SciMED: A Computational Framework For Physics-Informed Symbolic Regression with Scientist-In-The-Loop

Liron Simon Keren\textsuperscript{1,}\textasteriskcentered, Alex Liberzon\textsuperscript{1}, Teddy Lazebnik\textsuperscript{2}

\textsuperscript{1} Turbulence Structure Laboratory, School of Mechanical Engineering, Tel Aviv University, Israel
\textsuperscript{2} Department of Cancer Biology, Cancer Institute, University College London, UK

* lirons.gb@gmail.com

Abstract

Discovering a meaningful, dimensionally homogeneous, symbolic expression that explains experimental data is a fundamental challenge in many scientific fields. We present a novel, open-source computational framework called \textit{Scientist-Machine Equation Detector} (SciMED), which integrates scientific discipline wisdom in a scientist-in-the-loop approach with state-of-the-art symbolic regression (SR) methods. SciMED combines a genetic algorithm-based wrapper selection method with automatic machine learning and two levels of SR methods. We test SciMED on four configurations of the settling of a sphere with and without a non-linear aerodynamic drag force. We show that SciMED is sufficiently robust to discover the correct physically meaningful symbolic expressions from noisy data. Our results indicate better performance on these tasks than the state-of-the-art SR software package.

Introduction

Modern research is constricted from roughly three main phases: observation, hypothesis generation, and hypothesis validation \cite{1,3}. During the observation phase, researchers collect data about the world, which later, during the hypothesis generation, is used to generate a hypothesis that explains this data. A good explanation commonly allows for extrapolation, and thus the prediction of new data of the same observed system during the hypothesis validation phase \cite{4,5}. A common way of hypothesis generation is Symbolic Regression (SR), where researchers discover a symbolic expression (sometimes noted as an equation or a symbolic function) that accurately matches a given dataset \cite{6,8}. To be exact, researchers assume a set of measurements or features are taking part in some natural phenomena and record samples of these measurements. Intuitively, the SR task is to unveil a symbolic expression for the function connecting the observed measurements \cite{9).

SR stands at the root of many fields of research such as engineering \cite{10}, psychology \cite{11}, economy \cite{12}, physics \cite{13}, chemistry \cite{14}, and others \cite{15} since all mathematically expressed models are formally a function. Thus, the hypothesis generation process in all of these fields can be viewed as the discovery of a function that allows us to determine a value of interest, given a set of related measurements. As a result, multiple computational frameworks have been proposed to automate this
A common example of such symbolic regression is the linear regression that operates under the assumption that the source measurements and the target measurement are linearly dependent. This somewhat simplistic assumption produced many useful models via simple computations of a system of linear equations. However, this solution does not work for non-linear cases which seem to dominate most (if not all) fields of science. The general symbolic regression problem remains unsolved and super-exponential to the number of measurements which makes it infeasible to brute force for even medium-size datasets. Indeed, SR is known to be an NP-hard task.

SR and other NP-hard tasks such as Vertex Cover and Hamiltonian Circuit Completion are of great interest and hence multiple approximations to the optimal outcome of these tasks were investigated. In particular, one can divide these attempts into two main groups: analytical-based and heuristic-based approximations. The first aims to find an algorithm that can approximate the optimal solution within some bounded error, in a polynomial asymptotic time, based on the data alone. The latter approach adds assumptions to the task to divide it into simpler cases that can be solved on their own and well-approximated cases that satisfy the assumptions. The heuristic-based approach has shown promising results in general and for SR in particular. Indeed, SR has been tackled with many methods and heuristics such as genetic algorithms and sparse regression.

Though these methods provide promising results, they do not consider valuable domain knowledge that their expert users (from now on referred to as scientists) can provide to help direct the regression efforts. To tackle this shortcoming, we present the Scientist-Machine Equation Detector (SciMED) system, designed to deduce equations using four levels of search and optimization methods, structured to direct more attention and resources to promising search directions, somewhat similar to the search route of a scientist. The novelty of the proposed work lies in integrating a scientist-in-the-loop (SITL) approach with numerical and analytical methods. Namely, this approach allows the user to direct the search process, gaining credibility of the results while reducing the computational time and resources required by SciMED compared to other SR systems. The unique design of SciMED allows a user to easily control the search direction and reduce the search space, using several input junctions throughout the system’s pipeline.

The rest of this paper is organized as follows. First, we review the current state-of-the-art SR systems. Afterward, we formally introduce SciMED. Subsequently, we present five experiments representing the cases SciMED aim to tackle with their results. Lastly, we summarize our conclusions and discuss opportunities for future work.

Related work

The task of fitting a numerical or symbolic function on a set of data points is common in multiple fields of research. As opposed to a regression task, which provides a model structure and fits it to available data, symbolic regression (SR) simultaneously searches for a model and its parameters. Due to the constantly increasing amount of data and computational capabilities, multiple attempts to automate data transformation into knowledge using SR have been proposed.

The process of automating SR faces multiple challenges, such as an exponentially sizeable combinatorial space of symbolic expressions leading to a slow convergence speed in many real-world applications, or increased sensitivity to overfitting stemming from unjustifiably long program length.

SR can be especially useful in physics, frequently dealing with multivariate
noisy empirical data from nonlinear systems with unknown laws [36]. Moreover, the SR’s output must retain dimensional homogeneity, meaning all terms in SR expression have to carry the same dimensional units. On the one hand, this constraint reduces the potential search space for the SR, while on the other hand, it introduces a meta-data on the model construction that one needs to consider and handle [37]. [38] stated that symbolic equations in physics broaden human understanding by a) exposing expressions that can be indicators of underlying physical mechanisms, as well as b) identifying metavariables (variable combinations or transformations) that might ease later empirical modeling efforts. This sort of explainability helps to examine how the model’s behavior, variables, and metavariables correspond to available prior knowledge in the field.

There are numerous methods for performing SR [37, 39] that we can loosely divide into four main groups depending on the underlying computational technique: brute-force search, sparse regression, deep learning, and genetic algorithms. Brute-force search-based SR systems are, in principle, capable of successfully solving every SR task [40]. In practice, a naive implementation of brute-force methods is infeasible, even on small-sized datasets, because it is excessively computationally expensive. Furthermore, these systems tend to overfit given large and noisy data [41], which is the case of typical empirical results in physics.

Sparse regression systems can substantially reduce the search space of all possible functions by identifying parsimonious models using sparsity-promoting optimization. A notable sparse SR algorithm built specifically for scientific use cases is proposed by [42] called SINDy. SINDy uses a Lasso linear model for sparse identification of non-linear dynamical systems that underlie time-series data. SINDy’s algorithm iterates between a partial least-squares fit and a thresholding (sparsity-promoting) step. Since its inception, SINDy has been continuously improved. For example, [43] increased its ability to solve real-time model identification problems given noisy data, [44] added optimal model selection over various values of the threshold, and [45] have created PySINDy; an open-source Python package for applying SINDy.

