Beyond Size-Based Metrics: Measuring Task-Specific Complexity in Symbolic Regression

Krzysztof Kacprzyk
University of Cambridge

Mihaela van der Schaar
University of Cambridge

Abstract

Symbolic regression (SR) is a machine learning approach aimed at discovering mathematical closed-form expressions that best fit a given dataset. Traditional complexity measures in SR, such as the number of terms or expression tree depth, often fail to capture the difficulty of specific analytical tasks a user might need to perform. In this paper, we introduce a new complexity measure designed to quantify the difficulty of conducting single-feature global perturbation analysis (SGPA)—a type of analysis commonly applied in fields like physics and risk scoring to understand the global impact of perturbing individual input features. We present a unified mathematical framework that formalizes and generalizes these established practices, providing a precise method to assess how challenging it is to apply SGPA to different closed-form equations. This approach enables the definition of novel complexity metrics and constraints directly tied to this practical analytical task. Additionally, we establish a reconstruction theorem, offering potential insights for developing future optimization techniques in SR.

1 INTRODUCTION

Symbolic Regression Closed-form expressions are mathematical expressions consisting of a finite number of variables, constants, and some basic functions (e.g., $\sin, \exp, \log$) connected by binary arithmetic operations ($+, -, \times, \div$) and function composition. They are often represented as expression trees where each internal node is a binary operator or some basic function, and each terminal node is a variable or a constant (Schmidt and Lipson, 2009). These expressions are vital in depicting various phenomena and have been instrumental in advancing scientific understanding. Recently, machine learning (ML) has been used to automate the process of discovering such equations from data, giving rise to the area of symbolic regression (SR) (La Cava et al., 2021). SR has been employed in many areas such as physics (Udrescu and Tegmark, 2020), medicine (Alaa and van der Schaar, 2019), biology (Chen et al., 2019), and material science (Wang et al., 2019).

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

Need for Complexity Quantification The main advantage of using symbolic regression instead of other black-box methods is the supposed human-readable representation of the model that allows for performing some downstream tasks, e.g., debugging, editing, verification, and analysis. However, many contemporary algorithms tend to produce equations with hundreds of components (La Cava et al., 2021) raising concerns about their utility. As determining utility is challenging if the complexity measure employed is a poor proxy of the downstream task (de Franca et al., 2023), there is a pressing need for better complexity measures (Virgolin et al., 2022) and constraints. This is becoming increasingly important as SR is more broadly used in safety-critical domains (Alaa and van der Schaar, 2019; Guidetti et al., 2023).

Currently Used Complexity Measures Traditional complexity measures can be divided into size-based and semantic measures. Size-based measures are designed to measure how *compact* the equation is and include metrics based on the depth of the expression tree (Cranmer, 2020), the number of terms (Stephens, 2022), and the description length (Udrescu et al., 2021). That also includes sparsity measures employed in methods representing equations as neural networks (Martius and Lampert, 2017; Sahoo et al., 2018). These metrics quantify how compact the equation is but they may not directly correspond to the difficulty of performing tasks such as editing or debugging (further discussed in

Appendix B.7). These metrics also disregard the nature and order of the operations performed. Although this is addressed by the semantic measures, these approaches are designed to measure non-linearity (Vladislavleva et al., 2009) or to prevent bloat (Vanneschi et al., 2010; Kommenda et al., 2015) rather than to allow for a specific type of analysis. Moreover, most of the current complexity metrics do not take into account variable interactions. They would have the same value if all variables were exchanged for the same single variable. We say they are not *interaction-aware*. They also provide a single numeric value for the whole equation, while complexity may be a more nuanced notion. An orthogonal research direction is learning the complexity measure automatically from human feedback (Virgolin et al., 2020, 2021).

Task-Specific Complexity Measure In contrast to currently used metrics, we want to measure the complexity of an equation by assessing the difficulty of performing a specific analytical task. We call this task Single-Feature Global Perturbation Analysis (SGPA). SGPA looks globally at how the model's prediction changes as we perturb a particular input feature while keeping all other features fixed. Such analysis may be helpful, for instance, in debugging and model refinement by observing that the model does not behave appropriately under a particular feature change. For instance, lowering the predicted mortality risk after adding information about the patient's asthma (Caruana et al., 2015) or significantly increasing serum creatinine level (Lengerich et al., 2022). In general, SGPA may be challenging if the effect of perturbing a single variable depends on many other variables. Sometimes, however, the effect of perturbing a single feature is simple and depends only on the perturbed amount. For instance, in a linear regression model, $y = c_0 + \sum_{n=1}^{N} c_n x_n$, *increasing* x_n by λ *increases* y by λc_n. Thus, the effect is independent of other features. In this work, we introduce a mathematical formalism that allows us to measure the intricacy of these effects to assess how difficult SPGA is to perform, and thus assign a complexity measure to the equation.

Performing SGPA Although our work aims to ensure that all equations found by SR can be analyzed through SGPA, we do not impose how exactly this analysis is supposed to be performed. Usually, this can be done directly by inspecting the equation (see Section 2) and plotting some functions (see Section 5). It may seem that some of the eXplainable AI (XAI) (Barredo Arrieta et al., 2020) techniques, e.g., PD plots (Friedman, 2001), can be helpful for this kind of analysis. We discuss in Appendix B.4 how they are likely to be insufficient and thus superfluous for performing SGPA if one has access to the underlying equation.

Contributions and Outline We introduce a new way of measuring the complexity of closed-form equations based on the difficulty of performing a specific task that we call SGPA. We describe SGPA in Section 2 and demonstrate how it is implicitly utilized in areas ranging from physics to risk scoring. We develop a unified mathematical framework that formalizes SGPA through the introduction of *single-feature perturbation operators* and *interaction functions* (Section 3). Using this language, we can perform the following.

- Capture the expressivity limit of the "simplest equations" and justify the appeal of linear regression, the Cox model, and many physics equations. (Section 4)
- Characterize the difficulty of SGPA on any closed-form expression using *univariate perturbations* and *interaction decomposition*. (Sections 5 and 6)
- Design novel SR complexity measures and constraints based on the difficulty of performing SGPA that are interaction-aware and variable-specific. (Section 7)
- Prove a reconstruction theorem with potential implications for future SR algorithms. (Section 8)

2 SINGLE-FEATURE GLOBAL PERTURBATION ANALYSIS

SGPA looks at each feature individually and considers the global impact of changing (perturbing) it while all the other features are fixed. We demonstrate that this approach has already been implicitly utilized in areas such as physics or risk scoring to detect undesired model behaviors and draw insights about the training dataset. Hence, we motivate and illustrate SGPA using examples from these areas.

Perturbations in Physics Let us take a look at Equation (1) describing the potential energy of an object U with mass m at altitude z and with gravitational acceleration g.

$$U = mgz \tag{1}$$

When developing a model like this, performing a few thought experiments to confirm its soundness and agreement with prior knowledge and intuition is often instrumental. For instance, we would expect the potential energy to be equal to 0 if the object has zero mass or the altitude is zero. Then, we can observe that the energy and mass are proportional. If we *multiply* the mass by two, the energy gets *multiplied* by two. The same applies to the altitude (z) and the acceleration (g). This lets us draw insights about the studied system and make conjectures. For example, as the gravitational acceleration (g) on the moon is only 0.6 of what it is on Earth, the same can be said about the potential energy.

The preceding examples follow a similar structure. Let's consider a variable x and what would happen to the target y if x is replaced by $\lambda \times x$ for some $\lambda \in \mathbb{R}$. In particular, if m is replaced by $\lambda \times m$, then U is replaced by $\lambda \times U$. This is one type of SGPA. We write it as follows.

$$m \mapsto \lambda \times m \implies U \mapsto \lambda \times U \tag{2}$$

We refer to *perturbations* because we consider the impact of perturbing m by *multiplying* it by some $\lambda \in \mathbb{R}$. We use the term *global* because the resulting relationship is true for all triples (m, g, z).

Perturbations in Risk Scores We argue that SGPA is commonly used to assess whether a risk model performs as intended. In fact, the most common model in risk prediction, the Cox model (Cox, 1972, 1975), is ideally suited for SGPA. Consider the covariate part of the Cox model in Equation (3).

$$y = e^{a_1 x_1 + a_2 x_2 + b} \tag{3}$$

If we *increase* x_1 by λ then y is going to get *multiplied* by $e^{\lambda a_1} = (e^{a_1})^\lambda$. We can write it as follows.

$$x_1 \mapsto x_1 + \lambda \implies y \mapsto (e^{a_1})^\lambda \times y \tag{4}$$

This is how risk models are interpreted. For instance, "When the patient's age *increases* by one year, the risk is 1.1 *times higher*" (in this case $e^{a_1} = 1.1$). The Cox model is perfectly suited for SGPA because any *additive* perturbation of any of the features results in a *multiplicative* perturbation of the predicted risk score independent of the value of any of the features.

Perturbations in Linear Regression Consider a linear model $y = a_1 x_1 + a_2 x_2 + b$. Applying SGPA to it is straightforward. If we *increase* x_1 by λ, y will *increase* by $a_1\lambda$. Using our notation:

$$x_1 \mapsto x_1 + \lambda \implies y \mapsto y + a_1\lambda \tag{5}$$

Symmetrically, we could also consider an *additive* change in y given a *multiplicative* change in the input. For instance, $y = log(x_1 x_2)$ can be analyzed as follows.

$$x_1 \mapsto \lambda \times x_1 \implies y \mapsto y + \log(\lambda) \tag{6}$$

3 SINGLE-FEATURE PERTURBATION OPERATOR

In this section, we develop a unified mathematical framework that generalizes and formalizes the types of analyses described in Section 2 through the introduction of single-feature perturbation operators and interaction functions, whose complexity is a crucial factor in SGPA.

Interaction Function All examples in the previous section had the same underlying form.

$$x_n \mapsto x_n \ {}^{+}\!/\!_{\times}\ \lambda \implies y \mapsto y \ {}^{+}\!/\!_{\times}\ h(\lambda), \tag{7}$$

where $y = y(x_1, \ldots, x_N)$, $h : \mathbb{R} \to \mathbb{R}$ and we use ${}^{+}\!/\!_{\times}$ to denote "$+$ or $\times$". We focus on addition and multiplication because these operations are intuitive for many people, and together with their inverses $(-, \div)$, they often constitute all binary operators used in closed-form expressions. We claim that the appeal of all the models in Section 2 results from simple h. In particular, h is a univariate function. This, however, is not always true. In general, the amount added to y (or by which y is multiplied) can depend on λ and any subset of $\{x_1, \ldots, x_N\}$. In general, SGPA is written as:

$$x_1 \mapsto x_1 \ {}^{+}\!/\!_{\times}\ \lambda \implies y \mapsto y \ {}^{+}\!/\!_{\times}\ h[\lambda](\boldsymbol{x}), \tag{8}$$

where $h : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}$ is called an *interaction function*. To formally define an interaction function, we first introduce a helpful notation for a general arithmetic operation and single-feature perturbation.

Notation We often refer to the binary operators $\{+, -, \times, \div\}$ or more narrowly to $\{+, \times\}$. If we make a more general statement, we use the symbol $\circledast$ to refer to a binary operator. Any of these symbols can be used in infix notation (e.g., $a + b$ or $a \circledast b$) or prefix notation (e.g., $\times(a, b) = a \times b$, $\circledast(a, b) = a \circledast b$). For $\circledast \in \{+, \times\}$, we denote the inverse operation by $\circledast^{-1}$, i.e., $\circledast^{-1} = -$ for $\circledast = +$ and $\circledast^{-1} = \div$ for $\circledast = \times$. Sometimes, we find it useful to write $\circledast[a] : \mathbb{R} \to \mathbb{R}$ to denote a univariate function defined as $\circledast[a](b) = a \circledast b$.

The following notation is useful for describing multiplying or increasing by λ a single vector element.

Definition 1 (Single-feature perturbation)**.** Let $\circledast \in \{+, -, \times, \div\}$ and $N \in \mathbb{N}$. A single-feature perturbation of $\circledast$ with respect to x_n is a function $p_n^{\circledast} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ defined as

$$p_n^{\circledast}(\boldsymbol{x}, \lambda) = (x_1, \ldots, \circledast(x_n, \lambda), x_{n+1}, \ldots, x_N). \tag{9}$$

Single-Feature Perturbation Operator To formally define the interaction function, we first rewrite Equation (8) using single-feature perturbation and a new $\circledast$ notation instead of ${}^{+}\!/\!_{\times}$. In particular, we use $\circledast_1$ to denote perturbation of the output and $\circledast_2$ to denote perturbation of the input. Recall that $y \ {}^{+}\!/\!_{\times}\ h[\lambda](\boldsymbol{x})$ refers to y after x_n is perturbed by λ, i.e., $\boldsymbol{x} \mapsto p_n^{\circledast_2}(\boldsymbol{x}, \lambda)$, where $\circledast_2 \in \{+, \times\}$. Thus

$y(\boldsymbol{x}) \ast_1 h[\lambda](\boldsymbol{x}) = y(p_n^{\ast_2}(\boldsymbol{x}, \lambda))$, where $\ast_1 \in \{+, \times\}$. By rearranging, we arrive at the following.

Definition 2 (Single-feature perturbation operator)**.** Let $\ast_1, \ast_2 \in \{+, \times\}$ be two arithmetic operations. Let $f : \mathbb{R}^N \to \mathbb{R}$ be any function and denote its arguments (inputs) as $\boldsymbol{x} = (x_1, \ldots, x_N)$. Single-feature perturbation operator with respect to x_n is denoted by $\frac{\ast_1}{\ast_2 x_n}$. It maps a function f to an *interaction function* $\frac{\ast_1 f}{\ast_2 x_n} : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}$ defined as

$$\frac{\ast_1 f}{\ast_2 x_n}[\lambda](\boldsymbol{x}) = \ast_1^{-1}(f(p_n^{\ast_2}(\boldsymbol{x}, \lambda)), f(\boldsymbol{x})). \tag{10}$$

There are four types of single-feature perturbation operators for each possible pair of $(\ast_1, \ast_2) \in \{+, \times\}^2$.

1. $\frac{+f}{+x_n}[\lambda](\boldsymbol{x}) = f(p_n^{+}(\boldsymbol{x}, \lambda)) - f(\boldsymbol{x})$
2. $\frac{+f}{\times x_n}[\lambda](\boldsymbol{x}) = f(p_n^{\times}(\boldsymbol{x}, \lambda)) - f(\boldsymbol{x})$
3. $\frac{\times f}{+x_n}[\lambda](\boldsymbol{x}) = f(p_n^{+}(\boldsymbol{x}, \lambda))/f(\boldsymbol{x})$
4. $\frac{\times f}{\times x_n}[\lambda](\boldsymbol{x}) = f(p_n^{\times}(\boldsymbol{x}, \lambda))/f(\boldsymbol{x})$

Remark 1. If clear from the context, we often omit the explicit dependence on $\boldsymbol{x}$ or include the subset of $\{x_1, \ldots, x_N\}$ with only the relevant variables.

