcing f and the conductivity u to the solution v. In that sense, CHONKNORIS is learning a parametric family of solution operators $\mathcal{G}_u(\cdot) = \mathcal{G}(u, \cdot)$. We take the fixed forcing term f to be a single draw from a zero-mean GP with a $5/2$ Matérn kernel and constant lengthscale of $3/10$ across both dimensions.

3.3.2.4 Summary of the numerical results for CHONKNORIS on the forward problems

We used CHONKNORIS to emulate and solve the three problems above. The domains were discretized using regular grids, and derivatives were approximated using standard finite differences. A traditional NK solver was used to generate the training data, and the reference solutions with which the errors were computed. The Cholesky factors for the nonlinear elliptic PDE in Section 3.3.2.1 and the Darcy flow problem in Section 3.3.2.3 were predicted using a GP, while for the Burgers' equation in Section 3.3.2.2 we used a Multi-Layer Perceptron (MLP).

In all tested instances of the three problems, CHONKNORIS consistently achieves machine precision error. We study the speed of convergence in the nonlinear elliptic

PDE in Section 3.3.2.1, see Figure 3.2 (a), and found that CHONKNORIS starting from an initial guess of 0 typically converges in 10 iterations while predicting the approximate Hessian, whereas the traditional NK solver needs only 4 iterations using the precise approximate Hessian. We also observed that the convergence of the RMSE residual $\|\mathcal{F}(u, v_n)\|$, which we could track, follows the desired convergence of L^2 relative errors $\|v_n - v\|$ to machine precision (which is not generally known except when the true solution v is known as is the case here). We observed that providing an initial guess from an end-to-end operator learning approach gives modest speedups of one to two iterations to both the NK and CHONKNORIS iterations.

We tested the robustness of our method with the Burgers' problem in Section 3.3.2.2, see Figure 3.2 (b), and found that our method can achieve machine precision even when solutions contain shocks. Finally, the Darcy flow problem in Section 3.3.2.3 was the most challenging of the forward benchmarks as it required a lot of data to learn the parametric Cholesky factors. This experiment showcases the capacity of our method to utilize additional compute to achieve convergence even in difficult settings. As shown in Figure 3.2 (c), within 10 iterations CHONKNORIS usually reaches L^2 relative errors on the order of 10^{-3}, putting it on par with existing operator learning models. After 100 iterations, errors are typically on the order of 10^{-6} or better. Continuing to increase the number of CHONKNORIS iterations to 1000 enables convergence to machine precision in 95% of all cases. In this example, we observed that realizations from a rougher distribution require significantly more CHONKNORIS iterations for exact recovery.

3.3.2.5 Klein–Gordon and Sine–Gordon

For our last two forward problem examples, we consider the 1D Klein–Gordon and the Sine–Gordon equations. We will use these problems as a held out validation problem for testing the generalization error of FONKNORIS. When supplemented with initial and boundary condition, both PDEs take the form of

$$\begin{cases} \partial_{tt} f = \kappa_1 \partial_{xx} f - \kappa_2 \tau(f), & (x,t) \in \mathbb{T} \times [0,T], \\ f(0,t) = f(1,t), & t \in [0,T], \\ f(x,0) = f_0(x), & x \in \mathbb{T} \end{cases} . \tag{3.17}$$

Here $x \in \Omega = \mathbb{T}$ and $t \in [0,T]$, f_0 is the initial condition, and τ is a nonlinearity. For Klein–Gordon, $\tau(f) = f^3$, $\kappa_1 = 0.1$, $\kappa_2 = 10$, while for Sine–Gordon $\tau(f) = \sin(f)$

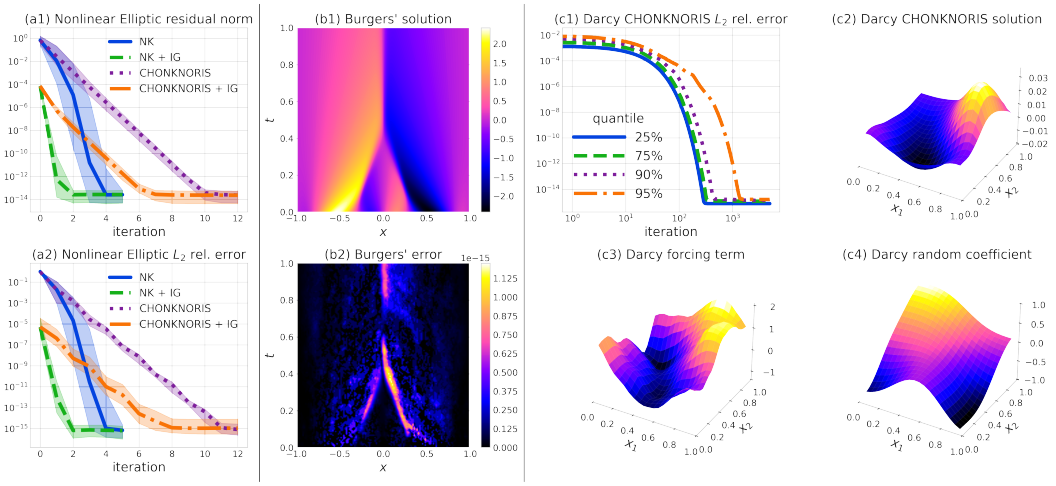


Figure 3.2: Forward problems. (a) Results for the nonlinear elliptic PDE problem. Quantiles of 10% – 90% are shown across test realizations. Our CHONKNORIS method is able to achieve machine precision accuracy in around 10 iterations. (b) Results for Burgers' equation. CHONKNORIS was able to achieve machine precision error in recovering the discretized solution which contained shocks. (c) Results for the Darcy flow PDE: (c1) shows that more challenging realizations require more CHONKNORIS iterations. (c4) shows a single realization of the random coefficient with the corresponding solution in (c2). (c3) shows the fixed forcing term.

and $\kappa_1 = \kappa_2 = 1$. Similar to the time marching scheme for Burgers' equation in Section 3.3.2.2, we discretize $[0, T]$ and approximate $\partial_{tt} f \approx (f^{i+2} - 2f^{i+1} + f^i)/\Delta t^2$. Knowing the state $u := (u_1, u_2) := (f^{i+1}, f^i)$, we want to compute the next time step $v := f^{i+2}$ defined as $\mathcal{F}((f^{i+1}, f^i), f^{i+2}) = 0$ with

$$\mathcal{F}(u, v) := v - 2u_1 + u_2 - (\Delta t)^2 (\nu \partial_{xx} v - \kappa_2 \tau(v)), \tag{3.18}$$

$$\left[\frac{\delta \mathcal{F}}{\delta v}(u, v)\right](h) = h - (\Delta t)^2 \left(\kappa_1 \partial_{xx} h - \kappa_2 \frac{\delta \tau}{\delta v}(v)[h]\right), \tag{3.19}$$

where $u = (u_1, u_2)$. For Klein–Gordon, $\frac{\delta \tau}{\delta v}(v)[h] = 3v^2 h$, and for Sine–Gordon $\frac{\delta \tau}{\delta v}(v)[h] = \cos(v)h$. We note again, that $\delta \mathcal{F}/\delta v$ is independent of u. The resulting solution operators are in both cases given by $\mathcal{G}(u) = v$ which map the previous time steps $u = (u_1, u_2) = (f^{i+1}, f^i)$ to the next time step $v = f^{i+2}$. Again, CHONKNORIS approximates the solution operator $\mathcal{G}$ which is used in the very same time marching scheme to obtain the solution for all time steps.

FONKNORIS	Partial differential equation	a	b	c
training PDEs	Nonlinear elliptic (3.2)	-1	0	$3\kappa v^2$
	Burgers' (3.14)	$-(\Delta t)v$	$(\Delta t)v$	$1+(\Delta t)\nabla v$
	Nonlinear Darcy flow (3.16)	$-e^u$	$-e^u\nabla u$	$3\kappa v^2$
testing PDEs	Sine–Gordon (3.18)/(3.19)	$-\kappa_1(\Delta t)^2$	0	$1+\kappa_2(\Delta t)^2\cos(v)$
	Klein–Gordon (3.18)/(3.19)	$-\kappa_1(\Delta t)^2$	0	$1+3\kappa_2(\Delta t)^2v^2$

Table 3.3: FONKNORIS coefficients.

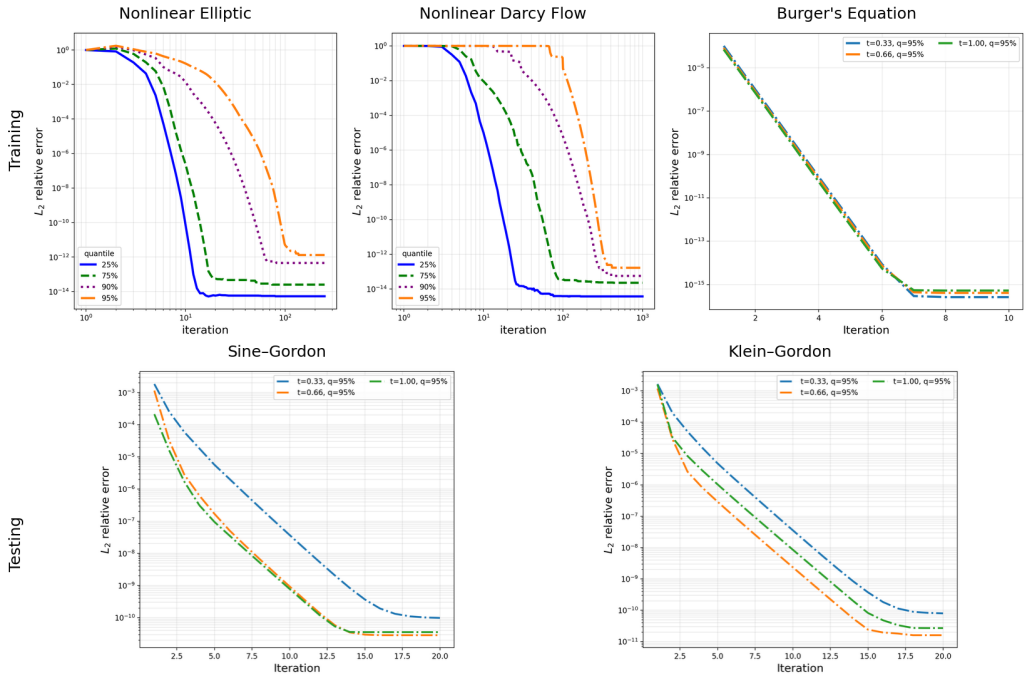


Figure 3.3: Quantiles of predictions of FONKNORIS for 100 realizations of initial conditions, external forces, and conductivities using a mixture of experts consisting of GPs for nonlinear elliptic, nonlinear darcy flow, and Burgers' equation and testing it for withheld Sine–Gordon and Klein–Gordon equations.

3.3.3 Foundation Model - FONKNORIS

For the FONKNORIS model, we train sub-model experts, each specific to a given PDE, and aggregate individual CHONKNORIS sub-models within mixture of experts [280]. The aggregation of the sub-models is done by the so-called nested Kriging method [280], see also Appendix B.2. This method relies on the fact that each sub-model has the same underlying Gaussian process and thus share the same kernel. In our FONKNORIS prediction framework, we employ Gaussian processes to model and predict the behavior of specific PDEs for each expert model. Constructing a single foundational Gaussian process model capable of representing a large class of PDEs would require an extensive dataset, which would quickly exceed

typical computational and memory limits. To overcome this limitation, we train each expert model independently and subsequently aggregate the individual experts within a foundational (meta) model.

3.3.3.1 Data Generation and Training

In our numerical experiments for FONKNORIS, we trained separate instances of the model $\hat{\mathcal{R}}_\theta$ for the one-dimensional nonlinear elliptic equation, Burgers' equation, and the one-dimensional nonlinear Darcy flow, using Matérn kernels with learned length scales ℓ_1, ℓ_2, and ℓ_3 for the input distributions μ. For the aggregated model, we used the arithmetic mean of the length scales, $\ell_{\mathcal{A}} = \frac{1}{p}\sum_{i=1}^{p} \ell_i$. The Sine–Gordon and Klein–Gordon equations were held out for testing. All PDEs were discretized with $N_x = 64$ collocation points. In the Newton–Kantorovich warm-up step of Algorithm 1 we used a fixed flow relaxation parameter $\lambda_{\text{flow}} = 0.01$ and the same value in the training relaxation set, $\lambda_{\text{train}} = 0.01$.

For the nonlinear elliptic equation and Burgers' equation, we generated $M_{\text{ell}} = 5000$ and $M_{\text{Burg}} = 5000$ offline realizations, respectively. The realizations for the nonlinear elliptic equation were generated by drawing 1000 external forcings f from a periodic kernel defining μ and running $n_{\text{warm}} = 5$ Newton–Kantorovich iterations. The realizations for Burgers' equation were generated by drawing 1000 initial conditions u_0 from a periodic kernel and again running $n_{\text{warm}} = 5$ Newton–Kantorovich iterations. For the nonlinear Darcy flow, we generated $M_{\text{Darcy}} = 10000$ realizations by drawing 2000 pairs of conductivities a and external forcings f from a periodic kernel and running $n_{\text{warm}} = 5$ Newton–Kantorovich iterations for each pair (a_i, f_i). The aggregation model was tested on the Klein–Gordon and Sine–Gordon equations by drawing realizations of the initial condition u_0 from a periodic kernel and choosing $v_0 = 0$ in the online phase of Algorithm 1. The data for training and testing were generated from the same distributions.

3.3.3.2 Summary of the numerical results for FONKNORIS on the forward problems

We applied FONKNORIS to the above problems, using the nonlinear elliptic, the Burgers', and the nonlinear Darcy flow equations as training problems, and testing on all five problems, including the Klein–Gordon and Sine–Gordon equations. In Figure 3.3, we see that the aggregated model $M(x)$ not only achieves machine precision on the trained instances, but also achieves near machine precision for the

withheld Klein–Gordon and Sine–Gordon equation. Table 3.3 shows the FONKNORIS coefficients for the aforementioned forward problems. The advantage of this approach is its simplicity and generating training data does not rely on a specific PDE as the coefficients a, b, c can be generated from a certain distribution, optimally from various distributions. The main difficulty is that the model can become very large and the coefficients a, b, c can easily get out of distribution when computing each Newton–Kantorovich step; one deals with the curse of dimensionality as the generated data might not inherit the statistics of a given PDE. Therefore, we generate the training data from the first three problems and generalize it to all problems including the Sine–Gordon and Klein–Gordon equation. Here, our mixture-of-experts Gaussian process model was capable of handling the large scale data necessary to generalize across PDE problems. This ability to generalize to unseen problems at machine precision is a state-of-the-art achievement that is due to the unique design of the FONKNORIS model.

3.3.4 Inverse Problems

In this section, we introduce various inverse problems which will be used to further test the performance of CHONKNORIS as an emulator for PDE constrained optimization. These include the Calderòn problem (Section 3.3.4.1), the inverse wave scattering problem (Section 3.3.4.2), and a problem in seismic imaging full waveform inversion (Section 3.3.4.3). Specific parameterizations for each of these problems are given in Table 3.2. The following subsections further detail our setup.

3.3.4.1 The Calderòn problem

Let $\Omega = [0,1]^2$ and assume $v \in L^\infty(\Omega)$ is a real-valued conductivity with $v(x) > 0$ for almost every $x \in \Omega$. Consider the boundary value problem

$$\begin{cases} \nabla \cdot (v(x)\nabla c(x)) = 0 & \text{for } x \in \Omega, \\ c(x) = g(x) & \text{for } x \in \partial\Omega, \end{cases} \tag{3.20}$$

where on the boundary, we prescribe a boundary voltage pattern $g \in H^{\frac{1}{2}}(\partial\Omega)$. We define the *Dirichlet-to-Neumann* (DtN) map $\Lambda_g : H^{\frac{1}{2}}(\partial\Omega) \to H^{-\frac{1}{2}}(\partial\Omega)$ that maps the boundary voltage pattern g to the current flux $v\frac{\partial c}{\partial n}\Big|_{\partial\Omega}$. Note that $\Lambda_g \in \mathcal{L}(H^{\frac{1}{2}}(\partial\Omega), H^{-\frac{1}{2}}(\partial\Omega))$ is a linear bounded operator. The *Calderòn problem* is the task of recovering the conductivity v from a given DtN map Λ_g. Denote with $\tilde{\mathcal{F}} : L^\infty(\Omega) \to \mathcal{L}(H^{\frac{1}{2}}(\partial\Omega), H^{-\frac{1}{2}}(\partial\Omega))$ the forward operator that maps the

conductivity to the Dirichlet-to-Neumann map. Then, the operator of interest is given by

$$\mathcal{F}(\Lambda_g, v) = \tilde{\mathcal{F}}(v) - \Lambda_g.$$

CHONKNORIS aims to approximate the operator $\mathcal{G} : \mathcal{L}(H^{\frac{1}{2}}(\partial\Omega), H^{-\frac{1}{2}}(\partial\Omega)) \to L^\infty(\Omega)$ that maps the DtN map to the conductivity, i.e., $\mathcal{G}(\Lambda_g) = \tilde{\mathcal{F}}^{-1}(\Lambda_g) = v$. We note that, in practice, we only have access to pairs of observations $(g_i, v\frac{\partial c_i}{\partial n}|_{\partial\Omega})_{i=1}^N$ on a finite number of sensor points on the boundary $\partial\Omega$, where c_i denotes the solution to the (3.20) with conductivity v and boundary condition prescribed by g_i. Thus the equation used in practice is

$$\hat{\mathcal{F}}(u, v) = \left(\tilde{\mathcal{F}}(v)g_i - v\frac{\partial c_i}{\partial n}|_{\partial\Omega}\right)_{i=1}^N = 0,$$

where $u = (g_i, v\frac{\partial c_i}{\partial n}|_{\partial\Omega})_{i=1}^N$.

We note that in the inverse problems, we adopt to the first discretize-then-linearize philosophy for practical reasons. Since the operator $\mathcal{F}$ is defined implicitly by solving the forward problem, discretizing the problem first, allows us to express the operator $\mathcal{F}$ explicitly as follows: The PDE is discretized via a suitable finite stencils method and written in the form $A(v)c = Gg$, where $A(v)$ is the discrete differential operator depending on the conductivity v, g denotes the imposed boundary data, and G is the extension operator that incorporates the boundary condition into the discrete domain representation. The corresponding discrete solution is $c = A(v)^{-1}Gg$, and the associated boundary flux is obtained by applying a linear observation operator $F(v)$ which depends on the conductivity v. Hence, the discrete operator $\tilde{\mathcal{F}}$ is represented by $\tilde{\mathcal{F}}(v) = F(v)A(v)^{-1}G$, which maps the conductivity to the DtN map. This map is used to construct a concrete DtN map $\Lambda_v := \tilde{\mathcal{F}}(v)$ for a randomly chosen conductivity v which we then use to recover the conductivity. The target operator is then given by $\mathcal{F}(\Lambda_v, v) = F(v)A(v)^{-1}G - \Lambda_v$. Using the explicit expression of the discrete $\mathcal{F}$, its Fréchet derivative can be computed directly, and in our implementation we use automatic differentiation.

We take the fixed forcing term to be a single draw from a zero-mean GP with $5/2$ Matérn kernel and constant lengthscale of $3/10$ across both dimensions.

3.3.4.2 Inverse Wave Scattering

Let $\Omega = [0, 1]^2$ and assume that $a \in L^\infty(\Omega)$ is a real-valued material property satisfying $a(x) > 0$ for almost every $x \in \Omega$. We study the following elliptic

boundary value problem:

$$\begin{cases} -\Delta u(x) - \omega^2 a(x) u(x) = 0, & x \in \Omega, \\ u(x) = g(x), & x \in \partial\Omega, \end{cases} \tag{3.21}$$

where $g \in H^{\frac{1}{2}}(\partial\Omega)$ denotes the prescribed boundary excitation.

We define the *Dirichlet-to-Neumann* (DtN) map $\Lambda_a : H^{\frac{1}{2}}(\partial\Omega) \to H^{-\frac{1}{2}}(\partial\Omega)$ that maps the boundary input g to the corresponding flux

$$\Lambda_a(g) = \left.\frac{\partial u}{\partial n}\right|_{\partial\Omega}.$$

Note that $\Lambda_a \in \mathcal{L}(H^{\frac{1}{2}}(\partial\Omega), H^{-\frac{1}{2}}(\partial\Omega))$ is a bounded linear operator depending on the material coefficient a.

The *inverse wave scattering problem* is the task of recovering the coefficient a from a given DtN map Λ_a. Denote with $\tilde{\mathcal{F}} : L^\infty(\Omega) \to \mathcal{L}(H^{\frac{1}{2}}(\partial\Omega), H^{-\frac{1}{2}}(\partial\Omega))$ the forward operator that maps the material property a to the DtN map. Then, the operator of interest is defined as

$$\mathcal{F}(\Lambda_a, a) = \tilde{\mathcal{F}}(a) - \Lambda_a.$$

Again, CHONKNORIS aims to approximate the operator

$$\mathcal{G} : \mathcal{L}\big(H^{\frac{1}{2}}(\partial\Omega), H^{-\frac{1}{2}}(\partial\Omega)\big) \to L^\infty(\Omega),$$

which maps the DtN map to the material property, i.e., $\mathcal{G}(\Lambda_a) = \tilde{\mathcal{F}}^{-1}(\Lambda_a) = a$.

In practice, we only have access to finitely many boundary input-output pairs $(g_i, \frac{\partial u_i}{\partial n}|_{\partial\Omega})_{i=1}^N$ measured on sensor locations along $\partial\Omega$, where u_i denotes the solution to (3.21) corresponding to the boundary excitation g_i. Hence, the equation used in practice is

$$\hat{\mathcal{F}}(u, a) = \left(\tilde{\mathcal{F}}(a) g_i - \left.\frac{\partial u_i}{\partial n}\right|_{\partial\Omega}\right)_{i=1}^N = 0,$$

where $u = (g_i, \frac{\partial u_i}{\partial n}|_{\partial\Omega})_{i=1}^N$.

Similar to the Calderon problem, we adopt to the first discretize-then-linearize philosophy which allows us to explicitly express the operator $\mathcal{F}$ on the discrete level by $\tilde{\mathcal{F}}(a) = FA(a)^{-1}G$ which maps the material property to the DtN map. The target operator is given by $\mathcal{F}(\Lambda_a, a) = FA(a)^{-1}G - \Lambda_a$. Again, this map is

used to construct a concrete DtN map $\Lambda_a := \tilde{\mathcal{F}}(a)$ for a randomly chosen material property a which is then recovered by CHONKNORIS. The Fréchet derivative of $\mathcal{F}$ is computed via automatic differentiation.

We take the fixed forcing term to be a single draw from a zero-mean Gaussian process with $5/2$ Matérn kernel and constant lengthscale of $2/10$ across both dimensions.

3.3.4.3 Seismic Imaging

Let $\Omega = [0,1]^2$ and let $T > 0$ denote the final observation time. Assume that $v \in L^\infty(\Omega)$ is a real-valued velocity coefficient satisfying $v(x) > 0$ for almost every $x \in \Omega$. We study the following time-dependent acoustic wave equation:

$$\begin{cases} \Delta p(t,x) - \frac{1}{v^2(x)} p_{tt}(t,x) = s(t,x), & (t,x) \in (0,T) \times \Omega, \\ p(0,x) = 0, \quad p_t(0,x) = 0, & x \in \Omega, \end{cases} \tag{3.22}$$

where p denotes the pressure variation and s represents the source term which we take to be a Ricker wavelet [349].

The reflected and refracted wavefields are measured on the surface $\mathcal{S} := \{(x_1, x_2) \in \Omega \mid x_2 = 0\} \subset \partial\Omega$, yielding the boundary observations $p|_{[0,T]\times\mathcal{S}}$. We define the *Source-to-Receiver* (StR) map $\Lambda_v : L^2((0,T)\times\Omega) \to L^2(0,T; H^{\frac{1}{2}}(\mathcal{S}))$ that maps a source s to the measured surface signal $\Lambda_v(s) = p|_{[0,T]\times\mathcal{S}}$. Note that $\Lambda_v \in \mathcal{L}(L^2((0,T)\times\Omega), L^2(0,T; H^{\frac{1}{2}}(\mathcal{S}))$ is a bounded linear operator depending on the velocity coefficient v.

The *seismic inverse problem* or *full wave inversion problem* is the task of recovering the coefficient v from a given Source-to-Receiver map Λ_v [343]. Denote with $\tilde{\mathcal{F}} : L^\infty(\Omega) \to \mathcal{L}(L^2((0,T)\times\Omega), L^2(0,T; H^{\frac{1}{2}}(\mathcal{S})))$ the forward operator that maps the velocity coefficient v to the corresponding StR map. Then, the operator of interest is defined as

$$\mathcal{F}(\Lambda_v, v) = \tilde{\mathcal{F}}(v) - \Lambda_v(s).$$

Similar to the inverse wave scattering and Calderón problems, CHONKNORIS aims to approximate the operator

$$\mathcal{G} : \mathcal{L}(L^2((0,T)\times\Omega), L^2(0,T; H^{\frac{1}{2}}(\mathcal{S}))) \to L^\infty(\Omega),$$

that maps the Source-to-Receiver map to the velocity coefficient, i.e., $\mathcal{G}(\Lambda_v) = \tilde{\mathcal{F}}^{-1}(\Lambda_v) = v$.

In practice, we only have access to finitely many input-output pairs $(s_i, p_i|_{[0,T]\times\mathcal{S}})_{i=1}^N$ measured at discrete sensor locations along $\mathcal{S}$, where p_i denotes the solution to (3.22) corresponding to the source s_i. Hence, the equation used in practice is

$$\hat{\mathcal{F}}(u, v) = \left(\tilde{\mathcal{F}}(v)s_i - p_i|_{[0,T]\times\mathcal{S}}\right)_{i=1}^N = 0, \tag{3.23}$$

where $u = (s_i, p_i|_{[0,T]\times\mathcal{S}})_{i=1}^N$. We use data from the OpenFWI dataset [82] and the forward solver from [348] (a $2-4$ finite difference scheme with 2nd-order accuracy in time and 4th-order in space). Convergence of the exact NK method for a single 14×14 resolution velocity map is shown in Figure 3.4 (a1-a5).

As for the other inverse problems, we adopt to the first discretize-then-linearize philosophy so we can explicitly express the operator $\widehat{\mathcal{F}}$ on the discrete level. As mentioned before, instead of recovering the velocity field v given the StR map, we recover v from input-output pairs $(s_i, p_i|_{[0,T]\times\mathcal{S}})_{i=1}^N$. This is accomplished by doing a space-time discretization and solving the resulting discretized PDE. This defines the operator $\tilde{\mathcal{F}}(v)s = A(v)^{-1}s$ which maps the velocity field to the solution of the acoustic wave equation for a given source term s. The target operator is given by

$$\hat{\mathcal{F}}(u, v) = \left(A(v)^{-1}s_i - p_i|_{[0,T]\times\mathcal{S}}\right)_{i=1}^N. \tag{3.24}$$

This map is used to recover v by CHONKNORIS. The Fréchet derivative of $\mathcal{F}$ is computed via automatic differentiation.

While our previous experiments predicted the inverse Cholesky of the approximate Hessian, for this problem we obtained better performance by directly predicting the Cholesky factors of the approximate Hessian and then use triangular solves to determine each increment. Invertibility of the triangular matrix is ensured by enforcing non-zero diagonals. We note that the resulting inference complexity remains unchanged when using the Cholesky factor rather than its inverse due to the quadratic cost of triangular matrix solves. We also found it necessary to simultaneously tune both the step size α and the Tikhonov relaxation λ in order to converge to machine precision in a reasonable number of iterations, see Algorithm 1. The inflation/deflation tuning scheme we used is described in Appendix B.5. In Table B.2 we compare seismic imaging dataset generation times and CHONKNORIS model fitting times across resolutions.

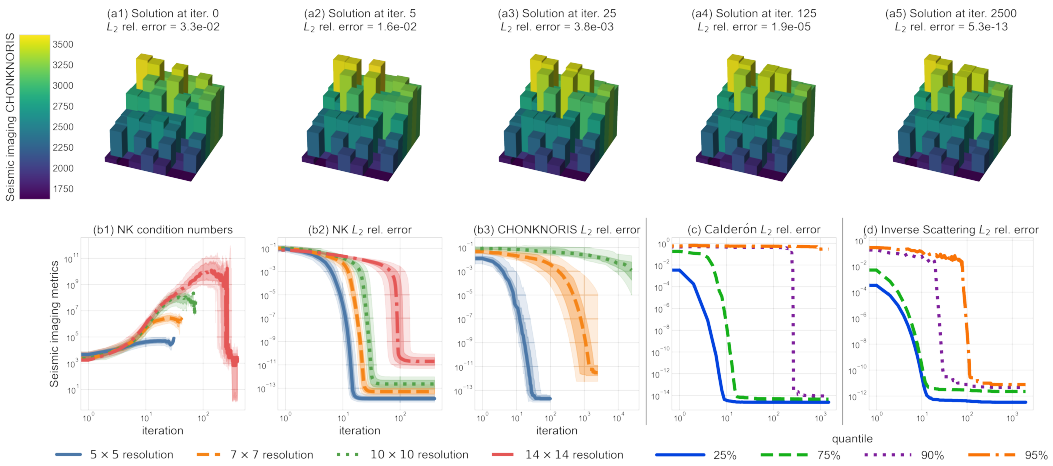


Figure 3.4: Inverse Problems. (a) Results for the seismic imaging problem, showing the iterative solutions and the relative L^2 error across CHONKNORIS iteration. (b) For the seismic imaging problem, evolution of the adaptive regularization term in the Newton–Kantorovich iterations for different resolutions, and comparison of the relative L^2 error between the Newton–Kantorovich method and our CHONKNORIS method. (c) Results for the Calderón problem. (d) Results for the inverse wave scattering problem.

3.3.4.4 Summary of the numerical results for CHONKNORIS on the Inverse Problems

We use CHONKNORIS to solve the three inverse problems above. The domains are discretized using regular grids. A traditional NK solver is used to generate training data where the converged iterates are used as reference solutions. These inverse problems are significantly harder to solve than the forward problems due to the non-locality of the operator $\mathcal{F}$ and the ill-posedness of the inverse problem. The results for the Calderón problem in Figure 3.4 (d) show that running the CHONKNORIS method for 10^3 iterations was able to recover solutions to machine precision for over 75% of withheld test realizations. The results for the inverse wave scattering problem in Figure 3.4 (e) shows that running the CHONKNORIS method for 40 iterations is able to recover solutions to machine precision. For the rough velocity maps encountered in the seismic imaging problem, increasing the resolution increases the condition numbers of the relaxed approximate Hessian. For example, Figure 3.4 shows that the seismic imaging problem with just a 10×10 velocity map resolution encounters condition numbers around 10^8 for near-convergence iterations. CHONKNORIS is unable to exactly predict the Cholesky factor of ill-conditioned matrices, and thus resorts to inferring gradient descent steps which can make CHONKNORIS inference slow to converge. As expected, we also observe that the relaxation is

decreased as NK nears convergence, indicating a smooth transition from gradient descent to Gauss–Newton updates. This behavior is more difficult to replicate with CHONKNORIS as the approximate Hessian near the solution is ill-conditioned and therefore difficult for CHONKNORIS to predict.

3.4 Theoretical results

In this section, we want to present several results on the convergence properties of the CHONKNORIS and FONKNORIS algorithms. This is accomplished by combining an inexact Newton–Kantorovich method with Tikhonov-regularized inverse and the kernel-based operator learning method.

3.4.1 Analytic Setting

Assume that we are in the operator learning framework of Section 3.2.1, i.e., let $(\mathcal{U}, \langle\cdot,\cdot\rangle_{\mathcal{U}}, \|\cdot\|_{\mathcal{U}})$, $(\mathcal{V}, \langle\cdot,\cdot\rangle_{\mathcal{V}}, \|\cdot\|_{\mathcal{V}})$, and $(\mathcal{W}, \langle\cdot,\cdot\rangle_{\mathcal{W}}, \|\cdot\|_{\mathcal{W}})$ be separable Hilbert spaces. For notational convenience, we suppress the dependence of $\mathcal{F}$ on the first argument, i.e., for each fixed $u \in \mathcal{U}$, we define $\mathcal{F}(\cdot) := \mathcal{F}(u,\cdot) : \mathcal{V} \to \mathcal{W}$ and its Fréchet derivative $\mathcal{F}'(v) := \frac{\delta\mathcal{F}}{\delta v}(u,v) \in \mathcal{L}(\mathcal{V},\mathcal{W})$ with Hilbert adjoint $\mathcal{F}'(v)^* \in \mathcal{L}(\mathcal{W},\mathcal{V})$.

For $\lambda > 0$ and $v \in \mathcal{V}$, define the *Tikhonov resolvent* and the *Tikhonov right inverse*

$$R_\lambda(v) := \big(\lambda I_{\mathcal{V}} + \mathcal{F}'(v)^*\mathcal{F}'(v)\big)^{-1} \in \mathcal{L}(\mathcal{V},\mathcal{V}),$$
$$B_\lambda(v) := R_\lambda(v)\,\mathcal{F}'(v)^* \in \mathcal{L}(\mathcal{W},\mathcal{V}).$$

Let $\widehat{R}_\lambda(v) \in \mathcal{L}(\mathcal{V},\mathcal{V})$ be a (learned) surrogate and set $\widehat{B}_\lambda(v) := \widehat{R}_\lambda(v)\,\mathcal{F}'(v)^*$. Given $v_0 \in \mathcal{V}$, consider the iteration

$$v_{k+1} = v_k + \delta v_k, \qquad \delta v_k := -\widehat{B}_{\lambda_k}(v_k)\,\mathcal{F}(v_k) = -\widehat{R}_{\lambda_k}(v_k)\,\mathcal{F}'(v_k)^*\,\mathcal{F}(v_k),$$

for a sequence of parameters $\lambda_k > 0$. Furthermore, we make the following assumptions.

Assumption 3.4.1. *Let $D \subset \mathcal{V}$ be open and convex and $\mathcal{F} : D \to \mathcal{W}$ be C^1. Fix*

$v_0 \in D$ *and* $R > 0$ *with* $B := \overline{B(v_0, R)} \subset D$. *Assume:*

(A1) $\mathcal{F}'(v_0)$ *is invertible and* $\|\mathcal{F}'(v_0)^{-1}\|_{\mathcal{L}(\mathcal{W},\mathcal{V})} \leq \beta \in \mathbb{R}_{>0}$,

(A2) There exists $L > 0$ *such that*

$$\|\mathcal{F}'(u) - \mathcal{F}'(v)\|_{\mathcal{L}(\mathcal{V},\mathcal{W})} \leq L\,\|u - v\|_{\mathcal{V}} \quad \forall\, u, v \in B,$$

(A3) $M := \sup_{v\in B} \|\mathcal{F}'(v)\|_{\mathcal{L}(\mathcal{V},\mathcal{W})} < +\infty$,

$$\sigma_* := \inf_{v\in B} \sigma_{\min}\big(\mathcal{F}'(v)\big) = \inf_{v\in B} \|\mathcal{F}'(v)^{-1}\|^{-1}_{\mathcal{L}(\mathcal{W},\mathcal{V})} > 0.$$

Now, we are in a position to state the main theorem on the Tikhonov-regularized inexact Newton–Kantorovich iteration method based on a classical inexact Newton–Kantorovich method [81]. The proofs in this section are postponed to the appendix Appendix B.8.

Theorem 3.4.2 (Tikhonov–inexact Newton–Kantorovich)**.** *Let Assumption 3.4.1 be satisfied. Then, for each* v_k *with* $v_k \in \overline{B(v_0, R)}$, *the linearized residual satisfies*

$$\frac{\|\mathcal{F}'(v_k)\delta v_k + \mathcal{F}(v_k)\|_{\mathcal{W}}}{\|\mathcal{F}(v_k)\|_{\mathcal{W}}} \;\leq\; \frac{\lambda_k}{\lambda_k + \sigma_*^2} \;+\; M^2\,\varepsilon_{\lambda_k} \;=:\; \theta_k, \tag{3.25}$$

where

$$\varepsilon_\lambda \;:=\; \sup_{v\in\overline{B(v_0,R)}} \big\|\, \widehat{R}_\lambda(v) - R_\lambda(v)\,\big\|_{\mathcal{L}(\mathcal{V},\mathcal{V})}$$

denotes the design/learning error. If $\sup_k \theta_k \leq \bar\theta < 1$ *and all iterates remain in* $\overline{B(v_0, R)}$, *then the inexact Newton–Kantorovich conclusions hold on* $\overline{B(v_0, R)}$ *with*

$$\widetilde{L} \;:=\; \frac{L}{1-\bar\theta}, \qquad \widetilde{h} \;:=\; \beta\,\widetilde{L}\,\eta, \qquad \eta := \|\mathcal{F}'(v_0)^{-1}\mathcal{F}(v_0)\|_{\mathcal{V}}.$$

In particular, if $\widetilde{h} \leq \frac{1}{2}$ *and* $t_* = (1 - \sqrt{1 - 2\widetilde{h}})/(\beta\widetilde{L}) \leq R$, *then the iterates are well-defined, remain in* $\overline{B(v_0, t_*)}$, *and converge to the unique zero* $v_* \in \overline{B(v_0, t_*)}$, *with the Kantorovich majorant bounds*

$$\|v_k - v_*\|_{\mathcal{V}} \leq t_* - t_k, \qquad \|v_{k+1} - v_k\|_{\mathcal{V}} \leq t_{k+1} - t_k,$$

where $t_{k+1} = t_k - \dfrac{\phi(t_k)}{\phi'(t_k)}$ *and* $\phi(t) = \eta - t + \frac{1}{2}\,\beta\widetilde{L}\,t^2$.

Moreover, if $\lambda_k \to 0$ *and* $\varepsilon_{\lambda_k} \to 0$, *then* $\theta_k \to 0$ *and the local rate approaches the quadratic rate of exact Newton.*

The following corollary makes the final statement of the preceding Theorem 3.4.2 on the convergence more precise. In particular, it shows for which Tikhonov parameters and design errors we obtain linear, superlinear, quadratic convergence.

Corollary 3.4.3 (Convergence Rates)**.** *Let* $e_k := \|v_k - v_*\|_{\mathcal{V}}$*. Under the hypotheses of Theorem 3.4.2 and for all* k *with* $v_k \in \overline{B(v_0, R)}$*,*

$$e_{k+1} \;\le\; \frac{1}{1-\bar{\theta}}\left(\frac{\beta L}{2}\, e_k^2 \;+\; \bar{\theta}\, e_k\right). \tag{3.26}$$

Then, there exist constants C_1, C_2, C_3, C_4 *such that*

(i) *If* $\bar{\theta} \in (0,1)$ *is fixed, then* $\limsup_{k\to\infty} \dfrac{e_{k+1}}{e_k} \;\le\; \dfrac{\bar{\theta}}{1-\bar{\theta}}$ *(at least Q-linear).*

(ii) *If* $\theta_k \to 0$*, then* $e_{k+1} \le \frac{\beta L}{2}\, e_k^2 (1 + o(1))$ *and the rate is Q-superlinear.*

(iii) *If* $\theta_k \le C_1 \|\mathcal{F}(v_k)\|_{\mathcal{W}}^{\alpha}$ *for some* $\alpha > 0$*, then* $e_{k+1} \le C_2\, e_k^{1+\alpha} + C_3 e_k^2$ *for* k *large, so the local order is* $\min\{2, 1+\alpha\}$*.*

(iv) *If* $\theta_k = O(\|\mathcal{F}(v_k)\|_{\mathcal{W}})$*, then* $e_{k+1} \le C_4\, e_k^2$ *and the convergence is Q-quadratic.*

Based on the preceding corollary, we can propose a schedule for the Tikhonov parameter λ_k and the design error ε_{λ_k}in order to achieve the desired convergence.

Remark 3.4.4 (Scheduling λ_k to reach superlinear/quadratic convergence)**.** *Let* $\theta_{\text{tik,k}} = \dfrac{\lambda_k}{\lambda_k + \sigma_*^2}$ *and* $\theta_{\text{des,k}} = M^2\, \varepsilon_{\lambda_k}$ *so that* $\theta_k = \theta_{\text{tik,k}} + \theta_{\text{des,k}}$*.*

(a) *If* $\lambda_k \to 0$ *and* $\varepsilon_{\lambda_k} \to 0$*, then* $\theta_k \to 0$ *and the rate is superlinear.*

(b) *If there exist* $c_1, c_2 > 0$ *such that*

$$\lambda_k \;\le\; c_1 \|\mathcal{F}(v_k)\|_{\mathcal{W}}, \qquad \varepsilon_{\lambda_k} \;\le\; c_2 \|\mathcal{F}(v_k)\|_{\mathcal{W}},$$

then $\theta_k = O(\|\mathcal{F}(v_k)\|_{\mathcal{W}})$ *and the convergence is Q-quadratic.*

(c) *Suppose for some* $C > 0$ *one has the regularity proxy* $\varepsilon_\lambda \approx C\, \lambda^{-2}$*. Consider* $\phi(\lambda) := \dfrac{\lambda}{\lambda + \sigma_*^2} + C\, \lambda^{-2}$*. Then any* λ_k *chosen near the minimizer* $\lambda_* \asymp (C\, \sigma_*^2)^{1/3}$ *balances Tikhonov bias and model error early on; subsequently decreasing* $\lambda_k \downarrow 0$ *(as the model improves and the iterates approach* v_**) drives* $\bar{\theta} \to 0$ *and approaches quadratic convergence.*

Remark 3.4.5 (Variant: learning the full $B_\lambda(v)$ directly)**.** *If the surrogate model directly learns $B_\lambda(v)$ and achieves $\sup_{v\in B} \| \widehat{B}_\lambda(v) - B_\lambda(v)\|_{\mathcal{L}(\mathcal{W},\mathcal{V})} \le \varepsilon_\lambda$, then the residual bound becomes*

$$\bar\theta \;\le\; \underbrace{\frac{\lambda}{\lambda+\sigma_*^2}}_{\theta_{\mathrm{tik}}} \;+\; \underbrace{M\,\varepsilon_\lambda}_{\theta_{\mathrm{des}}},$$

since $\|I - \mathcal{F}'\widehat{B}_\lambda\| \le \|I - \mathcal{F}'B_\lambda\| + \|\mathcal{F}'\|\,\|\widehat{B}_\lambda - B_\lambda\|$. All conclusions of Theorem 3.4.2 remain unchanged with this $\bar\theta$.

Example 3.4.6 (Application to nonlinear elliptic PDE)**.**

In this example, we want to apply the convergence theorem to the nonlinear elliptic PDE considered in Section Section 3.3.2.1. Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ and

$$\mathcal{V} := H^1_{\mathrm{per},0}(\mathbb{T}) = \Big\{u \in H^1_{\mathrm{per}}(\mathbb{T}) : \int_0^1 u = 0\Big\}, \qquad \mathcal{W} := \mathcal{V}^* \simeq H^{-1}_{\mathrm{per},0}(\mathbb{T}),$$

with $\|u\|_{\mathcal{V}} := \|u'\|_{L^2(0,1)}$. Let $f \in L^2(0,1)$ be fixed and define

$$\mathcal{F} : \mathcal{V} \to \mathcal{W}, \qquad \mathcal{F}(u) = -u'' + u^3 - f, \qquad \mathcal{F}'(u)v = -v'' + 3u^2 v.$$

Now, we present the main convergence result with explicit constants by verifying the conditions of Theorem 3.4.2.

Theorem 3.4.7 (Main convergence theorem with explicit constants)**.** *Let $v_0 = 0$ and $\lambda > 0$ be the Tikhonov parameter. Assume that the iterates $\{v_k\}$ remain in $B_r := \{v \in \mathcal{V} : \|v\|_{\mathcal{V}} \le r\}$ for some $r > 0$. Define*

$$\eta := \|\mathcal{F}'(0)^{-1}\mathcal{F}(0)\|_{\mathcal{V}} = \|f\|_{H^{-1}}, \qquad L = \frac{3}{2\pi^2}r, \qquad M = 1 + \frac{3}{4\pi^2}r^2.$$

Then all the conclusions of Theorem 3.4.2 hold true for this problem. In particular, the Lipschitz constant in Theorem 3.4.2 can be chosen as

$$\widetilde{L} \;:=\; \frac{L}{1-\bar\theta} \;=\; \frac{\frac{3}{2\pi^2}r}{1 - \frac{\lambda}{1+\lambda} - M^2\varepsilon_\lambda},$$

where ε_λ is the design/learning error, and $\bar\theta := \frac{\lambda}{1+\lambda} + M^2\varepsilon_\lambda$ bounds the forcing terms θ_k in (3.25) *for all v_k with $v_k \in B_r$.*

3.5 Conclusion and Outlook

In this paper, we introduced CHONKNORIS, a novel operator learning methodology rooted in the Newton–Kantorovich method for solving infinite-dimensional systems. Crucially, CHONKNORIS removes the longstanding ceiling in accuracy typically observed in operator learning frameworks, setting a new standard in precision. To our knowledge, this is the first instance of an operator learning framework that achieves machine precision in solving a variety of nonlinear partial differential equations (PDEs) and inverse problems.

The central innovation of CHONKNORIS lies in employing an iterative scheme based on the Newton–Kantorovich method, where we explicitly learn the dependence of the Cholesky factors of the matrix $(\lambda I + (\frac{\partial \mathcal{F}}{\partial v})^* \frac{\partial \mathcal{F}}{\partial v})^{-1}$ on coefficients u and solution estimates v. The explicit integration of the underlying equation within these iterations significantly enhances the accuracy and interpretability of the learned solutions. It also makes it uniquely suited for inverse problems, as the general equation $\mathcal{F}$ may incorporate observed data. The ability to tackle inverse problems directly within the operator learning framework is a significant improvement from traditional approaches, which depend on repeated iterations of forward solvers and often struggle in high-dimensional settings. Building on CHONKNORIS, we introduced FONKNORIS, a foundational model motivated by the observation that $\frac{\partial \mathcal{F}}{\partial v}$ always defines a linear PDE. While CHONKNORIS learns to solve the linearized PDEs associated with the Fréchet derivative of one equation $\mathcal{F}$, FONKNORIS extends this concept by learning to solve any linearized PDE. By plugging this learned linear PDE solver into the CHONKNORIS framework, we obtain a universal operator learner capable of solving a broad class of nonlinear PDEs.

We validated our methodology on diverse forward problems, including a nonlinear elliptic equation, Burgers' equation, a Darcy flow equation, the Sine–Gordon equation, and Klein–Gordon equation as well as inverse problems, including Calderòn's problem, an inverse wave scattering problem, and a problem from seismic imaging full waveform inversion. Both CHONKNORIS and FONKNORIS achieve machine precision in these cases, demonstrating their robustness and accuracy. Notably, FONKNORIS attained near machine precision on the Klein–Gordon and Sine–Gordon equations, despite not being trained on these equations, demonstrating remarkable generalization capabilities.

Despite these advances, our method is still constrained by the computational cost and occasional ill-conditioning that arise when computing and storing Cholesky

factors in high dimensions. In practice, avoiding explicit computation and inversion of the Fréchet derivative is crucial. For example, rather than forming the derivative exactly, one can use the approximation described in Appendix B.6. Additional improvements in high-dimensional scalability depend on the surrogate model used to solve the linear subproblem. We note that CHONKNORIS and FONKNORIS are agnostic to the underlying machine learning algorithm used to approximate the Cholesky factors, which may allow one to solve the computational challenges associated with high-dimensional problems. In particular, this adaptability enables the integration of various deep learning techniques, such as Fourier Neural Operators or Deep Operator Networks, as well as scalability enhancements for Gaussian process regression through inducing points or nested Kriging. In Appendix B.3, we show that a sparse-Cholesky strategy can scale and accelerate CHONKNORIS for Gaussian-process surrogates, albeit with an accuracy-speed trade-off: faster run-times come at the cost of reduced precision. Even though, we don't make use of the sparse-Cholesky algorithm, we do exploit the rank-revealing and sparsity properties of the max-min ordering, see Appendix B.3.0.3. Further ways of scaling and accelerating CHONKNORIS are by using nested kriging and approximating the Fréchet derivative, see Appendix B.2 and Appendix B.6, respectively. We also experimented with learning Newton—Kantorovich increments directly to reduce overhead; however, this has thus far yielded only limited gains. Finally, incorporating multi-fidelity, multi-scale, and hierarchical approaches offer promising directions for future research.

3.6 Code and data availability

An open source Python implementation of all experiments presented in this work is available at https://github.com/OwhadiGroup/CHONKNORIS. We primarily utilize the Python packages PyTorch [253], PyTorch Lightning [95], GPyTorch [111], and FastGPs [317]. Our implementation uses double-precision floating point arithmetic for which machine precision is 2.2×10^{-16}.

All synthetic datasets used in the forward and inverse PDE experiments are provided in the repository. For the seismic imaging (full waveform inversion) experiments we use the publicly available OpenFWI dataset [82].

3.7 Acknowledgments.

AB, XY, MD, TB and HO acknowledge support from the Air Force Office of Scientific Research under MURI awards number FA9550-20-1-0358 (Machine Learn-

ing and Physics-Based Modeling and Simulation), FOA-AFRL-AFOSR-2023-0004 (Mathematics of Digital Twins), the Department of Energy under award number DE-SC0023163 (SEA-CROGS: Scalable, Efficient, and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems), the National Science Foundation under award number 2425909 (Discovering the Law of Stress Transfer and Earthquake Dynamics in a Fault Network using a Computational Graph Discovery Approach). HO acknowledges support from the DoD Vannevar Bush Faculty Fellowship Program under ONR award number N000142512035. AH and BH acknowledge support from the National Science Foundation under awards 2208535 (Machine Learning for Bayesian Inverse Problems) and 2337678 (CAREER: Gaussian Processes for Scientific Machine Learning: Theoretical Analysis and Computational Algorithms). AH acknowledges support from a Carl S. Pearson Fellowship. EC acknowledges support from the Department of Defense (DoD) Vannevar Bush Faculty Fellowship held by Prof. Andrew Stuart (award N00014-22-1-2790), and the Resnick Sustainability Institute. AS acknowledges that material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists, Office of Science Graduate Student Research (SCGSR) program. The SCGSR program is administered by the Oak Ridge Institute for Science and Education for the DOE under contract number DE-SC0014664.

Chapter 4

PRUNING DEEP NEURAL NETWORKS VIA MARCHENKO–PASTUR THEORY AND REGULARIZATION

Leonid Berlyand, Théo Bourdais, Houman Owhadi, and Yitzchak Shmalo. Pruning deep neural networks via a combination of the Marchenko–Pastur distribution and regularization. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2503.01922.
TB co-developed the pruning methodology and theory, contributed to experiments, and co-wrote the manuscript.

4.1 Introduction

DNNs have emerged as a powerful tool in the realm of classification tasks, where they categorize objects from a set $S \subset \mathbb{R}^n$. They are a class of parametric functions and will be defined in section 4.2.1. The training of DNNs on labeled training datasets $T \subset \mathbb{R}^n$ is done by optimizing its parameters to minimize a loss in order to increase the network's accuracy, typically the cross-entropy loss, as shown in (4.3). DNNs have been applied to various real-world challenges, achieving remarkable results in fields such as handwriting recognition [178], image classification [174], speech recognition [147], and natural language processing [326] among others. One significant hurdle in DNN training is overfitting, where the model excessively adapts to the training data, leading to poor generalization of unseen data. This often results in a decline in performance on test datasets despite high accuracy on the training set T. To mitigate overfitting, researchers have developed various regularization strategies, including dropout [320], early stopping [265], and weight decay regularization [250].

Recently, Random Matrix Theory (RMT) has emerged as a promising approach to tackling overfitting in deep learning [204, 209]. This includes the development of RMT-based stopping criteria [213] and regularization methods [355]. Furthermore, RMT has been applied to analyze the spectrum of weight layer matrices [330] and study the input-output Jacobian matrix [251, 252]. Investigations into RMT-based initializations for DNNs have also been conducted, see e.g, [283]. It has been demonstrated that the spectral characteristics of DNN weight layer matrices can provide insightful information about the network's performance, even in the absence of test data. In particular, it has been shown in [208, 210] that the spectrum

of the weight layers of a DNN can be indicative of the DNN's accuracy on unseen test sets. In this work, we show that analyzing these spectral properties helps identify weights that have minimal impact on accuracy, enabling the pruning of a DNN's weights while maintaining its accuracy.

In the comprehensive review by [336], numerous studies on pruning are categorized into three main strategies: magnitude-based pruning, redundancy identification through clustering, and sensitivity analysis-based pruning. Other works and surveys on pruning in deep learning can be found in [62] and [161]. Several previous studies, such as [10, 49, 359–361], have leveraged Singular Value Decomposition (SVD) to prune small singular values from DNN weight matrices, focusing on empirical techniques like energy thresholds and validation set error monitoring. In our work, we introduce a theoretically grounded magnitude-based pruning method using RMT. The Marchenko–Pastur (MP) distribution, a famous RMT distribution (see Section 4.2.2), characterizes the distribution of the singular values of random matrices. Using the MP distribution, we may recover the amount of noise in the weight layers and compare their spectrum to that of purely random matrices. Eigenvalues that deviate significantly from what would be expected in a purely random matrix indicate weights that contain information and are critical to the network's function. Conversely, weights associated with eigenvalues that align closely with those of random matrices are considered candidates for removal. Additionally, we quantify the degree to which each layer's spectrum aligns with the MP distribution, providing a measure of "randomness" that guides targeted training and regularization efforts. For the classical case of fully connected DNNs, it has been demonstrated in [34] both numerically and theoretically using rigorous mathematical analysis that MP-based pruning can reduce a network's parameters by over 99% while increasing its accuracy.

The goal of this work is to develop state-of-the-art pruning algorithms for Vision Transformer (ViT) models based on the MP spectral approach from RMT theory. We compare our pruning algorithms with those presented in [123, 249, 316, 382]. A key feature of our method, compared to other works, is that our pruning does not rely on any access to a validation or test set. Furthermore, our numerical experiments reveal that the extent of effective parameter removal achieved via our RMT-based pruning method increases with the size of the ViT models. This is particularly significant in today's context, as DNNs continue to scale up, underscoring the essential role of our approach for effective pruning in large-scale models.

In principle, this pruning approach can be extended to other deep neural network models, such as Convolutional Neural Networks (CNNs). We also provide a rigorous mathematical analysis of how removing randomness from the weight layers of DNNs reduces loss without affecting accuracy. We demonstrate that during training, the randomness in the DNN weight layers diminishes over time. As DNN parameters converge to a local minimum, the randomness vanishes.

The paper has been organized as follows. In Sections 1 and 2, a brief overview of key notions and concepts is presented. In Section 4.3, we apply our MP-based pruning and fine-tuning approach to ViTs pre-trained on the 1K ImageNet dataset [83]. We study the reduction in parameters and how it affects the accuracy of the DNN on the ImageNet validation set. We compare our results with the following work [316] and the references therein, showing that we achieve higher pruning with less reduction in accuracy.

In Sections 4.4 and 4.5, we present our main mathematical results that explain the evolution of randomness in DNN weight layers during training. We prove two key theorems. Our first theorem demonstrates that for a DNN trained with $L2$ regularization, the random parameters within the network's weight matrices diminish and vanish as the network's loss approaches a local or global minimum. This result underscores the transition of the network's parameters from a regime dominated by randomness (from their initialization) to one where deterministic structures prevail, thereby contributing to how DNNs extract information from noisy data under training. The second theorem describes how DNN loss decreases as we remove randomness in the weight layers and shows a monotone dependence of the decrease in loss with the amount of randomness that we remove.

Our numerical findings and corresponding mathematical results illustrate the interplay between regularization and pruning (see Theorem 4.4.2), emphasizing the critical role of regularization techniques in the training and pruning of DNNs. Our numerical approach is underpinned by rigorous mathematical results and tested on advanced ViT models, demonstrating its applicability to modern deep-learning models.

4.2 Randomness in Deep Neural Networks

4.2.1 Introduction to Deep Neural Networks

In classification tasks, the goal is to categorize elements of a set S into one of K distinct classes. Let $C(s) \in \{1, .., K\}$ be the correct class of $s \in S$. Given a

labeled training set $T \subset S$, we wish to develop an approximate classifier to extend the accurate classification from the training set T to the entire set S. To do so, we use a DNN as a parameter-dependant classifier $\phi(s, \alpha)$, where $\alpha \in \mathbb{R}^\nu$ are its parameters. The DNN outputs a probability vector $(p_1(s), \ldots, p_K(s))$ such that $p_k(s) := \mathbb{P}[C(s) = k]$ is the estimated probability distribution of $C(s)$.
We consider DNNs which are a composition $\rho \circ X(\cdot, \alpha)$, with ρ the softmax (given in (4.2)) and $X(\cdot, \alpha)$ defined as follows. Let

$$X(\cdot, \alpha) = \lambda \circ M_L(\cdot, \alpha) \circ \cdots \circ \lambda \circ M_1(\cdot, \alpha) \tag{4.1}$$

with:

- $M_k(\cdot, \alpha)$ an affine function $\mathbb{R}^{N_{k-1}} \to \mathbb{R}^{N_k}$ parameterized by a weight matrix $W_k \in \mathbb{R}^{N_k \times N_{k-1}}$ and bias vector $\beta_k \in \mathbb{R}^{N_k}$ such that $M_k(x) = W_k \cdot x + \beta_k$.
- $\lambda : \mathbb{R}^m \mapsto \mathbb{R}^m$ is a nonlinear activation function. In this case, we assume λ is the absolute value activation function, such that $\lambda : x \in \mathbb{R}^m \mapsto (|x_1|, \ldots, |x_m|) \in \mathbb{R}^m$.
- The softmax function ρ normalizes the outputs of $X(\cdot, \alpha)$ into a probability vector over $\{1, \ldots, K\}$. The individual components of ρ are computed as:

$$\rho(v)_i = \frac{\exp(v_i)}{\sum_{i=1}^K \exp(v_i)} \text{ for } v \in \mathbb{R}^K \tag{4.2}$$

Given the neural network architecture above, we train our classifier $\phi(\cdot, \alpha)$ by minimizing the cross-entropy loss. This loss quantifies the accuracy of our classifier over the training set, and is defined as

$$L(\alpha) = -\frac{1}{|T|} \sum_{s \in T} \log\left(\phi_{C(s)}(s, \alpha)\right). \tag{4.3}$$

Training a DNN is fundamentally about exploring a complex, high-dimensional, and non-convex landscape of loss to identify the global minimum. However, the intricacies of these landscapes often result in the emergence of local minima, saddle points, and broad flat regions. These features can act as impediments in the learning process, obstructing the path to an optimal solution. The prevalence of such challenges escalates with the increase in dimensionality and complexity of the DNN [65, 76].

In this context, the potential of RMT is noteworthy. By applying RMT to prune DNN weight layers, we can simplify the loss landscape, thereby reducing local minima and saddle points. This simplification assists the optimization process in finding the global minimum, potentially leading to higher training accuracy without reaching a plateau and overall improved model performance. We observe this improvement in performance (higher accuracy) for fully connected DNNs trained with $L1$ and $L2$ regularization; see Appendix C.1.

4.2.2 The MP Distribution in Machine Learning contexts

The identification of the Marchenko-Pastur distribution, a fundamental result in RMT, has far-reaching implications in fields like signal processing, wireless communication, and machine learning. References such as [69, 114, 303, 341] detail its applications. This distribution provides insights into the spectral density of large random matrices, revealing the asymptotic eigenvalue distribution in these matrices and predicting their behavior under various conditions. Furthermore, the MP distribution is pivotal in dimension reduction techniques, including principal component analysis (PCA), as highlighted in [1, 44, 277].

We introduce the concept of the empirical spectral distribution (ESD) for an $N \times M$ matrix G, then state the Marchenko–Pastur theorem:

Definition 4.2.1. *The ESD for an $N \times M$ matrix X is defined as:*

$$\mu_{X_M} = \frac{1}{M} \sum_{i=1}^{M} \delta_{\sigma_i}, \tag{4.4}$$

where σ_i are the non-zero singular values of X, and δ signifies the Dirac measure.

Theorem 4.2.2 (Marchenko and Pastur (1967) [207])**.** *Consider an $N \times M$ random matrix W with $M \leq N$. Let the entries $W_{i,j}$ be independent, identically distributed with zero mean and finite variance σ^2. Let $X = \frac{1}{N} W^T W$. When $N \to \infty$ and $\frac{M}{N} \to c \in (0, +\infty)$, the ESD of X, denoted by μ_{X_M}, converges in distribution to the Marchenko-Pastur probability distribution:*

$$\frac{1}{2\pi\sigma^2} \frac{\sqrt{(\lambda_+ - x)(x - \lambda_-)}}{cx} \mathbf{1}_{[\lambda_-, \lambda_+]} dx \tag{4.5}$$

with

$$\lambda_\pm = \sigma^2 (1 \pm \sqrt{c})^2. \tag{4.6}$$

Theorem 4.2.2 states that as the dimensions of a random matrix increase, its eigenvalue distribution converges to the MP distribution. The MP distribution is deter-

ministic and hinges on two parameters: the variance of the matrix's entries, σ^2, and the aspect ratio of the matrix, denoted by c.

4.2.3 Reduction of randomness in DNN weights via MP Distribution

As outlined in Subsection 4.2.1, a DNN is composed of affine functions M_l, represented by an $N \times M$ matrix W_l of parameters and a bias vector β_l. Before training, these matrices are often initialized with i.i.d. entries according to a normal distribution $\mathcal{N}(0, \frac{1}{N})$. Given the large size of these matrices, the Empirical Spectral Distribution (ESD) of $X_l = \frac{1}{N} W_l^T W_l$ is close to the MP distribution *before training*. Yet, previous studies have used the spiked model in random matrices to examine W_l, showing that the ESD of X_l *after training* exhibits eigenvalues that diverge from the Marchenko-Pastur limit λ_+ [209, 321]. This observation suggests that the weight matrices contain signal obtained from the data after training.

With random initialization and training through Stochastic Gradient Descent (SGD), the sequence of parameters during training $\alpha(t)$ satisfies

$$\alpha(0) \text{ is i.i.d. Gaussian} \tag{4.7}$$

$$\alpha(t+1) = \alpha(t) - \tau \nabla L(\alpha(t)) \tag{4.8}$$

While $\nabla L(\alpha(t))$ contains information derived from the training data (signal) it also includes noise due to the data itself possibly being noisy, and the use of random minibatches by SGD. Thus, each update of α contains both noise and signal. To account for this observation, we model the training process using the following suppositions:

Supposition 1 After t steps of training, the weight matrix at layer l can be decomposed as

$$W_l(t) = R_l(t) + S_l(t) \tag{4.9}$$

where $S_l(t)$ represents the signal (structured information learned from the data) and $R_l(t)$ is an independent random noise perturbation.

As training progresses, we expect the magnitude of the signal $\|S_l(t)\|_F$ to increase while the magnitude of the noise $\|R_l(t)\|_F$ decreases, where $\|\cdot\|_F$ is the Frobenius norm of a matrix. The theory will explore this question in detail. Given this decomposition, we will use the MP distribution to separate the signal from the noise. To do so, we assume that the signal in W_l is strong enough to be distinguishable from the noise. Since the singular values of R_l are determined by the MP distribution, with maximal value $\sqrt{\lambda_+}$, this supposition can be reformulated as:

Supposition 2 $S_l(t)$ has low rank r, with r not depending on N.

In Section 4.3, we will use Supposition 2 to design a pruning strategy and apply it to VIT models. In Section 4.4, we formalize these suppositions and prove that removing $R_l(t)$ does not decrease accuracy (see Lemma 4.4.4). The theory, through Theorem 4.2.2, provides insights into how the training reduces the noise in weight matrices. We first present our result for a simplified case where the DNN has only three weight layer matrices and no bias vector. In future works, the results will be extended to more general ViT models.

4.3 Pruning Visual Transformers (ViTs)

ViTs have emerged as state-of-the-art models for image classification tasks on benchmarks such as ImageNet [83]. Most weights in ViTs are organized as dense matrices, representing affine transformations through matrix-vector products, such as in Multi-Layer Perceptrons (MLPs) and Attention layers [338]. This structure makes ViTs particularly compatible with the framework of RMT. For further details on the architecture of ViTs, see [83].

4.3.1 Pruning strategy based on Random Matrix Theory

In this section, we detail our pruning strategy that leverages insights from RMT. The pruning process involves selectively removing weights from a DNN based on certain criteria derived from RMT. Our goal is to reduce the model size while preserving its accuracy.

Our pruning strategy is based on the suppositions made in Section 4.2.3, and extensive experiments on ViT models. Given our supposition that the weight matrices are perturbed low-rank structures, we can prune them by setting small values, which are likely noise, to zero. We define the pruning function for $x, \theta \in \mathbb{R}$:

$$\text{Prune}(x, \theta) = \begin{cases} x \text{ if } |x| > \theta \\ 0 \text{ if } |x| \leq \theta \end{cases} \tag{4.10}$$

This function can be vectorized for matrices $W \in \mathbb{R}^{N \times M}$ s.t. $\text{Prune}(W, \theta)_{i,j} = \text{Prune}(W_{ij}, \theta)$.

There are three types of pruning that can be done:

1. **Singular vector pruning:** Small values in the singular vectors may be interpreted as noise. This step helps protect the model during pruning by enhancing the low-rank structure and preserving accuracy.

2. **Singular value pruning:** Although small singular values could be considered noise according to our suppositions, experiments showed that pruning these values significantly degraded the network's performance. Therefore, this technique was excluded from our final pruning procedure.

3. **Direct coefficients pruning:** We may also directly set to zero small coefficients in the weights layer, as they may be interpreted as noise.

After pruning, we observe that applying a combination of L_1 and L_2 regularization helps in refining the weight matrices, promoting sparsity while maintaining the network's performance.
To guide the pruning and define what is interpreted as noise, we introduce two key metrics:

- **Spike Metric γ:** This metric quantifies the proportion of spikes in the weight matrices of the layers. It is defined as:

$$\gamma = \frac{1}{N} \cdot \text{Card}\left\{\sigma \leq \sqrt{\lambda_+ N},\ \text{for } \sigma \text{ singular value of } W_\ell\right\}$$

  where N is the size of the weight matrix W_ℓ, and λ_+ is the upper edge of the Marchenko-Pastur distribution fitted to the Empirical Spectral Density (ESD) of $X_\ell = \frac{1}{N} W_\ell^T W_\ell$. In the base ViT described below, its average value is 88.55%. This metric suggests how large the rank of the matrix S is compared to the total rank of W. See Appendix C.4 for more on spikes.

- **MP Fit Metric μ:** This metric measures the error in fitting the MP distribution to the ESD of $X_\ell = \frac{1}{N} W_\ell^T W_\ell$. It is computed based on a fitting algorithm (see Subsection C.4.3). In the base VIT described below, its average value is 0.3.

4.3.1.1 Singular vector pruning

For a triplet (v, w, σ) of respectively left, right singular vectors and singular value, we apply pruning such that:

$$v \leftarrow \text{Prune}\left(v, \theta \max\left\{\frac{1}{750}, \left[1 - \frac{\sigma}{\sqrt{\lambda_+}}\right]^{30}\right\}\right) \tag{4.11}$$

A similar update is performed on w. The specifics of this pruning were found by extensive experiments. The pruning of the components of the singular vectors is based on their size, with smaller components being pruned. The threshold of this

pruning is based on $\theta = .00001125 \times r \times N \times M$, with N and M being the sizes of the matrix. This process reduces the complexity of the model by removing less significant components of the singular vectors. It seems to "protect" the DNN during the pruning phase and make the weight layers "more" low rank, see Subsection C.4.5.

4.3.1.2 Direct coefficients pruning

To prune the weight matrices, we use the metric $\zeta_1(t)$, which depends on the randomness of the layer and the current pruning cycle t. It is defined as:

$$\zeta_1(t) = [(1-\mu)\cdot\gamma]^{\frac{1.5}{t}} \cdot r \cdot \text{Card}\left\{|W_\ell|_{ij} > 0\right\} \tag{4.12}$$

where:

- μ is the MP Fit Metric of the layer.
- γ is the spike metric of the layer.
- r is the desired proportion of parameters to remove in each pruning cycle (set to $r = 6\%$).
- t is the current pruning cycle (with total cycles $N_{\text{cycle}} = 19$).
- $\text{Card}\left\{|W_\ell|_{ij} > 0\right\}$ is the number of non-zero parameters in the weight matrix W_ℓ.

The above equation for $\zeta_1(t)$ was derived through a process of trial and error, but it also embodies some common sense principles. In particular, weight matrices that are determined to be more random—based on the randomness criteria provided by the MP Fit Metric (μ) and the Spike Metric (γ)—are pruned more aggressively in the initial pruning cycles. This approach leverages the idea that less structured (i.e., more random) weights are likely to be less critical to the network's performance.

The exponent $\frac{1.5}{t}$ ensures that as we progress through the pruning cycles (i.e., as t increases), the influence of the randomness metrics μ and γ diminishes. This design allows us to focus more on pruning layers that are identified as more random in the early cycles and shift towards a more uniform pruning strategy in later cycles. Using this, each matrix is updated using the following rule:

$$W_\ell \leftarrow \text{Prune}(W_\ell, f\max\{3, 5[(1-\mu)\gamma]^{\frac{1.5}{t}}\}) \tag{4.13}$$

where f is the pruning factor. f is found to be the smallest number so that the total amount of weights pruned in the matrix is larger or equal to $\zeta_1(t)$. In practice, f is initialized at $f = 1e-6$ and increased by steps of $5e-6$ until the total amount of weights pruned is more than $\zeta_1(t)$.

4.3.1.3 Regularization

Regularization helps mitigate potential accuracy loss due to pruning. It also helps in refining the weight matrices after pruning, promoting sparsity while maintaining the network's performance.

After every pruning cycle, we minimize the loss function

$$\mathcal{L} = \mu_1 \|W\|_1 + \mu_2 \|W\|_2^2,$$

with $\mu_1 = 5 \times 10^{-6}$ and $\mu_2 = 2 \times 10^{-6}$ using SGD and a learning rate of 0.05×10^{-6} (without any training data), so that although an infinite number of minimization steps would drive all weights to zero, the finite number of epochs applied causes the weights to simply become smaller—with some small weights vanishing—thus allowing us to decay the weight magnitudes without directly pruning them. Again, note that no training data is used during this regularization, and there are no other constraints for this minimization.

4.3.2 Pruning procedure overview

Having defined the different steps used in our procedure, we may now describe the overall process. To progressively remove weights, we perform $N_{cycle} = 19$ times the following pruning cycle. First, we **prune the singular vectors** (only every other cycle). Then, coefficients of the weight matrix are pruned by **direct coefficient pruning**. Finally, we perform **regularization** for n_{reg} epochs. We start with $n_{reg} = 15$ and increase by 5 epochs each cycle, up to a maximum of 40 epochs.

Fine-tuning Finally, after completing all pruning and regularization cycles, we fine-tune the model to recover any potential loss in accuracy. This fine-tuning is performed while keeping the pruned weights frozen, thus keeping the sparsification we obtained through the procedure described above. We use SGD with a momentum of 0.9, a base learning rate of 3×10^{-5} with the cosine annealing scheduler and a warmup phase of 2 epochs (learning rate divided by 100). The model is fine-tuned for less than 20 epochs over the entire training set using cross-entropy loss (the exact number of epochs is given in the results below). Finally, the gradient norm is

clipped at 1 to prevent exploding gradients.
Results are presented before and after fine-tuning.

4.3.3 Results for the visual transformer Vit_Base_Patch16_224 Model

Our numerical simulations explore the pruning of a pre-trained `vit_base_patch16_224` model using RMT. This model has approximately 86.5 million parameters and achieves a 85.1% top-1 accuracy on the ImageNet validation set. We reduce the DNN's size by 30-40% and evaluate its accuracy. We assess the model's top-1 and top-5 accuracy on the ImageNet validation set, consisting of $\approx 200,000$ objects.

Remark 4.3.1. *Some layers of this DNN model fit the MP distribution (based on Algorithm C.4.3) with an error of* 1% *while others fit the MP distribution with an error of over* 50%*, see Subsection C.4.7 for more. The spike metric and MP fit metric depend on α; see Subsection C.4.7. When $\alpha = .25$, we have that the MP fit metric of this ViT model is* .18.

As the pruning progresses, the process yields a series of models with varying levels of sparsity and performance metrics. The final model selection can be based on a trade-off between accuracy and complexity. See Fig. 4.1 for the accuracies of the ViT model vs the number of parameters kept (there is no fine-tuning yet). We see that the DNN that was pruned by approximately 30% has a reduction in top-1 accuracy by only 1.1%. After 1 epoch of fine-tuning the ViT model on the ImageNet training set, the DNN accuracy improved to 84.5%, resulting in a total decrease in accuracy of only .6%.



(a) Top 1 accuracy vs params. kept (b) Top 5 accuracy vs params. kept

Figure 4.1: The top 1 accuracy and top-5 accuracy of ViT-base vs. percentage of parameters kept for pruning through RMT-based sparsification and no fine-tuning. The accuracy of the DNN is given as a percentage above the data points.

4.3.4 Pruning the visual transformer Vit_large_Patch16_224 Model

We performed the same pruning strategy on the Vit_large_Patch16_224 Model. This ViT model is bigger than the one in the previous subsection, having approximately

300 million parameters. It achieves a 85.85% accuracy on the ImageNet validation set. In Fig. 4.2, we see the accuracy vs percentage of parameters kept for this pruning. After pruning 30% of the DNN parameters, the accuracy only drops by .8%. Again, after fine-tuning for only one epoch, we recover that full reduction in accuracy (i.e., has a 85.85% accuracy). Finally, we also took the DNN for which 50% of the parameters were pruned and fine-tuned for 12 epochs, obtaining a 84.48% accuracy after the fine-tuning.



(a) Top 1 accuracy vs params. kept (b) Top 5 accuracy vs params. kept

Figure 4.2: The top 1 accuracy and top-5 accuracy of ViT-large model vs. percentage of parameters kept for pruning through RMT-based sparsification and no fine-tuning. The accuracy of the DNN is given as a percentage above the data points.

4.3.5 Pruning results for ViT-B16/224 and ViT-L16/224 on ImageNet

In this section, we compare our pruning results with CP-ViT [316], using the same architecture but different pre-trained weights. Our initial accuracies are higher: 85.1% for ViT-B16/224 and 85.85% for ViT-L16/224, compared to 76.5% and 77.91% in CP-ViT. Figure 4.3 shows the pruning results with FLOP reduction and accuracy reduction. Our fine-tuning was done for less than 20 epochs, while the fine-tuning in [316] was done with 30 epochs.

4.3.6 RMT-based pruning of DeiT

We perform the same pruning strategy with the same hyperparameters from Subsection 4.3.1 on the three DeiT models, tiny, small and base. Table 4.1 shows the result of our RMT-based pruning method and compares them with other pruning methods, both with and without fine-tuning. For these models, we fine-tune for only 7 epochs, which is much less than the fine-tuning done in the other works; for example, in [316], they fine-tune for 30 epochs. From Table 4.1 and Fig. 4.3, we see that we always outperform [316] (and the other works) when there is no fine-tuning. Furthermore, Fig. 4.3 shows that we outperform [316] for large DNNs even with fine-tuning, which makes sense from an RMT perspective given that the larger the DNN and the more parameters we have, the more RMT applies. Given

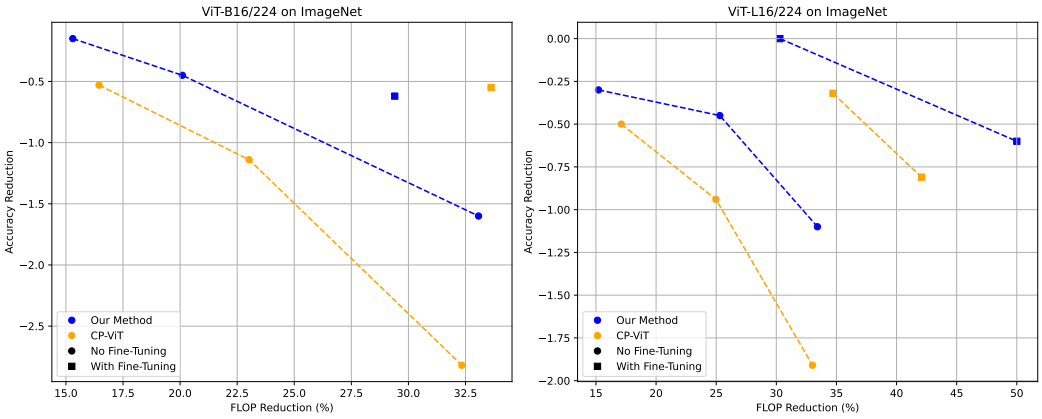


Figure 4.3: Comparison of pruning results between our method and CP-ViT [316]

the massive DNNs being trained these days, with over a trillion parameters, this RMT perspective becomes especially important. For the Vit_Base_Patch16_224 Model, after fine-tuning, we essentially achieve the same results as [316], but we only fine-tune for one epoch while they fine-tune for 30.

4.4 Main theoretical results

4.4.1 Assumptions for main theorems: Gaussian case

For simplicity of the presentation, we choose a DNN with only three weight layer matrices and have other simplifying assumptions. However, many of the assumptions on the DNN and the weight layer matrices can be relaxed. For example, we can include arbitrary layers (see Subsection 4.5.1), components of R_2 can be i.i.d. from a distribution different than the normal distribution, and other activation functions can be used (such as Leaky ReLU). In Section 4.5, we present these results for more general DNN structures. Recall the ℓ_1 norm of a matrix W with entries w_{ij}:

$$\|W\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |w_{ij}|.$$

Assumption 4.4.1. ***Architecture of the DNN and bounds on the norms of weight layer matrices:*** *We consider the following DNN ϕ:*

$$X(\alpha, s) = \lambda \circ W_3 \circ \lambda \circ W_2 \circ \lambda \circ W_1 s, \quad s \in T, \tag{4.14}$$

$$\phi(\alpha, s) = \rho \circ X(\alpha, s) \tag{4.15}$$

where W_1, W_2, W_3 are the weight layer matrices and λ is the absolute value or ReLU activation function, T is the training set, $X(\alpha, s)$ outputs the components of the

Table 4.1: Comparison with different ViT pruning methods on the ImageNet dataset. The accuracy without finetuning was not mentioned in VTP, PoWER, and HVT but were replicated in [316]. We fine-tune for only 7 epochs on the ImageNet training set. We see that the larger the DNN, the better our method performs, which makes sense from an RMT perspective. This is also reassuring, given that modern AI models are often much larger than these models.

	Not Finetune		Finetune	
Model	Top-1 Acc.(%)	FLOPs Saving	Top-1 Acc.(%)	FLOPs Saving
DeiT-Ti 16/224 [332]				
Baseline [332]	72.20	-	72.20	-
VTP [382]	69.37 (-2.83)	21.68%	70.55 (-1.65)	45.32%
PoWER [123]	69.56 (-2.64)	20.32%	70.05 (-2.15)	41.26%
HVT [249]	68.43 (-3.77)	21.17%	70.01 (-2.19)	47.32%
CP-ViT [316]	71.06 (-1.14)	23.02%	71.24 (-0.96)	43.34%
RMT-based (ours)	**71.09 (-1.11)**	24.83%	68.4 (-3.8)	47.23%
DeiT-S 16/224 [332]				
Baseline [332]	79.80	-	79.80	-
VTP [382]	77.35 (-2.45)	20.74%	78.24 (-1.56)	42.52%
PoWER [123]	77.02 (-2.78)	21.46%	78.30 (-1.50)	41.36%
HVT [249]	76.72 (-3.08)	20.52%	78.05 (-1.75)	47.80%
CP-ViT [316]	78.84 (-0.96)	20.96%	79.08 (-0.72)	42.24%
RMT-based (ours)	**78.91 (-0.89)**	23.72%	78.34 (-1.46)	41.92%
DeiT-B 16/224 [332]				
Baseline [332]	81.82	-	81.82	-
VTP [382]	79.46 (-2.36)	19.84%	80.70 (-1.12)	43.20%
PoWER [123]	79.09 (-2.73)	20.75%	80.17 (-1.65)	39.24%
HVT [249]	78.88 (-2.94)	20.14%	79.94 (-1.88)	44.78%
CP-ViT [316]	80.91 (-0.91)	22.16%	81.13 (-0.69)	41.62%
RMT-based (ours)	**81.44 (-0.38)**	21.38%	80.58 (-1.24)	44%

DNN, and ρ is the softmax. We also denote $\{W_l(t)\}_{l=1}^3$ the weight layer matrices to highlight the dependence in training time t.

We wish to study the behavior of the inner layer W_2 for large N, as training progresses. Thus, we assume the outer layers satisfy: For all s in the training set T, the quantity

$$a(N, s) := \frac{\|W_3\|_1 \|W_1 s\|_2}{N^{1.5/4}} \tag{4.16}$$

converges towards 0 as $N \to \infty$ (see Subsection C.2.3 for more details).

Assumption 4.4.2. ***Deformed matrix form of matrix W_2:***

The $N \times N$ matrix $W_2(t)$, is such that:

$$W_2(t) = R_2(t) + S_2(t), \quad (\textit{deformed matrix}) \tag{4.17}$$

where $R_2(t)$ is a matrix with i.i.d. random normal entries that have mean zero and variance less than $\frac{1}{N}$, i.e. such that $R_2(t)_{i,j} \overset{i.i.d.}{\sim} \mathcal{N}(0, \frac{g(t)}{N})$, where $g(t) \leq 1$. The bound of $g(t)$ is unimportant and taken to be less than 1 for simplicity. It is necessary that $Var[R_2(t)_{i,j}] = O(\frac{1}{N})$ so that $R_2(t)$ has a bounded spectral norm as $N \to \infty$. Finally, we assume that $S_2(t)$ is independent of $R_2(t)$.

Remark 4.4.3. *Similar results will apply with a rectangular matrix $W_2(t) \in \mathbb{R}^{N \times M}$ such that*

$$\frac{M}{N} \to c \in (0, +\infty) \tag{4.18}$$

Assumption 4.4.4. ***Low-rank nature of** $S_2(t)$**:***

$$S_2(t) = \sum_{i=1}^{N} \sigma_i(t)\, u_i(t)\, v_i^T(t) = \sum_{i=1}^{r} \sigma_i(t)\, u_i(t)\, v_i^T(t),$$

with

$$\sigma_i(t) = 0 \quad (N \geq i > r),$$

and the rank r is fixed (i.e., independent of N), confirming the low-rank nature of $S_2(t)$.

Remark 4.4.5. *For a detailed numerical analysis of $a(N, s)$, see Subsection C.2.3.*

4.4.2 A theorem on the asymptotic magnitude of the noise

We consider the cross-entropy loss function with L2 regularization for a DNN $\phi(\cdot, \alpha(t))$, expressed as follows:

$$L(\alpha(t)) = -\frac{1}{|T|} \sum_{s \in T} \log\left(\phi_{C(s)}(s, \alpha(t))\right) + \mu \sum_{i=1}^{L} \|W_i(t)\|_F^2, \mu > 0, \tag{4.19}$$

where $C(s)$ denotes the correct class of the object s. The term μ acts as a regularization hyperparameters and prevents overfitting in the DNN.