Deep learning (DL) for SR systems works well on noisy data due to the general resistance of neural networks to outliers. An example of a Deep Symbolic Regression (DSR) system is proposed by [46], which is built for general SR tasks rather than specifically for data from the physical domain. This DL-based model uses reinforcement learning to train a generative RNN model of symbolic expressions and adds a variation of the Monte Carlo policy gradient technique termed "risk-seeking policy gradient" to fit the generative model to the precise formula.

SR systems based on genetic algorithms (GA) can efficiently enforce prior knowledge to reduce the search space of possible functions. For example, SR can adhere to a specific shape of the solution [47–50], or utilize probabilistic models to sample grammar of rules that determine how solutions are generated [51–54]. A simple, yet effective, implementation of GA for SR is gplearn Python Library [55]. It begins by building a population of naive random formulas representing a relationship between known independent variables (features) and targets (dependent variables) as tree-like structures. Then, in a stochastic optimization process, it performs replacement and recombination of the sub-trees, evaluating the fitness by executing the trees and assessing their output, and stochastic survival of the fittest. This method performs well on linear real-world problems [37] and can be easily manipulated as a base for more complex systems.

The current state-of-the-art SR system in physics, based on brute-force search, is the so-called AI Feynman [56]. AI Feynman combines neural network fitting with a recursive algorithm that decomposes the initial problem into simpler ones. If the problem is not directly solvable by polynomial fitting or brute-force search, AI
Feynman trains a NN on the data to estimate functional symmetry and/or separability, presumably existing in underlying laws. Then symmetry and/or separability are exploited to simplify the problem solved recursively. An updated version of the algorithm adds Pareto optimization with an information-theoretic complexity metric to improve robustness to noise [37, 57]. If dimensional samples are provided, a dimensional analysis solver is applied, doubling as a feature selection method that reduces the search space of the unknown equation. This is done by constructing a new set of non-dimensional features that both contain at least one representation of each dimensional (original) feature and has the smallest number of non-dimensional features possible.

AI Feynman obtains phenomenal results presented at [56]. However, it uses a series of restrictive assumptions that might lead to indefinite failure in some cases outside the Feynman dataset [37]. First, physical mechanisms might be implicit, therefore undetectable if separability is assumed (e.g., the equation can presumably be written as a sum or product of two parts with no variables in common). Examples of such implicit functions in physics may be linkages behavior in mechanical engineering [58], or motion in fluids with a non-linear drag force [59]. Second, the automatic dimensional analysis method applied does not allow the construction of specific non-dimensional numbers that are known to be related to the target or suspected of it. Therefore, denying the integration of valuable domain knowledge may reduce the search space. Another shortcoming of this system is its high sensitivity even to small amounts of noise [37], making it hard to implement on real-world measurements.

Recently, [37] introduced SRBench, a benchmarking platform for SR that features 21 algorithms tested on 252 datasets, containing observational data collected from physical processes and data generated synthetically from static functions or simulations. The authors revealed that ”Operon” by [60] was the best performing framework in terms of accuracy. In contrast, GP-GOMEA by [61] was the best performing framework in terms of the simplicity of the found mathematical expressions. Both frameworks, like most of the SR frameworks examined in SRBench, were not constructed to work specifically in the physical domain, meaning they do not constrain the outcome by any physical reasoning like dimensionality. Other SR frameworks presented in SRBench which are physics-oriented, such as the strongly-typed GP [62], grammatical evolution [63], and grammar-guided [64] frameworks, were shown to perform, on average, worse than others models [37].

Scientist-Machine Equation Detector

In this section, we introduce Scientist-Machine Equation Detector (SciMED) for finding symbolic expressions from physical datasets with the SITL approach. SciMED is available as a free open source library in Python[1] SciMED is constructed from four components: a genetic algorithm-based feature selection, a genetic algorithm-based automatic machine learning (AutoML), a genetic algorithm-based symbolic regression, and a Las Vegas search symbolic regression, as illustrated in Fig. [1] Each component allows the user to easily insert physical knowledge or assumptions, directing the search process for a more credible result with fewer resources. In particular, we use an AutoML component to facilitate the SR task by enriching the data with synthetic samples. In addition, we use two different approaches for SR such that one is less resource and time-consuming but stochastic, which may result in sub-optimal results. At the same time, the second is more computationally expensive but more stable and accurate, on average. To the best of our knowledge, SciMED is the only

[1] Available at [https://github.com/LironSimon/SciMED]
physics-informed SR system that emphasizes knowledge specific to its current task via a scientist-in-the-loop approach rather than attempting to apply general assumptions to all SR tasks. A detailed description of each component, their interactions, and the physical assumption used by SciMED are provided below.

![SciMED Diagram]

**Figure 1.** A schematic illustration of SciMED’s structure. It is constructed from four components that each can be independently switched off or on. Theoretical knowledge or hypothesis can be entered at five input junctions, affecting the results of SciMED. SciMED provides two outputs that shed light on the unknown laws explaining the data.

**Dimensionality aware symbolic regression**

Let us assume that any function \( f \) can be represented by a combination of mathematical operators and available parameters (i.e., features) noted as \( M := \{m_1, \ldots, m_k\} \). Now, to enforce physical units, let us assume that each parameter, \( m_i \in M \), has a Standard Unit Vector (SUV) representing the physical units associated with \( m_i \) and donated by \( \zeta_i \in \mathbb{R}^\alpha \), where \( \alpha = 7 \) following the international system of units convention. Similarly, for each function \( f \), there is an accompanying
function $f^*$ such that $f^*$ gets the SUV vectors of the arguments and returns the corresponding output according to $f$. For example, given $f(x, y) = x \cdot y$, where $x, y \in M$ and have a SUV containing only the first 2 standard units (the rest of the 5 arguments being 0), equaling to $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, then $f^*(\zeta_x, \zeta_y) := \zeta_x + \zeta_y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

These assumptions produce two computational properties which SciMED exploits: 1) it enables the representation of the function’s search space using a parse tree with a relatively small number of span functions, 2) it enables dimensional analysis of the features, which guarantees any algebraic combination of $m$ adheres to domain knowledge.

It is important to note that SciMED works on both dimensional and non-dimensional datasets, chosen at the user’s discretion. Use can apply the dimensional analysis to the data before SciMED analysis, ensuring dimensionally homogeneous results. The user could integrate the knowledge of known non-dimensional numbers or suggest new non-dimensional combinations. SITL feature selection component will help to direct the search to a more insightful subset of features and achieve symbolic regression with fewer resources.

**A priori feature selection component**

The number of parameters for an SR task grows rapidly, especially in non-linear problems with unknown underlying laws. Like in traditional SR, the choice to include features that can potentially carry relevant information dramatically affects the result. This is because one needs to balance between including all relevant information and not obscuring the dynamics by creating a search space too large. In the feature selection component, we reduce the search space by extracting only the most relevant parameters using a GA-based approach.