Example 1. Below, we provide the interaction functions for the examples described in Section 2.

- For $U(m, g, z) = mgz$, $\frac{\times U}{\times m}[\lambda] = \lambda$
- For $y(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$, $\frac{+y}{+x_n}[\lambda] = \lambda a_n$
- For $y(x_1, x_2) = e^{a_1 x_1 + a_2 x_2 + b}$, $\frac{\times y}{+x_n}[\lambda] = e^{\lambda a_n}$
- For $y(x_1, x_2) = \log(x_1 x_2)$, $\frac{+y}{\times x_n}[\lambda] = \log(\lambda)$

We omit the dependence on m, g, z, x_1, x_2 for conciseness.

Every function has, for each variable, four different interaction functions associated with it (for each of the four perturbation operators). Some operators are more *natural* than others. For instance, for $y = e^{ax}$, $\frac{\times}{+x}$ can be considered more natural than $\frac{+}{+x}$ which would produce a more complex interaction function $e^{a(x+\lambda)} - e^{ax}$. Sometimes, there is a clear natural operator—when the interaction function depends only on λ—but this is less obvious for complicated expressions. We demonstrate a way to differentiate between complex interaction functions in Section 6 and provide a more thorough discussion in Appendix B.2.

Although all interaction functions in Example 1 depended only on λ, they may also depend on other variables. We call the number of relevant variables (excluding λ) the *degree* of the interaction function.

Definition 3 (Degree)**.** Let $h = \frac{\ast_1 f}{\ast_2 x_n}$ be an interaction function. x_i is an *active variable* of h if there exist $a, b \in \mathbb{R}$ such that h with x_i fixed at a is different from h with x_i fixed at b at at least one point. The degree of h denoted $\deg(h)$, is the number of active variables excluding λ.

It follows that all interaction functions in Example 1 have degree 0. Thus, these were the simplest interaction functions according to their degree. This begs the question:

> **Q:** Which closed-form expressions have the simplest (only dependent on λ) interaction functions?

This is what we explore in the next section.

4 SIMPLEST INTERACTION FUNCTIONS

The simplest interaction functions, according to their degree, are the ones that depend only on λ (they have degree 0). Examples of such functions were shown in Example 1. In this section, we investigate all possible interaction functions of the form $\frac{\ast_1 f}{\ast_2 x_n}[\lambda](\boldsymbol{x}) = h[\lambda]$, where $h : \mathbb{R} \to \mathbb{R}$ does not depend on $\boldsymbol{x}$. There is only one such family of functions for each of the four types of perturbation operators analogous to the ones in Example 1.

Theorem 1 (Classification Theorem for 0th-Degree Interaction Functions)**.** *Let $f : \mathbb{R}^N \to \mathbb{R}$.*[1] *Then, the following are the only 0th-degree interaction functions for each of the four perturbation operators.*

- $\frac{+f}{+x_n}[\lambda] = a\lambda$*, where $a \in \mathbb{R}$ is any constant*
- $\frac{+f}{\times x_n}[\lambda] = a\log(\lambda)$*, where $a \in \mathbb{R}$ is any constant*
- $\frac{\times f}{+x_n}[\lambda] = a^{\lambda}$*, where $a \in \mathbb{R}_{\geq 0}$ is any constant*
- $\frac{\times f}{\times x_n}[\lambda] = \lambda^{a}$*, where $a \in \mathbb{R}$ is any constant*

We use $\frac{\ast_1}{\ast_2} I_a$ to denote these functions such that $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = \frac{\ast_1}{\ast_2} I_a(\lambda)$, where $a \in \mathbb{R}$ is the parameter.

Proof. See Appendix A.1. □

We can characterize all functions with 0th-degree interaction functions with respect to all variables.

Theorem 2. *Let $\ast_1, \ast_1', \ldots, \ast_N, \ast_N' \in \{+, \times\}$ and $f : \mathbb{R}^N \to \mathbb{R}$. If $\forall n \in [N]$ $\deg\left(\frac{\ast_n f}{\ast_i' x_n}\right) = 0$ then*

- $\ast_1 = \ldots = \ast_N = \ast$*, where $\ast \in \{+, \times\}$,*

- $$f(\boldsymbol{x}) = \begin{cases} a_0 + \sum_{n=1}^{N} h_n(x_n) & \textit{if } \ast = + \\ a_0 \prod_{n=1}^{N} h_n(x_n) & \textit{if } \ast = \times \end{cases},$$

[1] All functions are assumed to be continuous almost everywhere

- $h_n(x_n) = \begin{cases} a_n x_n & \textit{if } \ast = + \wedge \ast'_n = + \\ a_n \log(x_n) & \textit{if } \ast = + \wedge \ast'_n = \times \\ x_n^{a_n} & \textit{if } \ast = \times \wedge \ast'_n = \times \\ e^{a_n x_n} & \textit{if } \ast = \times \wedge \ast'_n = + \end{cases}$,
- $\forall n \in \mathbb{N}_0 \ a_n \in \mathbb{R}$.

Proof. See Appendix A.2. □

Takeaway: All functions which have a 0th-degree interaction function with respect to all variables are either sums or products of the corresponding functions h_n. Nearly all of them[a] can be interpreted as generalized additive models (Hastie and Tibshirani, 1986) where the inverse of the *link function* is either identity or exp and where each *shape function* is either $a_n x_n$ or $a_n \log(x_n)$.

[a]Exceptions include functions such as $f(x,y) = x^2 y^{-3}$ where x and y can be negative, i.e., when $h_n(x_n) = x_n^{a_n}$ for an integer a_n, and x_n is allowed to be negative.

Often, such simple models are insufficient to capture more complicated relationships. In the next section, we investigate 1st-degree interaction functions. In particular, we focus on closed-form equations containing some well-known univariate functions.

5 UNIVARIATE FUNCTIONS

All functions in the previous section had a 0th-degree interaction function. However, often, this is not the case. In particular, "well-known" functions in closed-form expressions (e.g., trigonometric, exponential, logarithmic, square roots) can greatly complicate the interaction function. Let us consider $y(x) = \sin(x)$. All four possible interaction functions depend not only on λ but also on x itself. Thus making them first-degree interaction functions. In turn, how amenable y is for SGPA depends on familiarity with the sin function. A visual inspection of the line plot depicting sin may be beneficial for comprehending, for instance, by how much sin gets *multiplied* when its argument gets *multiplied* by λ (for $\frac{\times}{\times x}$) or by how much sin *increases* when its argument *increases* by λ (for $\frac{+}{+x}$). The following notation is useful when univariate functions are involved.

Definition 4 (Univariate perturbation)**.** Let $u : \mathbb{R} \to \mathbb{R}$ be a univariate function. The interaction function of u is called a *univariate perturbation* and is denoted $\frac{\ast_1 u}{\ast_2}$ (we do not specify the argument). As for perturbation operators, there are four types of univariate perturbations. Note, $\deg(\frac{\ast_1 u}{\ast_2}) \leq 1$.

Example 2. Consider $y = x_1 \sin(x_2)$. Using univariate perturbation, we can write the interaction function of y with respect to x_2 as $\frac{\times y}{\times x_2}[\lambda] = \frac{\times \sin}{\times}[\lambda](x_2)$, making it clear that understanding the effect of perturbing x_2 requires only understanding sin.

Example 3. Consider $y = x_1 + \frac{1}{\sqrt{1+x_2^2}}$. We can write the interaction functions of y with respect to x_2 as $\frac{+y}{+x_2}[\lambda] = \frac{+u}{+}[\lambda](x_2)$, where $u = \frac{1}{\sqrt{1+x_2^2}}$, making it clear that understanding the effect of perturbing x_2 requires only understanding u (which can be done, for instance, by plotting it).

We have shown how the degree of interaction functions of univariate functions are at most 1 (it may be 0 for some univariate functions, e.g., exp) and how they can usually be understood by looking at plots of the functions themselves. This begs the question:

Q: Can all first-degree interaction functions be understood through a plot of a univariate function?

The answer is yes, and this is what we show in the following theorem.

Theorem 3. *Let $f : \mathbb{R}^N \to \mathbb{R}$ and let $G : \mathbb{R}^2 \to \mathbb{R}$. If $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = G(x_n, \lambda)$ then $G(x_n, \lambda) = \frac{\ast_1 g}{\ast_2}[\lambda](x_n)$ for some $g : \mathbb{R} \to \mathbb{R}$. If $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = G(x_m, \lambda)$, where $m \neq n$ then $G(x_m, \lambda) = \frac{\ast_1}{\ast_2} I_{g(x_m)}(\lambda)$ where $\frac{\ast_1}{\ast_2} I$ is defined in Theorem 1 and $g : \mathbb{R} \to \mathbb{R}$ is some univariate function.*

Proof. See Appendix A.1. □

Takeaway: Every first-degree interaction function can be comprehended by looking at a graph of the corresponding univariate function.

6 SEQUENTIAL ANALYSIS THROUGH INTERACTION DECOMPOSITION

So far, we have introduced one quantitative measure of the complexity of an interaction function - its degree. However, the degree of the interaction function is by itself (beyond the first degree) an imperfect measure of how amenable a particular equation is for SGPA. Even equations with high-degree interaction functions may be analyzed by decomposing an expression into constituent parts and understanding them from the ground up. In this section, we develop a formalism for *decomposing* interaction functions that highlights how an equation can be analyzed and how complex this analysis is. Consider the equation for a difference in gravitational potential energies.

$$U(m_1, m_2, r_1, r_2) = G m_1 m_2 \left(\frac{1}{r_2} - \frac{1}{r_1}\right) \quad (11)$$

Interaction functions of U with respect to m_1 and m_2 are straightforward: $\frac{\times U}{\times m_1}[\lambda] = \lambda$, $\frac{\times U}{\times m_2}[\lambda] = \lambda$ However, interaction functions of U with respect to r_1 and r_2 are more complicated. All four possible interaction functions have deg equal to 3 or 2 and look quite complex.

$$\frac{+U}{+r_2}[\lambda] = Gm_1m_2\left(\frac{1}{r_2+\lambda} - \frac{1}{r_2}\right) \tag{12}$$

$$\frac{+U}{\times r_2}[\lambda] = Gm_1m_2\left(\frac{1}{\lambda r_2} - \frac{1}{r_2}\right) \tag{13}$$

$$\frac{\times U}{+r_2}[\lambda] = \frac{(\frac{1}{r_2+\lambda} - \frac{1}{r_1})}{(\frac{1}{r_2} - \frac{1}{r_1})} \tag{14}$$

$$\frac{\times U}{\times r_2}[\lambda] = \frac{(\frac{1}{\lambda r_2} - \frac{1}{r_1})}{(\frac{1}{r_2} - \frac{1}{r_1})} \tag{15}$$

This seems to contradict our intuition. The expression for U does not look as complicated as these interaction functions make it look because humans do not try to analyze the whole equation at once but rather decompose it into simpler objects (Newell et al., 1958). Moreover, although the original form is very convenient for understanding the effect of m_1 and m_2 (which have elementary interaction functions), it is not the most useful form for understanding the impact of r_2. We claim this expression is much better interpreted as a difference between two potentials. In particular, $U = U_2 - U_1$, and $U_i = \frac{Gm_1m_2}{r_i}$. Then, we can analyze the impact of r_2 in two steps. First, we recognize that U_2 is inversely proportional to r_2. That means $\frac{\times U_2}{\times r_2}[\lambda] = \frac{1}{\lambda}$. As mentioned earlier, this is one of the simplest interaction functions, as the output is just a function of λ. Then, we can analyze the impact of multiplying U_2 by λ on U. We can write it as follows.

$$r_2 \mapsto \lambda r_2 \implies U_2 \mapsto \frac{U_2}{\lambda} \tag{16}$$

$$U_2 \mapsto \lambda U_2 \implies U \mapsto U + (\lambda - 1)U_2 \tag{17}$$

$(\lambda - 1)U_2$ has degree 1 with respect to U_2. That makes it a relatively simple interaction function with intuitive interpretation, e.g., if U_2 is multiplied by $\lambda = 1.5$, then U increases by 50% of U_2.

We would like to mathematically express the fact that $\frac{+U}{\times r_2}$ can be analyzed by introducing new variables (U_1, U_2) and analyzing it through $\frac{+U}{\times U_2}$ and $\frac{\times U_2}{\times r_2}$. We can achieve it by *interaction composition.*

Definition 5 (Interaction composition)**.** Let $y = f(\boldsymbol{u})$, where $N \in \mathbb{N}$, and $f : \mathbb{R}^N \to \mathbb{R}$. Each u_n is described by $u_n = g_n(\boldsymbol{x})$, where $M \in \mathbb{N}$ and each $g_n : \mathbb{R}^M \to \mathbb{R}$. Then for any $k \in [N]$ and $l \in [M]$ we denote the interaction composition of $\frac{\circledast_1 y}{\circledast_2 u_k}$ and $\frac{\circledast_2 u_k}{\circledast_3 x_l}$ as $\frac{\circledast_1 y}{\circledast_2 u_k} \otimes \frac{\circledast_2 u_k}{\circledast_3 x_l}$ and define it as

$$\begin{gathered}\left(\frac{\circledast_1 y}{\circledast_2 u_k} \otimes \frac{\circledast_2 u_k}{\circledast_3 x_l}\right)[\lambda](\boldsymbol{x}) = \\ \frac{\circledast_1 y}{\circledast_2 u_k}\left[\frac{\circledast_2 u_k}{\circledast_3 x_l}[\lambda](\boldsymbol{x})\right](\boldsymbol{u}(\boldsymbol{x})).\end{gathered} \tag{18}$$

Using our new notation, we can write $\frac{+U}{\times r_2} = \frac{+U}{\times U_2} \otimes \frac{\times U_2}{\times r_2}$, where $\frac{+U}{\times U_2}[\lambda] = (\lambda - 1)U_2$ has degree 1 with respect to U_2 and $\frac{\times U_2}{\times r_2}[\lambda] = \frac{1}{\lambda}$ has degree 0.

Example 4. Consider a different physics equation from the area of diffraction, $\theta = \arcsin(\frac{l}{nd})$. Here, $\deg(\frac{\times \theta}{\times d}) = 3$, which suggests that it is a complex equation to analyze. However, we can rewrite it as $\frac{\times \theta}{\times d} = \frac{\times \theta}{\times z} \otimes \frac{\times z}{\times d}$, where $z = \frac{l}{nd}$. Thus, this equation can be analyzed sequentially, and each interaction function has at most first degree as $\deg(\frac{\times \theta}{\times z}) = \deg(\frac{\times \arcsin}{\times}) = 1$ and $\deg(\frac{\times z}{\times d}) = 0$.

