
Physical Symbolic Optimization

Wassim Tenachi

Université de Strasbourg, CNRS
Observatoire astronomique de Strasbourg
UMR 7550, F-67000 Strasbourg, France
wassim.tenachi@astro.unistra.fr

Rodrigo Ibata

Université de Strasbourg, CNRS
Observatoire astronomique de Strasbourg
UMR 7550, F-67000 Strasbourg, France
rodrigo.ibata@astro.unistra.fr

Foivos I. Diakogiannis

Data61, CSIRO
Kensington, WA 6155, Australia
foivos.diakogiannis@data61.csiro.au

Abstract

We present a framework for constraining the automatic sequential generation of equations to obey the rules of dimensional analysis by construction. Combining this approach with reinforcement learning, we built Φ -SO, a Physical Symbolic Optimization method for recovering analytical functions from physical data leveraging units constraints. Our symbolic regression algorithm achieves state-of-the-art results in contexts in which variables and constants have known physical units, outperforming all other methods on SRBench’s Feynman benchmark in the presence of noise (exceeding 0.1%) and showing resilience even in the presence of significant (10%) levels of noise.

1 Introduction

Physical theories traditionally stem from empirical laws. Physicists typically observe natural phenomena, formulate empirical laws to describe them, and subsequently construct overarching theories that encompass these laws. For example, Newton’s law of universal gravitation was designed to explain both the motion of terrestrial objects as well as Kepler’s law of planetary motion [9]. However, with the rise of deep learning, many empirical laws have transitioned into complex neural network representations, rendering their integration into overarching physical theories expressed as analytical equations much more complex. Symbolic regression (SR) emerges as a pivotal approach to bridge this gap in interpretability.

Symbolic regression SR consists in the inference of a symbolic analytical function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that accurately represents the relationship $y = f(\mathbf{x})$ provided a data pair (\mathbf{x}, y) . Unlike numerical parameter optimization methods, SR involves the exploration of the space of functional forms themselves by optimizing the arrangement of mathematical symbols (such as x , $+$, $-$, \times , $/$, \sin , \exp , \log , and more). The discrete combinatorial challenge presented by SR makes it an “NP hard” (nondeterministic polynomial time) problem [15] without even factoring the additional challenge of optimizing free constants in a continuous space. SR therefore requires the development of efficient strategies for avoiding sub-optimal guesses.

Related works SR has traditionally been tackled using genetic programming, and is notably at the heart of the Eureqa software [11] and its successor algorithm AFP_FE [12]. However, with the advance of deep learning techniques, multiple frameworks have been proposed to use neural networks for SR [5]. As in previous deep learning SR work [4, 1, 10], here we employ a recurrent

neural network (RNN) to repeatedly sample tokens representing mathematical symbols or variables: ultimately generating complete expressions in prefix notation[†] in which operators are written first and their arguments second, which alleviates the need for parentheses. This type of approach includes DSR [10, 7] which is the current state-of-the-art approach for exact symbolic recovery in the presence of noise [5]. This approach relies on a risk-seeking reinforcement learning policy, which we adopt in this study. We also note that in the context of the physical sciences, previous works (e.g., [13, 8]) have taken into consideration the units associated with data, proposing to render datasets dimensionless using the Buckingham Π theorem [2]. This effectively renders variables and constants dimensionless by means of multiplicative operations amongst them as an initial step before SR, thereby relinquishing symbolic arrangement constraints arising from dimensional analysis but ensuring the physical consistency of generated expressions in terms of units.

Contribution Here we propose a Physical Symbolic Optimization (Φ -SO) framework. This framework operates in contexts in which the units of variables and constants involved are known leveraging dimensional analysis to constrain the symbolic arrangement during the generation of equations. Specifically, Φ -SO builds upon the reinforcement learning SR framework pioneered in [10], equipping it with an *in situ* supervised learning component designed to teach the RNN dimensional analysis rules, guiding its exploration of the search space towards not only accurate but physically meaningful combinations.

In Section 2, we detail our methodology. In Section 3, we show that in physical contexts in which all units involved are known, Φ -SO outperforms state-of-the-art baselines in exact symbolic recovery and discuss those results. Finally, Section 4 summarizes our conclusions.