Users can suggest various plausible representations of dimensional or non-dimensional features based on knowledge or an educated hypothesis. To explore various plausible non-dimensional combinations the user can introduce meta-data in the form of distinct groups of features, where each feature contributes the same knowledge. If no meta-data is introduced, SciMED assumes that each feature is the sole representation of a distinct group. Formally, let $F := \{ f_1, \ldots, f_n \}$ be the set of provided features to SciMED and $\Phi := \{ \phi_1, \ldots, \phi_l \}$ a set of features sets such that $\bigcup_{i=1}^l \phi_i = F$ and $\forall i, j \in [1, \ldots, l] : \phi_i \cap \phi_j = \emptyset$ where $i \neq j$. Thus, given $F$ and $\Phi$, SciMED choose only a single feature for each set $\phi_i$ for $i \in [1, \ldots, l]$.

We implement this behavior using a GA-based approach as follows. A *gene* is defined by the subset of features from the feature pool ($F$), where each chromosome in the gene is a feature from a distinct group ($\phi$). This information is encoded using an $n$-bit vector, where the $i_{th}$ bit in the vector corresponds to the $i_{th}$ feature group and represents which feature from this group was selected. This ensures each gene contains exactly one feature from each group the user provides, as needed.

Without *a priori* knowledge provided by the user, the GA is designed to optimize two objectives: a) maximize the obtained model’s accuracy (or any other fitness metric used by the user) and b) minimize the number of features selected. To do so, we define the following fitness function:

$$FS_{\text{fitness}}(S) := \omega_{fs} M_{\text{fitness}}(S) + (1 - \omega_{fs}) \frac{|S|}{|M|},$$

where $\omega_{fs} \in [0, 1]$ is a balance weight between the model’s performance and the obtained feature subset’s size and $M_{\text{fitness}}(S)$ is the model’s fitness function outcome for a chosen feature subset $S$.  


In addition, the three genetic operators: selection, crossover, and mutation, are defined as follows. First, the **selection** operator used in the "tournament with royalty" selection operator \[65\] where each gene has a probability which is corresponding to its normalized fitness function to be chosen for the next generation while the genes in the top \(\delta \in [0, 1)\) portion are taken at least once to the next generation. The fitness score for each gene is assigned using an automatic machine learning component, described in the next chapter. Second, the **crossover** operator is the "single-point" crossover operator \[66\] where a point \(i\) is chosen at random so that the first \(i\) bits are from one parent and the remaining bits are from the second parent. Lastly, the **mutation** operator, where a chromosome in each individual mutates with a probability \(\rho\) determined by the normalized size of the feature group it represents. If a chromosome is chosen, its value is randomly altered to represent a different feature from the same group.

**Figure 2.** An illustration of the feature selection process. One can see how SciMED performs feature selection for experiment B, where all features and the outcome represent the actual process and desired result. Here, a dataset with 33 features \((f(\mathbf{\bar{x}}) = (x_1, x_2, \ldots, x_{33}))\) is divided to 9 feature groups, using the meta-data provided by the user, where a single feature is selected from each group. This subset is passed to an SR component, revealing the unknown equation containing only two features. The physical background of the features and the division into groups is explained in detail in the Appendix.

**Automatic machine learning extrapolation component**

In this component, we train an ML algorithm to perform “black-box” predictions of the target value. This is used to assign fitness scores to each gene from the *a priori* feature selection component and to provide the user with a tool to generate syntactic data from the sampled data. The latter contribution is needed to cover the input space for the SR task uniformly. Insufficient input space coverage is one of the leading challenges of applying SR methods on experimental data \[67\]. Of note, this analysis is
performed on the features’ subset as obtained from the previous component. Formally, given a dataset $D \in \mathbb{R}^{z \times k}$ with $k \in \mathbb{N}$ features and $z \in \mathbb{N}$ samples, we utilize the TPOT [68] AutoML library, that uses a GA-based approach, to generate and test ML pipelines based on the popular scikit-learn library [69]. Formally, we run the TPOT regressor search method with a computational budget limitation to obtain an ML pipeline that will approximate and generalize the data. To prevent overfitting, the $k$-fold method is used [70]. Moreover, we allow the model’s performance to be a vector of metrics (for instance $[\text{MAE}, \text{MSE}, R^2, \text{T-test’s } p \text{ value}]$), computing the Pareto front’s integral [71] to obtain the final model’s performance score. Once the model is obtained, the mean and standard deviation of the $k$ iterations are computed and tested against a user’s provided threshold values. If the model is extrapolating the data well and stable enough across the data, as reflected by these two values, additional $\tau \in \mathbb{N}$ synthetic values are sampled by querying the obtained model. The user defines the distribution and value of $\tau$. As a default, the user provides a radius $r \in \mathbb{R}^+$ and several neighbors points $\kappa \in \mathbb{N}$ such that each syntactic data point is of distance $d \leq r$ of at least $\kappa$ data points. Intuitively, these two parameters ensure that the synthetic data enrich the input domain while not introducing too far values in which our confidence about their value is small and thus can introduce errors to later predictions.

In addition, this component of SciMED sheds light on the feature groups selected by the user, who can infer physical insight into the expected solution before obtaining a symbolic expression.

**Genetic algorithm based symbolic regression component**

In this component, we again utilize the GA approach to find a symbolic regression model. In particular, we follow the work proposed by [72] which extends the gplearn library [55]. Formally, a gene is represented using the S-expression [73], mixing between variables, constants, and functions. Initially, we use the Full initialization method, where all the S-expressions represented trees in the first generation have all their leaf nodes (variables or constants) at the maximal distance from the root. Afterward, and for each generation, the three GA operators are implemented as follows:

- **Selection** - a combination of the tournament with royalty and the Genitor methods. Namely, the genes in the population are ranked by their fitness score. A portion of the population with the genes with the best fitness score is carried forward into the next generation. Afterward, from the remaining genes, a gene has a probability of being carried forward into the next generation relative to its fitness score, normalized to the sum of all the fitness scores in the population. When a gene is chosen to be included in the next generation, several mutations are performed corresponding to its normalized fitness score.

- **Mutation** - we use the point mutation method. Namely, a node in the S-expression tree is chosen randomly and replaced with another feasible value. In the context of SR, parameters are replaced with parameters, and functions are replaced with functions of the same number of arguments.

- **Crossover** - firstly, two genes are taken from the population at random. A random subtree of the first gene is then replaced with a subtree from the second gene program, and the other way around, to generate two new genes.

After applying these three operators, the evaluating phase takes place to determine the fitness score of the new gene population for the next generation. At this step, the
gene is evaluated in a \( k \)-fold manner \([70]\) and takes the value of the evaluation of the whole data rather than a certain section. Each evaluation of the gene on the dataset is done with a loss function provided by the user. Nonetheless, as default, we use the following loss function, inspired by \([74]\) that gets the gene \((g)\), the gene’s predicted value \((y_p)\), and the target’s true value \((y_t)\):

\[
L(g, y_p, y_t) := \omega_1 \|y_p - y_t\|_1 + \omega_2 \|y_p - y_t\|_2 + \omega_3 \|y_p - y_t\|_\infty + \Psi |g|, \tag{2}
\]

where \(\{\omega_i\}_{i=1}^{3} \in [0, 1]\) such that \(\sum_{i=1}^{3} \omega_i = 1\), \(\|z\|_k\) is the \(L^k\) norm of the vector \(z\), \(\Psi \in \mathbb{R}^+\) is a weight for punishing the gene proportional to its size. It is important to note that SciMED allows to fine-tune the value of \(\Psi\) over a range using the grid-search method \([75]\).