Takeaway: Interaction composition allows us to characterize how much effort is required for SGPA of more complicated expressions, where a single degree of the whole interaction function would provide insufficient information.

7 COMPLEXITY MEASURES AND CONSTRAINTS

This section compiles a 3-step process of describing the difficulty of performing SGPA of any closed-form equation. Then, we show how we can use the result of this process to design novel complexity measures and constraints aligned with SGPA.

Characterization in 3 steps. To characterize the complexity of performing SGPA of $f(x_1, \ldots, x_N)$ we need to

1. identify all independent variables $x_1, \ldots, x_N$ and for each x_n
2. choose the most *natural* perturbation operator from: $\frac{+}{+x_n}, \frac{+}{\times x_n}, \frac{\times}{+x_n}, \frac{\times}{\times x_n}$, and then
3. express the interaction function in the desired form.

Interaction functions up to the first degree can usually be represented in their standard form. However, it may be helpful to highlight the underlying structure of interaction functions of higher degrees by, for instance, writing it as an interaction composition of lower-degree functions.

7.1 Complexity Measures

Answering a call for better complexity measures for SR (Virgolin et al., 2022), we propose two measures based on our mathematical framework explicitly designed to capture how amenable an equation is to SGPA.

Given the result of the 3-step procedure outlined in the previous section, we can assign a complexity measure to a given closed-form expression. A simple measure, denoted $c_1(f) \in \mathbb{R}^N$, is a vector of degrees of interaction functions (here, we choose the interaction function with the smallest degree).

$$c_1(f)_n = \deg\left(\frac{\circledast_n f}{\circledast'_n x_n}\right) \tag{19}$$

where $n \in [N]$ and $\frac{\circledast_n}{\circledast'_n x_n}$ is the n^{th} perturbation operator as chosen in Step 2. This complexity measure is useful if we are predominantly interested in interaction functions up to the first degree. However, it disregards the diversity of interaction functions of higher degrees. Some of them can be represented using lower-degree interaction functions through interaction composition. Let us fix n and consider an interaction function with respect to x_n of the form $\frac{\circledast_1 f}{\circledast'_1 u_1} \otimes \ldots \otimes \frac{\circledast_M u_{(M-1)}}{\circledast'_M x_n}$, where $\{u_i\}_{i=1}^{M-1}$ are intermediate variables. The following measure is a vector of size N where each entry is a list of degrees of the constituent interaction functions.

$$c_2(f)_n = \left(\deg\left(\frac{\circledast_1 f}{\circledast'_1 u_1}\right), \ldots, \deg\left(\frac{\circledast_M u_{(M-1)}}{\circledast'_M x_n}\right)\right) \tag{20}$$

This measure depends on the particular decomposition of the interaction function. To make it well-defined, we need to specify how the decomposition is performed (see Appendix D for details).

Comparison We compare our complexity measures to the currently used ones in Table 1 and further discuss their limitations in Appendix C.2. To better demonstrate the added value of our framework, in Appendix E.2, we study 9 different equations and quantify their complexity using three metrics: the number of nodes (size-based metric), the metric proposed by Kommenda et al. (2015) (semantic metric) and c_1.

7.2 Novel Constraints

Physics is often celebrated for many of its profound yet simple equations that readily allow for human inspection and various interpretations. As we would expect these equations to have low complexities, we perform a simple sanity check of the proposed complexity constraints by counting how many physics equations satisfy them. We use physics equations collected in the Feynman Symbolic Regression Database (FSReD) (Udrescu and Tegmark, 2020) for this purpose. This set contains one hundred equations from Feynman Lectures. We show all equations in Tables 9 to 11 in Appendix E.5 together with their complexity c_1 and whether they satisfy any of the constraints proposed below. We think

Table 1: Comparison of different complexity measures employed in SR. Semantic: depends on the order and type of operations; interaction-aware: changes depending on which variables are combined; variable-specific: separate measures for each variable; SGPA-aligned: designed for SGPA. References: [1] (Cranmer et al., 2020), [2] (Stephens, 2022), [3] (Udrescu et al., 2021), [4] (Sahoo et al., 2018), [5] (Vladislavleva et al., 2009), [6] (Vanneschi et al., 2010), [7] (Kommenda et al., 2015)

Measure	Semantic	Interaction-aware	Variable-specific	SGPA-aligned
Expression tree depth [1]	✗	✗	✗	✗
Number of terms [2]	✗	✗	✗	✗
Description length [3]	✗	✗	✗	✗
Weight sparsity [4]	✗	✗	✗	✗
Order of linearity [5]	✓	✗	✗	✗
Functional complexity [6]	✓	✓	✗	✗
By Kommenda et al. [7]	✓	✗	✗	✗
c_1 and c_2	✓	✓	✓	✓

it is instructive for the reader to see, under different constraints, what kind of equations would be included in the search.

Only 0th-Degree Interaction Functions The most restrictive constraint we can put on closed-form expressions is to allow only the expressions with a 0th degree interaction function with respect to each variable. As we have proved in Theorem 2, this class of functions is very limited. Yet, 38 out of 100 equations in FSReD have this form.

Only up to 1st-Degree Interaction Functions Allowing for 1st-degree interaction functions adds new classes of equations. That includes univariate functions such as $f = e^{-\theta^2/2}/\sqrt{2\pi}$ and expressions where a variable (such as an angle) is first transformed by a univariate function (such as sin). Examples include $\tau = rF\sin\theta$ and $L = mrv\sin\theta$. That also includes equations where the univariate function is a rational function (e.g., $E = \frac{1}{\gamma-1}p_F V$). This increases our coverage to 59%.

Composed 1st-Degree Interaction Functions Interaction composition allows the expression of high-degree interaction functions as several lower-degree functions composed with one another. However, the more compositions we have, the more difficult it is to inspect the function. We constrain ourselves to *one* interaction composition, where each constituent interaction function has at most degree 1. This gives us additional equations where, for instance, the univariate function is applied to a product of variables instead (e.g., $\theta = \arcsin(\frac{\lambda}{nd})$ or $n = n_0 e^{-\frac{mgx}{k_b T}}$). This also allows for expression where addition and multiplication are combined, such as our previously discussed $U = Gm_1 m_2(\frac{1}{r_2} - \frac{1}{r_1})$. Overall, this constraint covers 83% of equations.

Table 2: Summary of our proposed constraints. Coverage is measured with respect to FSReD (Udrescu and Tegmark, 2020).

Constraint	Coverage	Examples
$\forall n\ c_1(f)_n = 0$	38%	$U = mgz$ $F = \frac{q_1 q_2}{4\pi\epsilon r^2}$
$\forall n\ c_1(f)_n \leq 1$	59%	$f = e^{-\theta^2/2}/\sqrt{2\pi}$ $L = mrv\sin\theta$
$\forall n\ \lvert c_2(f)_n\rvert \leq 2$ $\forall m\ c_2(f)_{n,m} \leq 1$	83%	$\theta = \arcsin(\frac{\lambda}{nd})$ $n = n_0 e^{-\frac{mgx}{k_b T}}$ $U = Gm_1 m_2\left(\frac{1}{r_2} - \frac{1}{r_1}\right)$

7.3 Implementation

We implement the described complexity metrics and constraints as a lightweight python module[2] compatible with `sympy` (Meurer et al., 2017). These can be used out of the box and contain optional parameters for further personalization (e.g., a user can specify an alternative strategy for choosing the perturbation operator (Step 2) or can decide to reduce vector valued c_1 and c_2 to a scalar). The implementation details and examples of its use are available in Appendix D.

Case Study: gplearn We incorporate our metrics in a popular symbolic regression algorithm, `gplearn` (Stephens, 2022), and run a case study on the Concrete dataset (Yeh, 1998) (UCI dataset). We compare our second constraint ($\max(c_1(f)) \leq 1$) with the traditional constraints based on the length of the program. The results can be seen in Appendix E. We can see that not only does the equation found with our constraint have a better performance, but, more importantly, SGPA is more straightforward to perform.

SimplEq: Novel Symbolic Regression Algorithm for the Simplest Equations We demonstrate how our framework can inspire novel symbolic regression techniques. We develop a simple algorithm, SimplEq, for discovering equations satisfying our strongest constraint (all interaction functions have degree 0). Thanks to Theorem 2, we know that any such function of n variables can be characterized by $n+1$ operators (each either $+$ or $\times$) and $n+1$ constants. Thus, for low dimensional settings, as in Feynman datasets, it is possible to check all 2^{n+1} possible combinations of operators and, for each of them, find the best fitting constants using standard algorithms such as L-BFGS (Liu and Nocedal, 1989).

We tested this simple algorithm on FSReD. It managed to achieve R^2 score greater than 0.99 on 34 equations and R^2 score greater than 0.9 on 61 equations. It can find some equations in less than a second and can provide a well-fitting (and very simple) equation even for ground-truth functions that do not satisfy this constraint. This is important, as in real-life scenarios, we may have datasets whose governing equation is beyond our search space. This demonstrates how we can use our framework to define a set of equations of interest (e.g., all interaction functions have degree 0) and then use the theory (e.g., Theorem 2) to design a novel optimization algorithm. Further discussion of the results can be found in Appendix E.

8 RECONSTRUCTION THEOREM

In this section, we highlight an important property of interaction functions:

> Knowing all interaction functions is akin to knowing the original function up to a constant.

The following theorem allows the reconstruction of the original function from its interaction functions.

Theorem 4 (Reconstruction theorem). *Let $f : \mathbb{R}^N \to \mathbb{R}$ and let $\frac{\ast_1 f}{\ast_1' x_i}, \ldots, \frac{\ast_N f}{\ast_N' x_N}$ be the interaction functions. Denote $e_\ast$ as the identity element for $\ast$, i.e., $e_+ = 0$ and $e_\times = 1$. Then*

$$f(\boldsymbol{x}) = \left(\ast_1 \left[\frac{\ast_1 f}{\ast_1' x_1}[x_1](e_{\ast_1'}, x_2, \ldots, x_N) \right] \circ \ldots \circ \ast_n \left[\frac{\ast_N f}{\ast_N' x_N}[x_N](e_{\ast_1'}, \ldots, e_{\ast_N'}) \right] \right) (f(e_{\ast_1'}, \ldots, e_{\ast_N'})) \tag{21}$$

Proof. See Appendix A.3. □

We show how this theorem can be applied to recover $U = Gm_1 m_2\left(\frac{1}{r_2} - \frac{1}{r_1}\right)$ in Appendix E.1.

[2] Code can be accessed at `https://github.com/krzysztof-kacprzyk/SGPA-in-SR`

Implications. This theorem suggests a novel type of search algorithm for SR, proposing to first search for interaction functions that meet our constraints and then reconstruct the equation. Although promising, this approach is still challenging in practice as a function f with specific interaction functions may not always exist. Further research into the compatibility of different interaction functions is needed to realize the potential of this optimization approach.

9 DISCUSSION AND CONCLUSION

Interaction Functions as New Objects of Study A crucial takeaway from our work is that interaction functions unlock a novel perspective on closed-form expressions. They allow for the development of novel complexity measures and constraints (as shown in Section 7) that may prevent SR algorithms from producing expressions that are not useful for a specific downstream task. They also allow us to justify the appeal of many prevalent models (Section 2), to delineate the limitations of "simple" models (Section 4), and to foster the development of new equation classes based on the difficulty of a specific downstream task (e.g., Theorem 2).

Beyond Closed-Form Expressions While our framework primarily addresses closed-form expressions, its applicability extends to generalized additive models (GAMs) (Hastie and Tibshirani, 1986; Lou et al., 2012), widely used in high-stakes scenarios (Caruana et al., 2015). Our framework rationalizes their widespread use as all GAMs have first-degree interaction functions across all variables. Moreover, it can also be applied to Shape Arithmetic Expressions (SHAREs) (Kacprzyk and van der Schaar, 2024) that unify closed-form expressions with generalized additive models. We also suspect a deeper connection between interaction decomposition and the proposed transparency rules for building *transparent SHAREs*.

Related Works Our work proposes a way to measure an equation's complexity. As such, we compare it to various complexity metrics used in the SR literature. We describe them in Section 1 and highlight conceptual differences in Table 1. Moreover, we provide an in-depth comparison in Appendix C.2 and in our case studies in Appendix E.2 (Table 6). Our analysis is largely independent of specific SR algorithms. We provide a brief overview of currently available methods in Appendix C.1 and refer the reader for more details to a recent survey by La Cava et al. (2021). We discuss the relationship with XAI literature in Appendix B.4.

Limitations Although SGPA is built on top of established practices in areas such as physics or risk scoring, a more comprehensive analysis may require an investigation of the underlying data distribution. Certain data characteristics, like high feature correlation, can limit the informativeness of single-feature perturbations. The main limitation of c_1 is that it loses specificity for higher-degree interaction functions. c_2 is designed to address this by considering a decomposition into lower-degree interaction functions. Although a constituent interaction function can have a low degree with respect to a new intermediate variable, the variable itself may be a complex function of the original features. We discuss these limitations further in Appendix B.1.

Acknowledgments

This work was supported by Roche. We would like to thank Kasia Kobalczyk, Julianna Piskorz, Yuchao Qin, Tennison Liu, Zhaozhi Qian, Andrew Rashbass, and anonymous reviewers for their useful comments and feedback on earlier versions of this work.

References

Alaa, A. M. and van der Schaar, M. (2019). Demystifying Black-box Models with Symbolic Metamodels. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Biggio, L., Bendinelli*, T., Neitz, A., Lucchi, A., and Parascandolo, G. (2021). Neural Symbolic Regression that Scales. In *38th International Conference on Machine Learning.*

Bongard, J. and Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948.

Bradburn, M. J., Clark, T. G., Love, S. B., and Altman, D. G. (2003). Survival analysis part II: Multivariate data analysis–an introduction to concepts and methods. *British journal of cancer*, 89(3):431–436.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. (2015). Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730, Sydney NSW Australia. ACM.

Chen, Y., Angulo, M. T., and Liu, Y.-Y. (2019). Revealing Complex Ecological Dynamics via Symbolic Regression. *BioEssays*, 41(12):1900069.

Cox, D. R. (1972). Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

Cox, D. R. (1975). Partial likelihood. *Biometrika*, 62(2):269–276.

Cranmer, M. (2020). PySR: Fast & parallelized symbolic regression in Python/Julia. Zenodo.

Cranmer, M., Sanchez Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. (2020). Discovering Symbolic Models from Deep Learning with Inductive Biases. In *Advances in Neural Information Processing Systems*, volume 33, pages 17429–17442. Curran Associates, Inc.