Theorem 4.4.1. *Suppose Assumptions 4.4.1, 4.4.2 and 4.4.4 hold. If the parameters α of the DNN converge as $t \to \infty$ to a local minimum of the loss* (4.19)*, denoted $\alpha^*(N)$, as $t \to \infty$, then $\forall \epsilon > 0$:*

$$\lim_{N \to \infty} \lim_{t \to \infty} \mathbb{P}(\|R_2\|_F \leq \epsilon) \to 1. \tag{4.20}$$

The proof of this theorem can be found in Section C.3.3.

The theorem states that the random part of the weight matrix W_2 vanishes as the network is trained and its parameters converge towards a local minimum ($\alpha(t) \to \alpha^*$). It is impossible to decompose W_2 into $R_2 + S_2$, where R_2 satisfies Assumption 2 and S_2 satisfies Assumption 3. If such a decomposition were possible, the ESD of W_2 would exhibit an MP bulk and spikes (see [17, 31, 70] and Appendices C.3.3 and C.4.1). However, at the local minimum, as we will show, W_2 's ESD lacks these features, ruling out any such decomposition.

It should be noted that for a matrix R with i.i.d, centered Gaussian entries,

$$\frac{1}{N}\|R\|_2^2 \underset{N\to\infty}{\overset{a.s.}{\sim}} 4\, Var[R_{ij}] \tag{4.21}$$

$$\|R\|_F^2 \underset{N\to\infty}{\overset{a.s.}{\sim}} N^2 Var[R_{ij}]. \tag{4.22}$$

Through Assumption 2, we assume i.i.d Gaussianity with mean zero, and $Var[R_{ij}] = O(\frac{1}{N})$. However, we may see from (4.22) that this does not straightforwardly imply $\|R\|_F \to 0$, as the theorem proves. Hence, we prove that the random fluctuations of the weight matrices have to be much smaller than originally assumed.

One might assume that a local minimum of the loss is deterministic, and thus the random part of the weights would have to vanish. While intuitive, this reasoning is not accurate. First, the theorem states that for **any** decomposition of W_2 into R_2+S_2 satisfying the above assumptions we would have that $\lim_{N\to\infty}\lim_{t\to\infty}\mathbb{P}\left(\|R_2\|_F \leq \epsilon\right) \to 1$. This means that at a local minimum of the loss if one takes a nonzero random matrix R_2 and then sets $S_2 = W_2 - S_2$, we would have that S_2 will **not** satisfy Assumption 3 (with some probability depending on N). Otherwise, the theorem would be violated. Furthermore, the training data T itself might have noise, and so the loss landscape, which depends on the training data, might have some noise/randomness. That is, the local minima might be somewhat random (meaning that the parameters α at the local min might be random). However, when we train with regularization, so long as we can decompose W_2 into $R_2 + S_2$ satisfying Assumptions 2 and 3, we have that $\lim_{N\to\infty}\lim_{t\to\infty}\mathbb{P}\left(\|R_2\|_F \leq \epsilon\right) \to 1$.

The role of the regularization in (4.19) is well known in Machine Learning. In particular, it limits the amount of noise from the data being learned, as the DNN's weight may otherwise reflect the random nature of the data, see [321]. In our theory, it is essential in showing that $\|R\|_F \to 0$. Thus regularization successfully mitigates randomness in the network. It is important to note that Assumption 2, which ensures that the matrix $S_2(t)$ has no small nonzero singular values, essentially is a statement about the data T. It says that any noise from the data cannot be (in

some sense) "much larger" than the deterministic part of the data (relative also to the hyperparameter μ).

It is interesting to note that this phenomenon does not necessarily occur when there is no regularization. In fact, without regularization, one can replace the random matrix R_2 with a different random matrix (or add any random matrix to W_2), and the magnitude of the loss will not change. Thus, around every local minima, there would be a flat region of minima, all corresponding with parameters $W_2 + R_2^*$, with R_2^* being some random matrix satisfying Assumptions 2 and 3.

4.4.3 A theorem on how pruning randomness reduces loss

We can also show that reducing the randomness in the parameters reduces the loss. Specifically, we show that for $W_2 = R_2 + S_2$, if we replace W_2 in the DNN with S_2 (at some fixed time) then with some probability (which goes to 1 as $N \to \infty$) we have that the loss of the DNN with S_2 is smaller than the loss of the DNN with W_2 by the amount $\mu\|R_2\|_F^2$.

Theorem 4.4.2. *Suppose Assumptions 4.4.1, 4.4.2 and 4.4.4 hold. For $W_2 = R_2 + S_2$, remove R_2 so that the second weight layer matrix is S_2 alone. Then $\forall \epsilon > 0$ and any fixed training time t we have*

$$\lim_{N\to\infty} \mathbb{P}(L(\alpha_{S_2}) < (1+\epsilon)(L(\alpha_{W_2}) - \mu\|R_2\|_F^2)) \to 1. \tag{4.23}$$

Furthermore, suppose Assumptions 4.4.1 and 4.4.2 hold alone (S_2 need not be low rank). Then $\exists G(N) \to 0$ as $N \to \infty$ s.t.

$$\lim_{N\to\infty} \mathbb{P}\left(|acc_{\alpha_{W_2}}(t) - acc_{\alpha_{S_2}}(t)| \leq G(N)\right) \to 1, \tag{4.24}$$

where α_{S_2} and α_{W_2} represent the DNNs with weight layer matrices S_2 and W_2 respectively, L is the loss function in (4.19) *and μ is the regularization*

The proof for this theorem can be found in Subsection C.3.3. For a numerical simulation that shows that removing (or adding) a random matrix from the DNN reduces its loss, see Example C.1.3. This theorem demonstrates that pruning random weights in a DNN reduces its loss. While Theorem 4.4.1 suggests that randomness vanishes near a local minimum, Theorem 4.4.2 provides a partial answer to whether removing randomness accelerates convergence, showing that it at least lowers the loss. The intuition behind this result lies in the perturbations caused by R_2 : the

cross-entropy term is not affected by adding or removing R_2, while the regularization term is influenced by its Frobenius norm and decreases as we remove R_2. Since the regularization perturbation dominates (as shown in (4.22) and (4.21)), removing R_2 significantly reduces the loss. Theorem 4.4.2 formalizes this, showing that removing the random part R_2 of the weight matrix W_2 in the DNN $\phi(\alpha, s)$ reduces the loss by an amount proportional to $\|R_2\|_F^2$. Specifically, when $W_2 = S_2$ (i.e., R_2 is removed), the loss satisfies:

$$L(\alpha_{S_2}, s) + \mu\|R_2\|_F^2 < (1+\epsilon)L(\alpha_{W_2}, s), \tag{4.25}$$

with high probability. This implies that the loss decreases by approximately $\mu\|R_2\|_F^2$ when R_2 is removed, as the regularization term involving R_2 vanishes.

This effect can be extended by introducing a scaling factor $\gamma \in [0, 1]$ to R_2 , defining the modified weight matrix as:

$$W_2(\gamma) = \gamma R_2 + S_2. \tag{4.26}$$

As γ decreases from 1 to 0, the randomness in W_2 is reduced, and the corresponding loss decreases proportionally by $\gamma\mu\|R_2\|_F^2$.

Thus, we can obtain the following corollary:

Corollary 4.4.3. *Suppose Assumptions 4.4.1, 4.4.2 and 4.4.4 hold. For any* $\gamma \in [0, 1]$*, define the modified second-layer weight matrix by*

$$W_2(\gamma) = \gamma R_2 + S_2,$$

and let $\alpha_{W_2(\gamma)}$ *denote the corresponding DNN. Then for every* $\epsilon > 0$ *and any fixed training time t, we have*

$$\lim_{N\to\infty} \mathbb{P}\Big(L(\alpha_{W_2(\gamma)}) < (1+\epsilon)\Big(L(\alpha_{W_2}) - (1-\gamma)\,\mu\,\|R_2\|_F^2\Big)\Big) \to 1.$$

The proof of this Corollary can be found in Subsection C.3.3. Supporting evidence from [321] shows that removing small singular values in weight matrices (reducing noise) improves accuracy for DNNs trained on noisy data. Additionally, numerical simulations (e.g., Example C.1.1) confirm that MP-based pruning reduces loss and increases accuracy, highlighting the practical benefits of this approach. See Appendices C.1 and C.2 for numerical results on classification and regression problems that confirm these theoretical findings.

4.4.4 Key lemmas: how removing randomness from DNN weight layers affects the output, loss and accuracy of a DNN

To prove the previous theorems, we rely on two key lemmas presented here. These lemmas demonstrate that replacing W_2 with its deterministic part has minimal effect on the DNN's output, accuracy, and loss. Since the random component primarily acts as noise, its removal results in negligible changes.

Lemma 4.4.4. *Suppose Assumptions 1 and 2 hold, and suppose we replace the weight layer matrix W_2 with the matrix S_2. Then*

$$\mathbb{P}\left(|X_i(s, \alpha_{S_2}) - X_i(s, \alpha_{W_2})| \leq a(N, s)\right) \geq 1 - 2\exp(-\frac{N^{1/4}}{2}). \tag{4.27}$$

Here, $a(N, s)$ =(4.16), and α_{S_2} are the parameters of the DNN, which has the weight matrix S_2 and α_{W_2} are the parameters of the DNN with the weight layer matrix W_2.

For a proof of the Lemma see Subsection C.2.2. This lemma examines the impact of removing the random component from a weight matrix in a DNN layer. It shows that replacing W_2 with $S_2(t)$ alone results in a small difference in the DNN's output with high probability. Intuitively, the random component $R_2(t)$ acts as noise and has a diminishing effect in large networks due to averaging. The part $S_2(t)$, which captures the network's learned structure, dominates the behavior. As a result, removing $R_2(t)$ has negligible impact on the output, reinforcing that the deterministic component sufficiently preserves the network's function. To prove Theorem 4.4.2 we use this lemma and then show that the regularization part of the loss decreases by $\mu\|R_2\|_F$ when we replace W_2 with S_2.

4.5 Generalized Main Theoretical Results

In this section, we generalize the previous results for DNNs with different architectures and for the case where the random matrix R is not necessarily i.i.d Gaussian.

4.5.1 Assumptions for Generalized Main Theorems

Assumption 4.5.1. *Consider a DNN denoted by ϕ, and assume that it can be written as:*

$$\phi = \rho \circ \psi_2 \circ (R + S) \circ \psi_1, \tag{4.28}$$

where ψ_1 and ψ_2 are arbitrary functions and ρ is softmax.

Take $W = R + S$ and assume that the matrices R and S satisfying the following assumptions:

Assumptions on the matrices R and S

We considered a class of admissible matrices W, where $W = R + S$ and W, R and S satisfy the following three assumptions:

Assumption 4.5.2. *Assume R is a $N \times M$ matrix such that for any vector v we have:*

$$\mathbb{P}(\|Rv\|_\infty > d_1(N)J(v)) \leq d_2(N), \tag{4.29}$$

with $d_1(N), d_2(N) \to 0$ as $N \to \infty$ and $J(v)$ some function that depends on v alone. Further, as $N \to \infty$, we have that $\sigma_{\max}(R) \to \sqrt{\lambda_+}$ a.s. Finally, we assume that $S(t)$ is independent of $R(t)$.

Example 4.5.1. *For example, when R is i.i.d Gaussian, one can show, using the Borell-TIS inequality (see Theorem C.2.2), that:*

$$\mathbb{P}\left(\|Rv\|_\infty > (\sqrt{\frac{2\log N}{N}} + \frac{1}{N^{\frac{3}{8}}})\|v\|_2\right) \leq 2\exp\left(-\frac{N^{\frac{1}{4}}}{2}\right)$$

See Subsection C.2.2 for a proof.

We then assume the following for the matrix S:

Assumption 4.5.3. *Assume S is a matrix with $S = \sum_{i=1}^{r} \sigma_i u_i v_i^T = U\Sigma V^T$, with σ_i the singular values and u_i, v_i^T column and row vectors of U and V. Thus, S has r non-zero singular values corresponding to the diagonal entries of Σ, and all other singular values of S are zero. We also assume that these r singular values of S have multiplicity* 1.

Finally, we assume for $W := R + S$:

Assumption 4.5.4. *For γ a variable between* 0 *and* 1*, take $W(\gamma) = \gamma R + S$. Then, all the singular values of $W(\gamma)$ decrease monotonically a.s. as $N \to \infty, \gamma \to 0$ (see Theorem C.3.2 as an example).*

Empirically, it has been observed that these assumptions are reasonable for weight matrices of a DNN; see [321, 330]. There are various spiked models in which Assumptions 4.5.2-4.5.4 hold, for more on the subject see [3, 17, 19, 20, 31, 60,

70, 86, 86, 181, 234, 245, 375]. Also, a number of works in RMT addressed the connection between a random matrix R and the singular values and singular vectors of the deformed matrix $W = R + S$, see [30, 31].

These assumptions are quite natural and hold for a wide range of DNN architectures. Assumption 4.5.2 focuses on the random matrix R. This assumption ensures that the random matrix R captures the essential randomness in the weight layer while also satisfying the requirements given in Theorem 4.2.2.

Assumption 4.5.3-4.5.4 pertains to the matrix S, which is assumed to have a specific structure, with r non-zero singular values and all other singular values being zero. Moreover, these r singular values have multiplicity 1, which is a reasonable expectation for a strongly correlated matrix that contributes to the information content in the weight layer matrix W.

The assumption that the singular values of the matrix S are larger than some $\bar{\theta}(\lambda_+)$ is also quite natural, see [321, 330]. This is because the matrix S represents the information contained in the weight layer, and its singular values are expected to be large, reflecting the importance of these components in the overall performance of the DNN. On the other hand, the random matrix R captures the inherent randomness in the weight layer, and with high probability depending on N, its singular values should be smaller than the MP-based threshold. This means that there is a clear boundary between the information and noise in the layer W, which is also natural, see [321].

This distinction between the singular values of S and R highlights the separation between the information and noise in the weight layer, allowing us to effectively remove the small singular values without impacting the accuracy of the DNN. The assumption thus provides a solid basis for studying the behavior of DNNs with weight layers modeled as spiked models. It contributes to our understanding of the effects of removing small singular values based on the random matrix theory MP-based threshold $\sqrt{\lambda_+}$.

One can show that the following two simpler properties on the matrices R and S are sufficient to ensure that R and S satisfy the above Assumptions 4.5.2-4.5.4.

Recall that a bi-unitary invariant random matrix R is a matrix with components taken from i.i.ds such that for any two unitary matrices U and V^T, the components of the matrix URV^T have the same distribution as the components of R. We then assume:

Property 1 (statistical isotropy): Assume R to be a bi-unitary invariant random $N \times M$ matrix with components taken from i.i.ds with zero mean and variance $\frac{1}{N}$.

We then assume the following for the matrix S:

Property 2 (low rank of matrix): Assume S is a matrix with $S = \sum_{i=1}^{r} \sigma_i u_i v_i^T = U\Sigma V^T$, with σ_i the singular values and u_i, v_i^T column and row vectors of U and V. Thus, S has r non-zero singular values contained on the diagonal entries of Σ, and all other singular values are zero. We also assume that these r singular values of S have multiplicity 1. Finally, we assume that $\frac{N}{M} \to c \in (0, +\infty)$ as $N \to \infty$.

An explicit relationship between Properties 1-2 and Assumptions 4.5.2-4.5.4 can be found in [31]. The Property 1 is indeed strong, as it implies that the random matrix R is random in every direction. In other words, for any unitary matrices U and V^T, the matrix URV^T has the same distribution as R. Random matrices with complex Gaussian entries, also known as Ginibre matrices, are a class of random matrices that are bi-unitary invariant [170].

For a DNN ϕ satisfying Assumption 4.5.1, we start by defining,

$$h_\phi(s) := J(\psi_1 s)\|\psi_2\|_{\max\to\max}, \tag{4.30}$$

where $\|\cdot\|_{\max\to\max}$ is the induced operator norm (or global Lipschitz constant with respect to ℓ_∞)

$$\|A\|_{\max\to\max} := \sup_{v\neq w} \frac{\|A(v) - A(w)\|_{\max}}{\|v - w\|_{\max}} \tag{4.31}$$

and $J(\cdot)$ is given in (4.5.1).

4.5.2 Main Theoretical results for RMT-based decrease in loss

In the context of analyzing DNNs, we consider the cross-entropy loss function with L2 regularization for a DNN $\phi(\cdot, \alpha(t))$, expressed as follows:

$$L(\alpha(t)) = -\frac{1}{|T|} \sum_{s\in T} \log\left(\phi_{C(s)}(s, \alpha(t))\right) + \mu \sum_{i=1}^{L} \|W_i(t)\|_F^2, \tag{4.32}$$

where W_i are the weight layer matrices of the DNN, and $C(s)$ denotes the correct class of the object s. The term μ acts as a regularization constant, which is employed to prevent overfitting in the DNN.

Theorem 4.5.2. *Let ϕ be a DNN satisfying Assumptions 4.5.1-4.5.4. Assume the following:*

1. *$\forall s \in T$, the quantity $a_\phi(N, s) := h_\phi(s) d_1(N) \to 0$ as $N \to \infty$ (with $h_\phi(s)$ defined in* (4.30) *and $d_1(N)$ coming from* (4.5.1) *see Subsection C.2.3 for more details).*

2. *The loss function* (4.19) *attains a local minimum at $\alpha^*(N)$, where N is the size of the matrix W.*

3. *As training time $t \to \infty$, the parameter $\alpha(t, N)$ converges to $\alpha^*(N)$.*

Then, $\forall \epsilon > 0$:

$$\lim_{N\to\infty} \lim_{t\to\infty} \mathbb{P}(\|R\|_F \leq \epsilon) \to 1. \tag{4.33}$$

We can also show that reducing the randomness in the parameters reduces the loss:

Theorem 4.5.3. *Assume the setup and conditions of Theorem 4.4.1, particularly regarding the DNN ϕ satisfying Assumptions 4.5.1-4.5.4. Assume that $\forall s \in T$, the quantity $a_\phi(N, s) := h_\phi(s) d_1(N) \to 0$ as $N \to \infty$ (see Subsection C.2.3 for more details).*

Let R be removed such that the second weight layer matrix is S alone. Then $\forall \epsilon > 0$ and for any fixed time t, we have

$$\lim_{N\to\infty} \mathbb{P}(L(\alpha_S) < (1+\epsilon)(L(\alpha_W) - \mu\|R\|_F^2)) \to 1 \tag{4.34}$$

Furthermore, suppose Assumptions 4.5.1 and 4.5.2 hold alone. Then $\exists G(N) \to 0$ as $N \to \infty$ s.t.

$$\lim_{N\to\infty} \mathbb{P}\Big(|acc_{\alpha_W}(t) - acc_{\alpha_S}(t)| \leq G(N)\Big) \to 1, \tag{4.35}$$

where α_S and α_W are the parameters of the DNNs with weight layer matrices S and W respectively, L is the loss function in (4.19) *and μ is the regularization hyperparameter.*

The proof of these theorems can be found in Subsection C.3.3.

4.5.3 Key Lemmas: how removing randomness from DNN weight layers affects the output, loss and accuracy of a DNN

In order to prove the main result, we will consider a few critical lemmas. These lemmas describe how the output, accuracy, and loss of a DNN remain largely unaffected when we replace the matrix W_2 with its part only. In other words, the random component of W_2 does not significantly affect the loss, accuracy, and output of the DNN. This outcome should be expected, as the random part should act more like noise, without causing substantial changes in the loss and accuracy. The following lemmas illustrate this concept.

Recall: We define the final output of the DNN before softmax as:

$$X = \psi_2 \circ (R + S) \circ \psi_1(s). \tag{4.36}$$

Lemma 4.5.4. *Given* (4.36)*, let $W + R + S$ be a $N \times N$ matrix such that $W(t) = R(t) + S(t)$, with $R(t)$ a random matrix with components i.i.d. and $S(t)$ a matrix satisfying Assumptions 4.5.1-4.5.2.*

Suppose we replace the weight layer matrix W with the matrix S. Then

$$\mathbb{P}\Big(|X_i(s, \alpha_S) - X_i(s, \alpha_W)| \leq d_1(N) h_\phi(s)\Big) \geq 1 - d_2(N)). \tag{4.37}$$

See (4.5.1) *for the details on $d_1(N)$ and $d_2(N)$, and* (4.30) *for the details of $h_\phi(s)$. Here, α_S are the parameters of the DNN, which has the weight matrix S and α_W are the parameters of the DNN with the weight layer matrix W.*

For a proof of the Lemma see Subsection C.2.2.

Acknowledgments

LB, HO and YS acknowledge support from NASA via the AIST program (Kernel Flows: Emulating Complex Models for Massive Data Sets). This work started when LB was on a sabbatical stay at Caltech hosted by H. Owhadi. Both LB and YS are grateful for the hospitality during their visit, supported by NASA.

The work of LB was partially supported by NSF grant DMS-2005262 and NSF grant IMPRESS-U 2401227. TB and HO acknowledge support from the Air Force Office of Scientific Research under MURI awards number FA9550-20-1-0358 (Machine Learning and Physics-Based Modeling and Simulation), FOA-AFRL-AFOSR-2023-0004 (Mathematics of Digital Twins), by the Department of Energy under award number DE-SC0023163 (SEA-CROGS: Scalable, Efficient, and Accelerated Causal

Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems). Additionally, HO acknowledges support from the DoD Vannevar Bush Faculty Fellowship Program.

Part II

Structure Discovery

Chapter 5

CO-DISCOVERING GRAPHICAL STRUCTURE AND FUNCTIONAL RELATIONSHIPS WITHIN DATA

Théo Bourdais, Pau Batlle, Xianjin Yang, Ricardo Baptista, Nicolas Rouquette, and Houman Owhadi. Codiscovering graphical structure and functional relationships within data: A Gaussian process framework for connecting the dots. *Proceedings of the National Academy of Sciences*, 121(32):e2403449121, 2024. doi: 10.1073/pnas.2403449121. URL https://doi.org/10.1073/pnas.2403449121.
TB was lead author: co-developed the CHD methodology and software, ran experiments, and co-wrote the manuscript.

5.1 Introduction

5.1.1 The three levels of complexity of function approximation.

As illustrated in Fig. 5.1.(a-c), Type 1, Type 2 and Type 3 problems can be formulated as completing or discovering hypergraphs where nodes represent variables and edges represent functional dependencies. The graph in Type 1 has only two variables and one unknown function. The graph in Type 2 has multiple variables and (some possibly unknown) functions, and the connectivity of the graph is known. The graph in Type 3 has an unknown connectivity (functional dependencies between variables may be unknown) and this is the focus of this work. Current methods for solving Type 1 and 2 problems include Deep Learning (DL) methods, which benefit from extensive hardware and software support but have limited guarantees. Despite their prevalence, Type 3 challenges have been largely overlooked due to their inherent complexity. Causal inference methods [121, 223] and probabilistic graphs [199, 324] and sparse regression methods [47, 88], offer potential avenues for addressing Type 3 problems. However, it is important to note that their application to these problems necessitates additional assumptions. Causal inference models, for instance, typically assume randomized data and some level of access to the data generation process or its underlying distributions. Sparse regression methods, on the other hand, rely on the assumption that functional dependencies have a sparse representation within a known basis. In this paper, we do not impose these assumptions, and thus, these particular techniques may not be applicable. Furthermore while the complexity of Bayesian causal inference methods may grow super-exponentially with the number d of variables, the complexity of our method is that of d parallel computations

of polynomial complexities bounded between $O(d)$ (best case) and $O(d^4)$ (worst case).

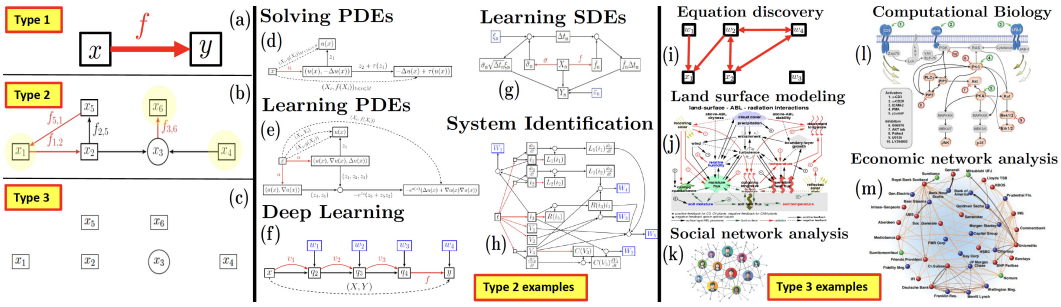


Figure 5.1: The three levels of complexity of function approximation.

5.1.2 Generalizing Gaussian Process methods.

Although Gaussian Process (GP) methods are sometimes perceived as a well-founded but old technology limited to curve fitting (Type 1 problems), they have recently been generalized, beyond Type 1 problems, to an interpretable framework (Computational Graph Completion or CGC [239]) for solving Type 2 problems [25, 55, 58, 74, 138, 240], all while maintaining the simple and transparent theoretical and computational guarantees of kernel/optimal recovery methods [215, 236]. This paper introduces a comprehensive GP framework for solving Type 3 problems, which is interpretable and amenable to analysis. This framework leverages the Uncertainty Quantification (UQ) properties of GP methods, which do not have an immediate natural counterpart in DL methods. It is based on a kernel generalization [243, 345] of variance-based sensitivity analysis guiding the discovery of the structure of the hypergraph. Here, variables are linked via GPs, and those contributing to the highest data variance unveil the hypergraph's structure. This GP variance decomposition of the data leads to signal-to-noise and a Z-score that can be employed to determine whether a given variable can be approximated as a nonlinear function of a subset of other variables.

5.1.3 The scope of Type 1, 2 and 3 problems.

The scope of Type 1, 2 and 3 problems is immense. Numerical approximation [236, 237, 292, 293], Supervised Learning, and Operator Learning [26, 136, 137, 290] can all be formulated as **Type 1 problems**, i.e., as approximating unknown functions given (possibly with noisy and infinite/high-dimensional) inputs/output data. The common GP based solution to these problems is to replace the underlying unknown function by a GP and compute its MAP estimator given available data. **Type 2 prob-**

lems include (Fig. 5.1.(d-h)) solving and learning (possibly stochastic) ordinary or partial differential equations [55, 74], Deep Learning [240], dimension reduction, reduced-order modeling, system identification [239], closure modeling, etc. Indeed, all these problems can be formulated as completing a computational graph [239]. In this formulation, variables and functions are represented by the nodes and the edges of the graph whose structure corresponds to the functional dependencies between variables. Some of the functions and variables may be unknown, and by completing, we mean approximating the unknown functions (colored in red in Fig. 5.1) given samples from the observed variables. The common GP-based solution to Type 2 problems is to simply replace unknown functions by GPs and compute their MAP/MLE estimators given available data and constraints imposed by the structure of the graph [239]. While most problems in Computational Sciences and Engineering (CSE) and Scientific Machine Learning (SciML) can be framed as Type 1 and Type 2 challenges, many problems in science can only be categorized as **Type 3 problems**, i.e., discovering the structure/connectivity of the graph itself from data prior to its completion. Indeed the scope of Type 3 problems extends well beyond Type 2 problems and includes equation discovery (Fig. 5.1.(i)); the modeling of land surface interactions in weather prediction (Fig. 5.1.(j) from [87], discovering possibly hidden functional dependencies between state variables for a finite number of snapshots of those variables); social network analysis (Fig. 5.1.(k) from [119], discovering functional dependencies between quantitative markers associated with each individual in situations where the connectivity of the network may be hidden); economic network analysis (Fig. 5.1.(m) from [300], discovering functional dependencies between the economic markers of different agents or companies, which is significant to systemic risk analysis); and computational biology (Fig. 5.1.(l) from [285], identifying pathways and interactions between genes from their expression levels).

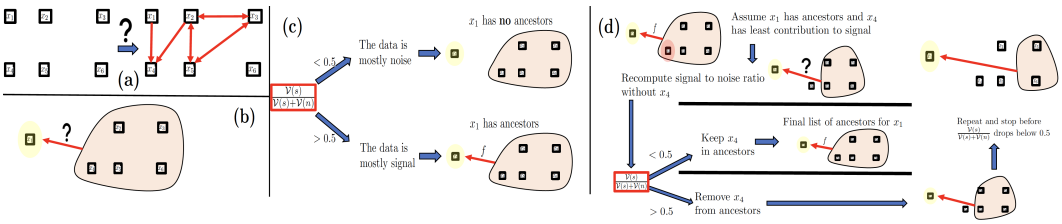


Figure 5.2: Ancestors identification in Type 3 problem.

5.2 Overview of the proposed approach for Type 3 problems.

We first present an algorithmic overview of the proposed GP-based approach for Type 3 problems. For ease of presentation, we consider the simple setting of Fig. 5.2.(a) where we are given N samples on the variables $x_1, \dots, x_6$. After measurements/collection, these variables are normalized to have zero mean and unit variance. Our objective is to uncover the underlying dependencies between them.

5.2.1 A signal-to-noise ratio to decide whether or not a node has ancestors.

Our algorithm's core concept is the identification of ancestors for each node in the graph. Let's explore this idea in the context of a specific node, say x_1, as depicted in Fig. 5.2(b). Determining whether x_1 has ancestors is akin to asking if x_1 can be expressed as a function of $x_2, x_3, \dots, x_6$. In other words, can we find a function f (living in a pre-specified space of functions that could be of controlled regularity) such that:

$$x_1 \approx f(x_2, \dots, x_6)\,? \tag{5.1}$$

To answer this question we regress f with a centered GP $\xi \sim \mathcal{N}(0, \Gamma)$ whose covariance function Γ is an additive kernel of the form $\Gamma = K_s + \gamma\delta(x - y)$, where K_s is a smoothing kernel, $\gamma > 0$ is a regularization parameter and $\delta(x - y)$ is the white noise covariance operator. This is equivalent to assuming the GP ξ to be the sum of two independent GPs, i.e., $\xi = \xi_s + \xi_n$ where $\xi_s \sim \mathcal{N}(0, K_s)$ is a smoothing/signal GP and $\xi_n \sim \mathcal{N}\big(0, \gamma\delta(x - y)\big)$ is a noise GP. Writing $\mathcal{H}_{K_s}$ for the Reproducing Kernel Hilbert Space (RKHS) induced by the kernel K_s, this is also equivalent to approximating f with a minimizer of

$$\inf_{f \in \mathcal{H}_{K_s}} \|f\|^2_{K_s} + \frac{1}{\gamma}\big\|f(X) - Y\big\|^2_{\mathbb{R}^N}\,, \tag{5.2}$$

where $\|\cdot\|^2_{\mathbb{R}^N}$ is the Euclidean norm on $\mathbb{R}^N$, X is the input data on f obtained as an $N \times 5$-matrix whose rows X_i are the samples on $x_2, \dots, x_6$, Y is the output data on f obtained as an N-vector whose entries are obtained from the samples on x_1, and $f(X)$ is a N-vector whose entries are the evaluations $f(X_i)$. At the minimum

$$\mathcal{V}(s) := \|f\|^2_{K_s} \tag{5.3}$$

quantifies the data variance explained by the signal GP ξ_s and

$$\mathcal{V}(n) := \frac{1}{\gamma}\big\|f(X) - Y\big\|^2_{\mathbb{R}^N} \tag{5.4}$$

quantifies the data variance explained by the noise GP ξ_n [243]. This allows us to define the signal-to-noise ratio

$$\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)} \in [0,1]\,. \tag{5.5}$$

If $\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)} < 0.5$[1], then, as illustrated in Fig. 5.2.(c), we deduce that x_1 has no ancestors, i.e., x_1 cannot be approximated as function of $x_2, \ldots, x_6$. Conversely if $\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)} > 0.5$, then, we deduce that x_1 has ancestors, i.e., x_1 can be approximated as function of $x_2, \ldots, x_6$.

5.2.2 Selecting the signal kernel K_s

This process is repeated by selecting the kernel K_s to be linear ($K_s(x,x') = 1 + \beta_1 \sum_i x_i x_i'$), quadratic ($K_s(x,x') = 1+\beta_1 \sum_i x_i x_i' + \beta_2 \sum_{i\leq j} x_i x_j x_i' x_j'$) or fully nonlinear to identify f as linear, quadratic, or nonlinear. In the case of a nonlinear kernel, we employ:

$$K_s(x,x') = 1 + \beta_1 \sum_i x_i x_i' + \beta_2 \sum_{i\leq j} x_i x_j x_i' x_j' + \beta_3 \prod_i (1 + k(x_i, x_i')) \tag{5.6}$$

where k is a universal kernel, such as a Gaussian or a Matérn kernel, with all parameters set to 1, and β_i assigned the default value 0.1. We select K_s as the first kernel that surpasses a signal-to-noise ratio of 0.5. If no kernel reaches this threshold, we conclude that x_1 lacks ancestors.

5.2.3 Pruning ancestors based on signal-to-noise ratio.

Once we establish that x_1 has ancestors, the next step is to prune its set of ancestors iteratively. We remove nodes with the least contribution to the signal-to-noise ratio and stop before that ratio drops below 0.5 as illustrated in Fig. 5.2.(d). To describe this, assume that K_s is as in (5.6). Then K_s is an additive kernel that can be decomposed into two parts:

$$K_s = K_1 + K_2\,, \tag{5.7}$$

where $K_1 = 1+\beta_1 \sum_{i\neq 1,2} x_i x_i' + \beta_2 \sum_{i\leq j, i,j\neq 1,2} x_i x_j x_i' x_j' + \beta_3 \prod_{i\neq 1,2}(1+k(x_i,x_i'))$ does not depend on x_2 and $K_2 = K_s - K_1$ depends on x_2. This decomposition allows us to express f as the sum of two components:

$$f = f_1 + f_2\,, \tag{5.8}$$

[1] We will later present a version with a more sophisticated method for pruning, but we keep the 0.5 threshold in this example for simplicity.

where f_1 does not depend on x_2, f_2 depends on x_2, and

$$(f_1, f_2) = \text{argmin}_{(g_1,g_2)\in\mathcal{H}_{K_1}\times\mathcal{H}_{K_2} \text{ s.t. } g_1+g_2=f} \|g_1\|^2_{K_1} + \|g_2\|^2_{K_2} .$$

Furthermore,

$$\|f\|^2_{K_s} = \|f_1\|^2_{K_1} + \|f_2\|^2_{K_2} ,$$

and $\frac{\|f_2\|^2_{K_1}}{\|f\|^2_{K_s}} \in [0, 1]$ quantifies the contribution of x_2 to the signal data variance. Following the procedure illustrated in Fig. 5.2.(d), if, for example, x_4 is found to have the least contribution to the signal data variance, we recompute the signal-to-noise ratio without x_4 in the set of ancestors for x_1. If that ratio is below 0.5, we do not remove x_4 from the list of ancestors, and x_2, x_3, x_4, x_5, x_6 is the final set of ancestors of x_1. If this ratio remains above 0.5, we proceed with the removal. This iterative process continues, and we stop before the signal-to-noise ratio drops below 0.5 to identify the final list of ancestors of x_1. The most efficient version of our proposed algorithm does not use a threshold of 0.5 on the signal-to-noise ratio to prune ancestors, but it rather employs an inflection point in the noise-to-signal ratio $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$ as a function of the number q of ancestors (Fig. 5.3.(d)). To put it simply, after ordering the ancestors in decreasing contribution to the signal, the final number q of ancestors is determined as the maximizer of $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q+1) - \frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$.

5.2.4 Computational complexity

We will now present a detailed analysis of the computational demands of the proposed method as a function of the number of variables, denoted as d, and the number of samples, N, pertaining to these variables. In the worst case, the proposed approach necessitates, for each of the d variables: for $i = 1, \ldots, d-1$, regressing a function mapping $d - i$ variables to the variable of interest and performing a mode decomposition, as exemplified in (5.8), to identify the variable with the minimal contribution to the signal. Since these two steps have the same cost, it follows that, in the worst case, the total computational complexity of the proposed method is $O(\mathbf{d^2N^3})$ which corresponds to product of the number of double-looping operations, d^2, and the cost of kernel regression from N samples which, without acceleration, is N^3 (i.e., the cost of inverting a $N \times N$ dense kernel matrix). However, if kernel scalability techniques are utilized, such as when the kernel has a rank k (for example, $k = d$ if the kernel is linear) or is approximated by a kernel of rank k (e.g., via a random feature map), then this worst-case bound can be reduced to $O(\mathbf{d^2Nk^2})$ by reducing the complexity of each regression step from $O(N^3)$ to $O(Nk^2)$. Note

that the statistical accuracy of the proposed approach requires that $N > d$ if the dependence of the unknown functions on their inputs is not sparse. Moreover, in the absence of kernel scalability techniques, the worst-case memory footprint of the method is $O(N^2)$ due to the necessity of handling dense kernel matrices. However, once the functional ancestors of each variable are determined, these matrices can be discarded. Consequently, only one such matrix needs to be retained in memory at any given time.

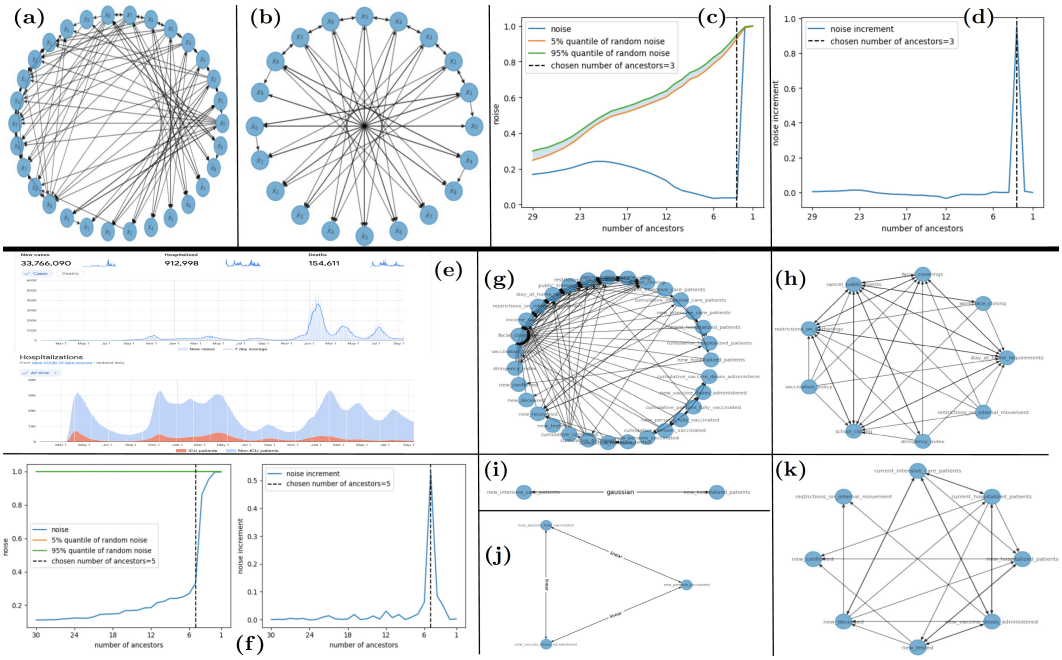


Figure 5.3: (a-d) The Fermi-Pasta-Ulam-Tsingou system. (e-k) The Google Covid 19 open data.

5.3 Results

The following examples and experiments illustrate the proposed approach.

5.3.1 The Fermi-Pasta-Ulam-Tsingou system

The Fermi-Pasta-Ulam-Tsingou (FPUT) system [246] is a prototypical chaotic dynamical system. It is composed of M masses indexed by $j \in \{0, \ldots, M-1\}$ with equilibrium position jh with $h = 1/M$. Each mass is tethered to its two adjacent masses by a nonlinear spring, and the displacement of the mass x_j adheres to the equation:

$$\ddot{x}_j = \frac{c^2}{h^2}(x_{j+1} + x_{j-1} - 2x_j)\left(1 + \alpha(x_{j+1} - x_{j-1})\right), \tag{5.9}$$

where $\alpha(x) = x^2$, $c = 1$ and $M = 10$. We use fixed boundary conditions by adding two more masses, with $x_{-1} = x_M = 0$. We take a total of 1000 snapshots from multiple trajectories and the observed variables are the positions, velocities, and accelerations of all the underlying masses. In the graph discovery phase, every other node is initially deemed a potential ancestor for a specified node of interest. We then proceed to iteratively remove the node with the least signal contribution. The step resulting in the largest surge in the noise-to-signal ratio is inferred as one eliminating a crucial ancestor, thereby pinpointing the final ancestor set. Fig. 5.3.(c) shows a plot of the noise-to-signal ratio $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$ as a function of the number q of proposed ancestors for the variable $\ddot{x}_7$ and with Z-test quantiles (in the absence of signal, the noise-to-signal ratio should fall within the shaded area with probability 0.9). Removing a node essential to the equation of interest causes the noise-to-signal ratio to markedly jump from approximately 25% to 99%. Fig. 5.3.(d) shows a plot of the noise-to-signal ratio increments $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q) - \frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q-1)$ as a function of the number q of ancestors for the variable $\ddot{x}_7$. Note that the increase in the noise-to-signal ratio is significantly higher compared to previous removals when an essential node was removed. Therefore, while solely relying on a fixed threshold to decide when to cease the removals might prove challenging, evaluating the increments in noise-to-signal ratios offers a clear guideline for efficiently and reliably pruning ancestors. The recovered full graph, depicted in Fig. 5.3.(a), is remarkably accurate despite the nonlinear nature of the model and the fact that our prior only encodes that the nonlinearity is smooth. Therefore, our algorithm does not require a dictionary or extensive knowledge of the structure of the unknown functions. Notably, velocity variables are accurately identified as non-essential and omitted from the ancestors of position and acceleration variables. Fig. 5.3.(b), which omits velocity variables for clarity, further elucidates the accurate recovery of dependencies. The dependencies are the simplest and clearest possible. They match exactly those of the original equations except for the boundary particles for which we recover valid equivalent equations.

5.3.2 The Google Covid 19 open data.

Consider the COVID-19 data from Google[2]. We focus on a single country, France, to ensure consistency in the data and avoid considering cross-border variations that are not directly reflected in the data. We select 31 variables that describe the state of the country during the pandemic, spanning over 500 data points, with each data

[2]The dataset can be accessed here

point corresponding to a single day. These variables are categorized as the following datasets: (1) Epidemiology dataset: Includes quantities such as new infections, cumulative deaths, etc. (2) Hospital dataset: Provides information on the number of admitted patients, patients in intensive care, etc. (3) Vaccine dataset: Indicates the number of vaccinated individuals, etc. (4) Policy dataset: Consists of indicators related to government responses, such as school closures or lockdown measures, etc. Some of these variables are illustrated in Fig. 5.3.(e). The problem is then to analyze this data and identify possible hidden functional relations between these variables. Fig. 5.3.(f) shows the noise-to-signal ratio (and its increments) as function of the number of ancestors of the "cumulative number of hospitalized patients" variable. Even for this real dataset, the proposed approach gives a clear signal for stopping the pruning process. Fig. 5.3.(g) shows the full recovered graph, which is highly clustered. Fig. 5.3.(h) shows the cluster corresponding to the variable "schools closing" revealing that the government either implemented multiple restrictive measures simultaneously or lifted them in unison (except for mask mandates that were on the verge of being identified as noise). The vaccination cluster (Fig. 5.3.(j)) reveals a linear relationship between variables (signaling redundant information) and the hospitalization cluster (Fig. 5.3.(i)) reveals a nonlinear one. Eliminating redundant nodes leads to the sparse graph shown in Fig. 5.3.(k), which is interpretable and amenable to (both quantitative and qualitative) analysis,

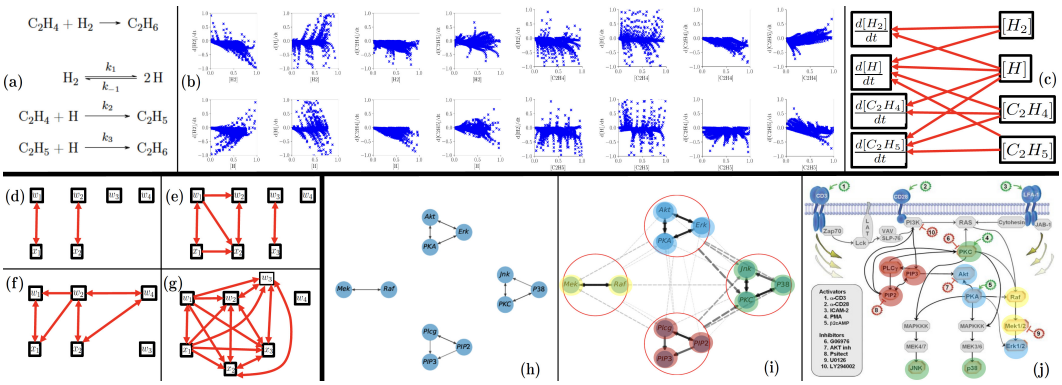


Figure 5.4: (a-c) Chemical reaction network. (d-g) Algebraic equations. (h-j) Cell signaling network.

5.3.3 Chemical reaction network.

In this example, we consider the recovery of a chemical reaction network from concentration snapshots. The reaction network, illustrated in Fig. 5.4.(a) is that of the hydrogenation of ethylene (C_2H_4) into ethane (C_2H_6). The problem is that of recovering the underlying chemical reaction network from snapshots (illustrated

in Fig. 5.4.(b)) of concentrations $[H_2]$, $[H]$, $[C_2H_4]$ and $[C_2H_5]$ and their time derivatives. $\frac{d[H_2]}{dt}$, $\frac{d[H]}{dt}$, $\frac{d[C_2H_4]}{dt}$ and $\frac{d[C_2H_5]}{dt}$. The proposed approach leads to a perfect recovery of the computational graph (shown in Fig. 5.4.(c)) and a correct identification of quadratic functional dependencies between variables.

5.3.4 Algebraic equations.

Fig. 5.4.(a-d) illustrate the application of the proposed approach to the recovery of functional dependencies from data satisfying hidden algebraic equations. In all these examples, we have $d = 6$ or $d = 7$ variables and $N = 1000$ samples from those variables. For $d = 6$ the variables are $w_1, w_2, w_3, w_4, x_1, x_2$. For $d = 7$ the variables are $w_1, w_2, w_3, w_4, x_1, x_2, x_3$. The samples from the variables w_1 to w_4 are i.i.d. $\mathcal{N}(0, 1)$ random variables, and the samples from x_1, x_2 (and x_3 for $d = 7$) are functionally dependent on the other variables. In the first example, $d = 6$ and the samples from x_1 and x_2 satisfy the equations $x_1 = w_1$ and $x_2 = w_2$. The algorithm selects the linear kernel and Fig. 5.4.(a) shows the recovered graph (which is exact). In the second example, $d = 7$ and the samples from x_1, x_2 and x_3 satisfy the equations $x_1 = w_1$, $x_2 = x_1^2 + 1 + 0.1w_2$, and $x_3 = w_3$. The algorithm selects the quadratic kernel and Fig. 5.4.(b) shows the recovered graph (which is exact). Even though x_2 can trace back its origin to either x_1 and w_2 or w_1 and w_2, the algorithm recognizes x_1, w_1, and w_2 as its ancestors underscoring the importance of eliminating redundant variables when aiming at deriving the sparsest graph. In the third example, $d = 6$ and the samples from x_1 and x_2 satisfy the equations $x_1 = w_1 w_2$ and $x_2 = w_2 \sin(w_4)$. The algorithm selects the nonlinear kernel and Fig. 5.4.(c) shows the recovered graph (which is exact). In the fourth example, $d = 7$ and the samples from x_1, x_2 and x_3 satisfy the equations $x_1 = w_1$, $x_2 = x_1^3 + 1 + 0.1w_2$ and $x_3 = (x_1 + 2)^3 + 0.1w_3$. Although these equations appear to be cubic, the algorithm correctly selects the quadratic kernel and makes an exact recovery of the graph shown in Fig. 5.4. (d) revealing hidden quadratic dependencies between variables.

5.3.5 Cell signaling network

Next, we apply the proposed framework to the example illustrated in Fig. 5.1.(l) from [285] and discover a hierarchy of functional dependencies in biological cellular signaling networks. We use single-cell data consisting of the $d = 11$ phosphoproteins and phospholipids levels in the human immune system T-cells that were measured using flow cytometry. This dataset was studied from a probabilistic modeling perspective in previous works. While [285] learned a directed acyclic graph to encode

causal dependencies, [107] learned an undirected graph of conditional independencies between the d molecule levels by assuming the underlying data follows a multivariate Gaussian distribution. The latter analysis encodes acyclic dependencies but does not identify directions. In this work, we aim to identify the functional dependencies without imposing strong distributional assumptions on the data. We simply use $N = 2,000$ samples chosen uniformly at random from the dataset consisting of 11 proteins and 7446 samples of their expressions. We apply the algorithm in two stages. The first stage of the algorithm uses only linear and quadratic kernels and recovers the graph shown in Fig. 5.4.(h). It consists of four disconnected clusters where the molecule levels in each cluster are closely related by linear or quadratic dependencies (all connections are linear except for the connection between Akt and PKA, which is quadratic). These edges match a subset of the edges found in the gold standard model identified in [285]. With perfect noiseless dependencies, one can define constraints that reduce the total number of variables in the system. Second, we learn the connections between groups of variables within each cluster with nonlinear kernels and obtain the graph shown in Fig. 5.4.(i) in which solid arrows indicate strong intra-cluster connections identified in the first level, and dashed lines indicate weaker connections between nodes and clusters identified in the second level. The width and grayscale intensities of each edge correspond to its signal-to-noise ratio. We emphasize that while causal graph recovery methods rely on the control of the sampling of the underlying variables (i.e., the simultaneous measurement of multiple phosphorylated protein and phospholipid components in thousands of individual primary human immune system cells, and perturbing these cells with molecular interventions), the reconstruction obtained by our method did not use this information and recovered functional dependencies rather than causal dependencies. Interestingly, the information recovered through our method appears to complement and enhance the findings presented in [285] (e.g., the linear and noiseless dependencies between variables in the JNK cluster is not something that could easily be inferred from the graph produced in [285] shown in Fig. 5.1.(j) where we have colored the clusters for comparison).

Comparisons. Using the expected graph reported in [285] as the ground truth (acknowledging that it may not be entirely accurate), we compare the edges our approach incrementally added to the true graph. Figure 5.5.(a) reports the number of additional edges that have been added and are not present in the ground truth (false positives) and edges removed that are present in the ground truth graph (false negatives). The added edges are based on the two-stage procedure described

above, where we first add the ten intra-cluster connections, followed by inter-cluster connections. Edges are added in decreasing order of signal-to-noise ratio, starting with the strongest. In the reported results, we do not account for the recovery of the direction of ground-truth edges. We note that, up to direction, all intra-cluster connections, along with the inter-cluster connections with the strongest signals are found in the ground truth graph, leading to the initial decrease in false negatives with only one false positive edge (the linear connection P38 $\longrightarrow$ Jnk that is not reported in the true graph). With the addition of the remaining (possibly non-spurious) edges, the number of false negatives drops to one, having recovered all edges, except for the one between PKC and Raf, which is identified to be statistically non-informative in our approach.

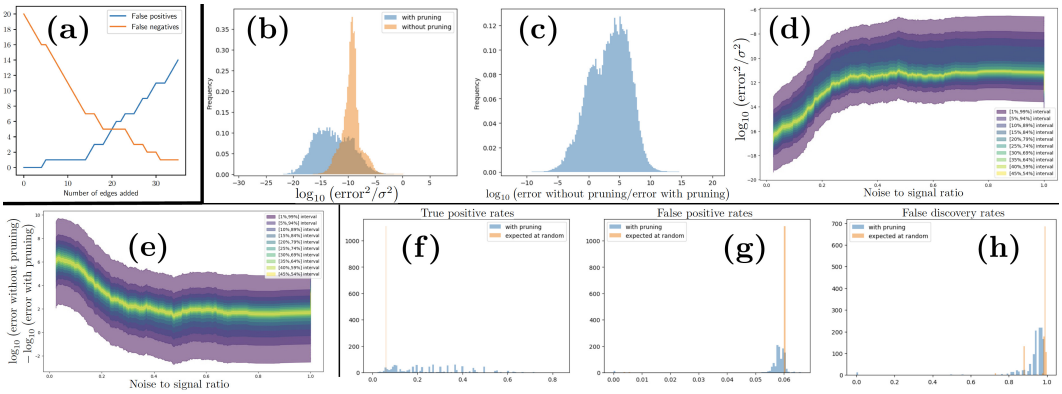


Figure 5.5: (a) Cell signaling network comparisons. (b-h) The BCR reaction benchmark.

5.3.6 A large-scale chemical reaction network: the BCR reaction benchmark

Lastly, we stress-test the scalability of our approach by applying it to a large-scale chemical reaction network: the BCR reaction benchmark from [197], which encompasses 1122 species. The dataset comprises 2400 snapshots of species concentrations and their corresponding time derivatives. We leveraged JAX's inherent parallelization capabilities [42] to accelerate our computations, allowing for the simultaneous pruning of multiple nodes while abstracting the complexity of parallel execution. While the scaling with respect to the number of data points is straightforward, scaling with the number of variables introduces a trade-off between computational speed and memory footprint. Specifically, the process of identifying the ancestors of various nodes can be expedited by storing a large array for all nodes. Using a DGX workstation equipped with four Nvidia V100 GPUs, each with 32GB of memory, pruning 190 nodes took approximately three days, projecting a total

experiment duration of around one month. Nonetheless, we can mitigate this computational burden by optimizing the computation of terms of the form $y^T K y$ for the specific quadratic kernel identified for this example. We include the details of such optimization in the supplementary material. By implementing this optimization, the duration of the entire experiment was reduced to just one hour.

In the first experiment, we simulated five trajectories of the associated system of ODEs, recording 1000 snapshots per trajectory. Out of these 5000 snapshots, 2400 were randomly selected as training data, and 2600 as testing data. Writing TP, TN, FP and FN for True/False Positives/Negatives and using the metrics True Positive Rate (TPR=TP/(TP+FN)), False Positive Rate (FPR=FP/(FP+TN)), and False Discovery Rate (FDR=FP/(TP+FP)), we observed a TPR of 39.9%, an FPR of 16.4%, and an FDR of 97.2% (indicating that 97.2% of predicted positives are false). This high FDR can be attributed to the limited exploration of the full variable range—1122 in total—by the five trajectories. The trajectories explored a subset of the possible space (near a limit cycle attractor), which led to the recovery of functional dependencies that represent both the chemical reactions and the specific subspace visited. Furthermore, with 1122 variables, the 630,003 coefficients of the underlying quadratic equations are vastly under-determined with only 2400 data points. Despite the high FDR in the recovered graph, as illustrated in Fig. 5.5.(b-c), the CHD pruning process vastly improves the accuracy (by orders of magnitude) of the estimated functions on the 2600 unseen snapshots by reducing the dimension of the regression problem whenever possible. We denote y_i as an observed data point, σ^2 as the variance of the observed data, $\hat{y}_i$ for a predicted data point without pruning, and $\bar{y}_i$ for a predicted data point post-pruning. Fig. 5.5.(b) illustrates the histogram of the log-normalized squared errors before and after pruning, expressed as $\log_{10}\left(|y_i - \hat{y}_i|^2/\sigma^2\right)$ and $\log_{10}\left(|y_i - \bar{y}_i|^2/\sigma^2\right)$. The 99th percentile of the normalized squared error is less than 10^{-2} for all species. Fig. 5.5.(c) displays the histogram of the log-normalized squared error improvements due to pruning, calculated as $\log_{10}\left(|y_i - \hat{y}_i|^2/|y_i - \bar{y}_i|^2\right)$. Fig. 5.5.(d-e) display the quantiles of the histograms post-pruning, conditioned on the noise-to-signal ratio observed at the final pruning step. These plots reveal a clear trend: a higher noise-to-signal ratio at the time of pruning correlates with increased error and diminished improvements in accuracy.

In a second experiment, we formed the data by randomly sampling concentrations uniformly in $[0, 1]$ (independently across species and snapshots) and recorded the resulting time derivatives. While this sampling increased the variability of the 2400

snapshots, the model remained vastly underdetermined. The noise-to-signal and bootstrapped (Z-test) ratios remained close to 0.5, suggesting insufficient data for statistically significant variable importance assessments. Nonetheless, as depicted in Fig. 5.5.(f-h), significant insights can still be gleaned from the activations, showing notable improvements when comparing the histograms of the values of TPR, FPR, and FDR obtained with pruning based on these ratios and pruning at random. This analysis reveals that even with high dimensionality and scarce data, between 10% and 80% of the true ancestors can still be accurately identified.

5.4 Discussions

5.4.1 Limitations

In its present form, the proposed approach is limited by several factors. (1) Without access to the sampling of the data, the direction of some edges may not be identifiable. For instance the functional relationship $x - 2y = 0$ can be represented as both $y = 2x$ ($x \to y$) and $x = y/2$ ($y \to x$). (2) It assumes an additive noise W on the functional relationship $y = f(x) + W$ between the variables x and y. In a fully probabilistic setting, this structure may be non-additive, i.e., of the form $y = f(x, W)$, which implies discovering a general transition kernel, i.e., a non-Gaussian generative model. Although our method achieves polynomial complexity, in settings where one has access to the distribution of the data, the price to pay, when compared with information-theoretic methods, is a reduction in generality imposed by the stronger assumption made on the data-generating process. Furthermore, the price to pay for the weaker data requirements (i.e., the absence of interventional data) is that our method recovers functional relationships rather than causal ones or conditional dependencies. (3) If the (noisy) functional relationship $y = f(x) + W$ is associated with a non-regular (e.g., discontinuous) function f then the kernels discussed above (linear, quadratic and fully nonlinear) will be misspecified and may lead to false negatives. The kernel selection and hyperparameter tuning problems in misspecified settings require further work. (4) As demonstrated in the BCR reaction application, while the method scales well computationally with an increase in the number of variables, it may still be impacted by the curse of dimensionality. This occurs particularly if the dataset only covers a limited subset of the full range of variable values. Given the results displayed in Fig. 5.5.(b-h) we suspect that this impact could be mitigated by adopting more advanced strategies in place of our current top-down pruning method. Such strategies could involve grouping variables and integrating both top-down and bottom-up iterative approaches.

5.4.2 Conclusions

We have developed a comprehensive Gaussian Process framework for solving Type 3 (hypergraph discovery) problems, which is interpretable and amenable to analysis. The breadth and complexity of Type 3 problems significantly surpass those encountered in Type 2 (hypergraph completion), and the initial numerical examples we present serve as a motivation for the scope of Type 3 problems and the broader applications made possible by this approach. Our proposed algorithm is designed to be fully autonomous, yet it offers the flexibility for manual adjustments to refine the graph's structure recovery. We emphasize that our proposed approach is not intended to supplant causal inference methods [256]; see Methods for a complete overview. Instead, it aims to incorporate a distinct kind of information into the graph's structure, namely, the functional dependencies among variables rather than their causal relationships. Additionally, our method eliminates the need for a predetermined ordering of variables, a common requirement in acyclic probabilistic models where determining an optimal order is an NP-hard problem usually tackled using heuristic approaches. Furthermore, our approach can actually be utilized to generate such an ordering by quantifying the strength of the connections it recovers. The Uncertainty Quantification properties of the underlying Gaussian Processes are integral to the method and could also be employed to quantify uncertainties in the structure of the recovered graph. We also observe that forming clusters from highly interdependent variables helps to obtain a sparser graph. Additionally, the precision of the pruning process is enhanced by avoiding the division of node activation within the cluster among its separate constituents. We employed this strategy in the recovery of the gene expression graph in Fig. 5.4.(i). Given the polynomial complexity of our method, promising avenues for future work include applications to large datasets in genomics and in systems biology, particularly in the reconstruction and intervention of metabolic pathways. These applications benefit from the ability to handle large-scale datasets efficiently, enabling the analysis of complex biological networks.

5.4.3 Data availability

The data in the paper and the Supplementary Information are available in the Github repository of the paper.

5.4.4 Code availability

The code for the algorithm and its application to various examples are available for download (and as an installable python library/package) in the Github repository of the paper.

5.4.5 Online content

Supplementary Information is available for this paper.

5.4.6 Acknowledgements

HO, TB, PB, XY, and RB acknowledge support from the Air Force Office of Scientific Research under MURI award number FA9550-20-1-0358 (Machine Learning and Physics-Based Modeling and Simulation). Additionally, HO, TB, and PB acknowledge support by the Department of Energy under award number DE-SC0023163 (SEA-CROGS: Scalable, Efficient and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems) and by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. HO and PB further acknowledge support by Beyond Limits (Learning Optimal Models) through CAST (The Caltech Center for Autonomous Systems and Technologies). TB acknowledges support from the Kortschak Scholar Fellowship Program. NR acknowledges support from the JPL Researchers on Campus (JROC) program. HO is grateful for the Department of Defense Vannevar Bush Faculty Fellowship. We are grateful to two referees for their insightful comments and valuable suggestions.

Chapter 6

GRAPHICAL CONDITIONAL GENERATIVE MODELING FOR DIGITAL TWIN MODELING

Zongren Zou, Théo Bourdais, Ricardo Baptista, and Houman Owhadi. Graphical conditional generative modeling for digital twin modeling. *arXiv preprint*, 2026. URL https://arxiv.org/abs/2606.16219.
TB co-developed the structure-discovery and conditional generative framework and co-wrote the manuscript.

6.1 Introduction

Understanding how variables influence one another is central to the analysis and modeling of complex systems. Across the physical, engineering, biological, economic, and geopolitical sciences, such relationships form structured networks of dependence: climate variables interact through coupled atmospheric and oceanic processes, chemical species through reaction networks, stochastic dynamical systems through state-dependent transition laws, and financial markets through chains of shocks transmitted across energy, logistics, production, and prices. Recent geopolitical disruptions provide a vivid example: war-related shocks can propagate through energy markets, shipping and supply chains, fertilizer production, agricultural costs, and ultimately affect food prices. A wide range of methods have been developed for this purpose. They perform structure discovery by identifying, for each variable, a reduced set of other system variables that are relevant for explaining or predicting its behavior. These approaches differ substantially in their assumptions, objectives, and statistical objects of interest, yet most implicitly adopt a functional perspective. A candidate variable is considered influential for a target if variation in that variable changes an estimated response function, governing equation, sensitivity index, or conditional expectation. Previous work includes variable selection techniques in regression (e.g.,[9, 85, 127, 189, 212]), global and local sensitivity analysis (e.g., Sobol indices for global sensitivity [175, 235], and analysis of variance (ANOVA) [205, 322, 345]), sparse identification of governing equations [47], graphical model learning [90, 377, 378], and causal inference frameworks [108, 222, 257].

When structure discovery is driven by prediction under squared-error loss, the fitted

dependence is primarily a dependence of the conditional mean. Consequently, variables that leave this conditional mean nearly unchanged may appear irrelevant, even if they strongly affect the conditional law of the target. In many stochastic systems, influence is distributional rather than mean-based: changes in candidate inputs may alter the conditional variance, higher-order moments, multimodality, tail probabilities, or the likelihood of extreme events without producing appreciable changes in the conditional mean. For example, these changes may arise in physical systems near bifurcation points where fluctuations amplify without large mean shifts [296]; in multiscale or coarse-grained systems where unresolved variables induce state-dependent stochasticity [64, 120], a setting arising naturally in data-driven stochastic differential equation (SDE) inference (e.g., [110]); and in biological, social, and materials systems where rare events and failure probabilities dominate system behavior [96, 260]. In finance and economics, asset returns often exhibit weak or near-zero mean dependence, while their volatility and downside risk depend strongly on market stress indicators, interest rates, or geopolitical risk [39, 93]. Figure 6.1(a) illustrates this phenomenon with daily gold and silver returns: while any mean dependence between the two is weak, the data suggest that the conditional distribution may vary substantially, motivating the question of whether distributional changes exist beyond what mean-based analysis can detect. Figure 6.1(b) shows an even more pronounced example for data $y = x\omega$ with $\omega \sim \mathcal{N}(0, 1)$. Here, $\mathbb{E}[y \mid x] = 0$, but x determines the conditional distribution of y. This makes it a dependency that remains undetectable to methods based only on functional or mean relationships.

Scientific interpretability, Transferable mechanisms, Robust finite-data learning, Data-efficient modeling.

Interpretability is the primary scientific output; transferability, data efficiency, modularity, and robustness are downstream benefits of that interpretation. The interpretability gives at least two kinds of value:

1. **Operational value for digital twins and control**: surrogate reuse, modularity, data efficiency, robustness.

2. **Scientific or analytical value for data analysis**: understanding which variables matter, identifying regime-dependent dependence patterns, detecting structural change over time.

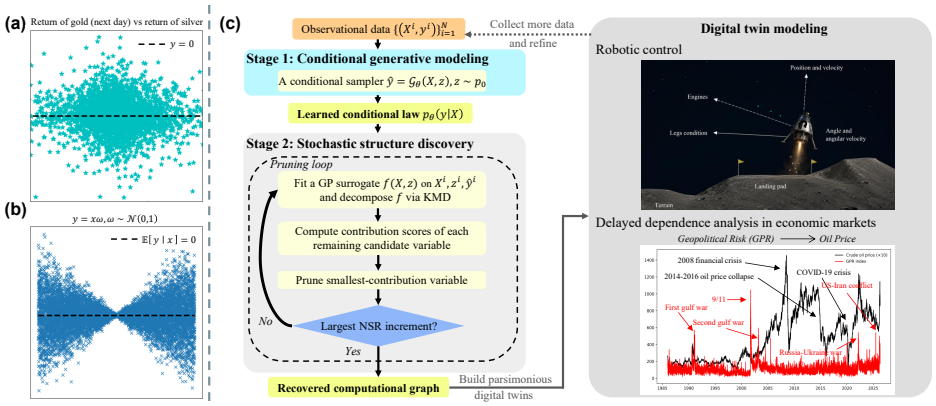


Figure 6.1: (a): Daily gold returns (next day) and silver returns (data from Google Finance) to illustrate stochastic relation in financial data. (b): Equation discovery with missing variables to illustrate the ancestor hidden in the distribution rather than the conditional mean. (c): A schematic of the proposed two-stage framework for digital twin modeling.

We generalize dependence discovery from functional modeling to conditional generative modeling. Rather than approximating y through a deterministic response function, or conditional mean, we learn a sampler for the conditional law of y given candidate covariates $x_1, \ldots, x_n$. A candidate variable is relevant if changing it affects the conditional law of y. To quantify relevance, we distill the learned conditional generative model into a Gaussian-process (GP) surrogate and apply kernel mode decomposition (KMD) [244] to obtain nonlinear ANOVA contribution scores. Iteratively pruning variables with small scores, as in [41], yields a parsimonious ancestor set while preserving the learned conditional law. The resulting procedure targets observational stochastic dependencies rather than causal mechanisms: it does not assume interventional data, controlled perturbations, or an acyclic causal structure [259, 319]. It also differs from conditional independence tests [124, 373] and information-theoretic methods [71, 298]: the output is a deployable conditional generative surrogate and its ancestor set, not a hypothesis test or significance score.

A central application is data-driven dynamics learning, where one seeks a surrogate transition law $p(x_{t+1} \mid x_t)$, or $p(x_{t+1} \mid x_t, a_t)$ in controlled systems. Such a surrogate can be viewed as a data-driven digital twin: a probabilistic, executable model for simulation, control, and decision-making [125, 272]. In this setting, structure discovery identifies the state, action, or memory variables needed to specify the transition law, yielding a lower-dimensional and more interpretable learned Markov decision process [266, 327]. In partially observed or coarse-grained systems, this includes selecting past states and actions needed to make the learned dynamics

effectively Markovian [64, 120]. The downstream control methods considered here, including model predictive control (MPC), model-free reinforcement learning (RL), and model-based RL, serve as evaluation instruments; the contribution is the discovered ancestor set defining a sufficiently small input space for the surrogate.

6.2 Results

6.2.1 Problem formulation

We consider a system of observed variables $s_1, \ldots, s_n$ sampled through observations

$$S^k = (s_1^k, \ldots, s_n^k), \qquad k = 1, \ldots, N,$$

and seek to recover a conditional dependency structure that describes their joint stochastic behavior. Specifically, for each target variable s_i, our goal is to identify a small set of parent variables

$$\mathcal{A}^\dagger(s_i) \subset \{s_1, \ldots, s_n\} \setminus \{s_i\}$$

that is sufficient to specify the conditional law of s_i. Equivalently, we represent each node by a stochastic map

$$s_i \overset{d}{=} T_i(\mathcal{A}^\dagger(s_i), \omega_i),$$

where $\omega_i \sim p_0$ is a reference random variable drawn independently of the conditioning variables $\mathcal{A}^\dagger(s_i)$, for instance from a standard Gaussian distribution, and T_i is an unknown conditional transport map that maps this reference noise to a sample from the conditional distribution of s_i given its parents. The associated conditional dependence is represented by a directed graph $G = (V, E)$ consisting of vertices and edges

$$V = \{s_1, \ldots, s_n\}, \qquad E = \{\, s_j \to s_i : \; s_j \in \mathcal{A}^\dagger(s_i) \,\}.$$

This formulation should be understood in a surrogate-sufficiency sense. Given finite observational data, we do not assume access to a unique minimal graph. Rather, we seek parent sets that are sufficiently informative to build accurate conditional generative surrogates. Formally, one may view the idealized objective as finding sparse parent sets for which the conditional law $P(s_i \mid \mathcal{A}^\dagger(s_i))$ can be accurately represented by the conditional sampling mechanism $T_i(\mathcal{A}^\dagger(s_i), \omega_i)$, for each $i = 1, \ldots, n$. In additive-noise regimes, where $T_i(\mathcal{A}^\dagger(s_i), \omega_i) = f_i(\mathcal{A}^\dagger(s_i)) + \eta_i(\omega_i)$, with input-independent noise, this reduces to classical functional-dependence discovery under squared-error prediction loss [41]. Since the parent sets and maps are found for each node, the graph recovery problem reduces to a sequence of single-target problems.

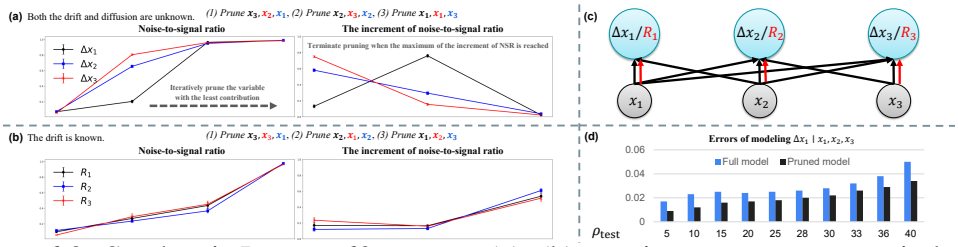


Figure 6.2: Stochastic Lorenz-63 system: (a)–(b) pruning process over ten independent runs, (c) recovered computational graph, and (d) generalization of the full and pruned models under parameter-induced distribution shift by training at $\rho = 28$ and testing at different values of ρ_{test}.

For a fixed target s_i, we write

$$y = s_i, \qquad (x_1, \ldots, x_{n-1}) = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n),$$

and ask which covariates x_j influence the conditional law of y. The local problem therefore has the general form

$$y \overset{d}{=} T(x_1, \ldots, x_{n-1}, \omega), \tag{6.1}$$

where $\omega \sim p_0$ is drawn independently of the covariates used to condition the sampler. Applying the proposed stochastic ancestor discovery procedure to each target variable s_i yields an estimated conditional dependency graph. When the identified conditioning set is minimal, it can be interpreted as a minimal Markov blanket of s_i [167, 255]. In settings with a known temporal or causal ordering, such as time-lagged dynamical systems, Markov blanket estimation directly translates to causal graph learning [340].

6.2.2 An illustrative example

The following examples illustrate the proposed approach across increasing levels of complexity. In each example, we specify the targets, candidate covariates, and validation criterion explicitly; notation is local to the example. When a reference computational graph is available, validation is by recovery of the relevant ancestor sets; otherwise, validation is by sufficiency of the discovered variables for the downstream task, such as prediction, control, or structural analysis. Data and code have been deposited in https://github.com/ZongrenZou/GraphicalConditionalGenerativeModeling. Additional details and examples can be found in Appendices E.3 and E.4.

6.2.3 Conditional-law modeling recovers stochastic graph structure for improved accuracy

This example serves as a verification test because the underlying computational graph is known. We consider the stochastic Lorenz-63 system with multiplicative fluctuation-dissipation noise [117, 200]:

$$dx_1 = \sigma(x_2 - x_1)dt + \alpha x_1 dW_1, \tag{6.2a}$$

$$dx_2 = \big(x_1(\rho - x_3) - x_2\big)dt + \alpha x_2 dW_2, \tag{6.2b}$$

$$dx_3 = (x_1 x_2 - \beta x_3)dt + \alpha x_3 dW_3, \tag{6.2c}$$

where x_i are the state variables and W_i are independent Wiener processes. We recover the dynamic dependencies of $q_{\mathcal{I}} = (x_1, x_2, x_3)$ by applying the proposed method given samples of state increments. We consider two cases. First, both drift and diffusion are treated as unknown. In this case, the targets are the full increments Δx_i, whose true ancestor sets $\mathcal{A}^\dagger(\Delta x_1) = \{x_1, x_2\}$, $\mathcal{A}^\dagger(\Delta x_2) = \{x_1, x_2, x_3\}$ and $\mathcal{A}^\dagger(\Delta x_3) = \{x_1, x_2, x_3\}$, are obtained by inspecting both the drift and diffusion terms. Second, when the drift is known, we apply the method to the residual stochastic increments $R_i := \Delta x_i - b_i(x_1, x_2, x_3)\Delta t$, where b_i is the ith drift component. Since the diffusion is diagonal, the true residual ancestor sets reduce to $\mathcal{A}^\dagger(R_1) = \{x_1\}$, $\mathcal{A}^\dagger(R_2) = \{x_2\}$, and $\mathcal{A}^\dagger(R_3) = \{x_3\}$. Figure 6.2 shows the recovered computational graph and the pruning process from ten independent runs. The method recovers the full ancestor sets when both drift and diffusion are unknown, and the reduced diagonal structure when only the drift is known.

We show that pruned model improves both (1) the accuracy of the generative modeling for in-distribution data and (2) the accuracy for out-of-distribution data. Specifically, we consider the conditional law of $\Delta x_1 | x_1, x_2, x_3$, and use the Wasserstein-1 distance as the error to evaluate the generative model. We use data from $\rho = 28$ for training, and consider ten test scenarios:

1. $\rho_{\text{test}} = 40$: 0.050 ± 0.066 (full model) vs 0.034 ± 0.046 (pruned model)
2. $\rho_{\text{test}} = 36$: 0.038 ± 0.049 (full model) vs 0.029 ± 0.037 (pruned model)
3. $\rho_{\text{test}} = 33$: 0.032 ± 0.036 (full model) vs 0.026 ± 0.030 (pruned model)
4. $\rho_{\text{test}} = 30$: 0.028 ± 0.028 (full model) vs 0.022 ± 0.023 (pruned model)
5. $\rho_{\text{test}} = 28$: 0.026 ± 0.023 (full model) vs 0.020 ± 0.017 (pruned model)

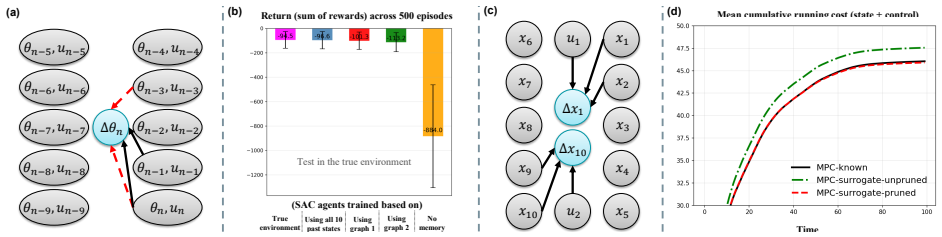


Figure 6.3: Control examples. (a)–(b) Missing-variable pendulum: memory discovery and policy transfer. (c)–(d) Stochastic heat equation: boundary discovery and MPC performance.

6. $\rho_{\text{test}} = 25$: 0.025 ± 0.021 (full model) vs 0.018 ± 0.015 (pruned model)

7. $\rho_{\text{test}} = 20$: 0.024 ± 0.020 (full model) vs 0.017 ± 0.012 (pruned model)

8. $\rho_{\text{test}} = 15$: 0.025 ± 0.020 (full model) vs 0.016 ± 0.011 (pruned model)

9. $\rho_{\text{test}} = 10$: 0.023 ± 0.015 (full model) vs 0.012 ± 0.007 (pruned model)

10. $\rho_{\text{test}} = 5$: 0.017 ± 0.007 (full model) vs 0.009 ± 0.005 (pruned model)

averaged from $10,000$ test data points and $1,000$ samples per data point.

6.2.4 Sparse state-control structure preserves downstream control performance.

We consider a PDE-based control problem where the discovered ancestor sets can be verified against the known finite-difference structure. The system is a one-dimensional stochastic heat equation with Neumann boundary control, given by $\partial_t T = \kappa \partial_{xx} T + \sigma T \xi$ on $x \in (0,1)$, with boundary conditions $-\kappa \partial_x T(0,t) = u_1(t)$ and $\kappa \partial_x T(1,t) = u_2(t)$. After spatial discretization using $m = 10$ nodes, we write $x_i(t_n) \approx T(r_i, t_n)$ for the discretized temperature at node r_i. Thus, the digital-twin state is $q(t_n) = (x_1(t_n), \ldots, x_{10}(t_n))$, with boundary controls $u(t_n) = (u_1(t_n), u_2(t_n))$. We focus on the boundary components $q_{\mathcal{I}} = (x_1, x_{10})$, whose dynamics are the controlled components replaced by learned surrogates inside MPC; the interior finite-difference updates are assumed to be known and retained. The finite-difference stencil gives

$$\Delta x_1 = -2rx_1 + 2rx_2 + cu_1 + x_1\omega_1, \qquad \Delta x_{10} = 2rx_9 - 2rx_{10} + cu_2 + x_{10}\omega_{10},$$

where r and c are constants determined by the discretization, and ω_i denotes the discretized multiplicative noise. Therefore, the true boundary ancestor sets are $\mathcal{A}(\Delta x_1) = \{x_1, x_2, u_1\}$ and $\mathcal{A}(\Delta x_{10}) = \{x_9, x_{10}, u_2\}$. Applying the proposed method

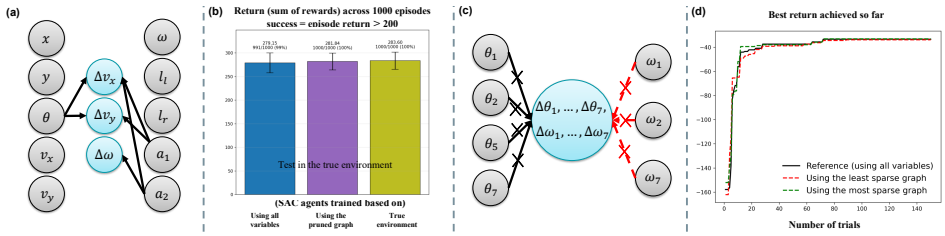


Figure 6.4: Robotics control examples and comparisons. (a)–(b) Lunar Lander: discovered thrust-stage dependence structure and policy-transfer performance. (c)–(d) 7-DOF reacher: discovered pruned variables and PETS performance.

to the full candidate set $\{x_1, \ldots, x_{10}, u_1, u_2\}$ recovers exactly these two ancestor sets, as shown in Figure 6.3(c). To evaluate the discovered structure, flow matching surrogates trained on the pruned ancestor sets are distilled into feedforward networks and embedded in an MPC controller, replacing only the boundary-node dynamics while the interior nodes retain the known discretization. MPC with the pruned surrogate achieves a cost comparable to MPC with the known dynamics, while the unpruned surrogate yields slightly higher cost; see Figure 6.3(d). The pruned surrogate also reduces the average MPC computation time from 31.0 to 24.7 seconds per simulation, because the lower-dimensional input reduces the cost of gradient evaluations inside the online optimization.

6.2.5 Discovering state and control dependence in the Lunar Lander control problem.

We apply the proposed method to the 2D Lunar Lander benchmark [334], a higher-dimensional control problem with stochastic dynamics. In the notation of Section 6.4.2, the state is $q = (x, y, \theta, v_x, v_y, \omega, l_l, l_r)$, consisting of position, angle, linear and angular velocities, and ground-contact indicators, while the control is $u = (a_1, a_2)$, the main engine throttle and lateral booster throttle. We focus on the stochastic thrust stage of the simulator. In this stage, the engines apply random impulses to the velocity variables (v_x, v_y, ω); the subsequent Newtonian integration updates positions, angles, and contact variables deterministically and is kept unchanged. Thus the targets are $q_{\mathcal{I}} = (v_x, v_y, \omega)$, or equivalently the thrust-stage increments Δv_x, Δv_y, and $\Delta\omega$, and the candidate covariates are the full state-control set $\{x, y, \theta, v_x, v_y, \omega, l_l, l_r, a_1, a_2\}$. The contact indicators l_l, l_r are included as candidates but are not expected to influence the instantaneous thrust impulse, since contact dynamics are handled in the Newtonian/contact stage rather than in the engine-response stage. The method recovers a sparse, physically interpretable struc-

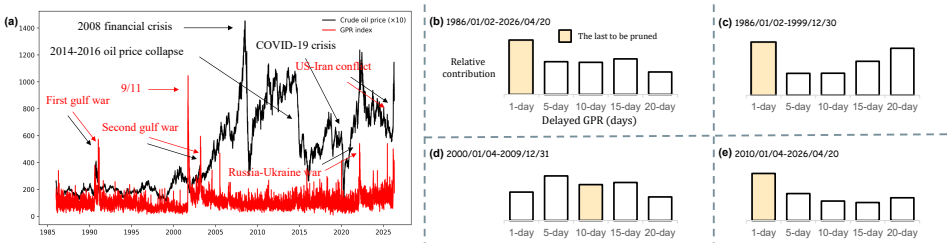


Figure 6.5: Delayed dependence analysis in economic markets. (a) Aligned WTI crude oil price and GPR index data. (b)–(e) First-iteration relative contribution scores of delayed GPR variables across time periods; the highlighted bar denotes the variable pruned last.

ture (Figure 6.4(a)):

$$\mathcal{A}(\Delta v_x) = \{\theta, a_1, a_2\}, \qquad \mathcal{A}(\Delta v_y) = \{\theta, a_1\}, \qquad \mathcal{A}(\Delta\omega) = \{a_2\}.$$

This is consistent with the physical design: the main engine primarily governs vertical acceleration, the lateral booster controls angular response, and the lander orientation θ geometrically couples thrust to lateral motion. To evaluate the discovered structure, flow matching surrogates trained on the pruned ancestor sets replace only the thrust stage in the simulator, while the Newtonian/contact stage remains unchanged. Following the same evaluation protocol as the previous pendulum example, a SAC agent is trained for this problem and transferred to the true environment without further training. The pruned surrogate achieves 1,000/1,000 successful landings, compared with 991/1,000 for the unpruned surrogate, and performs comparably to an agent trained in the true environment; see Figure 6.4(b).