2 Method

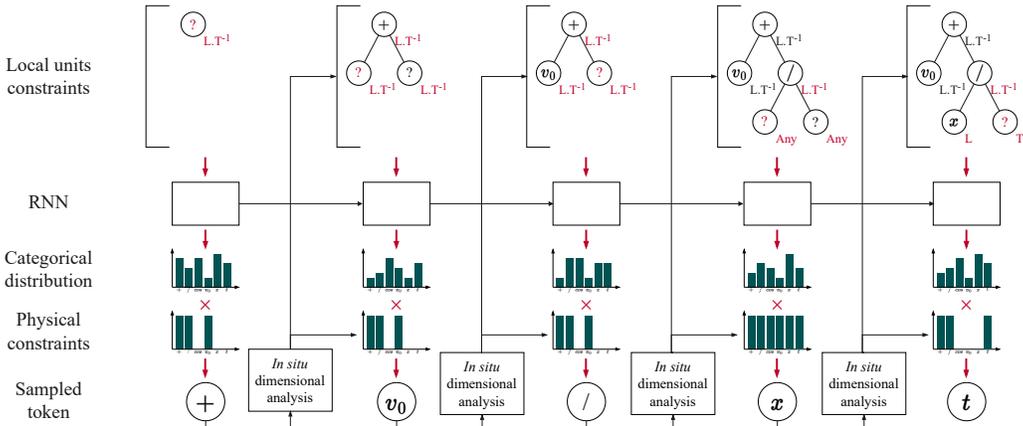


Figure 1: Expression generation sketch. For each token, the RNN is given symbolic and units related contextual information regarding the next token to generate. Based on this information, the RNN produces a categorical distribution over the space of available tokens (top histograms) as well as a memory state which is given to the RNN on its next call. Φ -SO generates *in situ* categorical labels encoding which tokens can be chosen based on dimensional analysis rules (bottom histograms). The generated distribution is then masked based on those constraints, forbidding tokens that would lead to nonsensical expressions. The resulting token is sampled from this distribution, Φ -SO then updates the local unit constraints of the partial tree representing the expression being generated. Repeating this process, from left to right, allows one to generate a complete physical expression, here $[+, v_0, /, x, t]$ which translates into $v_0 + x/t$ in the infix notation we are more familiar with.

Expression generation The expression generation procedure is detailed in Figure 1. At each token generation step, the RNN, here a long-short term memory (LSTM) type RNN [3], is given as input

[†]There is a one-to-one relation between this representation which is commonly used in symbolic computation and the “infix” representation we are more familiar with.

contextual information regarding the next token to be generated. This includes the nature of its parent token in a tree representation of the expression as well as its units or local units constraints if the token is a mathematical operation. Similar information is passed regarding the sibling and the previously generated token. In addition, the required units of the token to be generated and the number of dangling nodes in a tree representation (i.e. number of tokens required to finish the expression) are also passed to the RNN. This information is then leveraged by the RNN to produce a categorical distribution over the space of choosable tokens maximizing the probabilities of tokens obeying dimensional analysis rules and resulting in accurate expressions.

In situ dimensional analysis As illustrated in Figure 1 before each token generation step, we run an algorithm that updates the local units constraints² of the expression nodes based on the new token sampled at the previous step following the dimensional analysis prescriptions given in Table 1. This information is used to produce the units related inputs of the RNN as well as to constrain its output categorical distribution in favor of tokens obeying dimensional analysis. Effectively, we deterministically produce inputs and labels on the fly for the RNN to learn on in a supervised manner.

Expression	Units
$\tau_A \pm \tau_B$	Φ_A or Φ_B
$-\tau_A$	Φ_A
$\tau_A \times \tau_B$	$\Phi_A + \Phi_B$
τ_A / τ_B	$\Phi_A - \Phi_B$
τ_A^n	$n \times \Phi_A$
$\text{op}_0(\tau_A)$	$\mathbf{0}$

(a) Dimensional analysis rules

Expression	Requirement
$\tau_A \pm \tau_B$	$\Phi_A = \Phi_B$
$y = \tau_A$	$\Phi_y = \Phi_A$
$\text{op}_0(\tau_A)$	$\Phi_A = \mathbf{0}$

(b) Units requirements rules

Table 1: Dimensional analysis prescriptions to enforce. With $\tau_A, \tau_B, y, \Phi_A, \Phi_B, \Phi_y$ referring to two nodes, the output variable and the powers of their units vectors, op_0 denoting a dimensionless operation (e.g., $\{\cos, \sin, \exp, \log\}$) and τ_A^n representing any power operation (including e.g., $1/\tau_A = \tau_A^{-1}, \sqrt{\tau_A} = \tau_A^{\frac{1}{2}}$)

Training Expressions are rewarded based on their fit quality on a given target dataset (x, y) . For each expression, free constants appearing in it are first optimized using the LBFGS [16] algorithm and the reward defined as the squashed normalized root mean squared error ($\text{Reward} = 1/(1 + \text{NRMSE})$) is then computed. Subsequently, the RNN is trained based on those rewards following the risk-seeking policy gradients detailed in [10] with improvements from [7]. It should be noted that in order to prevent the expression generation phase going on forever, in addition to dimensional analysis considerations, the categorical distribution emitted by the RNN is masked in an expression length dependent manner such as to favor terminal nodes (i.e. variables and constants), thereby encouraging its termination, before a maximum expression size limit is reached. This indispensable length prior can conflict with the constraints arising from our dimensional analysis module, resulting in the expression being discarded. This conflict typically results in $\sim 90\%$ of expressions being discarded in the first few training epochs. This makes it essential for the RNN to learn dimensional analysis rules and not solely rely on the deterministic constraints imposed *a posteriori* in order to avoid such situations. This learning process can typically be monitored by keeping track of the rate of discarded expressions as a function of training epochs.