D'Ascoli, S., Kamienny, P.-A., Lample, G., and Charton, F. (2022). Deep symbolic regression for recurrence prediction. In *Proceedings of the 39th International Conference on Machine Learning*, pages 4520–4536. PMLR.

de Franca, F. O., Virgolin, M., Kommenda, M., Majumder, M. S., Cranmer, M., Espada, G., Ingelse, L., Fonseca, A., Landajuela, M., Petersen, B., Glatt, R., Mundhenk, N., Lee, C. S., Hochhalter, J. D., Randall, D. L., Kamienny, P., Zhang, H., Dick, G., Simon, A., Burlacu, B., Kasak, J., Machado, M., Wilstrup, C., and La Cava, W. G. (2023). Interpretable Symbolic Regression for Data Science: Analysis of the 2022 Competition.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, 24(1):44–65.

Guidetti, V., Dolci, G., Franceschini, E., Bacca, E., Burastero G. J., Ferrari, D., Serra, V., Di Benedetto, F., Mussini, C., and Mandreoli, F. (2023). Death After Liver Transplantation: Mining Interpretable Risk Factors for Survival Prediction. In *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10.

Hastie, T. and Tibshirani, R. (1986). Generalized additive models. *Statistical Science*, 1(3):297–318.

Holt, S., Qian, Z., and van der Schaar, M. (2023). Deep Generative Symbolic Regression. *The Eleventh International Conference on Learning Representations.*

Kacprzyk, K., Qian, Z., and van der Schaar, M. (2023). D-CIPHER: Discovery of Closed-form Partial Differential Equations. In *Advances in Neural Information Processing Systems*, volume 36, pages 27609–27644.

Kacprzyk, K. and van der Schaar, M. (2024). Shape Arithmetic Expressions: Advancing Scientific Discovery Beyond Closed-form Equations. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR.

Kaheman, K., Kutz, J. N., and Brunton, S. L. (2020). SINDy-PI: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2242):20200279.

Kamienny, P.-A., d'Ascoli, S., Lample, G., and Charton, F. (2022). End-to-end Symbolic Regression with Transformers. In *Advances in Neural Information Processing Systems.*

Kommenda, M., Beham, A., Affenzeller, M., and Kronberger, G. (2015). Complexity Measures for Multi-objective Symbolic Regression. In Moreno-Díaz, R., Pichler, F., and Quesada-Arencibia, A., editors, *Computer Aided Systems Theory – EUROCAST 2015*, Lecture Notes in Computer Science, pages 409–416, Cham. Springer International Publishing.

Koza, JohnR. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2).

La Cava, W., Orzechowski, P., Burlacu, B., de França, F. O., Virgolin, M., Jin, Y., Kommenda, M., and Moore, J. H. (2021). Contemporary Symbolic Regression Methods and their Relative Performance. In *35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.*

Lengerich, B. J., Caruana, R., Nunnally, M. E., and Kellis, M. (2022). Death by Round Numbers: Glass-Box Machine Learning Uncovers Biases in Medical Practice.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Long, Z., Lu, Y., and Dong, B. (2019). PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925.

Lou, Y., Caruana, R., and Gehrke, J. (2012). Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 150–158, New York, NY, USA. Association for Computing Machinery.

Lundberg, S. M. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Martius, G. S. and Lampert, C. (2017). Extrapolation and learning equations. In *5th International Conference on Learning Representations, ICLR 2017-Workshop Track Proceedings.*

Messenger, D. A. and Bortz, D. M. (2021a). Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443:110525.

Messenger, D. A. and Bortz, D. M. (2021b). Weak SINDy: Galerkin-Based Data-Driven Model Selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, Am., Ivanov, S., Moore, J. K., Singh, S., et al. (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, 3:e103.

Mothilal, R. K., Sharma, A., and Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617.

Newell, A., Shaw, J. C., and Simon, H. A. (1958). Elements of a theory of human problem solving. *Psychological Review*, 65(3):151–166.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). InterpretML: A Unified Framework for Machine Learning Interpretability.

Petersen, B. K., Larma, M. L., Mundhenk, T. N., Santiago, C. P., Kim, S. K., and Kim, J. T. (2021). Deep Symbolic Regression: Recovering Mathematical Expressions From Data via Risk-seeking Policy Gradients. In *ICLR 2021*.

Qian, Z., Kacprzyk, K., and van der Schaar, M. (2022). D-CODE: Discovering Closed-form ODEs from Observed Trajectories. *The Tenth International Conference on Learning Representations.*

Raissi, M. and Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA. Association for Computing Machinery.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., and Zhong, C. (2022). Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16(none):1–85.

Rudy, S. H., Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2017). Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614.

Sahoo, S., Lampert, C., and Martius, G. (2018). Learning Equations for Extrapolation and Control. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4442–4450. PMLR.

Schmidt, M. and Lipson, H. (2009). Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923):81–85.

Selvin, S. (2008). *Survival Analysis for Epidemiologic and Medical Research.* Cambridge University Press.

Stephens, T. (2022). Gplearn: Genetic programming in python, with a scikit-learn inspired and compatible api.

Udrescu, S.-M., Tan, A., Feng, J., Neto, O., Wu, T., and Tegmark, M. (2021). AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *34th Conference on Neural Information Processing Systems (NeurIPS 2020).*

Udrescu, S.-M. and Tegmark, M. (2020). AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631.

Vaddireddy, H. and San, O. (2019). Equation Discovery Using Fast Function Extraction: A Deterministic Symbolic Regression Approach. *Fluids*, 4(2):111.

Van Ness, M., Bosschieter, T., Din, N., Ambrosy, A., Sandhu, A., and Udell, M. (2023). Interpretable Survival Analysis for Heart Failure Risk Prediction.

Vanneschi, L., Castelli, M., and Silva, S. (2010). Measuring bloat, overfitting and functional complexity in genetic programming. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 877–884, Portland Oregon USA. ACM.

Virgolin, M., De Lorenzo, A., Medvet, E., and Randone, F. (2020). Learning a Formula of Interpretability to Learn Interpretable Formulas. In Bäck, T., Preuss, M., Deutz, A., Wang, H., Doerr, C., Emmerich, M., and Trautmann, H., editors, *Parallel Problem Solving from Nature – PPSN XVI*, Lecture Notes in Computer Science, pages 79–93, Cham. Springer International Publishing.

Virgolin, M., De Lorenzo, A., Randone, F., Medvet, E., and Wahde, M. (2021). Model learning with personalized interpretability estimation (ML-PIE). In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, pages 1355–1364, New York, NY, USA. Association for Computing Machinery.

Virgolin, M., Medvet, E., Alderliesten, T., and Bosman, P. A. N. (2022). Less is More: A Call to Focus on Simpler Models in Genetic Programming for Interpretable Machine Learning.

Vladislavleva, E. J., Smits, G. F., and den Hertog, D. (2009). Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 13(2):333–349.

Wang, Y., Wagner, N., and Rondinelli, J. M. (2019). Symbolic regression in materials science. *MRS Communications*, 9(3):793–805.

Yeh, I.-C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, Appendix B.6]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes, `https://github.com/krzysztof-kacprzyk/SGPA-in-SR`]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes, Sections 3 to 5 and 8]
   (b) Complete proofs of all theoretical results. [Yes, Appendix A]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes, Appendix D]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes, `https://github.com/krzysztof-kacprzyk/SGPA-in-SR`]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Table of supplementary materials

A THEORETICAL RESULTS

A.1 Classification Theorems

In this section, we prove Theorem 1 and Theorem 3.

We start by repeating the statement of Theorem 1.

Theorem (Classification Theorem for 0th-Degree Interaction Functions)**.** *Let $f : \mathbb{R}^N \to \mathbb{R}$.*[3] *Then, the following are the only 0^{th}-degree interaction functions for each of the four perturbation operators.*

- $\frac{+f}{+x_n}[\lambda] = a\lambda$*, where $a \in \mathbb{R}$ is any constant*
- $\frac{+f}{\times x_n}[\lambda] = a\log(\lambda)$*, where $a \in \mathbb{R}$ is any constant*
- $\frac{\times f}{+x_n}[\lambda] = a^{\lambda}$*, where $a \in \mathbb{R}_{\geq 0}$ is any constant*
- $\frac{\times f}{\times x_n}[\lambda] = \lambda^a$*, where $a \in \mathbb{R}$ is any constant*

We use $\frac{\ast_1}{\ast_2}I_a$ to denote these functions. Then we can write $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = \frac{\ast_1}{\ast_2}I_a(\lambda)$, where $a \in \mathbb{R}$ is the free parameter.

Proof. Let $\ast_1, \ast_2 \in \{+, \times\}^2$. $\deg(\frac{\ast_1 f}{\ast_2 x_n}) = 0$ means that there exists a function $h : \mathbb{R} \to \mathbb{R}$ such that $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = h(\lambda)$. By definition of the interaction function, this gives us:

$$\ast_1^{-1}(f(p_n^{\ast_2}(\boldsymbol{x}, \lambda), f(\boldsymbol{x}))) = h(\lambda) \tag{22}$$

This implies:

$$f(p_n^{\ast_2}(\boldsymbol{x}, \lambda)) = h(\lambda) \ast_1 f(\boldsymbol{x}) \tag{23}$$

We use $e_{\ast}$ to denote the identity element for $\ast$, i.e., $e_+ = 0$ and $e_\times = 1$. We also denote $\boldsymbol{x}_{-n}$ to denote the vector $\boldsymbol{x}$ with x_n removed. We reorder the arguments of f such that we can write $f(\boldsymbol{x})$ as $f(\boldsymbol{x}_{-n}, x_n)$. Then, we can rewrite Equation (23) as the following.

$$f(\boldsymbol{x}_{-n}, x_n \ast_2 \lambda) = h(\lambda) \ast_1 f(\boldsymbol{x}_{-n}, x_n) \tag{24}$$

Now let us consider what happens when we set x_n equal to $e_{\ast_2}$. $x_n \ast_2 \lambda$ becomes just λ. Overall, it implies:

$$f(\boldsymbol{x}_{-n}, \lambda) = h(\lambda) \ast_1 f(\boldsymbol{x}_{-n}, e_{\ast_2}) \tag{25}$$

Let us introduce $g : \mathbb{R}^{N-1} \to \mathbb{R}$ defined as

$$g(\boldsymbol{x}_{-n}) = f(\boldsymbol{x}_{-n}, e_{\ast_2}) \tag{26}$$

This implies (by renaming λ to x_n) that

$$f(\boldsymbol{x}_{-n}, x_n) = h(x_n) \ast_1 g(\boldsymbol{x}_{-n}). \tag{27}$$

This makes sense as $g(\boldsymbol{x}_{-n})$ part would disappear after applying $\frac{\ast_1}{\ast_2 x_n}$ to f. Substituting it back Equation (24) gives us the following.

$$h(x_n \ast_2 \lambda) \ast_1 g(\boldsymbol{x}_{-n}) = h(\lambda) \ast_1 h(x_n) \ast_1 g(\boldsymbol{x}_{-n}) \tag{28}$$

[3]All functions are assumed to be continuous almost everywhere

If $\ast_1 = +$ then we can safely subtract $g(\boldsymbol{x}_{-n})$ from both sides. If $\ast_1 = \times$ then we can divide both sides by $g(\boldsymbol{x}_{-n})$ and the resulting equality holds for all $\boldsymbol{x}$ such that $g(\boldsymbol{x}_{-n}) \neq 0$ (and consequently $f(\boldsymbol{x}) \neq 0$).

$$h(x_n \ast_2 \lambda) = h(\lambda) \ast_1 h(x_n) \tag{29}$$

This is a class of functional equations called Cauchy's Functional Equations. For different pairs of $\{\ast_1, \ast_2\}$, we get the following types:

- $(+, +)$: Cauchy's additive functional equation $h(x + y) = h(x) + h(y)$
- $(\times, \times)$: Cauchy's multiplicative functional equation $h(xy) = h(x)h(y)$
- $(\times, +)$: Cauchy's exponential functional equation $h(x + y) = h(x)h(y)$
- $(+, \times)$: Cauchy's logarithmic functional equation $h(xy) = h(x) + h(y)$

It is well-known that under mild regularity conditions, such as continuity at one point, there is only one family of functions for each of these equations. In particular,

- For $(+, +)$: $h(\lambda) = a\lambda$, where $a \in \mathbb{R}$ is any constant
- For $(\times, \times)$: $h(\lambda) = \lambda^a$, where $a \in \mathbb{R}$ is any constant
- For $(\times, +)$: $h(\lambda) = a^\lambda$, where $a \in \mathbb{R}_{\geq 0}$ is any constant
- For $(+, \times)$: $h(\lambda) = a \log(\lambda)$, where $a \in \mathbb{R}$ is any constant

We can now set

$$f(\boldsymbol{x}_{-n}, x_n) = h(x_n) \ast_1 g(\boldsymbol{x}_{-n}) \tag{30}$$

with an arbitrary $g : \mathbb{R}^{N-1} \to \mathbb{R}$ and directly verify that for each h Equation (22) holds. $\square$

Now, we prove Theorem 3. We start by reiterating the statement of the theorem.