6.2.6 Delayed distributional dependence changes across economic regimes

Lastly, we demonstrate the proposed method as an analytical tool for analyzing real economic data. Specifically, we study how geopolitical risk influences oil price dynamics and whether this dependence structure remains stable across different market regimes. We use the daily West Texas Intermediate (WTI) crude oil prices from the Federal Reserve Bank of St. Louis and the daily geopolitical risk (GPR) index [50], covering the period from 1986/01/02 to 2026/04/20 (Figure 6.5(a)). Let r_n denote the daily log-return of oil prices ($r_n = \log(\mathrm{Oil}_{n+1}) - \log(\mathrm{Oil}_n)$). We model the transition kernel

$$\begin{aligned} p\big(r_n \mid\ & r_{n-1}, r_{n-5}, r_{n-10}, r_{n-15}, r_{n-20}, \\ & \log \mathrm{GPR}_n, \log \mathrm{GPR}_{n-4}, \log \mathrm{GPR}_{n-9}, \log \mathrm{GPR}_{n-14}, \log \mathrm{GPR}_{n-19}\big), \end{aligned} \tag{6.3}$$

and apply the proposed method to the GPR lag variables to analyze how delayed geopolitical-risk information enters the oil-return transition kernel. The method

retains all five GPR lags as ancestors, so we further examine their relative contribution scores and pruning order. Figure 6.5(b)–(e) reports two diagnostics for the full 1986-2026 period and three sub-periods: the first-iteration relative contribution scores of the five GPR lags, and the lag pruned last by the iterative pruning procedure. Over the full period, the one-day lag has the largest first-iteration contribution and is also the last lag to be pruned. In contrast, the dominant surviving lag changes across sub-periods: the one-day lag is last pruned in 1986-1999 and 2010-2026, whereas the ten-day lag is last pruned in 2000-2009. These changes indicate that the delayed dependence structure between geopolitical risk and oil returns is not stationary across time. The 2000-2009 period coincides with major geopolitical and economic disruptions, including 9/11 and the 2008 financial crisis, which may have altered the observed dependency structure between geopolitical risk and oil returns. Eight five-year sub-periods are also analyzed with the proposed method, revealing a more detailed evolution of the transition-kernel structure; see Appendix E.3.7 for the results. These findings illustrate how the proposed method can be used to detect and localize structural changes in delayed distributional dependence from observational data.

6.3 Discussion

We introduced a framework for computational structure discovery in stochastic systems from observational data. The method characterizes conditional distributions rather than conditional means. By using conditional generative modeling with KMD of GP surrogates, we identify variables that affect distributional features such as variability, multimodality, or tail behavior. The examples validate the method in two ways. When reference dependence structure is available, the method recovers the relevant ancestor sets, including dependencies that are not identified by methods that only compute mean dependence. When the ground-truth dependence graph is not available, as in partially observed control, reinforcement learning, and economic data, the discovered structure is evaluated by surrogate or task sufficiency: models built on the discovered ancestors achieve performance comparable to full-variable models while using a smaller and more interpretable set of inputs. Some limitations remain: accurate conditional generative modeling is often data-intensive, leading to higher run-to-run variability with finite data and to the pruning of weak dependencies when they are not needed for surrogate fidelity (which can be misaligned with the other objective of discovering the graph). The noise-to-signal ratio stopping rule for variable pruning can also be ambiguous when no clear inflection point is present.

Lastly, it will be valuable to develop a comprehensive benchmark against methods for testing conditional independence or causal discovery, which target different statistical properties.

6.4 Methods

We now describe the single-target procedure used to estimate the parent set $\mathcal{A}^\dagger(y)$ in the formulation of Section 6.2. Given observational data $\{(X^i, y^i)\}_{i=1}^N$, with $X^i = (x_1^i, \ldots, x_n^i)$, the goal is to identify which components of X are needed to approximate the conditional law of $y \mid X$ with sufficient accuracy. In deterministic or additive-noise settings, this reduces to discovery of functional dependence and can be addressed directly by KMD or computational hypergraph discovery [41, 244]. In the present stochastic setting, the object of interest is not only a response function such as $\mathbb{E}[y \mid X]$, but the conditional distribution. Thus, we first learn a conditional sampler

$$y \mid X \overset{d}{\approx} \mathcal{G}_\theta(X, z), \qquad z \sim p_0, \tag{6.4}$$

depending on parameters θ where z is a reference random variable drawn from a simple base distribution p_0, and $\mathcal{G}_\theta$ maps noise to a sample from the conditional law of $y \mid X$.

In principle, any sufficiently expressive conditional generative model could be used in (6.4), including Gaussian approximations, diffusion models, or flow-based models. In this work, we use flow matching [190], because of its training stability, expressiveness and scalability; details are given in Appendix E.1. Most importantly, after the conditional law has been represented by the sampler $\mathcal{G}_\theta$, stochastic structure discovery becomes a functional dependence problem for the deterministic map $(X, z) \mapsto \mathcal{G}_\theta(X, z)$. We then distill this map into a GP surrogate so that KMD can be used to assign variable-wise contribution scores. Thus the two stages are:

1. **Conditional generative modeling.** Learn a conditional sampler $\mathcal{G}_\theta(X, z)$ for the law of $y \mid X$.

2. **Distillation and ancestor discovery.** Approximate $\mathcal{G}_\theta$ by a GP surrogate $f(X, z) \approx \mathcal{G}_\theta(X, z)$ and apply KMD [244], as in computational hypergraph discovery [41], to prune covariates that do not significantly contribute to the learned conditional law.

This procedure separates distribution approximation from interpretable structure discovery: neural conditional samplers are expressive and scalable, but do not

directly provide the variance decomposition needed for ancestor discovery; GP surrogates provide this decomposition, but are less suitable as primary conditional generative models for large datasets. A schematic is shown in Figure 6.1(c). The final ancestor set $\mathcal{A}(y)$ should be understood as an estimate of $\mathcal{A}^\dagger(y)$ at the resolution of the learned sampler, the GP surrogate, and the stopping criterion used to identify ancestors. In finite-data settings, the ancestors found may vary across random training or distillation runs; when needed, we report stable ancestors found across repeated runs or report conservative unions of ancestor sets. More details are provided in the corresponding examples.

6.4.1 Distillation and stochastic ancestor discovery

After training the conditional sampler $\mathcal{G}_\theta$, we generate synthetic samples from the learned conditional law. For inputs $X^i = (x_1^i, \dots, x_n^i)$ and reference samples $z^i \sim p_0$, we compute $\hat{y}^i = \mathcal{G}_\theta(X^i, z^i)$, yielding a dataset for distillation $\{(X^i, z^i, \hat{y}^i)\}_{i=1}^M$ we use to train our GP surrogate $f(X, z) \approx \mathcal{G}_\theta(X, z)$. While distillation of flow matching models has been used for fast inference and regularized training [38, 73, 284], here it is used to make the learned conditional sampler amenable to kernel-based variance decomposition.

For a kernel K_s and regularization parameter γ, the surrogate is obtained by kernel ridge regression. That is,

$$f = \arg\min_{\varphi \in \mathcal{H}_{K_s}} \|\varphi\|_{K_s}^2 + \frac{1}{\gamma} \sum_{i=1}^{M} |\varphi(X^i, z^i) - \hat{y}^i|^2, \tag{6.5}$$

where $\mathcal{H}_{K_s}$ denotes the reproducing kernel Hilbert space (RKHS) associated with K_s. Here we will use the fact that sums of kernels induce corresponding minimum-norm decompositions of RKHS functions [244]. Namely, if $k = k_a + k_b$ and $f_k \in \mathcal{H}_k$, then its kernel-mode components are defined by

$$(f_a, f_b) = \arg\min_{g_a, g_b} \left\{ \|g_a\|_{k_a}^2 + \|g_b\|_{k_b}^2 : \; g_a + g_b = f_k, \; g_a \in \mathcal{H}_{k_a}, \; g_b \in \mathcal{H}_{k_b} \right\}.$$

They satisfy $f_k = f_a + f_b$ and $\|f_k\|_k^2 = \|f_a\|_{k_a}^2 + \|f_b\|_{k_b}^2$, so $\|f_a\|_{k_a}^2 / \|f_k\|_k^2$ measures the fraction of the RKHS signal energy that is attributed to the component associated with k_a. We use this decomposition to measure surrogate fidelity. Problem (6.5) can be interpreted as regression with the augmented kernel $K_s + \gamma\delta$, where δ is the white-noise kernel. The corresponding decomposition separates the signal component associated with K_s from the noise component associated with $\gamma\delta$. We

define

$$\mathrm{NSR} := \frac{V_n}{V_s + V_n}, \qquad V_s := \|f\|_{K_s}^2, \qquad V_n := \|f_{\gamma\delta}\|_{\gamma\delta}^2 = \frac{1}{\gamma}\sum_{i=1}^{M} |f(X^i, z^i) - \hat{y}^i|^2. \tag{6.6}$$

A large increase in NSR during pruning indicates that the removed variable significantly affects the conditional law. The second decomposition measures variable importance. To identify variables, we use the product kernel

$$K_s((X,z),(X',z')) = (1 + k_z(z,z')) \prod_{i=1}^{n} (1 + k_x(x_i, x_i')), \tag{6.7}$$

where k_x and k_z are universal kernels, such as radial basis function (RBF) or Matérn kernels. Expanding this product yields a sum of interaction kernels over subsets of the covariates, inducing a nonlinear ANOVA decomposition [41, 244]. For each variable x_j, we set

$$K_j = (1 + k_z(z,z'))\, k_x(x_j, x_j') \prod_{i \neq j} (1 + k_x(x_i, x_i')), \qquad K_{-j} = K_s - K_j.$$

The kernel K_j collects all ANOVA components involving x_j, while K_{-j} collects the components that do not involve x_j. Applying the minimum-norm splitting to $K_s = K_j + K_{-j}$ gives $f = f_j + f_{-j}$, and we define

$$c_j := \frac{\|f_j\|_{K_j}^2}{\|f\|_{K_s}^2} \in [0,1].$$

The score c_j measures the fraction of RKHS signal energy, or equivalently the GP/nonlinear-ANOVA variance contribution, attributable to components involving x_j [41, 244]; variables with smaller scores are treated as less influential. Ancestor discovery proceeds by iterative pruning. At each step, the variable with the smallest contribution score is removed and the GP surrogate is retrained on the reduced input set. The pruning sequence is run to completion, the NSR is recorded at each step, and the selected ancestor set $\mathcal{A}(y)$ is the set of variables remaining immediately before the largest NSR increment. The full algorithm is given in Appendix E.2. Unlike conditional independence tests [124, 373], which return binary accept/reject decisions, this procedure assigns each candidate variable a continuous contribution score through a variance decomposition of the learned conditional sampler. This supports pruning when finite data or indirect associations create weak dependencies among variables. It also avoids testing variables against large conditioning sets,

whose cost and reliability degrade with dimension [373]. Instead, a sharp increase in NSR indicates that pruning has removed an important input for the learned conditional law.

6.4.2 Digital twin modeling and treatment of dynamical systems

While our method applies broadly to the analysis of stochastic dependencies between arbitrary variables, a particular area of interest is digital twin modeling. We consider here a dynamical-system view in which the state is represented by a high-dimensional vector $q(t) \in \mathbb{R}^d$, potentially influenced by a control variable $u(t) \in \mathbb{R}^c$. The system evolves according to

$$\dot{q}(t) = \Phi(q(t), u(t), \xi(t)), \tag{6.8}$$

where $\xi(t)$ denotes a source of randomness or unresolved variability. In uncontrolled settings, u may be omitted or set to zero. In this scenario, digital twin modeling aims to construct an approximation of the input-output behavior induced by the dynamics Φ from observations of q and, when available, u [125, 272, 368]. In many applications, however, it is not necessary to model the full state evolution. Instead, one may only need to predict or explain the dynamics of a subset of variables of interest, denoted q_I. After discretizing time, the evolution of these variables can be written in reduced form as

$$\Delta q_I(t) = q_I(t + \Delta t) - q_I(t) = T_I(q_I(t), u(t), \omega(t)), \tag{6.9}$$

where $\omega(t)$ aggregates unobserved state variables, memory, discretization error, and noise, as motivated by delay-coordinate embeddings for deterministic and stochastically forced systems [323, 328]. We are interested in recovering the ancestors of $\Delta q_I(t)$ among the observed variables, given $q_I(t)$ and, in the controlled case, $u(t)$. We refer to these ancestors as the dynamic dependencies of q_I. This corresponds to setting $\mathcal{S} = (\Delta q_I(t), q_I(t))$ in Section 6.2, or $\mathcal{S} = (\Delta q_I(t), q_I(t), u(t))$ in the controlled setting, and recovering the bipartite graph G from q_I to Δq_I, or from (q_I, u) to Δq_I when controls are present.

Acknowledgments

We acknowledge support from the Air Force Office of Scientific Research under MURI award number FOA-AFRL-AFOSR-2023-0004 (Mathematics of Digital Twins), the Department of Energy under award number DE-SC0023163 (SEA-CROGS: Scalable, Efficient, and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems), the National Science Foundation under award number 2425909 (Discovering the Law of Stress Transfer and Earthquake

Dynamics in a Fault Network using a Computational Graph Discovery Approach) and the Vannevar Bush Faculty Fellowship under ONR-N000142512035. This material is also based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112590112. Approved for public release; distribution is unlimited.

Part III

Algorithm Discovery

Chapter 7

DISCOVERING ALGORITHMS WITH COMPUTATIONAL LANGUAGE PROCESSING

Théo Bourdais, Abeynaya Gnanasekaran, Houman Owhadi, and Tuhin Sahai. Discovering algorithms with computational language processing. *Science Advances*, 12(37):eaea4216, 2026. doi: 10.1126/sciadv.aea4216. URL https://doi.org/10.1126/sciadv.aea4216.
TB co-designed the CLP framework, ran the main experiments, and co-wrote the manuscript.

7.1 Introduction

Discovering algorithms is among the most creative of human endeavors. Efforts to formalize or automate this process date back more than half a century, from early theories of program discovery and inductive inference [315], to automatic code generation for numerical solvers [302], and systematic algebraic searches for faster matrix multiplication algorithms initiated by Bini et al. [36] which were continued in modern tensor-decomposition approaches [99]. Yet most of these efforts have remained narrowly problem-specific, for instance, in the game-theoretic design of fast, operator-adapted solvers [236], underscoring the difficulty of developing general frameworks for algorithm discovery. Alternatively, more recent approaches adopt direct code-generation paradigms such as AlphaEvolve [227] in which ensembles of large language models (LLMs) serve as coding agents. These are often integrated with evaluation metrics and evolutionary strategies, such as genetic algorithms, to iteratively refine large code segments [182, 192, 278]. AlphaEvolve and similar methods yield a single solution that is applied uniformly across all instances within a problem-class. However, such algorithms often overlook structural information that varies from one instance to another. Algorithms that exploit instance-specific structure are likely to outperform those that do not, motivating the development of adaptive approaches to algorithm discovery. In this work, we address this challenge by tokenizing the computation process itself, an abstraction that simplifies and generalizes complex algorithmic representations, which we term Computational Language Processing (CLP).

Next, we introduce an RL *ensemble* variant of MCTS to efficiently explore the

space of grammatically consistent token chains (Fig. 7.1, A and B). Contrary to the prevalent paradigm, where an algorithm is designed for an entire class of problems (e.g., combinatorial optimization) and uniformly applied across all instances, our framework is instance-adaptive and performs reinforcement learning directly at the computational level of each specific problem instance. As a result, our approach can generate *both* problem class algorithms and *distinct, instance-specific algorithms finely tailored to the nuances of individual problems*. We demonstrate the effectiveness of this approach in algorithmic discovery through applications to the Quadratic Assignment Problem (a generic strongly NP-hard combinatorial optimization problem to which many other NP-hard problems can be reduced, see Sec. F.1), the Quantum (unstructured) Search Problem and Quantum Approximate Optimization Algorithm (QAOA). Crucially, this high-level framework not only simplifies the conceptualization and generalization of complex algorithms and interfaces naturally with downstream code-generation methods, but also has the potential to extend to other domains such as systems engineering and mathematical discovery.

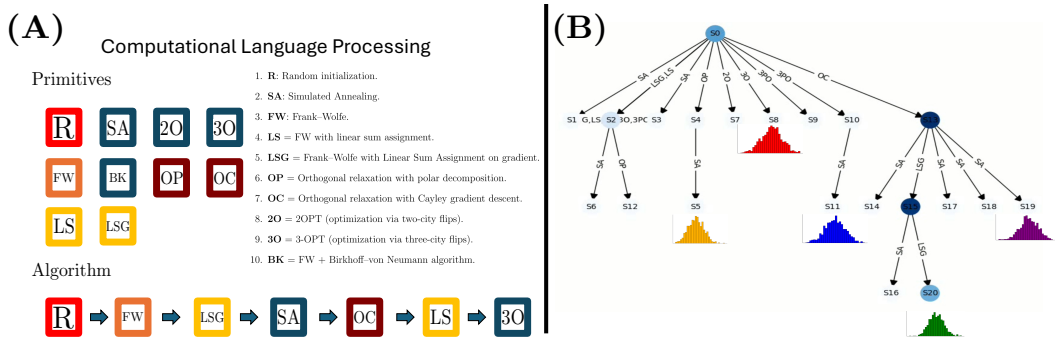


Figure 7.1: **Overview of Algorithm Tokens and MCTS+RL Search Tree:** (**A**) We conceptualize an algorithm as a chain of elementary computational steps (primitives/tokens), chained with a precise grammar/syntax. (**B**) Starting with elementary tokens, we discover new tokens and algorithms with ensemble Monte Carlo tree search (MCTS) + reinforcement learning (RL) as a babbling mechanism to search over the tree of all possible strings of tokens. Edges represent primitives/tokens and nodes the ensemble of computational state variables sampled from the string of (possibly stochastic) tokens leading that node.

7.2 Results

7.2.1 Results on the Quadratic Assignment Problem (QAP).

The QAP is a foundational combinatorial optimization problem that arises in many domains, including supply chain optimization, facility layout, airport design, data center placement, and very large-scale integrated circuit design. For additional

background on the QAP and its practical relevance, see Sec. F.1.1. Because of its combinatorial search space and strong NP-hardness, even moderately sized instances (e.g., $n = 20$) are computationally challenging, and achieving any constant-factor approximation is itself NP-hard [48]. Beyond its intrinsic difficulty, the QAP subsumes several classical optimization problems. For instance, setting F to be the adjacency matrix of the cycle graph of length n with symmetric distances $D_{i,j} = D_{j,i} \geq 0$, $D_{i,i} = 0$, and $C_{i,j} = 0$ in the QAP loss (Eqn. 7.1) reduces it to the Traveling Salesman Problem (TSP). Similarly, setting $C = 0$ and restricting $D_{i,j} \in \{0, 1\}$ yields the Graph Matching Problem [287]. Developing an automated framework capable of generating instance-adapted algorithms that outperform state-of-the-art solvers for the QAP would, therefore, have substantial scientific and industrial impact. To formally define the QAP, let D, F, and C be $n \times n$ matrices with arbitrary real-valued entries, where $n \geq 2$. Let Π_n be the set of all permutations of $\{1, 2, \ldots, n\}$, the objective of the QAP is to minimize $\min_{\pi \in \Pi_n} \mathcal{L}(\pi)$ where,

$$\mathcal{L}(\pi) := \sum_{i=1}^{n} \sum_{j=1}^{n} F_{i,j} D_{\pi(i),\pi(j)} + \sum_{i=1}^{n} C_{i,\pi(i)} = \mathrm{Tr}\left[FPD^T P^T + CP^T\right], \tag{7.1}$$

and $P \in \mathcal{P}_n$ is a $n \times n$ permutation matrix associated with π.

Algorithm discovery can be tackled at different levels of problem complexity, from hyperparameter tuning to heuristics design. To explore this spectrum, we evaluate our method on two benchmark generators, the *composite* QAP (CQAP) [91], the significantly more challenging Palubeckis QAP (PQAP) [247], and the Quadratic Assignment Problem Library (QAPLIB) library of challenge problems. In three different experiments, CLP is provided with a base set of tokens and learns to compose them into increasingly sophisticated algorithms using a lightweight Transformer model guided by an ensemble variant of MCTS. With this approach, we obtain the following results: (i) Using well-established algorithms as computational tokens, (see Sec. F.1.3 for the entire list of tokens), our algorithm learns to chain these strategies to beat state-of-the-art metaheuristics and the mixed-integer commercial solver Gurobi. This performance is achieved across a range of problem sizes and instances with varying underlying structure, while keeping the number of objective evaluations constant. As shown in Fig. 7.2, the resulting policies achieve optimal solutions in 100% of tested instances from CQAP for $n \leq 80$, $\leq 1\%$ optimality gap on PQAP for $n \leq 80$ and beat or equal all other tested methods in all 390 tested PQAP and CQAP instances. On previously unseen QAPLIB examples, CLP generalizes to new problem structures, solving 64% of QAPLIB instances to optimality and beating or

equaling all other tested methods in 100% of QAPLIB instances. See Sec. 7.7.2 for experimental details. (ii) Starting from a minimal set of primitive operations such as copy operations, gradient steps, negation, Hungarian assignment (to map continuous matrices to permutation matrices), and simple control-flow tokens, CLP expands the library to 100+ composite tokens using a variant of Byte-Pair Encoding [109] for computational tokens, which we term Algorithmic Byte-Pair Encoding (A-BPE) (see Materials and methods). After just three refinement rounds (of A-BPE and neural network retraining) it rediscovers many high-level tokens including the well-known k-OPT and Frank-Wolfe (FW) algorithms. It also learns to perform random restarts for FW (see Fig. 7.6), enabling it to reach optimality in over 90% of cases for CQAP instances of size 40, surpassing what standard implementations of FW achieve. See Sec. 7.7.1. (iii) When focused on step-size optimization, CLP discovers a new cyclic step-size schedule that significantly outperforms the standard step-size schedule implemented within the `SciPy` software package [344] (see Sec. 7.7.1.1).

In short, CLP demonstrates effectiveness across all aspects of algorithmic design: improving performance at a high-level, rediscovering known algorithms augmented with novel low-level restarting strategies, and optimizing associated parameters. CLP outperforms all baselines, including commercial solvers such as Gurobi, while scaling cubically and providing fast answers on large problem instances.

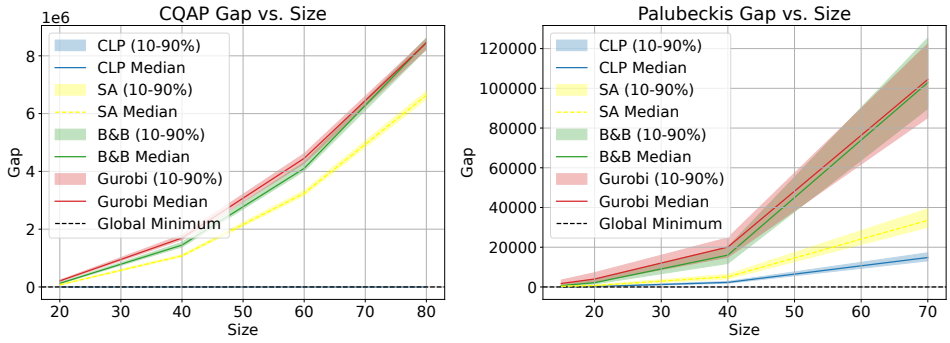


Figure 7.2: **Optimality gap for different methods:** the figure displays median, 10th and 90th quantiles of optimality gap. Computational Language Processing (CLP) beats all other methods, including Simulated Annealing (SA) and Branch and Bound (B&B), on all instances, and achieves global optimum on all tested Composite Quadratic Assignment Problem (CQAP) instances

7.2.2 Results on the Quantum Search Problem.

Consider a finite, unstructured set $\mathcal{X}$ (with no inherent ordering or other distinguishing characteristics) containing N elements, indexed by the set $\{0, 1, \ldots, N-1\}$. Among the elements of $\mathcal{X}$, there is exactly one *marked* or *target* element $x^* \in \mathcal{X}$. The task is to identify the marked element x^* by querying an oracle (black-box) function $f : \mathcal{X} \to \{0, 1\}$, where $f(x) = 1$ if $x = x^*$ and $f(x) = 0$ otherwise. This oracle provides no additional information about the location of the target beyond the binary responses to queries. The objective is to determine the marked element x^* efficiently, minimizing the number of queries to the oracle function f. In the quantum setting, the oracle is represented by a unitary operator U_f, acting on computational basis states $|\boldsymbol{x}\rangle$ as $U_f|\boldsymbol{x}\rangle = (-1)^{f(\boldsymbol{x})}|\boldsymbol{x}\rangle$. This oracle operator flips the sign of the amplitude corresponding to the marked state. Classically, since the set $\mathcal{X}$ is unstructured, the optimal solution is an exhaustive linear search whose complexity is $O(N)$. Using the Quantum Oracle Representation, quantum computing leverages principles of superposition and amplitude amplification to achieve a provable quantum advantage. Grover's algorithm [126] accomplishes this, solving the quantum search problem, in approximately $\frac{\pi}{4}\sqrt{N}$ oracle queries with a high probability of success. The quantum search problem is a cornerstone of quantum algorithmic theory because it (a) demonstrates a provable quantum speed-up, (b) has significant implications for cryptography, and (c) serves as a subroutine in numerous quantum algorithms.

An n-qubit quantum state $|\psi\rangle$ lives in a 2^n-dimensional Hilbert space, and is expressed in the computational basis as: $|\psi\rangle = \sum_{\boldsymbol{x}\in\{0,1\}^n} a_{\boldsymbol{x}}|\boldsymbol{x}\rangle$, where $\sum_{\boldsymbol{x}} |a_{\boldsymbol{x}}|^2 = 1$. Given a problem of size N on $n = \log_2(N)$ qubits, we formulate the discovery of an algorithm for the quantum search problem as finding a sequence of quantum gates that maximizes the probability of measuring the target state with *minimum circuit depth*. The CLP framework is set up with the following primitives (quantum gates) that make up Grover's algorithm: (I) The Hadamard gate on n qubits whose action is on $\boldsymbol{x} \in \{0, 1\}^n$ is given by $H^{\otimes n}|\boldsymbol{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{y}\in\{0,1\}^n} (-1)^{\boldsymbol{x}\cdot\boldsymbol{y}}|\boldsymbol{y}\rangle$ where $\boldsymbol{x} \cdot \boldsymbol{y} = x_1y_1 + x_2y_2 + \cdots + x_ny_n \pmod 2$. (II) The oracle that marks the target state $|\boldsymbol{w}\rangle$ as $U_f|\boldsymbol{x}\rangle = -|\boldsymbol{x}\rangle \quad \text{if} \quad \boldsymbol{x} = \boldsymbol{w}$ and $U_f|\boldsymbol{x}\rangle = |\boldsymbol{x}\rangle \quad \text{if} \quad \boldsymbol{x} \neq \boldsymbol{w}$. (III) The Pauli-X (NOT) gate on n qubits $X^{\otimes n}|\boldsymbol{x}\rangle = |1 - x_1, 1 - x_2, \ldots, 1 - x_n\rangle$. (IV) The multi-controlled Z (MCZ) gate that applies a phase flip when all qubits are $|\mathbf{1}\rangle$, i.e., $\text{MCZ}\,|\boldsymbol{x}\rangle = -|\boldsymbol{x}\rangle$ if $x_1 = x_2 = \cdots = x_n = 1$ and $\text{MCZ}\,|\boldsymbol{x}\rangle = |\boldsymbol{x}\rangle$ otherwise.

The circuit discovered by our framework is shown in Fig. 7.3. Compared to the

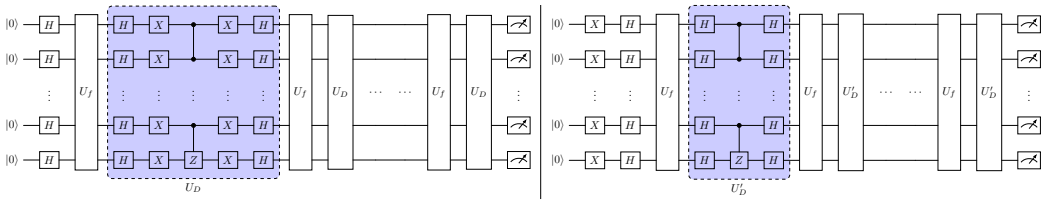


Figure 7.3: **Comparison of Grover's algorithm implementations**: standard (left, U_D is the standard diffusion operator) and optimized (right, with simplified diffusion operator U'_D).

standard Grover's circuit, also shown in Fig. 7.3, *the circuit depth is nearly halved*, yielding a reduction of $(\frac{\pi}{2}\sqrt{N} - 1)$ layers, and the number of gates is reduced by $(\frac{\pi}{2}\sqrt{N} - 1)\log N$. Quantum errors scale exponentially with circuit depth and number of qubits due to decoherence and noise. Consequently, our approach results in an exponential improvement in robustness measures. The *standard Grover's algorithm* can be summarized as follows. (1) Start with $|\mathbf{0}\rangle^{\otimes n}$. Apply Hadamard gates to create uniform superposition: $|\boldsymbol{s}\rangle = H^{\otimes n}|\mathbf{0}\rangle^{\otimes n} = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|\boldsymbol{x}\rangle$. (2) Repeat $O(\sqrt{N})$ times: (a) Apply oracle U_f to mark the target state. (b) Apply diffusion operator $D = 2|\boldsymbol{s}\rangle\langle\boldsymbol{s}| - I$. The *optimized Grover's algorithm* can be summarized as follows. (1) Start with $|\mathbf{0}\rangle^{\otimes n}$. (2) Apply X gates to all qubits: $X^{\otimes n}|\mathbf{0}\rangle^{\otimes n} = |\mathbf{1}\rangle^{\otimes n}$. Apply Hadamard gates to create a different superposition: $|\boldsymbol{s'}\rangle = H^{\otimes n}|\mathbf{1}\rangle^{\otimes n} = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}(-1)^{|\boldsymbol{x}|}|\boldsymbol{x}\rangle$ where $|\boldsymbol{x}|$ is the Hamming weight (number of 1's) in the binary representation of $\boldsymbol{x}$. (3) Repeat $O(\sqrt{N})$ times: (a) Apply oracle U_f to mark the target state. (b) Apply modified diffusion operator $D' = 2|\boldsymbol{s'}\rangle\langle\boldsymbol{s'}| - I$. We show in Sec. 7.8 that our discovered gate sequence is mathematically equivalent to Grover's algorithm and is an optimized version constructed using the same gate set. The key insight is that by starting with $|\mathbf{1}\rangle^{\otimes n}$ instead of $|\mathbf{0}\rangle^{\otimes n}$, we can directly implement the diffusion operator using just $H^{\otimes n} \rightarrow \text{MCZ} \rightarrow H^{\otimes n}$, eliminating the need for X gates at every iteration.

7.2.3 Results on the Quantum Approximate Optimization Algorithm (QAOA).

To show that our approach is applicable to settings where the practitioner is interested in circuits with prescribed depth, we apply CLP to QAOA [97], a hybrid quantum-classical optimization algorithm designed to approximately solve combinatorial optimization problems. Here, our goal is to reduce the optimization cost (compute *better* minima) for a given circuit depth. Specifically, we use our approach to generate mixer Hamiltonian circuits for solving the MAX-CUT problem on p-regular

graphs. In Sec. 7.9, we show that our approach achieves a 35% improvement when compared with ADAPT-QAOA [381], the adaptive version of QAOA implemented within NVIDIA's quantum computing library (CUDA-Q). The policy and value networks were trained on 1000 examples and then tested on graphs that were not seen previously. See supplementary materials for further details.

7.3 Discussion

In this work, we demonstrated the versatility of CLP through three representative examples of algorithm discovery, spanning classical combinatorial optimization and quantum computation. The framework composes and refines computational primitives to recover known algorithms and generate instance-adapted variants that match or exceed existing baselines. While our results focus on these specific domains, the underlying approach may extend to other areas of automated algorithm design. CLP currently relies on manually crafted computational primitives; a natural next step is to automate their generation. Furthermore, scaling CLP to larger problems will require mitigating the cost of its RL component.

7.4 Materials and methods

7.4.1 Computational Language Processing.

Our framework exploits the conceptualization of an algorithm as "a sequence of finite computational steps that transform an input into a desired output" [68]. Since these computational steps naturally correspond to computational graphs [41, 239], the problem of discovering an algorithm reduces to discovering a computational graph whose structure can be represented as a sequence of known elementary computational graphs/steps which we call *primitives* and that can be chained into more complex operations that we call tokens. Drawing a parallel between computational primitives and letters, we view an algorithm as a sentence (Fig. 7.1(A)), and algorithm discovery as learning to speak in a computational language. This perspective motivates a language-based learning approach, abstracting away from domain-specific tasks to focus on chaining arbitrary computational steps. This enables algorithm discovery across diverse problem domains with a *unified methodology*. Therefore, analogous to natural language, the set of primitives serves as the *alphabet* for our computational language.

Turbo codes (which transformed digital communications by enabling efficient and reliable data transmission near the theoretical limit) provide an example of how composing simple computational primitives can yield a novel and powerful algo-

rithm. As the inventors, Claude Berrou and Alain Glavieux later reflected, turbo codes were "the result of an empirical, painstaking construction of a global coding/decoding scheme, using existing bricks that had never been put together in this way before" [130].

Our framework automates and amplifies this process. Unlike natural language, no large dataset of training examples exists for our task. To address this, the framework alternates between two complementary phases (see Fig. 7.1(A), 7.4(A), and 7.5): (i) an ensemble MCTS–RL paradigm that automates and scales the exploration of chains of computational steps, composing elementary primitives into progressively more complex operations (analogous to forming words in natural language), i.e., tokens; and (ii) incremental vocabulary expansion by merging frequently occurring and efficient token sequences, inspired by Byte-Pair Encoding (BPE) [109].

7.4.2 Ensemble Monte Carlo tree search with reinforcement learning.

To explore and refine the chaining of computational primitives, we integrate MCTS [46] with trained policy and value neural networks. MCTS incrementally builds search trees using random sampling and targeted exploration, efficiently focusing on promising moves without exhaustively searching the entire tree. Combined with deep neural networks and self-play, this strategy reached breakthroughs like AlphaGo's superhuman performance in Go and AlphaZero's rapid dominance in chess [299, 309, 310]. Our approach is a new ensemble variant of the AlphaZero reinforcement learning framework as detailed in Sec. 7.6. In this variant, edges correspond to computational actions represented by tokens, and each node maintains an ensemble of computational states derived from the sequence of tokens leading to that node (Fig. 7.1(B)). The framework is specifically designed to handle stochastic operations such as random initializations (e.g., random permutations) or stochastic computational operations, as they are key building blocks for designing efficient algorithms. In addition to token sequences, Boltzmann-weighted averages (akin to a softmax) of computational states are computed and serve as input features for the policy and value neural networks. In the setting of algorithm discovery, our approach learns to chain tokens into algorithms *dynamically tailored to specific problem instances*. In several of our test cases, when a single algorithm proves effective across all instances of a problem class, the learned policy tends to converge toward that solution. This convergence suggests that the framework can recover general-purpose algorithms applicable to entire problem classes when such solutions exist.

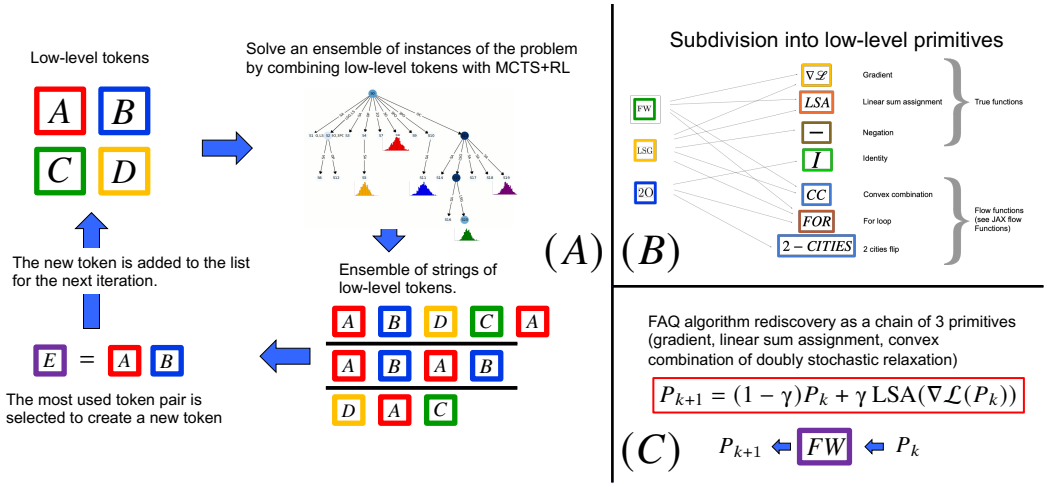


Figure 7.4: **Navigating the complexity of algorithmic tokens, complexification and subdivision using Monte Carlo Tree Search (MCTS) + Reinforcement Learning (RL)**: (**A**) Algorithmic Byte-Pair encoding to increase vocabulary by creating new tokens from old ones (**B**) Low-level primitives (**C**) Fast Approximate Quadratic assignment problem (FAQ) algorithm rediscovery as a chain of three primitives (gradient, linear sum assignment, convex combination of doubly stochastic relaxation). See supplementary Sec. F.1.3 for low-level primitive definitions.

7.4.3 Algorithmic byte-pair encoding and learning higher-complexity tokens.

The choice of primitives, i.e. the computational alphabet in CLP, directly governs the trade-off between expressivity and complexity of the search space. Imposing strong prior assumptions on the structure or choice of primitives introduces a significant bias towards the form of algorithm that is ultimately discovered. Conversely, while low-level primitives such as arithmetic operations or machine code offer maximum granularity, they also dramatically enlarge the search space, making discovery intractable. As a result, coding agents based on LLMs rely heavily on supervised pretraining to serve as frameworks for algorithm discovery. This inherently biases them towards existing algorithms and known patterns. Even when fine-tuned using RL, the often-used Kullback-Leibler (KL) penalty limits exploration by anchoring updates to the pretrained distribution. This results in distributional shifts rather than genuine capability expansion [169, 337]. Additionally, the vast parameter count of LLMs makes RL-based exploration at inference computationally prohibitive. In contrast, we define a discrete set of computational tokens, allowing us to precisely control the amount of prior structure and eliminate the need for pretraining. This enables unbounded exploration driven solely by RL, while drastically reducing model size.

However, a fixed, manually designed vocabulary may constrain expressivity and ne-

cessitate long token sequences to represent complex behaviors, thereby increasing the search space and computational burden. To address this, we draw inspiration from natural language processing (NLP) again to automatically discover tokens most adapted to the task at hand. Since we view algorithm discovery as learning to speak a computational language, RL rollouts can be viewed as "babbling", i.e. learning speech through trial and error. We collect these rollouts to construct a corpus of effective computational sequences for the current problem. Using this corpus, we apply a variant of the BPE algorithm, termed Algorithmic Byte-Pair Encoding (A-BPE), to iteratively merge frequently co-occurring token sequences into new, higher-level tokens. These tokens represent higher-level computational steps and can themselves be recursively composed, supporting hierarchical abstraction. This enables the iterative construction of complex algorithms and a dynamic combination of low and high-level representations (see Fig.7.4(A), Fig. 7.5, and Sec. 7.7.1–7.7.3).

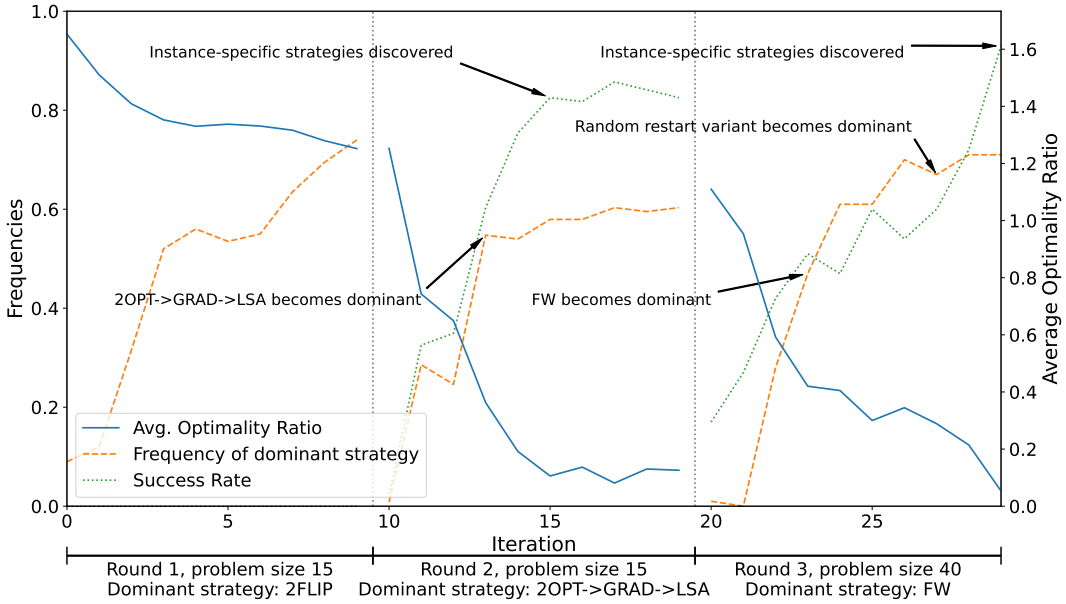


Figure 7.5: **Emergence of dominant strategies as difficulty increases**: (left-axis) Frequencies of occurrence of dominant strategy and finding the global optimum. (right-axis) Average optimality ratio, $(\mathcal{L} - \min \mathcal{L})/\min \mathcal{L}$. (x-axis) A-BPE rounds occurring every 10 iterations are marked by a vertical dotted line. CLP discovers increasingly more complex and effective algorithms, with identifiable dominant strategies and instance-specific adaptations, to reach a high success rate. Strategies include Frank-Wolfe (FW), 2-city flip (2FLIP), 2-city optimal (2OPT), gradient (GRAD), Linear Sum Assignment (LSA). Details in 7.7.1

7.4.4 Primitives, tokens and Algorithmic Byte-Pair Encoding for the QAP.

To illustrate the concepts of primitives and tokens, we describe selected primitives commonly employed for solving the QAP, into which most existing algorithms naturally decompose (Fig. 7.4(A)–(C)). A representative subset of these primitives operates on a relaxation of the QAP from permutation matrices to the larger set of $n \times n$ *doubly stochastic matrices*, denoted by $\tilde{\mathcal{P}}_n$. A matrix $P \in \tilde{\mathcal{P}}_n$ satisfies $P_{i,j} \in [0, 1]$ for all entries, with row and column sums equal to one, i.e., $\sum_j P_{i,j} = 1$ and $\sum_i P_{i,j} = 1$. By extending the loss function $\mathcal{L}$ to this relaxed domain, we obtain its gradient with respect to the matrix P as follows: $\nabla\mathcal{L}(P) = FPD^\top + F^\top PD + C$. We introduce another fundamental primitive, *interpolation between two doubly stochastic matrices*, defined by: $P \leftarrow (1-\gamma)\,Q + \gamma\,S$, where $Q, S \in \tilde{\mathcal{P}}_n$ and $\gamma \in [0, 1]$, resulting in another doubly stochastic matrix. Additionally, given an arbitrary cost matrix $C \in \mathbb{R}^{n\times n}$, we consider the Linear Sum Assignment (LSA) problem, which seeks a permutation π minimizing the total assignment cost $\sum_{i=1}^{n} C_{i,\pi(i)}$. The Hungarian (or Kuhn–Munkres) algorithm [176, 224] solves this problem optimally in $O(n^3)$ time. We denote the optimal permutation matrix solving the LSA defined by C as: $\mathrm{LSA}(C) := \mathrm{argmin}_{P\in\mathcal{P}_n} \sum_{i,j=1}^{n} C_{i,j}P_{i,j}$. Although we use the LSA algorithm as a primitive, in this case, we show how these algorithms can themselves be rediscovered using low-level primitives (see Fig. 7.5). Additional primitives are detailed in Sec. F.1.3. To illustrate vocabulary generation and token formation, consider the Frank–Wolfe (FW) algorithm's update step for the QAP:

$$P_{k+1} = (1-\gamma_k)P_k + \gamma_k\ \mathrm{LSA}(\nabla\mathcal{L}(P_k)).$$

This update naturally decomposes into an interpolation token and a token representing the LSA operation applied to the gradient, each corresponding to previously defined computational primitives (Fig. 7.4(b,c)). Such decompositions demonstrate the incremental creation and complexity building inherent in our computational language framework. Fig. 7.5 further demonstrates the practical application of our A-BPE strategy (see Fig. 7.4(A)) for token construction in the context of the QAP. Starting from a modest initial set of 8 elementary primitives (see Sec. 7.7.1), the CLP approach leveraging A-BPE progressively discovers increasingly sophisticated and effective algorithms (tokens), emerging as dominant (most frequently employed) strategies (i.e., strings of tokens) over multiple rounds. For example, the FW algorithm emerges as a dominant string of computational tokens. The resulting tokens exhibit both a recognizable dominant strategy and instance-specific adaptations, ultimately achieving high success rates (see Sec. 7.7.1 for details).

7.5 Acknowledgments

The authors thank Prof. Yannis Kevrekidis and Dr. Robert Lazar for insightful feedback and comments. This research was to a large extent initiated by Prof. Kevrekidis' vision [131, 218] articulated in his Defense Advanced Research Projects Agency (DARPA) DIAL program (Mathematics for the Discovery of Algorithms and Architectures) whose objectives were to "explore disruptive capabilities in computer-aided algorithm discovery via optimization".

Funding: This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112490483. Approved for public release; distribution is unlimited. TB and HO acknowledge support from the Vannevar Bush Fellowship program (Office of Naval Research (ONR) award number N000142512035).

Author contributions: TB, AG, HO, and TS contributed to conceptualization, formal analysis, investigation, methodology, software, validation, visualization, and writing–original draft. TB, AG, and HO contributed to data curation; TB, HO, and TS to resources and writing–review & editing; and HO and TS to funding acquisition, project administration, and supervision.

Competing interests: SRI International and the California Institute of Technology have filed a U.S. Provisional Patent Application (No. 63/861,802; filed 11 August 2025) entitled 'Discovering Algorithms with Computational Language Processing,' which is related to the work reported in this manuscript. All authors (T.B., H.O., T.S., and A.G.) are associated with this patent application. All other authors declare no competing interests.

Data, Code, and Materials Availability. All data and code needed to evaluate and reproduce the results in the paper are present in the paper, the Supplementary Materials, and the Zenodo repository at https://zenodo.org/records/17945389 (digital object identifier: 10.5281/zenodo.17945389). This study did not generate new materials.

7.6 Ensemble Monte Carlo tree search (MCTS) and reinforcement learning (RL) for algorithm discovery

We adapt the AlphaZero variant of MCTS [46, 299, 309, 310] as a "babbling" engine that chains tokens to obtain candidate algorithms. We use our adapted RL

approach to (i) find generic solvers for the QAP and construct quantum circuits for quantum search and optimization, and (ii) construct instance-adapted algorithms that are generated on-the-fly. Details of our ensemble MCTS approach follow, and we omit aspects that do not deviate from AlphaZero.

State Representation. At depth d, state $\boldsymbol{s}_d$ is defined by the sequence of previous actions taken $(a_1, \ldots, a_{d-1})$. Additional information can be extracted from the state to get a more informative representation to input the neural network, such as loss increments of the states $(\mathcal{L}(\boldsymbol{s}_2) - \mathcal{L}(\boldsymbol{s}_1), \ldots, \mathcal{L}(\boldsymbol{s}_d) - \mathcal{L}(\boldsymbol{s}_{d-1}))$ and the sequence of times taken to compute each action (see below).

Transformer-based policy and value networks. Having formulated the state as a sequence of actions, rather than a problem-specific state such as a permutation or a quantum state, it is natural to use transformers to perform the task of predicting the policy and value from the state. By abstracting the problem this way, we obtain a unified learning framework for algorithm discovery, independent of the underlying problem. It also ensures effective generalization across problem instances of varying sizes, as the tokens themselves are defined for any size.

Ensemble Variant and Handling Stochasticity. To accommodate stochasticity arising from random initializations and stochastic actions, we use ensembles of candidate solutions at each node, typically 5 to 10, and keep track of the best candidate ever seen along each path. Any time a node is reached, we compute a new batch of candidate solutions using $\boldsymbol{s}_d = (a_1, \ldots, a_{d-1})$. We compute the Gibbs average of the losses $(\sum_i \mathcal{L}_i e^{-\beta \mathcal{L}_i} / \sum_i e^{-\beta \mathcal{L}_i})$ of all candidates ever seen at that node, which gives an estimate of the expected loss achieved by the sequence of actions $\boldsymbol{s}_d$, $\mathcal{L}(\boldsymbol{s}_d)$. This gives more information about a node's quality than sampling one candidate, and is fed to the transformers.

Computational cost control. To account for the vast difference in cost between different primitives, our MCTS search incorporates computational cost into its decision process. A total compute budget is allocated, and execution halts once no further action is feasible within that budget. To account for the cost of the policy and value network evaluations, which are proportional to the depth of the tree, we also add a fixed compute penalty at each depth level. This compute budget information is also fed to the transformers, to allow greater planning and compute management. We also add an early stopping primitive, so that not all of the compute is necessarily expended. Combined with a discounting of the rewards for high use of compute, this fosters the discovery of computationally efficient algorithms.

Termination and correctness. All searches are bounded by a global compute budget that includes both primitive execution and policy/value evaluations; loops are implemented as bounded tokens (e.g., FOR(k)), and an explicit STOP primitive enables early termination. These constraints guarantee halting for every rollout. Correctness is handled differently depending on the task. For Grover's algorithm, we prove correctness *a posteriori* after algorithm generation (see Sec. 7.8). For optimization tasks such as the QAP, while mathematical guarantees like objective bounds are NP-hard, the grammar enforces feasibility of all proposed solutions, even when the generated algorithms rely on intermediate computations not explicitly involving permutations.

7.7 CLP applied to the QAP

7.7.1 Discovering new strategies with low-level primitives and CLP

Our algorithm discovery framework is capable of operating with both low-level and high-level primitives. Using the QAP as a prototypical example, we now describe how it discovers new algorithms by composing simple base primitives. The framework alternates between two phases: (1) learning to compose primitives using MCTS, and (2) introducing new primitives by concatenating existing ones, following a strategy inspired by Byte-Pair Encoding (BPE) [109].

Computational grammar. When analyzing existing algorithms, we observe that they often share common building blocks, which we will interpret as low-level primitives. Many of these primitives naturally correspond to matrix-level operations, such as gradient computations. However, these intermediary computations often impose specific input requirements, such as being a permutation, that are not guaranteed to be preserved in their outputs. This imposes constraints on how primitives can be chained. Furthermore, certain fundamental algorithmic constructs do not neatly map to operations at the matrix level. For instance, a for-loop does not itself directly operate on matrices; instead, it represents repeated application of another primitive. To address this, we introduce the notion of operator primitives, operators that act upon other primitives to generate new composite primitives. Because the effect of an operator primitive depends on the structure of the primitive it modifies, additional constraints arise when composing them. All these constraints naturally give rise to a grammar that defines the set of valid primitive compositions. There is an inherent trade-off between the expressivity afforded by a richer primitive set and grammar, and the complexity involved in effectively composing these primitives.

To manage this trade-off, we restrict operator primitives to operate exclusively on a single primitive at a time. As a result, certain algorithms, such as Frank-Wolfe, while expressible in principle, are not immediately accessible due to their reliance on nested applications of operator primitives. In such cases, expanding the primitive set through Byte-Pair Encoding becomes essential to recover complex algorithms.

Control-like behaviors are encapsulated as tokens (e.g., bounded FOR, early STOP, and selection via BEST or parallel evaluation with PU/2SWAP), extending expressivity beyond a purely linear sequence while preserving a compositional grammar and analyzable budgets.

Used tokens. Let $\mathcal{M}$ denote the relevant state space, here the space of matrices used to represent relaxed or discrete QAP states (square matrices of size n). The chosen primitives are of three kinds: matrix-to-matrix primitives, operator primitives, and the syntax primitive STOP. Matrix-to-matrix primitives such as COPY, GRAD, LSA, and NE (detailed below) have type

$$\mathcal{M} \to \mathcal{M}\,.$$

Operator primitives such as RU, PU, FOR, and 2SWAP (described below) take a matrix-to-matrix primitive as input and return a new matrix-to-matrix primitive; they therefore have type

$$(\mathcal{M} \to \mathcal{M}) \to (\mathcal{M} \to \mathcal{M}).$$

Matrix-to-matrix function primitives include the copy primitive (COPY: $x \mapsto x$), gradient (GRAD: $x \mapsto \nabla\mathcal{L}(x)$), Linear Sum Assignment (LSA: $x \mapsto \arg\min_\pi \langle x, \pi\rangle_F$), and negation (NE: $x \mapsto -x$). Operator primitives include a k-iteration loop (FOR) and residual updates (RU, which maps an arbitrary primitive P to RU$[P]$: $x \mapsto x + P(x)$).

Although $\mathrm{COPY}(x) = x$ at the value level, COPY is not included merely for padding. It is the neutral argument that allows an operator primitive to act without imposing an additional transformation. As an example, $\mathrm{RU}[\mathrm{COPY}](x) = 2x$, since $\mathrm{RU}[\mathrm{COPY}](x) = x + \mathrm{COPY}(x)$. This allows expressibility by recursive composition of RU, i.e. $\mathrm{RU}^k[\mathrm{COPY}] = x \mapsto (k+1)x$. Similarly, as described below, $\mathrm{2SWAP}[\mathrm{COPY}](x)$ enumerates all one-transposition or two-city-swaps of x, and returns the candidate with the lowest loss. Thus, k compositions of 2SWAP (i.e. $\mathrm{2SWAP}^k$) generate increasingly rich multi-swap neighborhoods, including all

neighborhoods used in $(k+1)$-OPT-style local search. This allows us to have a wide variety of expressible functions via composition while keeping the number of tokens limited. It also avoids excessively constraining the discovery process with prior knowledge.

We now describe in detail the parallel 2-city flips operator. Given a permutation matrix x and a primitive P (mapping permutation matrices to permutation matrices), 2SWAP$(P)(x)$ is computed as follows: (a) compute the permutations that are one 2-city flip away from x, i.e. $y_{i,j} = SWAP(x, i, j)$, for $i, j = 1, \ldots, n$, where $SWAP(x, i, j)$ denotes the permutation obtained from x by swapping indices i and j. (b) compute the images $z_{i,j} = P(y_{i,j})$, (c) return the first $z_{i,j}$, in index order, among those with minimal loss (such that $z_{i,j} = \arg\min_{z_{k,l}} \mathcal{L}(z_{k,l})$). Mathematically, 2SWAP$[P]$: $x \mapsto \arg\min_{z_{i,j}} \{\mathcal{L}(P(y_{i,j})) \mid z_{i,j} = P(y_{i,j}), y_{i,j} = SWAP(x, i, j)\}$.

We also introduce the parallel uniform operator primitive PU, which samples 10 random i.i.d. permutation matrices $y_1, \ldots, y_{10}$ according to the uniform distribution $\mathcal{U}$ over the finite set of permutation matrices, applies a given primitive P (mapping permutation matrices to permutation matrices) to each, and keeps the resulting state that has the smallest loss. This results in the operation PU$[P]$: $x \mapsto \arg\min_{z_i} \{\mathcal{L}(z_i),\ z_i = P(y_i),\ y_i \sim \mathcal{U},\ i = 1, \ldots, 10\}$.

Finally, we add the syntax primitive STOP to learn to perform early stopping and allow us to encourage more efficient algorithms through the reward (see computational cost control in 7.6).

Algorithmic Byte-Pair Encoding (A-BPE). After each round of babbling, we collect the sequences of tokens selected by ensemble MCTS. We count how often each pair of consecutive tokens appears in the corpus, excluding pairs that reconstruct existing tokens (e.g., applying the negative primitive twice to form the identity). The most common pair is merged into a new token, and all its instances in the corpus are replaced accordingly. This process repeats until no remaining pair occurs at least 10 times.

Results. We run multiple rounds of consecutive babbling, with 10 iterations of self-play each, and A-BPE on QAP problems of size 15 then 40. The results of this experiment are shown in Fig. 7.5. After one round of babbling, the majority of rollouts end with a 2-city swap and copy, i.e. the best flip (denoted 2FLIP, defined as $2SWAP(COPY)$). *A-BPE identifies this pair as frequent and treats it in*

subsequent rounds as an atomic token, thereby reducing search depth and improving exploration efficiency. After one round, composing 2FLIP with a bounded FOR loop then becomes feasible, yielding the popular 2OPT strategy used in algorithms such as the Lin–Kernighan heuristic. While around 60% of rollouts finish with 2OPT->GRAD->LSA, the success rate (i.e. the frequency of finding the global optimum) reaches 80%, indicating that other, instance-specific strategies are discovered. After the second round of A-BPE, we move to problems of size 40, which are significantly harder, and renders 2OPT impractical due to its much higher computational cost and reduced effectiveness. In earlier iterations of our method, where computational cost was not taken into account, subsequent rounds discovered 3OPT and 4OPT, which are very effective but prohibitively expensive when implemented in a naive fashion without pruning. We note that, just as in the RU^k case, the introduction of the 2SWAP and COPY as separate primitives, rather than solely the 2FLIP primitive, allows us to generate any k-OPT operation without having to define separate primitives for each k. In contrast, our current approach shifts to more efficient strategies as the problem size increases. Two rounds of A-BPE expand the vocabulary to 107 tokens, which include parts of existing and novel algorithms such as 2SWAP, 3FLIP, FOR(GRAD->LSA), GRAD->LSA->FLIP, LSA->GRAD, NE->LSA.

7.7.1.1 Improving algorithm hyperparameters using CLP

In this experiment, we optimize the learning rate of the Frank-Wolfe algorithm. Recall that the FW algorithm for QAP iteratively updates:

$$P_{k+1} = (1 - \gamma_k)P_k + \gamma_k \,\mathrm{LSA}(\nabla \mathcal{L}(P_k)),$$

where γ_k is a tunable step-size that is typically set to $\frac{2}{2+k}$. Using a line search [344] at each step yields a similar performance and decay in the learning rate. In order to optimize this schedule, we discretize the interval $[0, 1]$ with 20 equidistant values. We define our computational tokens to be the 20 different choices of learning rate. Using CLP, we optimize the chaining of these tokens, creating learning rate schedules that are tailored to each instance.

Automatic selection via CLP reveals that cyclical step-size schedules (as in Fig. 7.6) yield improved performance and an ability to reach the global optimum in tested instances. Notably, this phenomenon mirrors cyclical learning-rate strategies in neural networks, which achieve better accuracy and faster convergence without extensive tuning [312].

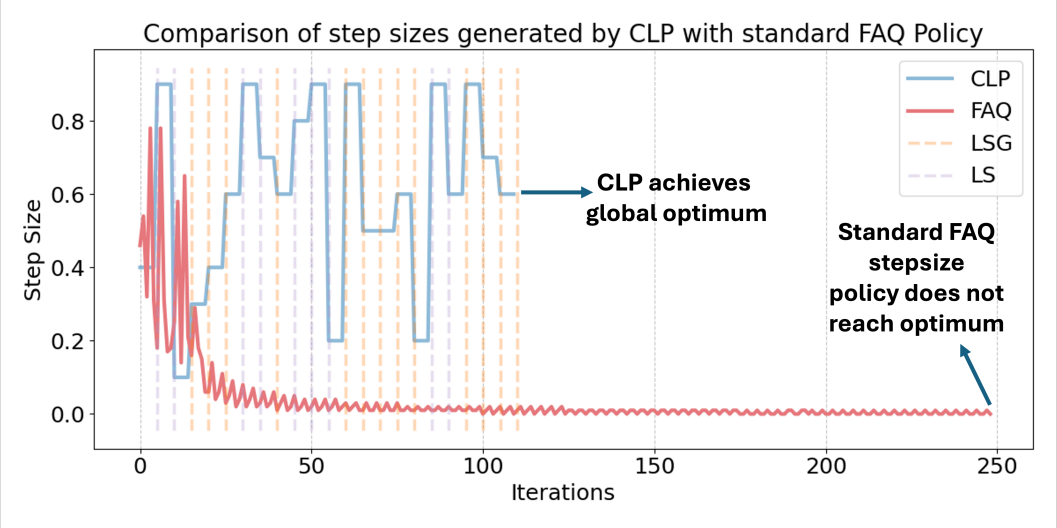


Figure 7.6: **Optimal step sizes for the Fast Approximate Quadratic assignment problem (FAQ) algorithm generated by our Computational Language Processing (CLP) approach compared to the standard step size schedule implemented in `SciPy`**. CLP discovers a cyclic rate (blue) with interleaving of Linear Sum assignments (LS) and Linear Sum assignments on the Gradient (LSG), and reaches global optimum unlike the standard line search-based approach used in `SciPy` (red)

7.7.2 Reaching state-of-the-art performance using high-level primitives and CLP

We define the high-level primitives as existing heuristics used to solve the QAP. They are simulated annealing, Frank-Wolfe (also known as Fast Approximate QAP) with two variants to project from doubly stochastic matrices to permutations, 2OPT, 3OPT and two variants of orthogonal optimization.

A pair of models, a policy network and a value network, is trained on examples generated from CQAP and PQAP, of varying size from 10 to 80, and evaluated on QAPLIB, with problems ranging in size from 10 to 256. We compare our results to simply using Simulated Annealing for many more iterations. This is a simple baseline, which is known to converge to the true solution if given an exponentially large number of steps [258]. We also code a simple branch-and-bound strategy, using the Gilmore-Lawler bound, with a limit on the total time spent. Finally, for a stronger comparison, we use the commercial Gurobi solver by interpreting the QAP as a mixed-integer quadratic problem, and setting the runtime to be the same as our method. The CLP runtime (at generation stage) includes the discovery of the instance-adapted algorithm and its execution to produce the final candidate solution. All examples up to size 80 run in under 200s (and most run in real-time), with the longest run taking 17 minutes for a QAPLIB problem of size 150. Note that

algorithm generation is performed in batches, so the reported durations correspond to batch-averaged times. All test runs were performed on a Mac Studio with M3 Ultra chip.

Composite QAP (CQAP) Generator. The *composite* QAP (CQAP) generator [91] produces problems with varying structural complexity, facilitating robust evaluation of algorithmic performance. Our framework achieves optimality in all tested instances. The instance-adapted algorithms generated vary in the primitives used, their order, and their length. For example, for two QAP instances of size 80, one is solved using the sequence (FW-LS -> FW -> OC -> SA -> FW-LSG -> 3PO -> STOP), while another used (2O -> FW -> OC -> SA -> FW-LSG -> FW-LSG -> FW-LS -> FW-LS -> OC -> STOP). In comparison, other methods obtain nonoptimal results, with a gap that widens as the size of the problem increases. Observing no gap between training and testing performance, we can conclude our framework has learned effective strategies for the whole CQAP class of problems.

Palubeckis QAP (PQAP) Generator. The Palubeckis QAP (PQAP) generator [247] represents a sophisticated method for crafting challenging QAP instances with precisely known optimal solutions. Through Hamming distance analysis, we observed that the PQAP optimization landscape is notably more challenging than CQAP. It features steep and narrow valleys around global optima and a high density of local optima, complicating heuristic search efforts significantly.
Despite this difficulty, our method achieves a 1% optimality gap in all tested instances, and beats the other methods in all examples. Once again, this performance is consistent with training performance, highlighting the capability of our method to generalize to other instances in the training distribution.

Benchmarking on QAPLIB. We evaluate our method using the widely recognized QAPLIB [48] library, known for its challenging benchmark problems and completely unseen in the training of our method. CLP consistently achieves state-of-the-art performance, achieving the best-known objective values across multiple tested instances. It beats or equals other methods in all instances. We also note that global optimum is achieved for problems of size up to 100. This shows the excellent generalization of our method to unseen problems, with potentially different structure from the CQAP and PQAP generators used in training. Detailed comparative results are provided in Table F.1.

7.7.3 Using CLP to manually create algorithms

While we present a method for automated algorithm discovery, CLP can be a useful framework for manually designing algorithms. In the process of developing our framework, we went through the steps of tokenizing, complexifying and improving by trial and error an algorithm, giving us valuable insight on how to automate this process. It resulted in an algorithm capable of achieving state-of-the-art performance with minimal computational resources when compared to other heuristics. While this algorithm was ultimately not implemented in automated discovery, it gives valuable insight on how high-level complexification within CLP is abstracted away from direct code-generation. Consequently, it fundamentally differs from approaches [182, 192, 227, 278].

Starting from the Frank-Wolfe algorithm, which is a gradient descent in the space of doubly stochastic matrices, we observed that projecting the final doubly stochastic matrix to a permutation could prove to be difficult. While using L_2 projection or the permutation of steepest descent gives good results, taking the best of the two improves results most. This yielded the best of both token. The gradient descent would often get stuck in local minima, as the optimization is not necessarily convex. To address this, we introduce a biasing token that adds a repulsive penalty to the loss. Specifically, given the current doubly stochastic matrix x_0 and a biasing matrix T, the biased loss is $\mathcal{L}_B(x) = \mathcal{L}(x) + \gamma \operatorname{Tr}(x^T T)$. This allows one to push the gradient descent away from the current solution. To introduce some variety in the exploration, we introduced another operator primitive similar to the parallel states primitive in 7.7.1. From a permutation x, take 10 uniform random permutations, x_i, and 10 random scalars $U_i \sim \mathcal{U}([0, 1])$. Letting $y_i = P(U_i x_i + (1 - U_i)x)$ for P an arbitrary token acting on doubly stochastic matrices, this parallel state with perturbation token picks $\arg\min \mathcal{L}(y_i)$ as its output. This is similar to implementing a deflation technique to discover new solutions [98]. This variety allows us to compute a Gibbs average, $y^\beta = \frac{\sum_i y_i e^{-\beta \mathcal{L}(y_i)}}{\sum_i e^{-\beta \mathcal{L}(y_i)}}$. It extracts information about the most likely values for the true optimum on average, and is used to define the bias token, $T = y^\beta \odot (1 - y^\beta) \odot x_0$ (writing $\odot$ for the entrywise product).

Having complexified our vocabulary with these new tokens, we manually crafted a highly performant algorithm that solved all tested instances of the CQAP generator up to size 160. This early success motivated us to extend and automate this approach, using the CLP framework.

7.8 Discovering a new quantum circuit for Grover's algorithm with lower depth using CLP

The `GroverGame` initializes with n qubits and four primitive operations. An oracle randomly selects a target state from the 2^n possible states, with all states permuted each run to ensure generalization across arbitrary targets. The iteration limit is set to Grover's theoretical bound, $\lceil \frac{\pi}{4}\sqrt{N} \rceil$, preventing the discovery of algorithms that exceed it. The game terminates either in *success* (+1), when a gate sequence identifies the target state within a tolerance of 3×10^{-2}, or in *failure* (-1), when the maximum allowed sequence length is reached without success.

Upon termination, training data for the neural network is generated at each step, consisting of an observation tensor (quantum states), the overlap between current and target states, chosen primitives, and a modified reward value. To improve learning efficiency and encourage minimal circuit depth, raw rewards (± 1) are adjusted: for successful cases, shorter gate sequences are rewarded using $v = 1 - \frac{\text{len(sequence)}}{\text{max\_iterations}}$, whereas failures receive partial credit based on correctly identified targets (states), calculated as $v = -1 + \frac{\#\,\text{targets achieved}}{N}$. This modification incentivizes efficiency and mitigates sparse-reward problems common in reinforcement learning. Early-stage training faces sparse positive examples, leading to batch balancing of positive and negative samples (approximately 50% each) via sampling with replacement when necessary. This ensures effective learning of rare successful gate sequences.

Analysis. We now demonstrate that our optimized gate sequence shown in Fig. 7.3 is mathematically equivalent to Grover's algorithm but more efficient in gate usage.

First, we summarize the *standard Grover's algorithm*. Start from the state $|\mathbf{0}\rangle^{\otimes n}$ and apply Hadamard gates to achieve the uniform superposition state

$$|\boldsymbol{s}\rangle = H^{\otimes n}|\mathbf{0}\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |\boldsymbol{x}\rangle.$$

Then, for $O(\sqrt{N})$ iterations, repeat two operations: (i) apply the oracle U_f that marks the target state $|\boldsymbol{w}\rangle$ by flipping its phase, and (ii) apply the diffusion operator $D = 2|\boldsymbol{s}\rangle\langle\boldsymbol{s}| - I$. Analyzing one iteration, after applying the oracle U_f, we have:

$$U_f|\boldsymbol{s}\rangle = |\boldsymbol{s}\rangle - \frac{2}{\sqrt{N}}|\boldsymbol{w}\rangle.$$

Next, applying the diffusion operator gives:

$$DU_f|\boldsymbol{s}\rangle = 2|\boldsymbol{s}\rangle\langle\boldsymbol{s}| \left(|\boldsymbol{s}\rangle - \frac{2}{\sqrt{N}}|\boldsymbol{w}\rangle\right) - \left(|\boldsymbol{s}\rangle - \frac{2}{\sqrt{N}}|\boldsymbol{w}\rangle\right).$$

Noting that $\langle \boldsymbol{s}|\boldsymbol{w}\rangle = 1/\sqrt{N}$, this simplifies to:

$$DU_f|\boldsymbol{s}\rangle = \left(1 - \frac{4}{N}\right)|\boldsymbol{s}\rangle + \frac{2}{\sqrt{N}}|\boldsymbol{w}\rangle.$$

The *optimized Grover's algorithm* proceeds as follows. Starting from $|\mathbf{0}\rangle^{\otimes n}$, apply X gates to all qubits, resulting in the state $|\mathbf{1}\rangle^{\otimes n}$. Next, apply Hadamard gates to form a modified superposition:

$$|\boldsymbol{s'}\rangle = H^{\otimes n}|\mathbf{1}\rangle^{\otimes n} = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}(-1)^{|\boldsymbol{x}|}|\boldsymbol{x}\rangle,$$

where $|\boldsymbol{x}|$ is the Hamming weight (number of ones) in the binary representation of $\boldsymbol{x}$. Then, iteratively apply the oracle U_f to mark the target state and use the modified diffusion operator:

$$D' = 2|\boldsymbol{s'}\rangle\langle\boldsymbol{s'}| - I,$$

repeating these steps for $O(\sqrt{N})$ iterations.

We now analyze the optimized circuit's effect on the initial state $|\mathbf{0}\rangle^{\otimes n}$. Applying X gates transforms it to $|\mathbf{1}\rangle^{\otimes n}$, followed by Hadamard gates yielding the modified superposition:

$$|\boldsymbol{s'}\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}(-1)^{|\boldsymbol{x}|}|\boldsymbol{x}\rangle.$$

The oracle U_f acts similarly as before, marking the target state $|\boldsymbol{w}\rangle$:

$$U_f|\boldsymbol{s'}\rangle = |\boldsymbol{s'}\rangle - \frac{2}{\sqrt{N}}(-1)^{|\boldsymbol{w}|}|\boldsymbol{w}\rangle.$$

The modified diffusion operator is defined as:

$$D' = 2|\boldsymbol{s'}\rangle\langle\boldsymbol{s'}| - I = H^{\otimes n}(2|\mathbf{1}\rangle\langle\mathbf{1}| - I)H^{\otimes n},$$

which can be efficiently implemented using the gate sequence $H^{\otimes n} \rightarrow \mathrm{MCZ} \rightarrow H^{\otimes n}$, as illustrated in Fig. 7.3. Applying D' after U_f gives:

$$D'U_f|\boldsymbol{s'}\rangle = \left(1 - \frac{4}{N}\right)|\boldsymbol{s'}\rangle + \frac{2}{\sqrt{N}}(-1)^{|\boldsymbol{w}|}|\boldsymbol{w}\rangle.$$

Notably, this result differs from standard Grover's algorithm by only a global phase factor $(-1)^{|w|}$, which does not affect measurement probabilities. Therefore, our optimized circuit remains mathematically equivalent. The crucial advantage is that initializing with $|\mathbf{1}\rangle^{\otimes n}$ directly simplifies the diffusion operator, removing the intermediate X gates required by the standard implementation.

Comparison to Related Work. Gilliam et al. [118] reduce circuit depth in Grover's algorithm by replacing Hadamard gates with $\mathrm{R}_X(\pi/2)$ rotations, eliminating extra X gates (see Table 7.1). However, we note that R_X gates are challenging to implement on real-world hardware platforms. Our circuits, on the other hand, are more practical. For example, our circuits are easily implementable on photonic-based quantum computing solutions by removing the need for R_X gates. Our approach achieves the same depth reduction using only H, X and MCZ gates, preserving compatibility with Clifford-based frameworks and providing a more conventional construction. Wu et al. [353] and Zhang and Korepin [372] propose multistage variants that reduce depth via hierarchical partitioning and local diffusion operators. Piron et al. [261] introduce a hybrid strategy that reduces oracle calls by combining fewer Grover iterations with bounded classical retries. In contrast, our method avoids block partitioning and classical feedback, and achieves depth reduction via a simple basis transformation. We also note that in future work, we plan to include hardware implementation details to design circuits that are tailored to the quantum platform of interest.

Table 7.1: **Comparison of Grover Optimization Strategies**

Method	**Diffusion operator**	**Approximation?**	**Hybrid?**	**Depth reduction**
Ours	$H \cdot \mathrm{MCZ} \cdot H$	No	No	2× fewer 1-qubit gates
Gilliam et al. [118]	$R_X \cdot \mathrm{MCZ} \cdot R_X$	No	No	2× fewer gates per iteration
Wu et al. [353]	Hierarchical	No	Yes (2-stage)	1.2× depth reduction
Zhang & Korepin [372]	Local diffusion	Yes	No	20% empirical reduction
Piron et al. [261]	Standard	No	Yes	10% reduction

7.9 CLP-based QAOA results

The Quantum Approximate Optimization Algorithm (QAOA) [97] is a hybrid quantum-classical optimization algorithm designed to approximately solve combinatorial optimization problems formulated as a bit-string search $z \in \{0,1\}^n$ that minimizes a given cost function $C(z)$. The algorithm was designed for the current generation of noisy quantum platforms with the aim of demonstrating their utility. The approach has been applied to problems in graph theory, supply chain optimiza-

tion, task scheduling, and energy management to name a few. By encoding the cost function in a Hamiltonian form: $H_C = \sum_{z\in\{0,1\}^n} C(z)|z\rangle\langle z|$., the algorithm approximates solutions by preparing and optimizing parameterized quantum states $|\psi(\gamma,\beta)\rangle$, which depend on adjustable parameters (γ,β). These parameters are tuned using classical optimization routines such as gradient descent. These quantum states are implemented on quantum devices using quantum circuits that consist of alternating layers: a *cost Hamiltonian layer* ($e^{-i\gamma H_C}$), that encodes the cost function, and a *mixer Hamiltonian layer* ($e^{-i\beta H_M}$), that are designed to facilitate exploration of the energy landscape. Therefore, one can map the initial state $|+\rangle^{\otimes n}$ to the final quantum state using: $|\psi(\gamma,\beta)\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \ldots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C} |+\rangle^{\otimes n}$. As mentioned previously, the optimal parameters (γ_i,β_i) are computed by classically minimizing the expectation of the cost Hamiltonian:

$$(\gamma^*,\beta^*) = \arg\min_{\gamma,\beta} \langle\psi(\gamma,\beta)|H_C|\psi(\gamma,\beta)\rangle.$$

Results on the Quantum Approximate Optimization Algorithm (QAOA) . An important challenge within the QAOA setting is the design of shallow mixer Hamiltonian quantum circuits to minimize the optimization cost. We focus our efforts on using our CLP framework for designing the mixer Hamiltonian circuits for QAOA. We compare our approach with an adaptive version of QAOA known as ADAPT-QAOA [381]. The approach is a standardized solution implemented within NVIDIA's quantum computing library (CUDA-Q). ADAPT-QAOA is an iterative classical-quantum algorithm that adaptively builds problem-specific mixer Hamiltonians for solving a given combinatorial optimization problem. As shown in Fig. 7.7 (A), unlike standard implementations of QAOA that use the *same* mixer Hamiltonian at each step, ADAPT-QAOA iteratively selects operators from a pool of candidate gates based on their gradient contributions to the cost function. The mixer producing the *steepest descent* at each step is selected and appended to the circuit ansatz. All parameters (γ,β) of the cost and mixer Hamiltonians are then optimized classically using variational quantum eigensolver (VQE). The process continues until either a convergence criterion is met or the user-prescribed maximum circuit depth is reached.

Our CLP-based approach significantly improves the ADAPT-QAOA approach by learning to propose the optimal problem-specific mixer Hamiltonian circuit at each step. Unlike ADAPT-QAOA that uses steepest descent to iteratively add successive gates to the circuit ansatz, our CLP-based QAOA framework, as depicted in Fig. 7.7

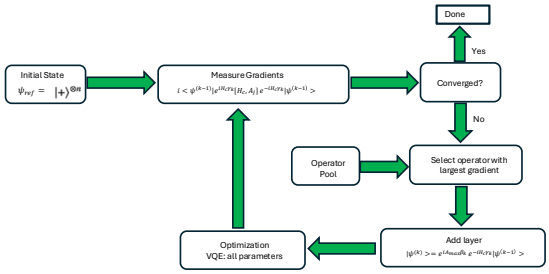
(B), simultaneously considers entire chains of gates for the mixer circuits by using MCTS and reinforcement learning to implement a *lookahead* policy.

To test our approach, we use the CLP framework to efficiently construct the mixer Hamiltonian circuits for MAX-CUT instances on p-regular graphs, with the goal of *reducing the optimization cost for a user-prescribed maximum circuit depth*. Our primitives consist of the same gates used with the ADAPT-QAOA framework for generating circuits [381] . Namely, we use the standard QAOA mixer pool [381] that consists of $\sum_{i\in Q}\{X_i\}$, single-qubit $\cup_{i\in Q}\{X_i\}$, and multi-qubit entangling gates $\cup_{i,j\in Q\times Q}\{B_iC_j|B_i, C_j \in \{X, Y, Z\}\}$, where Q is the set of qubits. We note that only Pauli strings that have an even number of Y or Z gates are retained in our mixer pool [381]. We generate 1000 p-regular graphs for the training of our neural networks and limit the depth of the mixer circuits. We then test the CLP framework on 21 graphs that were not seen in training. The CLP framework outperforms ADAPT-QAOA in 100% of test cases and corresponds to an average improvement of 34.62%. The results are tabulated in Table 7.2.

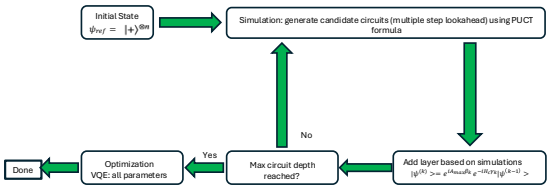
Table 7.2: **Performance Comparison**: CLP-QAOA vs. ADAPT-QAOA and % improvement

Circuit	1	2	3	4	5	6	7	8	9	10	11
CLP	-2.8	-2.6	-3.2	-4.1	-3.7	-2.3	-2.8	-2.4	-3.9	-2.9	-2.4
ADAPT	-2.1	-2.1	-2.2	-3.2	-2.5	-1.5	-2.0	-2.1	-2.6	-2.0	-2.1
% Improve	36	23	44	30	45	49	41	16	51	44	17

Circuit	12	13	14	15	16	17	18	19	20	21
CLP	-2.8	-3.3	-2.9	-3.5	-3.3	-3.4	-2.6	-3.3	-4.1	-3.3
ADAPT	-2.3	-2.4	-2.2	-2.6	-2.6	-2.4	-2.1	-2.7	-2.5	-2.4
% Improve	19	34	35	35	28	38	21	23	61	38



(A) ADAPT-QAOA workflow



(B) CLP-based QAOA workflow

Figure 7.7: **Comparison of workflows for generating quantum circuits for Quantum Approximate Optimization Algorithm (QAOA):** (**A**) ADAPT-QAOA, (**B**) our Computational Language Processing (CLP)-based approach.

BIBLIOGRAPHY

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[2] Nathanael L Ackerman, Cameron E Freer, and Daniel M Roy. On computability and disintegration. *Mathematical Structures in Computer Science*, 27(8): 1287–1314, 2017.

[3] Joshua Agterberg, Zachary Lubberts, and Carey E Priebe. Entrywise estimation of singular vectors of low-rank matrices with heteroskedasticity and dependence. *IEEE Transactions on Information Theory*, 68(7):4618–4650, 2022.

[4] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://arxiv.org/abs/1907.10902.

[5] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic Interpolants: A Unifying Framework for Flows and Diffusions, March 2023. URL http://arxiv.org/abs/2303.08797. arXiv:2303.08797 [cond-mat].

[6] Andrea Alfarano, Luca Maiano, Lorenzo Papa, and Irene Amerini. Estimating optical flow: A comprehensive review of the state of the art. *Computer Vision and Image Understanding*, 249:104160, December 2024. ISSN 1077-3142. doi: 10.1016/j.cviu.2024.104160. URL https://www.sciencedirect.com/science/article/pii/S1077314224002418.

[7] Robert Altmann, Patrick Henning, and Daniel Peterseim. Numerical homogenization beyond scale separation. *Acta Numerica*, 30:1–86, 2021.

[8] Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

[9] Charlotte Møller Andersen and Rasmus Bro. Variable selection in regression—a tutorial. *Journal of Chemometrics*, 24(11-12):728–737, 2010.

[10] Xing Anhao, Zhang Pengyuan, Pan Jielin, and Yan Yonghong. Svd-based dnn pruning and retraining. *Journal of Tsinghua University (Science and Technology)*, 56(7):772–776, 2016.

[11] Anthropic. Introducing Anthropic's AI for Science Program, 2025. URL https://www.anthropic.com/news/ai-for-science-program.

[12] David L Applegate. *The traveling salesman problem: a computational study*, volume 17. Princeton University Press, 2006.

[13] Aleksandr Y Aravkin, Robert Baraldi, and Dominique Orban. A Levenberg–Marquardt method for nonsmooth regularized least squares. *SIAM Journal on Scientific Computing*, 46(4):A2557–A2581, 2024.

[14] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning, June 2025. URL http://arxiv.org/abs/2506.09985. arXiv:2506.09985 [cs.AI].

[15] Žiga Avsec, Natasha Latysheva, Jun Cheng, Guido Novati, Kyle R. Taylor, Tom Ward, Clare Bycroft, Lauren Nicolaisen, Eirini Arvaniti, Joshua Pan, Raina Thomas, Vincent Dutordoir, Matteo Perino, Soham De, Alexander Karollus, Adam Gayoso, Toby Sargeant, Anne Mottram, Lai Hong Wong, Pavol Drotár, Adam Kosiorek, Andrew Senior, Richard Tanburn, Taylor Applebaum, Souradeep Basu, Demis Hassabis, and Pushmeet Kohli. Advancing regulatory variant effect prediction with AlphaGenome. *Nature*, 649(8099): 1206–1218, January 2026. ISSN 1476-4687. doi: 10.1038/s41586-025-10014-0. URL https://www.nature.com/articles/s41586-025-10014-0.

[16] Eser Aygün, Anastasiya Belyaeva, Gheorghe Comanici, Marc Coram, Hao Cui, Jake Garrison, Renee Johnston Anton Kast, Cory Y. McLean, Peter Norgaard, Zahra Shamsi, David Smalling, James Thompson, Subhashini Venugopalan, Brian P. Williams, Chujun He, Sarah Martinson, Martyna Plomecka, Lai Wei, Yuchen Zhou, Qian-Ze Zhu, Matthew Abraham, Erica Brand, Anna Bulanova, Jeffrey A. Cardille, Chris Co, Scott Ellsworth, Grace Joseph, Malcolm Kane, Ryan Krueger, Johan Kartiwa, Dan Liebling, Jan-Matthis Lueckmann, Paul Raccuglia, Xuefei, Wang, Katherine Chou, James Manyika, Yossi Matias, John C. Platt, Lizzie Dorfman, Shibl Mourad, and Michael P. Brenner. An AI system to help scientists write expert-level empirical software, September 2025. URL http://arxiv.org/abs/2509.06503. arXiv:2509.06503 [cs].