3 Results & Discussion

We evaluated Φ -SO following the SRBench standard SR benchmarking procedure [5] (github.com/cavalab/srbench) against 17 baseline SR methods. Conveniently, the bulk of this benchmark consists of 116 ground-truth challenges from [14] with physical units associated with each variable involved making it straightforward to apply our method. It is worth noting that, despite the

²Since the algorithm only has access to an incomplete expression containing dangling nodes, it is sometimes impossible to compute the required units of a token, in these cases, the token units are temporarily marked as free i.e. able to accommodate any units.

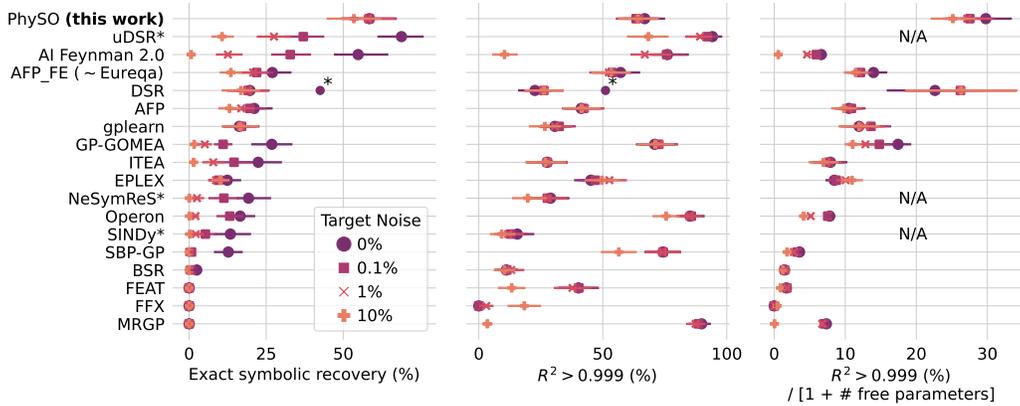


Figure 2: Exact symbolic recovery rate, rates of accurate expression (having $R^2 > 0.999$) and rate of accurate expressions normalized by the number of free parameters appearing in expressions for our PhySO algorithm and 17 baselines, averaged across 116 ground-truth Feynman SR problems following the standardized SRBench [5] benchmarking method.* denotes results taken from [6].

availability of units in this benchmark, Φ -SO and AI Feynman [14] are the only methods that have harnessed this dataset to leverage the associated units.

We maintained strict consistency by employing the same set of hyper-parameters for all challenges. These parameters included permitting the use of $\{+, -, \times, /, 1/\square, \sqrt{\square}, \square^2, -\square, \exp, \log, \cos, \sin\}$ as well as two dimensionless adjustable free constants and a constant equal to one $\{\theta_1, \theta_2, 1\}$. In addition, we strictly adhered to the established benchmarking rules, which included employing a dataset consisting of only 10,000 data points and limiting the maximum number of expression evaluations during the search to 1 million for each challenge. This computational process typically takes about 1 hour, utilizing all cores of an Intel Xeon W-2155 CPU. It should be noted that Φ -SO typically converges toward the correct expression well before reaching this evaluation limit or not at all. Φ -SO’s performances on these challenges against baseline methods are given in Figure 2.

Exact symbolic recovery In contrast to DSR, which relies solely on reinforcement learning, Φ -SO leverages both reinforcement learning and dimensional analysis. This combined approach yields significantly improved results, underscoring the substantial advantages conferred by unit constraints. While our method ranks second in noiseless scenarios, it excels in the presence of even minimal noise levels (exceeding 0.1%), outperforming all other baseline methods in exact symbolic recovery. Notably, even at a higher noise level of 10%, when scores for all other methods drop below 20%, Φ -SO consistently maintains a score above 50%. It is essential to note that many SR exploration strategies are driven by accuracy with minimal constraints on symbolic arrangement. Nevertheless, the paths leading to optimal fit quality and those leading to ideal symbolic arrangement (perfect fit quality and exact symbolic recovery) are not necessarily aligned. Essentially, one can enhance the fit quality of candidates over learning iterations while deviating further from the correct solution in terms of symbolic arrangement. As such, we attribute Φ -SO’s performance to the invaluable constraints on symbolic arrangements given by dimensional analysis.