Theorem. *Let $f : \mathbb{R}^N \to \mathbb{R}$ and let $G : \mathbb{R}^2 \to \mathbb{R}$. If $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = G(x_n, \lambda)$ then $G(x_n, \lambda) = \frac{\ast_1 g}{\ast_2}[\lambda](x_n)$ for some $g : \mathbb{R} \to \mathbb{R}$. If $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = G(x_m, \lambda)$, where $m \neq n$ then $G(x_m, \lambda) = \frac{\ast_1}{\ast_2} I_{g(x_m)}(\lambda)$.*

Proof. We start by proving the **first case**. Let us assume that there exists $G : \mathbb{R}^2 \to \mathbb{R}$ such that $\frac{\ast_1 f}{\ast_2 x_n}[\lambda] = G(x_n, \lambda)$. Let us rewrite using Definition 2 and the notation introduced in the previous proof.

$$f(\boldsymbol{x}_{-n}, x_n \ast_2 \lambda) = G(x_n, \lambda) \ast_1 f(\boldsymbol{x}_{-n}, x_n) \tag{31}$$

Consider what happens when $x_n = e_{\ast_2}$. We get the following.

$$f(\boldsymbol{x}_{-n}, \lambda) = G(e_{\ast_2}, \lambda) \ast_1 f(\boldsymbol{x}_{-n}, e_{\ast_2}) \tag{32}$$

Similarly, as in previous proof we let $h : \mathbb{R}^{N-1} \to \mathbb{R}$ to be defined by $h(\boldsymbol{x}_{-n}) = f(\boldsymbol{x}_{-n}, e_{\ast_2})$. That gives us

$$f(\boldsymbol{x}_{-n}, x_n) = G(e_{\ast_2}, x_n) \ast_1 h(\boldsymbol{x}_{-n}) \tag{33}$$

Let us now define $g : \mathbb{R} \to \mathbb{R}$ by

$$g(t) = G(e_{\ast_2}, t). \tag{34}$$

Then we can substitute everything back into Equation (31) to obtain:

$$g(x_n \ast_2 \lambda) \ast_1 h(\boldsymbol{x}_{-n}) = G(x_n, \lambda) \ast_1 g(x_n) \ast_1 h(\boldsymbol{x}_{-n}) \tag{35}$$

Similarly as in the previous proof, for all $\boldsymbol{x}$ such that $h(\boldsymbol{x}_{-n}) \neq 0$ (and consequently $f(\boldsymbol{x}) \neq 0$), we get the following.

$$g(x_n \ast_2 \lambda) = G(x_n, \lambda) \ast_1 g(x_n) \tag{36}$$

By rearranging, we get (for all x_n such that $g(x_n) \neq 0$)

$$G(x_n, \lambda) = \circledast_1^{-1}(g(x_n \circledast_2 \lambda), g(x_n)) \tag{37}$$

which can be rewritten as:

$$G(x_n, \lambda) = \frac{\circledast_1 g}{\circledast_2}[\lambda](x_n) \tag{38}$$

Now, we proceed to the **second case**. Let us assume that there exists $G : \mathbb{R}^2 \to \mathbb{R}$ such that $\frac{\circledast_1 f}{\circledast_2 x_n}[\lambda] = G(x_m, \lambda)$ where $m \neq n$. Let us rewrite using Definition 2 and the notation introduced in the previous proof.

$$f(\boldsymbol{x}_{-n}, x_n \circledast_2 \lambda) = G(x_m, \lambda) \circledast_1 f(\boldsymbol{x}_{-n}, x_n) \tag{39}$$

We set $x_n = e_{\circledast_2}$ and then set λ to x_n. This gives us:

$$f(\boldsymbol{x}_{-n}, x_n) = G(x_m, x_n) \circledast_1 f(\boldsymbol{x}_{-n}, e_{\circledast_2}) \tag{40}$$

Similarly, as in previous proof we let $h : \mathbb{R}^{N-1} \to \mathbb{R}$ to be defined by $h(\boldsymbol{x}_{-n}) = f(\boldsymbol{x}_{-n}, e_{\circledast_2})$. That gives us

$$f(\boldsymbol{x}_{-n}, x_n) = G(x_m, x_n) \circledast_1 h(\boldsymbol{x}_{-n}) \tag{41}$$

We substitute it back to Equation (39) to get:

$$G(x_m, x_n \circledast_2 \lambda) \circledast_1 h(\boldsymbol{x}_{-n}) = G(x_m, \lambda) \circledast_1 G(x_m, x_n) \circledast_1 h(\boldsymbol{x}_{-n}) \tag{42}$$

Similarly as in the previous proof, for all $\boldsymbol{x}$ such that $h(\boldsymbol{x}_{-n}) \neq 0$ (and consequently $f(\boldsymbol{x}) \neq 0$), we get the following.

$$G(x_m, x_n \circledast_2 \lambda) = G(x_m, \lambda) \circledast_1 G(x_m, x_n) \tag{43}$$

We can rewrite it defining $g_{x_m} : \mathbb{R} \to \mathbb{R}$ as $g_{x_m}(t) = G(x_m, t)$.

$$g_{x_m}(x_n \circledast_2 \lambda) = g_{x_m}(\lambda) \circledast_1 g_{x_m}(x_n) \tag{44}$$

We can see that g_{x_m} satisfies one of the Cauchy's Functional Equations. Thus

$$g_{x_m}(t) = \frac{\circledast_1}{\circledast_2} I_{g'(x_m)}(t), \tag{45}$$

where $g' : \mathbb{R} \to \mathbb{R}$. Finally:

$$G(x_m, \lambda) = \frac{\circledast_1}{\circledast_2} I_{g'(x_m)}(\lambda) \tag{46}$$

□

A.2 Characerization of Functions That Have 0th-Degree Interaction Function With Respect to Each Variable

Below, we reiterate the statement of Theorem 2.

Theorem. *Let* $\circledast_1, \circledast_1', \ldots, \circledast_N, \circledast_N' \in \{+, \times\}$ *and* $f : \mathbb{R}^N \to \mathbb{R}$. *If* $\forall n \in [N]$ $\deg\left(\frac{\circledast_n f}{\circledast_i' x_n}\right) = 0$ *then*

- $\circledast_1 = \ldots = \circledast_N = \circledast$, *where* $\circledast \in \{+, \times\}$,
- $f(\boldsymbol{x}) = \begin{cases} a_0 + \sum_{n=1}^N h_n(x_n) & \text{if } \circledast = + \\ a_0 \prod_{n=1}^N h_n(x_n) & \text{if } \circledast = \times \end{cases}$,
- $h_n(x_n) = \begin{cases} a_n x_n & \text{if } \circledast = + \wedge \circledast_n' = + \\ a_n \log(x_n) & \text{if } \circledast = + \wedge \circledast_n' = \times \\ x_n^{a_n} & \text{if } \circledast = \times \wedge \circledast_n' = \times \\ e^{a_n x_n} & \text{if } \circledast = \times \wedge \circledast_n' = + \end{cases}$,

- $\forall n \in \mathbb{N}_0 \ a_n \in \mathbb{R}$.

Proof. From proof of Theorem 1 we get that if $\frac{\circledast_1 f}{\circledast'_1 x_1}[\lambda] = g(\lambda)$ for some $g : \mathbb{R} \to \mathbb{R}$ then

$$f(\boldsymbol{x}_{-1}, x_1) = g(x_1) \circledast_1 h(\boldsymbol{x}_{-1}), \tag{47}$$

where $h : \mathbb{R}^{N-1} \to \mathbb{R}$ is some function. Now from $\frac{\circledast_2 f}{\circledast'_2 x_2}[\lambda] = g'(\lambda)$ for some $g' : \mathbb{R} \to \mathbb{R}$. Then it means that

$$f(\boldsymbol{x}_{-1,-2}, x_1, x_2 \circledast'_2 \lambda) = f(\boldsymbol{x}_{-1-2}, x_1, x_2) \circledast_2 g'(\lambda) \tag{48}$$

Substituting Equation (47), we obtain:

$$g(x_1) \circledast_1 h(\boldsymbol{x}_{-1,-2}, x_2 \circledast'_2 \lambda) = (g(x_1) \circledast_1 h(\boldsymbol{x}_{-1})) \circledast_2 g'(\lambda) \tag{49}$$

From properties of g, if we set $x_1 = e_{\circledast'_1}$ then $g(e_{\circledast'_1}) = e_{\circledast_1}$. Thus we can infer:

$$h(\boldsymbol{x}_{-1,-2}, x_2 \circledast'_2 \lambda) = h(\boldsymbol{x}_{-1}) \circledast_2 g'(\lambda) \tag{50}$$

Let us set x_2 to $e_{\circledast'_2}$ and then set $\lambda = x_2$.

$$h(\boldsymbol{x}_{-1,-2}, x_2) = h(\boldsymbol{x}_{-1,-2}, e_{\circledast'_2}) \circledast_2 g'(x_2) \tag{51}$$

Let us define $h' : \mathbb{R}^{N-2} \to \mathbb{R}$ as $h'(\boldsymbol{x}_{-1,-2}) = h(\boldsymbol{x}_{-1,-2}, e_{\circledast'_2})$. Then

$$h(\boldsymbol{x}_{-1}) = h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(x_2) \tag{52}$$

Substituting this back in Equation (49), we obtain:

$$g(x_1) \circledast_1 (h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(x_2 \circledast'_2 \lambda)) = (g(x_1) \circledast_1 (h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(x_2))) \circledast_2 g'(\lambda) \tag{53}$$

Once again, let us set x_2 to $e_{\circledast'_2}$ and then set $\lambda = x_2$.

$$g(x_1) \circledast_1 (h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(x_2)) = (g(x_1) \circledast_1 (h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(e_{\circledast'_2}))) \circledast_2 g'(x_2) \tag{54}$$

Then, from the property of g' we get that $g'(e_{\circledast'_2}) = e_{\circledast_2}$. This gives us:

$$g(x_1) \circledast_1 (h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(x_2)) = (g(x_1) \circledast_1 (h'(\boldsymbol{x}_{-1,-2}))) \circledast_2 g'(x_2) \tag{55}$$

As the value of g depends only on x_1, value of g' only on x_2, and value of h' only on $\boldsymbol{x}_{-1,-2}$. This holds if and only if the following is true:

$$(A \circledast_1 (B \circledast_2 C)) = (A \circledast_1 B) \circledast_2 C \tag{56}$$

for any $A, B, C \in \mathbb{R}$. It is straightforward to check that this is true if and only if $\circledast_1 = \circledast_2$.

We can now apply this result to other pairs of operators to infer that

$$\circledast_1 = \ldots = \circledast_N = \circledast \tag{57}$$

This proves the first part of Theorem 2.

We have proved so far that if $\frac{\circledast_1 f}{\circledast'_1 x_1}[\lambda] = g(\lambda)$ for some $g : \mathbb{R} \to \mathbb{R}$ then

$$f(\boldsymbol{x}_{-1}, x_1) = g(x_1) \circledast_1 h(\boldsymbol{x}_{-1}), \tag{58}$$

And then we showed that if $\frac{\circledast_2 f}{\circledast'_2 x_2}[\lambda] = g'(\lambda)$ for some $g' : \mathbb{R} \to \mathbb{R}$ then

$$h(\boldsymbol{x}_{-1}) = h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(x_2). \tag{59}$$

This implies

$$f(\boldsymbol{x}) = g(x_1) \circledast_1 (h'(\boldsymbol{x}_{-1,-2}) \circledast_2 g'(x_2)). \tag{60}$$

But as we have shown that $\circledast_1 = \ldots = \circledast_N = \circledast$, we can simplify it to:

$$f(\boldsymbol{x}) = g(x_1)\circledast g'(x_2)\circledast h'(\boldsymbol{x}_{-1,-2}). \tag{61}$$

Repeating the process but now with $\frac{\circledast_3 f}{\circledast'_3 x_3}[\lambda] = g''(\lambda)$ would yield:

$$h(\boldsymbol{x}_{-1}) = h''(\boldsymbol{x}_{-1,-3})\circledast_3 g''(x_3) \tag{62}$$

which we can expand to the following.

$$h'(\boldsymbol{x}_{-1,-2.-3}, x_3)\circledast_2 g'(x_2) = h''(\boldsymbol{x}_{-1,-3,-2}, x_2)\circledast_3 g''(x_3) \tag{63}$$

We use the fact that $\circledast_2 = \circledast_3 = \circledast$ and we set $x_2 = e_{\circledast'_2}$, so that $g'(e_{\circledast'_2}) = e_{\circledast}$. Then

$$h'(\boldsymbol{x}_{-1,-2}) = h''(\boldsymbol{x}_{-1,-3,-2}, e_{\circledast'_2})\circledast g''(x_3) \tag{64}$$

That means that

$$f(\boldsymbol{x}) = g(x_1)\circledast g'(x_2)\circledast g''(x_3)\circledast h''(\boldsymbol{x}_{-1,-3,-2}, e_{\circledast'_2}) \tag{65}$$

Continuing this process, we get that

$$f(\boldsymbol{x}) = g(x_1)\circledast g'(x_2)\circledast \ldots \circledast g^{(N)}(x_N)\circledast a_0 \tag{66}$$

where $\frac{\circledast_n f}{\circledast'_n x_n}[\lambda] = g^{(N)}(\lambda)$, and $a_0 \in \mathbb{R}$.

If $\circledast = +$ then

$$f(\boldsymbol{x}) = a_0 + \sum_{n=1}^{N} g^{(n)}(x_n) \tag{67}$$

where $g^{(n)}(t) = a_n t$ if $\circledast'_n = +$ and $g^{(n)}(t) = a_n \log(t)$ if $\circledast'_n = \times$.

If $\circledast = \times$ then

$$f(\boldsymbol{x}) = a_0 \prod_{n=1}^{N} g^{(n)}(x_n) \tag{68}$$

where $g^{(n)}(t) = a_n^t$ if $\circledast'_n = +$ and $g^{(n)}(t) = t^{a_n}$ if $\circledast'_n = \times$.

□

A.3 Reconstruction Theorem

We reiterate Theorem 4.

Theorem (Reconstruction theorem)**.** *Let $f : \mathbb{R}^N \to \mathbb{R}$ and let $\frac{\circledast_1 f}{\circledast'_1 x_i}, \ldots, \frac{\circledast_N f}{\circledast'_N x_N}$ be the interaction functions. Denote $e_{\circledast}$ as the identity element for $\circledast$, i.e., $e_+ = 0$ and $e_\times = 1$. Then*

$$f(\boldsymbol{x}) = \left(\circledast_1\left[\frac{\circledast_1 f}{\circledast'_1 x_1}[x_1](e_{\circledast'_1}, x_2, \ldots, x_N)\right] \circ \ldots \circ \circledast_n\left[\frac{\circledast_N f}{\circledast'_N x_N}[x_N](e_{\circledast'_1}, \ldots, e_{\circledast'_N})\right]\right)(f(e_{\circledast'_1}, \ldots, e_{\circledast'_N})) \tag{69}$$

Proof. By definition,

$$\frac{\circledast_1 f}{\circledast'_1 x_1}[\lambda](\boldsymbol{x}) = {\circledast_1}^{-1}(f(p_1^{\circledast'_1}(\boldsymbol{x}, \lambda), f(\boldsymbol{x})) \tag{70}$$

Thus, we can rearrange it to obtain the following:

$$f(p_1^{\circledast'_1}(\boldsymbol{x}, \lambda)) = f(\boldsymbol{x})\circledast_1 \frac{\circledast_1 f}{\circledast'_1 x_1}[\lambda](\boldsymbol{x}) \tag{71}$$

Using notation from previous proofs, we can write it as:

$$f(p_1^{\circledast'_1}(\boldsymbol{x}_{-1}, x_1, \lambda)) = f(\boldsymbol{x}_{-1}, x_1)\circledast_1 \frac{\circledast_1 f}{\circledast'_1 x_1}[\lambda](\boldsymbol{x}_{-1}, x_1) \tag{72}$$

We can now set x_1 to $e_{\circledast'_1}$. Observe that

$$p_1^{\circledast'_1}(\boldsymbol{x}_{-1}, e_{\circledast'_1}, \lambda) = p_1^{\circledast'_1}(\boldsymbol{x}_{-1}, \lambda, e_{\circledast'_1}) = (\boldsymbol{x}_{-1}, \lambda) \tag{73}$$