[17] Jinho Baik, Gérard Ben Arous, and Sandrine Péché. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. *Annals of Probability*, 33(5):1643–1697, 2005.

[18] Hamed Hamze Bajgiran and Houman Owhadi. Aggregation of Pareto optimal models, December 2021. URL http://arxiv.org/abs/2112.04161. arXiv:2112.04161 [econ, math, stat].

[19] Zhigang Bao and Dong Wang. Eigenvector distribution in the critical regime of bbp transition. *Probability Theory and Related Fields*, 182(1-2):399–479, 2022.

[20] Zhigang Bao, Xiucai Ding, and Ke Wang. Singular vector and singular subspace distribution for the matrix denoising model. *Annals of Statistics*, 49 (2):751–779, 2021.

[21] Ricardo Baptista, Youssef Marzouk, Rebecca E Morrison, and Olivier Zahm. Learning non-gaussian graphical models via hessian scores and triangular transport. *arXiv preprint arXiv:2101.03093*, 2021.

[22] Ricardo Baptista, Rebecca Morrison, Olivier Zahm, and Youssef Marzouk. Learning non-Gaussian graphical models via Hessian scores and triangular transport. *Journal of Machine Learning Research*, 25(85):1–46, 2024.

[23] Francesco Bartolucci, Andrea Bosco, Martina Bodini, Maria Colombo, Leo Hollenstein, Siddhartha Mishra, Roberto Molinaro, Jan Peszek, and Panagiotis E. Souganidis. Convolutional neural operators. In *Advances in Neural Information Processing Systems*, 2024. NeurIPS; official metadata available via the NeurIPS proceedings.

[24] Klaus-Jürgen Bathe. *Finite Element Method*, pages 1–12. John Wiley & Sons, Ltd, 2008. ISBN 9780470050118. doi: https://doi.org/10.1002/9780470050118.ecse159. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470050118.ecse159.

[25] Pau Batlle, Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Error analysis of kernel/gp methods for nonlinear and parametric pdes. *arXiv preprint arXiv:2305.04962*, 2023.

[26] Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owhadi. Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 2023.

[27] Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owhadi. Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 496:112549, 2024.

[28] Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owhadi. Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 496:112549, 2024.

[29] Dilyara Baymurzina, Eugene Golikov, and Mikhail Burtsev. A review of neural architecture search. *Neurocomputing*, 474:82–93, February 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2021.12.014. URL https://www.sciencedirect.com/science/article/pii/S0925231221018439.

[30] Florent Benaych-Georges and Raj Rao Nadakuditi. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *Advances in Mathematics*, 227(1):494–521, 2011.

[31] Florent Benaych-Georges and Raj Rao Nadakuditi. The singular values and vectors of low rank perturbations of large rectangular random matrices. *Journal of Multivariate Analysis*, 111:120–135, 2012.

[32] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967, March 2021. ISSN 1573-7462. doi: 10.1007/s10462-020-09896-5. URL https://doi.org/10.1007/s10462-020-09896-5.

[33] Leonid Berlyand, Pierre-Emmanuel Jabin, and C Alex Safsten. Stability for the training of deep neural networks and other classifiers. *Mathematical Models and Methods in Applied Sciences*, 31(11):2345–2390, 2021.

[34] Leonid Berlyand, Etienne Sandier, Yitzchak Shmalo, and Lei Zhang. Enhancing accuracy in deep learning using random matrix theory. *arXiv preprint arXiv:2310.03165*, 2023.

[35] Patrick Billingsley. *Convergence of probability measures*. Wiley series in probability and statistics. Probability and statistics section. Wiley, New York, 2nd ed edition, 1999. ISBN 978-0-471-19745-4.

[36] Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. $o(n^{2.7799})$ complexity for matrix multiplication. *Information Processing Letters*, 8(5): 234–235, 1979.

[37] Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Vaughan, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. A foundation model for the earth system. *Nature*, 641(8023):1180–1187, 2025. doi: 10.1038/s41586-025-09005-y.

[38] Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38, 2025.

[39] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[40] Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. In *Handbook of Numerical Analysis*, volume 25, pages 83–125. Elsevier, 2024.

[41] Théo Bourdais, Pau Batlle, Xianjin Yang, Ricardo Baptista, Nicolas Rouquette, and Houman Owhadi. Codiscovering graphical structure and functional relationships within data: A gaussian process framework for connecting the dots. *Proceedings of the National Academy of Sciences*, 121(32): e2403449121, 2024.

[42] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax, 2018.

[43] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

[44] Rasmus Bro and Age K Smilde. Principal component analysis. *Analytical methods*, 6(9):2812–2831, 2014.

[45] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020. URL http://arxiv.org/abs/2005.14165. arXiv:2005.14165 [cs] version: 4.

[46] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[47] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[48] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. Qaplib–a quadratic assignment problem library. *Journal of Global Optimization*, 10:391–403, 1997.

[49] Chenghao Cai, Dengfeng Ke, Yanyan Xu, and Kaile Su. Fast learning of deep neural networks via singular value decomposition. In *Pacific Rim International Conference on Artificial Intelligence*, pages 820–826. Springer, 2014.

[50] Dario Caldara and Matteo Iacoviello. Measuring geopolitical risk. *American Economic Review*, 112(4):1194–1225, 2022.

[51] Edoardo Calvello, Nikola B. Kovachki, Matthew E. Levine, and Andrew M. Stuart. Continuum attention for neural operators. *Journal of Machine Learning Research*, 26(300):1–52, 2025.

[52] Shuhao Cao. Choose a transformer: Fourier or Galerkin. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24924–24940. Curran Associates, Inc., 2021.

[53] Patrick Chatain, Michael Rizvi-Martel, Guillaume Rabusseau, and Adam Oberman. Numerical PDE solvers outperform neural PDE solvers, July 2025. URL http://arxiv.org/abs/2507.21269. arXiv:2507.21269 [math.NA].

[54] Jiong Chen, Florian Schaefer, and Mathieu Desbrun. Lightning-fast method of fundamental solutions. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi: 10.1145/3658199. URL https://doi.org/10.1145/3658199.

[55] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Solving and learning nonlinear pdes with gaussian processes. *Journal of Computational Physics*, 447:110668, 2021.

[56] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M. Stuart. Solving and Learning Nonlinear PDEs with Gaussian Processes, August 2021. URL http://arxiv.org/abs/2103.12959. arXiv:2103.12959 [cs, math, stat].

[57] Yifan Chen, Ethan N Epperly, Joel A Tropp, and Robert J Webber. Randomly pivoted cholesky: Practical approximation of a kernel matrix with few entry evaluations. *Communications on Pure and Applied Mathematics*, 2023.

[58] Yifan Chen, Houman Owhadi, and Florian Schäfer. Sparse cholesky factorization for solving nonlinear pdes via gaussian processes. *arXiv preprint arXiv:2304.01294*, 2023.

[59] Yifan Chen, Houman Owhadi, and Florian Schäfer. Sparse cholesky factorization for solving nonlinear pdes via gaussian processes. *Mathematics of Computation*, 2024.

[60] Yuxin Chen, Chen Cheng, and Jianqing Fan. Asymmetry helps: Eigenvalue and eigenvector analyses of asymmetrically perturbed low-rank matrices. *Annals of statistics*, 49(1):435, 2021.

[61] Guangran Cheng, Chuheng Zhang, Wenzhe Cai, Li Zhao, Changyin Sun, and Jiang Bian. Empowering Large Language Models on Robotic Manipulation with Affordance Prompting, April 2024. URL http://arxiv.org/abs/2404.11027. arXiv:2404.11027 [cs.AI].

[62] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*, 2023.

[63] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[64] Alexandre J. Chorin, Ole H. Hald, and Raz Kupferman. Optimal prediction with memory. *Physica D: Nonlinear Phenomena*, 166(3-4):239–257, 2002.

[65] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204. PMLR, 2015.

[66] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

[67] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. John Wiley & Sons, 1998. ISBN 9780471558941.

[68] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[69] Romain Couillet and Merouane Debbah. *Random matrix methods for wireless communications*. Cambridge University Press, 2011.

[70] Romain Couillet and Zhenyu Liao. *Random Matrix Methods for Machine Learning*. Cambridge University Press, 2022.

[71] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Hoboken, NJ, USA, 2nd edition, 2006.

[72] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[73] Quan Dao, Hao Phung, Trung Tuan Dao, Dimitris N. Metaxas, and Anh Tran. Self-corrected flow distillation for consistent one-step and few-step image generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 2654–2662, 2025.

[74] Matthieu Darcy, Boumediene Hamzi, Giulia Livieri, Houman Owhadi, and Peyman Tavallali. One-shot learning of stochastic differential equations with data adapted kernels. *Physica D: Nonlinear Phenomena*, 444:133583, 2023.

[75] MIT Critical Data, Justin D Salciccioli, Yves Crutain, Matthieu Komorowski, and Dominic C Marshall. Sensitivity analysis and model validation. *Secondary analysis of electronic health records*, pages 263–271, 2016.

[76] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems*, 27, 2014.

[77] Anna Dawid and Yann LeCun. Introduction to Latent Variable Energy-Based Models: A Path Towards Autonomous Machine Intelligence. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104011, October 2024. ISSN 1742-5468. doi: 10.1088/1742-5468/ad292b. URL http://arxiv.org/abs/2306.02572. arXiv:2306.02572 [cs.LG].

[78] Tim De Ryck and Siddhartha Mishra. Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning. *Acta Numerica*, 33:633–713, 2024. doi: 10.1017/S0962492923000089.

[79] Google DeepMind. Science, 2025. URL https://deepmind.google/science/.

[80] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong

Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature*, 645(8081):633–638, September 2025. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://arxiv.org/abs/2501.12948. arXiv:2501.12948 [cs.CL].

[81] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982. doi: 10.1137/0719025. URL https://doi.org/10.1137/0719025.

[82] Chengyuan Deng, Shihang Feng, Hanchen Wang, Xitong Zhang, Peng Jin, Yinan Feng, Qili Zeng, Yinpeng Chen, and Youzuo Lin. Openfwi: Large-scale multi-structural benchmark datasets for full waveform inversion. *Advances in Neural Information Processing Systems*, 35:6007–6020, 2022.

[83] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[84] Shujie Deng, Honghua Dong, and Xujie Si. Enhancing and Evaluating Logical Reasoning Abilities of Large Language Models, April 2024. URL https://openreview.net/forum?id=xw06d8NQAd&utm_source=chatgpt.com.

[85] Loann David Denis Desboulets. A review on variable selection in regression analysis. *Econometrics*, 6(4):45, 2018.

[86] Prathapasinghe Dharmawansa, Pasan Dissanayake, and Yang Chen. The eigenvectors of single-spiked complex wishart matrices: Finite and asymptotic analyses. *IEEE Transactions on Information Theory*, 68(12):8092–8120, 2022.

[87] Paul A Dirmeyer, Pierre Gentine, Michael B Ek, and Gianpaolo Balsamo. Land surface processes relevant to sub-seasonal to seasonal (s2s) prediction. In *Sub-Seasonal to Seasonal Prediction*, pages 165–181. Elsevier, 2019.

[88] Alireza Doostan and Houman Owhadi. A non-adapted sparse approximation of pdes with stochastic inputs. *Journal of Computational Physics*, 230(8): 3015–3034, 2011.

[89] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[90] Mathias Drton and Marloes H Maathuis. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393, 2017.

[91] Mădălina M Drugan. Generating qap instances with known optimum solution and additively decomposable cost function. *Journal of Combinatorial Optimization*, 30:1138–1172, 2015.

[92] Elizabeth E. Ebert. Ability of a Poor Man's Ensemble to Predict the Probability and Distribution of Precipitation. *Monthly Weather Review*, 129(10):2461–2480, October 2001. ISSN 1520-0493, 0027-0644. doi: 10.1175/1520-0493(2001)129<2461:AOAPMS>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/mwre/129/10/1520-0493_2001_129_2461_aoapms_2.0.co_2.xml. Publisher: American Meteorological Society Section: Monthly Weather Review.

[93] Robert Engle. GARCH 101: The use of ARCH/GARCH models in applied econometrics. *Journal of Economic Perspectives*, 15(4):157–168, 2001.

[94] Gianluca Fabiani, Ioannis G. Kevrekidis, Constantinos Siettos, and Athanasios N. Yannacopoulos. Randonets: Shallow networks with random projections for learning linear and nonlinear operators. *Journal of Computational Physics*, 520:113433, 2025. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2024.113433. URL https://www.sciencedirect.com/science/article/pii/S0021999124006818.

[95] William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL https://github.com/Lightning-AI/lightning.

[96] Mohammad Farazmand and Themistoklis P. Sapsis. Extreme events: Mechanisms and prediction. *Applied Mechanics Reviews*, 71(5):050801, 2019.

[97] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[98] Patrick E Farrell, Casper HL Beentjes, and Ásgeir Birkisson. The computation of disconnected bifurcation diagrams. *arXiv preprint arXiv:1603.00809*, 2016.

[99] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering

faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

[100] Michael Feischl and Daniel Peterseim. Sparse compression of expected solution operators. *SIAM Journal on Numerical Analysis*, 58(6):3144–3164, 2020.

[101] Michael Feischl, Christoph Schwab, and F. Zehetgruber. Neural general operator networks via banach fixed point iterations. SAM Research Report 2025-13, Seminar for Applied Mathematics, ETH Zurich, April 2025. URL http://hdl.handle.net/20.500.11850/734105.

[102] Samuel G. Finlayson, Adarsh Subbaswamy, Karandeep Singh, John Bowers, Annabel Kupke, Jonathan Zittrain, Isaac S. Kohane, and Suchi Saria. The Clinician and Dataset Shift in Artificial Intelligence. *New England Journal of Medicine*, 385(3):283–286, July 2021. ISSN 0028-4793, 1533-4406. doi: 10.1056/NEJMc2104626. URL http://www.nejm.org/doi/10.1056/NEJMc2104626.

[103] Andreas Fischer, Alexey F Izmailov, and Mikhail V Solodov. The levenberg–marquardt method: an overview of modern convergence theories and more. *Computational Optimization and Applications*, 89(1):33–67, 2024.

[104] Gregory Flato, Jochem Marotzke, Babatunde Abiodun, Pascale Braconnot, Sin Chan Chou, William Collins, Peter Cox, Fatima Driouech, Seita Emori, Veronika Eyring, et al. Evaluation of climate models. In *Climate change 2013: the physical science basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, pages 741–866. Cambridge University Press, 2014.

[105] Bengt Fornberg and David M. Sloan. A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, 3:203–267, 1994. doi: 10.1017/S0962492900002440.

[106] ARC Prize Foundation. ARC-AGI-3: A New Challenge for Frontier Agentic Intelligence, April 2026. URL http://arxiv.org/abs/2603.24621. arXiv:2603.24621 [cs.AI].

[107] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[108] Karl J. Friston, Lee Harrison, and Will Penny. Dynamic causal modelling. *Neuroimage*, 19(4):1273–1302, 2003.

[109] Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

[110] Ting-Ting Gao, Baruch Barzel, and Gang Yan. Learning interpretable dynamics of stochastic complex systems from experimental data. *Nature Communications*, 15(1):6029, 2024.

[111] Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. Gpytorch: blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

[112] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, 1979. ISBN 0-7167-1044-7. URL https://dl.acm.org/doi/book/10.5555/578533.

[113] Henri P Gavin. The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems. *Department of Civil and Environmental Engineering Duke University August*, 3(1-23):4, 2019.

[114] Jungang Ge, Ying-Chang Liang, Zhidong Bai, and Guangming Pan. Large-dimensional random matrix theory and its applications in deep learning and wireless communications. *Random Matrices: Theory and Applications*, 10 (04):2230001, 2021.

[115] Kyriakos Georgiou, Constantinos Siettos, and Athanasios N. Yannacopoulos. Fredholm neural networks for forward and inverse problems in elliptic pdes. *arXiv preprint arXiv:2507.06038*, 2025.

[116] Kyriakos Georgiou, Constantinos Siettos, and Athanasios N. Yannacopoulos. Fredholm neural networks. *SIAM Journal on Scientific Computing*, 47(4): C1006–C1031, 2025. doi: 10.1137/24M1686991. URL https://doi.org/10.1137/24M1686991.

[117] Bernard J. Geurts, Darryl D. Holm, and Erwin Luesink. Lyapunov exponents of two stochastic Lorenz 63 systems. *Journal of Statistical Physics*, 179(5–6): 1343–1365, 2020.

[118] Austin Gilliam, Marco Pistoia, and Constantin Gonciulea. Optimizing quantum search using a generalized version of grover's algorithm. *arXiv preprint arXiv:2005.06468*, 2020.

[119] Jody Hoffer Gittell and Hebatallah Naim Ali. *Relational analytics: Guidelines for analysis and action*. Routledge, 2021.

[120] Dror Givon, Raz Kupferman, and Andrew Stuart. Extracting macroscopic dynamics: Model problems and algorithms. *Nonlinearity*, 17(6):R55–R127, 2004.

[121] Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[122] Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operator networks. In *Machine Learning in Modeling and Simulation: Methods and Applications*, pages 219–254. Springer, 2023.

[123] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR, 2020.

[124] Arthur Gretton, Kenji Fukumizu, Choon Teo, Le Song, Bernhard Schölkopf, and Alex Smola. A kernel statistical test of independence. *Advances in Neural Information Processing Systems*, 20, 2007.

[125] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, pages 85–113. Springer, 2016.

[126] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 212–219, 1996. doi: 10.1145/237814.237866.

[127] Mengyang Gu. Jointly robust prior for Gaussian stochastic process in emulation, calibration and variable selection. *Bayesian Analysis*, 14(3):857–885, 2019. doi: 10.1214/18-BA1133. URL https://doi.org/10.1214/18-BA1133.

[128] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking, January 2025. URL http://arxiv.org/abs/2501.04519. arXiv:2501.04519 [cs].

[129] Alfredo Guevara, Alexandru Lupsasca, David Skinner, Andrew Strominger, and Kevin Weil. Single-minus gluon tree amplitudes are nonzero, March 2026. URL http://arxiv.org/abs/2602.12176. arXiv:2602.12176 [hep-th].

[130] Erico Guizzo. Closing in on the perfect code. *IEEE Spectrum*, 41(3):36–42, March 2004. Feature on turbo codes.

[131] Yue Guo, Felix Dietrich, Tom Bertalan, Danimir T Doncevic, Manuel Dahmen, Ioannis G Kevrekidis, and Qianxiao Li. Personalized algorithm generation: A case study in learning ode integrators. *SIAM Journal on Scientific Computing*, 44(4):A1911–A1933, 2022.

[132] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[133] Eldad Haber. Quasi-newton methods for large-scale electromagnetic inverse problems. *Inverse problems*, 21(1):305, 2004.

[134] Eldad Haber, Uri M Ascher, and Doug Oldenburg. On optimization techniques for solving nonlinear inverseproblems. *Inverse problems*, 16(5):1263, 2000.

[135] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[136] Boumediene Hamzi and Houman Owhadi. Learning dynamical systems from data: A simple cross-validation perspective, part i: Parametric kernel flows. *Physica D: Nonlinear Phenomena*, 421:132817, 2021.

[137] Boumediene Hamzi, Romit Maulik, and Houman Owhadi. Simple, low-cost and accurate data-driven geophysical forecasting with learned kernels. *Proceedings of the Royal Society A*, 477(2252):20210326, 2021.

[138] Boumediene Hamzi, Houman Owhadi, and Yannis Kevrekidis. Learning dynamical systems from data: A simple cross-validation perspective, part iv: case with partial observations. *Physica D: Nonlinear Phenomena*, 454: 133853, 2023.

[139] Bruce E. Hansen. Leats Squares Model Averaging. *Econometrica*, 75(4): 1175–1189, 2007. ISSN 0012-9682. URL https://www.jstor.org/stable/450 2024. Publisher: [Wiley, Econometric Society].

[140] Wenrui Hao, Xinliang Liu, and Yahong Yang. Newton informed neural operator for solving nonlinear partial differential equations. *Advances in neural information processing systems*, 37:120832–120860, 2024.

[141] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[142] David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, March 1978. ISSN 0095-0696. doi: 10.1016/0095-0696(78)9 0006-2. URL https://www.sciencedirect.com/science/article/pii/00950696 78900062.

[143] Alex Havrilla and Wenjing Liao. Understanding Scaling Laws with Statistical and Approximation Theory for Transformer Neural Networks on Intrinsically Low-dimensional Data, November 2024. URL http://arxiv.org/abs/2411.066 46. arXiv:2411.06646 [cs].

[144] Juncai He, Xinliang Liu, and Jinchao Xu. MgNO: Efficient Parameterization of Linear Operators via Multigrid. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024. https://arxiv.org/abs/2310.19809.

[145] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2311.

[146] Desmond J. Higham. Modeling and simulating chemical reactions. *SIAM Review*, 50(2):347–368, 2008.

[147] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[148] Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart. The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks, August 2022. URL http://arxiv.org/abs/2203.13181. arXiv:2203.13181 [math.NA].

[149] Yichen Huang and Lin F. Yang. Winning Gold at IMO 2025 with a Model-Agnostic Verification-and-Refinement Pipeline, 2025. URL https://arxiv.org/abs/2507.15855. Version Number: 4.

[150] Toru Ishihara. Enumeration of hypergraphs. *European Journal of Combinatorics*, 22(4):503–509, 2001.

[151] Yasamin Jalalian, Juan Felipe Osorio Ramirez, Alexander Hsu, Bamdad Hosseini, and Houman Owhadi. Data-efficient kernel methods for learning differential equations and their solution operators: Algorithms and error analysis. *arXiv preprint arXiv:2503.01036*, 2025.

[152] Yasamin Jalalian, Juan Felipe Osorio Ramirez, Alexander Hsu, Bamdad Hosseini, and Houman Owhadi. Data-Efficient Kernel Methods for Learning Differential Equations and Their Solution Operators: Algorithms and Error Analysis, April 2025. URL http://arxiv.org/abs/2503.01036. arXiv:2503.01036 [stat.ML].

[153] Anas Jnini, Flavio Vella, and Marius Zeinhofer. Gauss-newton natural gradient descent for physics-informed computational fluid dynamics. *Computers & Fluids*, 307:106955, 2026. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2025.106955. URL https://www.sciencedirect.com/science/article/pii/S0045793025004153.

[154] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596 (7873):583–589, August 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://www.nature.com/articles/s41586-021-03819-2.

[155] Hachem Kadri, Emmanuel Duflos, Philippe Preux, Stéphane Canu, Alain Rakotomamonjy, and Julien Audiffren. Operator-valued kernels for learning from functional response data. *The Journal of Machine Learning Research*, 17(1):613–666, 2016.

[156] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[157] Zheng Tracy Ke, Yucong Ma, and Xihong Lin. Estimation of the number of spiked eigenvalues in a covariance matrix by bulk eigenvalue matching analysis. *Journal of the American Statistical Association*, pages 1–19, 2021.

[158] Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[159] Johannes Kepler. *Astronomia nova aitiologetos, seu physica coelestis, tradita commentariis de motibus stellae Martis, ex observationibus G.V. Tychonis Brahe*. Gotthard Vögelin, Prague, 1609.

[160] Nicole N. Khatibi, Daniil A. Radamovich, and Michael P. Brenner. EEFSUVA: A New Mathematical Olympiad Benchmark, September 2025. URL https://arxiv.org/abs/2510.01227v1.

[161] Ashish Khetan and Zohar Karnin. Prunenet: Channel pruning via global importance. *arXiv preprint arXiv:2005.11282*, 2020.

[162] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[163] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

[164] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything, April 2023. URL http://arxiv.org/abs/2304.02643. arXiv:2304.02643 [cs.CV].

[165] Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

[166] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M. Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A Benchmark of in-the-Wild Distribution Shifts. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5637–5664. PMLR, July 2021. URL https://proceedings.mlr.press/v139/koh21a.html.

[167] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA, USA, 2009.

[168] Tjalling C Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.

[169] Tomasz Korbak, Ethan Perez, and Christopher L Buckley. Rl with kl penalties is better viewed as bayesian inference. *arXiv preprint arXiv:2205.11275*, 2022.

[170] Holger Kösters and Alexander Tikhomirov. Limiting spectral distributions of sums of products of non-hermitian random matrices. *arXiv preprint arXiv:1506.04436*, 2015.

[171] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[172] Nikola B Kovachki, Samuel Lanthaler, and Andrew M Stuart. Operator learning: Algorithms and analysis. *Handbook of Numerical Analysis*, 25: 419–467, 2024.

[173] Nikola B. Kovachki, Samuel Lanthaler, and Andrew M. Stuart. Operator Learning: Algorithms and Analysis, February 2024. URL http://arxiv.org/abs/2402.15715. arXiv:2402.15715 [cs, math].

[174] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[175] Sergei Kucherenko and Shugfang Song. Derivative-based global sensitivity measures and their link with Sobol' sensitivity indices. In *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014*, pages 455–469. Springer, 2016.

[176] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[177] Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '23, pages 1–11, New York, NY, USA, June 2023. Association for Computing Machinery. ISBN 979-8-4007-0190-0. doi: 10.1145/3592979.3593412. URL https://dl.acm.org/doi/10.1145/3592979.3593412.

[178] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, 2, 1989.

[179] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://www.nature.com/articles/nature14539. Number: 7553 Publisher: Nature Publishing Group.

[180] Yongbin Lee and Dong-Hoon Choi. Pointwise ensemble of meta-models using v nearest points cross-validation. *Structural and Multidisciplinary Optimization*, 50(3):383–394, September 2014. ISSN 1615-1488. doi: 10.1007/s00158-014-1067-1. URL https://doi.org/10.1007/s00158-014-1067-1.

[181] William E Leeb. Matrix denoising for weighted loss functions and heterogeneous signals. *SIAM Journal on Mathematics of Data Science*, 3(3): 987–1012, 2021.

[182] Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution through large models. In *Handbook of evolutionary machine learning*, pages 331–366. Springer, 2023.

[183] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

[184] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[185] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. URL http://arxiv.org/abs/2010.08895. arXiv:2010.08895 [cs, math].

[186] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

[187] Pengwei Liang, Shuai Zhang, Yong Pang, Jianji Li, and Xueguan Song. A Pointwise-Optimal Ensemble of Surrogate Models. *Journal of Mechanical Design*, 145(111705), August 2023. ISSN 1050-0472. doi: 10.1115/1.4062979. URL https://doi.org/10.1115/1.4062979.

[188] Zhongqi Liang, Li Cai, Suojin Wang, and Qihua Wang. K-fold cross-validation based frequentist model averaging for linear models with non-ignorable missing responses, October 2024. URL https://www.researchsquare.com/article/rs-5030035/v1. ISSN: 2693-5015.

[189] Charles Lindsey and Simon Sheather. Variable selection in linear regression. *The Stata Journal*, 10(4):650–669, 2010.

[190] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[191] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[192] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

[193] Haitao Liu, Shengli Xu, Xiaofang Wang, Jigang Meng, and Shuhua Yang. Optimal weighted pointwise ensemble of radial basis functions with different basis functions. *AIAA Journal*, 54(10):3117–3133, 2016. doi: 10.2514/1.J054664. URL https://doi.org/10.2514/1.J054664.

[194] Ping Liu, C. I. Siettos, C. W. Gear, and I. G. Kevrekidis. Equation-free model reduction in agent-based computations: Coarse-grained bifurcation and variable-free rare event analysis. *Mathematical Modelling of Natural Phenomena*, 10(3):71–90, 2015.

[195] Lorenzo Livi and Antonello Rizzi. The graph matching problem. *Pattern Analysis and Applications*, 16:253–283, 2013.

[196] Eliane Maria Loiola, Nair Maria Maia De Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European journal of operational research*, 176(2):657–690, 2007.

[197] Torkel E. Loman, Yingbo Ma, Vasily Ilin, Shashi Gowda, Niklas Korsbo, Nikhil Yewale, Chris Rackauckas, and Samuel A. Isaacson. Catalyst: Fast and flexible modeling of reaction networks, October 2023. URL http://dx.doi.org/10.1371/journal.pcbi.1011530.

[198] Da Long, Nicole Mrvaljević, Shandian Zhe, and Bamdad Hosseini. A kernel framework for learning differential equations and their solution operators. *Physica D: Nonlinear Phenomena*, 460:134095, 2024.

[199] David Lopez-Paz, Krikamol Muandet, Bernhard Schölkopf, and Iliya Tolstikhin. Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning*, pages 1452–1461. PMLR, 2015.

[200] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130–141, 1963.

[201] Edward N. Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, volume 1, pages 1–18. Reading, 1996.

[202] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3): 218–229, 2021.

[203] Alexandra Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power Hungry Processing: Watts Driving the Cost of AI Deployment? In *The 2024 ACM Conference on Fairness Accountability and Transparency*, pages 85–99, June 2024. doi: 10.1145/3630106.3658542. URL http://arxiv.org/abs/2311.16863. arXiv:2311.16863 [cs.LG].

[204] Michael Mahoney and Charles Martin. Traditional and heavy tailed self regularization in neural network models. In *International Conference on Machine Learning*, pages 4284–4293. PMLR, 2019.

[205] Thierry A. Mara and Stefano Tarantola. Variance-based sensitivity indices for models with dependent inputs. *Reliability Engineering & System Safety*, 107:115–121, 2012.

[206] Jo Marchant. Powerful antibiotics discovered using AI. *Nature*, February 2020. doi: 10.1038/d41586-020-00018-3. URL https://www.nature.com/articles/d41586-020-00018-3.

[207] Vladimir Alexandrovich Marchenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 114(4):507–536, 1967.

[208] Charles H Martin and Michael W Mahoney. Heavy-tailed universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 505–513. SIAM, 2020.

[209] Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *The Journal of Machine Learning Research*, 22(1):7479–7551, 2021.

[210] Charles H Martin, Tongsu Peng, and Michael W Mahoney. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):4122, 2021.

[211] Malek Mechergui and Sarath Sreedharan. Expectation Alignment: Handling Reward Misspecification in the Presence of Expectation Mismatch. *Advances in Neural Information Processing Systems*, 37:62458–62479, December 2024. doi: 10.52202/079017-1997. URL https://papers.nips.cc/paper_files/paper/2024/hash/72393bd47a35f5b3bee4c609e7bba733-Abstract-Conference.html?utm_source=chatgpt.com.

[212] Tahir Mehmood, Kristian Hovde Liland, Lars Snipen, and Solve Sæbø. A review of variable selection methods in partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 118:62–69, 2012.

[213] Xuran Meng and Jianfeng Yao. Impact of classification difficulty on the weight matrices spectra in deep learning and application to early-stopping. *Journal of Machine Learning Research*, 24:1–40, 2023.

[214] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

[215] Charles A Micchelli and Theodore J Rivlin. A survey of optimal recovery. In *Optimal estimation in approximation theory*, pages 1–54. Springer, 1977.

[216] Sebastian Mika, Bernhard Schölkopf, Alexander J Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *NIPS*, volume 11, pages 536–542, 1998.

[217] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. *IMA Journal of Numerical Analysis*, 42(2):981–1022, April 2022. doi: 10.1093/imanum/drab032.

[218] Alexander Mitsos, Jaromil Najman, and Ioannis G. Kevrekidis. Optimal deterministic algorithm generation. *Journal of Global Optimization*, 71(4): 891–913, 2018. doi: 10.1007/s10898-018-0611-8.

[219] Roberto Molinaro, Yunan Yang, Björn Engquist, and Siddhartha Mishra. Neural inverse operators for solving PDE inverse problems. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25105–25139. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/molinaro23a.html.

[220] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

[221] Carlos Mora, Amin Yousefpour, Shirin Hosseinmardi, Houman Owhadi, and Ramin Bostanabad. Operator learning with gaussian processes. *Computer Methods in Applied Mechanics and Engineering*, 434:117581, 2025.

[222] Stephen L. Morgan and Christopher Winship. *Counterfactuals and Causal Inference: Methods and Principles for Social Research*. Cambridge University Press, Cambridge, UK, 2nd edition, 2015.

[223] Stephen L Morgan and Christopher Winship. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

[224] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[225] Nicholas H Nelsen and Andrew M Stuart. The random feature model for input-output maps between banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021.

[226] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[227] Alexander Novikov, Ngân V u, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. White paper.

[228] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, June 2025. URL http://arxiv.org/abs/2506.13131. arXiv:2506.13131 [cs.AI].

[229] Royal Swedish Academy of Sciences. Nobel Prize in Chemistry 2024, 2024. URL https://www.nobelprize.org/prizes/chemistry/2024/press-release/.

[230] Royal Swedish Academy of Sciences. Nobel Prize in Physics 2024, 2024. URL https://www.nobelprize.org/prizes/physics/2024/press-release/.

[231] OpenAI. AI for Scientific Research and Discovery, 2025. URL https://openai.com/science/.

[232] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia

Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, March 2024. URL http://arxiv.org/abs/2303.08774. arXiv:2303.08774 [cs].

[233] Kyle Orland. Why Anthropic's Claude still hasn't beaten Pokémon, March 2025. URL https://arstechnica.com/ai/2025/03/why-anthropics-claude-still-hasnt-beaten-pokemon/.

[234] Sean O'Rourke, Van Vu, and Ke Wang. Random perturbation of low rank matrices: Improving classical bounds. *Linear Algebra and its Applications*, 540:26–59, 2018.

[235] Art B Owen. Variance components and generalized sobol'indices. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):19–41, 2013.

[236] H. Owhadi and C. Scovel. *Operator Adapted Wavelets, Fast Solvers, and Numerical Homogenization, from a game theoretic approach to numerical*

approximation and algorithm design. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2019.

[237] H. Owhadi, C. Scovel, and F. Schäfer. Statistical Numerical Approximation. *Notices of the AMS*, 66(10), 2019.

[238] Houman Owhadi. Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games. *SIAM Review*, 59(1):99–149, 2017.

[239] Houman Owhadi. Computational graph completion. *Research in the Mathematical Sciences*, 9(2):1–33, 2022.

[240] Houman Owhadi. Do ideas have shape? idea registration as the continuous limit of artificial neural networks. *Physica D: Nonlinear Phenomena*, 444: 133592, 2023.

[241] Houman Owhadi and Clint Scovel. *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*, volume 35. Cambridge University Press, 2019.

[242] Houman Owhadi and Gene Ryan Yoo. Kernel flows: From learning kernels from data into the abyss. *Journal of Computational Physics*, 389:22–47, 2019.

[243] Houman Owhadi, Clint Scovel, and Gene Ryan Yoo. *Kernel Mode Decomposition and the programming of kernels*. Springer, 2021.

[244] Houman Owhadi, Clint Scovel, and Gene Ryan Yoo. *Kernel Mode Decomposition and the Programming of Kernels*. Springer, Cham, 2021.

[245] SEAN O'ROURKE, VU VAN, and Ke Wang. Matrices with gaussian noise: optimal estimates for singular subspace perturbation. *arXiv e-prints*, pages arXiv–1803, 2018.

[246] Richard S. Palais. The symmetries of solitons, 1997.

[247] Gintaras Palubeckis. An algorithm for construction of test cases for the quadratic assignment problem. *Informatica*, 11(3):281–296, 2000.

[248] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models, February 2022. URL http://arxiv.org/abs/2201.03544. arXiv:2201.03544 [cs.LG].

[249] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *Proceedings of the IEEE/cvf international conference on computer vision*, pages 377–386, 2021.

[250] Jiyoung Park, Ian Pelakh, and Stephan Wojtowytsch. Minimum norm interpolation by perceptra: Explicit regularization and implicit bias. *NeurIPS*, 2023.

[251] Leonid Pastur. On random matrices arising in deep neural networks. gaussian case. *arXiv preprint arXiv:2001.06188*, 2020.

[252] Leonid Pastur and Victor Slavin. On random matrices arising in deep neural networks: General iid case. *Random Matrices: Theory and Applications*, 12 (01):2250046, 2023.

[253] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

[254] Maciej Pawlikowski and Agata Chorowska. Weighted ensemble of statistical models. *International Journal of Forecasting*, 36(1):93–97, January 2020. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2019.03.019. URL https://www.sciencedirect.com/science/article/pii/S0169207019301190.

[255] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, USA, 1988. ISBN 9780934613736.

[256] Judea Pearl. *Causality*. Cambridge university press, 2009.

[257] Judea Pearl, Madelyn Glymour, and Nicholas P. Jewell. *Causal Inference in Statistics: A Primer*. John Wiley & Sons, Chichester, UK, 2016.

[258] Tian Peng, Wang Huanchen, and Zhang Dongme. Simulated annealing for the quadratic assignment problem: A further study. *Computers & industrial engineering*, 31(3-4):925–928, 1996.

[259] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[260] Ethan Pickering, Stephen Guth, George Em Karniadakis, and Themistoklis P. Sapsis. Discovering and forecasting extreme events via active learning in neural operators. *Nature Computational Science*, 2(12):823–833, 2022.

[261] Romain Piron, Muhammad Idham Habibie, and Claire Goursaud. Mixed grover: A hybrid version to improve grover's algorithm for unstructured database search. *IEEE Transactions on Quantum Engineering*, 2025.

[262] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal Mathematics Statement Curriculum Learning, February 2022. URL http://arxiv.org/abs/2202.01344. arXiv:2202.01344 [cs.LG].

[263] Boris Teodorovich Polyak. Newton-kantorovich method and its global convergence. *Journal of Mathematical Sciences*, 133:1513–1523, 2006.

[264] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, January 2022. URL http://arxiv.org/abs/2201.02177. arXiv:2201.02177 [cs.LG].

[265] Lutz Prechelt. Early stopping—but when? *Neural Networks: Tricks of the Trade: Second Edition*, pages 53–67, 2012.

[266] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Hoboken, NJ, USA, 2005.

[267] Alfio Quarteroni and Alberto Valli. *Numerical approximation of partial differential equations*. Springer, 1994.

[268] Antonin Raffin. RL Baselines3 Zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

[269] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268): 1–8, 2021. URL https://jmlr.org/papers/v22/20-1364.html.

[270] Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A Yeh, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Pretraining codomain attention neural operators for solving multiphysics pdes. *Advances in Neural Information Processing Systems*, 37, 2024.

[271] M. Raissi, P. Perdikaris, and G.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, February 2019. ISSN 00219991. doi: 10.1016/j.jcp.2018.10.045. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999118307125.

[272] Adil Rasheed, Omer San, and Trond Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, 8:21980–22012, 2020.

[273] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass., 3. print edition, 2008. ISBN 978-0-262-18253-9.

[274] Niklas Reinhardt, Sven Wang, and Jakob Zech. Statistical learning theory for neural operators. *arXiv preprint arXiv:2412.17582*, 2024.

[275] Christopher X Ren, Sidhant Misra, Marc Vuffray, and Andrey Y Lokhov. Learning continuous exponential families beyond gaussian. *arXiv preprint arXiv:2102.09198*, 2021.

[276] Luke S. Richardson, Justin Fidock, and Indra Gunawan. Systematic Literature Review of Levels of Automation (Autonomy) Taxonomy: Critiques and Recommendations. *International Journal of Human–Computer Interaction*, 41(24):15824–15843, December 2025. ISSN 1044-7318. doi: 10.1080/10447318.2025.2502978. URL https://doi.org/10.1080/10447318.2025.2502978. _eprint: https://doi.org/10.1080/10447318.2025.2502978.

[277] Markus Ringnér. What is principal component analysis? *Nature Biotechnology*, 26(3):303–304, 2008.

[278] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625 (7995):468–475, 2024.

[279] Samuel Rudy, Alessandro Alla, Steven L Brunton, and J Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019.

[280] Didier Rullière, Nicolas Durrande, Bachoc François, and Chevalier Clément. Nested Kriging predictions for datasets with a large number of observations. *Statistics and Computing*, 28(4):849–867, 2018. doi: 10.1007/s11222-017-9766-2.

[281] Didier Rullière, Nicolas Durrande, François Bachoc, and Clément Chevalier. Nested Kriging predictions for datasets with large number of observations, July 2017. URL http://arxiv.org/abs/1607.05432. arXiv:1607.05432 [stat].

[282] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252, December 2015. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-015-0816-y. URL http://link.springer.com/10.1007/s11263-015-0816-y.

[283] Thiziri Nait Saada and Jared Tanner. On the initialisation of wide low-rank feedforward neural networks. *arXiv preprint arXiv:2301.13710*, 2023.

[284] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. *Advances in Neural Information Processing Systems*, 38, 2025.

[285] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[286] Tuhin Sahai. Dynamical systems theory and algorithms for np-hard problems. *Advances in Dynamics, Optimization and Computation: A volume dedicated to Michael Dellnitz on the occasion of his 60th birthday*, pages 183–206, 2020.

[287] Tuhin Sahai, Adrian Ziessler, Stefan Klus, and Michael Dellnitz. Continuous relaxations for the traveling salesman problem. *Nonlinear Dynamics*, 97: 2003–2022, 2019.

[288] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference, October 2023. URL http://arxiv.org/abs/2310.03003. arXiv:2310.03003 [cs.CL].

[289] Will Sawin. An explicit lower bound for the unit distance problem, May 2026. URL https://arxiv.org/abs/2605.20579v1.

[290] Florian Schäfer and Houman Owhadi. Sparse recovery of elliptic solvers from matrix-vector products. *SIAM Journal on Scientific Computing*, 2023.

[291] Florian Schäfer and Houman Owhadi. Sparse recovery of elliptic solvers from matrix-vector products. *SIAM Journal on Scientific Computing*, 46(2): A998–A1025, 2024. doi: 10.1137/22M154226X. URL https://doi.org/10.1137/22M154226X.

[292] Florian Schäfer, Matthias Katzfuss, and Houman Owhadi. Sparse cholesky factorization by kullback–leibler minimization. *SIAM Journal on Scientific Computing*, 43(3):A2019–A2046, 2021.

[293] Florian Schäfer, Timothy John Sullivan, and Houman Owhadi. Compression, inversion, and approximate pca of dense kernel matrices at near-linear computational complexity. *Multiscale Modeling & Simulation*, 19(2):688–730, 2021.

[294] Florian Schäfer, Timothy John Sullivan, and Houman Owhadi. Compression, inversion, and approximate pca of dense kernel matrices at near-linear computational complexity. *Multiscale Modeling & Simulation*, 19(2):688–730, 2021.

[295] Robert E. Schapire. The Boosting Approach to Machine Learning: An Overview. In P. Bickel, P. Diggle, S. Fienberg, K. Krickeberg, I. Olkin, N. Wermuth, S. Zeger, David D. Denison, Mark H. Hansen, Christopher C. Holmes, Bani Mallick, and Bin Yu, editors, *Nonlinear Estimation and Classification*, volume 171, pages 149–171. Springer New York, New York, NY, 2003. ISBN 978-0-387-95471-4 978-0-387-21579-2. doi: 10.1007/978-0-387-21579-2_9. URL http://link.springer.com/10.1007/978-0-387-21579-2_9. Series Title: Lecture Notes in Statistics.

[296] Marten Scheffer, Jordi Bascompte, William A. Brock, Victor Brovkin, Stephen R. Carpenter, Vasilis Dakos, Hermann Held, Egbert H. van Nes, Max Rietkerk, and George Sugihara. Early-warning signals for critical transitions. *Nature*, 461(7260):53–59, 2009.

[297] Wilhelmus HA Schilders, Henk A Van der Vorst, and Joost Rommes. *Model order reduction: theory, research aspects and applications*, volume 13. Springer, 2008.

[298] Thomas Schreiber. Measuring information transfer. *Physical Review Letters*, 85(2):461, 2000.

[299] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[300] Frank Schweitzer, Giorgio Fagiolo, Didier Sornette, Fernando Vega-Redondo, Alessandro Vespignani, and Douglas R White. Economic networks: The new challenges. *science*, 325(5939):422–425, 2009.

[301] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A Generalized Representer Theorem. In David Helmbold and Bob Williamson, editors, *Computational Learning Theory*, pages 416–426, Berlin, Heidelberg, 2001. Springer. ISBN 978-3-540-44581-4. doi: 10.1007/3-540-44581-1_27.

[302] M. R. Scott. Automatic generation of finite difference and finite element code. *ACM Transactions on Mathematical Software*, 6(4):456–465, 1980. doi: 10.1145/355900.355902.

[303] Vadim Ivanovich Serdobolskii. *Multivariate statistical analysis: A high-dimensional approach*, volume 41. Springer Science & Business Media, 2000.

[304] RAJEN D SHAH and JONAS PETERS. The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48(3):1514–1538, 2020.

[305] T. B. Sheridan and W. L. Verplank. Human and Computer Control of Undersea Teleoperators. Technical Report ADA057655, Massachusetts Inst of Tech Cambridge Man-Machine Systems Lab, 1978. URL https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/ADA057655.xhtml. Num Pages: 188.

[306] Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950. ISSN 00034851. URL http://www.jstor.org/stable/2236561.

[307] Yitzchak Shmalo, Jonathan Jenkins, and Oleksii Krupchytskyi. Deep learning weight pruning with rmt-svd: Increasing accuracy and reducing overfitting. *arXiv preprint arXiv:2303.08986*, 2023.

[308] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, December 2017. URL http://arxiv.org/abs/1712.01815. arXiv:1712.01815 [cs].

[309] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[310] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[311] Aaditya K. Singh, Muhammed Yusuf Kocyigit, Andrew Poulton, David Esiobu, Maria Lomeli, Gergely Szilvasy, and Dieuwke Hupkes. Evaluation data contamination in LLMs: how do we measure it and (when) does it matter?, November 2024. URL http://arxiv.org/abs/2411.03923. arXiv:2411.03923 [cs.CL].

[312] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[313] Ilya M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[314] IM Soboĺ. Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp.*, 1:407, 1993.

[315] Ray J. Solomonoff. A formal theory of inductive inference. part i and part ii. *Information and Control*, 7(1-2):1–22, 224–254, 1964. doi: 10.1016/S0019-9958(64)90223-2.

[316] Zhuoran Song, Yihong Xu, Zhezhi He, Li Jiang, Naifeng Jing, and Xiaoyao Liang. Cp-vit: Cascade vision transformer pruning via progressive sparsity prediction. *arXiv preprint arXiv:2203.04570*, 2022.

[317] Aleksei G. Sorokin, Pieterjan Robbe, and Fred J. Hickernell. Fast Gaussian process regression for high dimensional functions with derivative information. In Motonobu Kanagawa, Jon Cockayne, Alexandra Gessner, and Philipp Hennig, editors, *Proceedings of the First International Conference on Probabilistic Numerics*, volume 271 of *Proceedings of Machine Learning Research*, pages 35–49. PMLR, 2025. URL https://proceedings.mlr.press/v271/sorokin25a.html.

[318] Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72, 1991.

[319] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, USA, 2nd edition, 2001.

[320] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[321] Max Staats, Matthias Thamm, and Bernd Rosenow. Boundary between noise and information applied to filtering neural network weight matrices. *arXiv preprint arXiv:2206.03927*, 2022.

[322] Lars Ståhle and Svante Wold. Analysis of variance (ANOVA). *Chemometrics and Intelligent Laboratory Systems*, 6(4):259–272, 1989.

[323] J. Stark, D. S. Broomhead, M. E. Davies, and J. Huke. Delay embeddings for forced systems. II. Stochastic forcing. *Journal of Nonlinear Science*, 13(6):519–577, 2003.

[324] Oliver Stegle, Dominik Janzing, Kun Zhang, Joris M Mooij, and Bernhard Schölkopf. Probabilistic latent variable models for distinguishing between cause and effect. *Advances in neural information processing systems*, 23, 2010.

[325] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush

Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. A Deep Learning Approach to Antibiotic Discovery. *Cell*, 180(4):688–702.e13, February 2020. ISSN 0092-8674. doi: 10.1016/j.cell.2020.01.021. URL https://www.sciencedirect.com/science/article/pii/S0092867420301021.

[326] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

[327] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.

[328] Floris Takens. Detecting strange attractors in turbulence. In D. A. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence, Warwick 1980*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer, Berlin, Heidelberg, 1981. ISBN 978-3-540-11171-9. doi: 10.1007/BFb0091924.

[329] Terrence Tao. A digestion of the Jacobian conjecture counterexample, July 2026. URL https://terrytao.wordpress.com/2026/07/21/a-digestion-of-the-jacobian-conjecture-counterexample/.

[330] Matthias Thamm, Max Staats, and Bernd Rosenow. Random matrix analysis of deep neural network weight matrices. *Physical Review E*, 106(5):054124, 2022.

[331] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[332] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[333] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, February 2023. URL http://arxiv.org/abs/2302.13971. arXiv:2302.13971 [cs].

[334] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[335] Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, January 2024. ISSN 1476-4687. doi: 10.1038/s41586-023-06747-5. URL https://www.nature.com/articles/s41586-023-06747-5.

[336] Sunil Vadera and Salem Ameen. Methods for pruning deep neural networks. *IEEE Access*, 10:63280–63300, 2022.

[337] Jean Vassoyan, Nathanaël Beau, and Roman Plaud. Ignore the kl penalty! boosting exploration on critical tokens to enhance rl fine-tuning. *arXiv preprint arXiv:2502.06533*, 2025.

[338] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[339] Aldo V Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 50(2):297–312, 1988.

[340] Thomas Verma and Judea Pearl. Causal networks: Semantics and expressiveness. In *Machine Intelligence and Pattern Recognition*, volume 9, pages 69–76. Elsevier, 1990.

[341] Roman Vershynin. High-dimensional probability by roman vershynin, 2018.

[342] Felipe A. C. Viana, Raphael T. Haftka, and Valder Steffen. Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. *Structural and Multidisciplinary Optimization*, 39(4):439–457, October 2009. ISSN 1615-1488. doi: 10.1007/s00158-008-0338-0. URL https://doi.org/10.1007/s00158-008-0338-0.

[343] Jean Virieux and Stéphane Operto. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26, 2009.

[344] Joshua T Vogelstein, John M Conroy, Vince Lyzinski, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. Fast approximate quadratic programming for graph matching. *PLOS ONE*, 10(4):e0121002, 2015.

[345] Grace Wahba. An introduction to smoothing spline anova models in rkhs, with examples in geographical data, medicine, atmospheric sciences and machine learning. *IFAC Proceedings Volumes*, 36(16):531–536, 2003.

[346] Xiaoliang Wan and George Em Karniadakis. Multi-element generalized polynomial chaos for arbitrary probability measures. *SIAM Journal on Scientific Computing*, 28(3):901–928, 2006.

[347] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[348] Xin Wang. 2d 2-8 fd acoustic modeling lab. https://csim.kaust.edu.sa/files/SeismicInversion/Chapter.FD/lab.FD2.8/lab.html, 2023. Accessed: 2025-10-07.

[349] Yanghua Wang. Frequencies of the Ricker wavelet. *Geophysics*, 80(2): A31–A37, 2015.

[350] Yong Wang. Gauss–Newton method. *WIREs Computational Statistics*, 4 (4):415–420, 2012. ISSN 1939-0068. doi: 10.1002/wics.1202. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.1202. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.1202.

[351] Felix Wong, Erica J. Zheng, Jacqueline A. Valeri, Nina M. Donghia, Melis N. Anahtar, Satotaka Omori, Alicia Li, Andres Cubillos-Ruiz, Aarti Krishnan, Wengong Jin, Abigail L. Manson, Jens Friedrichs, Ralf Helbig, Behnoush Hajian, Dawid K. Fiejtek, Florence F. Wagner, Holly H. Soutter, Ashlee M. Earl, Jonathan M. Stokes, Lars D. Renner, and James J. Collins. Discovery of a structural class of antibiotics with explainable deep learning. *Nature*, 626 (7997):177–185, February 2024. ISSN 1476-4687. doi: 10.1038/s41586-023-06887-8. URL https://www.nature.com/articles/s41586-023-06887-8.

[352] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. *IEEE/CAA Journal of Automatica Sinica*, 10 (5):1122–1136, May 2023. ISSN 2329-9274. doi: 10.1109/JAS.2023.123618. URL https://ieeexplore.ieee.org/document/10113601/.

[353] Xi Wu, Qingyi Li, Zhiqiang Li, Donghan Yang, Hui Yang, Wenjie Pan, Marek Perkowski, and Xiaoyu Song. Circuit optimization of grover quantum search algorithm. *Quantum Information Processing*, 22(1):69, 2023.

[354] Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. Language Models Meet World Models: Embodied Experiences Enhance Language Models, October 2023. URL http://arxiv.org/abs/2305.10626. arXiv:2305.10626 [cs.CL].

[355] Xuanzhe Xiao, Zeng Li, Chuanlong Xie, and Fengwei Zhou. Heavy-tailed regularization of weight matrices in deep neural networks. *arXiv preprint arXiv:2304.02911*, 2023.

[356] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[357] Cheng Xu, Shuhao Guan, Derek Greene, and M.-Tahar Kechadi. Benchmark Data Contamination of Large Language Models: A Survey, June 2024. URL http://arxiv.org/abs/2406.04244. arXiv:2406.04244 [cs.CL].

[358] Haoyuan Xu, Chang Li, Xinyan Ma, Xianhao Ou, Zihan Zhang, Tao He, Xiangyu Liu, Zixiang Wang, Jiafeng Liang, Zheng Chu, Runxuan Liu, Rongchuan Mu, Dandan Tu, Ming Liu, and Bing Qin. The Evolution of Tool Use in LLM Agents: From Single-Tool Call to Multi-Tool Orchestration, April 2026. URL http://arxiv.org/abs/2603.22862. arXiv:2603.22862 [cs.SE].

[359] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Wenrui Dai, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. Trained rank pruning for efficient deep neural networks. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 14–17. IEEE, 2019.

[360] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369, 2013.

[361] Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 678–679, 2020.

[362] Qiuling Yang, Alireza Sadeghi, and Gang Wang. Data-driven priors for robust psse via gauss-newton unrolled neural networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 12(1):172–181, 2022.

[363] Yifan Ye, Zhanxue Wang, and Xiaobo Zhang. An optimal pointwise weighted ensemble of surrogates based on minimization of local mean square error. *Structural and Multidisciplinary Optimization*, 62(2):529–542, August 2020. ISSN 1615-147X, 1615-1488. doi: 10.1007/s00158-020-02508-4. URL http://link.springer.com/10.1007/s00158-020-02508-4.

[364] Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. Pdeformer: Towards a foundation model for one-dimensional partial differential equations. *arXiv preprint arXiv:2402.12652*, 2024. URL https://arxiv.org/abs/2402.12652.

[365] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty Years of Mixture of Experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, August 2012. ISSN 2162-2388. doi: 10.1109/TNNLS.2012.2200299. URL https://ieeexplore.ieee.org/document/6215056/?arnumber=6215056. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[366] Eberhard Zeidler. *Nonlinear functional analysis and its applications: I: Fixed-point Theorems*. Springer Science & Business Media, 1986.

[367] Luis E. Zerpa, Nestor V. Queipo, Salvador Pintos, and Jean-Louis Salager. An optimization methodology of alkaline–surfactant–polymer flooding processes using field scale numerical simulation and multiple surrogates. *Journal of Petroleum Science and Engineering*, 47(3):197–208, June 2005. ISSN 0920-4105. doi: 10.1016/j.petrol.2005.03.002. URL https://www.sciencedirect.com/science/article/pii/S0920410505000379.

[368] Benjamin J. Zhang, Siting Liu, Stanley J. Osher, and Markos A. Katsoulakis. Probabilistic operator learning: Generative modeling and uncertainty quantification for foundation models of differential equations. *arXiv preprint arXiv:2509.05186*, 2025.

[369] Cha Zhang and Yunqian Ma, editors. *Ensemble Machine Learning: Methods and Applications*. Springer New York, New York, NY, 2012. ISBN 978-1-4419-9325-0 978-1-4419-9326-7. doi: 10.1007/978-1-4419-9326-7. URL https://link.springer.com/10.1007/978-1-4419-9326-7.

[370] Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. Verbalized Sampling: How to Mitigate Mode Collapse and Unlock LLM Diversity, October 2025. URL https://arxiv.org/abs/2510.01171v4.

[371] K Zhang, J Peters, D Janzing, and B Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 804–813. AUAI Press, 2011.

[372] Kun Zhang and Vladimir E Korepin. Depth optimization of quantum search algorithms beyond grover's algorithm. *Physical Review A*, 101(3):032346, 2020.

[373] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 804–813, 2011.

[374] Xinyu Zhang and Chu-An Liu. Model averaging prediction by K -fold cross-validation. *Journal of Econometrics*, 235(1):280–301, July 2023. ISSN 03044076. doi: 10.1016/j.jeconom.2022.04.007. URL https://linkinghub.elsevier.com/retrieve/pii/S0304407622000975.

[375] Zhixiang Zhang and Guangming Pan. Tracy-widom law for the extreme eigenvalues of large signal-plus-noise matrices. *arXiv preprint arXiv:2009.12031*, 2020.

[376] Zhizhen Zhang, Hyemin Gu, Benjamin J. Zhang, Daniel Elenius, Michael Tyrrell, Theo J. Bourdais, Houman Owhadi, Markos A. Katsoulakis, and Tuhin Sahai. ISOMORPH: A Supply Chain Digital Twin for Simulation, Dataset Generation, and Forecasting Benchmarks, May 2026. URL http://arxiv.org/abs/2605.12768. arXiv:2605.12768 [stat.ML].

[377] Xun Zheng, Bryon Aragam, Pradeep K. Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[378] Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric Xing. Learning sparse nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, pages 3414–3425. PMLR, 2020.

[379] Jian Zhou, Chandra L. Theesfeld, Kevin Yao, Kathleen M. Chen, Aaron K. Wong, and Olga G. Troyanskaya. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nature Genetics*, 50(8):1171–1179, August 2018. ISSN 1546-1718. doi: 10.1038/s41588-018-0160-6. URL https://www.nature.com/articles/s41588-018-0160-6.

[380] Fei Zhu, Shijie Ma, Zhen Cheng, Xu-Yao Zhang, Zhaoxiang Zhang, Dacheng Tao, and Cheng-Lin Liu. Open-world machine learning: A review and new outlooks, June 2025. URL http://arxiv.org/abs/2403.01759. arXiv:2403.01759 [cs.LG].

[381] Linghua Zhu, Ho Lun Tang, George S Barron, FA Calderon-Vargas, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer. *Physical Review Research*, 4(3):033029, 2022.

[382] Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning. *arXiv preprint arXiv:2104.08500*, 2(6):7, 2021.

[383] Yuanran Zhu, Yu-Hang Tang, and Changho Kim. Learning stochastic dynamics with statistics-informed neural network. *Journal of Computational Physics*, 474:111819, 2023.

Part IV

Appendices

Appendix A

APPENDIX TO CHAPTER 2: MINIMAL VARIANCE MODEL AGGREGATION

A.1 Theorem on the stability of the MEVA

We consider the setting without a dependence on x to prove a theorem on the robustness of MEVA compared to MEEA. Let Y be a random real-valued variable and the random vector of models $M \in \mathbb{R}^n$. Suppose we have i.i.d observations $(M^1, Y^1), \ldots, (M^N, Y^N)$, collected in the matrices $\mathcal{M} \in \mathbb{R}^{N\times n}$ and $\mathcal{Y} \in \mathbb{R}^N$. We wish to study the Mean Squared Error (MSE) of an aggregation method of the form $M_A = \alpha^T M$. Define

$$\mathcal{L}(\alpha) = \mathbb{E}\left[(Y - \alpha^T M)^2\right] \tag{A.1}$$

Let $C = \mathbb{E}\left[MM^T\right]$, $\gamma = \mathbb{E}\left[YM\right]$, we have that the loss is minimized at $\alpha^* = C^{-1}\gamma$ the MEA. Let $M = Y\mathbb{1} + \epsilon Z$, where Z is the scaled error, with ϵ chosen so that $\mathbb{E}\left[ZZ^T\right] := A$ has Frobenius norm 1 ; and $b := \mathbb{E}\left[YZ\right]$. We have that

$$\gamma = E[Y^2]\mathbb{1} + \epsilon b \tag{A.2}$$

$$C = \epsilon^2 A + (E[Y^2]1 + \epsilon b)1^T + \epsilon 1 b^T \tag{A.3}$$

A.1.1 Characterization of α^*

To facilitate our computations, we introduce $D = \epsilon^2 A + (E[Y^2]1 + \epsilon b)1^T$. We define

$$s = \mathbb{1}^T A^{-1}\mathbb{1}, \tag{A.4}$$

$$t = b^T A^{-1}\mathbb{1}. \tag{A.5}$$

$$u = b^T A^{-1} b. \tag{A.6}$$

$$v = \mathbb{1}^T A^{-1}\left(E[Y^2]\mathbb{1} + \epsilon b\right) = E[Y^2]s + \epsilon t \tag{A.7}$$

Using the [306] formula, we find that

$$D^{-1} = \frac{1}{\epsilon^2}\left(A^{-1} - \frac{1}{\epsilon^2 + v}A^{-1}(E[Y^2]1 + \epsilon b)1^T A^{-1}\right) \tag{A.8}$$

$$C^{-1} = D^{-1} - \frac{\epsilon}{1+w}D^{-1}\mathbb{1}b^T D^{-1}, \tag{A.9}$$

$$w = \epsilon b^t D^{-1}\mathbb{1}. \tag{A.10}$$

Let $\alpha, \beta, \delta, \zeta$ such that

$$D^{-1}\mathbb{1} = \alpha A^{-1}\mathbb{1} + \beta A^{-1}b \tag{A.11}$$

$$D^{-1}b = \delta A^{-1}\mathbb{1} + \zeta A^{-1}b \tag{A.12}$$

Thus, $w = \epsilon b^t D^{-1}1 = \epsilon(\alpha t + \beta u)$ and $b^T D^{-1}b = \delta t + \zeta u$. The coefficients are:

$$\begin{aligned} \alpha &= \tfrac{1}{\epsilon^2} \cdot \tfrac{\epsilon^2+\epsilon t}{\epsilon^2+v}, \quad \beta = -\tfrac{1}{\epsilon} \cdot \tfrac{s}{\epsilon^2+v}, \\ \delta &= \tfrac{-E[Y^2]t}{\epsilon^2(\epsilon^2+v)}, \quad \zeta = \tfrac{\epsilon^2+E[Y^2]s}{(\epsilon^2+v)\epsilon^2}. \end{aligned} \tag{A.13}$$

The computation of $C^{-1}\gamma$ is broken into two parts:

$$C^{-1}\gamma = C^{-1}\left(E[Y^2]\mathbb{1} + \epsilon b\right) = E[Y^2]C^{-1}\mathbb{1} + \epsilon C^{-1}b \tag{A.14}$$

The intermediate terms are:

$$C^{-1}\mathbb{1} = \frac{1}{1+w}D^{-1}\mathbb{1}, \tag{A.15}$$

$$C^{-1}b = D^{-1}b - \frac{\epsilon(b^T D^{-1}b)}{1+w}D^{-1}\mathbb{1}. \tag{A.16}$$

Substitute $D^{-1}\mathbb{1}$ and $D^{-1}b$:

$$C^{-1}\mathbb{1} = \frac{\alpha}{1+w}A^{-1}\mathbb{1} + \frac{\beta}{1+w}A^{-1}b \tag{A.17}$$

$$C^{-1}b = \left(\delta - \frac{\epsilon(\delta t + \zeta u)}{1+w}\alpha\right)A^{-1}\mathbb{1} + \left(\zeta - \frac{\epsilon(\delta t + \zeta u)}{1+w}\beta\right)A^{-1}b. \tag{A.18}$$

Write $C^{-1}\gamma = xA^{-1}\mathbb{1} + yA^{-1}b$, then

$$x = E[Y^2]\frac{\alpha}{1+w} + \epsilon\delta - \frac{\epsilon^2(\delta t + \zeta u)}{1+w}\alpha \tag{A.19}$$

$$y = E[Y^2]\frac{\beta}{1+w} + \epsilon\zeta - \frac{\epsilon^2(\delta t + \zeta u)}{1+w}\beta. \tag{A.20}$$

Using Sympy [214] to simplify these expressions, we find that:

$$x = \frac{E[Y^2] - u}{s(E[Y^2] - u) + (\epsilon + t)^2} \tag{A.21}$$

$$y = \frac{t + \epsilon}{s(E[Y^2] - u) + (\epsilon + t)^2} \tag{A.22}$$

Defining $\alpha_V = \frac{A^{-1}\mathbb{1}}{\mathbb{1}^T A^{-1}\mathbb{1}}$ and $\alpha_R = \frac{A^{-1}b}{\epsilon + \mathbb{1}^T A^{-1}b}$, we have that

$$\alpha^* = \lambda\alpha_V + (1-\lambda)\alpha_R \text{ where } \lambda = \frac{s(\mathbb{E}\left[Y^2\right] - u)}{s(\mathbb{E}\left[Y^2\right] - u) + (\epsilon + t)^2} \tag{A.23}$$

One may find that

$$\mathbb{E}\left[Y^2\right] - u = \mathbb{E}\left[Y\right]^2 - \\ \mathbb{E}\left[Y(M - Y\mathbb{1}))\right]^T \mathbb{E}\left[(M - Y\mathbb{1}))(M - Y\mathbb{1}))^T\right]^{-1} \mathbb{E}\left[Y(M - Y\mathbb{1}))\right] \geq 0$$

using the Schur complement of the covariance matrix of the vector $(Y, M^T - Y\mathbb{1}^T)^T \in \mathbb{R}^{n+1}$. Thus, $\lambda \in [0, 1]$, and α^* is a convex combination of the MVA, α_V, and α_R. Note that

$$\mathbb{1}^T C^{-1} \gamma = x * s + y * t = 1 - \epsilon y \tag{A.24}$$

and

$$\gamma^T C^{-1} \gamma = \mathbb{E}\left[Y^2\right](y * t + x * s) + \epsilon * (y * u + x * t) \tag{A.25}$$

$$= \frac{\left(E[Y^2]\right)^2 s + 2E[Y^2]\epsilon t - E[Y^2]su + E[Y^2]t^2 + \epsilon^2 u}{s(E[Y^2] - u) + (\epsilon + t)^2} \tag{A.26}$$

$$= \mathbb{E}\left[Y^2\right] - \epsilon^2 \frac{\mathbb{E}\left[Y^2\right] - u}{s(E[Y^2] - u) + (\epsilon + t)^2} \tag{A.27}$$

Thus, we have that

$$\mathcal{L}(\alpha^*) = \frac{\lambda \epsilon^2}{s} = \lambda \mathcal{L}(\alpha_V) \tag{A.28}$$

and

$$\mathcal{L}(\alpha_V) = \epsilon^2 {\alpha_V}^T A \alpha_V = \frac{\epsilon^2}{s} \tag{A.29}$$

A.1.2 Pertubation of α_V

We want to study the finite sample properties of the empirical estimates, thus we define

$$\hat{A} = \frac{1}{\epsilon^2 N}(\mathcal{M}^T - \mathbb{1}\mathcal{Y}^T)(\mathcal{M} - \mathcal{Y}\mathbb{1}^T) := A + \frac{1}{\sqrt{N}}\delta A \tag{A.30}$$

$$\hat{b} = \frac{1}{\epsilon N}(\mathcal{M}^T - \mathbb{1}\mathcal{Y}^T)\mathcal{Y} := b + \frac{1}{\sqrt{N}}\delta b \tag{A.31}$$

$$\widehat{\mathbb{E}\left[Y^2\right]} = \frac{1}{N}\|\mathcal{Y}\|_2^2 := \mathbb{E}\left[Y^2\right] + \frac{1}{\sqrt{N}}\delta V \tag{A.32}$$

The central limit theorem asserts that $\delta A, \delta b$ and δV converge in distribution to a Gaussian random variable, with finite covariance matrices. If we study the perturbation of α_V when estimating it using $\hat{A}$, we have that $\mathcal{L}(\alpha_V) = \epsilon^2 \frac{\mathbb{1}\hat{A}^{-1}A\hat{A}^{-1}\mathbb{1}}{(\mathbb{1}\hat{A}^{-1}\mathbb{1})^2}$. At

the first order in $\frac{1}{\sqrt{N}}$,

$$\hat{A}^{-1}\mathbb{1} = A^{-1}\mathbb{1} - \frac{1}{\sqrt{N}}A^{-1}\delta A A^{-1}\mathbb{1} \tag{A.33}$$

$$\mathbb{1}^T\hat{A}^{-1}\mathbb{1}^2 = \mathbb{1}^T A^{-1}\mathbb{1}^2 - 2\frac{1}{\sqrt{N}}\mathbb{1}^T A^{-1}\mathbb{1}\mathbb{1}^T A^{-1}\delta A A^{-1}\mathbb{1} \tag{A.34}$$

$$\mathbb{1}^T\hat{A}^{-1}A\hat{A}^{-1}\mathbb{1} = \mathbb{1}^T A^{-1}\mathbb{1} - 2\frac{1}{\sqrt{N}}\mathbb{1}^T A^{-1}\delta A A^{-1}\mathbb{1} \tag{A.35}$$

$$\tag{A.36}$$

Using the first order expansion of a fraction,

$$\frac{M+\eta N}{O+\eta Q} = \frac{M}{O} + \eta\left(\frac{N}{O} - \frac{MQ}{O^2}\right) + O(\eta^2) \tag{A.37}$$

We find that first-order terms cancel out, meaning that

$$\mathcal{L}(\hat{\alpha}_V) = \mathcal{L}(\alpha_V) + O(\frac{\epsilon^2}{N}) \tag{A.38}$$

$O(\frac{\epsilon^2}{N}) = \frac{\epsilon^2}{N}f(\delta A)$, where f is a continuous function of δA such that $f(\delta A)/\|\delta A\|^2 = O(1)$ as $\delta A \to 0$. Since δA converges in distribution to a Gaussian Z with finite covariance, using the continuous mapping theorem [35], we have that $f(\delta A)$ converges to $f(Z)$ in distribution, a finite random variable.

A.1.3 Perturbation of α^*

A.1.3.1 Perturbations of t, u, and s

If $(A + \frac{1}{\sqrt{N}}\delta A)(x + \delta x) = y + \frac{1}{\sqrt{N}}\delta y$, then at the first order $\delta x = \frac{1}{\sqrt{N}}A^{-1}(\delta y - \delta A x)$:

$$\begin{aligned}
\delta s &= -\mathbb{1}^T A^{-1}(\delta A)A^{-1}\mathbb{1},\\
\delta t &= \mathbb{1}^T A^{-1}\delta b - \mathbb{1}^T A^{-1}(\delta A)A^{-1}b,\\
\delta u &= 2b^T A^{-1}\delta b - b^T A^{-1}(\delta A)A^{-1}b.
\end{aligned}$$

A.1.3.2 Perturbation of the Denominator

The denominator of $C^{-1}\gamma$ is:

$$D = s(E[Y^2] - u) + (\epsilon + t)^2.$$

The perturbation is:

$$\sqrt{N}(\hat{D} - D) = \delta D = \delta s(E[Y^2] - u) + s\delta v - s\delta u + 2(\epsilon + t)\delta t.$$

A.1.3.3 Perturbation of the Numerator

The numerator of $\hat{C}^{-1}\hat{\gamma}$ is:

$$\hat{\mathcal{N}} = (E[\hat{Y}^2] - \hat{u})\hat{A}^{-1}\mathbb{1} + (\epsilon + \hat{t})\hat{A}^{-1}\hat{b}.$$

A.1.3.4 First Term: $(E[\hat{Y}^2] - \hat{u})\hat{A}^{-1}\mathbb{1}$

Using $E[\hat{Y}^2] = E[Y^2] + \delta v$, $\hat{u} = u + \delta u$, and the perturbation formula for $\hat{A}^{-1}\mathbb{1}$:

$$(E[\hat{Y}^2] - \hat{u})\hat{A}^{-1}\mathbb{1} = (E[Y^2] - u)A^{-1}\mathbb{1} - \frac{1}{\sqrt{N}}(E[Y^2] - u)A^{-1}(\delta A)A^{-1}\mathbb{1} + \frac{1}{\sqrt{N}}(\delta v - \delta u)A^{-1}\mathbb{1}.$$

A.1.3.5 Second Term: $(\epsilon + \hat{t})\hat{A}^{-1}\hat{b}$

Using $\hat{t} = t + \delta t$, $\hat{b} = b + \delta b$, and the perturbation formula for $\hat{A}^{-1}\hat{b}$:

$$(\epsilon + \hat{t})\hat{A}^{-1}\hat{b} = (\epsilon + t)A^{-1}b + \frac{1}{\sqrt{N}}\Big[\delta t A^{-1}b + (\epsilon + t)A^{-1}\delta b - (\epsilon + t)A^{-1}(\delta A)A^{-1}b\Big].$$

A.1.3.6 Final Expression for $\hat{N}$

Combining the two terms:

$$\begin{aligned}\hat{\mathcal{N}} = {} & (E[Y^2] - u)A^{-1}\mathbb{1} + (\epsilon + t)A^{-1}b \\ & + \frac{1}{\sqrt{N}}\Big[-(E[Y^2] - u)A^{-1}(\delta A)A^{-1}\mathbb{1} + (\delta v - \delta u)A^{-1}\mathbb{1} \\ & \quad + \delta t A^{-1}b + (\epsilon + t)A^{-1}\delta b - (\epsilon + t)A^{-1}(\delta A)A^{-1}b\Big]. \end{aligned} \tag{A.39}$$

A.1.3.7 Full Perturbation of $C^{-1}\gamma$

The final perturbed $C^{-1}\gamma$ is:

$$\hat{C}^{-1}\hat{\gamma} = \frac{\hat{\mathcal{N}}}{\hat{D}}.$$

Where:

$$\hat{D} = D + \frac{1}{\sqrt{N}}\delta D, \quad \hat{\mathcal{N}} = \mathcal{N} + \frac{1}{\sqrt{N}}\delta\mathcal{N},$$

and:

$$\mathcal{N} = (E[Y^2] - u)A^{-1}\mathbb{1} + (\epsilon + t)A^{-1}b.$$

Now, we see that $\hat{\mathcal{N}}^T A\hat{\mathcal{N}} = \mathcal{N}^T A\mathcal{N} + 2\mathcal{N}^T A\delta\mathcal{N}$ at the first order. We have that

$$\sqrt{N}A\delta\mathcal{N} = -(E[Y^2] - u)\delta AA^{-1}\mathbb{1} + (\delta v - \delta u)\mathbb{1} + \delta t b \tag{A.40}$$
$$+ (\epsilon + t)\delta b - (\epsilon + t)\delta AA^{-1}b \tag{A.41}$$
$$\sqrt{N}\mathcal{N}A\delta\mathcal{N} = (E[Y^2] - u)\left[-(E[Y^2] - u)\mathbb{1}^T A^{-1}\delta AA^{-1}\mathbb{1} + (\delta v - \delta u)s + t\delta t\right. \tag{A.42}$$
$$\left. +(\epsilon + t)(\mathbb{1}^T A^{-1}\delta b - \mathbb{1}^T A^{-1}\delta AA^{-1}b\right] \tag{A.43}$$
$$+ (\epsilon + t)\left[-(E[Y^2] - u)b^T A^{-1}\delta AA^{-1}\mathbb{1} + (\delta v - \delta u)t\right. \tag{A.44}$$
$$\left. +u\delta t + (\epsilon + t)(b^T A^{-1}\delta b - b^T A^{-1}\delta AA^{-1}b\right] \tag{A.45}$$

Then, we have that

$$\sqrt{N}b^t\delta\mathcal{N} = -(E[Y^2] - u)b^T A^{-1}\delta AA^{-1}\mathbb{1} + (\delta v - \delta u)t \tag{A.46}$$
$$+ u\delta t + (\epsilon + t)(b^T A^{-1}\delta b - b^T A^{-1}\delta AA^{-1}b) \tag{A.47}$$

We have:

$$\hat{y}b^T\hat{\alpha} = \frac{1}{\hat{D}}(\epsilon + \hat{t})b^T\hat{\alpha} = \frac{1}{\hat{D}}\left[(\epsilon + t)b^T\alpha + \frac{1}{\sqrt{N}}\delta t b^t\alpha + \frac{1}{\sqrt{N}}(\epsilon + t)b^T\frac{\delta\mathcal{N}}{\hat{D}}\right] \tag{A.48}$$

and $\hat{y}^2 = \frac{(\epsilon+t)^2 + 2\frac{\epsilon}{\sqrt{N}}\delta t}{\hat{D}^2}$. Thus,

$$\mathcal{L}(\hat{\alpha}_E) = \frac{\epsilon^2}{\hat{D}^2}(\mathbb{E}\left[Y^2\right]\left[(\epsilon + t)^2 + 2\epsilon\delta t\right] \tag{A.49}$$
$$- 2\left[(\epsilon + t)b^T\alpha\hat{D} + \delta t b^t\alpha\hat{D} + (\epsilon + t)b^T\delta\mathcal{N}\right] \tag{A.50}$$
$$+ \mathcal{N}^T A\mathcal{N} + 2\mathcal{N}^T A\delta\mathcal{N}) \tag{A.51}$$

We define

$$\mathcal{L}_o = \mathbb{E}\left[Y^2\right](\epsilon + t)^2 - 2(\epsilon + t)b^T\alpha D + \mathcal{N}^T A\mathcal{N} \tag{A.52}$$
$$\mathcal{L}_1 = 2\epsilon\delta t\mathbb{E}\left[Y^2\right] - 2\left[\delta t b^T\alpha D + (\epsilon + t)b^T\delta\mathcal{N}\right] + 2\mathcal{N}^T A\delta\mathcal{N} \tag{A.53}$$

We have that

$$\mathcal{L}(\hat{\alpha}_E) = \epsilon^2\left[\frac{\mathcal{L}_o}{D^2} + \frac{1}{\sqrt{N}}\left(\frac{\mathcal{L}_1}{D^2} - \frac{2\mathcal{L}_0\delta D}{D^3}\right)\right] + O(\frac{\epsilon^2}{N}) \tag{A.54}$$

Sympy computations show that

$$\mathcal{L}(\hat{\alpha}_E) = \mathcal{L}(\alpha^*) + \frac{\epsilon^2 2E[Y^2]t}{\sqrt{N}D^2}\left(-\mathbb{1}^T A^{-1}\delta b + b^T A^{-1}\delta AA^{-1}\mathbb{1}\right) + O(\frac{\epsilon^2}{N}) \tag{A.55}$$

Here again, δA and δb jointly converge to Gaussian random variables, so using the continuous mapping theorem, $\xi_N = -\mathbb{1}^T A^{-1}\delta b + b^T A^{-1}\delta A A^{-1}\mathbb{1}$ converges to a Gaussian ξ with mean 0 and finite covariance (because it is a linear combination of δA and δb, which are jointly Gaussian in the limit).

A.1.4 Comparison between MEVA and MEEA

Building upon the previous sections, we derived the following results for the loss functions of the two methods:

$$\mathcal{L}(\hat{\alpha}_V) = \frac{1}{\lambda}\mathcal{L}(\alpha^*) + O(\frac{\epsilon^2}{N}) \tag{A.56}$$

$$\mathcal{L}(\hat{\alpha}_E) = \mathcal{L}(\alpha^*) + \frac{\epsilon^2 2E[Y^2]t}{\sqrt{N}D^2}\xi_N + O(\frac{\epsilon^2}{N}) \tag{A.57}$$

where

$$\lambda = \frac{s(\mathbb{E}\left[Y^2\right] - u)}{s(\mathbb{E}\left[Y^2\right] - u) + (\epsilon + t)^2},$$
$$\mathcal{L}(\alpha^*) = \frac{\lambda\epsilon^2}{s}.$$

We can reformulate these results as:

$$\mathcal{L}(\hat{\alpha}_V) = \mathcal{L}(\alpha^*)\left(1 + \frac{(\epsilon + t)^2}{s(\mathbb{E}\left[Y^2\right] - u)} + O\left(\frac{1}{N}\right)\right) \tag{A.58}$$

$$\mathcal{L}(\hat{\alpha}_E) = \mathcal{L}(\alpha^*)\left(1 + \frac{2E[Y^2]t}{D(\mathbb{E}\left[Y^2\right] - u)}\frac{\xi_N}{\sqrt{N}} + O\left(\frac{1}{N}\right)\right) \tag{A.59}$$

In the context of model aggregation, we expect errors to be small, leading to ϵ being close to 0. While the MVA does not assume that $b = \mathbb{E}\left[Y\frac{M - Y\mathbb{1}}{\epsilon}\right] = 0$, a small b makes λ closer to 1. This corresponds to the errors being uncorrolated with the target. This setting of fixed $t \neq 0$ is the one studied in the theorem. We may see that even for $\lambda = 0.5$, and small N, say $N = 100$, the difference between $\mathcal{L}(\alpha_V)$ and $\mathcal{L}(\alpha^*)$ is a factor 2, while the difference between $\frac{1}{N}$ and $\frac{1}{\sqrt{N}}$ is a factor 10, making the latter effect much dominant.

However, $t = 0$ makes the $N^{-\frac{1}{2}}$ term disappear, resulting in MEEA and MEVA having the same convergence rate. In this scenario: $\mathcal{L}(\hat{\alpha}_V) - \mathcal{L}(\alpha^*) = O(\epsilon^4 + \frac{\epsilon^2}{N})$. Thus, when t = 0, both methods perform well, with the performance gap being of the order ϵ^4.

We now consider the more general case of a possibly small error correlation with

the target, which leads to $t = \kappa\epsilon$. In this case,

$$\mathcal{L}(\hat{\alpha}_V) = \mathcal{L}(\alpha^*)\left(1 + \epsilon^2 \frac{(1+\kappa)^2}{s(\mathbb{E}\left[Y^2\right] - u)} + O(\frac{1}{N})\right) \tag{A.60}$$

$$\mathcal{L}(\hat{\alpha}_E) = \mathcal{L}(\alpha^*)\left(1 + \frac{\epsilon}{\sqrt{N}} \frac{2E[Y^2]\kappa\xi_N}{D(\mathbb{E}\left[Y^2\right] - u)} + O(\frac{1}{N})\right) \tag{A.61}$$

$$\tag{A.62}$$

This comparison reduces to evaluating the relative magnitudes of $\frac{\epsilon}{\sqrt{N}}$ and ϵ^2. While it is not possible to compare them a priori, our theorem is aimed at explaining the behavior of the aggregation when extrapolating to all x, the regime of very scarce data. For models that are accurate, it is expected that the model error ϵ is much less than the Monte-Carlo rate of the validation samples, $\frac{1}{\sqrt{N}}$, especially for N not very large. In this case, MEVA remains superior to MEEA, particularly for highly accurate models.

A.2 Aggregating PDE solvers in the Gaussian setting

This section describes the aggregation performance one may get with GP-based PDE solvers introduced in Chen et al. [56]. Subsection A.2.1 summarizes the method and subsection A.2.2 describes the performance of the aggregation of low-fidelity models obtained by constraining the GPs to satisfy the PDE at a small number of collocation points.

A.2.1 Solving PDEs with GPs and their aggregation

We now recall the GP-based PDE solver introduced in Chen et al. [56] and describe the proposed method for aggregating such solvers. Consider the Laplace equation as a prototypical example:

$$\begin{cases} -\Delta u^\dagger(x) = f(x) & \text{for} \quad x \in [-1,1]^2 \\ u^\dagger(x) = g(x) & \text{for} \quad x \in \partial([-1,1]^2) \end{cases} \tag{A.63}$$

Place a Gaussian prior on the solution, i.e., assume $u^\dagger = \xi \sim \mathcal{N}(0, \kappa)$ where K is a Radial Basis Function (RBF) kernel.