This component can be initialized several times with different initial populations. After all the runs are finished, the stability of the outcomes is tested in two ways. First, the evolution of the standard deviation of a selected performance metric is evaluated to identify whether it’s converging. Second, the found equation from each run is compared to check if a specific equation is repeated for a minimum of a user-defined percentage of the runs.

**Las Vegas symbolic regression component**

In this component, we search for a symbolic expression in a more stable but computationally expensive manner as compared to the GA-based SR component. Similar to the GA-based SR component, we define a candidate solution to be represented by an S-expression tree. However, in this phase, all functions that get more than two inputs are divided into an S-expression tree of function that does satisfy this condition. Similarly, all single input functions are rephrased to get two inputs such that the second input is ignored. This allows us to represent all candidate solutions as full binary trees (FBT). As such, given a range of candidate solution sizes \(\xi_1, \xi_2\) (\(\xi_1 \leq \xi_2\)), we compute all possible FBTs. Next, randomly, an allocation of functions and variables are chosen for one FBT and evaluated using Eq. (2). During the evaluation process, the outcome of the FBT computation is used to train a linear regression to find coefficients of the obtained symbolic expression. After a pre-defined number, \(\theta \in \mathbb{N}\), of such candidate solutions are evaluated, we update the probability that a new sample would be chosen by setting it to be the normalized value of K-nearest neighbors \([76]\) from the already sampled candidate solutions, inspired by \([77]\). This process is terminated once a user-defined number of attempts (or given computation time) has been reached or if all possible allocations were evaluated. Either way, the candidate solution with the smallest loss value is returned. The motivation for this approach is to find and sample the most promising areas in the search space iteratively.

A user may opt to direct the stochastic search process by introducing two types of knowledge: a) a hypothesis for the structure of the optimal solution, and b) a user can use any sampling strategy believed to obtain an optimal solution faster.

Notably, as this component is computationally heavy, a user can avoid it entirely and settle for the previous GA-based SR. This waiver can be automatically implemented if the results obtained from the GA-based SR are consistent, appearing over a pre-defined portion of the outcomes from multiple runs and maintaining the T-test’s \(p\) value and coefficient of determination \(R^2\) of over a user-defined percentage.
Experiment Design

Motivation

We evaluated SciMED on four separate tasks, each designed to highlight the importance of a different component in SciMED. First, we assessed SciMED ability to detect linear relations between features from scarce and noisy data (experiment A). Here, we aim to highlight the contribution of the GA-based SR component and its ability to perform SR efficiently. Second, we tested the ability of SciMED to find a linear equation from a vast dataset of tens of features (experiment B). This experiment aims to demonstrate the contribution of the \textit{a priori} feature selection component by incorporating domain knowledge and reducing the search space. Third, we examined the ability of SciMED to find a non-linear equation from data with noise and a large number of features (compared to the average number of features in a benchmark set of 100 physical equations \cite{56}). This experiment (experiment C) is intended to demonstrate the contribution of the LV-based SR and its robustness to noise. In experiment D, we demonstrate how the AutoML component may alert the user that a parameter of crucial importance is missing. To do so, we evaluated SciMED on a dataset with non-linear feature relations that is missing one essential feature. This experiment mimics a reasonable scenario in scientific research, where a researcher assumes to know all the parameters governing a phenomenon but neglects to consider (at least) one. To increase the difficulty of this experiment, the chosen feature has hidden physical relations to other introduced features. In turn, this may lead to misleading performance scores and highlights the difficulty of obtaining a reliable symbolic expression.

As a final experiment (experiment E), we performed robustness or noise analysis, demonstrating SciMED’s performance in the presence of three different types of noise and at various noise levels.

Setup

The basic settings for experiments A-D are summarized in Table 1. The data for each experiment is generated in a table-like manner, demonstrated in Fig. 1 where columns represent variables with the last column being the target value calculated with them (e.g. $x_1, x_2, \ldots, f$ where $f(x_1, x_2, \ldots)$). The rows of the table contain the numbers representing samples of each variable. The functions used to generate the data of each experiment are listed in Table 2. Each of these functions is unknown to SciMED, and SciMED is required to deduce it from the data. The dataset for experiment D is generated similarly to the case of experiment C, except that the column containing the $z_4$ variable is deleted after the target column is generated, meaning there’s no possible way of constructing the true equation for the target from the given variable columns.

All but experiment B had noise added to target values. The dataset of experiment A has 400 samples split 75/25% between training and testing. In the rest of the experiments, the dataset contains $10^4$ samples split 80/20% between training and testing. All GA and AutoML components of SciMED are tuned in a 5-fold manner. In experiment C, a grid search is performed on the parsimony term of the GA-based SR component within the range presented in Table 1. In all other experiments, a parsimony term of 0.02 is used instead of the grid search.

Results

For every experiment, we present three results: 1) A scatter plot of AutoML predictions versus ground truth. 2) A vector of performance scores for all components
Table 1. Settings used in experiments A-D

| Setting                                                      | Value                     |
|--------------------------------------------------------------|---------------------------|
| Number of samples                                           | 10,000 or 400             |
| Test size                                                   | 20% or 75%                |
| Number of times the AutoML & GA based SR components were run | 20                        |
| GA based SR parsimony term range                            | 0.01-0.025                |
| Stability threshold for GA-based SR outcomes                 | 60%                       |
| T-test’s p value threshold for GA-based SR outcomes          | 0.8                       |
| Las Vegas component size range                               | 5-17                      |
| Number of syntactic data points (τ)                         | 10,000 or 400             |
| The automatic machine learning sampling radius (r)          | 7.5                       |
| The automatic machine learning number of neighbors points    | 3                         |
| Termination criteria in hours                                | 24                        |
| Levels of target noise (in all exp. but B)                  | 2%                        |
| Elementary functions used in the SR components              | add, sub, mul, div        |

Table 2. The unknown equations for the experiments SciMED and AI Feynman were tested on. The data for exp. D was generated using Eq. C, but a partial dataset was given to SciMED. Therefore there is no true function underlying the samples. For a physical representations of \( \vec{x}, \vec{y}, \vec{z} \) see the Appendix.

| Experiment | Unknown Equation |
|------------|------------------|
| A          | \( f_1(x_1, x_2, x_3) = x_1 + x_2 x_3 \) |
| B          | \( f_2(y_1, y_2, \ldots, y_{33}) = 1.33 y_{30} y_{31} \) |
| C          | \( f_3(z_1, z_2, z_3, z_4) = \left( 13.08 (z_1 - z_2) z_3 \right) / \left( z_2 z_4^2 \right) \) |
| D          | \( f_4(z_1, z_2, z_3) = NA \) |

of SciMED. 3) The discovered equation. In experiment C, we highlight inferring the correct numerical value of the prefactor. This result means SciMED could estimate the gravitational acceleration from noisy data correctly, a difficult task by itself [78].