So when we then set λ to x_1, we get the following.

$$f(\boldsymbol{x}_{-1}, x_1) = f(\boldsymbol{x}_{-1}, e_{\circledast'_1}) \circledast_1 \frac{\circledast_1 f}{\circledast'_1 x_1}[x_1](\boldsymbol{x}_{-1}, e_{\circledast'_1}) \tag{74}$$

It is helpful to rewrite it as follows:

$$f(\boldsymbol{x}) = \circledast_1 \left[\frac{\circledast_1 f}{\circledast'_1 x_1}[x_1](\boldsymbol{x}_{-1}, e_{\circledast'_1})\right](f(\boldsymbol{x}_{-1}, e_{\circledast'_1})) \tag{75}$$

So we managed to write $f(\boldsymbol{x})$ using $\frac{\circledast_1 f}{\circledast'_1 x_1}$ and $f(\boldsymbol{x}_{-1}, e_{\circledast'_1})$. We can repeat the process with the next pair of operators to get:

$$f(\boldsymbol{x}) = \circledast_2 \left[\frac{\circledast_2 f}{\circledast'_2 x_2}[x_2](\boldsymbol{x}_{-2}, e_{\circledast'_2})\right](f(\boldsymbol{x}_{-2}, e_{\circledast'_2})) \tag{76}$$

If we now set x_1 to $e_{\circledast'_1}$ we get:

$$f(\boldsymbol{x}_{-1}, e_{\circledast'_1}) = \circledast_2 \left[\frac{\circledast_2 f}{\circledast'_2 x_2}[x_2](\boldsymbol{x}_{-2,-1}, e_{\circledast'_1}, e_{\circledast'_2})\right](f(\boldsymbol{x}_{-2,-1}, e_{\circledast'_1}, e_{\circledast'_2})) \tag{77}$$

We can now substitute it back to Equation (75) to get:

$$f(\boldsymbol{x}) = \circledast_1 \left[\frac{\circledast_1 f}{\circledast'_1 x_1}[x_1](\boldsymbol{x}_{-1}, e_{\circledast'_1})\right]\left(\circledast_2 \left[\frac{\circledast_2 f}{\circledast'_2 x_2}[x_2](\boldsymbol{x}_{-2,-1}, e_{\circledast'_1}, e_{\circledast'_2})\right](f(\boldsymbol{x}_{-2,-1}, e_{\circledast'_1}, e_{\circledast'_2}))\right) \tag{78}$$

We can rewrite it using function composition.

$$f(\boldsymbol{x}) = \left(\circledast_1 \left[\frac{\circledast_1 f}{\circledast'_1 x_1}[x_1](\boldsymbol{x}_{-1}, e_{\circledast'_1})\right] \circ \circledast_2 \left[\frac{\circledast_2 f}{\circledast'_2 x_2}[x_2](\boldsymbol{x}_{-2,-1}, e_{\circledast'_1}, e_{\circledast'_2})\right]\right)(f(\boldsymbol{x}_{-2,-1}, e_{\circledast'_1}, e_{\circledast'_2})) \tag{79}$$

By applying this approach N times, we get that

$$f(\boldsymbol{x}) = \left(\circledast_1 \left[\frac{\circledast_1 f}{\circledast'_1 x_1}[x_1](\boldsymbol{x}_{-1}, e_{\circledast'_1})\right] \circ \ldots \circ \circledast_N \left[\frac{\circledast_2 f}{\circledast'_2 x_N}[x_N](e_{\circledast'_1}, \ldots, e_{\circledast'_N})\right]\right)(f(e_{\circledast'_1}, \ldots, e_{\circledast'_N})) \tag{80}$$

□

A.4 Perturbation Calculus

This section provides a series of valuable rules for calculating interaction functions.

Proposition 1 (Perturbation Calculus)**.** *Let $f : \mathbb{R}^N \to \mathbb{R}$, and $g : \mathbb{R}^N \to \mathbb{R}$, $h : \mathbb{R} \to \mathbb{R}$. Let $\circledast_1, \circledast_2, \circledast_3 \in \{+, \times\}$.*

1. *If x_n is not active variable in f then $\frac{\circledast_1 f}{\circledast_2 x_n} = e_{\circledast_1}$*
2. $\frac{\circledast_1}{\circledast_2 x_n}(f \circledast_1 g) = \frac{\circledast_1 f}{\circledast_2 x_n} \circledast_1 \frac{\circledast_1 g}{\circledast_2 x_n}$
3. $\frac{\circledast_1}{\circledast_2 x_n}(h \circ f) = \frac{\circledast_1 h}{\circledast_3} \otimes \frac{\circledast_3 f}{\circledast_2 x_n}$ *for any $\circledast_3 \in \{+, \times\}$*

Proof. **Proof of 1.**

$$\frac{\circledast_1 f}{\circledast_2 x_n}[\lambda](\boldsymbol{x}) = \circledast_1^{-1}(f(p_n^{\circledast_2}(\boldsymbol{x}, \lambda)), f(\boldsymbol{x})) \tag{81}$$

As x_n is not an active variable of f then $f(p_n^{\circledast_2}(\boldsymbol{x},\lambda)) = f(\boldsymbol{x})$. Then

$$\frac{\circledast_1 f}{\circledast_2 x_n}[\lambda](\boldsymbol{x}) = \circledast_1^{-1}(f(\boldsymbol{x}), f(\boldsymbol{x})) = e_{\circledast_1} \tag{82}$$

Proof of 2.

$$\begin{aligned}\frac{\circledast_1}{\circledast_2 x_n}(f \circledast_1 g)[\lambda](\boldsymbol{x}) &= \circledast_1^{-1}((f \circledast_1 g)(p_n^{\circledast_2}(\boldsymbol{x},\lambda)), (f \circledast_1 g)(\boldsymbol{x})) \\ &= \circledast_1^{-1}((f(p_n^{\circledast_2}(\boldsymbol{x},\lambda)) \circledast_1 g(p_n^{\circledast_2}(\boldsymbol{x},\lambda)), f(\boldsymbol{x}) \circledast_1 g(\boldsymbol{x})) \\ &= \left[\circledast_1^{-1}(f(p_n^{\circledast_2}(\boldsymbol{x},\lambda)), f(\boldsymbol{x}))\right] \circledast_1 \left[\circledast_1^{-1}(g(p_n^{\circledast_2}(\boldsymbol{x},\lambda)), g(\boldsymbol{x}))\right] \\ &= \frac{\circledast_1 f}{\circledast_2 x_n}[\lambda](\boldsymbol{x}) \circledast_1 \frac{\circledast_1 g}{\circledast_2 x_n}[\lambda](\boldsymbol{x})\end{aligned} \tag{83}$$

Proof of 3.

$$\begin{aligned}\frac{\circledast_1 (h \circ f)}{\circledast_2 x_n}[\lambda](\boldsymbol{x}) &= \circledast_1^{-1}((h \circ f)(p_n^{\circledast_2}(\boldsymbol{x},\lambda)), (h \circ f)(\boldsymbol{x})) \\ &= \circledast_1^{-1}(h(f(p_n^{\circledast_2}(\boldsymbol{x},\lambda))), h(f(\boldsymbol{x})))\end{aligned} \tag{84}$$

Let $\circledast_3 \in \{+, \times\}$. Then we can write for any $a, b \in \mathbb{R}$

$$\circledast_3^{-1}(a,b) \circledast_3 b = a \tag{85}$$

So we can rewrite the previous expression as:

$$\circledast_1^{-1}(h(\circledast_3^{-1}(f(p_n^{\circledast_2}(\boldsymbol{x},\lambda)), f(\boldsymbol{x})) \circledast_3 f(\boldsymbol{x})), h(f(\boldsymbol{x}))) \tag{86}$$

We interpret $\circledast_3^{-1}(f(p_n^{\circledast_2}(\boldsymbol{x},\lambda)), f(\boldsymbol{x}))$ as $\frac{\circledast_3 f}{\circledast_2 x_n}[\lambda](\boldsymbol{x})$. Thus, we can rewrite it as:

$$\circledast_1^{-1}\left(h\left(\frac{\circledast_3 f}{\circledast_2 x_n}[\lambda](\boldsymbol{x}) \circledast_3 f(\boldsymbol{x})\right), h(f(\boldsymbol{x}))\right) \tag{87}$$

This can be once again rewritten as follows.

$$\frac{\circledast_1 h}{\circledast_3}\left[\frac{\circledast_3 f}{\circledast_2 x_n}[\lambda](\boldsymbol{x})\right](f(\boldsymbol{x})) \tag{88}$$

We can now use interaction composition to write it as:

$$\frac{\circledast_1 (h \circ f)}{\circledast_2 x_n}[\lambda](\boldsymbol{x}) = \left(\frac{\circledast_1 h}{\circledast_3} \otimes \frac{\circledast_3 f}{\circledast_2 x_n}\right)[\lambda](\boldsymbol{x}) \tag{89}$$

□

Proposition 2. *Let $f : \mathbb{R}^2 \to \mathbb{R}$ be defined as $f(u_1, u_2) = u_1 \circledast_1 u_2$. Let $u_i = g_i(\boldsymbol{x})$ where each $g_i : \mathbb{R}^N \to \mathbb{R}$ ($N \in \mathbb{N}$). Let $n \in [N]$ and assume that g_2 does not depend on x_n. Then for $y(\boldsymbol{x}) = f \circ (g_1, g_2)(\boldsymbol{x})$ the following holds.*

$$\frac{\circledast_2 y}{\circledast_3 x_n}[\lambda](\boldsymbol{x}) = \left(\frac{\circledast_2 f}{\circledast_1 u_1} \otimes \frac{\circledast_1 u_1}{\circledast_3 x_n}\right)[\lambda](\boldsymbol{x}) \tag{90}$$

Proof.

$$\begin{aligned}
\frac{\circledast_2 y}{\circledast_3 x_n}[\lambda](\boldsymbol{x}) &= \circledast_2^{-1}(y(p_n^{\circledast_3}(\boldsymbol{x},\lambda)), y(\boldsymbol{x})) \\
&= \circledast_2^{-1}(g_1(p_n^{\circledast_3}(\boldsymbol{x},\lambda))\circledast_1 g_2(p_n^{\circledast_3}(\boldsymbol{x},\lambda)), g_1(\boldsymbol{x})\circledast_1 g_2(\boldsymbol{x})) \\
&= \circledast_2^{-1}(g_1(p_n^{\circledast_3}(\boldsymbol{x},\lambda))\circledast_1 \left(g_1(\boldsymbol{x})\circledast_1^{-1} g_1(\boldsymbol{x})\right)\circledast_1 g_2(\boldsymbol{x}), g_1(\boldsymbol{x})\circledast_1 g_2(\boldsymbol{x})) \\
&= \circledast_2^{-1}(g_1(p_n^{\circledast_3}(\boldsymbol{x},\lambda))\circledast_1^{-1} g_1(\boldsymbol{x})\circledast_1 g_1(\boldsymbol{x})\circledast_1 g_2(\boldsymbol{x}), g_1(\boldsymbol{x})\circledast_1 g_2(\boldsymbol{x})) \\
&= \circledast_2^{-1}\left(\frac{\circledast_1 g_1}{\circledast_3 x_n}[\lambda](\boldsymbol{x})\circledast_1 g_1(\boldsymbol{x})\circledast_1 g_2(\boldsymbol{x}), g_1(\boldsymbol{x})\circledast_1 g_2(\boldsymbol{x})\right) \\
&= \circledast_2^{-1}\left(f\left(g_1(\boldsymbol{x}) + \frac{\circledast_1 g_1}{\circledast_3 x_n}[\lambda](\boldsymbol{x}), g_2(\boldsymbol{x})\right), f(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}))\right) \\
&= \frac{\circledast_2 f}{\circledast_1 u_1}\left[\frac{\circledast_1 g_1}{\circledast_3 x_n}[\lambda](\boldsymbol{x})\right](g_1(\boldsymbol{x}), g_2(\boldsymbol{x})) \\
&= \left(\frac{\circledast_2 f}{\circledast_1 u_1} \otimes \frac{\circledast_1 u_1}{\circledast_3 x_n}\right)[\lambda](\boldsymbol{x})
\end{aligned} \tag{91}$$

□

Proposition 3. *Consider a setup as in Definition 5. Let us assume that* $\frac{\circledast_1(f\circ \boldsymbol{g})}{\circledast_2 x_l} = \frac{\circledast_1 f}{\circledast_3 u_k} \otimes \frac{\circledast_3 u_k}{\circledast_2 x_l}$ *for some* $k \in [N]$ *and* $l \in [M]$. *If* $\deg\left(\frac{\circledast_1 f}{\circledast_3 u_k}\right) = 0$ *and* $\deg\left(\frac{\circledast_3 u_k}{\circledast_2 x_l}\right) = 0$ *then* $\deg\left(\frac{\circledast_1(f\circ \boldsymbol{g})}{\circledast_2 x_l}\right) = 0$.

Proof. By Theorem 1,

$$\frac{\circledast_1 f}{\circledast_3 u_k}[\lambda](\boldsymbol{u}) = \frac{\circledast_1}{\circledast_3} I_a(\lambda) \tag{92}$$

for some $a \in \mathbb{R}$. Similarly,

$$\frac{\circledast_3 u_k}{\circledast_2 x_l}[\lambda](\boldsymbol{x}) = \frac{\circledast_3}{\circledast_2} I_b(\lambda) \tag{93}$$

for some $b \in \mathbb{R}$. Thus by Definition 5

$$\frac{\circledast_1(f\circ \boldsymbol{g})}{\circledast_2 x_l}[\lambda](\boldsymbol{x}) = \frac{\circledast_1}{\circledast_3} I_a\left(\frac{\circledast_3}{\circledast_2} I_b(\lambda)\right) \tag{94}$$

As the right-hand side depends only on λ, it follows that

$$\deg\left(\frac{\circledast_1(f\circ \boldsymbol{g})}{\circledast_2 x_l}\right) = 0 \tag{95}$$

□

B ADDITIONAL DISCUSSION

B.1 Limitations of Complexity Measures and Constraints

We want to emphasize that the complexity measures and constraints we introduced serve as illustrative examples of how our framework can be utilized. We hope better measures and constraints based on interaction functions can be developed in the future. Their utility is also dependent on a particular problem setting.