Given N interior collocation points $X_i \in [-1,1]^2$ and N_b boundary collocation points $X_j^b \in \partial([-1,1]^2)$, we approximate the solution u with the Gaussian process estimator

$$\hat{u} = \mathbb{E}\left[\xi \,\middle|\, -\Delta\xi(X) = f(X),\, \xi(X^b) = g(X^b)\right] \tag{A.64}$$

To express $\hat{u}$, we extend the action of κ to distributions (linear functionals) as described in [241]: for two distributions d_1, d_2, we write

$$K(d_1, d_2) := \int d_1(x)\kappa(x, y)d_2(y)\, dx\, dy \tag{A.65}$$

Note that $K(\delta_x, \delta_y) = \kappa(x, y)$, while $K(\delta_x \circ \Delta, \delta_y) = \Delta_1\kappa(x, y)$ where Δ is the laplacian, and Δ_1 is the laplacian applied to the first variable.

Using this extended kernel, the Gaussian prior $\xi \sim \mathcal{N}(0, \kappa)$ can be interpreted in terms of distributions, allowing us to define $\xi(x) = \delta_x(\xi)$ and $\Delta\xi(x) = \delta_x \circ \Delta(\xi)$. In particular, this Gaussian prior implies that $\mathbb{E}\left[d_1(\xi)d_2(\xi)\right] = K(d_1, d_2)$. Defining the stack of distributions $\phi = (\delta_{X_1} \circ \Delta, .., \delta_{X_N} \circ \Delta, \delta_{X_1^b}, ..\delta_{X_{N_b}^b})$, we can define the kernel matrix

$$K(\phi, \phi)_{ij} = K(\phi_i, \phi_j) \tag{A.66}$$

and similarly, $K(x, \phi)_i = K(\delta_x, \phi_j)$. Using this, we have

$$\mathbb{E}\left[\xi \,\middle|\, -\Delta\xi(X),\, \xi(X^b)\right] = K(x, \phi)K(\phi, \phi)^{-1}\begin{pmatrix} -\Delta\xi(X) \\ \xi(X^b) \end{pmatrix} \tag{A.67}$$

In scenarios with multiple GP approximations of $u^\dagger$, leveraging the GP prior allows computation of correlations. Let

$$\hat{u}^{(k)} := \mathbb{E}\left[\xi \,\middle|\, -\Delta\xi(X^{(k)}),\, \xi(X^{b,k})\right] \tag{A.68}$$

be the distinct approximations of the solution of the PDE obtained from different interior and boundary collocation points $X^{(k)}$ and $X^{b,k}$. Write $\phi_k := (\delta_{X_1^k} \circ \Delta, .., \delta_{X_N^k} \circ \Delta, \delta_{X_1^{b,k}}, ..\delta_{X_{N_b}^{b,k}})$. Since these approximants $\hat{u}^{(k)}$, interpreted as models, are also Gaussian processes, we can explicitly compute their correlations:

$$\begin{aligned}\mathbb{E}\left[\hat{u}^{(k)}(x)\hat{u}^{(l)}(x)\right] &= K(x, \phi_k)K(\phi_k, \phi_k)^{-1}\mathbb{E}\left[\begin{pmatrix} -\Delta\xi(X^k) \\ \xi(X^{b,k}) \end{pmatrix}\begin{pmatrix} -\Delta\xi(X^l) \\ \xi(X^{b,l}) \end{pmatrix}^T\right] \\ &\quad \times K(\phi_l, \phi_l)^{-1}K(\phi_l, x) \\ &= K(x, \phi_k)K(\phi_k, \phi_k)^{-1}K(\phi_k, \phi_l) \\ &\quad \times K(\phi_l, \phi_l)^{-1}K(\phi_l, x).\end{aligned}$$

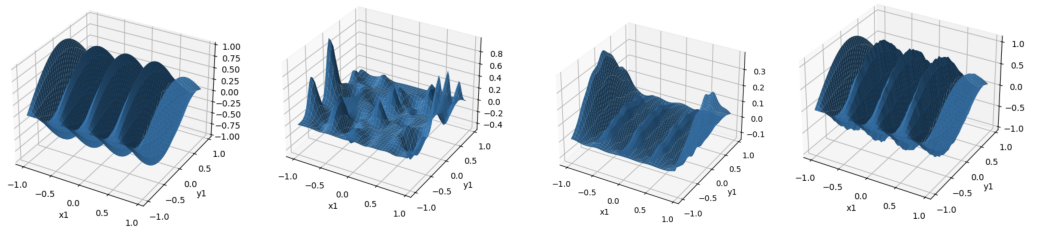
Similarly,

$$\mathbb{E}\left[\xi(x)\hat{u}^{(k)}(x)\right] = K(x, \phi_k)K(\phi_k, \phi_k)^{-1}K(\phi_k, x).$$

We then use these correlations in (2.3) to compute the aggregation analyzed in the following subsection.

A.2.2 Performance of the aggregation

Using the GP-based PDE-solver introduced in Chen et al. [56] and described above, we generate 100 distinct low-fidelity models of the solution of the PDE (A.63), an example of which is given in figure A.1b. As this method is GP-based, the correlations outlined in equation (2.3) are known, and this approach reduces to nested kriging [281]. In this context, (2.3) is not only the best (in mean squared error) linear aggregate. It is also the best non-linear aggregate of all models. Figure A.1 shows the results of this experiment. While the individual models are far from approximating the solution due to the limited number of collocation points, the aggregation shown in Figure A.1d closely matches the true solution and significantly surpasses any single model used in the aggregation. We also observe that mere averaging across all models does not yield the same accuracy level. This highlights that significant gains in aggregation performance can be achieved by optimally incorporating covariance information both within and across models relative to the target.



(a) (b) (c) (d)

Figure A.1: (a) Real solution of the PDE (b) One of the models aggregated (c) Uniform average of all models (d) Proposed aggregate (2.3)

A.3 Another Pathological example: Unknown region

In this second example illustrated in Figure A.2 we select

$$\begin{cases} Y(x) &= 3\cos(2\pi x) \\ M_G(x) &= Y(x) + \epsilon(x) \quad \text{where } \epsilon(x) \sim \mathcal{N}(0, 0.3) \\ M_B(x) &= 3 \\ M_N(x) &= -3 \end{cases} \tag{A.69}$$

Write $M = (M_G, M_B, M_N)^T$ for the vector defined by the three models. It is common for aggregation methods, such as MoE, to constrain $\hat{\alpha}$ to be a convex combination, which we implement here by seeking an aggregate of the form $M_A(x) = Softmax(\hat{v}(x))^T M(x)$ where $Softmax(\hat{v}(x))_k := \frac{\exp(\hat{v}_k(x))}{\sum_{j=1}^{3}\exp(\hat{v}_j(x))}$. To identify the

functions $\hat{\nu}_k$ we sample $N = 100$ points X_i uniformly in $[-1, -0.5] \cup [0.5, 1]$ leaving the region $[-0.5, 0.5]$ with no data. We use a kernel method with an RBF kernel and loss 2.4, in which $\|\cdot\|_{\mathcal{H}}$ is the RKHS norm of κ. Using the representer theorem [301], we reduce this loss to a finite-dimensional non-linear optimization problem and solve it using Gauss-Newton iterations [350]. In this example, the good model is consistently better than the bad models over the entire dataset. Nevertheless, our aggregate fits the dataset perfectly by combining M_B and M_N instead of using M_G. In doing so, it fails to learn the quality of the models. Instead, it only interpolates the data, leading to poor performance outside the dataset despite having a model with good performance everywhere. This pathological behavior arises because the aggregate fails to account for model errors in its approximation of Y, even though it is a convex combination.

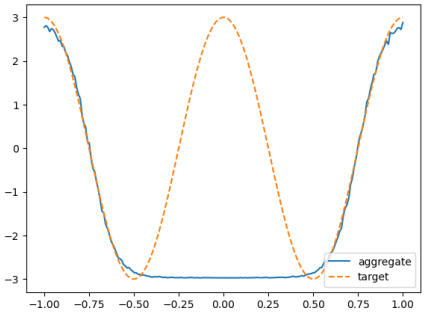


Figure A.2: Pathological example (sec. 2.2.2)

A.4 Non-independent model errors

A.4.1 New formulas for the aggregation

In section 2.3.3, we assume the models have independent errors. In general, this may not be true. Thus, we present here an extension to non-independent errors. Recall our approximation of the covariance matrix $A(x)$:

$$\hat{A}(x) = P^T \begin{pmatrix} e^{\lambda_1(x)} & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & 0 & e^{\lambda_n(x)} \end{pmatrix} P \quad \text{where} \begin{cases} PP^T = I_n \\ \lambda_i \text{ are regressors} \end{cases} \tag{A.70}$$

We may adapt the loss (2.13) to account for $P \neq I_n$, to get:

$$\lambda = \arg\min_{l \in \mathcal{H}} \sum_{i=1}^{N} \|P^T Diag(e^{l_k(X^i)})P - e^i (e^i)^T\|_F^2 + a\|l\|_{\mathcal{H}}^2 \tag{A.71}$$

$$= \arg\min_{l \in \mathcal{H}} \sum_{i=1}^{N} \sum_{k=1}^{n} \left(e^{l_k(X^i)} - (Pe^i)_k^2 \right)^2 + a\|l\|_{\mathcal{H}}^2 \tag{A.72}$$

With this formulation, we get a more general aggregation. In particular, despite summing to 1, the coefficients $\alpha(x)$ are not necessarily positive, giving added flexibility in case models have negative conditional correlations. The full Minimal Empirical Variance Aggregate (MEVA) with non-independent errors is given as:

$$\lambda = \arg\min_{l \in \mathcal{H}} \sum_{i=1}^{N} \sum_{k=1}^{n} \left(e^{l_k(X^i)} - (Pe^i)_k^2 \right)^2 + a\|l\|_{\mathcal{H}}^2 \tag{A.73}$$

$$\alpha(x)^T = \frac{\mathbb{1}^T P^T D(x) P}{\mathbb{1}^T P^T D(x) P \mathbb{1}} \text{ where } D(x) = Diag\left[\exp(-\lambda_k(x)), k = 1, ..., n\right] \tag{A.74}$$

$$M_A(x) = \sum_{i=1}^{n} \alpha_i(x) M_i(x) \tag{A.75}$$

A.4.2 Choosing the matrix P

We have explored two options for the matrix P:

- $P = I$, the identity matrix, is the simplest choice.
- P as the orthonormal basis that diagonalizes the empirical covariance matrix $C := \frac{1}{N} \sum_{i=1}^{N} e_i e_i^T$.

In the experiments presented below, we only show results for $P = I$. This choice is straightforward and delivers accuracy comparable to or better than using the eigenvectors of C.

We also observed that using the empirical covariance matrix can render the method unstable if the models are highly correlated. Indeed, consider, for instance, two models whose errors are significantly correlated, leading to the following empirical covariance matrix:

$$C := \begin{pmatrix} 1.1 & 0.9 \\ 0.9 & 1 \end{pmatrix} \tag{A.76}$$

In this case, the second eigenvector of C, $P_2 \approx \frac{1}{\sqrt{2}}(0.97, -1.02)$, results in $\mathbb{1}^T P_2 \approx -0.04$, which effectively subtracts the two models. This eigenvector is associated with a small eigenvalue (0.15), indicating that the difference between the two models has a low variance on average. Since equation (2.7) depends on the inverse of the estimated covariance matrix, a small estimated variance eigenvalue $e^{\lambda_2(x)}$ leads to $\alpha(x)$ having large coefficients with opposing signs. In this scenario, we observed that the aggregation method can become sensitive to small errors and thereby suffer from poor generalization.

A.5 An alternative, direct Minimal Empirical Variance Aggregation loss

Here, we propose an alternate end-to-end loss for the minimal empirical variance aggregation (MEVA). In section 2.3.1, we observed that our approach can be interpreted as minimizing the aggregate's variance, using an unbiased combination α s.t. $\mathbb{1}^T\alpha = 1$. Under the model (2.6), $Cov[M(x)] = A(x)$, so our MVA is such that

$$\alpha(x) = \arg\min_{\nu\in\mathbb{R}^n \sum_{i=1}^n \nu_i=1} \nu^T A(x)\nu \tag{A.77}$$

Instead of estimating $A(x)$ and inverting this estimation to obtain the minimal variance aggregate as in (2.7), we propose using an empirical version of the above loss. Specifically, let the empirical covariance matrices A_i be:

$$A_i := \begin{cases} P^T Diag((Pe^i)_k^2, k = 1..,n)P & \text{for general } P \\ Diag((Y^i - M_k(X^i))^2, k = 1..,n) & \text{if } P = I_n \end{cases} \tag{A.78}$$

Write $\mathcal{H}_{\mathbb{1}}$ for a normed subspace of the space of unbiased aggregators $\{u : x \mapsto u(x) \in \mathbb{R}^n \text{ s.t. } \sum_{i=1}^n u_i(x) = 1, \forall x\}$. Then, the proposed alternative approach identifies an aggregator $\tilde{\alpha}$ by minimizing the sample variance of the aggregate:

$$\tilde{\alpha} = \arg\min_{u\in\mathcal{H}_{\mathbb{1}}} \sum_{i=1}^N u(X^i)^T A_i u(X^i) + a\|u\|^2_{\mathcal{H}_{\mathbb{1}}} \tag{A.79}$$

We employ this strategy in the Boston dataset example (section 2.4.1). Here, $\mathcal{H}_{\mathbb{1}}$ is a set of vector-valued functions parameterized by a neural network, where the output is constrained to have positive entries summing to one via a softmax layer. The term $\|u\|^2_{\mathcal{H}_{\mathbb{1}}}$ is an L^2-norm regularizer applied to the network's weights and biases. This method is both straightforward and practical.

A.6 Adding a bias correction

Assuming the models $M(x)$ to be unbiased is necessary to obtain a linear aggregate, as adding a bias correction will result in an affine aggregation of the form $M_A(x) = \alpha^T(x)M(x) + \beta(x)$. This assumption may, however, be violated in practice, and in this case, bias will be interpreted as variance. In a context with limited data and little knowledge of the models and target, it would be difficult to know if an error can be attributed to bias or variance. In the case where it is clear models are biased and interpreting it as variance hurts accuracy, we may modify our method by accounting for the bias

$$M(x) = Y(x)\mathbb{1} + \tilde{Z} \text{ where } \begin{cases} \mathbb{E}[\tilde{Z}(x)] = \mu(x) \\ \text{Cov}[\tilde{Z}(x)] = A(x) \end{cases} \tag{A.80}$$

In such a model, we must, in addition to λ, train μ. Using the definition of $\tilde{Z}$, we know the loss for μ is of the form $\sum_{i=1}^{N}(\mu(X^i) - e^i)^T A^{-1}(X^i)(\mu(X^i) - e^i)$. Taking $P = I$, this bias-corrected aggregation would be:

$$\tilde{\lambda}, \tilde{\mu} = \arg\min_{l,m \in \mathcal{H}_1 \times \mathcal{H}_2} \sum_{i=1}^{N} \sum_{k=1}^{n} \Big[\left(e^{l_k(X^i)} - (e^i_k - m_k(X^i))^2\right)^2 + e^{-l_k(X^i)} \left(m_k(X^i) - e^i_k\right)^2 \Big] + a\|l\|^2_{\mathcal{H}_1} + b\|m\|^2_{\mathcal{H}_2} \tag{A.81}$$

$$\tilde{\alpha}(x) = Softmax(-\tilde{\lambda}(x)) \tag{A.82}$$

$$\tilde{M_A}(x) = \sum_{i=1}^{n} \tilde{\alpha}_i(x)(M_i(x) - \tilde{\mu}_i(x)) \tag{A.83}$$

where $\mathcal{H}_1, \mathcal{H}_2$ are two spaces of functions used to approximate the log-eigenvalues of $A(x)$ and the bias of the models.

In the Boston housing dataset example in section 2.4.1, we tried to implement this bias-corrected aggregation, and it did not yield better results than the simpler method MEVA. While we do not prescribe its use in general, we did not consider this model in the rest of the paper.

A.7 Aggregation using vector-valued Gaussian processes and matrix-valued kernels

We will now model aggregation as vector-valued Gaussian process regression to elucidate the pathologies observed in the two abovementioned examples. We start with a brief reminder on this vector/matrix-valued generalization of GPs/kernels [8], before computing the MEA in the GP context.

A.7.1 Vector-valued Gaussian process and matrix-valued kernel

This section is a reminder on vector/matrix-valued GPs/kernels [8]. We define a matrix-valued kernel as a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{n \times n}$ such that,

- $\forall x, x' \in \mathcal{X}, K(x, x') = K(x', x)^T$
- $\forall x_1, .., x_N \in \mathcal{X}, \forall y_1, .., y_N \in \mathbb{R}^n$,

$$\sum_{1 \leq i,j \leq N} y_i^T K(x_i, x_j) y_j \geq 0 \tag{A.84}$$

This generalizes the positivity and symmetry conditions for defining a kernel. Each matrix-valued kernel uniquely defines a zero-mean vector-valued Gaussian process ξ such that :

- $\forall x \in \mathcal{X}, \xi(x) \sim \mathcal{N}(0, K(x,x))$
- $\forall x, x' \in \mathcal{X}, Cov\left[\xi(x), \xi(x')\right] = K(x,x')$

One interesting example of such a vector-valued Gaussian process is the case of n independent Gaussian processes, corresponding to K being diagonal. One can then understand general vector-valued GP as adding correlation to the coordinates, each being its own GP.

A.7.2 MEA with matrix-valued kernels

We choose a matrix-valued kernel $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}^{n\times n}$, and write $\mathcal{H}_K$ for its associated Reproducing Kernel Hilbert Space (RKHS). We get the MEEA by solving (2.4) (see Appendix A.8):

$$M_A(x) = \hat{\alpha}_E(x)^T M(x) = \tilde{k}(x, X)(\tilde{k}(X, X) + \lambda I)^{-1} Y \tag{A.85}$$

where $\tilde{k}(x,y) = M(x)^T K(x,y) M(y)$. This is also the GP regressor

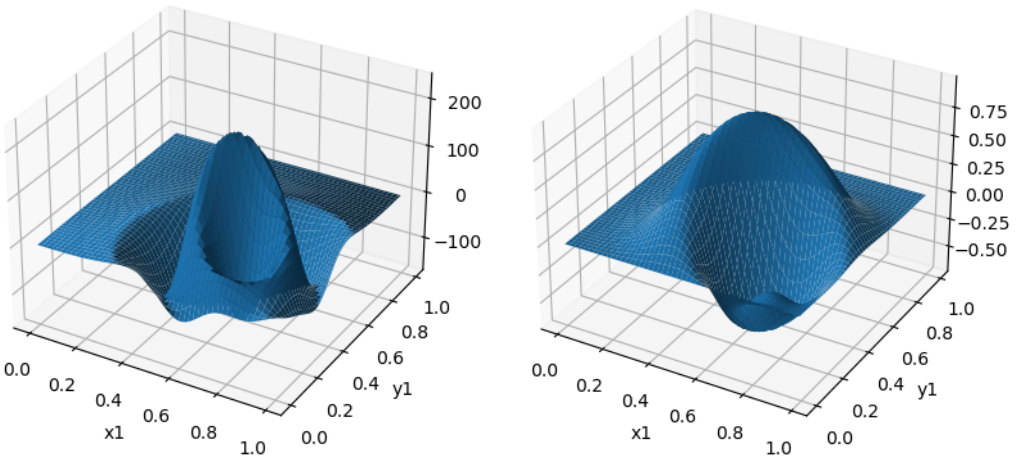
$$M_A = \mathbb{E}\left[\xi \,|\xi(X) = Y + \mathcal{N}(0, \lambda I)\right]$$

with prior $\xi \sim \mathcal{N}(0, \tilde{k})$. This observation is crucial as it shows that the model values are used as a feature for the regression, as expected. However, it demonstrates that the Mean Empirical Error Aggregate (MEEA) only tries to regress Y without considering the underlying models' accuracy. This observation is counterintuitive as we expect model aggregation to leverage the individual models to approximate the target function better rather than merely regressing directly to the target. This behavior is particularly evident in pathological examples, illustrating the limitations of the direct regression approach.

A.8 Minimizing the loss in section A.7

We aim to minimize the following optimization problem:

$$\hat{\alpha} = \arg\min_{\alpha\in\mathcal{H}_K} \frac{1}{N}\sum_{i=1}^{N}\left[Y^i(x_i) - \alpha(x_i)^T M^i(x_i)\right]^2 + \gamma\|\alpha\|^2_{\mathcal{H}_K} \tag{A.86}$$



(a) f (b) u

Figure A.3: A pair (f, u) s.t. $-\Delta u = f$ and $u(\partial\Omega) = 0$, sampled using (2.19)

Applying the representer theorem, we deduce that the solution $\hat{\alpha}$ can be represented as $\hat{\alpha} = (K(\cdot, x_1), \ldots, K(\cdot, x_N))V$, where $V \in \mathbb{R}^{nN}$. Additionally, we define the matrix $\mathcal{M} \in \mathbb{R}^{N \times nN}$ as follows:

$$\mathcal{M} = \begin{pmatrix} M_1 \\ \vdots \\ M_N \end{pmatrix}, \tag{A.87}$$

where each $\mathcal{M}_i = (M(x_i)^T K(x_i, x_1), \ldots, M(x_i)^T K(x_i, x_N))$. The optimal vector V is then given by:

$$V = \arg\min_{v \in \mathbb{R}^{nN}} \|Y - \mathcal{M}v\|^2 + \lambda v^T \mathcal{K} v \tag{A.88}$$

where $\mathcal{K}$ is a block matrix such that $\mathcal{K}_{ij} = K(x_i, x_j)$ for $1 \leq i, j \leq N$. Define the matrix $D \in \mathbb{R}^{N \times nN}$ as:

$$D = \begin{pmatrix} M(x_1)^T & 0 & \ldots & 0 \\ 0 & M(x_2)^T & \ldots & \ldots \\ \vdots & \ldots & \ddots & \vdots \\ 0 & \ldots & \ldots & M(x_N)^T \end{pmatrix} \tag{A.89}$$

Observe that $\mathcal{M} = D\mathcal{K}$. Using the matrix identity:

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = P B^T (B P B^T + R)^{-1} \tag{A.90}$$

where $P^{-1} = \lambda K$, $B = \mathcal{M}$, and $R = I$, we derive:

$$V = \frac{1}{\lambda} K^{-1} \mathcal{M}^T \left(\frac{1}{\lambda} \mathcal{M} K^{-1} \mathcal{M}^T + I \right)^{-1} Y \tag{A.91}$$

$$= D^T (D \mathcal{K} D^T + I)^{-1} Y \tag{A.92}$$

It follows that $(D\mathcal{K}D^T)_{ij} = M^T(x_i) K(x_i, x_j) M(x_j) = \tilde{k}(x_i, x_j)$. Thus, the final model prediction at a new point x is given by:

$$M_A(x) = M(x)^T K(x, X) V \tag{A.93}$$

where $K(x, X) = (K(x, x_1), \ldots, K(x, x_N))$ so that $\hat{\alpha}(x) = K(x, X)V$. By identifying that $M(x)^T K(x, X) D^T = \tilde{k}(x, X)$, we can conclude:

$$M_A(x) = \tilde{k}(x, X)(\tilde{k}(X, X) + \lambda I)^{-1} Y \tag{A.94}$$

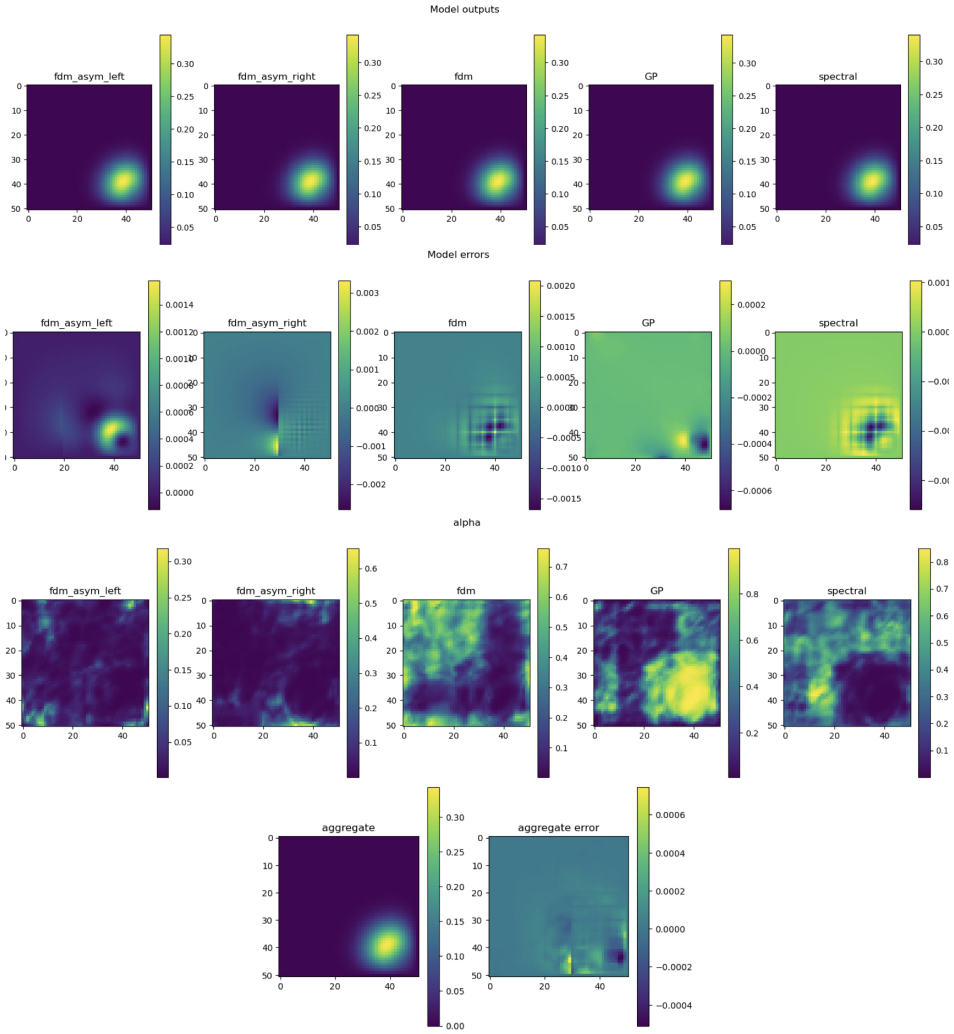


Figure A.4: Example of aggregation for the Laplace equation (section 2.4.3). Line 1: Outputs of the different models for a given input f. Line 2: Prediction errors for each model. Line 3: Values of α. Line 4: Aggregate and its error

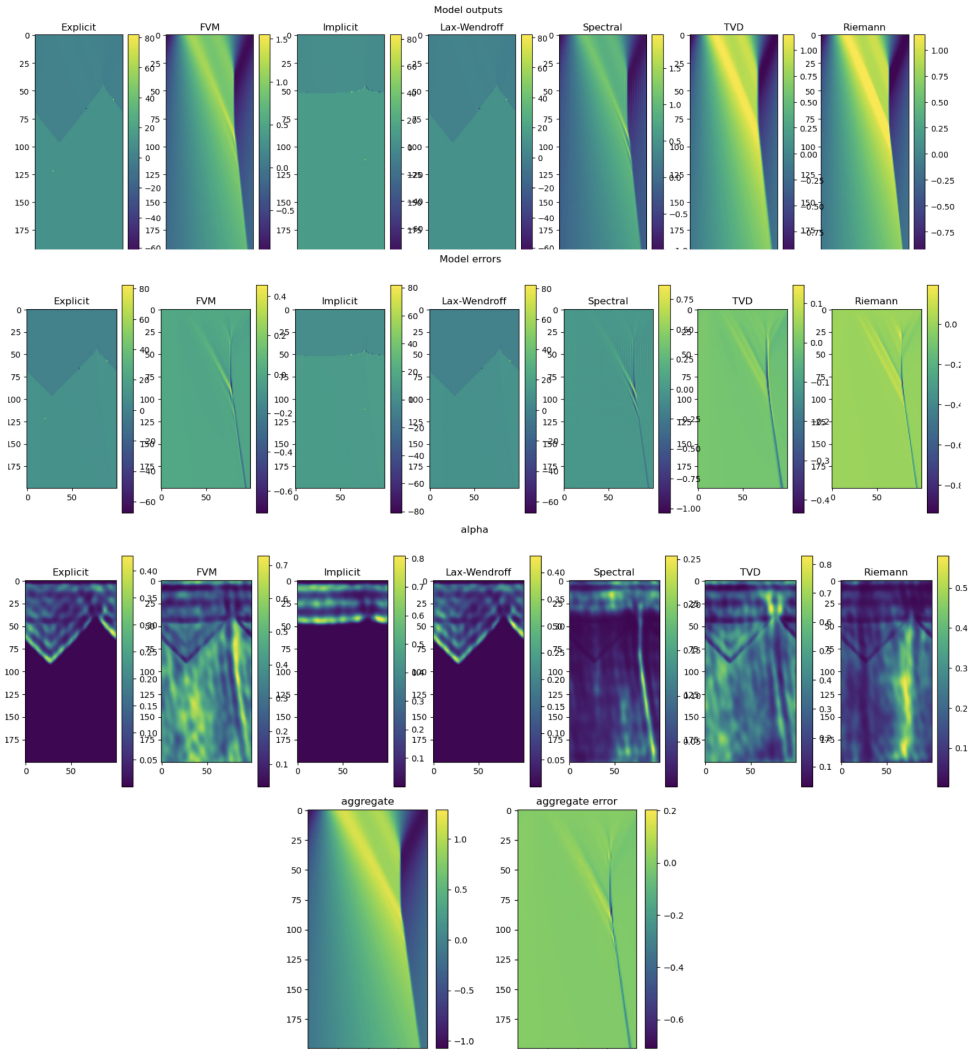


Figure A.5: Example of aggregation for Burgers' equation (section 2.4.4). Line 1: Outputs of the different models for a given input f. Line 2: Prediction errors for each model. Line 3: Corresponding values of α. Line 4: Aggregate and its error. Note that the explicit, implicit, and Lax-Wendroff methods all diverged, while the spectral method exhibited spurious oscillations. In contrast, the aggregated result avoids these issues.

Appendix B

APPENDIX TO CHAPTER 3: OPERATOR LEARNING AT MACHINE PRECISION

B.1 Additional Benchmarks

Table B.1 provides an inference time comparison between the benchmark models and our CHONKNORIS models. Table B.2 provides an additional dataset generation and GP model fitting time comparison across seismic imaging resolutions. While classic operator learning methods require only a single model evaluation per prediction, CHONKNORIS requires separate evaluations at each iteration of the Newton–Kantorovich method. CHONKNORIS is able to achieve orders of magnitude better accuracy, as evidence in Table 3.1, but requires more model evaluations and greater inference time. For inverse problems, the total inference time is often longer than the total time spend evaluating the model due to costly Jacobian evaluations. Figure B.1 reinforces this insight by comparing the error against inference time for the seismic imaging problem.

The Gaussian process (GP) benchmarks in Table 3.1 and Table B.1 are based on [27]. The GP measurements are given by either pointwise measurements or projection onto PCA coefficients, and the intermediate vector valued map is learned using a Gaussian process with a linear combination of a Matérn and dot product kernels. All hyper-parameters (number of PCA coefficients, lengthscale, regularization...) are learned via 5 fold cross validation on the training set and optimized using Optuna [4].

B.2 Nested Kriging

Nested Kriging is a hierarchical extension of Gaussian process regression designed to handle very large datasets that cannot be represented efficiently by a single GP. The idea is to train separate specialized GP models (experts) on subsets of the data or related problems, and then combine (aggregate) their predictions using another GP or a statistically optimal weighting scheme. This aggregation step, often based on minimizing the overall prediction variance, produces a global predictor that retains much of the accuracy of a full GP model while drastically reducing computational and memory costs.

	GP	FNO	TNO	CHONKNORIS (ours)		
	1 eval	1 eval	1 eval	1 eval	# evals	inference total
*Nonlinear elliptic	1.2e-3	3.4e-3	1.3e-03	1.1e-3	5	1.5e-2
*Burgers'	6.7e-4	3.5e-3	1.2e-3	2.3e-4	15×150	4.4e0
*Nonlinear Darcy	1.1e-3	2.8e-3	1.2e-3	3.3e-3	5000	1.7e1
Calderòn	1.5e-3	–	–	2.2e-3	1500	1.5e2
Inverse wave scattering	1.7e-3	–	–	1.1e-3	2000	1.6e2
*Seismic imaging 5×5	1.2e-3	–	–	1.4e-3	100	2.1e0
*Seismic imaging 7×7	1.2e-3	–	–	1.5e-3	2500	8.1e1
*Seismic imaging 10×10	1.3e-3	–	–	2.0e-3	25000	1.0e3

Table B.1: Inference time (seconds) across multiple forward and inverse problem benchmarks. Problems with an * were run on a single GPU (NVIDIA A100), with other problems run on the CPUs (Intel(R) Xeon(R) Platinum 8380 CPUs @ 2.30GHz).

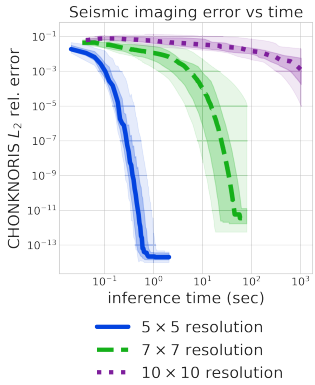


Figure B.1: Comparison of L_2 relative errors versus inference time for the seismic imaging full waveform inversion problem in Section 3.3.4.3.

resolution	time per NK iter.	# NK iter.	total NK time	small CHONKNORIS-GP fit time
5×5	0.64	20	13	38
7×7	0.80	25	20	137
10×10	1.0	35	35	257
14×14	1.6	95	152	–

Table B.2: Seismic imaging full waveform inversion (FWI) dataset generation and model fitting times at different resolutions. We record the time per Newton–Kantorovich (NK) iteration, the median number of iterations required for convergence, the total NK dataset generation time for a single realization, and the time required to fit a small version of the CHONKNORIS-GP model (including PyTorch compilation time). All times are in seconds. We emphasize the dataset generation times use the NK methods with the full Cholesky decomposition. Compared to the small CHONKNORIS-GP models fit here, the full CHONKNORIS-GP models we deploy in inference use between two and three times as much data and between five and ten times as many GP optimization iterations depending on the resolution. All timings in this table were carried out on a single NVIDIA A100 GPU.

More formally, let $M_1(x), M_2(x), \ldots, M_p(x) \in \mathbb{R}^m, x \in \mathbb{R}^n$ be a set of $p \in \mathbb{N}$ experts that are described by the same underlying Gaussian process $Y(x)$ with kernel k, i.e., there holds

$$M_i(x) = k(x, X_i)k(X_i, X_i)^{-1}Y(X_i), \quad i = 1, 2, \ldots, p.$$

Furthermore, define $M(x) = (M_1(x), M_2(x), \ldots, M_p(x))^T$ as well as the covariance matrix $K_M(X) = \mathrm{Cov}(M(x), M(x))$ and the vector $k_M(x) = \mathrm{Cov}(Y(x), M(x))$. Then, the aim is to minimize the variance

$$E[(Y(x) - \alpha^T M(x))^2] = k(x, x) - 2\alpha^T k_M(x) + \alpha^T K_M(x)\alpha$$

with respect to the aggregation vector α. The solution to this mean squared problem is given by the

$$\alpha^* = K_M(x)^{-1}k_M(x)$$

leading to the unconstrained best linear unbiased predictor (BLUP) given by

$$M_{\mathcal{A}}(x) = K_M(x)^{-1}k_M(x)M(x).$$

However, we aim to aggregate our models by weighting the results of the individual experts by incorporating the constraint $\sum_{i \in A} \alpha_i(x) = \mathbf{1}^T\alpha(x) = 1$ for all $x \in \mathbb{R}^n$, where $\mathbf{1}$ is a vector of ones with the same dimension as α. The constrained optimization problem is then solved via Lagrange multipliers. The corresponding Lagrange function is given by

$$\begin{aligned} \mathcal{L}(\alpha, \lambda) &= E[(Y(x) - \alpha^T M(x))^2] + \lambda(\mathbf{1}^T\alpha - 1) \\ &= k(x, x) - 2\alpha^T k_M(x) + \alpha^T K_M(x)\alpha + \lambda(\mathbf{1}^T\alpha - 1) \end{aligned} \tag{B.1}$$

where λ is the Lagrange multiplier associated with the constraint $\mathbf{1}^T\alpha = 1$. This gives the following result proved in Appendix B.8

Proposition B.2.1. *The Lagrange function given by* (B.1) *is minimized by*

$$\alpha = K_M(x)^{-1}\left(k_M(x) - \frac{\mathbf{1}\left(\mathbf{1}^T K_M(x)^{-1}k_M(x) - 1\right)}{\mathbf{1}^T K_M(x)^{-1}\mathbf{1}}\right). \tag{B.2}$$

Proof. To minimize $\mathcal{L}(\alpha, \lambda)$, we take the derivative of $\mathcal{L}$ with respect to α and λ, and set them equal to zero. This gives the equations:

$$K_M(x)\alpha = k_M(x) - \frac{\lambda}{2}\mathbf{1}, \quad \mathbf{1}^T\alpha = 1.$$

Substituting the α in the first equation into the second equation, solving for λ, and substituting λ back into the first equation gives the new values of α that satisfy the constraint $\sum_{i \in A} \alpha_i(x) = 1$. This finally yields the desired formula (B.2). □

By enforcing the constraint $\sum_{i \in A} \alpha_i(x) = 1$, the coefficients $\alpha_i(x)$ are adjusted from the unconstrained BLUP solution, and the Lagrange multiplier method gives us the corrected formula. This ensures that the coefficients sum to 1 while still minimizing the mean squared error in the best linear unbiased way.

Computational complexity

The computational cost of the Nested Kriging can be divided into two parts: the offline training of the individual experts and the online prediction (aggregation). During training, each expert M_i requires the inversion (or Cholesky factorization) of its covariance matrix $K_i \in \mathbb{R}^{n_i \times n_i}$, resulting in a computational cost of order $O(n_i^3)$ and a memory requirement of $O(n_i^2)$. Training all p experts therefore costs $O\left(\sum_{i=1}^{p} n_i^3\right)$, which is significantly cheaper than training a single global Gaussian process on all $N = \sum_{i=1}^{p} n_i$ data points, whose cost would be $O(N^3)$. For experts of equal size ($n_i = n$), this corresponds to a reduction by a factor of approximately p^2.

At prediction time, computing the prediction $M_{\mathcal{A}}(x)$ for a new input x involves two main steps. First, for each expert, we compute the predictive weights $v_i(x) = K_i^{-1} k(X_i, x)$, which requires $O(n_i^2)$ operations per expert, yielding a total cost of $O\left(\sum_{i=1}^{p} n_i^2\right)$. These vectors are then used to construct the aggregated covariance terms: $k_M(x) \in \mathbb{R}^p$ and $K_M(x) \in \mathbb{R}^{p \times p}$, where forming all pairwise covariances typically costs $O(N^2)$ if cross-covariances between experts are precomputed. Finally, the aggregation weights $\alpha(x)$ are obtained by inverting $K_M(x)$, which adds an additional $O(p^3)$ cost. Consequently, the total online prediction complexity per test point is of order $O(N^2 + p^3)$, compared to $O(N^2)$ for a single global Gaussian process.

Hence, Nested Kriging achieves a substantial reduction in offline training cost—from cubic in the total dataset size to the sum of the individual cubic costs—while maintaining comparable prediction complexity and offering improved scalability with respect to both memory and computation.

B.3 Choice of ordering in the Cholesky Factorization

The Cholesky factors of Q depend on the ordering of its rows and columns, which corresponds to the degrees of freedom in $\mathbb{R}^N$. While this ordering can be inherited from the discretization of the space $\mathcal{V}$, the accuracy of the approximation in (3.9) can be significantly improved by selecting an ordering that induces a hierarchical or multiresolution structure in the operator problem [238, 241]. To describe this, consider first the case where $\mathcal{F}$ is a local differential operator (e.g., a PDE) acting on functions defined over a domain $\Omega \subset \mathbb{R}^d$. Here, $\frac{\delta \mathcal{F}}{\delta v}^* \frac{\delta \mathcal{F}}{\delta v} + \lambda I$ can be interpreted as a (discretized) elliptic PDE, and its inverse, Θ, represents a (discretized) Green's function. Suppose the discretization of $\mathcal{F}$ is obtained using a numerical method (e.g., finite element, finite difference, or collocation), with elements centered around points $x_1, \ldots, x_N \in \Omega$. Although the Cholesky factors of Θ are generally dense when using a lexicographic ordering, they exhibit significant sparsity when a *max-min ordering* is used. As presented in [294], a max-min ordering π of $\{1, \ldots, N\}$ is defined such that:

$$\pi(1) = \text{argmax}_i \, \text{dist}(x_i, \partial\Omega) \,,$$

and for $i \geq 1$:

$$\pi(i+1) = \text{argmax}_j \, \text{dist}\left(x_j, \partial\Omega \cup \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}\right) .$$

Writing Π for the permutation matrix associated with π and

$$\Pi^T L^\dagger L^{\dagger,T} \Pi = \Theta$$

for the exact Cholesky factorization of Θ in the maxmin ordering, those Cholesky factors have two desirable properties [59, 292, 294]: (1) they are rank revealing, and (2) they are sparse.

B.3.0.1 Rank-Revealing Property

The matrix $L^{\dagger,(k)}$, defined as the truncation of $L^\dagger$ to its first k columns (with all remaining columns set to zero), provides a near-optimal rank-k approximation (measured in operator norm) within a constant factor [294, Thm. 2.3]. Specifically, it satisfies:

$$\left\|\Theta - \Pi^T L^{\dagger,(k)} L^{\dagger,(k),T}\right\|_{\text{Fro}} \lesssim \inf_{M \text{ of rank } k} \left\|\Theta - M\right\|_{\text{Fro}},$$

where $\|\cdot\|_{\text{Fro}}$ denotes the Frobenius matrix norm. This rank-revealing property is: (a) Analogous to the properties achieved with gamblets or operator-adapted wavelets

[238, 241] for the compression of elliptic PDEs. (b) more desirable for operator-learning than mesh invariance because it induces, within a constant factor, an optimal approximation of the underlying operator at any level of truncation/discretization which has been the core objective of numerical homogenization [7].

B.3.0.2 Sparsity

The Cholesky factors $L^\dagger$ exhibit exponential decay in their entries, as shown in [294, Thm. 5.23]. Specifically:

$$|L^\dagger_{i,j}| \lesssim \exp\left(-C\frac{\text{dist}(x_{\pi(i)}, x_{\pi(j)})}{h}\right),$$

where h is the mesh norm associated with $\{x_1, \ldots, x_N\}$, and C is some positive constant. To describe the sparsity structure, let $l_1 := \max_i \text{dist}(x_i, \partial\Omega)$ and, for $i \geq 1$,

$$l_{i+1} := \max_j \text{dist}\left(x_j, \partial\Omega \cup \{x_{\pi(1)}, \ldots, x_{\pi(i)}\}\right),$$

represent the sequence of decreasing distances associated with the max-min ordering. Given $\rho \in \mathbb{N}$, the sparsity set is defined as:

$$S_\rho := \left\{(i,j) \in \{1,\ldots,N\}^2 \mid i \geq j \text{ and } \text{dist}(x_{\pi(i)}, x_{\pi(j)}) \leq \rho l_i\right\}.$$

The set S_ρ contains approximately $\mathcal{O}(N\rho^d)$ elements. Using an incomplete Cholesky factorization with sparsity pattern S_ρ, one obtains a lower triangular matrix L_ρ with non-zero entries restricted to S_ρ. This approach achieves the approximation:

$$\left\|\Pi^T L_\rho L_\rho^T \Pi - \Theta\right\|_{\text{Fro}} \lesssim e^{-C\rho}.$$

Consequently, it is possible to achieve an accuracy ϵ by accessing only $\mathcal{O}(N \log^d \frac{1}{\epsilon})$ entries of Θ. The resulting Cholesky factors L_ρ will have just $\mathcal{O}(N \log^d \frac{1}{\epsilon})$ non-zero entries.

B.3.0.3 Exploiting the maxmin ordering for operator learning

The rank revealing and sparsity properties of the maxmin ordering have been leveraged in [291] for achieving SOTA in terms of both complexity and data efficiency for learning the solution operator of arbitrary elliptic PDEs ([291] shows that those solution operators can be rigorously approximated to accuracy ϵ from only $\mathcal{O}(\log N \log^d \frac{N}{\epsilon})$ source-solution pairs). Here we employ the maxmin ordering to improve the accuracy of the map $v \to L(v)$ such that

$$\Pi^T L L^T \Pi \approx \Theta\,. \tag{B.3}$$

B.3.0.4 Exploiting Random Ordering for Operator Learning

The rank-revealing and sparsity properties of the max-min ordering can also be achieved using a random ordering, as it implicitly induces a max-min structure (see [238, Thm. 2.9], derived from [135]). In cases where $\mathcal{F}$ is high-dimensional or represents an arbitrary operator not necessarily associated with a PDE, the rank-revealing property is still preserved by the random ordering [57], therefore in those settings we can simply replace Π in (B.3) by random permutation matrix.

B.4 Discretizations and Cholesky factorization

In practice, we work with an arbitrary discretization of the operator $\mathcal{G}$, setting $\mathcal{U} = \mathbb{R}^J$ and $\mathcal{V} = \mathbb{R}^K$. This discretization can be derived from various numerical methods such as finite-element, or finite-difference methods or any other discretization method such as the optimal recovery approach in [27]. For simplicity, we retain the original notation for the discretized operators, continuing to use $\mathcal{G}$ and $\mathcal{F}$ to denote their discretized counterparts. The spaces $\mathcal{U}$ and $\mathcal{V}$ are equipped with the Euclidean inner product. Consequently, the derivative $\frac{\delta \mathcal{F}}{\delta v}$ becomes a function mapping $\mathbb{R}^J$ to the space of $K \times K$ matrices. The elliptic operator

$$\frac{\delta \mathcal{F}}{\delta v}^* \frac{\delta \mathcal{F}}{\delta v} + \lambda I \, : \, \mathcal{V} \to \mathcal{L}_+(\mathcal{V})$$

then simplifies to a map from $\mathbb{R}^J$ to the space of $J \times J$ symmetric positive definite matrices. Then learning the operator Q reduces to learning a mapping from $\mathbb{R}^J$ to the space of $J \times J$ symmetric positive definite matrices. To facilitate this, we represent Q through its Cholesky factorization:

$$Q = \mathcal{R}\mathcal{R}^T \, ,$$

and seek to learn a lower-triangular-matrix-valued operator:

$$\mathcal{R} : \mathcal{V} \to \{R \in \mathbb{R}^{J \times J} \mid R \text{ is lower triangular}\},$$

Then a neural network or matrix valued RKHS can be used to parameterize and learn $\mathcal{R}$. In practice we find that this parameterization of the DNO leads to higher accuracy as the composed operators will enforce positivity and symmetry of Q, however, one can design intermediate algorithms where Q or $\mathcal{N}$ are directly learned.

B.5 Choice of the step size α and relaxation λ

We often also found it necessary to adapt the relaxation λ across iterations for convergence to machine precision in a reasonable number of steps while maintaining

global convergence. Generally speaking, λ is decreased as our NK/CHONKNORIS approximation approaches the true solution in order to achieve the quadratic convergence of Gauss–Newton steps. In such scenarios, the learned Cholesky factor $\hat{\mathcal{R}}$ has an additional dependence on the relaxation λ. To simultaneously tune the step size α and relaxation λ, at any given iteration we may choose to inflate, deflate, or keep both parameters from the previous iteration. Specifically, if (λ_n, α_n) are the values at iteration n, then we choose $\lambda_{n+1} \in \{\beta_\lambda \lambda_n, \lambda_n, \kappa_\lambda \lambda_n\}$ and $\alpha_{n+1} \in \{\beta_\alpha \alpha_n, \alpha_n, \kappa_\alpha \alpha_n\}$ among all 9 possible combinations to minimize the loss associated to the computed step, where $\kappa_\lambda, \kappa_\alpha \in (0, 1)$ are decay factors and $\beta_\lambda, \beta_\alpha \in (1, \infty)$ are inflation factors. The choices are visualized in Figure B.2with $\kappa_\lambda = \kappa_\alpha = 1/2$ and $\beta_\lambda = \beta_\alpha = 2$. While we found this inflation/deflation 9-option scheme provided robust performance in our numerical experiments, other schemes based comparing observed residuals to linearized residuals have been proposed for parameter tuning, e.g., [151, Appendix c.2.1] provides a direct and automatic rule to select λ, see also [13, 113]. Detailed convergence theory for the LM method can be found in [103], including the effects of inexact steps where convergence hinges on sufficiently reducing the loss on each iteration. Our direct approach of choosing $(\alpha_{n+1}, \lambda_{n+1})$ from the 9 options so as to minimize the loss naturally allows for conservative steps (large λ, small α) when either the error in $\widehat{R}$ is large or the linearization is inaccurate, and aggressive steps (small λ, $\alpha \to 1$), when the error in $\widehat{R}$ is small and $\mathcal{F}$ is near linear in a neighborhood containing both the current iterate and solution.

B.6 Approximation of the Fréchet derivative

In this section, we discuss some numerical approximations of the Jacobian in cases when the explicit computation and storage of the Jacobian becomes infeasible, e.g., for high-dimensional problems such as problems in seismic imaging cf. Section 3.3.4.3. The idea is to approximate the Fréchet derivative of $\mathcal{F}$ by the finite difference

$$\frac{\delta \mathcal{F}}{\delta v}(u, v)[h] \approx \frac{\mathcal{F}(u, v + th) - \mathcal{F}(u, v)}{t},$$

for sufficiently small $t > 0$ justified by the limit

$$\frac{\delta \mathcal{F}}{\delta v}(u, v)[h] = \lim_{t \searrow 0} \frac{\mathcal{F}(u, v + th) - \mathcal{F}(u, v)}{t}.$$

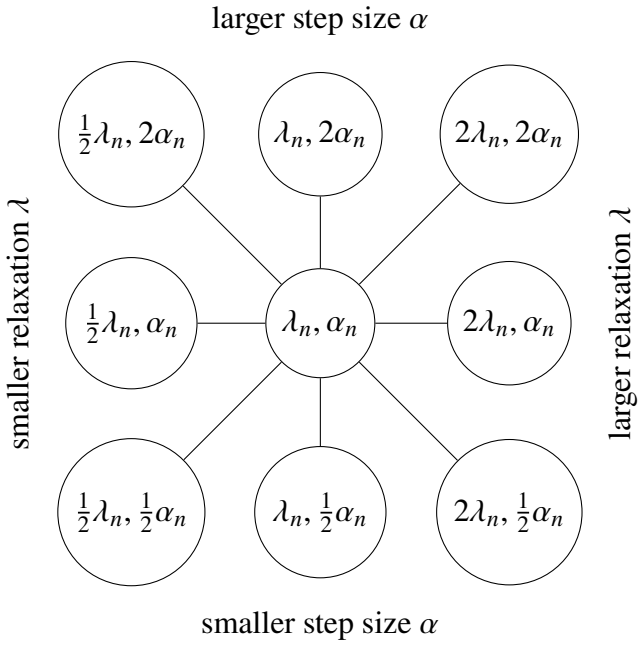


Figure B.2: Given the relaxation λ_n and step size α_n at iteration n, the next parameters $(\lambda_{n+1}, \alpha_{n+1})$ are chosen from the above 9 options to minimize the loss. The deflation factors $1/2$ and inflation factors 2 can be replaced by more general factors, see the discussion in Appendix B.5.

We obtain for all $h \in \mathcal{V}$

$$\begin{aligned}
&\left\langle \left(\frac{\delta \mathcal{F}}{\delta v}(u,v)\right)^* \mathcal{F}(u,v), h \right\rangle_{\mathcal{V}} \\
&\quad = \left\langle \mathcal{F}(u,v), \frac{\delta \mathcal{F}}{\delta v}(u,v) h \right\rangle_{\mathcal{W}} \\
&\quad \approx \frac{1}{t} \left\langle \mathcal{F}(u,v), \mathcal{F}(u, v+th) - \mathcal{F}(u,v) \right\rangle_{\mathcal{W}} . \qquad \text{(B.4)}
\end{aligned}$$

Denoting again with $\mathbb{R}^{N_{\mathcal{V}}}$ the finite-dimensional approximation of the space $\mathcal{V}$, we can find the values $\frac{\delta \mathcal{F}}{\delta v}^* \mathcal{F}(u,v)$ by choosing h to be the values of a basis $\mathcal{B}$ of $\mathbb{R}^{N_{\mathcal{V}}}$. In order to further reduce the complexity, we can perform a singular value decomposition of $\frac{\delta \mathcal{F}}{\delta v}^* \mathcal{F}(u,v) : \mathcal{U} \times \mathcal{V} \to \mathcal{V}$ on a given dataset $\{(u_i, v_i)\}_{i=1}^m \subset \mathcal{U} \times \mathcal{V}$: Denoting with $F \in \mathbb{R}^{m \times N_{\mathcal{V}}}$ the matrix collocating the point evaluations $\frac{\delta \mathcal{F}}{\delta v}^* \mathcal{F}(u_i, v_i)$, we obtain $F = U\Sigma V^T$ with $U \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{m \times N_{\mathcal{V}}}$, and $V \in \mathbb{R}^{N_{\mathcal{V}} \times N_{\mathcal{V}}}$. Given a low-rank approximation $r << N_{\mathcal{V}}$, we have $F \approx U_r \Sigma_r V_r^T$. Then, an approximation of $\left(\frac{\delta \mathcal{F}}{\delta v}(u,v)\right)^* \mathcal{F}(u,v)$ can be achieved by expanding it in terms of the reduced basis $\mathcal{B}_r = \{v_1, \ldots, v_r\}$ where the coefficients of this can be found by testing with $h = v_i, i = 1, \ldots, r$ in (B.4).

B.7 Connection to Attention Mechanism

The *operator-valued operator* $\mathcal{N}(v_n) = \left(\left(\frac{\delta \mathcal{F}}{\delta v}(u, v_n)\right)^* \frac{\delta \mathcal{F}}{\delta v}(u, v_n) + \lambda I\right)^{-1}$ depends nonlinearly on its input (the current iterate v_n), and its output is a nonlocal, linear operator. This observation leads to the connection with the attention mechanism proposed in [338], which defines itself an operator of this nature. The attention mechanism is at the heart of transformers, a neural network architecture widely used for language [333] and vision [89] tasks and recently employed in the context of operator learning, for example as in [51, 52, 141, 183, 270]. Indeed, self-attention and cross-attention are extended to the function space setting in [51]. Following this framework, we write the extension to function space of the attention mechanism from [338] in its most general form. We define the attention operator $\mathcal{A} : \mathcal{V} \times \mathcal{W} \times \mathcal{U} \to \mathcal{U}$ by

$$\big(\mathcal{A}(v, w, u)\big)(x) = \mathbb{E}_{y \sim p(\cdot; v, w, x)} V u(y), \tag{B.5}$$

where

$$p(y; v, w, x) = \frac{\exp\Big(\big\langle Q v(x), K w(y) \big\rangle\Big)}{\int_\Omega \exp\Big(\big\langle Q v(x), K w(s) \big\rangle\Big)\, \mathrm{d}s},$$

where the learnable $Q \in \mathbb{R}^{d \times d_v}, K \in \mathbb{R}^{d \times d_w}, V \in \mathbb{R}^{d \times d_u}$ parametrize the attention operator $\mathcal{A}$. Given this definition, we may approximate

$$\mathcal{N}(v_n) \approx \mathcal{A}(v_n, v_n, \cdot) \in \mathcal{L}(\mathcal{U}, \mathcal{V}). \tag{B.6}$$

The above is unlike the standard way attention is implemented in practice, as the second and third input functions of $\mathcal{A}$ differ. An alternative approximation of $\mathcal{N}$ based on the cross-attention operator is given by $\mathcal{A}(v_n, \mathcal{F}(u, v_n), \cdot)$; note that, here, the second input is the residual itself. Using an approximation of $\mathcal{N}$ given by (B.6) leads to the reformulation of the iteration (3.7) as a neural network block

$$\begin{cases} r_{n+1} &= \mathcal{F}(u, v_n), \\ \delta v_{n+1} &= \mathcal{A}(v_n, v_n, r_{n+1}) \approx \mathcal{N}(v_n)(r_{n+1}), \\ v_{n+1} &= v_n + \delta v_{n+1}. \end{cases} \tag{B.7}$$

We highlight that as the transformer from [338] consists of the application of the attention mechanism (a nonlocal operator) along with residual connections and pointwise linear transformations, the block in (B.7) is defined itself by a pointwise transformation, attention and a residual connection. This perspective may be used as the basis for an end-to-end learning approach. In particular, we may unroll CHONKNORIS, as outlined in [220]. This entails choosing a fixed number of iterations

for our iterative method and interpreting this iterative method as a composition of blocks of the form (B.7). We leave further investigation of this end-to-end approach to future work.

Remark B.7.1. *Note that while common operator learning models fail to achieve high accuracy with increasing depth, the proposed iterative structure* (3.7) *is specifically designed to converge toward the true operator $\mathcal{G}$ as depth increases as the compositions emulate a quasi-Newton algorithm. This connection aligns with observations in the ANN literature [220, 240], where successful algorithms are effective not solely due to their depth or complexity, but because they possess: (1) Sufficient expressivity to emulate convergent numerical approximation methods; and (2) Enough structure in their computational graphs [239] to leverage the same principles that make such numerical methods effective. Without (2) increasing depth in an ANN algorithm may only increase expressivity and complexity without improving accuracy.*

B.8 Proofs

B.8.1 Theoretical results

Proof of Theorem 3.4.2. Write $A(v) := \mathcal{F}'(v)$, $A_k := A(v_k)$, and $F_k := \mathcal{F}(v_k)$. For $\lambda > 0$, $R_\lambda(v) := (\lambda I_{\mathcal{V}} + A(v)^* A(v))^{-1}$.

Step 1. On $\mathcal{W}$,

$$I_{\mathcal{W}} - A(v)\, R_\lambda(v)\, A(v)^* \;=\; \lambda\big(\lambda I_{\mathcal{W}} + A(v)A(v)^*\big)^{-1}. \tag{B.8}$$

Indeed, $(\lambda I_{\mathcal{V}} + A^*A)R_\lambda = I_{\mathcal{V}}$; multiplying on the left by A and on the right by A^* gives $A(\lambda I + A^*A)R_\lambda A^* = AA^*$, i.e., $\lambda A R_\lambda A^* + AA^* R_\lambda A^* = AA^*$, and rearranging yields (B.8) since $(\lambda I_{\mathcal{W}} + AA^*)$ is invertible by (A3).

Step 2. With $\delta v_k = -\widehat{R}_{\lambda_k}(v_k) A_k^* F_k$,

$$\begin{aligned} A_k \delta v_k + F_k &= \big(I - A_k \widehat{R}_{\lambda_k}(v_k) A_k^*\big) F_k \\ &= \underbrace{\big(I - A_k R_{\lambda_k}(v_k) A_k^*\big)}_{\text{(I)}} F_k \\ &\quad + \underbrace{A_k \big(R_{\lambda_k}(v_k) - \widehat{R}_{\lambda_k}(v_k)\big) A_k^*}_{\text{(II)}} F_k. \end{aligned}$$

By (B.8) at v_k and (A3),

$$\|(I)\| = \big\|\lambda_k (\lambda_k I_{\mathcal{W}} + A_k A_k^*)^{-1}\big\| \le \frac{\lambda_k}{\lambda_k + \sigma_*^2} \;=:\; \theta_k^{\text{tik}}.$$

For (II), using submultiplicativity and Assumption 3.4.1(A3),

$$\|(II)\| \;\le\; \|A_k\|\,\|R_{\lambda_k}(v_k) - \widehat{R}_{\lambda_k}(v_k)\|\,\|A_k^*\| \;\le\; M^2\,\varepsilon_{\lambda_k} \;=:\; \theta_k^{\mathrm{des}}.$$

Therefore

$$\|A_k\delta v_k + F_k\| \;\le\; (\theta_k^{\mathrm{tik}} + \theta_k^{\mathrm{des}})\,\|F_k\| \;=:\; \theta_k\,\|F_k\|,$$

which is the inexact Newton forcing inequality (3.25).

Step 3. By (A2), for any v and s with $v, v+s \in \overline{B(v_0, R)}$,

$$\|\mathcal{F}(v+s) - \mathcal{F}(v) - \mathcal{F}'(v)s\| \;\le\; \tfrac{1}{2}L\|s\|^2.$$

The standard (Dembo–Eisenstat–Steihaug) majorant analysis applies with

$$\bar{\theta} := \sup_k \theta_k \;<\; 1, \qquad \widetilde{L} := \frac{L}{1-\bar{\theta}},$$

yielding existence/uniqueness in the ball, the majorant bounds, and convergence provided $\widetilde{h} = \beta\widetilde{L}\eta \le \frac{1}{2}$ and $t_* \le R$. If additionally $\lambda_k \to 0$ and $\varepsilon_{\lambda_k} \to 0$, then $\theta_k \to 0$ and the local rate approaches quadratic; if $\theta_k = O(\|F_k\|)$, the rate is quadratic. □

Proof of Corollary 3.4.3. We sketch the standard derivation; see also DES [81]. By the mean-value theorem for Gâteaux differentiable functions and (A2), we obtain

$$\mathcal{F}(v_k + \delta v_k) - \mathcal{F}(v_k) - A(v_k)\delta v_k \;=\; \int_0^1 \big(A(v_k + \tau\delta v_k) - A(v_k)\big)\,\delta v_k\,d\tau,$$

so $\|\mathcal{F}(v_{k+1})\| \le \|A(v_k)\delta v_k + \mathcal{F}(v_k)\| + \frac{L}{2}\|\delta v_k\|^2$. Left-multiplying by $A(v_k)^{-1}$ and using $\|A(v_k)^{-1}\| \le \beta$ on $\overline{B}(v_0, R)$ yields

$$\|v_{k+1} - v_k - A(v_k)^{-1}\mathcal{F}(v_k)\| \;\le\; \beta\,\|A(v_k)\delta v_k + \mathcal{F}(v_k)\| \;+\; \frac{\beta L}{2}\,\|\delta v_k\|^2.$$

By the forcing condition with $\bar{\theta}$ and the stability of the step $\|\delta v_k\| \le \|A(v_k)^{-1}\|\,\|\mathcal{F}(v_k)\| + o(\|\mathcal{F}(v_k)\|) \le \beta\|\mathcal{F}(v_k)\| + o(\|\mathcal{F}(v_k)\|)$, one arrives at

$$\|v_{k+1} - v^*\| \;\le\; \frac{\bar{\theta}}{1-\bar{\theta}}\,\|v_k - v^*\| \;+\; \frac{\beta L}{2(1-\bar{\theta})}\,\|v_k - v^*\|^2$$

which is (3.26). Assertions (i)–(iv) follow immediately: (i) by dropping the quadratic term and applying limes superior; (ii) because $\theta_k \to 0$ eliminates the linear term; (iii) and (iv) use the local equivalence $\|\mathcal{F}(v_k)\| \asymp \|v_k - v_*\|$ (since $A(v_*)$ is continuous and invertible for large k) to replace $\|\mathcal{F}(v_k)\|^\alpha$ by e_k^α. □

Proof of Remark 3.4.4. (a) is immediate from $\theta_{\text{tik,k}} \to 0$ and $\theta_{\text{des,k}} \to 0$. For (b), $\theta_{\text{tik,k}} \le \lambda_k/\sigma_*^2 = O(\|\mathcal{F}(v_k)\|)$ and $\theta_{\text{des,k}} = M^2\,\varepsilon_{\lambda_k} = O(\|\mathcal{F}(v_k)\|)$, hence $\theta_k = O(\|\mathcal{F}(v_k)\|)$; apply Corollary 3.4.3 (iv). For (c), differentiate ϕ on $(0,\infty)$:

$$\phi'(\lambda) = \frac{\sigma_*^2}{(\lambda+\sigma_*^2)^2} - 2\,C\,\lambda^{-3}.$$

Setting $\phi'(\lambda) = 0$ yields $(\lambda+\sigma_*^2)^{-2} \sim 2C\,\sigma_*^{-2}\lambda^{-3}$; for the coarse scaling one may drop the $+\sigma_*^2$ inside parentheses (or solve exactly), giving $\lambda^3 \sim 3C\,\sigma_*^2$ and the stated λ_*. This balances the two terms in $\bar\theta$; an annealing $\lambda_k \downarrow 0$ then ensures $\theta_k \to 0$ and, by Corollary 3.4.3, superlinear/quadratic rates. □

Proof of Theorem 3.4.7. We verify the assumptions of Theorem 3.4.2 for the PDE setting and identify the constants appearing there.

Step 1: Setting and choice of base point. For a fixed $f \in \mathcal{W}$, we suppress the dependence on f and write $\mathcal{F}(u) = \mathcal{F}(u,f) = -\Delta u + \kappa u^3 - f : \mathcal{V} \to \mathcal{W}$. For simplicity, we choose the base point $v_0 = 0$ and $\kappa = 1$.

By definition of the residual map,

$$\mathcal{F}(0) = -f \in \mathcal{W}, \qquad \mathcal{F}'(0) = A := -\Delta : \mathcal{V} \to \mathcal{W}.$$

Recall that the H^{-1}-norm is defined by duality with respect to the $\mathcal{V}$-inner product induced by A, so that for all $g \in \mathcal{W} = H^{-1}$,

$$\|g\|_{\mathcal{W}} \;=\; \|A^{-1}g\|_{\mathcal{V}}.$$

Therefore

$$\eta \;:=\; \|\mathcal{F}'(0)^{-1}\mathcal{F}(0)\|_{\mathcal{V}} \;=\; \|A^{-1}(-f)\|_{\mathcal{V}} \;=\; \|f\|_{\mathcal{W}},$$

which is the first identity in the statement.

Step 2: Verification of (A1) and explicit β. Assumption (A1) of Assumption 3.4.1 requires that $\mathcal{F}'(v_0)$ is invertible and $\|\mathcal{F}'(v_0)^{-1}\| \le \beta$. As just noted, $\mathcal{F}'(0) = A$ is an isomorphism $\mathcal{V} \to \mathcal{V}^* = \mathcal{W}$, so (A1) holds with

$$\beta = \|\mathcal{F}'(0)^{-1}\| = \|A^{-1}\| = 1,$$

where the last equality follows from the way the H^{-1}-norm is induced by the $\mathcal{V}$-inner product via A.

Step 3: Lipschitz bound (A2) and choice of L. Let $B_r := \{v \in \mathcal{V} : \|v\|_{\mathcal{V}} \le r\}$. Then, for each $v \in X$, we have

$$\mathcal{F}'(v)h \;=\; Ah + 3v^2 h.$$

Hence, for $u, v \in B_r$,

$$\big(\mathcal{F}'(u) - \mathcal{F}'(v)\big)h = 3\,(u^2 - v^2)h = 3\,(u+v)(u-v)h.$$

Using the one-dimensional Sobolev embedding and the Poincaré inequality on $\mathbb{T}$, we obtain

$$\|w\|_{L^\infty} \;\le\; \frac{1}{\pi}\,\|w\|_{\mathcal{V}}, \qquad \|w\|_{L^2} \;\le\; \frac{1}{\pi}\,\|w\|_{\mathcal{V}}.$$

Together with the Cauchy-Schwarz inequality, we obtain the bound

$$\|\,(u^2 - v^2)h\,\|_{H^{-1}} \;\le\; \frac{1}{2\pi^2}\,\big(\|u\|_{\mathcal{V}} + \|v\|_{\mathcal{V}}\big)\,\|u - v\|_{\mathcal{V}}\,\|h\|_{\mathcal{V}}.$$

Since $u, v \in B_r$, this yields

$$\|\mathcal{F}'(u) - \mathcal{F}'(v)\| \;\le\; \frac{3}{2\pi^2}\,r\,\|u - v\|_{\mathcal{V}},$$

so (A2) holds with

$$L := \frac{3}{2\pi^2}\,r.$$

Step 4: Uniform bound (A3) and choice of M and σ_*. For any $v \in B_r$ and $h \in \mathcal{V}$ with $\|h\|_{\mathcal{V}} = 1$,

$$\|\mathcal{F}'(v)h\|_{H^{-1}} \le \|Ah\|_{H^{-1}} + \|3v^2 h\|_{H^{-1}}.$$

By definition of the norms, $\|Ah\|_{H^{-1}} = \|h\|_{\mathcal{V}} = 1$. Arguing as in Step 3, we obtain

$$\|3v^2 h\|_{H^{-1}} \;\le\; \frac{3}{4\pi^2}\,\|v\|_{\mathcal{V}}^2\,\|h\|_{\mathcal{V}} \;\le\; \frac{3}{4\pi^2}\,r^2.$$

Hence,

$$\|\mathcal{F}'(v)\| = \sup_{\|h\|_{\mathcal{V}}=1} \|\mathcal{F}'(v)h\|_{H^{-1}} \;\le\; 1 + \frac{3}{4\pi^2}\,r^2 =: M$$

for all $v \in B_r$, which is the M stated in Theorem 3.4.7.

Moreover, by coercivity of A and the positivity of the nonlinearity, we obtain

$$\langle \mathcal{F}'(v)h, h\rangle_{\mathcal{W},\mathcal{V}} = \langle Ah, h\rangle_{\mathcal{W},\mathcal{V}} + \langle 3v^2 h, h\rangle_{\mathcal{W},\mathcal{V}} \;\ge\; \|h\|_{\mathcal{V}}^2,$$

so that the smallest singular value of $\mathcal{F}'(v)$ is bounded below by 1. Therefore

$$\sigma_* := \inf_{v \in B} \sigma_{\min}(\mathcal{F}'(v)) \;\ge\; 1.$$

Thus (A3) holds with the above M and $\sigma_* \geq 1$.

Step 5: Forcing term and choice of $\bar{\theta}$. Let $\lambda > 0$ be fixed and assume that we use a constant Tikhonov parameter, i.e. $\lambda_k \equiv \lambda$. For the learned resolvent $\widehat{R}_\lambda$, we define the design error

$$\varepsilon_\lambda := \sup_{v \in B_r} \left\| \widehat{R}_\lambda(v) - R_\lambda(v) \right\|_{\mathcal{L}(\mathcal{V},\mathcal{V})}.$$

Then, Theorem 3.4.2 (Tikhonov–inexact NK) gives, for any iterate $v_k \in B_r$,

$$\frac{\|\mathcal{F}'(v_k)\delta v_k + \mathcal{F}(v_k)\|_{\mathcal{W}}}{\|\mathcal{F}(v_k)\|_{\mathcal{W}}} \;\leq\; \frac{\lambda}{\lambda + \sigma_*^2} + M^2\,\varepsilon_\lambda =: \theta_k.$$

Using $\sigma_* \geq 1$, we further obtain

$$\theta_k \;\leq\; \frac{\lambda}{1+\lambda} + M^2\,\varepsilon_\lambda =: \bar{\theta}.$$

If $\bar{\theta} < 1$, then the hypothesis $\sup_k \theta_k \leq \bar{\theta} < 1$ of Theorem 3.4.2 is satisfied.

Step 6: Application of Theorem 3.4.2 and explicit $\widetilde{L}$. Theorem 3.4.2 then yields that all the inexact Newton–Kantorovich conclusions hold with modified Lipschitz constant

$$\widetilde{L} \;=\; \frac{L}{1-\bar{\theta}}.$$

Plugging in the PDE-specific values

$$L = \frac{3}{2\pi^2}\,r, \qquad \bar{\theta} = \frac{\lambda}{1+\lambda} + M^2\,\varepsilon_\lambda, \qquad M = 1 + \frac{3}{4\pi^2}\,r^2,$$

we obtain the expression stated in Theorem 3.4.7,

$$\widetilde{L} \;=\; \frac{L}{1-\bar{\theta}} \;=\; \frac{\frac{3}{2\pi^2}r}{1 - \frac{\lambda}{1+\lambda} - M^2\varepsilon_\lambda}.$$

Together with the identification $\eta = \|\mathcal{F}'(0)^{-1}\mathcal{F}(0)\|_{\mathcal{V}} = \|f\|_{H^{-1}}$ from Step 1, this shows that all conclusions of Theorem 3.4.2 hold in the PDE setting with the explicit constants given in Theorem 3.4.7. □

Appendix C

APPENDIX TO CHAPTER 4: NETWORK PRUNING VIA MARCHENKO–PASTUR THEORY

C.1 Other Numerical Results

This appendix provides some numerical results which shows that MP-based pruning leads to a reduction in loss and an increase in accuracy, which is relevant to Theorem 4.4.2. We also confirm that adding (which in this context is the same as removing) a random matrix to the weight layer matrices of the DNN does not lead to a change in accuracy or cross-entropy loss but does lead to a change in the L_2 loss, numerically confirming Theorem 4.4.2 and Lemma 4.4.4.

Example C.1.1. *In this example, we trained on Fashion MNIST a fully connected DNN with ReLU activation and the following topology:*

$$[784, 3000, 3000, 3000, 3000, 500, 10],$$

This means our network has 6 layers followed by activation functions, with layer k being a linear layer $L_k : \mathbb{R}^{T_k} \to \mathbb{R}^{T_{k+1}}$. The training was performed multiple times, with no regularization and no PM-based pruning, with L_1 regularization, L_2 regularization, $L_1 + L_2$ regularization and MP pruning (see Algorithm 4) and so on to compare these different forms of training. We also compare our approach with stable rank regularization, that is adding the sum over l of:

$$stable(W_l) = \frac{\|W_l\|_F^2}{\|W_l\|_2^2} \tag{C.1}$$

to the loss function, see [355] for more. This regularization is supposed to help induce heavy tail in the weight layer matrices, making the weight layer matrices "less random".

In all cases, the training was done for 100 *epochs. MP-pruning based training essentially involved pruning the small singular values in the weight layer matrices based on the MP distribution, and using this reduction in rank to reduce the number of parameters in the DNN, see Appendix 4 and [34, 307] for more information. The ReLU activation function was applied after every layer, including the final layer (which is not typical). While it might be easier to train DNNs without applying the*

activation function to the final layer, we found that we obtained the highest difference in accuracies (between normal training and training with MP-based pruning) when training with the affirmation structure while using MP-based pruning (see Table C.1). Furthermore, without the ReLU activation function being applied to the final layer (softmax is still applied), the accuracy was less than 90.5% *(without MP-pruning). For the same architecture, using* $L1 + L2$ *regularization and MP-pruning, we obtained a* 91.37% *accuracy on the Fashion MNIST test set, which is the highest accuracy we observed on the data set using a fully connected DNN (see Example C.1.2). MP-based pruning also increases the accuracy of fully connected DNNs used for regularization tasks; for example, see Subsection C.2.*

Every 4 *epochs, we kept a portion of the smallest singular values (singular values less than* $\sqrt{\lambda_+}$ *given in Assumption* 2 *in Subsection 4.4.1) based on the formula:*

$$f(epoch) = \max\left(0, -\frac{1}{200} \cdot epoch + 1\right). \tag{C.2}$$

By doing this, we are essentially removing "some" of the random matrices R_l *during training. Once the training is finished, we sparsify the weight layer matrices of the DNN by removing the weights smaller than some threshold* ξ*, see Algorithm 6. The results are given in Fig. C.1 and Table C.1. In [34], the authors showed that one can reduce the number of parameters by a much larger amount (a reduction in parameters by up to* 99.98%*) if we remove*

$$f(epoch) = \max\left(0, -\frac{1}{100} \cdot epoch + 1\right) \tag{C.3}$$

of the smallest singular values and sparsify the matrix once the training is complete. However, the accuracy that is obtained is also lower (see [34] for more on the relationship between MP-based pruning and sparsification). In Fig. C.1, we show how accuracy drops as we sparsify the DNNs.

The hyperparameters for the training are given in Table C.3.

Example C.1.2. *In this example, we trained on Fashion MNIST a fully connected DNN with ReLU activation (on all but the final layer) and topology:*

$$[784, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 10].$$

This means our network has 7 layers followed by activation functions (excluding the final layer), with layer k *being a linear layer* $L_k : \mathbb{R}^{T_k} \to \mathbb{R}^{T_{k+1}}$*. The training was*

Training method	Training acc.	Testing acc.
~~MP-based pruning, L_1, L_2~~	78.03%	71.64%
MP-based pruning, ~~L_1~~, ~~L_2~~	88.12%	81.22%
MP-based pruning, L_1, L_2	**99.82%**	**90.53%**
~~MP-based pruning~~, L_1, L_2	87.57%	81.70%
~~MP-based pruning~~, L_1, ~~L_2~~	87.42%	81.46%
~~MP-based pruning~~, ~~L_1~~, L_2	87.37%	81.31%
MP-based pruning, L_1, L_2, stable rank	**99.97%**	90.16%
~~MP-based pruning~~, L_1, L_2, stable rank	79.01%	71.38%
MP-based pruning, L_2	**99.98%**	90.16%

Table C.1: Performance of fully connected DNN at final epoch for various training strategies.

Training method	Final loss on training set
~~MP-based pruning, L_1, L_2~~	0.629775
MP-based pruning, ~~L_1~~, ~~L_2~~	0.170905
MP-based pruning, L_1, L_2	**0.003104**
~~MP-based pruning~~, L_1, L_2	0.325156
~~MP-based pruning~~, L_1, ~~L_2~~	0.399908
~~MP-based pruning~~, ~~L_1~~, L_2	.431001
MP-based pruning, L_1, L_2, stable rank	**.001134**
~~MP-based pruning~~, L_1, L_2, stable rank	.476791
MP-based pruning, L_2	**0.001588**

Table C.2: Performance of fully connected DNN at final epoch for various training strategies.

performed with L_1 + L_2 regularization and PM-based pruning (see Algorithm 4) and compared with using regularization alone.

In all cases, the training was done for 1000 *epochs. The ReLU activation function was applied after every layer, excluding the final layer (which is typical). Without MP-based pruning, the accuracy on the test set was lower than* 90.5%. *For the MP-based pruning, every* 12 *epochs we kept a portion of the smallest singular values (singular values less than $\sqrt{\lambda_+}$ given in Assumption* 2 *in Subsection 4.4.1) based on the formula:*

$$f(epoch) = \max\left(0, -\frac{1}{1000} \cdot epoch + 1\right). \tag{C.4}$$

We used SGD together with a momentum of .97 and a learning rate of .001 *with the CosineAnnealingLR scheduler. The L1 and L2 regularization hyperparameters were* 0.0000005. *Using MP-based pruning, the test set accuracy was* 91.37.

hyperparameters	**Value**
Stable rank (SR)	0.000001
Every how many epochs the SR was used in the loss	6
L_1	0.000001
L_2	0.000001
Every how many epochs the MP pruning was done	4
Momentum	0.9
Learning rate	0.01 (decay by .96 every 4 epochs)

Table C.3: hyperparameter values

Example C.1.3. *In this example, we trained on Fashion MNIST a fully connected DNN with relu activation and topology:*

$$T = [784, 3000, 3000, 3000, 3000, 500, 10],$$

This means our network has 6 layers followed by activation functions, with layer k being a linear layer $L_k : \mathbb{R}^{T_k} \to \mathbb{R}^{T_{k+1}}$. *We use the MP-pruning approach that can be found in Example C.1.1, to achieve a* 90.5% *accuracy on the test set. Note that in Lemma C.2.5, the method for training the DNN does not matter; only the loss and accuracy of the DNN at time t is what matters. Furthermore, this lemma is equally valid if instead of replacing* W_2 *with* S_2*, we replace the weight layer matrix* W_2 *with a new matrix* $W_2 + R_2'$*, with* R_2' *a random matrix with components i.i.d. taken from* $N(1, \epsilon)$ *with* $\epsilon \sim \frac{1}{N}$. *Thus, to all but the final layer matrix of the DNN we add a random matrix* R_b' *with components i.i.d. from the distribution* $N(0, \epsilon)$ *and, in Fig. C.2, we plot the loss (both with and without L2 regularization) and accuracy of the DNN as we vary* ϵ. *We see that for* $\epsilon \sim \frac{1}{N}$ *we have that the cross-entropy loss and accuracy don't change much, which numerically confirms Lemma C.2.5.*

C.2 A simple illustration of the relationship between RMT-based pruning and regularization

The core concept underlying the developed pruning method relies on the observation that Neural Network weight matrices frequently exhibit characteristics akin to those of a low-rank matrix addition to a random matrix component. The idea of the pruning method developed relies on the observation that oftentimes, the weight matrices of Neural Networks behave like a very low-rank matrix with an added random matrix.

Algorithm 2 Computing Parameter λ_+ using MP and Tracy-Widom Distribution

procedure
  Pick parameters $\alpha \in (0, 1/2), \beta \in (0, 1)$.
  for each $\alpha N \leq k \leq (1-\alpha)N$ **do**
    Calculate q_k, the (k/N) upper-quantile of the MP distribution with $\sigma^2 = 1$ and $c = 1$.
    q_k satisfies $\int_0^{q_k} \frac{1}{2\pi} \frac{\sqrt{(4-\lambda)\lambda}}{\lambda} = k/N$.
  end for
  Evaluate $\hat{\sigma}^2 = \frac{\sum_{\alpha N \leq k \leq (1-\alpha)N} q_k \lambda_k}{\sum_{\alpha N \leq k \leq (1-\alpha)N} q_k^2}$.
  Derive $t_{1-\beta}$, the $(1-\beta)$ quantile of the Tracy-Widom distribution.
  Conclude with $\lambda_+ = \hat{\sigma}^2[4 + 2^{4/3} t_{1-\beta} \cdot N^{-2/3}]$.
end procedure

Algorithm 3 Spectral Analysis using BEMA Technique

procedure
  Take $X = \frac{1}{N} W^T W$ as input where W is $N \times M$.
  Derive the spectrum of $X = \{\sigma_1, \ldots, \sigma_M\}$.
  Ascertain the observed cumulative spectral distribution of X, denoted F_X.
  Apply the BEMA technique with parameters α and β to discern $\hat{\sigma}^2$, the expected coordinate variance of W.
  Determine $0 \leq i_{\min} < i_{\max} \leq M$ with:
    - $i_{\min}$ as the least integer for which $\frac{i_{\min}}{M} \geq \alpha$.
    - $i_{\max}$ as the greatest integer for which $\frac{i_{\max}}{M} \leq 1 - \alpha$.
  Describe F'_X as the theoretical CDF for the MP distribution using parameters $\hat{\sigma}^2$ and $\lambda = N/M$.
  Compute $s = \max_{i \in [i_{\min}, i_{\max}]} \left| F_X(i) - F'_X(i) \right|$.
  If $s > \gamma$, rule out the idea that X is governed by the stated distribution. Else, if $s \leq \gamma$, retain this notion.
end procedure

Algorithm 4 MP-based pruning algorithm

Require: ℓ, a predetermined number of epochs; τ, a threshold for the MP fit criteria in Subsection C.4.3; $f(\text{epoch})$, a monotonically decreasing function from 1 to 0 (i.e. (C.2)) and for each $1 \leq l \leq L$ and weight layer matrix W_l **state** $split_l = false$.