Fit quality When considering the fraction of expressions with $R^2 > 0.999$ (defined as $R^2 = 1 - \frac{\sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2}{\sum_{i=1}^N (y_i - \bar{y})^2}$), numerous methods attain high scores by incorporating a substantial number of free constants. This approach often leads to the creation of complex expressions that frequently lack interpretability and deviate from dimensional analysis principles. While this may not pose significant issues in many fields in which accuracy is the only priority, it becomes a crucial concern in the context of the physical sciences. We therefore present in Figure 2 the rate of accurate expressions normalized by the number of free constants. This metric provides insight into the efficiency of models while taking into account their complexity. In this context, Φ -SO stands out as the leading method, excelling in the generation of concise, physically consistent, and interpretable

expressions that approximate datasets, that is when it does not actually find the exact target symbolic expression.

Limitations It should be noted that our system requires all variables and constants involved to have defined physical units to exploit dimensional analysis constraints and that it is mostly adapted for exact symbolic recovery of concise and interpretable expressions. This largely limits its relevance to the physical sciences, in contrast to other SR methods that aim to produce longer but more accurate approximations that are computationally less expensive to evaluate than neural networks. In addition, we note that contrary to end-to-end supervised learning based methods such as [4, 1], in the trial-and-error based strategy adopted here the neural network is reinitialized at the beginning of each SR task, preventing it from learning more generalized relations between datasets and analytical expressions. Future work will focus on equipping our framework with this ability as in [6].

4 Conclusion

We presented a Physical Symbolic Optimization (Φ -SO) framework that is able to constrain the generation of equations on the fly to ensure that the rules of dimensional analysis are obeyed by construction. We expect this framework to be useful for many symbolic computation applications in the physical sciences. Here, we combined it with the state-of-the-art reinforcement learning strategies from [10, 7] resulting in a physics relevant SR method achieving state-of-the-art results on the standard Feynman benchmark from SRBench.

Code availability

The documented code for the Φ -SO algorithm, along with demonstration notebooks, is accessible on GitHub at github.com/WassimTenachi/PhySO , complete with comprehensive documentation. A frozen version related to this work is released under tag `v1.0.0`  and deposited on zenodo: [10.5281/zenodo.8415435](https://zenodo.org/record/8415435).

For the sake of result reproducibility, we offer a straightforward method to replicate the outcomes presented in Figure 2 by simply executing the following command: `python feynman_run.py -equation i -noise n`. This command will run PhySO on challenge number $i \in \{0, 1, \dots, 119\}$ of the Feynman benchmark, employing a noise level of $n \in [0, 1]$.

Acknowledgments and Disclosure of Funding

RI acknowledges funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 834148). The authors would like to acknowledge the High Performance Computing Center of the University of Strasbourg for supporting this work by providing scientific support and access to computing resources. Part of the computing resources were funded by the Equipex Equip@Meso project (Programme Investissements d’Avenir) and the CPER Alsacalcul/Big Data.

References

- [1] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 936–945. PMLR, 18–24 Jul 2021.
- [2] Edgar Buckingham. On physically similar systems; illustrations of the use of dimensional equations. *Physical review*, 4(4):345, 1914.
- [3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [4] Pierre-Alexandre Kamienny, Stéphane d’Ascoli, Guillaume Lample, and Francois Charton. End-to-end symbolic regression with transformers. In Alice H. Oh, Alekh Agarwal, Danielle

- Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [5] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio de Franca, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason Moore. Contemporary symbolic regression methods and their relative performance. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.
- [6] Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. A unified framework for deep symbolic regression. *Advances in Neural Information Processing Systems*, 35:33985–33998, 2022.
- [7] Mikel Landajuela, Brenden K Petersen, Soo K Kim, Claudio P Santiago, Ruben Glatt, T Nathan Mundhenk, Jacob F Pettit, and Daniel M Faissol. Improving exploration in policy gradient search: Application to symbolic optimization. In *1st Mathematical Reasoning in General Artificial Intelligence, International Conference on Learning Representations (ICLR)*, 2021.
- [8] Konstantin T Matchev, Katia Matcheva, and Alexander Roman. Analytical modeling of exoplanet transit spectroscopy with dimensional analysis and symbolic regression. *The Astrophysical Journal*, 930(1):33, 2022.
- [9] Isaac Newton. *Philosophiae naturalis principia mathematica*, volume 1. G. Brookman, 1833.
- [10] Brenden K Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021.
- [11] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.
- [12] Michael Schmidt and Hod Lipson. *Age-Fitness Pareto Optimization*, pages 129–146. Springer New York, New York, NY, 2011.
- [13] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *Advances in Neural Information Processing Systems*, 33:4860–4871, 2020.
- [14] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.
- [15] Marco Virgolin and Solon P Pissis. Symbolic regression is NP-hard. *Transactions on Machine Learning Research*, 2022.
- [16] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.