The main limitation of c_1 is that it loses specificity for higher-degree interaction functions. We could claim that every 0^{th}-degree interaction function is equally straightforward to analyze. There are only four distinctive possibilities (Theorem 1), all of which require understanding a relatively simple function (linear, exponential, logarithmic, or power). Similarly, all first-degree interaction functions can be understood by looking at a graph of a single univariate function (Theorem 3). However, beyond the first degree, there is a growing variety of interaction functions. Some of them can be nicely decomposed (e.g., $\sin(x_1 x_2)$), whereas others cannot (e.g., $\frac{m_1 m_2}{m_1+m_2}$). c_1 is incapable of differentiating between those interaction functions.

c_2 is designed to address the poor specificity of c_1 for higher-degree interaction functions by considering a decomposition into lower-degree functions. There are two important limitations. First, to make it a well-defined measure, the rules for decomposing need to be specified. An interaction function may have different possible decompositions, so there needs to be a way to make the process deterministic. A possible way to address it is to decide that the most "natural" decomposition is one where the maximum degree of all constituent interaction functions is the lowest (or a similar condition). However, more research is needed to make sure that such rules guarantee uniqueness. The second issue is related to the way interaction decomposition is interpreted. Although a constituent interaction function can have a low degree with respect to a new intermediate variable, the variable itself may be a complex function of the original features. We have not observed this problem in practice due to two reasons. Often, the intermediate variables have an intuitive meaning (e.g., see example in Section 6), and the constraints on interaction functions with respect to other variables may implicitly guarantee that this relationship cannot be too complex. However, this issue still needs to be investigated. It may be helpful to introduce additional constraints on the functions describing intermediate variables.

The limitations of the current implementation are discussed in Appendix B.6.

B.2 Natural Perturbation Operator

In Section 7, we ask the user to choose the most "natural" perturbation operator out of the four possible ones. The notion of "natural" can often be subjective, but we describe a few guidelines.

If there exists a perturbation operator that produces an interaction function with degree 0, then this is the most natural operator. It is also unique.

If none of the interaction functions has degree 0, we search for the perturbation operator that gives an interaction function of the lowest degree. If this degree is 1, then this is a natural perturbation operator. Note that this may not be unique.

If none of the interaction functions has a degree at most 1, then the issue becomes more complex. The most straightforward solution is to choose the one with the lowest degree (this is what we use in our implementation to calculate c_1), but this disregards the variety of higher-degree functions and how different they are to analyze. One way to address this is to try to decompose the interaction functions so that the maximum degree of the constituent interaction functions is the lowest. This would provide a suitable notion of a natural perturbation operator, which we ultimately use in our implementation of c_2 (see Appendix D for details).

B.3 Motivation and Impact

B.3.1 Motivation

Need for Complexity Quantification Although SR originates as a method of rediscovering known physics equations (Schmidt and Lipson, 2009), the appeal of a human-readable representation of the model has led to applications of SR in areas ranging from material science (Wang et al., 2019) to healthcare (Alaa and van der Schaar, 2019). In such places (often without a closed-form ground truth), there will likely be a trade-off between complexity and the goodness of fit. Thus, having good complexity metrics and constraints is paramount.

Limited Research on Complexity Metrics The vast majority of current SR literature focuses on developing optimization algorithms with higher ground-truth recovery (La Cava et al., 2021), faster fitting (Vaddireddy and San, 2019), or scalability (Biggio et al., 2021). However, more research needs to be done into meaningful quantification of complexity (Virgolin et al., 2022). The equations are usually constrained using crude size-based metrics designed to measure the compactness of the equation (such as the depth of the expression tree (Cranmer, 2020), the number of terms (Stephens, 2022), and the description length (Udrescu et al., 2021)). These metrics quantify how compact the equation is but they may not directly correspond to the difficulty of performing tasks such as editing or debugging. These metrics also disregard the nature and order of the operations performed. Although this is addressed by the semantic measures, these approaches are designed to measure non-linearity (Vladislavleva et al., 2009) or to prevent bloat (Vanneschi et al., 2010; Kommenda et al., 2015) rather than to allow for a specific type of analysis. Moreover, most of the current complexity metrics do not take into account variable interactions. They would have the same value if all variables were exchanged for the same single variable. We say they are not *interaction-aware*. They also provide a single numeric value for the whole equation, while complexity may be a more nuanced notion.

B.3.2 Impact

Conceptual Shift We propose a shift in how the complexity of the equation is quantified. Not only do we precisely define the task we would like to perform (SGPA), but we also show how we can arrive at metrics specifically designed for that task.

Interaction-Aware Metric A unique feature of our framework, compared to size-based measures, is that it is "interaction-aware". That means our metrics are not invariant to the exchange of variables in the expression tree. For instance, c_1 would take different values for $y_1 = \sin(x_1)x_2 + \cos(x_1x_2)$ and $y_2 = \sin(x_1)x_1 + \cos(x_2^2)$ as well as $y_3 = \sin(x_1)x_1 + \cos(x_1^2)$. Different values make sense from the analysis point of view. Analyzing y_3 requires inspection of a single univariate function that can be easily plotted $(c_1(y_3) = (1, 0))$. y_2 requires investigating two univariate functions $(c_1(y_2) = (1, 1))$ whereas the complex interactions in y_1 make it quite challenging to analyze. Thus $c_1(y_1) = (2, 2)$. None of the size-based metrics can capture these differences. y_1, y_2, y_3 have the same number of terms, depth of the expression tree, or description length. Most semantic measures are also not "interaction-aware".

Impact on SR Research Interaction functions constitute a new and exciting object to study. They can inspire novel (and meaningful) constraints and innovative optimization algorithms. As we demonstrate in Theorem 4, a function can be reconstructed from its interaction functions. Thus, it may be possible in the future to optimize the interaction functions directly.

Impact on Practical Use Our complexity metrics and constraints will make it easier for practitioners to use SR methods. By operating on meaningful constructs (degree of the interaction function), they will have much better control over the complexity of the found expressions than through current indirect measures (such as the number of terms) that do not guarantee the simplicity of the found expressions (de Franca et al., 2023).

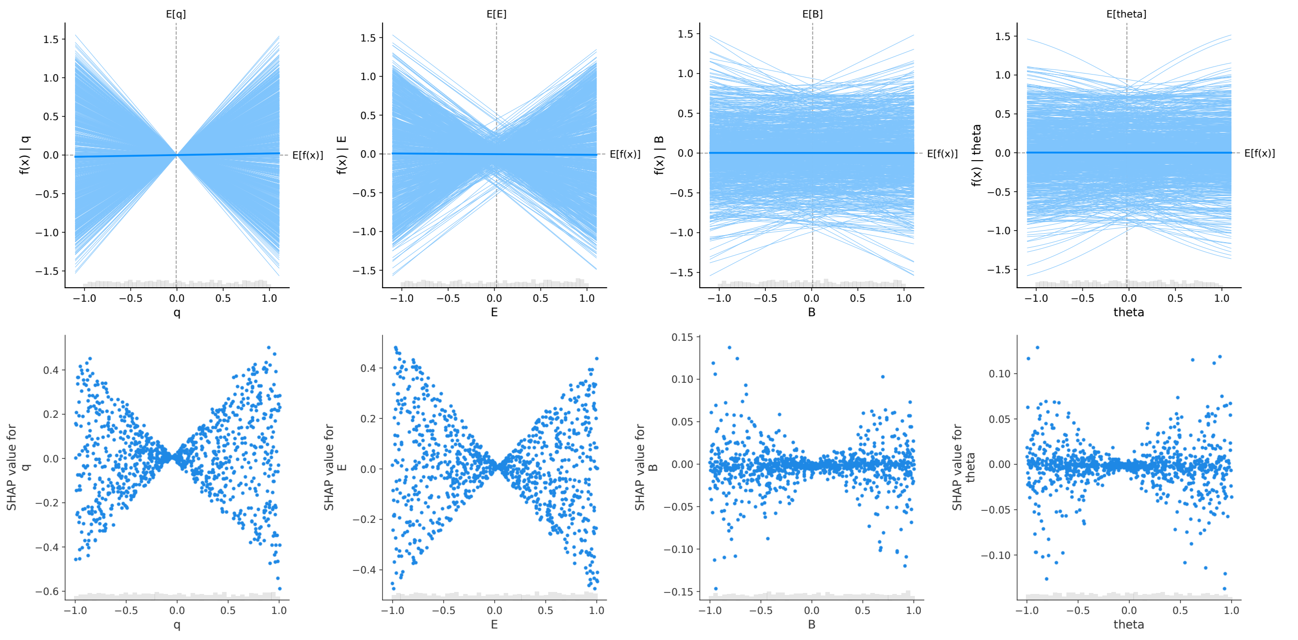
Broader Impacts A better understanding of discovered equations serves critical purposes such as model debugging and identifying and mitigating potential harmful biases. However, complexity measures can also be misused to foster unwarranted trust in models or to achieve surface-level compliance with regulatory standards. Domains such as medicine or finance involve high-stakes scenarios. Therefore, prior to deploying the discovered equation in such contexts, a rigorous examination is necessary to ensure it does not recommend decisions that could be harmful.

B.4 Relationship With XAI

SGPA can usually be done by directly inspecting the equation (see Section 2) and plotting some functions (see Section Section 5), but a human may also calculate the interaction function by hand or use software to do so.

Thus, it is valid to ask whether any of the current eXplainable AI techniques (XAI) (Barredo Arrieta et al., 2020) techniques are helpful in performing SGPA. Local explanation techniques, such as LIME (Ribeiro et al., 2016) or DiCE (Mothilal et al., 2020), are unlikely to be helpful due to the need for global insights. Of course, these are still useful but insufficient for SGPA. That leaves global model explanations such as partial dependence (PD) (Friedman, 2001), individual conditional expectation (ICE) plots (Goldstein et al., 2015), or SHAP (Lundberg and Lee, 2017) (when aggregated over the whole dataset). However, SHAP merely tells us how much a particular feature contributes to the outcome (on average). It does not tell us how much the outcome changes when we change this feature. Although PD plots may tell us "increasing x, increases y on average", we cannot determine whether it is true for all samples and how the amount of increase depends on the other variables. ICE plots show the heterogeneity of the effect of perturbing one of the features, but once again, the plot heavily depends on the dataset used and hides how the curves depend on other features. Even if, in some instances, ICE plots may help form hypotheses (e.g., if all curves are straight lines, the effect of the variable is multiplicative), to verify that it holds for all possible samples, we would need to look at the equation itself, which likely makes using ICE superfluous.

We show XAI techniques (SHAP, PD and ICE plots) applied to the same equation in Figure 1 and demonstrate conceptual differences between the interaction function, PD plot, and ICE plot in Table 3.



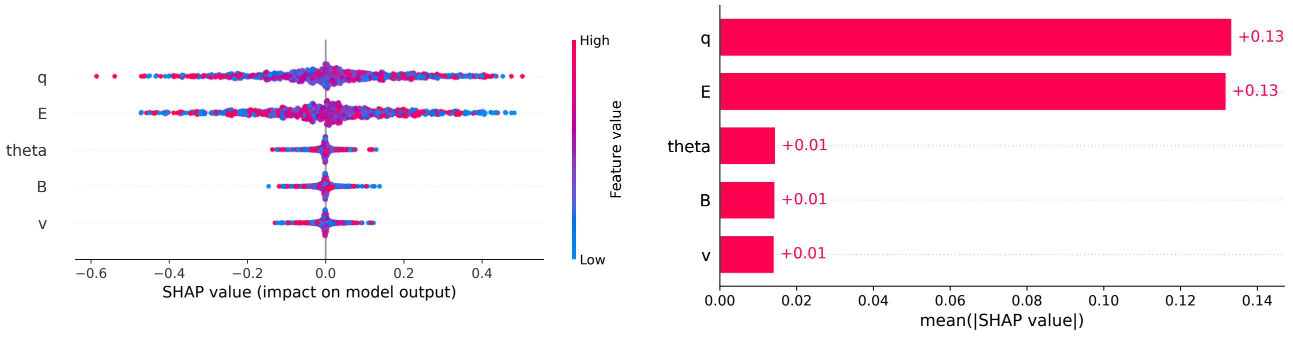


Figure 1: XAI techniques used to analyze equation given by $F = q(E_f + vB\sin(\theta))$ (Equation I.12.11, Lorentz force). The first row shows the partial dependence (PD) plots (the bold lines) for four of the variables (v and B are interchangeable). They do not give much information; they are all constant lines centered at 0. The first row also shows the individual conditional expectation (ICE) curves (the semitransparent lines). These demonstrate the heterogeneous effect of the variables, but it is unclear what determines each curve's shape. It is difficult to infer how perturbing any feature impacts the predicted value. The next two rows show SHAP values and SHAP importance scores. However, SHAP merely tells us how much a particular feature contributes to the outcome (on average). It does not tell us how much the outcome changes when we change this feature. Ultimately, none of the visualizations is likely to help with SGPA.

Table 3: Comparison between the interaction function, partial dependence (PD), and individual conditional expectation (ICE) plots.

Property	Interaction Function	PD plot	ICE plot
Constructed from	a closed-form expression	black box ML model	black box ML model
Goal/Motivation	Measuring the difficulty of performing SGPA (for x_i)	Understanding the impact of a feature x_i for the predicted value	Understanding the impact of a feature x_i for the predicted value
Type of object	$f : \mathbb{R}^{n+1} \to \mathbb{R}$	$f : \mathbb{R} \to \mathbb{R}$	$\{f_d : \mathbb{R} \to \mathbb{R}\}_{d=1}^{D}$
Input	Values of all features and λ	Value of x_i denoted as v	Value of x_i denoted as v
Output	How much the predicted value is perturbed if x_i is perturbed by λ.	What the model predicts on average when each data instance has $x_i = v$.	What the model predicts for each sample d if the feature x_i is set to v.
Usage	Used by the optimization algorithm to constrain the search space of the equations	Visualized as a line plot for a human to explain the black box model	Visualized as a set of line plots for a human to explain the black box model

B.5 Connection to Interpretability

We do not claim that simplicity under the framework represents human-interpretability. Instead, we claim that our framework characterizes how difficult it is to perform SGPA. With no well-established definition of interpretability and a high chance that a general definition does not exist (Rudin et al., 2022), we focus explicitly on this particular task. We do not claim that this corresponds one-to-one with any specific (or general) notion of interpretability. However, there is a connection with the models traditionally deemed "interpretable". In particular, we demonstrate how the simplest models according to our framework (with 0th-degree interaction functions with respect to every variable) correspond to linear regression and the Cox model (Cox, 1975), which are almost universally recognized as some of the most interpretable models (Guidetti et al., 2023). Similarly, generalized additive models (Hastie and Tibshirani, 1986) are characterized by our framework as being relatively simple, having a 1st-degree interaction function with respect to each variable. That coincides with the general recognition of GAMs as intelligible or glass-box models (Caruana et al., 2015; Nori et al., 2019).