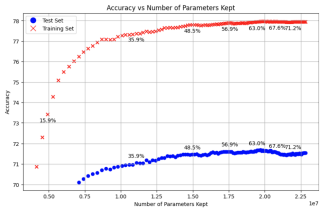
Initialize: Train the DNN for ℓ epochs. Take epoch := ℓ.
while a predefined training condition is met (i.e. epoch $\leq$ 100) **do**
    for each l, if $split_l = false$ then for weight matrix W_l in the DNN ϕ **do**
        Perform SVD on W_l to obtain $W_l = U_l \Sigma_l V_l^T$.
        Calculate eigenvalues of $W_l^T W_l$.
        Apply BEMA algorithm (see Subsection C.4.1) to find the best fit MP distribution for ESD of $X = W_l^T W_l$ and corresponding λ_+.
        Check if ESD of X fits the MP distribution using MP fit criteria from Subsection C.4.3 and threshold τ.
        if ESD fits the MP distribution **then**
            Eliminate the portion $(1 - f(\text{epoch}))$ of singular values smaller than $\sqrt{\lambda_+}$ to obtain Σ' and form $W_l' = U_l \Sigma_l' V_l^T$.
            Use Σ' to create $W_{1,l}' = U_l \sqrt{\Sigma_l'}$ and $W_{2,l}' = \sqrt{\Sigma_l'} V_l^T$.
            if $W_{1,l}'$ and $W_{2,l}'$ together have fewer parameters than W_l' **then**
                Replace W_l in the DNN ϕ with $W_{1,l}' W_{2,l}'$, change $split_l = true$.
            else
                Replace W_l in the DNN ϕ with W_l'.
            end if
        else
            Don't replace W_l.
        end if
    end for
    Train the DNN for ℓ epochs. Take epoch := epoch + ℓ.
    for each l, if $split_l = true$ **do**
        if for $W_l := W_{1,l}' W_{2,l}'$ the ESD of X_l fits the MP distribution with thresholds τ and λ_+ **and** if, we (hypothetically) applied steps 4-12 to W_l, the number of parameters in the DNN ϕ would decrease **then**
            replace $W_{1,l}' W_{2,l}'$ with W_l and $split_l = false$.
        else
            Don't change anything.
        end if
    end for
end while

Algorithm 5 Sparsify Singular Vectors

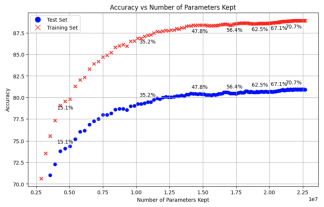
Let W_l denote the l-th layer matrix of a DNN of size $N \times N$ and an initial threshold θ.
Perform SVD on W_l: $W_l = U\Sigma V^T$.
Given threshold $\sqrt{\lambda_+}$.
for $i = 1, \ldots, N$ **do**
    if $\Sigma_{ii} < \sqrt{\lambda_+}$ **then**
        $M_i = 1$
    else
        $M_i = 0$
    end if
    $T_i(\sigma) = \theta \cdot \left(1 - \frac{\sigma}{\sqrt{\lambda_+}}\right)^{30}$
    if $M_i = 1$ **then**
        $U_{:,i} = \text{apply\_sparsity}(U_{:,i}, T_i(\Sigma_{ii}))$
        $V_{i,:} = \text{apply\_sparsity}(V_{i,:}, T_i(\Sigma_{ii}))$
    end if
end for
Extra universal sparsification step:
Given a threshold, θ.
for $i = 1, \ldots, N$ **do**
    $U_{:,i} = \text{apply\_sparsity}(U_{:,i}, \frac{\theta}{750})$
    $V_{i,:} = \text{apply\_sparsity}(V_{i,:}, \frac{\theta}{750})$
end for
Recompose matrix: $W_l' = U\Sigma V^T$.

Algorithm 6 Element-wise Sparsification

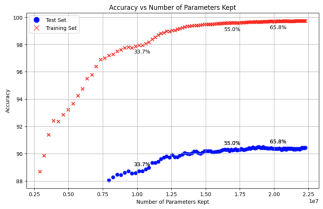
Input: Matrix W_l, threshold ξ
Output: Sparsified matrix W_l'
for each element w_{ij} in W_l **do**
    if $w_{ij} < \xi$ **then**
        $w_{ij}' = 0$
    else
        $w_{ij}' = w_{ij}$
    end if
end for
return W_l'



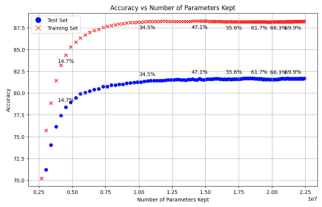
(a) No MP-based pruning, and no L_1 or L_2 regularization.



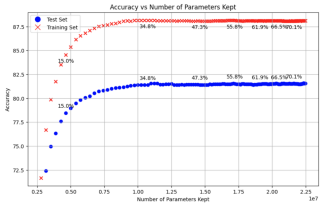
(b) MP–based pruning, no L_1 or L_2 regularization.



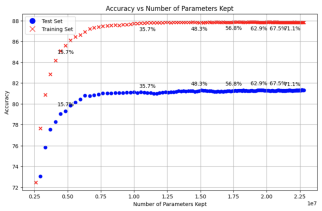
(c) MP-based training + L_1 and L_2 regularization.



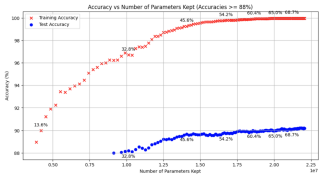
(d) No MP-based pruning, used L_1+L_2 regularization.



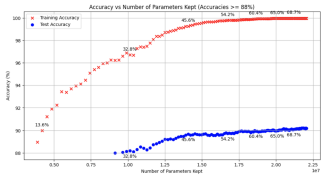
(e) Only used L_1 regularization.



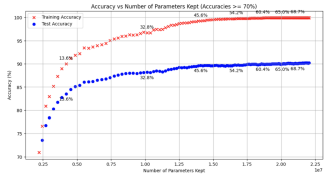
(f) Only used L_2 regularization.



(g) MP-based pruning, L_1, L_2, stable rank.



(h) L_1, L_2 and stable rank.

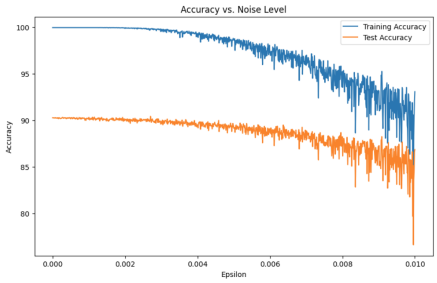


(i) MP-based pruning with L_2.

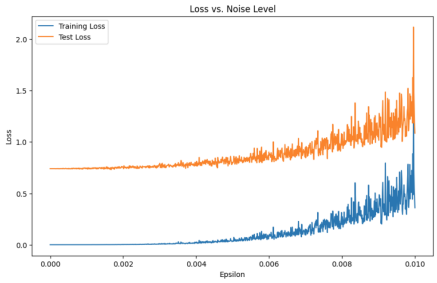
Figure C.1: DNN performance vs. sparsification

The origin of the randomness is two-fold. Firstly, Neural Networks undergo training through various iterations of stochastic gradient descent. Consequently, the resultant trained network serves as an approximate minimizer of the associated loss function, with its weight distribution potentially reflecting the inherent stochastic nature of the optimization algorithm employed. Secondly, the sample data used to train the network introduces randomness to the training process, particularly when the dataset incorporates random noise. The act of pruning can be conceptualized as a regularization technique operating across the spectrum aimed at promoting the recovery of this low-rank matrix structure.

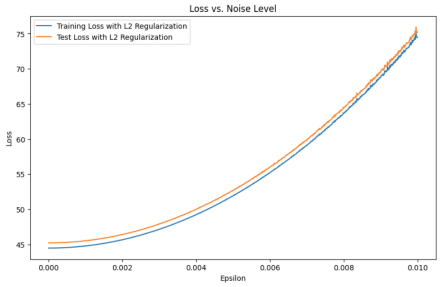
In order to better understand the implications of pruning, we present a simple problem designed to illuminate the effect outcomes of pruning in comparison to alternative forms of regularization.



(a) Accuracy of the DNN on the training and test sets, vs. ϵ.



(b) DNN loss (with no regularization) on training and test set vs. ϵ.



(c) DNN loss with L_2 regularization on training and test set vs. ϵ. The hyperparameter μ in the regularization is .01.

Figure C.2: Accuracy and loss as we add noise to the weight layers of the DNN.

C.2.1 Regularization in a simple regression setting

Having a low-rank weight matrix implies that the DNN has found a feature map representation of the task that is lower dimensional than the set dimension of the weight matrix. In order to emulate this situation, we introduce a simple regression model where an optimal representation is known. We sample M random function $y^{(k)} \in Span\{\cos(\pi f \cdot), \sin(\pi f \cdot),$ for $f \in \{0, 0.2, .., 2\}\}$, and take random, noisy samples $(X_j, Y_j) \in \mathbb{R} \times \mathbb{R}^M$ s.t. $(Y_j)_k = y^{(k)}(X_j) + 7\epsilon_j$, with $\epsilon_j \sim \mathcal{N}(0, 1)$. Our objective entails the recovery of the $y^{(k)}$ functions from this dataset. An illustrative example of such data along one coordinate is provided in Figure C.3. Because of the structure of this problem, an efficient feature map representation of the data is through the Fourier transform, as y belongs to a vector space of low frequency. It has been argued that Neural Networks can be interpreted as a kernel with a learned feature map based on the data. For the sake of simplicity in this context, we assume a fixed feature map, the Fourier feature map. Accordingly, the Neural Network materializes as a linear layer on top of the selected feature map, equivalent to the kernel regression defined by the said feature map.

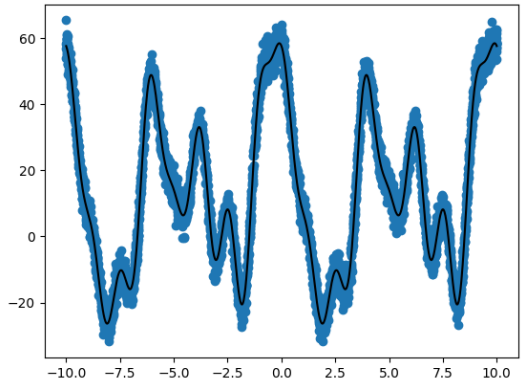


Figure C.3: Example of data generated from the problem statement on one axis. The blue line is the function y_0, and the blue dots are the data points.

In the following paragraphs, we will study the impact of different regularizations, including pruning, on the singular values of the linear layer's weight matrix. Given the significant variations in these singular values on a logarithmic scale, we visualize the cumulative distribution of singular values rather than conventional histograms.

Ideal case In the absence of noise, the optimal weight matrix has a low rank, as y belongs to a functional space with a finite number of distinct frequencies. Using kernel regression theory, we find the optimal weight matrix and visualize its singular values in figure C.4. Most singular values, barring 16, approach zero (numerically below 10^{-15}). This confirms that the optimal weight matrix is inherently low-rank in the absence of noise. Subsequently, our regression task pertains to recovering this low-dimensional structure when confronted with noisy data.

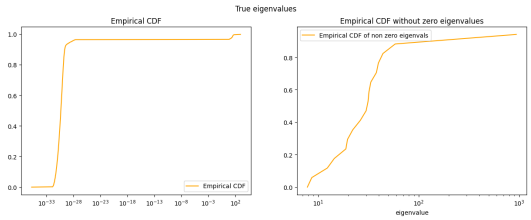


Figure C.4: Cumulative distribution of singular values of the optimal weight matrix

We will now try the pruning regularisation and compare it to two classical regularisations, L^2 and L^1 regularisation.

L^2 regularisation in the Presence of Noise Employing Kernel Ridge Regression, we obtain a L^2 regularised weight matrix. We plot the singular values of this matrix in figure C.5. We can see a large number of minimal singular values. By

running the BEMA algorithm on this matrix, we identify a spectrum resembling the Marchenko-Pastur distribution. Moreover, the algorithm finds an upper bound for random singular values consistent with the ideal spectrum we observed above. This alignment with the Marchenko-Pastur distribution implies that the predominant source of randomness in our weight matrices is data noise rather than the stochastic nature of the optimization algorithm. Indeed, the weight matrix obtained here is deterministic when conditioned on the data, as the loss admits a minimizer with a closed-form solution.

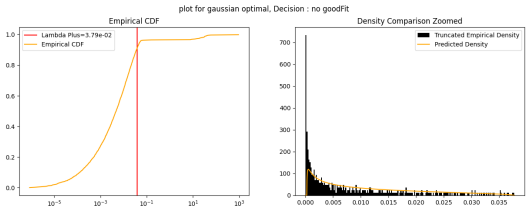


Figure C.5: Cumulative distribution of singular values of the L^2 regularised weight matrix

L^1 regularisation in the Presence of Noise Turning to L^1 regularization, using kernelized Lasso Regression, we plot the singular values of the resulting matrix in Figure C.6. Notably, 40% of the singular values have been zeroed out compared to the L^2 case. However, the remaining spectrum approximates an MP distribution, underscoring that the noise-induced randomness persists to some degree. It must be pointed out that L^1 regularisation tends to nullify coefficients of the weight matrix instead of singular values. In this scenario, our selected ideal feature map endows the optimal weight matrix with sparsity, thus rendering the recovery of a sparse weight matrix equivalent to the recovery of a low-rank one. It warrants emphasis, however, that with an alternate feature map, L^1 regularization might prove less effective in revealing the weight matrix's low-rank structure and could even significantly diminish performance if weighed too heavily.

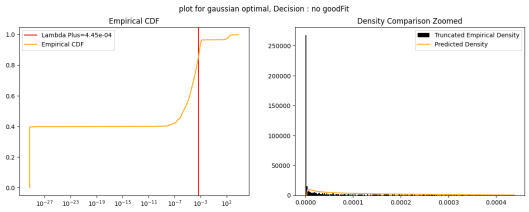


Figure C.6: Cumulative distribution of singular values of the L^1 regularised weight matrix

Pruning in the Presence of Noise We visualize the singular values of the unregularised matrix in Figure C.7. We see that this spectrum is very similar to the spectrum of the L^2 regularised weight matrix. This confirms the intuition that L^2 is not an efficient spectrum regularisation. Nonetheless, the BEMA algorithm successfully identifies a precise upper bound for the matrix's random singular values. This shows the capacity of pruning regularization to eliminate the noise-driven randomness from the weight matrix.

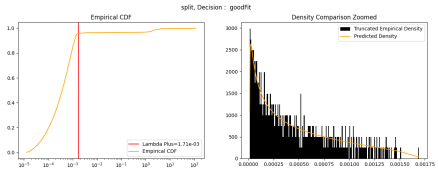


Figure C.7: Cumulative distribution of singular values of the weight matrix prior to pruning

In this simple illustrative instance, pruning regularization emerges as an effective tool for sparsifying the weight matrix's spectrum, thereby facilitating the recuperation of the underlying low-rank structure intrinsic to the problem at hand. Table C.4 reports the effects of the different regularizations on the mean squared error. We see that successfully removing the random part of the weight matrix spectrum allows better accuracy.

Model	Mean Squared Error
No Regularization	2.5735
L2 Regularization	2.5627
L1 Regularization	0.4157
Pruning	0.1491

Table C.4: Mean Squared Errors of Different Models for the simple regression task

C.2.2 Proof of Lemma 4.4.4

In this proof, $t \in \mathbb{R}$ does not refer to training time. We use this variable t because classically the Borell-TIS is given in terms of t.

We want to bound the probability of a single component of the output of the DNN when a random matrix R is included. Specifically, we aim to bound the probability that the i-th component of the following expression exceeds t:

Proof. **Step 1: Apply the Triangle Inequality**

By the triangle inequality, we can remove the activation functions (absolute value) and simplify the terms inside the norm:

$$Z = \mathbb{P}\left(|W_3 \circ \lambda \circ (R+S) \circ \lambda \circ W_1 s - W_3 \circ \lambda \circ S \circ \lambda \circ W_1 s|_i > t\right) \tag{C.5}$$

Step 2: Factor Out W_3

We can now factor out W_3, using the inequality $|(Av)_i| \leq \|A\|_1 \|v\|_\infty$, to move W_3 outside of the norm:

$$Z = \mathbb{P}\left(\|W_3\|_1 \cdot \|(R+S) \circ \lambda \circ W_1 s - S \circ \lambda \circ W_1 s)\|_\infty > t\right) \tag{C.6}$$

Step 3: Focus on R

We now separate the terms involving R and S, leaving us with:

$$Z \leq \mathbb{P}\left(\|W_3\|_1 \cdot \|(R \circ \lambda \circ W_1 s)\|_\infty > t\right) \tag{C.7}$$

Step 4: Focus on the Random Matrix R Let $Y = R \circ \lambda \circ W_1 s$. Each element of Y is a weighted sum of independent Gaussian random variables:

$$Y_i = \sum_j R_{ij} (\lambda \circ W_1 s)_j$$

The variance of each Y_i is:

$$\mathrm{Var}(Y_i) = \frac{1}{N} \sum_j (\lambda \circ W_1 s)_j^2 = \frac{\|W_1 s\|_2^2}{N}$$

Step 5: Apply Borell-TIS Inequality We can now apply the Borell-TIS inequality to Y. Note that each Y_i is a Gaussian random variable with mean 0 and variance $\frac{\|W_1 s\|_2^2}{N}$. Let $s_Y^2 = \frac{\|W_1 s\|_2^2}{N}$.

From the Borell-TIS inequality, for any $t > 0$:

$$\mathbb{P}\left(\|Y\|_\infty > \sqrt{\frac{2 \log N}{N}} \|W_1 s\|_2 + k\right) \leq 2 \exp\left(-\frac{N k^2}{2 \|W_1 s\|_2^2}\right)$$

Step 6: Incorporate Matrix W_3

Next, incorporate the effect of the matrix W_3. Using the properties of matrix norms, we have:

$$\|W_3 Y\|_i \leq \|W_3\|_1 \|Y\|_\infty$$

Therefore, the probability bound becomes:

$$Z \leq \mathbb{P}\left(\|W_3\|_1 \|Y\|_\infty > (\sqrt{\frac{2\log N}{N}} \|W_1 s\|_2 + k) \|W_3\|_1 \right) \tag{C.8}$$

$$\leq 2\exp\left(-\frac{Nk^2}{2\|W_1 s\|_2^2} \right), \tag{C.9}$$

with $t = (\sqrt{\frac{2\log N}{N}} \|W_1 s\|_2 + k)\|W_3\|_1$.

Step 7: Final Bound

Substituting $k = \frac{\|W_1 s\|_2}{N^{1.5/4}}$ results in:

$$Z \leq 2\exp\left(-\frac{N^{1-1.5/2}}{2} \right) = 2\exp\left(-\frac{N^{1/4}}{2} \right)$$

Conclusion:

$$Z = \mathbb{P}\Big(\big|\lambda \circ W_3 \circ \lambda \circ (R+S) \circ \lambda \circ W_1 s - \lambda \circ W_3 \circ \lambda \circ S \circ \lambda \circ W_1 s\big|_i > \tag{C.10}$$

$$\sqrt{\frac{2\log N}{N}} \|W_1 s\|_2 \|W_3\|_1 + \frac{\|W_3\|_1 \|W_1 s\|_2}{N^{1.5/4}} \Big) \leq 2\exp\left(-\frac{N^{1/4}}{2} \right) \tag{C.11}$$

□

For a DNN with more than three-layer matrices, the proof would be the same. For example, in the case of four-layer matrices, we have:

Proof. **Step 1: Express the Problem** We start with the probability we want to bound, now with four weight layers:

$$Z = \mathbb{P}\Big(|\lambda \circ W_4 \circ \lambda \circ W_3 \circ \lambda \circ (S+R) \circ \lambda \circ W_1 s$$

$$- \lambda \circ W_4 \circ \lambda \circ W_3 \circ \lambda \circ S \circ \lambda \circ W_1 s|_i > t \Big) \tag{C.12}$$

Step 2: Use Matrix Norm Properties We can bound this using the ℓ_∞ norm and properties of matrix norms:

$$Z \leq \mathbb{P}\left(\|W_4\|_1 \cdot \|W_3\|_1 \cdot \|R \circ \lambda \circ W_1 s\|_\infty > t \right) \tag{C.13}$$

The rest of the proof then proceeds as above. □

For a more general DNN, we first restate the relevant assumptions:

Assumption 4:

Consider a DNN denoted by ϕ, and assume that it can be written as:

$$\phi = \rho \circ \psi_2 \circ (R + S) \circ \psi_1, \tag{C.14}$$

where ψ_1 and ψ_2 are arbitrary functions and ρ is softmax. Furthermore, assume that $\exists C_1$ constant such that for any two arbitrary vectors v and w we have that $\|\psi_2 v - \psi_2 w\|_\infty \leq C_1 \|\psi_2(v - w)\|_\infty$, with $\|\cdot\|_\infty$ the max norm of the vector. Take $W = R+S$ and assume that the matrices R and S satisfying the following assumptions:

Assumptions on the matrices R and S

We considered a class of admissible matrices W, where $W = R + S$ and W, R and S satisfy the following three assumptions. The first assumption is a condition on R:

Assumption 5: Assume R is a $N \times M$ matrix such that for any vector v we have:

$$\mathbb{P}(\|Rv\|_\infty > d_1(N) J(v)) \leq d_2(N), \tag{C.15}$$

with $d_1(N), d_2(N) \to 0$ as $N \to \infty$ and $J(v)$ some function that depends on v alone. Further, as $N \to \infty$, we have that $\sigma_{\max}(R) \to \sqrt{\lambda_+}$ a.s.

Example C.2.1. *As mentioned, when R is i.i.d Gaussian one can show, using the Borell-TIS inequality (see Theorem C.2.2), that:*

$$\mathbb{P}\left(\|Rv\|_\infty > (\sqrt{\frac{2\log N}{N}} + \frac{1}{N^{\frac{3}{8}}})\|v\|_2\right) \leq 2\exp\left(-\frac{N^{\frac{1}{4}}}{2}\right)$$

We now prove Lemma 4.5.4.

Proof. **Step 1: Apply Assumption 4**

Take

$$Z := \mathbb{P}\left(|\psi_2 \circ (R + S) \circ \psi_1 s - \psi_2 \circ (S) \circ \psi_1 s|_i > t\right). \tag{C.16}$$

Step 2: Factor Out ψ_2

We can now factor out ψ_2, using the inequality $\|(\psi_2 v - \psi_2 w)\|_{\max} \leq \|\psi_2\|_{\max\to\max}\|v - w\|_{\max}$, to move ψ_2 outside of the norm:

$$Z \leq \mathbb{P}\left(\|\psi_2\|_{\max\to\max} \cdot \|(R + S) \circ \psi_1 s - S \circ \psi_1 s)\|_{\max} > t\right) \tag{C.17}$$

Step 3: Focus on R

We now separate the terms involving R and S, leaving us with:

$$Z \leq \mathbb{P}\left(\|\psi_2\|_{\max\to\max} \cdot \|(R \circ \psi_1 s)\|_{\max} > t\right) \tag{C.18}$$

Step 4: Focus on the Random Matrix R From Assumption 5, for any $t > 0$:

$$\mathbb{P}\left(\|R \circ \psi_1 s\|_\infty > d_1(N)J(\psi_1 s))\right) \leq d_2(N))$$

Step 6: Incorporate ψ_2

Next, incorporate the effect of ψ_2. Using the properties of matrix norms, we have:

$$Z \leq \mathbb{P}(\|\psi_2\|_{\max\to\max}\|R \circ \psi_1 s\|_\infty > \|\psi_2\|_{\max\to\max} d_1(N)J(\psi_1 s)) \leq d_2(N)), \tag{C.19}$$

with $t = \|\psi_2\|_{\max\to\max} d_1(N)J(\psi_1 s)$.

□

Borell-TIS Inequality for i.i.d. Gaussian Variables

In this theorem, $t \in \mathbb{R}$ does not refer to training time. We use this variable t because classically the Borell-TIS is given in terms of t.

Theorem C.2.2 (Borell-TIS Inequality)**.** *Let $X_1, X_2, \ldots, X_n$ be i.i.d. centered Gaussian random variables with $X_i \sim N(0, \sigma^2)$. Set $s_X^2 := \max_{i=1,\ldots,n} \mathbb{E}(X_i^2) = \sigma^2$. Then for each $t > 0$:*

$$P\left(\max_{i=1,\ldots,n} |X_i| - \mathbb{E}\left[\max_{i=1,\ldots,n} X_i\right] > t\right) \leq \exp\left(-\frac{t^2}{2s_X^2}\right).$$

For the absolute value bound:

$$P\left(|\max_{i=1,\ldots,n} |X_i| - \mathbb{E}\left[\max_{i=1,\ldots,n} X_i\right]| > t\right) \leq 2\exp\left(-\frac{t^2}{2s_X^2}\right).$$

In particular, if $X_i \sim N(0, \frac{1}{N})$*:*

$$s_X^2 = \frac{1}{N},$$

and for any $t > 0$*:*

$$P\left(\max_{i=1,\ldots,n} |X_i| > \sqrt{\frac{2\log n}{N}} + t\right) \leq 2\exp\left(-\frac{Nt^2}{2}\right).$$

Generalized Chernoff Bound

In this and the next theorems, $t \in \mathbb{R}$ **does not refer to training time. We use this variable** t **because classically the Chernoff bound is given in terms of** t**.**

Theorem C.2.3 (Chernoff Bound)**.** *For any random variable* X *with moment-generating function* $M_X(t) = \mathbb{E}[e^{tX}]$ *and for any* $t > 0$*:*

$$P(X \geq a) = P(e^{tX} \geq e^{ta}) \leq M_X(t)e^{-ta}.$$

For the absolute value bound:

$$P(|X| \geq a) \leq M_X(t)e^{-ta} + M_{-X}(t)e^{-ta}.$$

If the distribution is symmetric, such that $M_X(t) = M_{-X}(t)$*:*

$$P(|X| \geq a) \leq 2M_X(t)e^{-ta}.$$

Chernoff Bound for Normal Distribution

Theorem C.2.4 (Chernoff Bound: normal distribution)**.** *For a random variable* X *that is normally distributed with* $X \sim N(0, 1/N)$*, the moment-generating function is* $M_X(t) = \mathbb{E}[e^{tX}] = e^{\frac{1}{2N}t^2}$*. For any* $t > 0$*:*

$$P(X \geq a) \leq e^{\frac{1}{2N}t^2}e^{-ta}.$$

By choosing $t = Na$*:*

$$P(X \geq a) \leq e^{-\frac{Na^2}{2}}.$$

For the absolute value bound:

$$P(|X| \geq a) \leq 2e^{-\frac{Na^2}{2}}.$$

C.2.3 Training of a DNN with three weight layers using L1 and L2 regularization with varying layer widths

The goal of this subsection is to check how $a(N) = \sqrt{\frac{2\log N}{N}}\|W_1 s\|_2\|W_3\|_1 + \frac{\|W_3\|_1\|W_1 s\|_2}{N^{1.5/4}}$ depends on N and the training process. From equation (C.10), we know that removing the random matrix R_2 from the DNN will change the components of the outputs X by less than $a(N)$. We want to see if this change in the output is small in practice. We train a DNN with three fully connected weight layers: $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}$. The middle layer matrix $\mathbf{W_2}$ is of variable width N, which we vary across different experiments. We aim to analyze how varying N affects both the network's accuracy and certain key norms of the weight matrices.

We use both L1 and L2 regularization during training, with the regularization strengths kept small (.0000001) so that they minimally affect the training process. Additionally, the learning rate is set to a small value of 0.0001 to ensure that the weight matrices do not change significantly during the training period.

Normalization of the Fashion MNIST Dataset

The Fashion MNIST dataset consists of grayscale images of size 28×28. To normalize the data, we ensure that the largest element across the entire dataset has an L2 norm of 0.1. This normalization is essential for controlling the magnitude of the input vector, which ensures that $a(N)$ stays small.

The normalization procedure involves the following steps:

1. Compute the L2 norm of each input vector (each image flattened to a 1D array).
2. Find the maximum norm among all vectors in the dataset.
3. Scale all input vectors so that the largest vector has norm 0.1 .

Training procedure

The network is trained for 10 epochs, and we keep the learning rate at 0.0001. The weights are initialized from $N(0, 1/N)$, with N the number of column vectors.

Analysis and metrics

For each value of N (the layer weight matrix W_2 has size $N \times N$), we compute two key quantities:

- $a(N)$: This is a metric based on the norms of the weight matrices, calculated as:
$$a(N) = \sqrt{\frac{2\log N}{N}}\|W_1 s\|_2 \|W_3\|_1 + \frac{\|W_3\|_1 \|W_1 s\|_2}{N^{1.5/4}}$$
where $\|W_3\|_1$ is the L1 norm of the final weight layer and $\|W_1 s\|_2$ is the largest L2 norm of the transformed input data.

- Test Accuracy: The percentage of correctly classified test samples, which allows us to track how network performance evolves as the width N increases.

We analyze the behavior of both $a(N)$ and the test accuracy as functions of the layer weight matrix W_2 of size $N \times N$. The values of N that we test for are $[500, 1000, 1500, 2000, 3000, 4000, 5000, 7000, 10000, 20000]$. Fig. C.8 shows that as we increase the size W_2, we have that $a(N)$ goes to zero quickly.

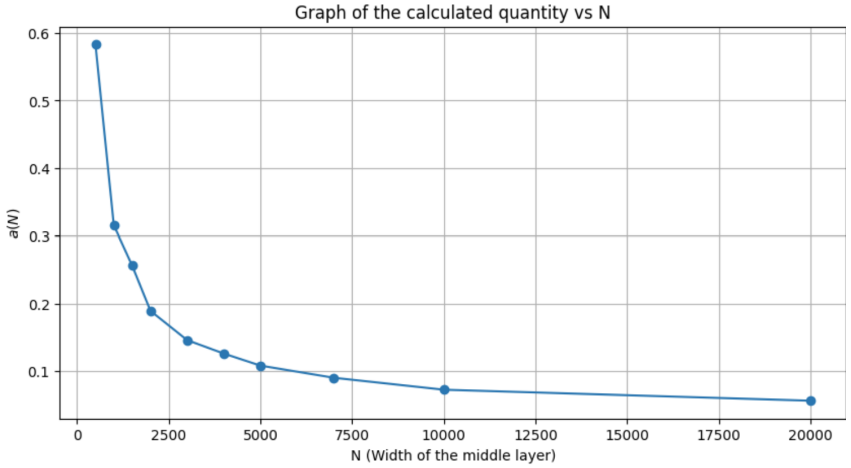


Figure C.8: Plot of $a(N)$ vs. the width N of the middle layer $\mathbf{W_2}$. This plot shows how the complexity-related metric $a(N)$ behaves as the width N is increased, reflecting changes in the norms of the weight matrices and their dependence on N.

Normalization and Initialization of Weight Matrices

In this experiment, we normalize the input vectors such that the largest input vector has a norm of 10. The normalization process is identical to that described previously, but instead of scaling the largest vector to 0.1, we scale it to 1.

Additionally, the weight matrices for each layer are initialized such that each entry is sampled from a normal distribution with zero mean and variance $1/N^2$, where N is the number of columns in the weight matrix.

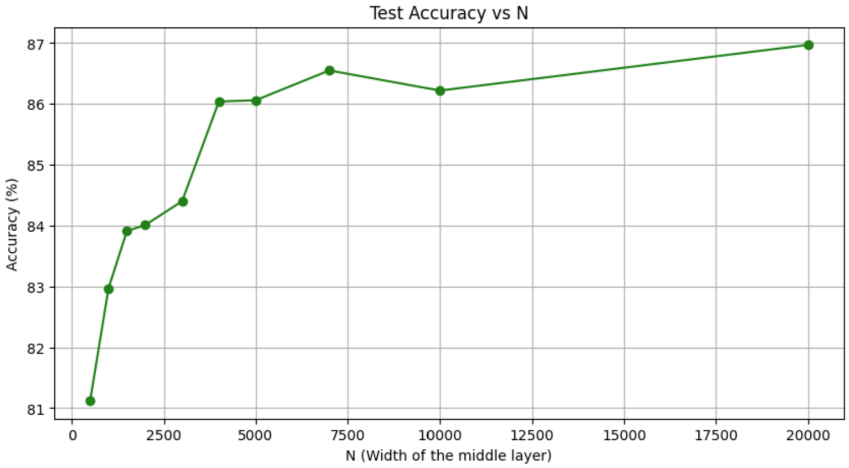


Figure C.9: Plot of test accuracy vs. the width N of the middle layer $\mathbf{W_2}$. This plot demonstrates how the performance of the DNN changes with increasing N, showing the relationship between network width and generalization ability on the Fashion MNIST dataset.

C.2.3.1 Analysis of $b(N)$

Similar to the previous metric $a(N)$, we introduce a new metric $b(N)$, which is computed exactly like $a(N)$, but each occurrence of N is squared in the formula. Specifically, $b(N)$ is defined as:

$$b(N) = \sqrt{\frac{2\log N^2}{N^2}}\|W_1 s\|_2\|W_3\|_1 + \frac{\|W_3\|_1\|W_1 s\|_2}{(N^2)^{1.5/4}}$$

where $\|W_3\|_1$ is the L1 norm of the final weight layer and $\|W_1 s\|_2$ is the largest L2 norm of the transformed input data.

Note, given the above initialization, equation C.10 has $b(N)$ on the LHS rather then $a(N)$.

Results and Observations

We analyze the behavior of both $b(N)$ and the test accuracy as functions of the width N of the middle weight matrix $\mathbf{W_2}$. The values of N tested are

$$500,\ 1000,\ 1500,\ 2000,\ 3000,\ 4000,\ 5000,\ 7000,\ 10000,\ 20000.$$

Here, the DNN was trained for 20 epochs with a learning rate of .001.

From Fig. C.10, we observe that as we increase the size of W_2, the metric $b(N)$ converges toward zero at a faster rate compared to $a(N)$. This suggests that the dependence on N^2 significantly impacts the network's behavior.

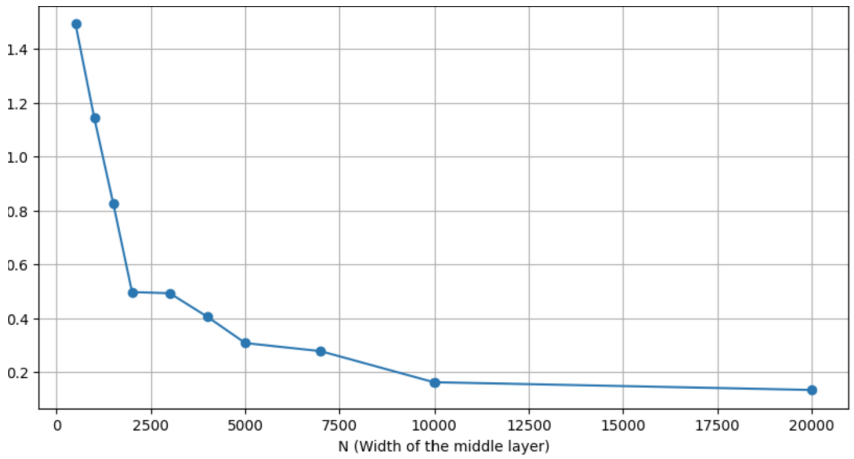


Figure C.10: Plot of $b(N)$ vs. the width N of the middle layer $\mathbf{W_2}$. This plot shows how the metric $b(N)$ behaves as the size of the middle layer increases, incorporating a stronger dependence on N^2.

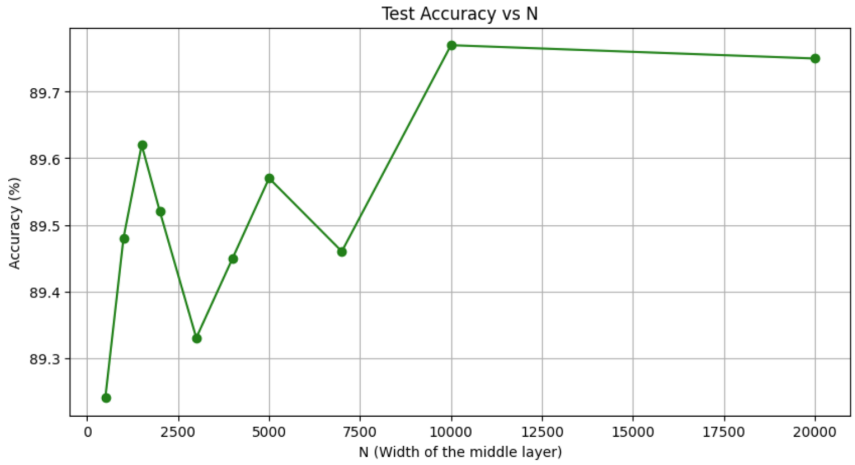


Figure C.11: Plot of test accuracy vs. the width N of the middle layer $\mathbf{W_2}$. This plot demonstrates how the network's accuracy changes as N increases, with weight matrices initialized using $1/N^2$.

Additionally, in Fig. C.11, we see the test accuracy improving as the width N increases, consistent with the expectations from larger model capacity, though the gains diminish at higher values of N. The training period for each network was kept constant at 10 epochs with a learning rate of 0.0001, ensuring consistent comparisons between the different experiments.

C.2.4 Some key lemmas: how accuracy and loss are affected when removing the random matrix R_2

Next, we introduce a lemma that describes how the loss, without regularization, and the accuracy of the DNN are affected when we remove the random matrix R_2 from

the weight layer matrix W_2. We define:

$$L_{\not{\text{reg}}}(\alpha(t)) = -\frac{1}{|T|}\sum_{s\in T}\log\left(\phi_{C(s)}(s,\alpha(t))\right). \tag{C.20}$$

thus

$$L_{\not{\text{reg}}} = -\frac{1}{|T|}\sum_{s\in T}\log\left(\frac{e^{X_{C(s)}(\alpha(t),s)}}{\sum_{i=1}^{K} e^{(X_i(\alpha(t),s))}}\right), \tag{C.21}$$

with K the number of classes in our classification problem.

We now introduce the *classification confidence*, see [33]. Take $X(s,\alpha)$ to be the output of the final layer in our DNN before softmax. The classification confidence is defined as follows:

$$\delta X(s,\alpha) := X_{C(s)}(s,\alpha) - \max_{j\neq C(s)} X_j(s,\alpha). \tag{C.22}$$

In other words,

- $\delta X(s,\alpha(t)) > 0 \Rightarrow s$ is well-classified by ϕ.
- $\delta X(s,\alpha(t)) < 0 \Rightarrow s$ is misclassified by ϕ

For T' the test set we can now define the accuracy of the DNN on T' using the classification confidence,

$$\text{acc}_\alpha(t) = \frac{\#\left(\{s\in T' : \delta X(s,\alpha(t)) > 0\}\right)}{\#T'}. \tag{C.23}$$

Lemma C.2.5. *Let ϕ be a DNN with weight layer matrices W_1, W_2, W_3 and λ the absolute value activation function:*

$$\phi(\alpha, s) = \rho\circ\lambda\circ W_3\circ\lambda\circ W_2\circ\lambda\circ W_1 s, \quad s\in\mathbb{R}^n. \tag{C.24}$$

Given (4.19)*, let W_2 be a $N\times N$ matrix such that $W_2(t) = R_2(t) + S_2(t)$, with R_2 and S_2 satisfying Assumptions 2-3.*

Suppose we replace the weight layer matrix W_2 with the matrix S_2. Then $\exists G_1(N)$ s.t.

$$\mathbb{P}\left(|L_{\not{reg}}(\alpha_{S_2}(t)) - L_{\not{reg}}(\alpha_{W_2}(t))| \leq G(N)H(\phi)\right) \geq 1 - G(N), \tag{C.25}$$

with $G_1(N)\to 0$ as $N\to\infty$ and $H(\phi)$ some function depending on ϕ with $H(\phi)\to 0$ as $N\to\infty$ if $a(N,s) := \frac{\|W_3\|_1\|W_1 s\|_2}{N^{1.5/4}} \to 0$ as $N\to\infty$.

See Subsection C.2.5 for a proof of this lemma. See also Example C.1.3 for a numerical example of this lemma.

C.2.5 Proof of Lemma C.2.5

Recall we define:

$$L_{\not{\text{reg}}}(\alpha(t)) = -\frac{1}{|T|}\sum_{s\in T}\log\left(\phi_{C(s)}(s,\alpha(t))\right). \tag{C.26}$$

thus

$$L_{\not{\text{reg}}} = -\frac{1}{|T|}\sum_{s\in T}\log\left(\frac{e^{X_{C(s)}(\alpha(t),s)}}{\sum_{i=1}^{K} e^{(X_i(\alpha(t),s))}}\right), \tag{C.27}$$

with K the number of classes in our classification problem.

From Lemma 4.4.4, we have that:

$$\mathbb{P}\left((|X_i(\alpha_W,s) - X_i(\alpha_S,s)| \le a(N,s)\right) \ge 1 - F(N), \tag{C.28}$$

with $F(N) \to 0$ as $N \to \infty$.

Thus, for any fixed $s \in T'$, we have that, for big enough N, $\exists G(N) \to 0$ as $N \to \infty$ such that:

$$\mathbb{P}\Bigg(\Bigg|\log\left(\frac{e^{X_{C(s)}(\alpha_S(t),s)}}{\sum_{i=1}^{K} e^{(X_i(\alpha_S(t),s))}}\right) - \log\left(\frac{e^{X_{C(s)}(\alpha_W(t),s)}}{\sum_{i=1}^{K} e^{(X_i(\alpha_W(t),s))}}\right)\Bigg| \le G(N)C(s,\phi)\Bigg) \ge 1 - G(N). \tag{C.29}$$

Furthermore, because the exponential and log functions are continuous, we have that if $a(N,s) \to 0$, then $G(N)C(s,\phi) \to 0$. Because that T' is a finite set, and because (C.29) holds for all $s \in T'$, we have that $\exists H(\phi)$ such that:

$$\mathbb{P}\Bigg(\Bigg|-\frac{1}{|T|}\sum_{s\in T}\log\left(\frac{e^{X_{C(s)}(\alpha_S(t),s)}}{\sum_{i=1}^{K} e^{(X_i(\alpha_S(t),s))}}\right) + \frac{1}{|T|}\sum_{s\in T}\log\left(\frac{e^{X_{C(s)}(\alpha_S(t),s)}}{\sum_{i=1}^{K} e^{(X_i(\alpha_S(t),s))}}\right)\Bigg| \le G(N)H(\phi)\Bigg) \ge 1 - G(N). \tag{C.30}$$

This proves the first part of Lemma C.2.5 and gives us equation (4.27).

C.3 Some known results on perturbation of matrices

Matrix perturbation theory is concerned with understanding how small changes in a matrix can affect its properties, such as eigenvalues, eigenvectors, and singular values. In this section, we state a couple of known results from matrix perturbation theory.

C.3.1 Asymptotics of singular values and singular vectors of deformation matrix

The results in this subsection are taken from [31]. Given the Properties 1 and 2 on R and S described in Section 4.5.1, the authors were able to show that the largest eigenvalues and corresponding eigenvectors of $W = S + R$ are well approximated by the largest eigenvalues and eigenvectors of S.

We start by defining the following function:

$$D_{\mu_R}(z) = \left[\int \frac{z}{z^2 - t^2} d\mu_R(t)\right] \times \left[c \int \frac{z}{z^2 - t^2} d\mu_R(t) + \frac{1-c}{z}\right] \tag{C.31}$$

for $z > \sqrt{\lambda_+}$, with λ_+ given by the MP distribution of $R^T R$. Take $D^{-1}_{\mu_R}(\cdot)$ to be its functional inverse.

Set

$$\bar{\theta} = D_{\mu_R}(\sqrt{\lambda_+})^{-\frac{1}{2}} \tag{C.32}$$

Theorem C.3.1. *Theorem for large singular values [Benaych-Georges and Nadakuditi (2012)] Take $W = R + S$, with W, R and S all $N \times M$ matrices satisfying assumptions $1' - 2'$. The r largest singular values of W, denoted as $\sigma'_i(W)$ for $1 \leq i \leq r$, exhibit the following behaviour as $N \to \infty$:*

$$\sigma'_i(W) \xrightarrow{a.s.} \begin{cases} D^{-1}_{\mu_R}(\frac{1}{(\sigma_i)^2}) & \sigma_i > \bar{\theta} \\ \sqrt{\lambda_+} & \sigma_i < \bar{\theta} \end{cases} \tag{C.33}$$

Example C.3.2. *Take $S = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ to be a $N \times N$ matrix, with σ_i the singular values and v_i and u_i the singular vectors of S. Take R to be a $N \times N$ random matrix with real i.i.d components taken from the normal distribution $N(0, \frac{1}{N})$. For $W = R + S$ we have:*

Theorem C.3.3. *(Theorem for large singular values for Example C.3.2) The r largest singular values of W, denoted $\sigma'_i(W)$ for $1 \leq i \leq r$, exhibit the following*

behaviour as $N \to \infty$*:*

$$\sigma_i'(W) \xrightarrow{a.s.} \begin{cases} \frac{1+\sigma_i^2}{\sigma_i} & \sigma_i > 1 \\ 2 & \sigma_i < 1 \end{cases}$$

C.3.2 Asymptotic behavior of singular values and vectors of the deformed matrix model given in Subsection 4.4.1

The following results can be found in [17, 31]. See [70] for other similar results.

Example C.3.4. *Take* $S = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ *to be a* $N \times N$ *matrix, with* σ_i *the singular values and* v_i *and* u_i *the singular vectors of* S*. Take* R *to be a* $N \times N$ *random matrix with real i.i.d. components taken from normal distribution* $N(0, \frac{1}{N})$*. For* $W = R + S$ *we have:*

Theorem C.3.5. *(Theorem for large singular values for Example C.3.2) The* r *largest singular values of* W*, denoted* $\sigma_i'(W)$ *for* $1 \leq i \leq r$*, exhibit the following behaviour as* $N \to \infty$*:*

$$\sigma_i'(W) \xrightarrow{a.s.} \begin{cases} \frac{1+\sigma_i^2}{\sigma_i} & \sigma_i > 1 \\ 2 & \sigma_i < 1 \end{cases}$$

Remark C.3.1. *We say that* $X_n \to X$ *in probability if* $\forall \epsilon\ \mathbb{P}(|X_n - X| > \epsilon) \to 0$ *as* $n \to \infty$*. One can show that* $X_n \to X$ *a.s. implies that* $X_n \to X$ *in law. Thus for Theorem C.3.3* $\forall \epsilon$ *we have:*

$$\mathbb{P}(|\sigma_i'(W) - \frac{1+\sigma_i^2}{\sigma_i}| > \epsilon) < B_N(\epsilon). \tag{C.34}$$

In our case, we want the variance of $R(t)$ to depend on t. That is, The matrix $W_b(t)$ is expressed as:

$$W_b(t) = R_b(t) + S_b(t), \quad (\textit{deformed matrix}) \tag{C.35}$$

where $R_b(t)$ is an $N \times M$ matrix with i.i.d. entries drawn from $N(0, \frac{g(t)}{N})$, with $g(t) \leq 1$ representing the training time-dependent variance, and $S_b(t)$ is a matrix.

Lemma C.3.6 (Lemma for large singular values with training time-dependent variance)**.** *Consider the matrix* $W_b(t) = R_b(t) + S_b(t)$*, where* $R_b(t)$ *is an* $N \times M$ *random matrix with i.i.d. entries drawn from* $N(0, \frac{g(t)}{N})$ *and* $S_b(t)$ *is a matrix. The* r *largest*

singular values of $W_b(t)$, denoted $\sigma_i'(W_b(t))$ for $1 \leq i \leq r$, exhibit the following behaviour as $N \to \infty$:

$$\sigma_i'(W_b(t)) \xrightarrow{a.s.} \sqrt{g(t)} \times \begin{cases} \frac{1+(\frac{\sigma_i}{\sqrt{g(t)}})^2}{\frac{\sigma_i}{\sqrt{g(t)}}} & \frac{\sigma_i}{\sqrt{g(t)}} > 1 \\ 2 & \frac{\sigma_i}{\sqrt{g(t)}} < 1 \end{cases}$$

Proof. Define $W' = R' + S'$ where $R' = \frac{1}{\sqrt{g(t)}} R_b(t)$ and $S' = \frac{1}{g(t)} S_b(t)$. The entries of R' are now drawn from $N(0, \frac{1}{N})$, matching the classical case.

For W', the classical theorem gives the behavior of its largest singular values. Specifically, if σ_i' are the singular values of S', then the singular values of W', denoted as $\sigma_i'(W')$, behave as:

$$\sigma_i'(W') \xrightarrow{a.s.} \begin{cases} \frac{1+\sigma_i'^2}{\sigma_i'} & \sigma_i' > 1 \\ 2 & \sigma_i' < 1 \end{cases}$$

Since $\sigma_i' = \frac{\sigma_i}{\sqrt{g(t)}}$, the theorem for W' becomes:

$$\sigma_i'(W') \xrightarrow{a.s.} \sqrt{g(t)} \times \begin{cases} \frac{1+(\frac{\sigma_i}{\sqrt{g(t)}})^2}{\frac{\sigma_i}{\sqrt{g(t)}}} & \frac{\sigma_i}{\sqrt{g(t)}} > 1 \\ 2 & \frac{\sigma_i}{\sqrt{g(t)}} < 1 \end{cases}$$

To revert to the original matrix $W_b(t)$, we multiply the singular values of W' by $\sqrt{g(t)}$. This gives the asymptotic behavior of the singular values of $W_b(t)$ as stated in the lemma. □

C.3.3 Proof of Theorems 4.4.1, 4.4.2, 4.5.2 and 4.5.3.

Proof. In this section, we prove Theorem 4.4.1. Using the more general Assumptions 4.5.1-4.5.2 for Theorem 4.5.2, we can use the proof for these results as well. We will prove the main result by showing that if the random component $R_2(t)$ of the weight matrix $W_2(t)$ is non-zero at a local minimum, it contradicts the assumption of a local minimum.

1. Notation and Assumptions:

- We assume that α^* is a local minimum of the loss function $L(\alpha(t))$ defined in Eq. (C.45).

- We assume that $\|R_2(\alpha^*)\|_2 \neq 0$, which implies the random component $R_2(\alpha^*)$ is not zero.

2. Constructing a DNN at the local minimum:

- We initialize a DNN with parameters α^* and a loss function $\bar{L}(\alpha(a))$ such that $\bar{L}(\alpha(0)) = L(\alpha^*)$. That is, we create a new training time, denoted with the variable a, such that when $a = 0$, the DNN parameters are those at the critical point α^*. This is achievable as the loss function is continuous.

3. Introducing a controlled random noise:

- We define a random matrix $R(a)$ with i.i.d. entries drawn from $N(0, \frac{g(a)}{N})$. We will choose $g(a)$ to vanish as a approaches infinity.

4. Key Inequality and its Consequence:

- We claim that there exists a constant $g(a)$ such that $g(a) \to 0$ as $a \to \infty$ and for all $a > 0$ and $N > N_0$ (for some N_0) we have that $\exists D(N)$ such that:

$$\mathbb{P}\Big(L(\alpha^*) \geq \bar{L}(\alpha(a))\Big) > 1 - D(N), \tag{C.36}$$

  with $D(N) \to 0$ as $N \to \infty$.

5. Proof of the Key Inequality:

- We will utilize the following properties:
  - Lemma C.2.5: This lemma states that removing the random component R_2 from the weight matrix W_2 has a minimal impact on the loss function with high probability. That is based on Lemma C.2.5, we know that:

$$\mathbb{P}\Big(|L_{\cancel{\text{reg}}}(\alpha_{S_2}(a)) - L_{\cancel{\text{reg}}}(\alpha_{W_2}(a))| \leq G(N)H(\phi)\Big) \geq 1 - G(N), \tag{C.37}$$

    where $G(N) \to 0$ as $N \to \infty$ and $\alpha_{S_2}(t)$ denotes the parameters of the DNN with the weight matrix $S_2(t)$.
  - Frobenius norm:

$$\|A\|_F = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}^2} = \sqrt{\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_N^2}. \tag{C.38}$$

– The loss:

$$L(\alpha(a)) = -\frac{1}{|T|}\sum_{s\in T}\log\left(\phi_{C(s)}(s,\alpha(a))\right) + \sum_{i=1}^{L}\mu\|W_i(a)\|_F^2, \quad \text{(C.39)}$$

– Lemma C.3.6. We are assuming that for the non-zero singular values of S we have, $\sigma_i(a) > 1$. Because $0 \leq g(a) \leq 1$, we have, by Lemma C.3.6, that:

$$\sigma_i'(W') \xrightarrow{a.s.} g(a)\frac{1+(\frac{\sigma_i}{\sqrt{g(a)}})^2}{\frac{\sigma_i}{\sqrt{g(a)}}}. \quad \text{(C.40)}$$

That is, $\exists D_2(N,\epsilon)$ such that $\forall\epsilon$:

$$\mathbb{P}\left(|\sigma_i'(W') - C(a)\frac{1+(\frac{\sigma_i}{\sqrt{C(a)}})^2}{\frac{\sigma_i}{\sqrt{C(a)}}}| > \epsilon\right) \leq D_2(N,\epsilon), \quad \text{(C.41)}$$

with $D_2(N,\epsilon) \to 0$ as $N \to \infty$.

• Take α^* to be a local min of (C.45), and suppose $\|R(\alpha^*)\|_2 \neq 0$. Initialize the DNN

$$\phi(\alpha, s) = \rho \circ \lambda \circ W_3 \circ \lambda \circ W_2 \circ \lambda \circ W_1 s \quad \text{(C.42)}$$

with parameters α^* and loss $\bar{L}(\alpha(a))$ s.t. $\bar{L}(\alpha(0)) = L(\alpha^*)$. Assume that $R_2(0)$ is a random matrix with i.i.d. $N(0, \frac{g(a=0)}{N})$, such that at $a = 0$ we have that $W_2(0) = R_2(0) + S_2(0)$ satisfying the assumptions in Subsection 4.4.1. Then there $\exists g(a) \to 0$ monotonically as $a \to \infty$, and take $R_2(a)$ to be a random matrix with i.i.d. taken from $N(0, \frac{g(a)}{N})$ and $W_2(a) = R_2(a) + S_2(0)$. By Lemma C.2.5, we know that:

$$\mathbb{P}\left(|L_{\not{\text{reg}}}(\alpha_{S_2}(a)) - L_{\not{\text{reg}}}(\alpha_{W_2}(a))| \leq G(N)H(\phi)\right) \geq 1 - G(N). \quad \text{(C.43)}$$

Furthermore, one can apply the same proof that was used to prove Lemma C.2.5 to show that $\forall a$:

$$\mathbb{P}\left(|L_{\not{\text{reg}}}(\alpha_{S_2}(0)) - L_{\not{\text{reg}}}(\alpha_{W_2}(a))| \leq G(N)H(\phi)\right) \geq 1 - G(N), \quad \text{(C.44)}$$

given that $W_2(a) = R_2(a) + S_2(0)$ and the random part $R_2(a)$ has i.i.d. components with zero mean and variance $\frac{g(a)}{N}$. That is, as $a \to \infty$ if the only

change in W_2 is that the random part R_2 goes to zero, then the loss $L_{\cancel{\text{reg}}}$ won't change significantly with probability depending on N.

However, the loss:

$$L(\alpha(a)) = -\frac{1}{|T|}\sum_{s\in T}\log\left(\phi_{C(s)}(s,\alpha(a))\right) + \mu\sum_{i=1}^{L}\|W_i(a)\|_F^2, \tag{C.45}$$

will be decreasing because the term $\mu\sum_{i=1}^{L}\|W_i(a)\|_F^2$ will get smaller monotonically as $a\to\infty$. This can be seen from the equations:

$$\|A\|_F = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}A_{ij}^2} = \sqrt{\sigma_1^2+\sigma_2^2+\ldots+\sigma_N^2}, \tag{C.46}$$

and (C.41). By (C.41), the r largest singular values of W_2 will becomes smaller as $a\to\infty$ (with some probability depending on N), making $\mu\sum_{i=1}^{L}\|W_i(a)\|_F^2$ smaller monotonically (again with probability depending on N). Furthermore, the $N-r$ smallest singular values of W_2 will also go to zero monotonically as $a\to\infty$, again ensuring that $\mu\sum_{i=1}^{L}\|W_i(a)\|_F^2$ decrease (with probability depending on N). Thus, because for (C.45) the term $-\frac{1}{|T|}\sum_{s\in T}\log\left(\phi_{C(s)}(s,\alpha(a))\right)$ does not change as $a\to\infty$ (with probability depending on N) and $\mu\sum_{i=1}^{L}\|W_i(a)\|_F^2$ decreases as $a\to\infty$ (again with probability depending on N), we have that the loss given in (C.45) must be decreasing as $a\to\infty$ (with probability depending on N). That is, for $a>0$, $\exists D(N)$ such that $\forall\epsilon$:

$$\mathbb{P}\left(L(\alpha^*) > \bar{L}(\alpha(a))\right) > 1-D(N). \tag{C.47}$$

That is, α^* is not a local min (with probability depending on N).

Furthermore, for a fixed training time t as $N\to\infty$ we have that removing the random matrix R_2 will not affect the cross entropy term in the loss but will remove the singular values of W_2 smaller then $\sqrt{\lambda_+}$. That is, the singular values of W_2 corresponding to those of R_2 will be set to zero, which means that the loss will decrease by $\mu\|R_2\|_F^2$. This proves Theorem 4.4.2. In fact, if $\|R_2\|_F>\epsilon$, then one can remove it (by sending $g(t)\to 0$), and the loss will decrease monotonically (with some probability depending on N) by $\mu\|R_2\|_F^2$, showing that we were not in a local min.

□

C.4 Algorithms for RMT-Based pruning of DNN

C.4.1 BEMA technique for computing λ_+

We outline the BEMA technique for identifying the optimal fit λ_+ for $\frac{1}{N}R^TR$ from the ESD of $X = R + S$. This is beneficial when analyzing matrices with information added to noise, aiming to pinpoint the furthest boundary of the MP distribution's compact support. The BEMA technique is known for its computational efficiency and precision for matrices with added noise. The complete procedure can be located in [157]. Below, we present a more streamlined version for an $N \times N$ matrix R:

Example C.4.1. *In this example, we create a random $N \times N$ matrix R with random i.i.d. $\mathcal{N}(0,1)$ (zero mean and unit variance ($\sigma^2 = 1$), Gaussian).*

In this example, we generate a random $N \times N$ matrix R, where each entry is independently sampled from a standard normal distribution $\mathcal{N}(0,1)$ with zero mean and unit variance ($\sigma^2 = 1$). We take S to be a $N \times N$ matrix with components given by

$$S[i,j] = \tan(\frac{\pi}{2} + \frac{1}{j+1}) + \cos(i) \cdot \log(i+j+1) + \sin(j) \cdot \cos(\frac{i}{j}), \qquad \text{(C.48)}$$

$W = R + S$ and $X = \frac{1}{N}W^TW$. The BEMA algorithm is used to find the λ_+ of the ESD of X, as described in Algorithm 2. R is a random matrix satisfying the conditions of Theorem 4.2.2, and so the ESD of $\frac{1}{N}R^TR$ converges to the Marchenko-Pastur distribution as $N \to \infty$ and has a λ_+ that determines the rightmost edge of its compact support. We can imagine a situation in which R is not directly known, and the goal is to find an estimator of λ_+ from the ESD of X. See Fig. C.12 for the result of the ESD of X with the Marchenko-Pastur distribution that best fits the ESD shown in red.

Remark C.4.1. *The method relies on parameters $\alpha \in (0, 1/2), \beta \in (0,1)$. This dependency can be observed by adjusting α and β as showcased in Fig. C.13. The highlighted line represents $\lambda_+ = 4$, the accurate λ_+ for $\frac{1}{N}R^TR$. In this case, while the influence of α is minimal for ample values, adjusting β regulates the assurance that the eigenvalues of the matrix R will remain below the λ_+ estimator of the MP distribution.*

The BEMA technique will be utilized to deduce λ_+ from $X_l(t)$'s ESD. As DNN learning evolves, most of the $X_l(t)$ eigenvalues are anticipated to fit the MP distribution. Yet, certain eigenvalues might surpass the MP distribution's bulk and

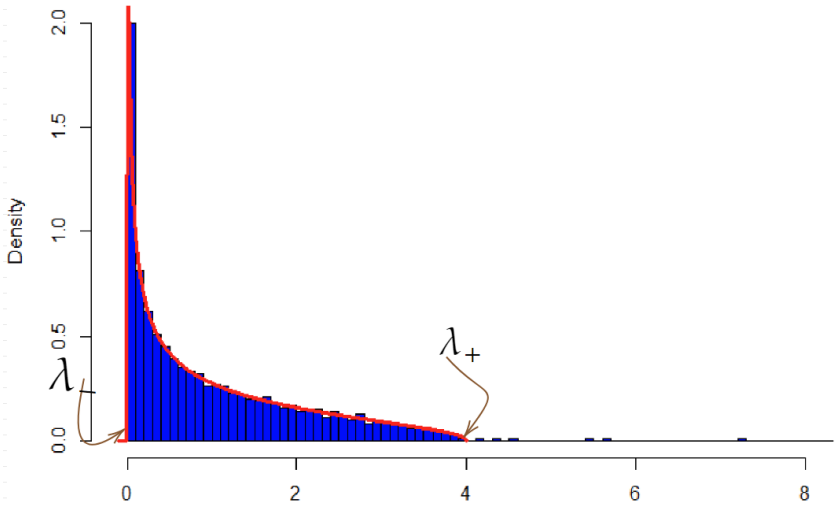
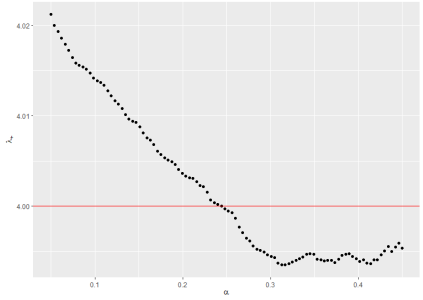
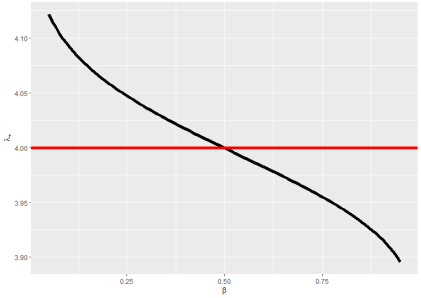


Figure C.12: In blue we have the ESD of X, in red the Marchenko-Pastur distribution which best fits the ESD based on the BEMA algorithm.



(a) Dependence of algorithm the choice of α, $\beta = 0.5$. In this example the rank of the matrix S is fairly low.



(b) Dependence of algorithm on the choice of β, $\alpha = 0.25$.

Figure C.13: How λ_+ depends on the hyperparameters α and β.

correspond to $S_l(t)$'s singular values. The BEMA's objective is to spot the MP distribution's outermost boundary, which aids in assessing λ_+. Grasping λ_+ is pivotal as it provides an understanding of the DNN's functionality during learning and its adaptability to fresh data.

Together with the SVD, the BEMA technique can guide which singular values of DNN's weight matrices W_l to omit during learning. The SVD dissects the weight matrix, yielding a breakdown into singular values and vectors, which facilitates an RMT-driven analysis of their distribution. By employing the BEMA, one can distinguish eigenvalues corresponding to S_l's singular values from those linked to R_l's singular values. Removing eigenvalues tied to R_l can enhance the DNN learning process's effectiveness.

C.4.2 Importance of singular value decomposition in Deep Learning

For a matrix A of dimensions $N \times M$, its SVD comprises a factorization $A = U\Sigma V^T$, where:

- U is a square orthogonal matrix of size $N \times N$.
- V is a square orthogonal matrix of size $M \times M$.
- Σ is a diagonal matrix of dimensions $N \times M$, with its diagonal entries corresponding to singular values σ_i, and non-diagonal entries being zero.

For matrix $X = W^T W$, with eigenvalues λ_i, the associated singular values of W are denoted by $\sigma_i = \sqrt{\lambda_i}$. This ties singular values to eigenvalues of a matrix W's symmetrization.

For a DNN's weight matrix W_l, studies have shown that omitting insignificant singular values through its SVD during training can diminish parameters yet augment accuracy. In the upcoming sections, we will elucidate how RMT can assist in determining which singular values to exclude from a DNN layer, ensuring the DNN's precision remains intact.

Specifically, integrating the BEMA technique with the SVD of W_l can determine which singular values to exclude during training. This is accomplished by first computing W_l's SVD, followed by determining the eigenvalues of the symmetrical matrix $X_l = \frac{1}{N} W_l^T W_l$. The derived eigenvalues can be related to the singular values of W_l via $N\lambda_i = \sigma_i^2$. Using the BEMA to estimate λ_+ enables setting a threshold for W_l's singular values. Those below this threshold can be discarded without affecting accuracy, as they are probably less vital for the DNN's output. This procedure can be recurrent during training, updating the threshold as required.

C.4.3 How well does the ESD of X fit the MP Distribution with Spikes

In this subsection, we outline a method to investigate whether the ESD of X may be derived from a particular MP distribution that may contain spiked eigenvalues. The starting point of this method uses the BEMA technique to pinpoint the most appropriate MP distribution. The best-fitting distribution offers a theoretical cumulative distribution function (CDF), while the observed cumulative spectral distribution for X is obtainable. Contrasting these distributions, we can rule out the idea that

X is governed by the proposed MP distribution if there is a notable disparity between them. We will now elaborate on these notions, beginning with the empirical cumulative spectral distribution.

Definition C.4.2. *Consider G as an $N \times M$ matrix with its ESD μ_{G_M} as presented in Definition 4.2.1. The observed cumulative spectral distribution of G, denoted as $F_G : \mathbb{R} \to \mathbb{R}$, is:*

$$F_G(a) = \mu_{G_m}((-\infty, a]) \tag{C.49}$$

Remarkably, the CDFs for the MP distribution are available in an explicit form. Using these formulas, we can lucidly detail our approach. A tuning parameter $\gamma \in (0, 1)$ specifies the sensitivity of our evaluation.

This approach measures the utmost disparity between the predicted and observed CDFs by sampling at each point in the observed distribution. As this is meant for the special scenario of testing for MP distributions with spikes, this knowledge allows us to refine our evaluation over simply determining the L^∞ disparity between the two distributions.

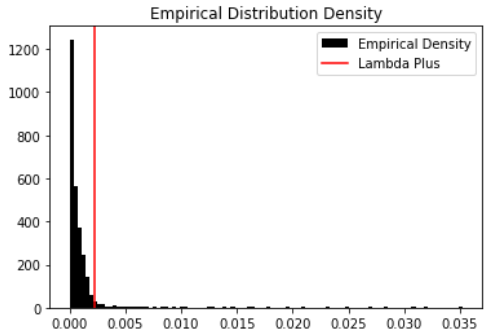
This refinement is evident in the step that determines $i_{\min}$ and $i_{\max}$. Given that BEMA solely utilizes data in the quantile between $(\alpha, 1 - \alpha)$ for the optimal fit, it's reasonable to inspect the fit within the same boundaries. In this scenario, a spiked MP distribution would likely be inadequately represented by its generative MP distribution around the predominant eigenvalues (the spikes), so it's logical to test only the core values for fit accuracy.

C.4.4 Eliminating singular values while preserving accuracy

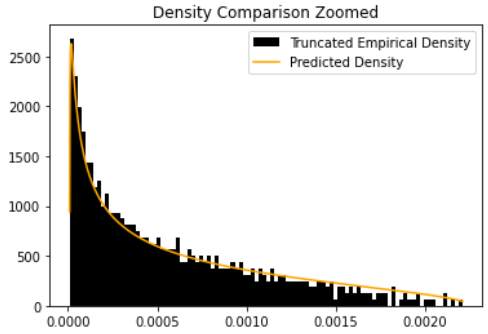
Algorithm 4 is how we train DNNs together with MP-based pruning.

Example C.4.3. *Consider an original DNN with two hidden layers, each consisting of 10 nodes. The total number of parameters in this case would be 100. By employing SVD and removing 8 small singular values in the weight layer matrix of this DNN, we can split the hidden layer into two, resulting in a new DNN with three hidden layers. The first layer will have 10 nodes, the second layer will have 2 nodes, and the third layer will have 10 nodes. By keeping only two singular values in the SVD, we now have only 20 parameters, see Figure C.15. In practice, we don't actually split the layer*

Example C.4.4. *We used the above approach for a DNN trained on MNIST. In this example the DNN has two layers, the first with a 784×1000 matrix W_1 and the*



(a) Full Empirical Density



(b) Zoomed Density

Figure C.14: The ESD of X_l and its best fit MP distribution

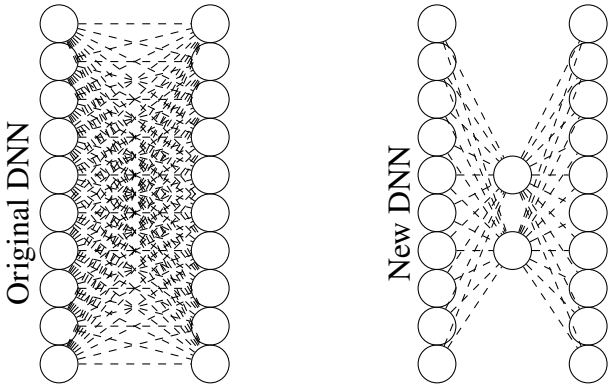


Figure C.15: Original DNN with two hidden layers, each with 10 nodes (total 100 parameters), is transformed into a new DNN with three hidden layers. The first layer has 10 nodes, the second layer has 2 nodes (keeping only two singular values in the SVD), and the third layer has 10 nodes, resulting in a total of 20 parameters.

second with a 1000×10 *matrix* W_2*. The activation function was ReLU. We trained the DNN for* 10 *epocs and achieved a* 98% *accuracy on the test set.*

We perform a SVD on W_1*, in this case* Σ *is a* 784×1000 *matrix. Even if we only keep the biggest* 60 σ_i *of* W_1 *and transform the first layer into two layers* $W_{1,1}$ *and* $W_{2,1}$ *the accuracy is still* 98%*.* W_1 *has* $784 \times 1000 = 784000$ *parameters but* $W_{1,1}$ *and*

$W_{2,1}$ together have $60 \times 784 + 60 \times 1000 = 106440$ parameters. This is a significant reduction in the number of parameters without any loss of accuracy.

C.4.5 Algorithm for Sparsifying Singular Vectors of a DNN Layer

Let W_l denote the l-th layer matrix of a DNN and for simplicity we assume that it is $N \times N$. The SVD of W_l is given by $W_l = U\Sigma V^T$, where U and V are the left and right singular vectors, and Σ is a diagonal matrix of singular values.

Given a threshold $\sqrt{\lambda_+ N}$, the algorithm proceeds as follows:

Note: The algorithm focuses on sparsifying the singular vectors based on the dynamic threshold determined by the singular values and a predefined threshold θ. Here, apply_sparsity$(U_{:,i}, T_i(\Sigma_{ii}))$ sets to zero all elements in the vector $U_{:,i}$ below the threshold $T_i(\Sigma_{ii})$.

C.4.6 Algorithm for element-wise sparsification of a matrix

C.4.7 The MP and spike metrics of a DNN model

First, we present two figures of the ESD of different layers of the base ViT model studied in Subsection 4.3, see Figs. C.16, C.17. We see that some layers fit the MP distribution "very" well, while some layers don't. When pruning layers of the DNN, we remove larger parameters in the layers that are more random because of the larger amount of randomness in those layers.

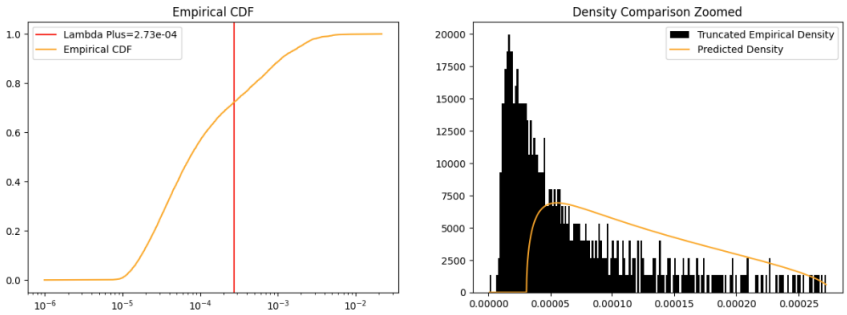


Figure C.16: Best fit MP distribution (on the left) for $\alpha = .25$ and $\beta = .8$ for a layer of the base ViT model. Here the MP fit error is .74 while the percentage of singular values smaller than $\sqrt{\lambda_+}$ is 73%. For more on the right figure, see Section C.2.

Second, we show how the hyperparameter α discussed in Remark C.4.1 affects the MP fit and spike metrics; see Fig. C.18.

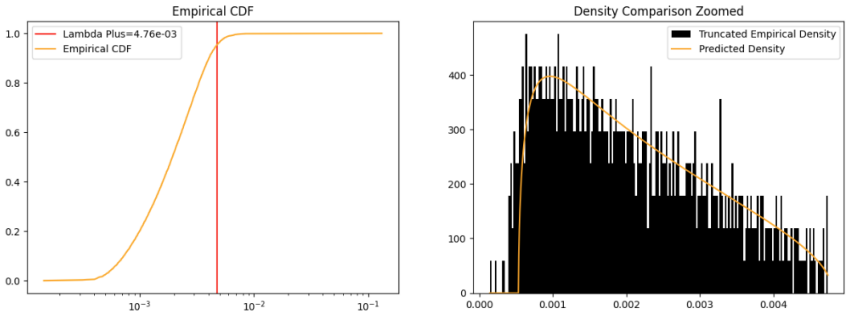


Figure C.17: Best fit MP distribution (on the left) for $\alpha = .25$ and $\beta = .8$ for a layer of the base ViT model. Here the MP fit error is .01 while the percentage of singular values smaller than $\sqrt{\lambda_+}$ is 99.73%. For more on the right figure, see Section C.2.

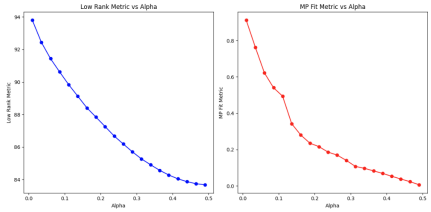


Figure C.18: In the left figure, we have α vs LRM and on the right we have α vs MP fit metric. This is for the base ViT model studied in 4.1.

Appendix D

SUPPLEMENTARY INFORMATION FOR CHAPTER 5: COMPUTATIONAL HYPERGRAPH DISCOVERY

Supplementary information

This supplementary document provides an overview of refinements and generalizations on our proposed approach (Sec. D.1) detailed in subsequent sections. It includes a summary of the principal components of our algorithm (Sec. D.2). It includes a reminder on Type 2 problems (Sec. D.3) and their common GP-based solutions. It discusses the hardness of Type 3 problems, presents an overview of causal inference methods, and a well-posed formulation of Type 3 problems (Sec. D.4). Additionally, this document offers an in-depth description of our developed GP-based solution specifically designed for Type 3 problems (Section D.5), along with the corresponding algorithmic pseudo-codes (Section D.6). It also includes an analysis of the signal-to-noise ratio (SNR) test that is integral to our method (Section D.7), and furnishes supplementary details concerning the examples discussed in the main manuscript (Section D.8).

D.1 Additional details on our proposed approach.

The efficacy of our proposed approach is enhanced through a series of refinements (implemented in all our examples), which are summarized below and detailed in sections D.5, D.6 and D.7.

D.1.1 Ancestor pruning.

As discussed earlier, rather than using a threshold on the signal-to-noise ratio to prune ancestors, we order the ancestors in decreasing contribution to the signal, the final number q of ancestors is determined as the maximizer of noise to signal ratio increment $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q+1) - \frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$.

D.1.2 Parameter Selection.

The choice of the parameter γ in (5.2) is a critical aspect of our proposed approach. We provide a structured approach for selecting γ based on the characteristics of the kernel matrix K_s. Specifically, when K_s is derived from a finite-dimensional feature map ψ (i.e., when $K_s(x, x') := \psi(x)^T\psi(x')$ where the range of ψ is finite-

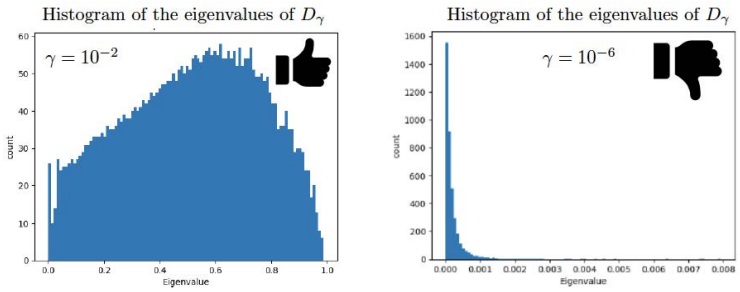


Figure D.1: Histogram of the eigenvalues of D_γ=(D.2) for $\gamma = 10^{-2}$ (good choice) and $\gamma = 10^{-6}$ (bad choice).

dimensional) and the data cannot be interpolated exactly with K_s (the dimension of the range of ψ is smaller than the number of data points), we employ the regression residual to determine γ as follows:

$$\gamma = \min_v \left\| v^T \psi(X) - Y \right\|^2_{\mathbb{R}^N} . \tag{D.1}$$

Write $K_s(X, X)$ for the $N \times N$ matrix with entries $K_s(X_i, X_j)$. Alternatively, when the data can be interpolated exactly with K_s (e.g., when K_s is a universal kernel), we select γ (see Fig. D.1) by maximizing the variance of the eigenvalue histogram of the $N \times N$ matrix

$$D_\gamma := \gamma \big(K_s(X, X) + \gamma I \big)^{-1} , \tag{D.2}$$

whose eigenvalues are bounded between 0 and 1 and converge towards 0 as $\gamma \downarrow 0$ and towards 1 as $\gamma \uparrow \infty$. We can also select γ as the median of the eigenvalues of D_γ.

D.1.3 Z-test quantiles.

The noise-to-signal ratio $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}$ associated with (5.2) admits the representer formula $\frac{Y^T D_\gamma^2 Y}{Y^T D_\gamma Y}$. Therefore if the data is only comprised of noise (if $Y \sim \sigma^2 Z$ where Z is a random vector with i.i.d. $\mathcal{N}(0, 1)$ entries), then the distribution of the noise-to-signal ratio follows that of the random variable

$$B := \frac{Z^T D_\gamma^2 Z}{Z^T D_\gamma Z} . \tag{D.3}$$

Therefore, the quantiles of B can be used as an interval of confidence on the noise-to-signal ratio if $Y \sim \sigma^2 Z$. Fig. 5.3.(c) shows these Z-test quantiles (in the absence of signal, the noise-to-signal ratio should fall within the shaded area with probability 0.9).

D.1.4 Generalizations on our proposed approach.

D.1.4.1 Complexity Reduction with Kernel PCA Variant.

Write K for the kernel associated with the RKHS $\mathcal{H}$ in Problem 1. We use a variant of Kernel PCA [216] to significantly reduces the computational complexity of our proposed method, making it primarily dependent on the number of principal nonlinear components in the kernel matrix $K(X, X)$ (the $N \times N$ matrix with entries $K(X_i, X_j)$) rather than the number of data points. To describe this write $\lambda_1 \geq \cdots \geq \lambda_r > 0$ for the nonzero eigenvalues of $K(X, X)$ indexed in decreasing order and write $\alpha_{\cdot,i}$ for the corresponding unit-normalized eigenvectors, i.e. $K(X, X)\alpha_{\cdot,i} = \lambda_i \alpha_{\cdot,i}$. Then $|f(X)|^2 = |f(\phi)|^2$, where $f(\phi)$ is the r vector with entries $f(\phi_i) := \sum_{s=1}^{N} f(X_s)\alpha_{s,i}$. Furthermore, writing $r' \leq r$ for the smallest index i such that $\lambda_i/\lambda_1 < \epsilon$ where $\epsilon > 0$ is some small threshold, the complexity of the problem can be further reduced (as in PCA) by truncating $f(\phi)$ to $f(\phi') = (f(\phi_1), \ldots, f(\phi_{r'}))$ and approximating $\mathcal{F}$ with the space of functions $f \in \mathcal{H}$ such that $|f(\phi')|^2 \approx 0$.

D.1.4.2 Generalizing Descendants and Ancestors with Kernel Mode Decomposition.

We can extend the concept of descendants and ancestors to cover more complex functional dependencies between variables, including implicit ones. This generalization is achieved through a Kernel-based adaptation of Row Echelon Form Reduction (REFR), initially designed for affine systems, and leveraging the principles of Kernel Mode Decomposition [243]. To describe the connection with REFR consider the example in which $\mathcal{M}$ is the manifold of $\mathbb{R}^3$ defined by the affine equations $x_1 + x_2 + 3x_3 - 2 = 0$ and $x_1 - x_2 + x_3 = 0$, which is equivalent to selecting $\mathcal{F} = \text{Span}\{f_1, f_2\}$ with $f_1(x) = x_1 + x_2 + 3x_3 - 2$ and $f_2(x) = x_1 - x_2 + x_3$ in the problem formulation 1. Then, irrespective of how we recover the manifold from data, the hypergraph representation of that manifold is equivalent to the row echelon form reduction of the affine system, and this representation and this reduction require a possibly arbitrary choice of free and dependent variables. So, for instance, if we declare x_3 to be the free variables and x_1 and x_2 to be the dependent variables, then we can represent the manifold via the equations $x_1 = 1 - 2x_3$ and $x_2 = 1 - x_3$ which have the hypergraph representation depicted in Fig. D.6.(b). To describe the kernel generalization of REFR assume that the kernel K can be decomposed as the additive kernel

$$K = K_a + K_s + K_z\,, \tag{D.4}$$

and write $\mathcal{H}_a$, $\mathcal{H}_s$, and $\mathcal{H}_z$ for the RKHS induced by the kernels K_a, K_s, K_z. Then a function $f \in \mathcal{H}$ can be decomposed as $f = f_a + f_s + f_z$ with $(f_a, f_s, f_z) \in \mathcal{H}_a \times \mathcal{H}_s \times \mathcal{H}_z$. Then, generalizing REFR we can approximate the manifold $\mathcal{M}$ via a manifold parametrized by equations of the form

$$f_a + f_s + f_z = 0 \Leftrightarrow g_a = f_s \tag{D.5}$$

where $f_a = -g_a$ and g_a is a given function in $\mathcal{H}_a$ representing a dependent mode, $f_z = 0$ represents a zero mode, and $f_s \in \mathcal{H}_s$ is identified (regularized) as the minimizer of the following variational problem

$$\min_{f_s \in \mathcal{H}_s} \|f_s\|_{K_s}^2 + \frac{1}{\gamma}\left|(-g_a + f_s)(\phi)\right|^2 . \tag{D.6}$$

Taking $g_a(x) = x_1$ and $\mathcal{H}_s + \mathcal{H}_z$ to be a space of functions that does not depend on x_1 recovers our initial example (5.1) (with the pruning process encoded into the selection of $\mathcal{H}_z$). This generalization is motivated by its potential to recover implicit equations. For example, consider the implicit equation $x_1^2 + x_2^2 = 1$, which can be retrieved by setting the mode of interest to be $g_a(x) = x_1^2$ and allowing f_s to depend only on the variable x_2.