Additionally, the state-of-the-art AI Feynman system was used to find all four unknown equations of experiments A-D. For each experiment, we repeated the trial 20 times, and the most repeated outcome from all runs was chosen as the outcome. These results were compared to that of SciMED in Table 4.

For the noise analysis (i.e., experiment E), we repeated experiments A and C while increasing the percent of noise introduced to either the input variables, target variable, or both variables. For each type and amount of noise, we repeated experiments A and C for \( n = 100 \) times, reporting the percent of correct equations from all results.

Fig. 3 shows the prediction capabilities of the ML acquired by the AutoML component for experiments A-D. For each plot, a linear regression line is fitted to the values predicted by the ML (noted as \( f_{pred} \)) as a function of the true target values (noted as \( f_{true} \)). All coefficients of determination (\( R^2 \)) scores indicate the found ML was accurate enough to enrich the data domain reliably. Specifically, in experiment B, the linear regression was optimal (i.e. \( f_{pred} = f_{true} \)). The fact this was achieved only in experiment B is reasonable, as this is the only experiment in which noise was not added to the target. In the rest of the experiment, which included noise, a small number of outliers are seen.

In experiment D, where a single variable was removed from the data, the accuracy of the predictions declined, as seen by both the \( R^2 \) score and the number of outliers. Nonetheless, the decline is not as dramatic as expected, as the variable removed from
Figure 3. Predictions acquired with the ML pipeline found in the AutoML component versus the true target value. Lines represent the regression, and the respective equation is shown in the legend.

The data depends on other variables given to SciMED, meaning it might have revealed the necessary information from the given data.

Table 3 reports the performance scores of the SciMED’s AutoML and both SR components for experiments A-D. Since the AutoML and GA-based SR components were run multiple times, their scores are presented as a mean ± standard deviation. Following Fig. 3, one can see that the AutoML component shows good performance over all four metrics (MAE, MSE, $R^2$, T-test’s $p$ value). As expected, the LV-based SR component consistently outperformed the GA-based SR component, presenting excellent results for all but experiment D.

The combination of good performance overall metrics by the AutoML component and poor performance overall metrics by the LV-SR component is an indication that at least one dependent variable is missing in the dataset. The performance scores indicate that ML accurately learned the necessary information from the given variables. Still, the robust SR component failed to find an equation that remotely describes the data (as seen by the zero-valued T-test’s $p$ value). Hence no accurate equation can be formulated with the given variables, meaning at least one variable is
missing from the equation.

In addition, Table 3 shows the advantages of combining the GA-based SR with the LV search; GA performs well on relatively simple SR tasks but fails when there is an extensive search space or noise. Following that, it is evident that experiments B-D have higher MAE and MSE scores, coupled with lower $R^2+$ and T-test’s $p$ value, compared to experiment A, which is more straightforward.

### Table 3. Performance scores of the AutoML and both SR components of SciMED, for experiments A-D.

As the AutoML and GA-based SR components are run multiple times, their scores are presented as a mean ± standard deviation.

| Experiment | MAE       | MSE       | $R^2$     | T-test’s $p$ value |
|------------|-----------|-----------|-----------|--------------------|
| AutoML     | 0.006 ± 0.001 | 0.000 ± 0.000 | 1.000 ± 0.000 | 0.960 ± 0.007     |
| GA - SR    | 0.457 ± 0.000 | 0.386 ± 0.000 | 0.999 ± 0.000 | 0.993 ± 0.000     |
| A          | 0.439 ± 0.000 | 0.371 ± 0.000 | 0.999     | 0.980              |
|            | 0.000 ± 0.000 | 0.000 ± 0.000 | 1.000     | 1.000              |
|            | 1.111 ± 0.000 | 13.979 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000     |
| LV - SR    | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.986 ± 0.004 | 0.918 ± 0.170     |
|            | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000     |
|            | 0.002 ± 0.000 | 0.000 ± 0.000 | 0.989     | 0.994              |
| C          | 0.009 ± 0.000 | 0.000 ± 0.000 | 0.953 ± 0.008 | 0.948 ± 0.047     |
|            | 1.407 ± 0.000 | 8.134 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000     |
|            | 416.865 ± 0.000 | 231,982.744 ± 0.000 | 0.000     | 0.000              |

The outcome of SciMED is presented in Table 3 alongside AI Feynman’s. In experiment A, both systems found the unknown equation despite the noise applied to the target. In experiment B, both systems correctly identified two out of 33 variables appearing in the equation and their algebraic relation. However, SciMED found a numerical prefactor smaller by 0.02 than the true value and added a constant term of 0.03, compared to AI Feynman, which found a prefactor smaller by 0.33 than the true value and did not add a constant term.

In experiment C, AI Feynman’s failed to find the correct equation, leaving out one parameter and incorrectly identifying the algebraic relationships and the numerical prefactor (identifying a prefactor smaller by 4.75 than the true value). SciMED correctly identified the equation and its numerical prefactor with an error of 0.1 but added a small constant term of 0.04. In this experiment, the prefactor is linked to a physical constant, the gravitational acceleration $g$ (for an explanation, see Appendix). Therefore, SciMED’s identification of a prefactor within a 0.76% error means it could accurately learn the value of $g$ that was used to construct the target from noisy data. This is considered a difficult task [78], that AI Feynman failed in.

In experiment D, where it is impossible to construct an equation for the target from the parameters of the data, SciMED resulted in an equation with the minimal MAE score it found, and AI Feynman failed to terminate even after 12 computation hours (with the Intel Core i7-1185G7 processor and Ubuntu 18.04 operation system), continuously adding terms to the equation it tried to match the data. It indicates an advantage of SciMED, as the outcome with poor SR performance and good AutoML performance alerts the user to re-examine the data. On the other hand, AI Feynman exhibited a common bloat issue that potentially leads to good performance scores by adding more terms to the equation but fails to generalize [79].

A noise analysis on SciMED is presented in Fig. Here, both SR components of SciMED were run 100 times for each randomly generated dataset with increasing noise levels. Namely, Fig. shows the percent of correct outputs that each SR component...
Table 4. The unknown equations found at each experiment by both SR components of SciMED and by AI Feynman

| Experiment | SciMED                          | AI Feynman                           |
|------------|---------------------------------|--------------------------------------|
| A          | $x_1 + x_2 \cdot x_3$          | $x_1 + x_2 \cdot x_3$                |
| B          | $1.31y_{30} \cdot y_{31} + 0.03$ | $y_{30} \cdot y_{31}$                |
| C          | $(12.98(z_1 - z_2)z_3)/(z_2z_4^2) + 0.04$ | $(8.33(z_2 + z_3)z_3)/(z_2(z_4 + z_4))$ |
| D          | $z_2/(z_1 z_3)$                | Failed to terminate (over 50 terms)  |

obtained for a given percentage of noise, divided into three types: noise on input parameters, noise on the target parameter, and noise on both. As expected, the more complex the unknown equation is, the more sensitive to noise SciMED becomes, as revealed by comparing the results of Experiments A and C. In addition, the Las Vegas-based SR performed better on higher noise levels than the GA-based SR component for both cases, as revealed by comparing the results in the first and second rows.

Figure 4. A noise analysis on SciMED showing the percent of correct outcomes (equations) that each SR component (red for the GA-based and blue for the Las Vegas-based) obtained from $n = 100$ runs, for a given percentage of noise, divided into noise on input variables, noise on the target variable, and noise on both.
Summary and conclusions

This work presents SciMED, a novel SR system combining the current state-of-the-art computational framework for physically-informed SR with a scientist-in-the-loop approach. SciMED emphasizes knowledge specific to its current task rather than attempting to apply general assumptions applicable to all physical equations. SciMED includes four components: 1) a novel GA-based feature selection that allows scientists to examine multiple hypotheses efficiently, 2) AutoML data extrapolation that facilitates the SR task, 3) GA-based SR, and 4) Las Vegas-based SR. SciMED’s structure is designed to allow a user to introduce domain-knowledge throughout the system’s pipeline, aiming to improve SciMED’s accuracy and reduce computational time by reducing the search space in several ways. In addition, users can switch off or on each component independently to suit their needs.

To facilitate quantitative benchmarking of our and other symbolic regression algorithms, we tested SciMED and AI Feynman on four cases, summarized in the Setup section. In the first two cases (experiments A-B), we highlighted the contribution of the GA-based SR and feature selection components. For these cases, it is not surprising that AI Feynman also demonstrated good performance, as it brute-forces all the polynomials up to a fourth-order, including the two linear configurations of these cases. Nevertheless, experiment B SciMED slightly outperformed AI Feynman, finding a more accurate numerical prefactor of the equation. In the next two cases (experiments C-D), we emphasized the contribution of the LV-based SR and AutoML components. In experiment C, SciMED significantly outperformed AI Feynman by finding the correct equation within a 0.76% error of the numerical prefactor, compared to AI Feynman that converged to a false equation (as summarized in Table 4). Furthermore, the deduction of an accurate prefactor, linked to the gravitational acceleration constant, from data with Gaussian noise poses a known challenge to SR [78] that SciMED succeeded in. Experiment D mimicked a possible scenario in which the user might fail to enter all the needed variables to explain the target. In such a case, an SR system should report its failure to converge to an equation of reasonable length rather than reporting a bloated equation of tens of variables that fails to generalize [79]. In this experiment SciMED alerted the user that there is a possible dependent variable missing from the data and presented the equation with the lowest MAE score it found. AI Feynman on the other hand, failed to terminate after significant computation resources were exerted and reached an unreasonable equation of over 50 terms. Both results are shown in Table 4.

We obtained the results of experiments A-D from data with noise introduced to the target variable to accord with previous work [16, 56]. In practice, there are three types of noise one can experience in real-world data; noise in the target variable, noise in the input variables, and noise in both variables. The latter two, which were rarely presented in prior work although common in practice, pose a more difficult challenge for SR as the amount of noise added to the target accumulates. A noise analysis on SciMED (experiment E) confirmed that: 1) SciMED is robust and withstands high levels of noise (compared to the levels tested in [16]) of all three types. 2) SciMED becomes increasingly sensitive to noise the more complex the unknown equation is (i.e., in terms of length or algebraic combinations of variables). 3) The Las Vegas-based SR performs better on higher noise levels than the GA-based SR component, meaning that the LV-SR component should be applied in case data is gathered with significant uncertainty.

We did not add equation structure hypotheses to the Las Vegas search, which would have significantly reduced the search space and increased the chances of quickly obtaining a good solution. Future work should examine this contribution in detail. In addition, further work is needed to analyze the feature selection component of
SciMED. To the best of our knowledge, allowing users to set distinct pairwise sets for the feature selection process is an unprecedented method of physical hypothesis evaluation that significantly reduces the search space. Thus, feature groups is a new efficient way for researchers to examine several hypotheses of the variables governing unknown dynamics that are otherwise unfeasible due to complex interactions between different feature groups.

Although the presented results are promising, SciMED has limitations. First, because SciMED’s main advantage is in the domain knowledge provided by the user, it is also its main limitation. Introducing false hypotheses may reduce the search space too much, making it more complicated or impossible to deduce the correct equation. For example, if during the search of the unknown equation $f_3$ from Table 2, the user falsely assumes the result should contain the delta between $z_1$ and $z_3$, SciMED should find a more complex term of $f_3 = (13.08(z_1 - z_3)z_3 - (z_2 - z_3)z_3)/(z_2z_3^2)$ instead of $f_3 = (13.08(z_1 - z_2)z_3)/(z_2z_3^2)$. One can partially remedy this issue by introducing meta-learning to SciMED’s pipeline. Specifically, one can train an ML model on data of expert user’s usage of SciMED that lead to positive results and generalize to similar tasks, thus, providing an initial recommendation for similar tasks [80, 81]. Second, SciMED becomes increasingly sensitive to noise in the data the more complex the unknown equation becomes, as shown in Fig. 4. Thus, a more robust regularization method, inspired by recent accomplishments of ML and deep learning techniques, should be integrated to tackle this difficulty [82, 83]. Alternatively, one can test the performance of SciMED on more case examples to better understand how the performance declines in the presence of each type of noise, concluding ground rules of performance. Third, the GA-based and Las Vegas-based SR components are as robust as the elementary functions provided (see Table 1). For example, in the current case, SciMED would not be able to discover a symbolic expression with a square root of a variable unless it is given as an additional variable in the dataset. Hence, finding an optimal set of elementary functions for SciMED can be of great interest.

**Declarations**

**Funding**

This research received no specific grant from public, commercial, or not-for-profit funding agencies.

**Data availability**

The data in this study is available in the manuscript with the relevant sources.

**Conflicts of interest**

The authors have no financial or proprietary interests in any material discussed in this article.

**Ethics approval statement**

NA

**Patient consent statement**

NA
Author Contributions

Liron Simon: Conceptualization, data curation, formal analysis, investigation, methodology, software, visualization, and writing - original draft.
Alex Liberzon: Conceptualization, supervision, validation, and writing - review & editing.
Teddy Lazebnik: Conceptualization, formal analysis, investigation, project administration, software, writing - original draft, and writing - review & editing.
References

[1] A. Rip and B. J. R. van der Meulen. “The post-modern research system”. In: Science and Public Policy 23.6 (1996), pp. 343–352.

[2] D. C. Miller and N. J. Salkind. Handbook of Research Design and Social Measurement. Sage Publishing, 2002.

[3] R. Sobh and C. Perry. “Research design and data analysis in realism research”. In: European Journal of Marketing 40.11 (2006), pp. 1194–1209.