D.2 Algorithm Overview for Type 3 problems: An Informal Summary

In this section, we provide an accessible overview of our algorithm's key components, which are further detailed in Algorithms 7 and 8 in Section D.6. Our method focuses on determining the edges within a hypergraph. To achieve this, we consider each node individually, finding its ancestors and establishing edges from these ancestors to the node in question. While we present the algorithm for a single node, it can be applied iteratively to all nodes within the graph.

Algorithm for finding the ancestors of a node:

1. **Initialization:** We start by assuming that all other nodes are potential ancestors of the current node.

2. **Selecting a Kernel**: We choose a kernel function, such as linear, quadratic, or fully nonlinear kernels (refer to Example D.5.4). The kernel selection process is analogous to the subsequent pruning steps, involving the determination of a parameter γ, regression analysis, and evaluation based on signal-to-noise ratios.

- **Kernel Selection Method:** The choice of kernel follows a process similar to the subsequent pruning steps, including γ selection, regression analysis, and signal-to-noise ratio evaluation.
- **Low Signal-to-Noise Ratio for All Kernels:** If the signal-to-noise ratio is insufficient for all possible kernels, the algorithm terminates, indicating that the node has no ancestors.

3. **Pruning Process:** While there are potential ancestors left to consider (details in Section D.5.3.5):

   a) **Identify the Least Important Ancestor:** Ancestors are ranked based on their contribution to the signal (see Sec. D.5.3.3).

   b) **Noise prior:** Determine the value of γ (see Section D.7.2).

   c) **Regression Analysis:** Predict the node's value using the current set of ancestors, excluding the least active one (i.e., the one contributing the least to the signal). We employ Kernel Ridge Regression with the selected kernel function and parameter γ (see Sec. D.5.3.3 and D.5.3.3).

   d) **Evaluate Removal:** Compute the regression signal-to-noise ratio (see Sec. D.5.3.4 and D.7):

      - **Low Signal-to-Noise Ratio:** If the signal-to-noise ratio falls below a certain threshold, terminate the algorithm and return the current set of ancestors (see Section D.5.3.6).
      - **Adequate Signal-to-Noise Ratio:** If the signal-to-noise ratio is sufficient, remove the least active ancestor and continue the pruning process.

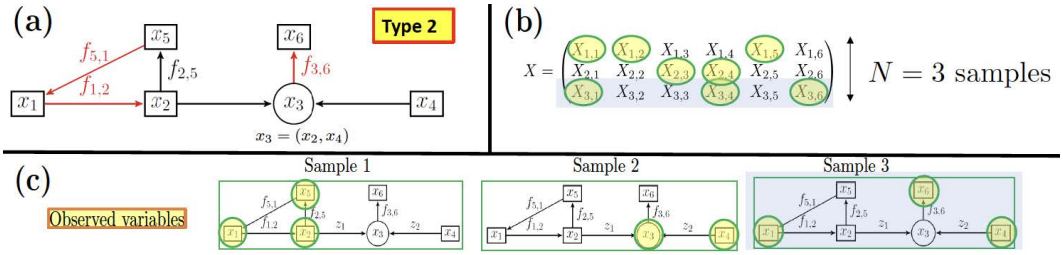


Figure D.2: Formal description of Type 2 problems.

D.3 Type 2 problems: Formal description and GP-based Computational Graph Completion

D.3.1 Formal description of Type 2 problems

Consider a computational graph (as illustrated in Fig. D.2.(a)) where nodes represent variables and edges are directed and they represent functions. These functions may be known or unknown. In Fig. D.2.(a), edges associated with unknown functions ($f_{5,1}$, $f_{1,2}$, $f_{3,6}$) are colored in red, and those associated with known functions ($f_{2,5}$) are colored in black. Round nodes are utilized to symbolize variables, which are derived from the concatenation of other variables (e.g, in Fig. D.2.(a), $x_3 = (x_2, x_4)$). Therefore, the underlying graph is, in fact, a hypergraph where functions may map groups of variables to other groups of variables, and we use round nodes to illustrate the grouping step. Given partial observations derived from N samples of the graph's variables, we introduce a problem, termed a Type 2 problem, focused on approximating all unobserved variables and unknown functions. Using Fig. D.2.(a)-(b) as an illustration we call a vector $(X_{s,1}, \ldots, X_{s,6})$ a sample from the graph if its entries are variables satisfying the functional dependencies imposed by the structure of the graph (i.e., $X_{s,1} = f_{5,1}(X_{s,5})$, $X_{s,2} = f_{1,2}(X_{2,s})$, $X_{s,3} = (X_{s,2}, X_{s,4})$, $X_{s,5} = f_{s,5}(X_{s,s})$, and $X_{s,6} = f_{3,6}(X_{s,3})$. These samples can be seen as the rows of given matrix X illustrated in Fig. D.2.(b) for $N = 3$. By partial observations, we mean that only a subset of the entries of each row may be observed, as illustrated in Fig. D.2.(b)-(c). Note that a Type 2 problem combines a regression problem (approximating the unknown functions of the graph) with a matrix completion/data imputation problem (approximating the unobserved entries of the matrix X).

D.3.2 Reminder on Computational Graph Completion for Type 2 problems

Within the context of Sec. D.3.1, the proposed GP solution to Type 2 problems is to simply replace unknown functions by GPs and compute their Maximum A Posteriori (MAP)/Maximum Likelihood Estimation (MLE) estimators given available data and constraints imposed by the structure of the graph. Taking into account the example depicted in Fig. D.2, and substituting $f_{5,1}$, $f_{1,2}$, and $f_{3,6}$ with independent GPs, each with kernels K, G, and Γ respectively, the objective of this MAP solution becomes minimizing $\|f_{5,1}\|_K^2 + \|f_{1,2}\|_G^2 + \|f_{3,6}\|_\Gamma^2$ (writing $\|f\|_K$ for the RKHS norm of f induced by the kernel K) subject to the constraints imposed by the data and the functional dependencies encoded into the structure of the graph.

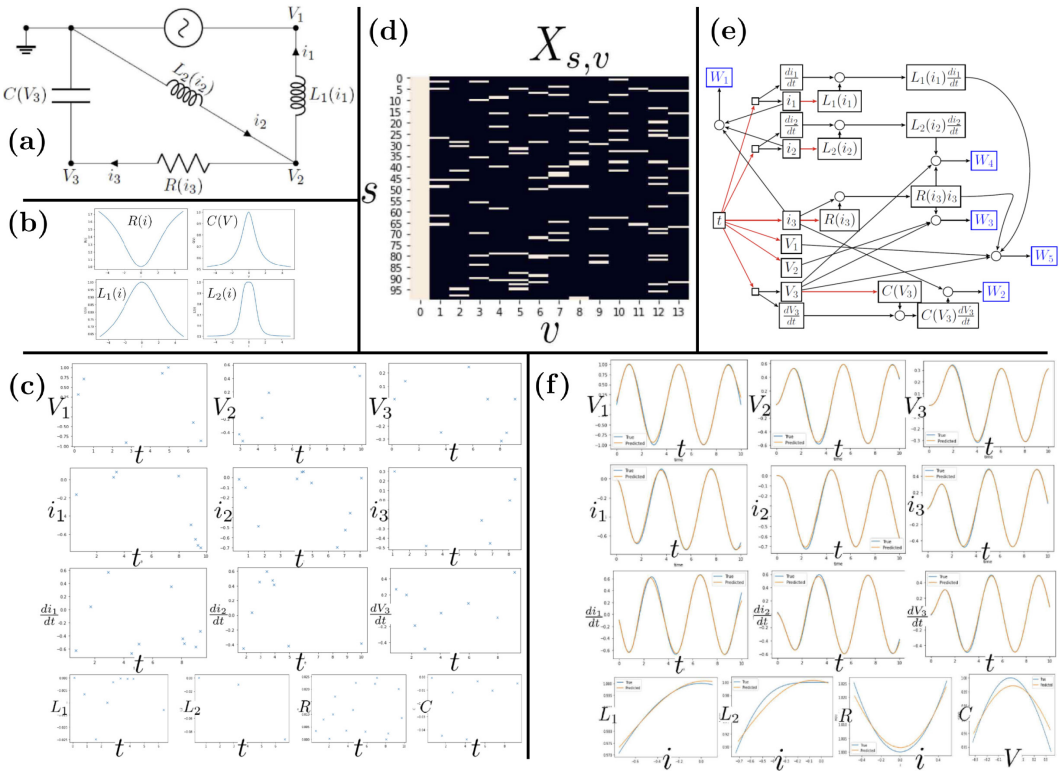


Figure D.3: (a) Electric circuit. (b) Resistance, capacitance, and inductances are nonlinear functions of currents and voltages (c) Measurements. (d) Kirchhoff's circuit laws. (e) The computational graph with unknown functions represented as red edges. (f) Recovered functions.

D.3.3 A system identification example.

In order to exemplify Computational Graphical Completion (CGC), consider the system identification problem depicted in Fig. D.3, sourced from [239]. Our objective is to identify a nonlinear electric circuit, as illustrated in Fig. D.3.(a), from scarce measurement data. The nonlinearity of the circuit emanates from the resistance, capacitance, and inductances, which are nonlinear functions of currents and voltages, as shown in Fig. D.3.(b). Assuming these functions to be unknown, along with all currents and voltages as unknown time-dependent functions, we operate the circuit between times 0 and 10. Measurements of a subset of variables, representing the system's state, are taken at times $t_s = s/10$ for $s \in 0, \ldots, 99$. Given these measurements, the challenge arises in approximating all unknown functions that define currents and voltages as time functions, capacitance as a voltage function, and inductances and resistance as current functions. Fig. D.3.(c) displays the available measurements, which are notably sparse, preventing us from reconstructing the underlying unknown functions independently. Thus, their interdependencies must

be utilized for approximation. It is crucial to note that the system's state variables are interconnected through functional relations, as per Kirchhoff's laws for this nonlinear electric circuit, illustrated in Fig. D.3.(d). These functional dependencies can be conceptualized as a computational graph, depicted in Fig. D.3.(e), where nodes represent variables and directed edges represent functions. Known functions are colored in black, unknown functions in red, and round nodes aggregate variables, meaning edges map groups of variables, forming a hypergraph. The CGC solution involves substituting the graph's unknown functions with Gaussian Processes (GPs), which may be independent or correlated, and then approximating the unknown functions with their Maximum A Posteriori (MAP) estimators, given the available data and the functional dependencies embedded in the graph's structure. Fig. D.3.(f) showcases the true and recovered functions, demonstrating a notably accurate approximation despite the data's scarcity.

This simple example generalizes to an abstract framework detailed in [239]. This framework has a wide range of applications because most problems in CSE can also be formulated as completing computational graphs representing dependencies between functions and variables, and they can be solved in a similar manner by replacing unknown functions with GPs and by computing their MAP/EB estimator given the data. These problems include those illustrated in Fig. 5.1.(d-h).

D.4 Hardness and well-posed formulation of Type 3 problems.

In this subsection, we describe why Type 3 problems are challenging and why they can even be intractable if not formalized and approached properly.

D.4.1 Curse of combinatorial complexity.

First, the problem suffers from the curse of combinatorial complexity in the sense that the number of hypergraphs associated with N nodes blows up rapidly with N. As an illustration, Fig. D.4 shows some of the hypergraphs associated with only three nodes. A lower bound on that number is the A003180 sequence, which answers the following question [150]: given N unlabeled vertices, how many different hypergraphs in total can be realized on them by counting the equivalent hypergraphs only once? For $N = 8$, this lower bound is $\approx 2.78 \times 10^{73}$.

D.4.2 Nonidentifiability and implicit dependencies.

Secondly, it is important to note that, even with an infinite amount of data, the exact structure of the hypergraph might not be identifiable. To illustrate this point, let's

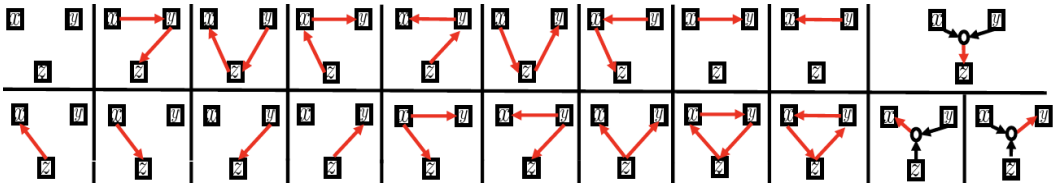


Figure D.4: Computational Hypergraph Discovery with three variables

consider a problem where we have N samples from a computational graph with variables x and y. The task is to determine the direction of functional dependency between x and y. Does it go from x to y (represented as $x \xrightarrow{f} y$), or from y to x (represented as $y \xrightarrow{f} x$)?

If we refer to Fig. D.5.(a), we can make a decision because y can only be expressed as a function of x. In contrast, if we examine Fig. D.5.(b), the decision is also straightforward because x can solely be written as a function of y. However, if the data mirrors the scenario in Fig. D.5.(c), it becomes challenging to decide as we can write both y as a function of x and x as a function of y. Further complicating matters is the possibility of implicit dependencies between variables. As illustrated in Fig. D.5.(d), there might be instances where neither y can be derived as a function of x, nor x can be represented as a function of y.

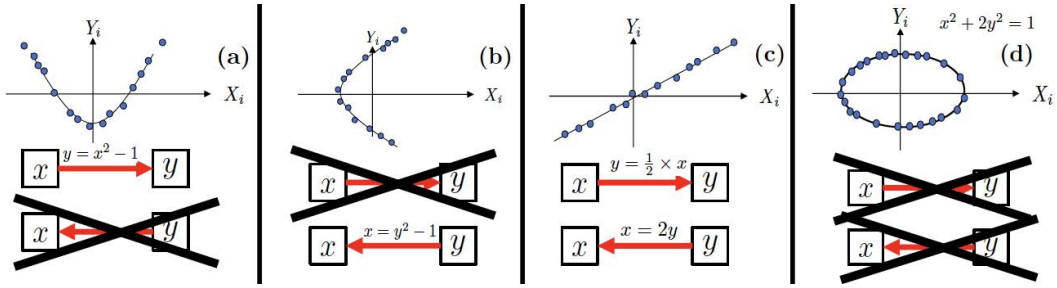


Figure D.5: The structure of the hypergraph is identifiable in (a), (b), and non-identifiable in (c). The relationship between variables is implicit in (d).

D.4.3 Causal inference and probabilistic graphs.

Causal inference methods broadly consist of two approaches: constraint and score-based methods. While constraint-based approaches are asymptotically consistent, they only learn the graph up to an equivalence class [318]. Instead, score-based methods resolve ambiguities in the graph's edges by evaluating the likelihood of the observed data for each graphical model. For instance, they may assign a higher evidence to $y \to x$ over $x \to y$ if the conditional distribution $x|y$ exhibits less complexity than $y|x$. The complexity of searching over all possible graphs, however,

grows super-exponentially with the number of variables. Thus, it is often necessary to use approximate, but more tractable, search-based methods [63, 259] or alternative criteria based on sensitivity analysis [75]. For example, the preference could lean towards $y \to x$ rather than $x \to y$ if y demonstrates less sensitivity to errors or perturbations in x. In contrast, our proposed GP method avoids the growth in complexity by performing a guided pruning process that assesses the contribution of each node to the signal. We also emphasize that our method is not limited to learning acyclic graph structures as it can identify feedback loops between variables. Alternatively, methods for learning probabilistic undirected graphical models, also known as Markov networks, identify the graph structure by assuming the data is randomly drawn from some probability distribution [90]. In this case, edges in the graph (or lack thereof) encode conditional dependencies between the nodes. A common approach learns the graph structure by modeling the data as being drawn from a multivariate Gaussian distribution with a sparse inverse covariance matrix, whose zero entries indicate pairwise conditional independencies [107]. Recently, this approach has been extended using models for non-Gaussian distributions, e.g., in [21, 275], as well as kernel-based conditional independence tests [371]. In this work, we learn functional dependencies rather than causality or probabilistic dependence. We emphasize that we also do not assume the data is randomized or impose strong assumptions, such as additive noise models, in the data-generating process.

We complete this paragraph by comparing the hypergraph discovery framework to structure learning for Bayesian networks and structural equation models (SEM). Let $x \in \mathbb{R}^d$ be a random variable with probability density function p that follows the autoregressive factorization $p(x) = \prod_{i=1}^{d} p_i(x_i|x_1, \ldots, x_{i-1})$ given a prescribed variable ordering. Structure learning for Bayesian networks aims to find the ancestors of variable x_i, often referred to as the set of parents $Pa(i) \subseteq \{1, \ldots, i-1\}$, in the sense that $p_i(x_i|x_1, \ldots, x_{i-1}) = p_i(x_i|x_{Pa(i)})$. Thus, the variable dependence of the conditional density p_i is identified by finding the parent set so that x_i is conditionally independent of all remaining preceding variables given its parents, i.e., $x_i \perp x_{1:i-1\setminus Pa(i)}|x_{Pa(i)}$. Finding ancestors that satisfy this condition requires performing conditional independence tests, which are computationally expensive for general distributions [304]. Alternatively, SEMs assume that each variable x_i is drawn as a function of its ancestors with additive noise, i.e, $x_i = f(x_{Pa(i)}) + \epsilon_i$ for some function f and noise ϵ [259]. For Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, each marginal conditional distribution in a Bayesian network is

given by $p_i(x_i|x_{1:i-1}) \propto \exp(-\frac{1}{2\sigma^2}\|x_i - f(x_{1:i-1})\|^2)$. Thus, finding the parents for such a model by maximum likelihood estimation corresponds to finding the parents that minimize the expected mean-squared error $\|x_i - f(x_{Pa(i)})\|^2$. Our approach minimizes a related objective, without imposing the strong probabilistic assumptions that are required in SEMs and Bayesian Networks. We also observe that while the graph structure identified in Bayesian networks is influenced by the specific sequence in which variables are arranged (a concept exploited in numerical linear algebra [292, 293] where Schur complementation is equivalent to conditioning GPs and a carefully ordering leads to the accuracy of the Vecchia approximation $p_i(x_i|x_1, \ldots, x_{i-1}) \approx p_i(x_i|x_{i-k}, \ldots, x_{i-1})$ [339]), the graph recovered by our approach remains unaffected by any predetermined ordering of those variables.

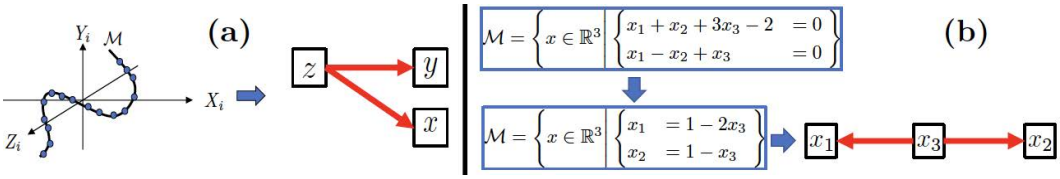


Figure D.6: (a) CHD formulation as a manifold discovery problem and hypergraph representation (b) The hypergraph representation of an affine manifold is equivalent to its Row Echelon Form Reduction.

D.4.4 Well-posed formulation of the problem.

In this paper, we focus on a formulation of the problem that remains well-posed even when the data is not randomized, i.e., we formulate the problem as the following manifold learning/discovery problem.

Problem 1. *Let $\mathcal{H}$ be a Reproducing Kernel Hilbert Space (RKHS) of functions mapping $\mathbb{R}^d$ to $\mathbb{R}$. Let $\mathcal{F}$ be a closed linear subspace of $\mathcal{H}$ and let $\mathcal{M}$ be a subset of $\mathbb{R}^d$ such that $x \in \mathcal{M}$ if and only if $f(x) = 0$ for all $f \in \mathcal{F}$. Given the (possibly noisy and nonrandom) observation of N elements, $X_1, \ldots, X_N$, of $\mathcal{M}$ approximate $\mathcal{M}$.*

To understand why problem 1 serves as the appropriate formulation for hypergraph discovery, consider a manifold $\mathcal{M} \subset \mathbb{R}^d$. Suppose this manifold can be represented by a set of equations, expressed as a collection of functions $(f_k)_k$ satisfying $\forall x \in \mathcal{M},\ f_k(x) = 0$. To keep the problem tractable, we assume a certain level of regularity for these functions, necessitating they belong to a RKHS $\mathcal{H}$, ensuring the applicability of kernel methods for our framework. Given that any linear combination of the f_k will also be evaluated to zero on $\mathcal{M}$, the relevant functions are those within the span of the f_k, forming a closed linear subspace of $\mathcal{H}$ denoted as

$\mathcal{F}$. The manifold $\mathcal{M}$ can be subsequently represented by a graph or hypergraph (see Fig. D.6.(a)), whose ambiguity can be resolved through a deliberate decision to classify some variables as free and others as dependent. This selection could be arbitrary, informed by expert knowledge, or derived from probabilistic models or sensitivity analysis.

D.5 A Gaussian Process method for Type 3 problems

D.5.1 Affine case and Row Echelon Form Reduction.

To describe the proposed solution to Problem 1, we start with a simple example. In this example $\mathcal{H}$ is a space of affine functions f of the form

$$f(x) = v^T \psi(x) \text{ with } \psi(x) := \begin{pmatrix} 1 \\ x \end{pmatrix} \text{ and } v \in \mathbb{R}^{d+1}, . \tag{D.7}$$

As a particular instantiation (see Fig. D.6.(b)), we assume $\mathcal{M}$ to be the manifold of $\mathbb{R}^3$ ($d = 3$) defined by the affine equations

$$\mathcal{M} = \left\{ x \in \mathbb{R}^3 \middle| \begin{cases} x_1 + x_2 + 3x_3 - 2 & = 0 \\ x_1 - x_2 + x_3 & = 0 \end{cases} \right\}, \tag{D.8}$$

which is equivalent to selecting $\mathcal{F} = \text{Span}\{f_1, f_2\}$ with $f_1(x) = x_1 + x_2 + 3x_3 - 2$ and $f_2(x) = x_1 - x_2 + x_3$ in the problem formulation 1.

Then, irrespective of how we recover the manifold from data, the hypergraph representation of that manifold is equivalent to the row echelon form reduction of the affine system, and this representation and this reduction require a possibly arbitrary choice of free and dependent variables. So, for instance, for the system (D.8), if we declare x_3 to be the free variables and x_1 and x_2 to be the dependent variables, then we can represent the manifold via the equations

$$\mathcal{M} = \left\{ x \in \mathbb{R}^3 \middle| \begin{cases} x_1 & = 1 - 2x_3 \\ x_2 & = 1 - x_3 \end{cases} \right\}, \tag{D.9}$$

which have the hypergraph representation depicted in Fig. D.6.(b).

Now, in the $N > d$ regime where the number of data points is larger than the number of variables, the manifold can simply be approximated via a variant of PCA. Take $f^* \in \mathcal{F}$, we have $f^*(x) = v^{*T} \psi(x)$ for a certain $v^* \in \mathbb{R}^{d+1}$. Then for $X_s \in \mathcal{M}$, $f^*(X_s) = \psi(X_s)^T v^* = 0$. Defining

$$C_N := \sum_{s=1}^{N} \psi(X_s) \psi(X_s)^T \tag{D.10}$$

we see that $f^*(X_s) = 0$ for all X_s is equivalent to $C_N v^* = 0$. Since $N > d$, we can thus identify $\mathcal{F}$ exactly as $\{v^T\psi \text{ for } v \in Ker(C_N)\}$. We then obtain the manifold

$$\mathcal{M}_N = \left\{x \in \mathbb{R}^d \mid v^T\psi(x) = 0 \text{ for } v \in \text{Span}(v_{r+1}, \ldots, v_{d+1})\right\} \tag{D.11}$$

where $\text{Span}(v_{r+1}, \ldots, v_{d+1})$ is the zero-eigenspace of C_N. Here we write $\lambda_1 \geq \cdots \geq \lambda_r > 0 = \lambda_{r+1} = \cdots = \lambda_{d+1}$ for the eigenvalues of C_N (in decreasing order), and $v_1, \ldots, v_{d+1}$ for the corresponding eigenvectors ($C_N v_i = \lambda_i v_i$). The proposed approach extends to the noisy case (when the data points are perturbations of elements of the manifold) by simply replacing the zero-eigenspace of the covariance matrix by the linear span of the eigenvectors associated with eigenvalues that are smaller than some threshold $\epsilon > 0$, i.e., by approximating $\mathcal{M}$ with (D.11) where r is such that $\lambda_1 \geq \cdots \geq \lambda_r \geq \epsilon > \lambda_{r+1} \geq \cdots \geq \lambda_{d+1}$. In this affine setting (D.11) allows us to estimate $\mathcal{M}$ directly without RKHS norm minimization/regularization because linear regression does not require regularization in the sufficiently large data regime. Furthermore the process of pruning ancestors can be replaced by that of identifying sparse elements $v \in \text{Span}(v_{r+1}, \ldots, v_{d+1})$ such that $v_i = 1$.

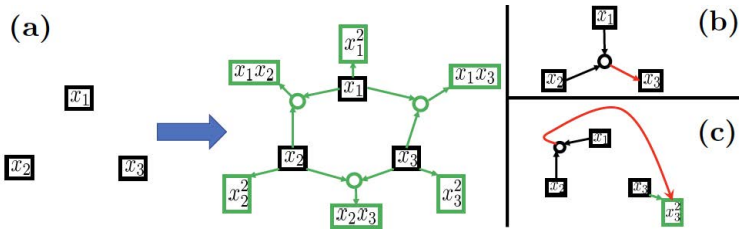


Figure D.7: Feature map generalization

D.5.2 Feature map generalization.

This simple approach can be generalized by generalizing the underlying feature map ψ used to define the space of functions (writing $d_{\mathcal{S}}$ for the dimension of the range of ψ)

$$\mathcal{H} = \left\{f(x) = v^T\psi(x) \mid v \in \mathbb{R}^{d_{\mathcal{S}}}\right\}. \tag{D.12}$$

For instance, if we use the feature map

$$\psi(x) := \left(1, \ldots, x_i, \ldots, x_i x_j, \ldots\right)^T \tag{D.13}$$

then $\mathcal{H}$ becomes a space of quadratic polynomials on $\mathbb{R}^d$, i.e.,

$$\mathcal{H} = \left\{f(x) = v_0 + \sum_i v_i x_i + \sum_{i \leq j} v_{i,j} x_i x_j \mid v \in \mathbb{R}^{d_{\mathcal{S}}}\right\}, \tag{D.14}$$

and, in the large data regime ($N > d_{\mathcal{S}}$), identifying quadratic dependencies between variables becomes equivalent to (1) adding nodes to the hypergraph corresponding to secondary variables obtained from primary variables x_i through known functions (for (D.13), these secondary variables are the quadratic monomials $x_i x_j$, see Fig. D.7.(a)), and (2) identifying affine dependencies between the variables of the augmented hypergraph. The problem can, therefore, be reduced to the previous affine case. Indeed, as in the affine case, the manifold can then be approximated in the regime where the number of data points is larger than the dimension $d_{\mathcal{S}}$ of the feature map by (D.11), where $v_r, \ldots, v_N$ are the eigenvectors of C_N =(D.10) whose eigenvalues are zero (noiseless case) or smaller than some threshold $\epsilon > 0$ (noisy case).

Furthermore, the hypergraph representation of the manifold is equivalent to a feature map generalization of Row Echelon Form Reduction to nonlinear systems of equations. For instance, choosing x_3 as the dependent variable and x_1, x_2 as the free variables, $\mathcal{M} = \{x \in \mathbb{R}^3 \mid x_3 - 5x_1^2 + x_2^2 - x_1 x_2 = 0\}$ can be represented as in Fig. D.7.(b) where the round node represents the concatenated variable (x_1, x_2) and the red arrow represents a quadratic function. The generalization also enables the representation of implicit equations by selecting secondary variables as free variables. For instance, selecting x_3^2 as the free variable and x_1, x_2 as the free variables, $\mathcal{M} = \{x \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 - 1 = 0\}$ can be represented as in Fig. D.7.(c).

D.5.3 Kernel generalization and regularization.

This feature-map extension of the previously discussed affine case can evidently be generalized to arbitrary degree polynomials and to other basis functions. However, as the dimension $d_{\mathcal{S}}$ of the range of the feature map ψ increases beyond the number N of data points, the problem becomes underdetermined: the data only provides partial information about the manifold, i.e., it is not sufficient to uniquely determine the manifold. Furthermore, if the dimension of the feature map is infinite, then we are always in that low data regime, and we have the additional difficulty that we cannot directly compute with that feature map. On the other hand, if $d_{\mathcal{S}}$ is finite (i.e., if the dictionary of basis functions is finite), then some elements of $\mathcal{F}$ (some constraints defining the manifold $\mathcal{M}$) may not be representable or well approximated as equations of the form $v^T \psi(x) = 0$. To address these conflicting requirements, we need to kernelize and regularize the proposed approach (as done in interpolation).

D.5.3.1 The kernel associated with the feature map.

To describe this kernelization, we assume that the feature map ψ maps $\mathbb{R}^d$ to some Hilbert space $\mathcal{S}$ that could be infinite-dimensional, and we write K for the kernel defined by that feature map. To be precise, we now consider the setting where the feature map ψ is a function from $\mathbb{R}^d$ to a (possibly infinite-dimensional separable) Hilbert (feature) space $\mathcal{S}$ endowed with the inner product $\langle \cdot, \cdot \rangle_{\mathcal{S}}$. To simplify notations, we will still write $v^T w$ for $\langle v, w \rangle_{\mathcal{S}}$ and vw^T for the linear operator mapping v' to $v\langle w, v' \rangle_{\mathcal{S}}$. Let

$$\mathcal{H} := \{v^T \psi(x) \mid v \in \mathcal{S}\} \tag{D.15}$$

be the space of functions mapping $\mathbb{R}^d$ to $\mathbb{R}$ defined by the feature map ψ. To avoid ambiguity, assume (without loss of generality) that the identity $v^T\psi(x) = w^T\psi(x)$ holds for all $x \in \mathbb{R}^d$ if and only if $v = w$. It follows that for $f \in \mathcal{H}$ there exists a unique $v \in \mathcal{S}$ such that $f = v^T\psi$. For $f, g \in \mathcal{H}$ with $f = v^T\psi$ and $g = w^T\psi$, we can then define

$$\langle f, g \rangle_{\mathcal{H}} := v^T w \,. \tag{D.16}$$

Observe that $\mathcal{H}$ is a Hilbert space endowed with the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. For $x, x' \in \mathcal{X}$, write

$$K(x, x') := \psi(x)^T \psi(x') \,, \tag{D.17}$$

for the kernel defined by ψ and observe that $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ is the RKHS defined by the kernel K (which is assumed to contain $\mathcal{F}$ in Problem 1). Observe in particular that for $f = v^T\psi \in \mathcal{H}$, K satisfies the reproducing property

$$\langle f, K(x, \cdot) \rangle_{\mathcal{H}} = v^T\psi(x) = f(x) \,. \tag{D.18}$$

D.5.3.2 Complexity Reduction with Kernel PCA Variant.

We will now show that the previous feature-map PCA variant (characterizing the subspace of $f \in \mathcal{H}$ such that $f(X) = 0$) can be kernelized as a variant of kernel PCA [216]. To describe this write $K(X, X)$ for the $N \times N$ matrix with entries $K(X_i, X_j)$. Write $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0$ for the nonzero eigenvalues of $K(X, X)$ indexed in decreasing order and write $\alpha_{\cdot,i}$ for the corresponding unit-normalized eigenvectors, i.e.

$$K(X, X)\alpha_{\cdot,i} = \lambda_i \alpha_{\cdot,i} \text{ and } |\alpha_{\cdot,i}| = 1 \,. \tag{D.19}$$

Write $f(X)$ for the N vector with entries $f(X_s)$. For $i \leq r$, write

$$\phi_i := \sum_{s=1}^{N} \delta_{X_s} \alpha_{s,i} \tag{D.20}$$

and

$$f(\phi_i) := \sum_{s=1}^{N} f(X_s)\alpha_{s,i} \,. \tag{D.21}$$

Write $f(\phi)$ for the r vector with entries $f(\phi_i)$.

Then, we have the following proposition.

Proposition D.5.1. *The subspace of functions $f \in \mathcal{H}$ such that $f(\phi) = 0$ is equal to the subspace of $f \in \mathcal{H}$ such that $f(X) = 0$. Furthermore for $f \in \mathcal{H}$ with feature map representation $f = v^T\psi$ with $v \in \mathcal{S}$ we have the identity (where C_N =(D.10))*

$$v^T C_N v = |f(\phi)|^2 = |f(X)|^2 \,. \tag{D.22}$$

Proof. Write $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_{\hat{r}} > 0$ for the nonzero eigenvalues of C_N =(D.10) indexed in decreasing order. Write $v_1, \ldots, v_r$ for the corresponding eigenvectors, i.e.,

$$C_N v_i = \hat{\lambda}_i v_i \,. \tag{D.23}$$

Observing that

$$C_N = \sum_{i=1}^{r} \hat{\lambda}_i v_i v_i^T \tag{D.24}$$

we deduce that the zero-eigenspace of C_N is the set of vectors $v \in \mathcal{S}$ such that $v^T v_i = 0$ for $i = 1, \ldots, r$. Write $f_i := v_i^T \psi$. Observe that for $f = v^T\psi$, we have $v_i^T v = \left\langle f_i, f \right\rangle_K$. Multiplying (D.23) by $\psi^T(x)$ implies

$$\sum_{s=1}^{N} K(x, X_s) f_i(X_s) = \hat{\lambda}_i f_i(x) \tag{D.25}$$

(D.25) implies that for $f = v^T\psi$

$$v_i^T v = \sum_{s=1}^{N} \hat{\lambda}_i^{-1} f_i(X_s) \left\langle K(\cdot, X_s), f \right\rangle_K = \sum_{s=1}^{N} \hat{\lambda}_i^{-1} f_i(X_s) f(X_s) \tag{D.26}$$

where we have used the reproducing property (D.18) of K in the last identity. Write

$$\hat{\alpha}_{s,i} := \lambda_i^{-1/2} f_i(X_s) \,. \tag{D.27}$$

Using (D.25) with $x = X_{s'}$ implies that $\hat{\alpha}_{\cdot,i}$ is an eigenvector of the $N \times N$ matrix $K(X, X)$ with eigenvalue $\hat{\lambda}_i$. Taking $f = f_i$ in (D.26) implies that $1 = v_i^T v_i = |\hat{\alpha}_{\cdot,i}|^2$. Therefore, the $\hat{\alpha}_{\cdot,i}$ are unit-normalized. Summarizing, this analysis (closely related to the one found in kernel PCA [216]) shows that the nonzero eigenvalues of $K(X, X)$

coincide with those of C_N and we have $\hat{r} = r$, $\hat{\lambda}_i = \lambda_i$ and $\hat{\alpha}_{\cdot,i} = \alpha_{\cdot,i}$. Furthermore, (D.26) and (D.27) imply that for $i \leq r$, $v \in \mathcal{S}$ and $f = v^T\psi$, we have

$$v_i^T v = \lambda_i^{-1/2} f(X)\alpha_{\cdot,i} \,. \tag{D.28}$$

The identity (D.28) then implies (D.22). □

Remark D.5.2. *As in PCA the dimension/complexity of the problem can be further reduced by truncating ϕ to $\phi' = (\phi_1, \ldots, \phi_{r'})$ where $r' \leq r$ is identified as the smallest index i such that $\lambda_i/\lambda_1 < \epsilon$ where $\epsilon > 0$ is some small threshold.*

D.5.3.3 Kernel Mode Decomposition.

When the feature map ψ is infinite-dimensional, the data only provides partial information about the constraints defining the manifold in the sense that $f(X) = 0$ or equivalently $f(\phi) = 0$ is a necessary but not sufficient condition for the zero level set of f to be a valid constraint for the manifold (for f to be such that $f(x) = 0$ for all $x \in \mathcal{M}$). So we are faced with the following problems: (1) How to regularize? (2) How do we identify free and dependent variables? (3) How do we identify valid constraints for the manifold? The proposed solution will be based on the Kernel Mode Decomposition (KMD) framework introduced in [243] (which shares conceptual foundations with Smoothing Spline ANOVA [345]).

Reminder on KMD We will now present a quick reminder on KMD in the setting of the following mode decomposition problem. So, in this problem, we have an unknown function $f^\dagger$ mapping some input space $\mathcal{X}$ to the real line $\mathbb{R}$. We assume that this function can be written as a sum of m other unknown functions $f_i^\dagger$ which we will call modes, i.e.,

$$f^\dagger = \sum_{i=1}^{m} f_i^\dagger \,. \tag{D.29}$$

We assume each mode $f_i^\dagger$ to be an unknown element of some RKHS $\mathcal{H}_{K_i}$ defined by some kernel K_i. Then we consider the problem in which given the data $f^\dagger(X) = Y$ (with $(X, Y) \in \mathcal{X}^N \times \mathbb{R}^N$) we seek to approximate the m modes composing the target function $f^\dagger$. Then, we have the following theorem.

Theorem D.5.3. *[243] Using the relative error in the product norm $\|(f_1, \ldots, f_m)\|^2 := \sum_{i=1}^m \|f_i\|_{K_i}^2$ as a loss, the minimax optimal recovery of $(f_1^\dagger, \ldots, f_m^\dagger)$ is $(f_1, \ldots, f_m)$ with*

$$f_i(x) = K_i(x, X)K(X, X)^{-1}Y \,, \,, \tag{D.30}$$

where K is the additive kernel

$$K = \sum_{i=1}^{m} K_i \,. \tag{D.31}$$

The GP interpretation of this optimal recovery result is as follows. Let $\xi_i \sim \mathcal{N}(0, K_i)$ be m independent centered GPs with kernels K_i. Write ξ for the additive GP $\xi := \sum_{i=1}^{m} \xi_i$. (D.30) can be recovered by replacing the modes $f_i^\dagger$ by independent centered GPs $\xi_i \sim \mathcal{N}(0, K_i)$ with kernels K_i and approximating the mode i by conditioning ξ_i on the available data $\xi(X) = Y$ where $\xi := \sum_{i=1}^{m} \xi_i$ is the additive GP obtained by summing the independent GPs ξ_i, i.e.,

$$f_i(x) = \mathbb{E}\big[\xi_i(x) \mid \xi(X) = Y\big] \,. \tag{D.32}$$

Furthermore $(f_1, \ldots, f_m)$ can also be identified as the minimizer of

$$\begin{cases} \text{Minimize} & \sum_{i=1}^{m} \|f_i\|_{K_i}^2 \\ \text{over} & (f_1, \ldots, f_m) \in \mathcal{H}_{K_1} \times \cdots \times \mathcal{H}_{K_m} \\ \text{s. t.} & (\sum_{i=1}^{m} f_i)(X) = Y \,. \end{cases} \tag{D.33}$$

The variational formulation (D.33) can be interpreted as a generalization of Tikhonov regularization which can be recovered by selecting $m = 2$, K_1 to be a smoothing kernel (such as a Matérn kernel) and $K_2(x, y) = \sigma^2 \delta(x - y)$ to be a white noise kernel.

Now, this abstract KMD approach [243] is associated with a quantification of how much each mode contributes to the overall data or how much each individual GP ξ_i explains the data. More precisely, the activation of the mode i or GP ξ_i can be quantified as

$$p(i) = \frac{\|f_i\|_{K_i}^2}{\|f\|_K^2} \,, \tag{D.34}$$

where $f = \sum_{i=1}^{m} f_i$. These activations $p(i)$ satisfy $p(i) \in [0, 1]$ and $\sum_{i=1}^{m} p(i) = 1$ they can be thought of as a generalization of Sobol sensitivity indices [235, 313, 314] to the nonlinear setting in the sense that they are associated with the following variance representation/decomposition [243] (writing $\langle \cdot, \cdot \rangle_K$ for the RKHS inner product induced by K):

$$\mathrm{Var}\left[\langle \xi, f \rangle_K\right] = \|f\|_K^2 = \sum_{i=1}^{m} \|f_i\|_{K_i}^2 = \sum_{i=1}^{m} \mathrm{Var}\left[\langle \xi_i, f \rangle_K\right] \tag{D.35}$$

Application to CHD, general case. Now, let us return to our original manifold approximation problem 1 in the kernelized setting of (D.17). Given the data X we cannot regress an element $f \in \mathcal{F}$ directly since the minimizer of $\|f\|_K^2 + \gamma^{-1}\|f(X)\|_{\mathbb{R}^N}^2$ is the null function. To identify the functions $f \in \mathcal{F}$, we need to decompose them into modes that can be interpreted as a generalization of the notion of free and dependent variables. To describe this, assume that the kernel K can be decomposed as the additive kernel

$$K = K_a + K_s + K_z. \tag{D.36}$$

Then $\mathcal{H}_K = \mathcal{H}_{K_a} + \mathcal{H}_{K_s} + \mathcal{H}_{K_z}$ implies that for all function $f \in \mathcal{H}_K$, f can be decomposed as $f = f_a + f_s + f_z$ with $(f_a, f_s, f_z) \in \mathcal{H}_a \times \mathcal{H}_s \times \mathcal{H}_z$.

Example D.5.4. *As a running example, take K to be the following additive kernel*

$$K(x, x') = 1 + \beta_1 \sum_i x_i x_i' + \beta_2 \sum_{i \le j} x_i x_j x_i' x_j' + \beta_3 \prod_i (1 + k(x_i, x_i')), \tag{D.37}$$

that is the sum of a linear kernel, a quadratic kernel, and a fully nonlinear kernel. Take K_a to be the part of the linear kernel that depends only on x_1, i.e.,

$$K_a(x, x') = \beta_1 x_1 x_1' . \tag{D.38}$$

Take K_s to be the part of the kernel that does not depend on x_1, i.e.,

$$K_s = 1 + \beta_1 \sum_{i \neq 1} x_i x_i' + \beta_2 \sum_{i \le j, i, j \neq 1} x_i x_j x_i' x_j' + \beta_3 \prod_{i \neq 1} (1 + k(x_i, x_i')) . \tag{D.39}$$

And take K_z to be the remaining portion,

$$K_z = K - K_a - K_s . \tag{D.40}$$

Therefore the following questions are equivalent:

- Given a function g_a in the RKHS $\mathcal{H}_{K_a}$ defined by the kernel K_a is there a function f_s in the RKHS $\mathcal{H}_{K_s}$ defined by the kernel K_s such that $g_a(x) \approx f_s(x)$ for $x \in \mathcal{M}$?
- Given a function $g_a \in \mathcal{H}_{K_a}$ is there a function f in the RKHS $\mathcal{H}_K$ defined by the kernel K such that $f(x) \approx 0$ for $x \in \mathcal{M}$ and such that its f_a mode is $-g_a$ and its f_z mode is zero?

Then, the natural answer to the questions is to identify the modes of the constraint $f = f_a + f_s + f_z \in \mathcal{H}$ (such that $f(x) \approx 0$ for $x \in \mathcal{M}$) such that $f_a = -g_a$ and $f_z = 0$ by selecting f_s to be the minimizer of the following variational problem

$$\min_{f_s \in \mathcal{H}_s} \|f_s\|^2_{K_s} + \frac{1}{\gamma}\left|(-g_a + f_s)(\phi)\right|^2 . \tag{D.41}$$

This is equivalent to introducing the additive GP $\xi = \xi_a + \xi_s + \xi_z + \xi_n$ whose modes are the independent GPs $\xi_a \sim \mathcal{N}(0, K_a)$, $\xi_s \sim \mathcal{N}(0, K_s)$, $\xi_z \sim \mathcal{N}(0, K_z)$, $\xi_n \sim \mathcal{N}(0, \gamma\delta(x - y))$ (we use the label "n" in reference to "noise"), and then recovering f_s as

$$f_s = \mathbb{E}\left[\xi_s \mid \xi(X) = 0, \xi_a = -g_a, \xi_z = 0\right] . \tag{D.42}$$

Application to CHD, particular case. Taking $g_a(x) = x_1$ for our running example D.5.4, the previous questions are, as illustrated in Fig. 5.2(b), equivalent to asking whether there exists a function $f_s \in \mathcal{H}_{K_s}$ that does not depend on x_1 (since K_s does not depend on x_1) such that

$$x_1 \approx f_s(x_2, \ldots, x_d) \text{ for } x \in \mathcal{M} . \tag{D.43}$$

Therefore, the mode f_a can be thought of as a dependent mode (we use the label "a" in reference to "ancestors"), the mode f_s as a free mode (we use the label "s" in reference to "signal"), the mode f_z as a zero mode.

While our numerical illustrations have primarily focused on the scenario where g_a takes the form of $g_a(x) = x_i$, and we aim to express x_i as a function of other variables, the generality of our framework is motivated by its potential to recover implicit equations. For example, consider the implicit equation $x_1^2 + x_2^2 = 1$, which can be retrieved by setting the mode of interest to be $g_a(x) = x_1^2$ and allowing f_s to depend only on the variable x_2.

D.5.3.4 Signal-to-noise ratio.

Now, we are led to the following question: since the mode f_s (the minimizer of (D.41)) always exists and is always unique, how do we know that it leads to a valid constraint? To answer that question, we compute the activation of the GPs used to regress the data. We write

$$\mathcal{V}(s) := \|f_s\|^2_{K_s} , \tag{D.44}$$

for the activation of the signal GP ξ_s and

$$\mathcal{V}(n) := \frac{1}{\gamma}\left|(-g_a + f_s)(X)\right|^2 \tag{D.45}$$

for the activation of the noise GP ξ_n, and then these allow us to define a signal-to-noise ratio defined as

$$\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)}\,. \tag{D.46}$$

Note that this corresponds to activation ratio of the noise GP defined in (D.34). This ratio can then be used to test the validity of the constraint in the sense that if $V(s)/(V(s)+V(n)) > \tau$ (with $\tau = 0.5$ as a prototypical example), then the data is mostly explained by the signal GP and the constraint is valid. If $V(s)/(V(s)+V(n)) < \tau$, then the data is mostly explained by the noise GP and the constraint is not valid.

D.5.3.5 Iterating by removing the least active modes from the signal.

If the constraint is valid, then we can next compute the activation of the modes composing the signal. To describe this, we assume that the kernel K_s can be decomposed as the additive kernel

$$K_s = K_{s,1} + \cdots + K_{s,m}\,, \tag{D.47}$$

which results in $\mathcal{H}_{K_s} = \mathcal{H}_{K_{s,1}} + \cdots + \mathcal{H}_{K_{s,m}}$, which results in the fact that $\forall f_s \in \mathcal{H}_s$, f_s can be decomposed as

$$f_s = f_{s,1} + \cdots + f_{s,m}\,, \tag{D.48}$$

with $f_{s,i} \in \mathcal{H}_{K_{s,i}}$. The activation of the mode i can then be quantified as $p(i) = \|f_{s,i}\|^2_{K_{s,i}}/\|f_s\|^2_{K_s}$, which combined with $\|f_s\|^2_{K_s} = \sum_{i=1}^m \|f_{s,i}\|^2_{K_{s,i}}$ leads to $\sum_{i=1}^m p(i) = 1$.

As our running example D.5.4, we can decompose K_s =(D.39) as the sum of an affine kernel, a quadratic kernel, and a fully nonlinear kernel, i.e., $m = 3$, $K_{s,1} = 1 + \beta_1 \sum_{i\neq 1} x_i x'_i$, $K_{s,2} = \beta_2 \sum_{i\leq j, i,j\neq 1} x_i x_j x'_i x'_j$ and $K_{s,3} = \beta_3 \prod_{i\neq 1}(1 + k(x_i, x'_i))$.

As another example for our running example, we can take K_s to be the sum of the portion of the kernel that does not depend on x_1 and x_2 and the remaining portion, i.e., $m = 2$, $K_{s,1} = 1+\beta_1 \sum_{i\neq 1,2} x_i x'_i + \beta_2 \sum_{i\leq j, i,j\neq 1,2} x_i x_j x'_i x'_j + \beta_3 \prod_{i\neq 1,2}(1+k(x_i, x'_i))$ and $K_{s,2} = K_s - K_{s,1}$.

Then, we can order these sub-modes from most active to least active and create a new kernel K_s by removing the least active modes from the signal and adding them to the mode that is set to be zero (see Fig. D.8). To describe this, let $\pi(1), \cdots, \pi(m)$ be an ordering of the modes by their activation, i.e., $\|f_{s,\pi(1)}\|^2_{K_{s,\pi(1)}} \geq \|f_{s,\pi(2)}\|^2_{K_{s,\pi(2)}} \geq \cdots$.

Writing $K_t = \sum_{i=r+1}^m K_{s,\pi(i)}$ for the additive kernel obtained from the least active modes (with $r + 1 = m$ as the value used for our numerical implementations), we

update the kernels K_s and K_z by assigning the least active modes from K_s to K_z, i.e., $K_s - K_t \rightarrow K_s$ and $K_z + K_t \rightarrow K_z$ (we zero the least active modes).

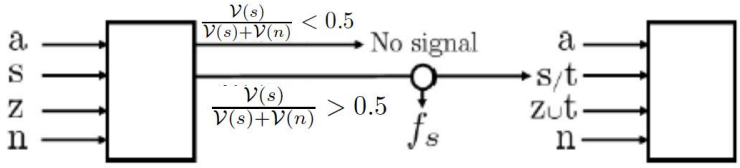


Figure D.8: Iterating by removing the least active modes from the signal

Finally, we can iterate the process. This iteration can be thought of as identifying the structure of the hypergraph by placing too many hyperedges and removing them according to the activation of the underlying GPs.

For our running example D.5.4, where we try to identify the ancestors of the variable x_1, if the sub-mode associated with the variable x_2 is found to be least active, then we can try to remove x_2 from the list of ancestors and try to identify x_1 as a function of x_3 to x_d. This is equivalent to selecting $K_a(x, x') = \beta_1 x_1 x_1'$,

$$K_{s/t} = 1 + \beta_1 \sum_{i \neq 1,2} x_i x_i' + \beta_2 \sum_{i \leq j, i, j \neq 1,2} x_i x_j x_i' x_j' + \beta_3 \prod_{i \neq 1,2} (1 + k(x_i, x_i')) \,, \qquad \text{(D.49)}$$

and $K_{z \cup t} = K - K_a - K_{s/t}$ to assess whether there exists a function $f_s \in \mathcal{H}_K$ that does not depend on x_1 and x_2 s.t. $x_1 \approx f_s(x_3, \ldots, x_d)$ for $x \in \mathcal{M}$.

D.5.3.6 Alternative determination of the list of ancestors.

Our initial approach to determining the list of ancestors of a given node is to use a fixed threshold (e.g., $\tau = 0.5$) to prune nodes. We propose a refined approach that mimics the strategy employed in Principal Component Analysis (PCA) for deciding which modes should be kept and which ones should be removed. The PCA approach is to order the modes in decreasing order of eigenvalues/variance and (1) either keep the smallest number modes holding/explaining a given fraction (e.g., 90%) of the variance in the data, (2) or use an inflection point/sharp drop in the decay of the eigenvalues to select which modes should be kept. Here, we propose a similar strategy. First we employ an alternative determination of the least active mode: we iteratively remove the mode that leads to the smallest increase in noise-to-signal ratio, i.e., we remove the mode t such that,

$$t = \text{argmin}_t \, \frac{\mathcal{V}(n)}{\mathcal{V}(s/t) + \mathcal{V}(n)} \,. \qquad \text{(D.50)}$$

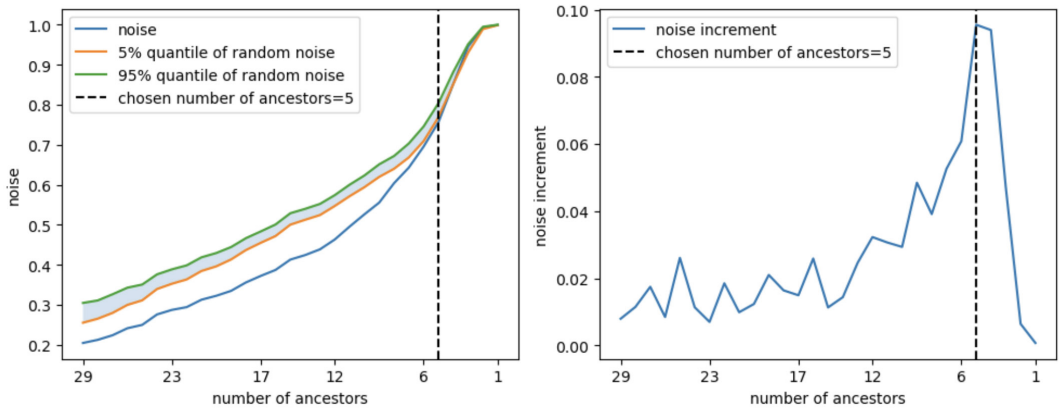


Figure D.9: Computing the ancestors of the variable $\dot{x}_0$ in the Fermi-Pasta-Ulam-Tsingou problem. (a) Noise-to-Signal Ratio, denoted as $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$, with respect to the number of proposed ancestors, represented by q. Additionally, we include a visualization of the quantiles derived from the Z-test, as described in Section D.7.3. Notably, when there is no signal present, the noise-to-signal ratio is expected to fall within the shaded area with a probability of 0.9. (b) Increments in the Noise-to-Signal Ratio, defined as $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q) - \frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q-1)$, as a function of the number of ancestors, denoted as q. The horizontal axis represents the number of proposed ancestors for $\dot{x}_0$. Determining an appropriate stopping point based solely on absolute noise-to-signal ratio levels can be challenging. In contrast, the increments in the noise-to-signal ratio clearly exhibit a discernible maximum, offering a practical point for decision-making.

For our running example D.5.4 in which we try to find the ancestors of the variable x_1 this is equivalent to removing the variables or node t whose removal leads to the smallest loss in signal-to-noise ratio (or increase in noise-to-signal ratio) by selecting

$$K_{s/t} = 1 + \beta_1 \sum_{i \neq 1,t} x_i x_i' + \beta_2 \sum_{i \leq j, i,j \neq 1,t} x_i x_j x_i' x_j' + \beta_3 \prod_{i \neq 1,t} (1 + k(x_i, x_i')) \,.$$

Next, we iterate this process, and we plot (a) the noise-to-signal ratio, and (b) the increase in noise-to-signal ratio as a function of the number of ancestors ordered according to this iteration. Fig. D.9 illustrates this process and shows that the removal of an essential node leads to a sharp spike in increase in the noise-to-signal ratio (the noise-to-signal ratio jumps from approximately 50-60% to 99%). The identification of this inflection point can be used as a method for effectively and reliably pruning ancestors.

Algorithm 7 CHD by thresholding the signal-to-noise ratio

Require: Data D, set of nodes V, threshold τ ($\tau = 0.5$ as a default value)
Ensure: Learned hypergraph ▷ Set of ancestors for each node

```
1:  D ← NormalizeData(D)                                   ▷ Normalize the data
2:  for v ∈ V do
3:      for kernel ∈ ["linear", "quadratic", "nonlinear"] do   ▷ Find the kernel
4:          SetOfAncestors(v) ← All other nodes
5:          SignalToNoiseRatio ← ComputeSignalToNoiseRatio(kernel, node, D)
6:          if SignalToNoiseRatio > τ then choose that kernel and exit the for loop
7:          else remove all ancestors from node
8:          end if
9:      end for
10:     while SignalToNoiseRatio > τ do                    ▷ Prune ancestors
11:         Find least important ancestor
12:         Recompute SignalToNoiseRatio without ancestor
13:         if SignalToNoiseRatio > τ then Remove that ancestor
14:         end if
15:     end while
16: end for
```

D.6 Algorithm pseudocode.

Our overall method is summarized in the pseudocode Alg. 7 and Alg. 8 that we will now describe. Alg. 7 takes the data D (encoded into the samples $X_1, \ldots, X_N$ of Problem 1) and the set of nodes V as an input and produces, as described in Sec. D.5.3, for each node $i \in V$ its set of minimal ancestors A_i and the simplest possible function f_i such that $x_i \approx f_i\big((x_j)_{j\in A_i}\big)$. It employs the default threshold of 0.5 on the signal-to-noise ratios for its operations. Line 1 normalizes the data (via an affine transformation) so that the samples X_i are of mean zero and variance 1. Given a node with index $i = 1$ in Line 2 (i runs through the set of nodes, and we select $i = 1$ for ease of presentation), the command in Line 3 refers to selecting a signal kernel of the form K_s =(D.39) (where k is selected to be a vanilla RBF kernel such as Gaussian or Matérn), with $1 \geq \beta_1 > 0 = \beta_2 = \beta_3$ for the linear kernel, $1 \geq \beta_1 \geq \beta_2 > 0 = \beta_3$ for the quadratic kernel and $1 \geq \beta_1 \geq \beta_2 \geq \beta_3 > 0$ for the fully nonlinear (interpolative) kernel. The ComputeSignalToNoiseRatio function in Line 5 computes the signal-to-noise ratio with $g_a(x) = x_1$ and with the kernel selected in Line 3. The value of γ is selected automatically by maximizing the variance of the histogram of eigenvalues of D_γ as described in Sec. D.7.2 (with the kernel $K = K_s$ =(D.39) selected in Line 3 and $Y = g_a(X)$ with $g_a(x) = x_1$). The value of γ is re-computed whenever a node is removed from the list of ancestors, and

K_s is nonlinear. Lines 11, 12 and 13 are described in Sec. D.5.3.5. They correspond to iteratively identifying the ancestor node t contributing the least to the signal and removing that node from the set of ancestors of the node 1 if the removal of that node t does not send the signal-to-noise ratio below the default threshold 0.5.

Algorithm 8 CHD by inflection point in the noise-to-signal ratio

Require: Data D, set of nodes V, threshold τ ($\tau = 0.5$ as a default value)
Ensure: Learned hypergraph ▷ Set of ancestors for each node
1: $D \leftarrow$ NormalizeData(D) ▷ Normalize the data
2: **for** node $v \in V$ **do**
3: **for** kernel ∈ [“linear”, “quadratic”, “nonlinear”] **do** ▷ Find the kernel
4: SetOfAncestors ← All other nodes
5: SignalToNoiseRatio ← ComputeSignalToNoiseRatio(kernel, node, D)
6: **if** SignalToNoiseRatio $> \tau$ **then** choose that kernel and exit the for loop
7: **else** remove all ancestors from node
8: **end if**
9: **end for**
10: q ← Cardinal(All other nodes)
11: SetOfAncestors(q) ← All other nodes
12: **while** q ≥ 1 **do**
13: NoiseToSignalRatio(q) ← ComputeNoiseToSignalRatio(kernel, node, D)
14: LeastImportantAncestor ← Find least important ancestor in SetOfAncestors(q)
15: SetOfAncestors($q-1$) ← SetOfAncestors(q) \ LeastImportantAncestor
16: $q \leftarrow q - 1$
17: **end while**
18: $q^\dagger \leftarrow$ Inflection point in ($q \rightarrow$ NoiseToSignalRatio(q)) or spike in ($q \rightarrow$ NoiseToSignalRatio(q) - NoiseToSignalRatio($q-1$))
19: FinalSetOfAncestors(v) ← SetOfAncestors($q^\dagger$)
20: **end for**

Algorithm 8 distinguishes itself from Algorithm 7 in its approach to pruning ancestors based on signal-to-noise ratios. Instead of using a default threshold of 0.5 like Algorithm 7, Algorithm 8 computes the noise-to-signal ratio, represented as $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$. This ratio is calculated as a function of the number q of ancestors, which are ordered based on their decreasing contribution to the signal. The detailed methodology behind this computation can be found in Section D.5.3.6 and is visually depicted in Figure D.9. The final number q of ancestors is then determined by finding the value that maximizes the difference between successive noise-to-signal ratios, $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q+1) - \frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}(q)$.

D.7 Analysis of the signal-to-noise ratio test.

D.7.1 The signal-to-noise ratio depends on the prior on the level of noise.

The signal-to-noise ratio (D.46) depends on the value of γ, which is the variance prior on the level of noise. The goal of this subsection is to answer the following two questions: (1) How do we select γ? (2) How do we obtain a confidence level for the presence of a signal? Or equivalently for a hyperedge of the hypergraph? To answer these questions, we will now analyze the signal-to-noise ratio in the following regression problem in which we seek to approximate the unknown function $f^\dagger : \mathcal{X} \to \mathbb{R}$ based on noisy observations

$$f^\dagger(X) + \sigma Z = Y \tag{D.51}$$

of its values at collocation points X_i ($(X, Y) \in \mathcal{X}^N \times \mathbb{R}^N$, $Z \in \mathbb{R}^N$, and the entries Z_i of Z are i.i.d $\mathcal{N}(0, 1)$). Assuming σ^2 to be unknown and writing γ for a candidate for its value, recall that the GP solution to this problem is approximate $f^\dagger$ by interpolating the data with the sum of two independent GPs, i.e.,

$$f(x) = \mathbb{E}[\xi(x)|\xi(X) + \sqrt{\gamma}Z = Y] \,, \tag{D.52}$$

where $\xi \sim \mathcal{N}(0, K)$ is the GP prior for the signal $f^\dagger$ and $\sqrt{\gamma}Z \sim \mathcal{N}(0, \gamma I_N)$ is the GP prior for the noise σZ in the measurements. Following Sec. D.5.3.3 f can also be identified as a minimizer of

$$\text{minimize}_{f'} \|f'\|_K^2 + \frac{1}{\gamma}\|f'(X) - Y\|_{\mathbb{R}^N}^2 \,, \tag{D.53}$$

the activation of the signal GP can be quantified as $s = \|f\|_K^2$, the activation of the noise GP can be quantified as $\mathcal{V}(n) = \frac{1}{\gamma}\|f(X) - Y\|_{\mathbb{R}^N}^2$. We can then define the noise to signal ratio $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)}$, which admits the following representer formula,

$$\frac{\mathcal{V}(n)}{\mathcal{V}(s) + \mathcal{V}(n)} = \gamma \frac{Y^T \big(K(X, X) + \gamma I\big)^{-2} Y}{Y^T \big(K(X, X) + \gamma I\big)^{-1} Y} \,. \tag{D.54}$$

Observe that when applied to the setting of Sec. D.5.3.4, this signal-to-noise ratio is calculated with $K = K_s$ and $Y = g_a(X)$.

Now we have the following proposition, which follows from (D.54).

Proposition D.7.1. *It holds true that* $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)} \in [0, 1]$*, and if* $K(X, X)$ *has full rank,*

$$\lim_{\gamma \downarrow 0} \frac{\mathcal{V}(n)}{\mathcal{V}(s) + \mathcal{V}(n)} = 0 \text{ and } \lim_{\gamma \uparrow \infty} \frac{\mathcal{V}(n)}{\mathcal{V}(s) + \mathcal{V}(n)} = 1 \,. \tag{D.55}$$

Therefore, we are led to the following question: if the signal $f^\dagger$ and the level of noise σ^2 are both unknown, how do we select γ to decide whether the data is mostly signal or noise?

D.7.2 How do we select the prior on the level of noise?

Our answer to this question depends on whether the feature-map associated with the base kernel K is finite-dimensional or not.

D.7.2.1 When the kernel is linear, quadratic or associated with a finite-dimensional feature map.

If the feature-map associated with the base kernel K is finite-dimensional, then γ can be estimated from the data itself when the number of data-points is sufficiently large (at least larger than the dimension of the feature-space $\mathcal{S}$). A prototypical example (when trying to identify the ancestors of the variable x_1) is $K = K_s$=(D.39) with $\beta_3 = 0$. In the general setting assume that $K(x, x') := \psi(x)^T\psi(x')$ where the range $\mathcal{S}$ of ψ is finite-dimensional. Assume that $f^\dagger$ belongs to the RKHS defined by ψ, i.e., assume that it is of the form $f^\dagger = v^T\psi$ for some v in the feature-space. Then (D.51) reduces to

$$v^T\psi(X) + \sigma Z = Y\,, \tag{D.56}$$

and, in the large data regime, σ^2 can be estimated by

$$\bar{\sigma}^2 := \frac{1}{N}\inf_{w\in\mathcal{S}} \left\|w^T\psi(X) - Y\right\|^2_{\mathbb{R}^N}. \tag{D.57}$$

Our strategy, when the feature map is finite-dimensional, is then to select

$$\gamma = N\bar{\sigma}^2 = \inf_{w\in\mathcal{S}} \left\|w^T\psi(X) - Y\right\|^2_{\mathbb{R}^N}. \tag{D.58}$$

D.7.2.2 When the kernel is interpolatory (associated with an infinite-dimensional feature map).

If the feature-map associated with the base kernel K is infinite-dimensional (or has more dimensions than we have data points) then it can interpolate the data exactly and the previous strategy cannot be employed since the minimum of (D.57) is zero. A prototypical example (when trying to identify the ancestors of the variable x_1) is $K = K_s$=(D.39) with $\beta_3 > 0$. In this situation, we do not attempt to estimate the level of noise σ but select a prior γ such that the resulting noise-to-signal ratio can effectively differentiate noise from signal. To describe this, observe that the

noise-to-signal ratio (D.54) admits the representer formula

$$\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)} = \frac{Y^T D_\gamma^2 Y}{Y^T D_\gamma Y}\,, \tag{D.59}$$

involving the $N \times N$ matrix

$$D_\gamma := \gamma\big(K(X,X)+\gamma I\big)^{-1}\,. \tag{D.60}$$

Observe that $0 \le D_\gamma \le I$, and

$$\lim_{\gamma\downarrow 0} D_\gamma = 0 \text{ and } \lim_{\gamma\uparrow\infty} D_\gamma = I\,. \tag{D.61}$$

Write (λ_i, e_i) for the eigenpairs of $K(X,X)$ ($K(X,X)e_i = \lambda_i e_i$) where the λ_i are ordered in decreasing order. Then the eigenpairs of D_γ are (ω_i, e_i) where

$$\omega_i := \frac{\gamma}{\gamma+\lambda_i}\,. \tag{D.62}$$

Note that the ω_i are contained in $[0,1]$ and also ordered in decreasing order.

Writing $\bar{Y}_i$ for the orthogonal projection of Y onto e_i, we have

$$\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)} = \frac{\sum_{i=1}^n \omega_i^2 \bar{Y}_i^2}{\sum_{i=1}^n \omega_i \bar{Y}_i^2}\,, \tag{D.63}$$

It follows that if the histogram of the eigenvalues of D_γ is concentrated near 0 or near 1, then the noise-to-signal ratio is non-informative since the prior γ dominates it. To avoid this phenomenon, we select γ so that the eigenvalues of D_γ are well spread out in the sense that the histogram of its eigenvalues has maximum or near-maximum variance (see Fig. D.1 for a good choice and a bad choice for γ). If the eigenvalues have an algebraic decay, then this is equivalent to taking γ to be the geometric mean of those eigenvalues.

In practice, we use an off-the-shelf optimizer to obtain γ by maximizing the sample variance of $(\omega_i)_{i=1}^n$. If this optimization fails, we default to the median of the eigenvalues. This ensures a balanced, well-spread spectrum for $D\gamma$, with half of the eigenvalues λ_i being lower and half being higher than the median.

D.7.2.3 Rationale for the choices of γ

The purpose of this section is to present a rationale for the proposed choices for γ in Sec. D.7.2.1 and D.7.2.2. For the choice Sec. D.7.2.1, we present an asymptotic

analysis of the signal-to-noise ratio in the setting of a simple linear regression problem. According to (D.58), γ must scale linearly in N; this scaling is necessary to achieve a ratio that represents the signal-to-noise per sample. Without it (if γ remains bounded as a function of N), this scaling of the signal-to-noise would converge towards 0 as $N \to \infty$. To see how we will now consider a simple example in which we seek to linearly regress the variable y as a function of the variable x, both taken to be scalar (in which case $\psi(x) = x$). Assume that the samples are of the form $Y_i = aX_i + \sigma Z_i$ for $i = 1, \ldots, N$, where $a, \sigma \neq 0$, the Z_i are i.i.d. $\mathcal{N}(0, 1)$ random variables, and the X_i satisfy $\frac{1}{N}\sum_{i=1}^N X_i = 0$ and $\frac{1}{N}\sum_{i=1}^N X_i^2 = 1$. Then, the signal-to-noise ratio is $\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)}$ with $\mathcal{V}(s) = |v|^2$ and $\mathcal{V}(n) = \frac{1}{\gamma}\sum_{i=1}^N |vX_i - Y_i|^2$ and v is a minimizer of

$$\min_{v\in\mathbb{R}} |v|^2 + \frac{1}{\gamma}\sum_{i=1}^N |vX_i - Y_i|^2 \,. \tag{D.64}$$

In asymptotic $N \to \infty$ regime, we have $v \approx \frac{aN}{\gamma+N}$ and

$$\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)} \approx \frac{\frac{\gamma}{N}a^2}{-a^2(\gamma/N+1) + (a^2+\sigma^2)(\gamma/N+1)^2} \,. \tag{D.65}$$

If γ is bounded independently from N, then $\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)}$ converges towards zero as $N \to \infty$, which is undesirable as it does not represent a signal-to-noise ratio per sample. If $\gamma = N$, then $\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)} \approx \frac{a^2}{4\sigma^2+2a^2}$, which does not converge to 1 as $a \to \infty$ and $\sigma \to 0$, which is also undesirable. If γ is taken as in (D.58), then $\gamma \approx N\sigma^2$ and

$$\frac{\mathcal{V}(s)}{\mathcal{V}(s)+\mathcal{V}(n)} \approx \frac{a^2}{(\sigma^2+1)(a^2+\sigma^2+1)} \,, \tag{D.66}$$

which converges towards 0 as $\sigma \to \infty$ and towards $1/(1+\sigma^2)$ as $a \to \infty$, which has, therefore, the desired properties.

Moving to Sec. D.7.2.2, because the kernel can interpolate the data exactly we can no longer use (D.57) to estimate the level of noise σ. For a finite-dimensional feature map ψ, with data (X, Y), we can decompose $Y = v^T\psi(X) + \sigma Z$ into a signal part Y_s and noise part Y_s, s.t. $Y = Y_s + Y_n$. While Y_s belongs to the linear span of eigenvectors of $K(X, X)$ associated with non-zero eigenvalues, Y_n also activates the eigenvectors associated with the null space of $K(X, X)$ and the projection of Y onto that null-space is what allows us to derive γ in Sec. D.7.2.1. Since in the interpolatory case, all eigenvalues are strictly positive, we need to choose which eigenvalues are associated with noise differently, as is described in the previous section. With a fixed γ, we

see that if $\lambda_i \gg \gamma$, then $\omega_i \approx 0$, which contributes in (D.63) to yield a low noise-to-signal ratio. Similarly, if $\lambda_i \ll \gamma$, this eigenvalue yields a high noise-to-signal ratio. Thus, we see that the choice of γ assigns a noise level to each eigenvalue. While in the finite-dimensional feature map setting, this assignment is binary, here we perform soft thresholding using $\lambda \mapsto \gamma/(\gamma + \lambda)$ to indicate the level of noise of each eigenvalue. This interpretation sheds light on the selection of γ in equation (D.58). Let ψ represent the feature map associated with K. Assuming the empirical mean of $\psi(X_i)$ is zero, the matrix $K(X, X)$ corresponds to an unnormalized kernel covariance matrix $\psi^T(X)\psi(X)$. Consequently, its eigenvalues correspond to N times the variances of the $\psi(X_i)$ across various eigenspaces. After conducting Ordinary Least Squares regression in the feature space, if the noise variance is estimated as $\bar{\sigma}^2$, then any eigenspace of the normalized covariance matrix whose eigenvalue is lower than $\bar{\sigma}^2$ cannot be recovered due to the noise. Given this, we set the soft thresholding cutoff to be $\gamma = N\bar{\sigma}^2$ for the unnormalized covariance matrix $K(X, X)$.

D.7.3 Z-score/quantile bounds on the noise-to-signal ratio.

If the data is only comprised of noise, then an interval of confidence can be obtained on the noise-to-signal ratio. To describe this consider the problem of testing the null hypothesis $\mathbf{H_0} : f^\dagger \equiv 0$ (there is no signal) against the alternative hypothesis $\mathbf{H_1} : f^\dagger \not\equiv 0$ (there is a signal). Under the null hypothesis $\mathbf{H_0}$, the distribution of the noise-to-signal ratio (D.59) is known and it follows that of the random variable

$$B := \frac{Z^T D_\gamma^2 Z}{Z^T D_\gamma Z}\,. \tag{D.67}$$

Therefore, the quantiles of B can be used as an interval of confidence on the noise-to-signal ratio if $\mathbf{H_0}$ is true. More precisely, selecting β such that $¶[B \le \beta_\alpha] \approx \alpha$ with $\alpha = 0.05$ as a prototypical example, we expect the noise to signal ratio (D.59) to be, under $\mathbf{H_0}$, to be larger than β_α with probability $\approx 1 - \alpha$. The estimation of β requires Monte-Carlo sampling.

An alternative approach (in the large data regime) to using the quantile β_α is to use the Z-score

$$\mathcal{Z} := \frac{\frac{Y^T D_\gamma^2 Y}{Y^T D_\gamma Y} - \mathbb{E}[B]}{\sqrt{\mathrm{Var}[B]}}\,, \tag{D.68}$$

after estimating $\mathbb{E}[B]$ and $\mathrm{Var}[B]$ via Monte-Carlo sampling. In particular if $\mathbf{H_0}$ is true then $|\mathcal{Z}| \ge z_\alpha$ should occur with probability $\approx \alpha$ with $z_{0.1} = 1.65$, $z_{0.05} = 1.96$ and $z_{0.01} = 2.58$.

Remark D.7.2. *Although the quantile β_α or the Z-score $\mathcal{Z}$ can be employed to produce an interval of confidence on the noise-to-signal ratio under* $\mathbf{H_0}$ *we cannot use them as thresholds for removing nodes from the list of ancestors as discussed in Sec. D.5.3.4 Indeed, observing a noise-to-signal ratio* (D.59) *below the threshold β_α does not imply that all the signal has been captured by the kernel; it only implies that some signal has been captured by the kernel K. To illustrate this point, consider the setting where one tries to approximate the variable x_1 as a function of the variable x_2. If x_1 is not a function of x_2, but of x_2 and x_3, as in $x_1 = \cos(x_2) + \sin(x_3)$, then applying the proposed approach with Y encoding the values of x_1, X encoding the values of x_2, and the kernel K depending on x_2 could lead to a noise-to-signal ratio below β_α due to the presence of a signal in x_2. Therefore, although we are missing the variable x_3 in the kernel K, we would still observe a possibly low noise-to-signal ratio due to the presence of* some *signal in the data. Summarizing if the data only contains noise then $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)} \geq \beta_\alpha$ should occur with probability $1-\alpha$. If the event $\frac{\mathcal{V}(n)}{\mathcal{V}(s)+\mathcal{V}(n)} < \beta_\alpha$ is observed in the setting of $K = K_{s/t}$=(D.49) where we try to identify the ancestors of x_1, then we can only deduce that $x_3, \ldots, x_d$ contain some signal but perhaps not all of it (we can use this a criterion for pruning x_2).*

D.8 Supplementary information on examples.

D.8.1 Algebraic equations.

Although we have used Alg. 8 for the algebraic equations examples presented in Fig. 5.4, Alg. 7 yields the same results with the default signal-to-noise threshold $\tau = 0.5$.

D.8.2 The chemical reaction network.

Consider the chemical reaction network example illustrated in Fig. 5.4.(a). The proposed mechanism for the hydrogenation of ethylene (C_2H_4) to ethane (C_2H_6), is (writing $[H]$ for the concentration of H) modeled by the following system of differential equations

$$\begin{aligned}
\frac{d[H_2]}{dt} &= -k_1[H_2] + k_{-1}[H]^2 \\
\frac{d[H]}{dt} &= 2k_1[H_2] - 2k_{-1}[H]^2 - k_2[C_2H_4][H] - k_3[C_2H_5][H] \\
\frac{d[C_2H_4]}{dt} &= -k_2[C_2H_4][H] \\
\frac{d[C_2H_5]}{dt} &= k_2[C_2H_4][H] - k_3[C_2H_5][H]
\end{aligned} \tag{D.69}$$

The primary variables are the concentrations $[H_2]$, $[H]$, $[C_2H_4]$ and $[C_2H_5]$ and their time derivatives $\frac{d[H_2]}{dt}$, $\frac{d[H]}{dt}$, $\frac{d[C_2H_4]}{dt}$ and $\frac{d[C_2H_5]}{dt}$. The computational hypergraph encodes the functional dependencies (D.69) associated with the chemical reactions. The hyperedges of the hypergraph are assumed to be unknown and the primary variables are assumed to be known. Given N samples from the graph of the form

$$\big([H_2](t_i), [H](t_i), [C_2H_4](t_i), [C_2H_5](t_i)\big)_{i=1,\dots,N} \tag{D.70}$$

our objective is to recover the structure of the hypergraph given by (D.69), representing the functions by hyperedges. We create a dataset of the form (D.70) by integrating 50 trajectories of (D.69) for different initial conditions, and each equispaced 50 times from $t = 0$ to $t = 5$. The dataset is represented in Fig. 5.4.(b) (the time derivatives of concentrations are estimated by taking the derivatives of the interpolants of those concentrations). We impose the information that the derivative variables are function of the non-derivative variables to avoid ambiguity in the recovery, as (D.69) is not the unique representation of the functional relation between nodes in the graph. We implement Alg. 7 with weights $\beta = [0.1, 0.01, 0.001]$ for linear, quadratic, and nonlinear, respectively (Alg. 8 recovers the same hypergraph). The output graph can be seen in Fig. 5.4.(b). We obtain a perfect recovery of the computational graph and a correct identification of the relations being quadratic.

D.8.3 The Google Covid 19 open data.

Consider the example illustrated in Fig. 5.3.(e-k). Categorical data are treated as scalar values, with all variables scaled to achieve a mean of 0 and a variance of 1. We implement three distinct kernel types: linear, quadratic, and Gaussian, with a length scale of 1 for the latter. A weight ratio of $1/10$ is assigned between kernels, signifying that the quadratic kernel is weighted ten times less than the linear kernel. Lastly, the noise parameter, γ, is determined using the optimal value outlined in Sec. D.7. Initially, a complete graph is constructed using all variables, depicted in Fig. 5.3.(g). This construction is done using only linear and quadratic kernels. The full graph is highly clustered and redundant information is eliminated by selecting representative nodes for each cluster. Eliminating redundant nodes is important for two reasons: firstly, it improves the graph's readability, especially with 31 variables; secondly, it avoids hindering graph discovery. In an extreme case, treating two identical variables as distinct would result in one variable's ancestor simply being its duplicate, yielding an uninformative graph. Subsequently, the graph discovery algorithm is rerun, with reduced variables due to eliminating redundancy, ushering

us into a predominantly noisy regime. With fewer variables available, we use additionally the nonlinear kernel. Two indicators are employed to navigate our discovery process: the signal-to-noise ratio and the Z-test. The former quantifies the degree to which our regression is influenced by noise, while the latter signals the existence of any signal. We follow the procedure in algorithm 8, resulting in the graph presented in Fig. 5.3.(k).

D.8.4 Cell signaling network

Consider the example Fig. 5.1.(l) from [285] and Fig. 5.4.(h-j). To identify the ancestors of each node, we apply the algorithm in two stages. First, we learn the dependencies using only linear and quadratic kernels. Fig. 5.4.(h) identifies the resulting graph learned given a subset of $N = 2,000$ samples chosen uniformly at random from the dataset. We observe that the graph identified by the algorithm consists of four disconnected clusters where the molecule levels in each cluster are closely related by linear or quadratic dependencies (all connections are linear except for the connection between Akt and PKA, which is quadratic). These edges match a subset of the edges found in the gold standard model identified in [285]. With perfect dependencies that have no noise, one can define constraints that reduce the total number of variables in the system. For this noisy dataset that, we treat these dependencies as forming groups of similar variables and introduce a hierarchical approach to learn the connections between groups. Second, we run the graph discovery algorithm after grouping the molecules into clusters. For each node in the graph, we identified the ancestors of each node by constraining the dependence to be a subset of the clusters. In other words, when identifying the ancestors of a given node i in cluster C, the algorithm is only permitted to (1) use ancestors that do not belong to cluster C, and (2) include all or none of the variables in each cluster (j in cluster $D \neq C$ is listed as an ancestor if and only if all other nodes j' in cluster D are also listed as ancestors). The ancestors were identified using a Gaussian (fully nonlinear) kernel and the number of by ancestors were selected manually based on the inflection point in the noise-to-signal ratio. The resulting graph is depicted in Fig. 5.4.(i). Each edge is weighted based on its signal-to-noise ratio. We observe that there is a stronger dependence of the Jnk, PKC, and P38 cluster on the PIP3, Plcg, and PIP2 cluster, which closely matches the gold standard model. As compared to approaches based on acyclic DAGs, however, the graph identified by our algorithm also contains feedback loops between the various molecule levels. Fig. 5.4.(i-j) displays a side-by-side comparison between the graph identified with our method

and the graph generated in [285]. To aid in this comparison, we have highlighted different clusters in distinct colors. We emphasize that while the Bayesian network analysis in [285] relied on the control of the sampling of the underlying variables (the simultaneous measurement of multiple phosphorylated protein and phospholipid components in thousands of individual primary human immune system cells, and perturbing these cells with molecular interventions), the reconstruction obtained by our method did not use this information and recovered functional dependencies rather than causal dependencies. Interestingly, the information recovered through our method appears to complement and enhance the findings presented in [285] (e.g., the linear and noiseless dependencies between variables in the JNK cluster is not something that could easily be inferred from the graph produced in [285]).

D.8.5 BCR reaction network

In the high-dimensional example of the BCR reaction network, the computations of terms of the form $y^T k_o(X,X)y$ (i.e., the activations), where $y \in \mathbb{R}^n$ and $k_o(X,X)$ is the o-th coordinate of the quadratic kernel ($k(x_i,x_j) = (1+\langle x_i, x_j\rangle)^2$) becomes the computational bottleneck of our method. If we let $x_1,\ldots,x_n \in \mathbb{R}^p$ be the points and x_i^o be the o-th coordinate of x_i, we can compute the activation of the $o-$th coordinate using

$$k_o(x_i,x_j) = (1+x_i^o x_j^o)^2 - 1 + 2x_i^o x_j^o \langle x_i^{-o}, x_j^{-o}\rangle \tag{D.71}$$

where k_o is the $o-$th coordinate of the kernel and x_i^{-o} represents the remaining coordinates of x_i. To compute the $n\times n$ kernel matrix of k_o for each $o \in \{1,..,p\}$, we must compute $p\times n\times n$ inner products in $\mathbb{R}^p$, which is a very large computation. Instead, we may use the following reformulation to speed up computations. Notice $\langle x_i,x_j\rangle = x_i^o x_j^o + \langle x_i^{-o}, x_j^{-o}\rangle$, and therefore $k_o(x_i,x_j) = 2x_i^o x_j^o\langle x_i,x_j\rangle + 2x_i^o x_j^o - (x_i^o x_j^o)^2$. Now, define $v^o = (x_i^o y_i)_{i=1}^p$ and $w^o = ((x_i^o)^2 y_i)_{i=1}^p$, and note that

$$y^T K_o y = \sum_{i,j} 2y_i x_i^o y_j x_j^o (1+\langle x_i,x_j\rangle) - \sum_{i,j} y_i y_j (x_i^o x_j^o)^2 \tag{D.72}$$

and so defining $\tilde{K} = (2(1+\langle x_i,x_j\rangle))_{i,j=1}^n$ we have that

$$y^T K_o y = v^{oT}\tilde{K}v^o - \left(\sum_{i=1}^p w_i^o\right)^2 \tag{D.73}$$

Note that $\tilde{K}$ is computed just once for all p, and only v^o and w^o change for every ancestor calculation, which is where the main computational gain comes from. One may find in the GitHub repository of the paper a comparison of the two methods

of computations and observe a tenfold speedup. This speedup is even larger in our implementation of the BCR example, as GPU acceleration enables the second method to run even faster.