[4] J. Michopoulos and S. Lambrakos. “On the Fundamental Tautology of Validating Data-Driven Models and Simulations”. In: 5th International Conference. Vol. 3515. Atlanta, GA, USA, 2005, pp. 1194–1209.

[5] W. Chua et al. “Data-driven discovery and validation of circulating blood-based biomarkers associated with prevalent atrial fibrillation”. In: European Heart Journal 40.16 (2019), pp. 1268–1276.

[6] M. Quade et al. “Prediction of dynamical systems by symbolic regression”. In: Physical Review E 94 (2016), p. 012214.

[7] Y. Chen, M. T. Angulo, and Y.-Y. Liu. “Revealing Complex Ecological Dynamics via Symbolic Regression”. In: BioEssays 41.12 (2019), p. 1900069.

[8] S. Stijven et al. “Prime-Time: Symbolic Regression Takes Its Place in the Real World”. In: Genetic Programming Theory and Practice XIII. Genetic and Evolutionary Computation (2016).

[9] P. Mahouti et al. “Symbolic Regression for Derivation of an Accurate Analytical Formulation Using ”Big Data”: An Application Example”. In: The Applied Computational Electromagnetics Society Journal 32.5 (2021), pp. 372–380.

[10] B. Can and C. Heavey. “Comparison of experimental designs for simulation-based symbolic regression of manufacturing systems”. In: Computers & Industrial Engineering 61.3 (2011), pp. 447–462.

[11] V. Aryadoust. “Application of evolutionary algorithm-based symbolic regression to language assessment: Toward nonlinear modeling”. In: Psychological Test and Assessment Modeling 57.3 (2015), pp. 301–337.

[12] P. D. Truscott and M. F. Korns. “Detecting Shadow Economy Sizes with Symbolic Regression”. In: Genetic Programming Theory and Practice IX. Springer New York, 2011, pp. 195–210.

[13] H. Vaddireddy et al. “Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data”. In: Physics of Fluids 32 (2020), p. 015113.

[14] B. V. Babu and S. Karthik. “Genetic Programming for Symbolic Regression of Chemical Process Systems”. In: Engineering Letters 14.2 (2007).

[15] P. Orzechowski, W. La Cava, and J. H. Moore. “Where are we now?: a large benchmark study of recent symbolic regression methods”. In: GECCO18: Proceedings of the Genetic and Evolutionary Computation Conference (2018).

[16] J. Zegklitz and P. Posik. “Benchmarking state-of-the-art symbolic regression algorithms”. In: Genetic Programming and Evolvable Machines 22 (2021), pp. 5–33.

[17] S. Weisberg. Applied Linear Regression. Wiley-Interscience, 2005.

[18] I. Boldina and P. G. Beninger. “Strengthening statistical usage in marine ecology: Linear regression”. In: Journal of Experimental Marine Biology and Ecology 474 (2016), pp. 81–91.
[19] H. J. Einhorn, D. N. Kleinmuntz, and B. Kleinmuntz. “Linear regression and process-tracing models of judgment”. In: *Psychological Review* 86.5 (1979), pp. 465–485.

[20] B. K. Slinker and S. A. Glantz. “Multiple linear regression is a useful alternative to traditional analyses of variance”. In: *Psychological Review* 255.3 (1988).

[21] E. F. Vonesh. “Non-linear models for the analysis of longitudinal data”. In: *Psychological Review* 11.14-15 (1992), pp. 1929–1954.

[22] M. P. Clements, P. H. Franses, and N. R. Swanson. “Forecasting economic and financial time-series with non-linear models”. In: *International Journal of Forecasting* 20.2 (2004), pp. 169–183.

[23] P. Royston. “A useful monotonic non-linear model with applications in medicine and epidemiology”. In: *International Journal of Forecasting* 19.15 (2000), pp. 2053–2066.

[24] M. Virgolin and S. P. Pissis. “Symbolic Regression is NP-hard”. In: *arXiv* (2022).

[25] D. Coppersmith and U. Vishkin. “Solving NP-hard problems in ”almost trees”: Vertex cover”. In: *Discrete Applied Mathematics* 10.1 (1985), pp. 27–45.

[26] W. J. Welch. “Algorithmic complexity: three NP-hard problems in computational statistics”. In: *Journal of Statistical Computation and Simulation* 15.1 (1982), pp. 17–25.

[27] D. S. Hochba. “Approximation Algorithms for NP-Hard Problems”. In: *SIGACT News* 28.2 (1997), pp. 40–52.

[28] D. P. Searson, D. E. Leahy, and M. J. Willis. “GPTIPS: an open source genetic programming toolbox for multigene symbolic regression”. In: *Proceedings of the International Multiconference of Engineers and Computer Scientists* 1 (2010), pp. 77–80.

[29] M. Quade et al. “Sparse identification of nonlinear dynamics for rapid model recovery”. In: *Chaos* 28 (2018), p. 063116.

[30] E. Alibekov, J. Kubalik, and R. Babuška. “Symbolic method for deriving policy in reinforcement learning”. In: *IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 2789–2795.

[31] L. Billard and E. Diday. “Symbolic regression analysis”. In: *Classification, clustering, and data analysis*. Springer, 2002, pp. 281–288.

[32] D. Clery and D. Voss. “All for one and one for all”. In: *Science* 308.5723 (2005), pp. 809–809.

[33] C. Chen, C. Luo, and Z. Jiang. “Elite bases regression: A real-time algorithm for symbolic regression”. In: *13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE. 2017, pp. 529–535.

[34] G. Kronberger, M. Kommenda, and M. Affenzeller. “Overfitting detection and adaptive covariant parsimony pressure for symbolic regression”. In: *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary computation*. 2011, pp. 631–638.

[35] P. A. Reinbold et al. “Robust learning from noisy, incomplete, high-dimensional experimental data via physically constrained symbolic regression”. In: *Nature Communications* 12.1 (2021), pp. 1–8.

[36] M. Schmidt and H. Lipson. “Distilling free-form natural laws from experimental data”. In: *Science* 324.5923 (2009), pp. 81–85.
[37] W. La Cava et al. “Contemporary symbolic regression methods and their relative performance”. In: *arXiv* (2021), p. 2107.14351.

[38] G. F. Smits and M. Kotanchek. “Pareto-front exploitation in symbolic regression”. In: *Genetic programming theory and practice II* (2005), pp. 283–299.

[39] Y. Wang, N. Wagner, and J. M. Rondinelli. “Symbolic regression in materials science”. In: *MRS Communications* 9.3 (2019), pp. 793–805.

[40] M. J. Heule and O. Kullmann. “The science of brute force”. In: *Communications of the ACM* 60.8 (2017), pp. 70–79.

[41] R. Riolo. *Genetic programming theory and practice X*. Springer, 2013.

[42] S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937.

[43] E. Kaiser, J. N. Kutz, and S. L. Brunton. “Sparse identification of nonlinear dynamics for model predictive control in the low-data limit”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2219 (2018), p. 20180335.