Appendix E

APPENDIX TO CHAPTER 6: GRAPHICAL CONDITIONAL GENERATIVE MODELING

E.1 Conditional generative modeling via flow matching

In this section, we describe the conditional generative modeling stage. The objective is to learn a sampler for the conditional law of y given $X = (x_1, \ldots, x_n)$. We use flow matching models [190], which learn a velocity field transporting a simple base distribution to the target conditional distribution through the ordinary differential equation

$$\frac{d\zeta_t}{dt} = v_\theta(t, \zeta_t, X), \qquad t \in (0, 1), \tag{E.1a}$$

$$\zeta_0 = z, \qquad z \sim p_0, \tag{E.1b}$$

where v_θ is a neural network parameterized by θ, and p_0 is a tractable base distribution, typically a standard Gaussian. The corresponding solution map defines the conditional sampler

$$\hat{y} = \mathcal{G}_\theta(X, z) := \zeta_1, \tag{E.2}$$

where ζ_1 is obtained by integrating (E.1) from $t = 0$ to $t = 1$. Training is performed by the flow matching regression objective. Given data pairs (X, y) and an independent sample $z \sim p_0$, define the linear interpolation

$$\tilde{\zeta}_t = (1 - t)z + ty, \tag{E.3}$$

whose target velocity is $\dot{\tilde{\zeta}}_t = y - z$. We then learn v_θ by solving

$$\min_\theta \ \mathbb{E}_{(X,y),\, z\sim p_0,\, t\sim\mathcal{U}(0,1)} \left[\left\| (y - z) - v_\theta(t, \tilde{\zeta}_t, X) \right\|^2 \right]. \tag{E.4}$$

After training, samples from the learned approximation of $y \mid X$ are generated by drawing $z \sim p_0$ and evaluating $\hat{y} = \mathcal{G}_\theta(X, z)$. This procedure is known to converge in the limit of infinite data [5] under sufficient regularity, and well-specified approximation class assumptions.

E.2 Algorithm for stochastic ancestor discovery

Here we present the algorithm for stochastic ancestor discovery.

Algorithm 9 Stochastic ancestor discovery via generative distillation

Require: Observational data $\{(X^i, y^i)\}_{i=1}^N$
Ensure: Identified stochastic ancestor set $\mathcal{A}$ for y

Stage 1: Conditional generative modeling
Train a conditional flow matching model to learn the velocity field $v_\theta(t, \zeta, X)$ from $\{(X^i, y^i)\}_{i=1}^N$
Obtain the conditional sampler $\hat{y} = \mathcal{G}_\theta(X, z)$ with $z \sim p_0$
Stage 2: Distillation and ancestor discovery
Generate synthetic samples $\hat{y}^i = \mathcal{G}_\theta(X^i, z^i)$ with $z^i \sim p_0$
Construct the distilled dataset $\mathcal{D} = \{(X^i, z^i, \hat{y}^i)\}_{i=1}^M$
Initialize candidate set $\mathcal{A}_0 = \{x_1, \ldots, x_n\}$
Initialize NSR record $\mathcal{R} = []$ and ancestor-set record $\mathcal{S} = []$
Set $k = 0$
while $\mathcal{A}_k$ is not empty **do**
    Train a GP surrogate $f_k(X_{\mathcal{A}_k}, z)$ on $\mathcal{D}$
    Compute and append NSR_k to $\mathcal{R}$
    Append $\mathcal{A}_k$ to $\mathcal{S}$
    Compute contribution scores c_j for all $x_j \in \mathcal{A}_k$ via KMD
    Let $j_k \in \arg\min_{x_j \in \mathcal{A}_k} c_j$
    Remove the least influential variable: $\mathcal{A}_{k+1} = \mathcal{A}_k \setminus \{x_{j_k}\}$
    Set $k \leftarrow k + 1$
end while
Compute increments $\Delta\mathcal{R}_k = \mathcal{R}_{k+1} - \mathcal{R}_k$
Let $k^* \in \arg\max_k \Delta\mathcal{R}_k$
return $\mathcal{A} = \mathcal{S}_{k^*}$

E.3 Additional details

We provide implementation and simulation details for the numerical examples in the main text. In all numerical examples, unless stated otherwise, we use (1) the Adam optimizer [163] with learning rate 1e-4 to train the flow matching model, (2) the Diffrax ODE solver [162] (adaptive fifth-order Runge–Kutta integration with PID step-size control; 1e-5 relative and absolute tolerances) to generate samples from the trained flow matching model, and (3) the nugget 1.0 in GP regression.

E.3.1 Recovering the computational graph of a stochastic differential equation

The stochastic Lorenz–63 system used in Section 6.2.3 is

$$dx_1 = \sigma(x_2 - x_1)\,dt + \alpha x_1\,dW_1, \tag{E.5a}$$

$$dx_2 = \big(x_1(\rho - x_3) - x_2\big)\,dt + \alpha x_2\,dW_2, \tag{E.5b}$$

$$dx_3 = (x_1 x_2 - \beta x_3)\,dt + \alpha x_3\,dW_3, \tag{E.5c}$$

where W_i are independent Wiener processes. We use the standard parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, noise amplitude $\alpha = 0.5$, and time step $\Delta t = 0.01$. Initial conditions are sampled independently from $\mathrm{Unif}([-15, 15]^3)$. The system is simulated with the Euler–Maruyama method. We generate 50 independent trajectories; each trajectory is run for 200 burn-in steps and then contributes 1,000 state-increment pairs.

To assess run-to-run robustness, we repeat the experiment ten times independently. In both settings (unknown drift and diffusion in Figure 6.2(b), and known drift with unknown diffusion in Figure 6.2(c)), all ten runs recover the same graphs. The NSR and NSR-increment curves in Figure 6.2(b)–(c) report the mean over the ten runs, with error bars indicating one standard deviation. The target-wise pruning orders are also largely consistent: for Δx_2, R_2, and R_3, all ten runs yield the same pruning orders, while for Δx_1, Δx_3, and R_1, nine, nine, and eight runs yield the same pruning orders, respectively.

We also use this example to assess the approximation error introduced by the two-stage architecture: a flow matching model is first learned from data, then distilled into a GP surrogate on which KMD is performed. Specifically, we evaluate the GP surrogate in two ways: first, by its regression accuracy on distilled samples $\{(X^i, z^i, \hat{y}^i)\}_{i=1}^N$ generated from the learned conditional sampler; and second, by the accuracy with which it reproduces the corresponding conditional distributions on test inputs generated from the same data-generating process. Results for $\Delta x_{1,n}$ are presented in Table E.1. The Wasserstein-1 distances of the GP surrogate and the trained flow matching model relative to the exact data-generating process are statistically indistinguishable, confirming that distillation introduces no significant additional error beyond that of the flow matching stage. The number of validation distilled samples is 10,000. We further examine the effect of the number of distilled samples N on ancestor discovery. The GP surrogate is trained separately for each target variable $\Delta x_{k,n}$. For $\Delta x_{1,n}$ and $\Delta x_{3,n}$, correct ancestors are recovered for all values of N tested. For $\Delta x_{2,n}$, $N = 500$ and $N = 2{,}000$ yield $\mathcal{A}(\Delta x_{2,n}) = \{x_{1,n}, x_{3,n}\}$, missing $x_{2,n}$, while $N = 5{,}000$ and $N = 10{,}000$ recover the complete ancestor set. This reflects the fact that the dependence of $\Delta x_{2,n}$ on $x_{2,n}$ is the weakest among the three components, and a sufficiently faithful GP surrogate is required to resolve it.

In this example, we model the conditional distributions of the increments $\Delta x_{k,n}$, $k = 1, 2, 3$, given the current state $(x_{1,n}, x_{2,n}, x_{3,n})$. We use a fully-connected neural network (FNN) with four hidden layers of 64 neurons each and sigmoid linear unit

	$N = 500$	$N = 2{,}000$	$N = 5{,}000$	$N = 10{,}000$
MSE on validation distilled samples	6.02e-2	2.08e-2	8.34e-3	5.19e-3
The Wasserstein-1 distance (mean±std)	7.47e-2±1.23e-1	4.09e-2±6.54e-2	3.31e-2±3.82e-2	3.11e-2±3.30e-2

Table E.1: Approximation quality of the GP surrogate for the generative model of $\Delta x_{1,n} := x_{1,n+1} - x_{1,n}$. *Top row*: the regression mean-squared error (MSE) on validation distilled samples generated from the learned conditional sampler. *Bottom row*: Wasserstein-1 distance to the exact data-generating process, evaluated on 1,000 test inputs with 1,000 surrogate samples drawn per test point. For reference, the Wasserstein-1 distance of the trained flow matching model to the exact data-generating process is 3.42e-2±2.55e-2.

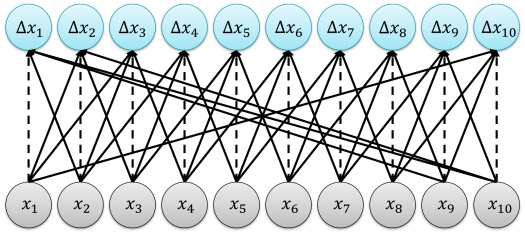


Figure E.1: Recovered computational graph of the multiscale Lorenz 96 system with missing fast variables: dashed lines represent that Δx_j having x_j as an ancestor is given as a prior knowledge to the method.

(SiLU) activation function to parameterize the velocity field of the flow matching model, with batch size and number of epochs both set to 1,000. In the second stage, we use $N = 5{,}000$ distilled samples $\{x^i_{k,n}, z^i_k, \Delta\hat{x}_{k,n}, k = 1, 2, 3\}_{i=1}^{5{,}000}$ to train the GP surrogate. We use the RBF kernel with lengthscale 1.0 for both the state and the random variables.

E.3.2 Recovering dependence of the slow variables in a multiscale system.

In this example, we consider the multi-scale Lorenz 96 dynamical system [22, 201]:

$$\frac{dx_j}{dt} = (x_{j+1} - x_{j-2})x_{j-1} - x_j + F - \frac{hc}{b}\sum_{k=1}^{J} y_{k,j}, j = 1, ..., m, \tag{E.6a}$$

$$\frac{dy_{k,j}}{dt} = -cby_{k+1,j}(y_{k+2,j} - y_{k-1,j}) - cy_{k,j} + \frac{hc}{b}x_j, k = 1, ..., J. \tag{E.6b}$$

Here, $x_j, j = 1, ..., m$ are slow variables and $y_{k,j}, k = 1, ..., J$ are fast variables. We consider $J = 10$ and $m = 10, 20, ..., 90, 100, 150, 200$. We use the standard parameters $F = 10, c = 10, b = 10$, and coupling strength $h = 0.1$. To generate data, we solve (E.6) using a fourth-order Runge-Kutta scheme with time step $\Delta t = 0.001$,

where the initial conditions are small random perturbation (Gaussian distribution with mean zero and standard deviation 0.01) around F and 0 for the slow and fast variables, respectively. One trajectory is simulated for 102,000 steps where the first $2,000$ are considered as burn-in, and snapshots are saved every 10 steps, contributing to 10,000 state-increment pairs.

For $m = 10$, we model the distributions of $\Delta x_{j,n}, j = 1, ..., m$ individually by employing the proposed method m times independently (Figure E.1). For $m > 10$ where the scalability of our method is assessed, we only consider the distribution of $\Delta x_{1,n}$ as the rest is the same in terms of the scalability. In the first stage of the proposed method, we use a FNN with four hidden layers of 64 neurons each with SiLU activation function to parameterize the velocity field, with batch size and number of epochs set to 1,000 and 2,000, respectively. In the second stage, we use 2,000 distilled samples and the RBF kernel with lengthscales 5.0 and 1.0 for the state and the random variables, respectively.

E.3.3 Controlling stochastic pendulum with missing variables

The stochastic pendulum dynamics are

$$\ddot{\theta}_t = \frac{3g}{2l}(1 + w_t)\sin\theta_t + \frac{3}{ml^2}u_t, \qquad w_t \sim \mathcal{N}(0, \sigma^2). \tag{E.7}$$

To generate data, we collect transition data from (E.7) by discretizing the system with time step $\Delta t = 0.05$, torque $u_n \in [-2, 2]$, mass $m = 1$, length $l = 1$, gravity $g = 9.81$, and noise scale $\sigma = 0.3$. We use a customized Gymnasium-compatible simulator [334] with the dynamics in (E.7), which integrates semi-implicit Euler updates with angular velocity clipped to the maximum speed ± 8. We run 500 episodes of length $T = 200$ from random initial angles $\theta_0 \sim \text{Unif}([-\pi, \pi])$ and angular velocities $\dot{\theta}_0 \sim \text{Unif}([-1, 1])$, applying torques $u_t \sim \text{Unif}([-2, 2])$. Each sample records $(\theta_n, u_n, \theta_{n+1})$ with unwrapped θ; for learning, we use inputs $(\cos(\theta_n), \sin(\theta_n), u_n)$ and target $\Delta\theta_n = \theta_{n+1} - \theta_n$, yielding 100,000 state-increment pairs.

To discover the ancestors of the dynamics with missing angular velocity from the 10 past state-action pairs $\{(\theta_{n-k}, u_{n-k})\}_{k=0}^{9}$, we first use a FNN with four hidden layers of 64 neurons each and SiLU activation function to parameterize the velocity field, and train it with batch size and number of epochs set to 1,000. We then use 5,000 distilled samples and the periodic kernel ($k_x(x, x') = \exp(-2\sin^2((x - x')/2)/l^2)$) with lengthscale 1.0, the RBF kernel with lengthscale 1.0, and the RBF kernel with lengthscale 10.0 for the state, control, and random variables,

respectively. In the second stage, we replace $\cos(\theta_{n-k}), \sin(\theta_{n-k})$ with $\theta_{n-k} = \text{atan2}(\sin(\theta_{n-k}), \cos(\theta_{n-k}))$.

As mentioned in the main text, we evaluate the sufficiency and accuracy of the discovered graphs through control performance. This setup reflects a common scenario in real applications: the true dynamics are unknown, and one must learn a surrogate from data. Such a surrogate can then serve as the environment for a range of control methods, including MPC and RL approaches, where the surrogate is used as a simulator within which a control policy is learned. An accurately learned surrogate should produce a well-trained agent that transfers successfully to the true environment. In this example, we adopt the model-free RL framework to test the surrogate learned with the discovered graphs. Specifically, we train a conditional flow matching model [190] on each discovered graph to learn the conditional distribution of $\Delta\theta_n$. Then, following the standard model-free RL protocol, we use the trained flow matching model as a surrogate environment to train a SAC agent [132] using Stable-Baselines3 [269], with custom environments implemented in Gymnasium [334]. The agent observes a fixed window of $W' = 3$ past angles and actions to compensate for not observing the angular velocity. The trained policy is then transferred to the true stochastic environment without any further training, and evaluated by mean episode reward computed from 500 independently executed episodes. A surrogate built on a sufficient ancestor set should produce a policy that transfers successfully; one built on an insufficient set should fail.

E.3.3.1 Details of the downstream control setup

The control objective is to swing-up and stabilize a stochastic, partially observed pendulum. The controller applies torque to drive the pendulum toward the upright equilibrium $\theta = 0 \pmod{2\pi}$. The agent does not observe angular velocity $\dot{\theta}$; instead, it receives a short history of angle encodings and past actions. This setting models control under latent velocity, where the policy must infer dynamical information from delayed observations.

The true environment follows a discretized, torque-controlled pendulum with multiplicative gravitational noise. At each step n, with latent state $(\theta_n, \dot{\theta}_n)$ and control

input u_n,

$$w_n \sim \mathcal{N}(0, \sigma^2), \qquad \sigma = 0.3, \tag{E.8}$$

$$\alpha_n = \frac{3g}{2l}(1 + w_n)\sin(\theta_n) + \frac{3}{ml^2}u_n, \tag{E.9}$$

$$\dot{\theta}_{n+1} = \mathrm{clip}\big(\dot{\theta}_n + \alpha_n\,\Delta t,\ -\dot{\theta}_{\max},\ \dot{\theta}_{\max}\big), \qquad \dot{\theta}_{\max} = 8 \text{ rad/s}, \tag{E.10}$$

$$\theta_{n+1} = \theta_n + \dot{\theta}_{n+1}\,\Delta t. \tag{E.11}$$

The noise term w_n perturbs the gravitational acceleration, yielding stochastic transitions even under deterministic policies.

The agent does not receive $\dot{\theta}_n$. Instead, at step n it observes a sliding window of the most recent $W' = 3$ frames. Each frame consists of $(\cos\theta, \sin\theta, u)$, where u denotes the previously applied torque (set to zero during warm-up). Frames are stacked with the most recent frame first, giving

$$\begin{aligned} o_n = \big[&\cos\theta_n,\ \sin\theta_n,\ u_{n-1}, \cos\theta_{n-1},\ \sin\theta_{n-1},\ u_{n-2}, \\ &\cos\theta_{n-2},\ \sin\theta_{n-2},\ u_{n-3}\big] \in \mathbb{R}^{3W'}. \end{aligned} \tag{E.12}$$

Thus $\dim(o_n) = 9$ when $W' = 3$. Using $(\cos\theta, \sin\theta)$ avoids discontinuities at $\pm\pi$. This defines a partially observed Markov decision process: the latent state is $(\theta_n, \dot{\theta}_n)$, while the policy $\pi(u_n \mid o_n)$ acts on the 9-dimensional observation only.

The control input is a scalar continuous torque applied at each control step after clipping. Our experiments use the *angle-only* reward, which is fully determined by quantities accessible from the observation history:

$$r_n = -\left(\bar{\theta}_{n+1}^2 + 0.001\,u_n^2\right), \tag{E.13}$$

where the wrapped angle is

$$\bar{\theta} = ((\theta + \pi) \bmod 2\pi) - \pi \in [-\pi, \pi). \tag{E.14}$$

This cost penalizes deviation from the upright target and control effort. The agent maximizes the finite-horizon return

$$G = \sum_{n=0}^{T-1} r_n. \tag{E.15}$$

where $T = 200$ is the total number of control steps. In evaluation, we report the episode return G over multiple random seeds.

Each episode proceeds in three stages. First, we sample the latent initial state:

$$\theta_0 \sim \text{Uniform}[-\pi, \pi], \qquad \dot{\theta}_0 \sim \text{Uniform}[-1, 1].$$

Second, we apply a zero-control warm-up by rolling the true dynamics forward for $K = 10$ steps with $u = 0$, thereby filling the observation buffer before control begins. Third, we execute the learned policy for up to $T = 200$ control steps. Episodes terminate by time limit; there is no early termination upon reaching the goal. The warm-up protocol is shared across training and evaluation so that all policies face the same initial observation distribution.

Policies are trained with SAC [132] implemented in Stable-Baselines3 [269] (version 2.7.1), using the default `MlpPolicy` and default hyperparameters (learning rate 3×10^{-4}, replay buffer size 200,000, batch size 256, soft-update coefficient 0.005, discount factor $\gamma = 0.99$, 4 parallel environments, 200,000 training timesteps). The checkpoint with the highest evaluation return is retained for testing (evaluated every 10,000 steps over 10 episodes). The same RL setup is used for true-environment and flow-matching surrogate training.

We compare two training regimes: true-environment training and flow-matching surrogate training. For the former, SAC interacts directly with the stochastic pendulum dynamics. This setting serves as an upper bound on performance when the simulator is exact. For the latter, SAC interacts with a learned simulator that replaces the true transition. At each step, a conditional flow-matching model samples an angular increment $\Delta\theta$ given a conditioning vector built from past values of $\sin\theta$, $\cos\theta$, and u. The surrogate state update is

$$\theta_{n+1} = \theta_n + \Delta\theta_n, \tag{E.16}$$

where $\Delta\theta_n$ is sampled from the trained flow matching model. Importantly, the *agent observation and reward remain identical* to the true environment ($W' = 3$); only the transition model used during training changes. Different surrogate models correspond to different discovered dependency structures:

1. **Full history:** $\sin\theta$, $\cos\theta$, and u at lags $n, n-1, \ldots, n-9$ (30 dimensions);
2. **Graph 1:** $\sin\theta$, $\cos\theta$, and u at lags $n, n-1$ (6 dimensions);
3. **Graph 2:** $\sin\theta$, $\cos\theta$, and u at lags $n, n-3$ (6 dimensions);
4. **Markov:** $\sin\theta$, $\cos\theta$, and u at n only (3 dimensions).

In all cases, the policy still observes the same 9-dimensional window; the difference lies only in how the surrogate advances the hidden angle state during training. Surrogate episodes also begin with the same true-dynamics zero-control warm-up ($K = 10$ steps).

All trained policies, regardless of the simulator used during training, are evaluated on the *same* true stochastic pendulum with $W' = 3$, the same reward, $\sigma = 0.3$, and the shared warm-up protocol. For each evaluation seed s, the environment is reset deterministically from s, and the policy acts in deterministic mode (mean action). Performance is measured by the episode return G_s. This *train-on-surrogate/test-on-truth* protocol isolates how well each learned dynamics model supports transfer of control policies to the real plant.

E.3.4 Stochastic heat equation boundary control

We set $\sigma = 0.2$ and $\kappa = 0.01$. The spatial domain $[0, 1]$ is discretized into $m = 10$ equispaced grid points $x_i = (i-1)h$, $h = 1/(m-1)$, $i = 1, \ldots, m$. Applying second-order finite differences with ghost point elimination for the Neumann boundary conditions, and forward Euler time integration with step size $\Delta t = 0.2$, yields the discrete-time stochastic system

$$X_{n+1} = AX_n + BU_n + \mathrm{diag}(X_n)\omega_n, \qquad \omega_n \sim \mathcal{N}(0, \sigma^2 I_m), \tag{E.17}$$

where $X_n = (x_{1,n}, ..., x_{m,n})^\top \in \mathbb{R}^m$ is the discretized temperature field at time step n and $U_n = (u_{1,n}, u_{2,n})^\top \in \mathbb{R}^2$ is the control imposed on the boundaries. The state transition matrix $A \in \mathbb{R}^{m\times m}$ is tridiagonal and sparse, given by

$$A = I_m + r\tilde{A}, \qquad r = \frac{\kappa \Delta t}{h^2},$$

where $\tilde{A}$ is the discrete Laplacian with Neumann boundary conditions:

$$\tilde{A} = \begin{pmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{pmatrix}.$$

The control input matrix $B \in \mathbb{R}^{m\times 2}$ is sparse, with nonzero entries only at the boundary nodes:

$$B = \frac{2\Delta t}{h}\tilde{B}, \qquad \tilde{B}_{1,1} = 1, \quad \tilde{B}_{m,2} = 1,$$

and all other entries of $\tilde{B}$ equal to zero. The stability condition for the forward Euler scheme requires $r \leq 1/2$. To generate data, we simulate 200 trajectories using above discretization. For each trajectory, we sample the initial condition from a Gaussian distribution $X_0 \sim \mathcal{N}(0, I_m)$, apply i.i.d. uniform boundary controls $U_n \sim \text{Unif}([-u_{\max}, u_{\max}]^2)$ with $u_{\max} = 0.1$ independent of state, and record pairs (X_n, U_n, X_{n+1}) over $T = 200$ steps.

For ancestor discovery, we model the distributions of $\Delta x_{1,n}$ and $\Delta x_{10,n}$ individually where $\Delta x_{m,n} := x_{m,n+1} - x_{m,n}$, and find their stochastic ancestors from the full ancestor set $\{x_{1,n}, ..., x_{m,n}, u_{1,n}, u_{2,n}\}$. We use a FNN with four hidden layers of 64 neurons each and SiLU activation function to parameterize the velocity field, with batch size and number of epochs set to $1,000$. We use 5,000 distilled samples and the RBF kernel with lengthscales 1.0 for $x_i, i = 1, ..., 10$, u_1, u_2, and z.

As discussed in the main text, we evaluate control by replacing the boundary-node dynamics with data-driven surrogates inside MPC, while interior nodes keep the known finite-difference update. In this case, we only use the first 10 trajectories to form the training data and train two conditional flow-matching models on pruned ancestor sets, yielding boundary maps $\Delta x_{1,n} = G_1(x_{1,n}, x_{2,n}, u_{1,n}, z_{1,n})$ and $\Delta x_{10,n} = G_{10}(x_{9,n}, x_{10,n}, u_{2,n}, z_{10,n})$ where G_i denotes the time-1 flow generated by integrating the learned velocity field. Because repeated ODE integration is too costly for online MPC, we distill each teacher into a FNN by regressing the student output to the teacher map with MSE loss. Each student is a three-layer MLP (128 hidden units, SiLU activation function), trained for 20,000 steps with Adam (1e-3 learning rate) using mini-batches of 512 synthetic data pairs labeled by the teacher. Wall-clock MPC comparisons were run on a Mac Studio with an Apple M3 Ultra chip.

E.3.4.1 Details of the MPC setup

Recall that in this example, the control objective is to regulate the temperature profile to zero under a quadratic cost. Over $T = 100$ steps, $n = 0, ..., T-1$, we solve a finite-horizon stochastic optimal control problem using a receding horizon strategy. Specifically, given the current state X_n, we minimize a scenario-averaged finite-horizon quadratic cost over $U_{n:n+H-1}$:

$$\min_{U_n,...,U_{n+H-1}} \frac{1}{S} \sum_{s=1}^{S} \sum_{k=0}^{H-1} \left(\left\| X_{n+k}^{(s)} \right\|_Q^2 + \|U_{n+k}\|_R^2 \right) + \left\| X_{n+H}^{(s)} \right\|_Q^2, \tag{E.18}$$

where $s = 1, \ldots, S$ indexes the scenarios, and $\|v\|_M^2 := v^\top M v$. Here, $Q = I_m \in \mathbb{R}^{m \times m}$ penalizes deviation of the temperature field from zero, and $R = \rho I_2 \in \mathbb{R}^{2 \times 2}$

penalizes control effort, with $\rho = 0.1$. Each U_{n+k} is clipped to $[-0.1, 0.1]^2$ at each step. Samples $X_{n+k}^{(s)}, s = 1, \ldots, S$ are drawn from $X_{n+k} \mid X_{n+k-1}, U_{n+k-1}$. The expectation over the stochastic dynamics is approximated by averaging over S independent scenarios. Only the first control U_n^* is applied to the true system, after which the optimization is resolved at $n + 1$.

We assume that inside the MPC rollout, the update functions for the two boundary nodes, Δx_1 and Δx_{10}, are unknown and must be learned from data. The dynamics of the remaining interior nodes Δx_i, $i = 2, \ldots, m - 1$, are assumed known exactly. The learned update is expressed as a stochastic generative model. We compare three controllers, all targeting the same objective (E.18):

1. **MPC with known dynamics** as reference.
2. **MPC with full ancestors**: The dynamics of Δx_1 and Δx_{10} are modeled conditioned on the full ancestor set $\{x_{1,n}, ..., x_{m,n}, u_{1,n}, u_{2,n}\}$.
3. **MPC with pruned ancestors**: The dynamics of Δx_1 and Δx_{10} are modeled conditioned on the pruned ancestor sets ($\{x_1, x_2, u_1\}$ for Δx_1 and $\{x_9, x_{10}, u_2\}$ for Δx_{10}).

All three cases differ only in the learned boundary updates used inside the rollout.

The optimization problem (E.18) is solved using the L-BFGS algorithm [191] with the maximum iterations set to 300. The control sequence is warm-started at each time step by shifting the previous solution. We solve the MPC optimization over a planning horizon of $H = 20$ steps with $S = 64$ Monte Carlo scenarios, and evaluate over 50 independent trials.

E.3.5 Discovering state and control dependence in the Lunar Lander control problem

We consider the `LunarLanderContinuous-v3` environment from Gymnasium [334] (version 1.2.3). The continuous action $a = (a_1, a_2) \in [-1, 1]^2$ controls the main engine throttle and lateral booster throttle, respectively.

The dynamics decompose into two stages at each time step: (1) a thrust stage, in which the engines apply instantaneous velocity impulses to the lander, perturbing (v_x, v_y, ω), and (2) a Newtonian stage, which integrates the equations of motion–updating positions and angles and handling contact dynamics–deterministically

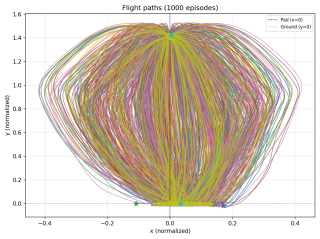


Figure E.2: Lunar Lander control problem: Controlled flight paths from 1,000 episodes using the SAC agent trained in the surrogate environment with the pruned ancestor sets.

given the post-impulse velocities. We use the default environment configuration: wind and turbulence are disabled, so the only stochasticity enters through the engine impulse dispersion. The terrain is procedurally generated at each episode reset, the helipad is centered horizontally, and the lander is initialized near the top center of the viewport with small random perturbations to position, velocity, and angle. No modifications are made to the reward function, termination conditions, or observation normalization.

Data are collected under a uniform random policy with actions drawn independently as $a \sim \mathrm{Unif}([-1,1]^2)$ at each step. We simulate 5,000 independent episodes of at most 1,000 steps, terminating early upon crash, successful landing, or the lander leaving the horizontal viewport bounds. The velocity increments $\Delta v_x, \Delta v_y, \Delta\omega$ are computed as the difference in the lander's linear and angular velocities immediately before and after the engine impulses are applied, prior to the Newtonian integration step. Each transition contributes one record to the dataset, consisting of a 10-dimensional input vector

$$X = [x, y, v_x, v_y, \theta, \omega, l_l, l_r, a_1, a_2]$$

concatenating the full 8-dimensional state observed before the step with the 2-dimensional action, together with the three scalar thrust-stage outputs $\Delta v_x, \Delta v_y, \Delta\omega$.

The ancestor sets of $\Delta v_x, \Delta v_y, \Delta\omega$ are discovered separately from the full candidate set

$$\{x, y, v_x, v_y, \theta, \omega, l_l, l_r, a_1, a_2\}.$$

For each target, the velocity field is parameterized by a FNN with four hidden layers of 64 neurons and SiLU activation function, trained with batch size 1,000 for 1,000 epochs. Structure discovery uses 8,000 distilled samples and a RBF kernel with lengthscale 1.0 for all state, control, and latent variables.

E.3.5.1 Details of the downstream control setup

The control objective is to land the lander softly on the helipad. The step reward is

$$r_n = \Delta S_n - 0.30\, m_n - 0.03\, s_n,$$

where $\Delta S_n = S_n - S_{n-1}$ is the change in the shaping potential

$$S_n = -100\sqrt{x_n^2 + y_n^2} - 100\sqrt{v_{x,n}^2 + v_{y,n}^2} - 100|\theta_n| + 10\, l_{l,n} + 10\, l_{r,n},$$

$m_n \in [0, 1]$ and $s_n \in [0, 1]$ are the main and side engine throttle magnitudes at step n (nonlinear functions of $a_{1,n}$ and $a_{2,n}$, respectively), and $l_{l,n}, l_{r,n} \in \{0, 1\}$ indicate leg ground contact. A terminal bonus of +100 is awarded when the lander comes to rest on the ground (the rigid body becomes inactive), and −100 when the lander hull contacts the ground or exits the horizontal viewport bounds. An episode is considered successful if the cumulative return exceeds 200. We refer to [334] for the full specification.

As in the pendulum example, we evaluate the discovered structure by training a control policy in the surrogate environment and transferring it to the true environment without further training. The thrust stage is replaced by flow matching surrogates conditioned on the pruned ancestor sets:

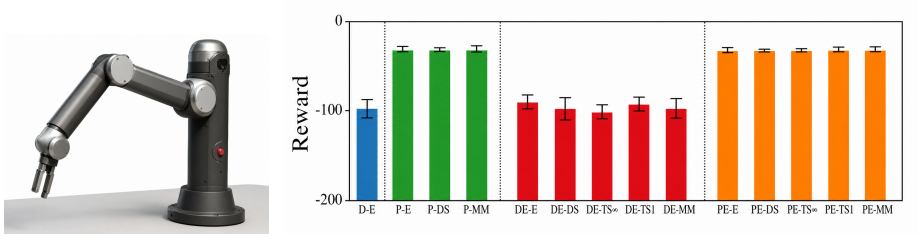
$$\Delta v_x = G_{v_x}(\theta, a_1, a_2, z_x), \quad \Delta v_y = G_{v_y}(\theta, a_1, z_y), \quad \Delta\omega = G_\omega(a_2, z_\omega),$$

where $z_x, z_y, z_\omega \sim \mathcal{N}(0, 1)$ are independent latent variables. The Newtonian stage remains identical to the true environment.

Policies are trained with SAC implemented in Stable-Baselines3 [269], using the tuned hyperparameters from RL Baselines3 Zoo [268] (version 2.8.0) for `LunarLanderContinuous-v3`. Policy weights are updated exclusively using interactions with the surrogate. This protocol directly targets transfer performance and is applied consistently across all three training regimes–(1) the true environment, (2) the surrogate with the full ancestor set, and (3) the surrogate with the pruned ancestor set–ensuring a fair comparison. Controlled flight paths from 1,000 episodes under the pruned surrogate are shown in Figure E.2.

E.3.5.2 Discussion of the result

The discovered ancestor sets may not be complete: dependencies with small contributions to the total variance may be pruned by the method, and the true dynamics



(a) (b)

Figure E.3: The 7-DOF reacher problem. (a) A schematic illustration of a robotic arm. (b) Performance comparison of the model-based RL problem between probabilistic and deterministic dynamics models, adapted from [66]. Higher rewards indicate better performance.

may exhibit additional weak dependencies not captured in the discovered graph. Recovering such dependencies would require substantially more data or strong parametric assumptions, conditions that rarely hold in practice. With finite data, pruning weak dependencies is statistically justified and practically beneficial, as it yields a lower-dimensional surrogate that is easier to fit accurately. Nevertheless, the surrogate trained on the pruned ancestor sets transfers successfully to the true environment, achieving 1,000/1,000 successful landings–illustrating a general principle: a parsimonious surrogate that captures the dominant structure may outperform a higher-dimensional one, as the latter can introduce unnecessary complexity that harms rather than helps policy transfer.

The marginally lower success rate of the unpruned surrogate (991/1,000 versus 1,000/1,000) may partly reflect the inclusion of spurious ancestors–in particular, the leg-contact indicators l_l, l_r, which are binary and physically unrelated to the instantaneous velocity changes induced by engine thrust, since the thrust stage acts before the Newtonian integration and is therefore independent of ground contact. Conditioning on such irrelevant discrete variables can introduce noise into the learned surrogate. That said, the difference is small and may equally reflect run-to-run variability from the random seed or finite training data.

E.3.6 Discovering state dependence for probabilistic surrogate modeling in model-based reinforcement learning

This reacher problem involves a simulated PR2 robot arm that must move its end-effector to a randomly placed target in 3D space (Figure E.3(a)) [66]. The true transition dynamics are deterministic, governed by MuJoCo's rigid body simulation [331]. Following the probabilistic ensembles with trajectory sampling (PETS)

framework introduced in [66], however, we model the transition dynamics probabilistically as $P(s_{t+1} \mid s_t, a_t)$, where stochasticity represents epistemic uncertainty from finite data rather than inherent randomness in the physical system. In PETS, this learned probabilistic model is used inside MPC to evaluate candidate action sequences before executing actions in the true environment. Figure E.3(b), adapted from [66], shows that probabilistic learned dynamics models can outperform deterministic learned dynamics models on this task, motivating the use of a probabilistic surrogate even though the simulator itself is deterministic. This is precisely the setting where distributional structure discovery is relevant: the object of interest is not stochasticity in the true simulator, but the input structure of the learned conditional distribution used for planning.

The discovery target in this example is the conditional transition surrogate, not the reward function or the MPC planning objective. As in the main text, we consider the 21 state-action inputs (s_n, a_n) as candidate parents for the 14 state-increment outputs $\Delta s_n := s_{n+1} - s_n$. We used the publicly released PETS codebase [66] for the 7-DOF Reacher environment and data-generation protocol. Data used for the ancestor discovery were generated by rolling out the reacher environment under uniformly random actions for 100 independent rollouts. At the start of each rollout, the environment was reset with a deterministic seed offset by rollout index; the target position was sampled by adding independent Gaussian noise with standard deviation 0.1 to the target coordinates. Each rollout lasted 150 environment steps, yielding 15,000 transition samples in total. At each step, a 7-dimensional action was sampled from the environment action space, bounded by the MuJoCo motor limits $[-20, 20]$, and the simulator advanced with frame skip 2 and timestep 0.01. The raw simulator observation is 17-dimensional, consisting of the 7 joint angles, the 3 target coordinates, and the 7 joint angular velocities. For structure discovery, we restrict the candidate transition inputs to the 14 arm state variables and the 7 actions; the target location is recorded and used by the task cost, but is not treated as a candidate parent in the transition graph. For every transition, the dataset stores the current raw observation, action, next raw observation, reward, goal position, and observation change.

We discover ancestors for the transition increments $(\Delta\theta_1, \ldots, \Delta\theta_7, \Delta\omega_1, \ldots, \Delta\omega_7)$ from the full candidate input set $\{\theta_i, \omega_i, a_i : i = 1, \ldots, 7\}$. Specifically, we fit one flow matching model for the joint conditional distribution of the 14-dimensional state increment given the 21 state-action inputs. The velocity field is parameterized

by a FNN with four hidden layers of 128 neurons and SiLU activation function, trained with batch size 1,000 for 1,000 epochs. In the second stage, we use 2,000 distilled samples and an RBF kernel with lengthscale 10.0 for both the state-action and the random variables. We adopt a conservative graph-construction strategy by taking the union of the discovered ancestor sets across all 14 output coordinates: a variable is retained if it is identified as an ancestor of any $\Delta\theta_i$ or $\Delta\omega_i$. We repeat the experiment five times independently to assess run-to-run variability.

E.3.6.1 The evaluation setup

We evaluate the discovered graphs within the original PETS model-based RL pipeline. All reacher experiments use the publicly released PETS implementation and experimental protocol of [66]; we refer readers to that work for details of the model-based RL loop, including the probabilistic dynamics model, trajectory sampling scheme, MPC planner, and cross-entropy method (CEM) optimization.

In all comparisons, we use the standard PETS probabilistic dynamics model and change only the input coordinates supplied to it. The full baseline receives all 21 state-action inputs, whereas the pruned models receive only the variables retained by the discovered graphs. The output space, planning objective, cost function, action bounds, data collection schedule, and MPC/CEM procedure are kept fixed across all models. Thus, the comparison isolates the effect of changing the input structure of the probabilistic transition surrogate. For visualization in Figure 6.4(d), we increase the number of trials from 100 to 150 and plot the average over three independent experiments of the best return achieved so far.

E.3.6.2 Discussion of the result

We close this example with several specific remarks. First, the discovered graphs are data-dependent: the results reported in this work are based on 100 random-control rollouts used for ancestor discovery, and a different dataset may yield different estimated graphs. Second, PETS operates as an iterative loop: starting from random-control data, a probabilistic dynamics model is fitted and used for MPC planning; the resulting trajectories are then added to the dataset, and the model is refined. As this loop progresses, the data distribution shifts from random exploration toward task-relevant, model-guided trajectories.

Consequently, the effective input structure of the learned probabilistic surrogate may also change across PETS iterations. Variables that appear irrelevant under random

controls may become relevant in task-critical regions of the state-action space, or variables retained under broad random exploration may become unnecessary once the data concentrate near successful trajectories. A natural extension of the proposed approach is therefore to rediscover the ancestor structure periodically as the dataset is updated, allowing the graph to be renewed alongside the dynamics model. This would yield a fully adaptive pipeline in which structure discovery and model refinement are interleaved, potentially improving both interpretability and efficiency in high-dimensional model-based RL.

E.3.7 The economic data example

Before constructing the dataset, we synchronize the oil-price and GPR time series by retaining only dates for which both quantities are available. Since oil prices are reported only on trading days while the GPR index is available daily, all non-trading days are removed from the GPR series. Dates containing missing values or extreme anomalous observations (e.g., negative prices) in either variable are also discarded. The resulting dataset therefore consists of a filtered sequence of aligned observation dates. The oil-price return is then defined on this filtered sequence as

$$r_n = \log(\mathrm{Oil}_{n+1}) - \log(\mathrm{Oil}_n), \tag{E.19}$$

where Oil_n denotes the oil price at the nth retained observation date in the filtered dataset. Consequently, the return is computed between consecutive retained observations rather than strictly between consecutive calendar days.

We note that the selected lag should not be interpreted as the unique physical response time of the oil market to geopolitical risk. The lag candidates are user-defined coarse representatives of delayed information channels. Changes in the dominant surviving lag instead indicate structural changes in the dependency organization between geopolitical risk and oil prices across different market regimes.

In addition to three approximate-10-year time periods analyzed in the main text, we also consider eight five-year time periods (the start and end year are inclusive): 1987-1991, 1992-1996, 1997-2001, 2002-2006, 2007-2011, 2012-2016, 2017-2021, 2022-2026/04/20, and employ the proposed method to analyze the structure of the transition kernel. Specifically, we focus on the variable with the largest surviving contribution score in modeling the dynamics in (6.3), defined as the variable pruned the last by the proposed method. Each time period contains approximately the same number of data points.

We find that the one-day lag is found as the largest surviving variable in 1987-1991, 1992-1996, 2002-2006, and 2022-2026/04/20, the 10-day lag in 1997-2001 and 2007-2011, the 5-day lag in 2012-2016, and the 20-day lag in 2017-2021. Observe that the dominant surviving representative

1. the one-day lag in 1987-1991 and 1992-1996,
2. changes to the 10-day lag in 1997-2001,
3. changes back to one-day lag in 2002-2006,
4. and changes again to 10-day lag in 2007-2011.

This result is consistent with the longer-period analysis presented in the main text. The periods 1997-2001 and 2007-2011 coincide with major geopolitical and economic disruptions, including the 9/11 attacks and the 2008 financial crisis, which may have altered the observed dependency structure between geopolitical risk (GPR) index and oil returns.

We also observe that the 2017-2021 period yields the 20-day lag as the dominant surviving representative. This period coincides with the COVID-19 crisis beginning in early 2020, during which oil prices experienced substantial market shocks. Similarly, the 2012−2016 period yields the 5-day lag as the dominant surviving representative, which may be related to structural changes in oil-market dynamics associated with shale supply expansion and OPEC policy shifts. Overall, these results suggest that the delayed dependence structure between geopolitical risk and oil returns is not stationary across time, and that the proposed method can be used to identify and localize structural changes in stochastic dependency organization directly from observational economic data.

In this example, for all cases, we use a FNN with four hidden layers of 64 neurons each and SiLU activation function to parameterize the velocity field, and train it with batch size 1,000 and 2,000 epochs. We then use 10,000 distilled samples and the RBF kernel with lengthscale 5.0 for both the state and random variables. The nugget is set to $\exp(1.0)$. We only prune GPR lags.

E.4 Additional examples

In this section, we provide additional examples supporting the stochastic ancestor-discovery framework. The first and third examples have known reference dependency structures and therefore serve as verification tests. The second example

illustrates memory discovery in a coarse-grained system with hidden momentum, where validation is based on the sufficiency of the discovered lag variables for reproducing the observed transition law.

E.4.1 Recovering the computational graph of a stochastic differential equation

This example serves as a verification test for the proposed method since both the conditional distributions and the underlying computational graph are known. Specifically, we consider the stochastic Kraichnan–Orszag (KO) system [346]:

$$dx_1 = x_1 x_3\, dt + \sigma_1\, dW_1, \tag{E.20a}$$

$$dx_2 = -x_2 x_3\, dt + \sigma_2\, dW_2, \tag{E.20b}$$

$$dx_3 = (-x_1^2 + x_2^2)\, dt + \sigma_3\, dW_3, \tag{E.20c}$$

where W_i are independent Wiener processes. The system is simulated using the Euler–Maruyama method with randomly sampled initial conditions and noise parameters $\sigma_1 = \sigma_2 = \sigma_3 = 0.1$.

For dependence discovery, we consider the discretized update equations

$$x_{1,n+1} = x_{1,n} + x_{1,n}x_{3,n}\Delta t + \sigma_1\sqrt{\Delta t}\xi_1, \tag{E.21a}$$

$$x_{2,n+1} = x_{2,n} - x_{2,n}x_{3,n}\Delta t + \sigma_2\sqrt{\Delta t}\xi_2, \tag{E.21b}$$

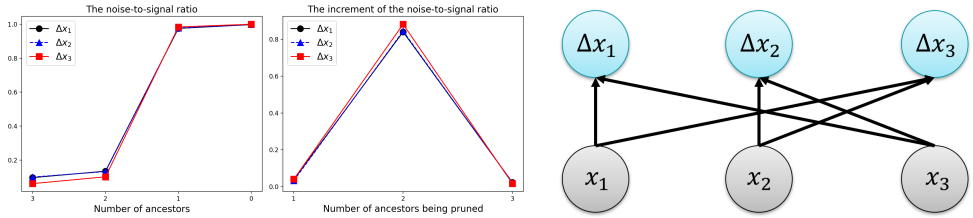
$$x_{3,n+1} = x_{3,n} + (-x_{1,n}^2 + x_{2,n}^2)\Delta t + \sigma_3\sqrt{\Delta t}\xi_3, \tag{E.21c}$$

and define $\Delta x_{i,n} := x_{i,n+1} - x_{i,n}$. The goal is to learn stochastic update functions

$$\Delta x_{i,n} \approx \Phi_i(x_{1,n}, x_{2,n}, x_{3,n}, \omega_i), \qquad i = 1, 2, 3.$$

Figure E.4 shows the pruning process and the recovered computational graph. To evaluate the accuracy of the learned generative models, we compute the Wasserstein-1 (W_1) distance between the learned conditional distributions and the exact distributions. The results are reported in Table E.2. The comparison shows that using the pruned ancestor set yields accuracy comparable to using all candidate variables, indicating that the proposed method successfully identifies the correct dependencies without degrading generative modeling performance.

We use $\Delta t = 0.001$. Initial conditions are sampled from $\mathcal{N}(0, 2^2 I_3)$ where I_3 denotes the 3×3 identity matrix, and the system is simulated with the Euler–Maruyama method. We generate 50 independent trajectories; each trajectory is run for 10,000 steps, yielding 500,000 state-increment pairs in total. We model the conditional

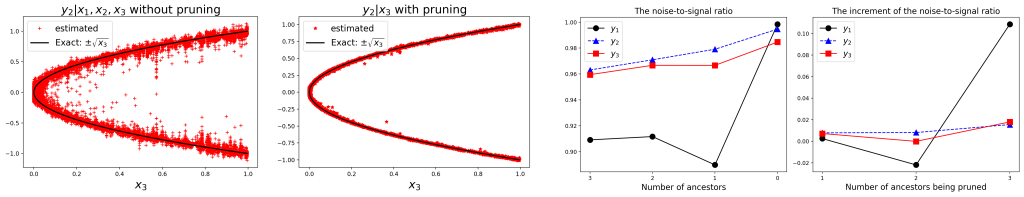


(a) (b)

Figure E.4: Stochastic KO system: (a) pruning process and (b) recovered computational graph.

	Φ_1	Φ_2	Φ_3
Using x_1, x_2, x_3 as ancestors ($\times 10^{-4}$)	1.15 ± 1.30	1.16 ± 0.60	1.59 ± 2.06
Using pruned variables as ancestors ($\times 10^{-4}$)	0.95 ± 0.82	0.92 ± 0.46	1.40 ± 1.84

Table E.2: The W_1 distance in average between the learned generative update model and the exact conditional distribution. We randomly select $1,000$ test data points. For each one of them, W_1-distance is estimated based on $10,000$ samples.



(a) (b)

Figure E.5: The algebraic system with hidden conditional dependence: (a) conditional generative modeling of $y_2 \mid x_1, x_2, x_3$ (without pruning) and $y_2 \mid x_3$ (with pruning) and (b) pruning process.

distributions of the increments $\Delta x_{k,n}, k = 1, 2, 3$, separately, using the current state $(x_{1,n}, x_{2,n}, x_{3,n})$ as candidate covariates. We use a FNN with two hidden layers of 64 neurons each and SiLU activation function to parameterize the velocity field of the flow matching model, with batch size and number of epochs both set to 5,000 and 500, respectively. In the second stage, we use $N = 5{,}000$ distilled samples to fit the GP surrogate. We use the RBF kernel with lengthscale 1.0 for both the state and the random variables.

E.4.2 Algebraic system with hidden conditional dependence

Next, we consider an algebraic system in which dependence appears at the level of the conditional distribution rather than the conditional mean. This example illustrates the ability of the proposed method to detect distributional dependence

beyond the mean; see Figure E.5(a). Specifically, let

$$y_1 = x_1\omega, \qquad y_2 = \varepsilon\sqrt{x_3}, \qquad y_3 = \sqrt{1-\alpha(x_2)}\,\xi + \sqrt{\alpha(x_2)}\,\eta,$$

where $\alpha(x) = \sin(\pi x)$, $x_3 \geq 0$, $\omega, \xi \sim \mathcal{N}(0,1)$, $\eta \sim \text{Laplace}(0, 1/\sqrt{2})$, and ε is an independent Rademacher random variable. In this setting,

$$\begin{aligned}&\mathbb{E}[y_1 \mid x_1, x_2, x_3] = \mathbb{E}[y_2 \mid x_1, x_2, x_3] = \mathbb{E}[y_3 \mid x_1, x_2, x_3] = 0,\\ &\text{Var}[y_3 \mid x_1, x_2, x_3] = 1.\end{aligned}$$

Thus the dependencies of y_1, y_2, y_3 on x_1, x_2, x_3 are not identified in their conditional means. Moreover, the dependence of y_3 on x_2 is also not present in the conditional variance, since it appears only through the shape of the conditional distribution. We apply the proposed method to discover the ancestors of y_1, y_2, and y_3 while treating $\omega, \varepsilon, \xi, \eta$ as unobserved variables. The pruning process is shown in Figure E.5(b). The method correctly recovers $\mathcal{A}(y_1) = \{x_1\}$, $\mathcal{A}(y_2) = \{x_3\}$, and $\mathcal{A}(y_3) = \{x_2\}$. Notably, the dependence of y_3 on x_2 is manifested only through the changing relative weights of the Gaussian and Laplace noise components, and hence through tail behavior rather than through either the conditional mean or variance. This demonstrates that the proposed framework detects distributional structure beyond what mean-based or second-moment methods can access.

In this example, 1,000 data points are used for stochastic ancestor discovery for y_1 and y_2, while 10,000 data points are used for y_3. More data points are required for discovering the ancestors of y_3 accurately, compared to y_1 and y_2, because the dependence manifests only through the tail behavior rather than through the mean or variance, as discussed in the main text. In particular, data of x_1 and x_2 are sampled from $\text{Unif}([-0.5, 0.5])$ and $\text{Unif}([0, 1])$, respectively, and data y_1 and y_3 are computed according to the exact formula using data of x_1 and x_2 with $\alpha(x) := \sin(\pi x)$, while data of y_2 are sampled from $\text{Unif}([-1, 1])$ and data x_3 are computed accordingly using data of y_2.

For ancestor discovery, we model the distributions of y_1, y_2, y_3 individually by employing the proposed method three times independently. For each y_k, $k = 1, 2, 3$, we use a FNN with four hidden layers of 64 neurons each and rectified linear unit (ReLU) activation function to parameterize the velocity field, with batch size and number of epochs set to 1,000 and 5,000, respectively. In the second stage, we use 5,000 distilled samples and the RBF kernel with lengthscale 5.0 for x_k, $k = 1, 2, 3$. For y_1 and y_3, we use lengthscale 1.0 for the random variable z and nugget 0.1,

while for y_2, we use lengthscale 0.1 for z and nugget 1.0. This choice is made based on the observation that the conditional distribution of $y_2 \mid x_1, x_2, x_3$ is a two-point distribution concentrated at $\pm\sqrt{x_3}$. This discrete structure is poorly suited to both flow matching and GP regression, motivating the adjusted kernel hyperparameters.

E.4.3 Discovering memory dependence in stochastic dynamics

We consider a coarse-grained stochastic dynamical system in which the momentum is hidden, so the observed position dynamics exhibit memory. The underlying system follows a Langevin equation for a double-well potential

$$\frac{dx}{dt} = p, \tag{E.22a}$$

$$\frac{dp}{dt} = -V'(x) - \gamma p + \sigma\xi(t), \tag{E.22b}$$

where

$$V(x) = V_0 \left(1 - \left(\frac{x}{x_0}\right)^2\right)^2$$

is a symmetric double-well potential with two metastable states [194, 383].

In this example, we assume that only the position variable x is observed while the momentum variable p is hidden. The resulting coarse-grained dynamics of x exhibit memory effects. We therefore model the dynamics using a stochastic update rule

$$x_{n+1} = x_n + \Phi(x_n, x_{n-1}, \ldots, x_{n-m}, \omega_n), \tag{E.23}$$

where the goal is to identify the relevant memory states. Applying the proposed method allows us to determine which past states influence the transition dynamics. Figure E.6 shows the pruning process, the discovered dependency structure, and trajectories generated using the learned update model. The results show that the method identifies a compact set of lagged positions sufficient to reproduce the observed coarse-grained transition law.

We follow the setting in [383] and set $\gamma = 1, \sigma = \sqrt{2}, V_0 = 4, x_0 = 1$. We use the Euler–Maruyama method with $\Delta t = 0.001$ to simulate the system. We generate 5,000 trajectories from zero initial condition; each trajectory is run for 320,000 steps and the last 160,000 steps are kept, from which data are saved every 200 steps. We take the first 100 data points to construct the training data for discovering the memory dependence with $m = 9$. We use a FNN with two hidden layers of 64 neurons each and hyperbolic tangent activation function to parameterize the velocity

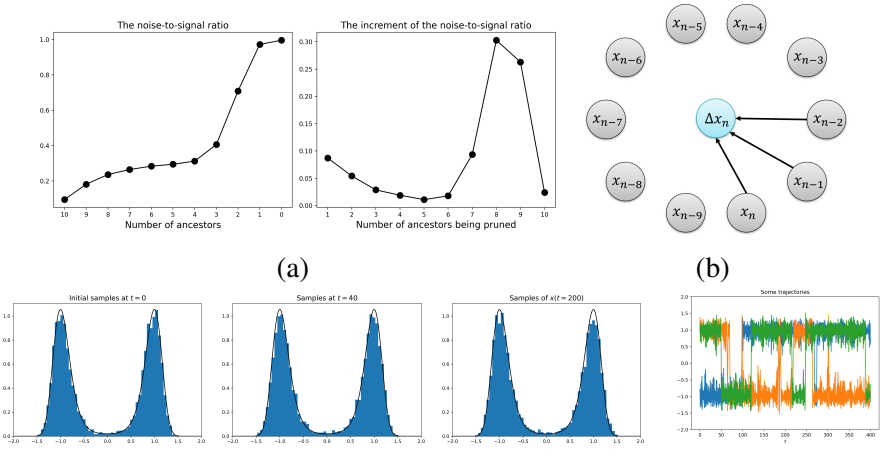


Figure E.6: The coarse-grained transition dynamics with missing variables: (a) pruning process, (b) discovered graph, (c) generated samples of x at different times from the learned conditional law, and (d) generated trajectories from the learned conditioned law.

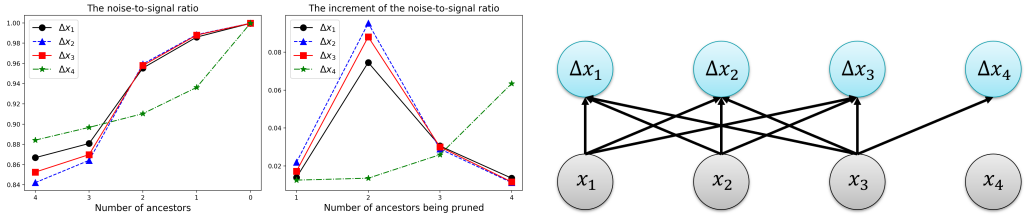


Figure E.7: The stochastic chemical system: (a) pruning process and (b) discovered graph.

field of the flow matching model, with batch size and number of epochs both set to 5,000 and 1,000, respectively. In the second stage, we use $N = 5{,}000$ distilled samples to fit the GP surrogate. We use the RBF kernel with lengthscales 1.0 and 10.0 for the state and the random variables, respectively.

E.4.4 Discovering state dependence in a stochastic chemical system

This experiment illustrates the ability of the proposed method to discover state dependence in stochastic reaction networks from data generated by exact stochastic simulations. We consider the classical Michaelis–Menten enzymatic reaction mechanism involving four chemical species [146]:

S_1 : a substrate, S_2 : an enzyme, S_3 : a complex, S_4 : a product.

The reactions are

$$S_1 + S_2 \xrightarrow{c_1} S_3, \tag{E.24a}$$

$$S_3 \xrightarrow{c_2} S_1 + S_2, \tag{E.24b}$$

$$S_3 \xrightarrow{c_3} S_4 + S_2, \tag{E.24c}$$

where c_1, c_2, c_3 are the rate constants. Let $X(t) = [x_1(t), x_2(t), x_3(t), x_4(t)]$ denote the state vector where $x_i(t)$ represents the number of molecules of species S_i at time t. The stochastic reaction dynamics are simulated using Gillespie's stochastic simulation algorithm (SSA) [146]. At each step, a reaction is selected with probability proportional to its propensity function and the state is updated according to the corresponding stoichiometric change.

While SSA provides an exact simulation of the stochastic chemical kinetics, it does not directly yield a fixed-time update rule. To analyze the dependence structure using the proposed framework, we consider the discrete-time increments

$$\Delta x_{i,n} = x_{i,n+1} - x_{i,n}, \qquad i = 1, \ldots, 4,$$

with a fixed observation interval $\tau = 0.1$. The dataset consists of pairs $(X_n, \Delta X_n)$ obtained from SSA trajectories with randomly sampled initial conditions.

Using these data, we train conditional generative models for the stochastic updates

$$\Delta x_{i,n} \approx \Phi_i(x_{1,n}, x_{2,n}, x_{3,n}, x_{4,n}, \omega_i), \qquad i = 1, \ldots, 4,$$

and apply the proposed ancestor discovery procedure to identify the state variables that influence each update.

The pruning results and the discovered computational graph are shown in Figure E.7. The method successfully recovers the correct dependency structure implied by the reaction network. Since the reaction propensities depend on x_1, x_2 and x_3, while x_4 does not enter any propensity, the reference dependency structure excludes x_4 as an ancestor of the fixed-time increments. The recovered graph is consistent with this structure.

We follow the parameter setting, initializations, and time span $T = 50$ from [146], except that we set $c_2 = 0.1$, and simulate the Michaelis–Menten system using SSA independently for 2,000 trajectories. We use a FNN with two hidden layers of 64 neurons each and ReLU activation function to parameterize the velocity field of the flow matching model, with batch size and number of epochs both set to 5,000 and

1,000, respectively. In the second stage, we use $N = 5{,}000$ distilled samples to fit the GP surrogate. We use the RBF kernel with lengthscale 1.0 for both the state and the random variables, respectively.

E.4.5 Hidden fast variables do not obscure local slow-variable dependencies

We consider the multiscale Lorenz 96 system [22, 201], which couples m slow variables x_j with J fast variables $y_{k,j}$ for every slow variable. The slow variables satisfy

$$\frac{dx_j}{dt} = (x_{j+1} - x_{j-2})x_{j-1} - x_j + F - \frac{hc}{b}\sum_{k=1}^{J} y_{k,j}, \qquad j = 1, \ldots, m, \tag{E.25}$$

where cyclic indexing is imposed by $x_{j+m} = x_j$. In line with Section 6.4.2, the fast variables are treated as unobserved nuisance variables. Thus, we are interested in the dynamic dependency of each variable in $q_{\mathcal{I}} = (x_1, \ldots, x_m)$ on others. For $m = 10$ and $J = 10$ in a weak-coupling regime with $h = 0.1$, the method recovers $\mathcal{A}(\Delta x_j) = \{x_{j-2}, x_{j-1}, x_j, x_{j+1}\}$, consistent with the explicit functional structure of the slow-variable equation. The influence of the unobserved fast variables is encoded in the learned conditional law of Δx_j given the observed slow variables, rather than being spuriously attributed to distant slow variables. To assess scalability, we vary $m = 10, 20, \ldots, 200$. The method recovers the correct ancestor sets across all values of m, with a log-log slope of 1.81 for computational time versus m with a fixed number of data N, consistent with the computational cost of the discovery procedure ($O(mN^3 + m^2N^2)$). In these experiments, x_j is provided as prior knowledge; the method identifies the remaining ancestors from $\{x_i : i \neq j\}$.

E.4.6 Probabilistic model-based control tolerates substantial input pruning.

We apply the proposed method to the 7-DOF reacher problem [66], a high-dimensional model-based RL benchmark in which a simulated PR2 robot arm moves its end-effector to a target in 3D space. In the notation of Section 6.4.2, the observed state is $q(t) = q_{\mathcal{I}}(t) = (\theta(t), \omega(t))$, consisting of 7 joint angles and 7 angular velocities, and the control is $u(t) \in \mathbb{R}^7$, consisting of joint torques. Although the MuJoCo simulator dynamics are deterministic, the probabilistic ensembles with trajectory sampling (PETS) framework [66] models the transition as random using $P(q_{t+\Delta t} \mid q_t, u_t)$, where the stochasticity represents epistemic uncertainty from finite data. Thus, the object of interest is the inputs of the learned surrogate for the transition law. The target variables are the 14 state increments Δq_i, $i = 1, \ldots, 14$, and the candidate covariates are the 21 state-action inputs (q_t, u_t). We apply the proposed

method to obtain ancestor sets $\mathcal{A}(\Delta q_i)$ separately for each output coordinate, and define the graph for a run by the conservative union $\mathcal{A}(\Delta q) := \bigcup_{i=1}^{14} \mathcal{A}(\Delta q_i)$. This retains any variable identified as an ancestor of at least one transition coordinate. Across five independent runs, the method consistently prunes $\theta_1, \theta_2, \theta_5, \theta_7$. Run-to-run variability occurs for $\omega_1, \omega_2, \omega_7$, which are pruned in some runs but retained in others. As shown in Figure 6.4(c), the least sparse graph prunes only $\{\theta_1, \theta_2, \theta_5, \theta_7\}$, while the most sparse graph additionally prunes $\{\omega_1, \omega_2, \omega_7\}$. To evaluate the discovered structure, we use the PETS probabilistic model-based RL framework [66] and modify only the input coordinates of its learned dynamics model according to the discovered graphs. The output space, planning objective, data collection protocol, and MPC planner are kept fixed. Both the least and most sparse surrogates achieve performance comparable to the full unpruned PETS baseline, as shown in Figure 6.4(d), where return denotes the cumulative task reward and the plotted curve reports the mean best return achieved so far across the trials. This indicates that the consistently pruned joint angles are non-critical for this control task, and that the marginal angular velocities are not essential for the reported control performance even when some inputs are removed.

Appendix F

SUPPLEMENTARY MATERIALS FOR CHAPTER 7: QAP BACKGROUND AND PRIMITIVES

F.1 Why is the QAP hard?

The QAP [168, 196] is a strongly NP-hard problem [112], meaning that not only does no known polynomial-time algorithm exist to solve the QAP (and it is unlikely that such an algorithm will ever be found unless $P = NP$), but the problem remains hard even if its numerical parameters remain bounded by a polynomial in the input size. Moreover, for the QAP constructing a constant-factor approximation algorithm is itself NP-hard. The QAP is widely recognized as computationally challenging, with instances becoming difficult even at moderate sizes (e.g., $n = 20$) [48]. Furthermore, any problem in NP (nondeterministic polynomial time) can be reduced to the QAP in polynomial time [112]. Thus, if we were able to solve the QAP efficiently (in polynomial time), we would be able to solve all NP problems efficiently, which would have profound implications for computational complexity theory. The QAP also encompasses several well-known problems in combinatorial optimization. For instance, by setting $F_{i,j} = 1$, $D_{j,i} = D_{i,j} \geq 0$, $D_{i,i} = 0$ and $C_{i,j} = 0$, the QAP simplifies to the Traveling Salesman Problem (TSP) [12], which is a special case of the QAP and is generally considered much easier to solve. Moreover, by setting $C = 0$ and allowing $D_{i,j} \in \{0, 1\}$, the QAP reduces to the Graph Matching Problem [195], a problem concerned with finding optimal node correspondences between two graphs. Finally, by setting both $F = 0$ and $D = 0$ the QAP simplifies to the linear assignment problem, which can be solved in polynomial time using the Hungarian Algorithm [176].

F.1.1 Why is the QAP important?

The optimization of logistics and supply chains, such as allocating resources to multiple sites during operations and determining inventory levels over time at various facilities, is inherently challenging due to the combinatorial scaling of search spaces and the dynamic evolution of states. These complexities often cause current algorithms to produce suboptimal solutions, with performance deteriorating significantly as problem size increases, potentially leading to undesirable outcomes ranging from increased operational costs to loss of life. In the context of supply

chain management, QAP is particularly important for strategically placing facilities to minimize the time required to move items through a complex network. The problem's dynamic and uncertain nature, coupled with evolving structural changes over time, further complicates the optimization process. Existing methods frequently fail to adapt to these changes, underscoring the need for algorithms that can dynamically generate solutions in response to shifting conditions. In supply chain optimization, moderate-sized problems can involve $O(10^6)$ parameters, making the simulation and optimization of these networks inefficient with current algorithms. Additionally, other applications of QAP include facility layout and location (e.g., airport design), data center optimization, and very-large-scale integrated circuit design. Developing an automated algorithm discovery framework would therefore represent a significant industry and operational breakthrough, enabling the dynamic generation of algorithms that adapt to changing conditions and achieve lower-cost solutions by effectively relocating items to critical areas. Such advancements would not only enhance the efficiency and resilience of logistics but also offer substantial benefits across various sectors reliant on complex supply chain and facility optimization.

F.1.2 Current methods for solving the QAP

Current methodologies for solving the QAP span exact algorithms, heuristic/metaheuristic methods, and hybrid/emerging approaches as detailed in surveys [67, 196].

Exact methods include systematic techniques such as Branch and Bound (B&B), Cutting Plane methods, Dynamic Programming (DP), and Integer Programming (IP), providing guarantees of optimality but with significant computational demands for general instances.

Heuristic algorithms offer efficient approximations suitable for larger problems. Prominent examples include Simulated Annealing (SA), Genetic Algorithms (GA), Tabu Search (TS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Iterated Local Search (ILS), and Greedy Randomized Adaptive Search Procedure (GRASP).

Hybrid and emerging approaches blend multiple methodologies to enhance performance. These include Memetic Algorithms (MA), Hybrid Genetic Algorithms (HGA), Parallel Algorithms, and novel techniques such as Quantum Algorithms and Machine Learning-assisted heuristics.

F.1.3 Exhaustive list of primitives for the QAP

F.1.3.1 Low-level primitives

Low-level primitives are tokens that do not directly solve the problem at hand, here the QAP, but can be chained to get an algorithm.
The following low-level primitives are implemented in Sec. 7.7.1:

- **Copy** (COPY): $x \mapsto x$
- **Linear Sum Assignment** (LSA): $x \mapsto \arg\min_\pi \langle x, \pi \rangle_F$
- **Negative** (NE): $x \mapsto -x$
- **Gradient** (GRAD): $x \mapsto \nabla \mathcal{L}(x)$
- **Early stopping** (STOP): No further computation, saves compute budget for higher reward.

Additionally, Sec. 7.7.1 uses operator primitives, whose outputs are determined by the current state and the supplied primitive P:

- **For loop** (usually 50 iterations) (FOR)(P): $x \mapsto P \circ \cdots \circ P(x)$
- **Residual update** (RU)(P): $x \mapsto x + P(x)$
- **Parallel uniform** (PU)(P): $x \mapsto \arg\min_{z_i} \{\mathcal{L}(z_i),\ z_i = P(y_i),\ y_i \sim \mathcal{U},\ i = 1, \ldots, 10\}$
- **Parallel 2-city swaps** (2SWAP)(P): $x \mapsto \arg\min_{z_{i,j}} \{\mathcal{L}(P(y_{i,j})) \mid z_{i,j} = P(y_{i,j}), y_{i,j} = \text{SWAP}(x, i, j)\}$, where $y_k^{i,j} = x_k$ if $k \neq i, j$, and $y_j^{i,j} = x_i$, $y_i^{i,j} = x_j$ is the permutation x, with indices i and j swapped

Finally, several primitives are considered for manual algorithm discovery 7.7.3:

- **Gibbs permutation average** (uses multiple permutations) (GIBBS): $(x_i)_{i=1}^k \mapsto \frac{\sum_i x_i e^{-\beta \mathcal{L}(x_i)}}{\sum_i e^{-\beta \mathcal{L}(x_i)}}$
- **Biasing tokens** (BIAS): Given a matrix T and a doubly stochastic matrix x_0, changes the loss to $\mathcal{L}_b(x) = \mathcal{L}(x) + \gamma \operatorname{Tr}(x_0^T T)$
- **Best of both** (BEST): $x \mapsto \arg\min_{i \in \{1,2\}} (\mathcal{L}(y_i))$ where $y_1 = LSA \circ NE(x)$ and $y_2 = LSA \circ GRAD(x)$

- **Parallel state with permutation** (PSP): $x, P \mapsto \arg\min\{\mathcal{L}[P(y_i)] \mid y_i = U_i x_i + (1 - U_i)x\}$ where x_i is a random uniform permutation, and $U_i \sim \mathcal{U}([0, 1])$

For clarity, these operator primitives can be read as higher-order procedures: they take a matrix-to-matrix primitive $P : \mathcal{M} \to \mathcal{M}$ as input and return a new matrix-to-matrix primitive. In pseudocode,

$RU[P](x)$: return $x + P(x)$,

$PU[P](x)$: sample $y_1, \ldots, y_{10} \sim \mathcal{U}$,
$z_i \leftarrow P(y_i), \quad i = 1, \ldots, 10$,
return the first z_i, in index order, with minimal $\mathcal{L}(z_i)$,

$2SWAP[P](x)$: for each pair $i < j$, set $y_{ij} \leftarrow \text{SWAP}(x, i, j)$,
$z_{ij} \leftarrow P(y_{ij})$,
return the first z_{ij}, in lexicographic order of (i, j),
with minimal $\mathcal{L}(z_{ij})$.

Thus $COPY$ acts as the neutral argument for these operator primitives: $2SWAP[COPY]$ is the ordinary best two-city-swap move, i.e., the best one-transposition move, while $RU[COPY]$ is the residual update $x \mapsto 2x$.

F.1.3.2 High-level primitives

The following algorithms are used to get the high-level primitives results in 7.7.2. Note that many of these primitives are rediscovered in 7.7.1.

- **Two-location optimal swaps** (2OPT): iteratively swaps two distinct indices i, j using the best possible swap, until no improvement is possible. Note that this is approximately achieved by $FOR(2SWAP(COPY))$ (with enough iterations).
- **Three-location optimal swaps** (3OPT): same as 2OPT, but a 3-location swap can change up to 3 indices, i, j, k. Note that this can be approximately achieved using $FOR(2SWAP(2SWAP(COPY)))$. In practice, we limit the number of explorations of 3-city swaps, as this can be prohibitively expensive.
- **Path reversal 2OPT** (P-2OPT): defining the path reversal such that $\boldsymbol{y}^{i,j} = \text{REVERSE}(\boldsymbol{x}, i, j)$, $y_k^{i,j} = x_{j+i-k}$ if $i \le k \le j$, else $y_k^{i,j} = x_k$, P-2OPT applies iteratively the best path reversal until no improvement.

- **Path reversal 3OPT** (P-3OPT) choose the best 3 indices, i, j, k, and iteratively applies two path reversals, REVERSE(REVERSE($\boldsymbol{x}, i, j$), j, k), until no improvement or maximum iteration number is reached.

- **Simulated annealing** (SA): This primitive performs m steps of simulated annealing using two-location flips on an initial permutation x. At step k, simulated annealing picks two random indices, i, j, and performs $\widetilde{\boldsymbol{x}}^k = \text{SWAP}(\boldsymbol{x}^k, i, j)$. With probability $\min\left(1, \exp\left(-\frac{\Delta\mathcal{L}}{T_k}\right)\right)$ where $\Delta\mathcal{L} = \mathcal{L}(\widetilde{\boldsymbol{x}}^k) - \mathcal{L}(\boldsymbol{x}^k)$, the move is accepted and $\boldsymbol{x}^{k+1} = \widetilde{\boldsymbol{x}}^k$; else $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$. The temperature follows a decreasing schedule $T_k = T_0(1-\epsilon)^k$. This approach probabilistically accepts worse solutions, allowing escape from local minima, until m iterations are completed.

- **Frank-Wolfe** (FW): Performs gradient descent in the space of doubly stochastic matrices. One iteration has the update $x^{k+1} = (1-\gamma_k)x^k + \gamma_k \,\text{LSA}(\text{GRAD}(x^k))$, where $\gamma_k = 2/(2+k)$. Variations include FW-LS, Frank-Wolfe followed by linear sum assignment, and FW-LSG, Frank-Wolfe followed by taking the gradient and applying linear sum assignment.

- **Orthogonal relaxation with QR decomposition** (OP). Following the relaxation strategy described in [287], OP performs gradient descent in the space of orthogonal matrices using the Procrustes solution [286, 287]. Defining $Q[x]$ as the Q factor of the QR-decomposition, the update rule is $x^{k+1} = Q[x^k - \gamma^k \,\text{GRAD}(x^k)]$, with the learning rate $\gamma^k = 0.5 * 0.95^k$.

- **Orthogonal relaxation with QR decomposition and momentum** (OC): More complex update rule for gradient descent with momentum in the orthogonal matrices space, see [286, 287].

Note that the space of permutation matrices is exactly the intersection of the doubly stochastic matrices and orthogonal matrices, justifying the relaxations of both FW and OC.

Table F.1: **Relative gap ($(\mathcal{L} - \min \mathcal{L})/\min \mathcal{L}$, in %) on QAPLIB Benchmark Instances** against Gurobi, Branch and Bound (BB) and Simulated Annealing (SA)

Problem	Best model	**CLP (ours)** (%)	SA (%)	BB (%)	Gurobi (%)
bur26a	CLP	0.0	1.0	5.1	7.7
bur26b	CLP	0.0	0.2	5.4	9.5

Continued on next page

Problem	Best model	**CLP (ours)** (%)	SA (%)	BB (%)	Gurobi (%)
bur26c	CLP	0.0	1.3	5.3	10.1
bur26d	CLP	0.0	0.5	6.1	12.1
bur26e	CLP	0.0	1.2	6.4	9.5
bur26f	CLP	0.0	0.4	7.8	7.5
bur26g	CLP	0.0	1.1	5.3	10.4
bur26h	CLP	0.0	1.1	6.3	9.1
chr25a	CLP	7.2	64.5	261.9	432.9
chr22a	CLP	1.8	11.9	71.8	140.7
chr22b	CLP	1.9	23.8	172.2	138.8
chr20a	CLP	4.5	24.2	161.5	379.6
chr20b	CLP	6.9	66.3	144.3	431.9
chr20c	CLP	0.0	87.6	346.4	737.6
chr18a	CLP	0.2	49.4	284.6	706.8
chr18b	CLP	0.0	10.0	25.3	227.8
chr15a	CLP	0.0	40.7	231.7	657.5
chr15b	CLP	0.0	49.9	350.4	688.9
chr15c	CLP	0.0	76.8	343.6	554.4
chr12a	CLP	0.0	26.2	40.9	516.4
chr12b	CLP	0.0	85.3	117.2	392.2
chr12c	CLP	0.0	59.2	28.9	238.9
els19	CLP	0.0	70.4	69.9	174.7
esc128	CLP	0.0	125.0	362.5	340.6
esc32e	CLP/SA	0.0	0.0	1400.0	3400.0
esc32g	CLP/SA	0.0	0.0	366.7	533.3
esc16a	CLP/SA	0.0	0.0	23.5	52.9
esc16b	CLP/SA	0.0	0.0	1.4	10.3
esc16c	CLP/SA	0.0	0.0	7.5	40.0
esc16d	CLP/SA	0.0	0.0	62.5	212.5
esc16e	CLP	0.0	7.1	35.7	157.1
esc16g	CLP	0.0	7.7	15.4	138.5
esc16h	CLP/SA	0.0	0.0	5.4	25.5
esc16i	CLP/SA	0.0	0.0	85.7	271.4
esc16j	CLP/SA	0.0	0.0	75.0	275.0
had20	CLP	0.0	0.1	7.0	9.7
had18	CLP	0.0	0.8	4.8	12.3
had16	CLP	0.0	0.8	5.6	12.7
had14	CLP/SA	0.0	0.0	4.3	18.6
had12	CLP	0.0	0.1	4.1	15.9
kra32	CLP	0.0	13.9	40.5	50.4
kra30a	CLP	0.0	14.2	35.9	48.6
kra30b	CLP	0.0	9.7	32.4	41.4
lipa90a	CLP	0.7	1.3	1.9	1.8
lipa90b	CLP	0.0	25.1	30.2	30.1
lipa80a	CLP	0.7	1.3	1.9	2.1
lipa80b	CLP/Gurobi	0.0	25.3	30.5	0.0
lipa70a	CLP	0.8	1.4	2.0	2.6
lipa70b	CLP/Gurobi	0.0	24.9	28.6	0.0
lipa60a	CLP	0.9	1.6	2.4	2.8
lipa60b	CLP/Gurobi	0.0	23.0	27.5	0.0
lipa50a	CLP	1.0	1.7	2.7	2.9
lipa50b	CLP/Gurobi	0.0	21.7	26.9	0.0
lipa40a	CLP	1.1	2.0	3.2	3.6

Continued on next page

Problem	Best model	**CLP (ours)** (%)	SA (%)	BB (%)	Gurobi (%)
lipa40b	CLP/Gurobi	0.0	21.5	28.0	0.0
lipa30a	CLP	0.0	2.5	3.9	4.3
lipa30b	CLP/Gurobi	0.0	18.9	21.8	0.0
lipa20a	CLP	0.0	3.2	5.3	7.5
lipa20b	CLP/Gurobi	0.0	16.8	22.5	0.0
nug30	CLP	0.0	9.0	27.4	30.5
nug28	CLP	0.0	7.2	26.4	28.1
nug27	CLP	0.0	4.7	30.1	39.8
nug25	CLP	0.0	3.8	27.0	33.6
nug24	CLP	0.0	9.6	23.6	23.0
nug22	CLP	0.0	6.6	23.5	48.5
nug20	CLP	0.0	6.8	17.3	39.3
nug18	CLP	0.0	5.4	17.0	31.0
nug17	CLP	0.0	3.0	12.0	33.8
nug16a	CLP	0.0	1.7	17.4	31.3
nug16b	CLP	0.0	3.1	14.2	32.6
nug15	CLP	0.0	1.4	7.0	18.3
nug14	CLP	0.0	3.4	7.7	38.1
nug12	CLP	0.0	2.1	6.2	36.0
rou20	CLP	0.0	3.4	16.0	28.5
rou15	CLP	0.0	9.4	7.8	24.0
rou12	CLP	0.0	4.3	2.0	31.1
scr20	CLP	0.0	12.7	35.7	127.6
scr15	CLP	0.0	22.9	47.9	96.1
scr12	CLP	0.0	16.5	9.8	103.9
sko100a	CLP	0.0	8.4	18.1	17.8
sko100b	CLP	0.1	8.3	18.7	17.6
sko100c	CLP	0.0	8.4	17.4	18.5
sko100d	CLP	0.1	8.5	16.5	17.3
sko100e	CLP	0.0	8.4	18.6	17.4
sko100f	CLP	0.1	7.7	16.7	17.0
sko90	CLP	0.0	8.8	16.5	19.3
sko72	CLP	0.1	8.5	19.6	21.4
sko64	CLP	0.0	8.4	22.2	17.2
sko56	CLP	0.0	9.3	22.8	23.1
sko49	CLP	0.1	6.6	20.8	27.5
sko42	CLP	0.0	6.6	24.8	26.8
ste36a	CLP	0.8	28.1	38.3	90.7
ste36b	CLP	0.0	47.2	136.5	515.1
ste36c	CLP	0.6	13.8	63.3	102.4
tai256c	CLP	0.2	5.8	20.9	18.9
tai150b	CLP	0.6	16.5	31.1	31.2
tai100a	CLP	1.1	8.8	14.2	15.2
tai100b	CLP	0.1	15.6	51.0	47.4
tai80a	CLP	1.2	8.0	15.0	15.4
tai80b	CLP	0.1	15.9	49.9	50.8
tai64c	CLP	0.0	4.8	217.6	38.7
tai60a	CLP	1.1	9.8	16.6	16.8
tai60b	CLP	0.0	20.6	64.6	77.2
tai50a	CLP	1.0	10.1	16.2	19.0
tai50b	CLP	0.0	18.8	64.1	69.4
tai40a	CLP	0.7	6.6	17.0	22.6

Continued on next page

Problem	Best model	**CLP (ours)** (%)	SA (%)	BB (%)	Gurobi (%)
tai40b	CLP	0.0	23.0	79.1	74.6
tai35a	CLP	0.4	9.0	15.4	19.2
tai35b	CLP	0.1	11.6	62.8	93.7
tai30a	CLP	0.0	5.9	16.0	18.3
tai30b	CLP	0.0	9.3	52.3	112.5
tai25a	CLP	0.4	7.9	15.3	24.1
tai25b	CLP	0.0	31.2	122.8	188.5
tai20a	CLP	0.3	6.8	14.1	22.9
tai20b	CLP	0.0	2.9	99.5	249.5
tai17a	CLP	0.0	7.4	13.3	28.5
tai15a	CLP	0.0	2.0	11.2	25.3
tai15b	CLP	0.0	0.4	571.3	572.9
tai12a	CLP	0.0	16.2	15.2	29.6
tai12b	CLP	0.0	18.0	39.8	118.2
tho150	CLP	0.3	11.1	20.6	19.8
tho40	CLP	0.0	12.7	31.5	39.2
tho30	CLP	0.0	9.7	32.2	49.6
wil100	CLP	0.0	4.8	9.7	10.7
wil50	CLP	0.0	4.5	12.2	14.6

F.1.4 Computational Cost-Based Algorithm Generation

We have developed a modular, multi-node, multi-GPU computational framework for discovering novel algorithms utilizing ensemble MCTS, RL, and tokenization techniques. Our framework includes a profiling module designed to estimate the computational cost of individual primitives. By leveraging this profiling, we set a fixed computational budget for training an ensemble MCTS-based RL agent, enabling it to identify optimal sequences of primitives that minimize the optimality gap while adhering to computational constraints. For supply chain networks exceeding 60 nodes, the solution space is astronomically large. Current state-of-the-art methods, such as simulated annealing (complexity $O(n^4)$) and branch and bound (complexity $O(2^n)$ in the worst case), become computationally prohibitive as problem size grows. In contrast, our CLP approach automatically generates tailored algorithms with substantially better computational complexity, typically ranging from $O(n^2)$ to $O(n^3)$. This improvement corresponds to an average 10,000-fold reduction in computational effort (measured in floating-point operations (FLOPs)) for problems of size 100. Additionally, the CLP method consistently achieves superior solution quality, frequently finding optimal solutions that state-of-the-art methods fail to identify. For instance, for problems of size 80, our approach consistently achieves the optimum.