[44] N. M. Mangan et al. “Model selection for dynamical systems via sparse regression and information criteria”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2204 (2017), p. 20170009.

[45] A. A. Kaptanoglu et al. “PySINDy: A comprehensive Python package for robust sparse system identification”. In: *arXiv preprint arXiv:2111.08481* (2021).

[46] B. K. Petersen et al. “Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients”. In: *arXiv preprint arXiv:1912.04871* (2019).

[47] R. Salustowicz and J. Schmidhuber. “Probabilistic incremental program evolution”. In: *Evolutionary Computation* 5.2 (1997), pp. 123–141.

[48] K. Sastry and D. E. Goldberg. “Probabilistic model building and competent genetic programming”. In: *Genetic Programming Theory and Practice*. Springer, 2003, pp. 205–220.

[49] K. Yanai and H. Iba. “Estimation of distribution programming based on bayesian network”. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC’03*. Vol. 3. IEEE. 2003, pp. 1618–1625.

[50] E. Hemberg et al. “An investigation of local patterns for estimation of distribution genetic programming”. In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. 2012, pp. 767–774.

[51] Y. Shan et al. “Grammar model-based program evolution”. In: *Proceedings of the 2004 Congress on Evolutionary Computation*. Vol. 1. IEEE. 2004, pp. 478–485.

[52] P. A. Bosman and E. D. d. Jong. “Learning probabilistic tree grammars for genetic programming”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2004, pp. 192–201.

[53] P.-K. Wong et al. “Grammar-based genetic programming with Bayesian network”. In: *2014 IEEE Congress on Evolutionary Computation*. IEEE. 2014, pp. 739–746.

[54] L. F. D. P. Sotto and V. V. de Melo. “A probabilistic linear genetic programming with stochastic context-free grammar for solving symbolic regression problems”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2017, pp. 1017–1024.
T. Stephens. *Genetic Programming in Python With a Scikit-Learn Inspired API: Gplearn*. 2016.

S.-M. Udrescu and M. Tegmark. “AI Feynman: A physics-inspired method for symbolic regression”. In: *Science Advances* 6.16 (2020), eaay2631.

S.-M. Udrescu et al. “AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4860–4871.

F. L. Litvin. “Application of theorem of implicit function system existence for analysis and synthesis of linkages”. In: *Mechanism and Machine Theory* 15.2 (1980), pp. 115–125.

J. J. Monaghan. “Implicit SPH drag and dusty gas dynamics”. In: *Journal of Computational Physics* 138.2 (1997), pp. 801–820.

B. Burlacu, G. Kronberger, and M. Kommenda. “Operon C++ an efficient genetic programming framework for symbolic regression”. In: *Proceedings of 2020 Genetic and Evolutionary Computation Conference Companion*. 2020, pp. 1562–1570.

M. Virgolin et al. “Improving model-based genetic programming for symbolic regression of small expressions”. In: *Evolutionary Computation* 29.2 (2021), pp. 211–237.

D. J. Montana. “Strongly typed genetic programming”. In: *Evolutionary Computation* 3.2 (1995), pp. 199–230.

M. O’Neill and C. Ryan. “Grammatical evolution”. In: *IEEE Transactions on Evolutionary Computation* 5.4 (2001), pp. 349–358.

R. I. McKay et al. “Grammar–based genetic programming: a survey”. In: *Genetic Programming and Evolvable Machines* 11.3 (2010), pp. 365–396.

Z. W. Bo, L. Z. Hua, and Z. G. Yu. “Optimization of process route by genetic algorithms”. In: *Robotics and Computer-Integrated Manufacturing* 22 (2006), pp. 180–188.

A. B. A. Hassanat and E. Alkafaween. “On Enhancing Genetic Algorithms Using New Crossovers”. In: *International Journal of Computer Applications in Technology* 55.3 (2017).

Q. Chen and B. Xue. “Generalisation in Genetic Programming for Symbolic Regression: Challenges and Future Directions”. In: *Women in Computational Intelligence: Key Advances and Perspectives on Emerging Topics*. Springer International Publishing, 2022, pp. 281–302.

R. S. Olson and J. H. Moore. “TPOT: A tree-based pipeline optimization tool for automating machine learning”. In: *Workshop on Automatic Machine Learning*. PMLR. 2016, pp. 66–74.

F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

R. Kohavi. “A Study of Cross Validation and Bootstrap for Accuracy Estimation and Model Select”. In: *International Joint Conference on Artificial Intelligence*. 1995.

J. Wan et al. “Improvement of machine learning enhanced genetic algorithm for nonlinear beam dynamics optimization”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 946 (2019), p. 162683.
[72] V. Sathia, V. Ganesh, and S. R. T. Nanditale. “Accelerating Genetic Programming using GPUs”. In: arXiv (2021).

[73] J. McCarthy. “LISP: A Programming System for Symbolic Manipulations”. In: 14th National Meeting of the Association for Computing Machinery. Association for Computing Machinery, 1959, pp. 1–4.

[74] L. Bar and N. Sochen. “Strong Solutions for PDE-Based Tomography by Unsupervised Learning”. In: SIAM Journal on Imaging Sciences 14.1 (2021), pp. 128–155.

[75] R. Liu et al. “Optimizing the Hyper-parameters for SVM by Combining Evolution Strategies with a Grid Search”. In: Intelligent Control and Automation 344 (2006).

[76] B. Zang et al. “An Improved KNN Algorithm Based on Minority Class Distribution for Imbalanced Dataset”. In: 2016 International Computer Symposium (ICS). 2016, pp. 696–700.

[77] D. Krongauz and T. Lazebnik. “Collective Evolution Learning Model for Vision-Based Collective Motion with Collision Avoidance”. In: bioRxiv (2022).

[78] B. M. de Silva et al. “Discovery of Physics From Data: Universal Laws and Discrepancies”. In: Frontiers in Artificial Intelligence 3 (2020), p. 25.

[79] G. Dick. “Bloat and generalisation in symbolic regression”. In: Asia-Pacific Conference on Simulated Evolution and Learning. Springer. 2014, pp. 491–502.

[80] M. M. Afsar, T. Crump, and B. Far. “Reinforcement Learning Based Recommender Systems: A Survey”. In: ACM Computing Surveys (2022).

[81] M. Huisman, J. N. van Rijn, and A Plaat. “A survey of deep meta-learning”. In: Artificial Intelligence Review 54 (2021), 4483–4541.

[82] X. Lu et al. “Probabilistic Regularized Extreme Learning Machine for Robust Modeling of Noise Data”. In: IEEE Transactions on Cybernetics 48.8 (2018), pp. 2368–2377.

[83] V. Raychev et al. “Learning Programs from Noisy Data”. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. Association for Computing Machinery, 2016, pp. 761–774.

[84] G Elert. The Physics Hypertextbook, Equations of Motion.

[85] F. M. White and J. Majdalani. Viscous fluid flow. Vol. 3. McGraw-Hill New York, 2006.