B.6 Computational Complexity

The current implementation of the proposed complexity measures uses `sympy`'s `simplify` function, which is quite computationally intensive. It is used to determine the degree of the interaction function. However, we also implemented an empirical evaluation of the degree by sampling the interaction function at different points to determine the degree with high probability. This procedure improves the computational efficiency substantially. c_2 is calculated by checking all possible sequences of perturbation operators, which may become intractable for equations with very deep expression trees. However, we have implemented a "greedy" version of c_2 where the natural perturbation operator is constructed greedily by choosing the operator that gives the smallest degree of the current interaction function.

B.7 Number of Terms and SGPA

The observation that the number of terms may not be directly related to the difficulty of inspecting an equation has been previously made in SR literature. For instance, by de Franca et al. (2023). On page 8, they note: "The simplicity measure only takes into account the size of the expression, but there are some constructs that hinder the interpretability of an expression without adding too much to the size."

Our approach allows us to make this statement much more formal by exchanging the word "interpretability" for "SGPA". This also highlights one of the interesting insights of our approach. We may have equations with very different numbers of terms, e.g., $y = \sin(x)$ (number of terms: 2) and $y = \frac{\exp(-\theta^2/2)}{\sqrt{2\pi}}$ (number of terms: 12). However, both require a similar amount of effort perform SGPA: looking at a single univariate plot. At the same time, as shown in Table 5, we may have equations with the same number of terms, e.g., $y = a_1x_1 + a_2x_2 + a_3x_3$

(number of terms: 11) and $y = a\sqrt{x_3} + b\sqrt{x_1 x_2}$ (number of terms: 11) but performing SGPA on the first one is much easier than on the second one. Indeed, for the first equation,

$$x_1 \mapsto x_1 + \lambda \implies y \mapsto y + a_1\lambda$$

and for the second one

$$x_1 \mapsto x_1 \times \lambda \implies y \mapsto y + b\sqrt{x_1 x_2}(\sqrt{\lambda} - 1)$$

Of course, this equation is probably better analyzed sequentially. We introduce a variable $z = \sqrt{x_1 x_2}$ and write SGPA as follows.

$$x_1 \mapsto x_1 \times \lambda \implies z \mapsto z \times \sqrt{\lambda}$$

$$z \mapsto z \times \lambda \implies y \mapsto y + z(\lambda - 1)$$

Either way, it should be clear that SGPA of the second equation is more involved.

B.8 Equivalent Expressions

In contrast to many other currently used metrics, c_1 metric is invariant to the particular representation of the equation, assuming `sympy`'s `simplify` is able to reduce it appropriately. If we exchange this implementation by the sampling procedure, the metric will always be invariant to equivalent representations. This is, however, not true for the current implementation of c_2, which uses the inherent `sympy` representation of the equation.

B.9 Vector-Valued Functions

SGPA can be trivially extended to multiple dimensions by analyzing the effect independently on each output value. However, we believe a more interesting research direction would be to define new meaningful perturbation vector operators. For instance, based on rotation.

B.10 Difficulty of Performing SGPA on Equations With 0th and 1st-Degree Interaction Functions

0th-Degree Interaction Functions We establish that for each perturbation operator, there is only one 0th-degree family of interaction functions (Theorem 1). Each family describes a simple "constant effect". For instance, $\frac{+f}{+x}[\lambda] = a\lambda$ is interpreted as "for every increase of x by 1, f increases by a", and $\frac{\times f}{+x}[\lambda] = b^{\lambda}$ is interpreted as "for every increase of x by 1, f is multiplied by b". The simplicity of this "constant effect" is reflected by the widespread use of linear regression and the Cox model, where these relationships hold for every variable. As stated by Bradburn et al. (2003), "The Cox model is the most commonly used multivariate approach for analysing survival time data in medical research". Selvin (2008) describes the interpretable nature of the "hazard ratios" (b above) as Cox model's crucial property: "the relative hazard ratio indicates the amount of change in the hazard ratio for a one-unit increase of the jth explanatory variable regardless of the values of the other variables in the model. Additive models are extremely effective statistical tools exactly because of this property". Cox model's widespread use is often specifically attributed to its simplicity and interpretability (Van Ness et al., 2023) and is deemed "most classic and easy-to-interpret model that can estimate the effects of exposure variables ..." (Guidetti et al., 2023).

1st-Degree Interaction Functions We demonstrate that every 1st-degree interaction function can be comprehended by looking at a graph of some univariate function (Theorem 3). Even though the analysis is more complex (as the effect is no longer constant), it can still be performed by looking at a single graph. The fact that this kind of analysis is usually considered easy is reflected by the widespread use of generalized additive models (Hastie and Tibshirani, 1986) whose appeal comes from the fact that understanding the model requires only examining a single univariate function at a time. As stated by Caruana et al. (2015), "Since the contribution of a single feature to the final prediction can be easily understood by examining f_j , such models are considered intelligible".

B.11 Correlation with the proxy of human interpretability

In this section, we investigate the correlation between our complexity metrics and the proxy of human interpretability proposed by Virgolin et al. (2020). We have generated 10000 random equations and calculated their proposed proxy of human interpretability, denoted $\mathcal{M}^{\phi}$, and our metric c_1. We then looked at the correlation between different reduced versions of c_1 (the maximum degree, the mean degree, and the sum of all degrees) and $\mathcal{M}^{\phi}$. The results can be seen in Figure 2. We indeed get moderately strong negative Spearman's correlations (around -0.68) between the two measures in all three cases (i.e., the higher our complexity is, the less interpretable the equation is according to $\mathcal{M}^{\phi}$). This shows the underlying connection of our metrics to interpretability (which we also discuss in Appendix B.5) and the added benefit of our measures. As seen in the figure, there are equations where SGPA is easy to perform but $\mathcal{M}^{\phi}$ is low and there are equations where SGPA is challenging even though $\mathcal{M}^{\phi}$ is high.

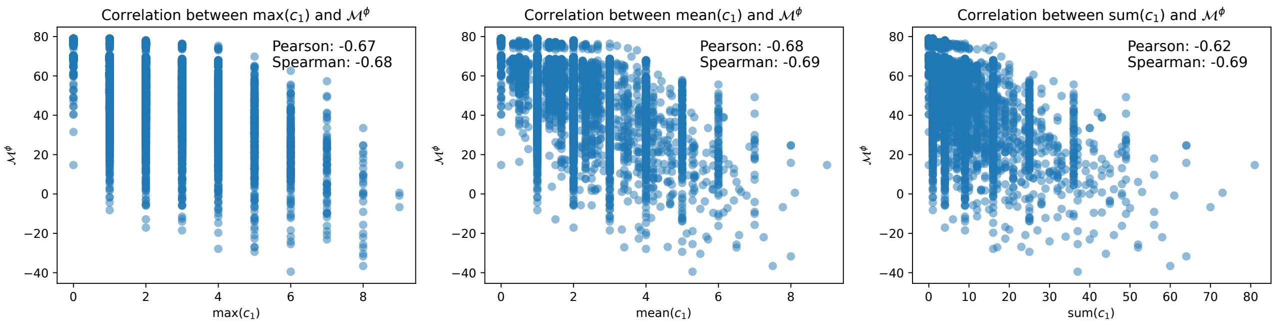


Figure 2: Correlation between different reduced versions of c_1 (the maximum degree, the mean degree, and the sum of all degrees) and $\mathcal{M}^{\phi}$ proposed by Virgolin et al. (2020).

C RELATED WORKS

C.1 Symbolic Regression

Symbolic Regression (SR) is a branch of machine learning that aims to construct a model as a closed-form expression. Traditionally Genetic Programming (Koza, 1994) has been used for this task (Bongard and Lipson, 2007; Schmidt and Lipson, 2009; Cranmer, 2020; Stephens, 2022). Recently, neural networks have been used to either prune the search space of possible expressions (Udrescu and Tegmark, 2020; Udrescu et al., 2021) or to represent the equations directly through their weights (Martius and Lampert, 2017; Sahoo et al., 2018). Other ways neural networks are employed in SR include large pre-trained transformers (Biggio et al., 2021; D'Ascoli et al., 2022; Kamienny et al., 2022), deep reinforcement learning (Petersen et al., 2021) and hybrids of both (Holt et al., 2023). More broadly, symbolic regression also encompasses algorithms for discovering Ordinary Differential Equations (ODEs) (Brunton et al., 2016; Qian et al., 2022; Kaheman et al., 2020; Messenger and Bortz, 2021b) and Partial Differential Equations (Rudy et al., 2017; Long et al., 2019; Raissi and Karniadakis, 2018; Messenger and Bortz, 2021a; Kacprzyk et al., 2023).

C.2 Complexity Measures in SR

Traditional complexity measures can be divided into two kinds: size-based measures and semantic measures. Size-based measures are designed to measure how *compact* the equation is and include metrics based on the depth of the expression tree (Cranmer, 2020; Petersen et al., 2021), the number of terms (Stephens, 2022), and the description length (Udrescu et al., 2021). That also includes sparsity measures employed in methods that represent the closed-form expressions as weights of a neural network with changed activation functions (Sahoo et al., 2018). All these metrics correlate with the difficulty of analyzing an equation, but they disregard the nature and order of the operations performed. To illustrate this limitation, consider a linear regression model.

$$y = a_0 \sum_{n=1}^{N} a_n x_n \tag{96}$$

We established in Section 4 that this kind of model is quite simple to analyze because all interaction functions have degree 0. However, linear regression is not a particularly compact model. Represented as an expression tree,

Table 4: Rules for calculating order of linearity (Vladislavleva et al., 2009). n_f is the minimal degree of a Chebyshev approximation of f with approximation error at most ϵ. n_{div} is is the minimal degree of a Chebyshev approximation of $1/x$ with approximation error at most ϵ.

Term	Complexity C
Constant	0
x_n	1
$f \circ g$	$C(g) \times n_f$
$g_1 + g_2$	$\max\{C(g_1), C(g_2)\}$
$g_1 - g_2$	$\max\{C(g_1), C(g_2)\}$
$g_1 \times g_2$	$C(g_1) + C(g_2)$
g_1/g_2	$C(g_1) + C(g_2) \times n_{\text{div}}$

it has N additions and $N+1$ leaves. Its depth can range from around $\log(N)$ for a "symmetric" tree up to N for a very one-sided tree. Moreover, the compactness of this expression would not change if we changed any of the addition operators for multiplication or division. However, that might make analyzing such expressions much more difficult.

This means we require measures that take into account the type and order of the operations performed. This is what semantic measures aim to do. One of the earliest examples was the order of non-linearity (Vladislavleva et al., 2009) that uses Chebyshev polynomials to arrive at the rules for calculating complexity (Table 4). Although this measure may be fitting for measuring how non-linear the model is, it is not clear how that is related to any downstream task. The biggest shortcoming of that metric is the fact that it is not "interaction-aware". The active variables of functions that are added or multiplied together are not taken into consideration. Another concerning feature is that multiplication and addition are treated very differently, whereas we demonstrated in Section 2 that they are quite similar from an SGPA perspective.

Another semantic measure was introduced by Kommenda et al. (2015) to simplify measures such as the one by Vladislavleva et al. (2009) while still providing information about the semantics of the expression. The rules of calculating it are presented in Table 5. Although this measure is indeed easier to calculate as it does not require any kind of interpolation, it loses the theoretical underpinning of Vladislavleva et al. (2009) and thus becomes more arbitrary. For instance, it is unclear why composition with exp has a drastically different effect than composition with $\sqrt{}$. Especially as we saw in Section 4 we can have very simple functions with both of them. Similarly to the order of linearity, multiplication is treated very differently from addition, and the measure is not interaction-aware. How divergent this measure is from the framework we propose can be illustrated using the example from Kommenda et al. (2015). Let $f_1(x) = e^{\sin(\sqrt{x})}$ and $f_2(x) = 7x^2 + 3x + 5$. Our framework would classify f_1 and f_2 as quite similar as they are both univariate functions. Their interaction functions have degrees 1, and they can be analyzed by looking at their line graph. However, the complexity measure proposed by Kommenda et al. (2015) assigns value 17 to f_2 and value 65536 to f_1.

Functional complexity proposed by Vanneschi et al. (2010) has a quite involved definition that looks at the actual values the function takes rather than their symbolic representation. As analysis is mostly performed by inspecting the symbolic form (and possibly plotting the univariate functions), this measure is unsuitable for this task.

All proposed measures also provide a single numeric value for the whole equation, while our measures are specific to each variable. An orthogonal research direction is learning the complexity measure automatically from human feedback (Virgolin et al., 2020, 2021).

D IMPLEMENTATION

We implemented the described complexity metrics and constraints as a lightweight Python module compatible with `sympy` (Meurer et al., 2017). The code is available at `https://github.com/krzysztof-kacprzyk/SGPA-in-SR`.

The `Perturbation` class allows the calculation of interaction functions using symbolic manipulation in `sympy`. It can be used as follows.

Table 5: Rules for calculating the complexity measure by Kommenda et al. (2015). u refers to univariate functions $\sin, \cos, \tan, \exp, \log$.

Term	Complexity C
Constant	1
x_n	2
$f \circ g$	$C(g) \times n_f$
$g_1 + g_2$	$C(g_1) + C(g_2)$
$g_1 - g_2$	$C(g_1) + C(g_2)$
$g_1 \times g_2$	$C(g_1) \times C(g_2) + 1$
g_1 / g_2	$C(g_1) \times C(g_2) + 1$
f^2	$C(f)^2$
$\sqrt{f}$	$C(f)^3$
$u \circ f$	$2^{C(f)}$

```
import sympy as sp
from sgpa import Perturbation
F, theta = sp.symbols("F theta")
r = sp.symbols("r", positive=True)
y = r*F*sp.sin(theta)
print(Perturbation("*","*",r)(y))
```

```
l
```

Here l stands for λ. The function `complexity_1` can be used to calculate c_1 or its "reduced" version.

```
import numpy as np
from sgpa import complexity_1
print(complexity_1(y,[r,F,theta]))
print(complexity_1(y,[r,F,theta],reduce_strategy="max"))
print(complexity_1(y,[r,F,theta],reduce_strategy=np.mean))
# theta treated as a constant:
print(complexity_1(r**theta,[r]))
```

```
{r: 0, F: 0, theta: 1}
1
0.3333333333333333
{r: 0}
```

The default strategy for choosing the perturbation operator is to select the one yielding the interaction function with the smallest degree. However, you can pass your own rule using the `operator_strategy` argument.

We believe our metric's vector-valued nature is an advantage, as it allows a more detailed description of the complexity. However, in some scenarios, a single number may be required. Thus, we have added `reduce_strategy` arguments (shown above), which reduce the whole vector to a single value.

`complexity_2` calculates c_2 (or its reduced versions) using the inherent decomposition in the `sympy` expression (accessed through the `ComputationTree` class).

```
from sgpa import ComputationTree
G, m1, m2, r1, r2 = sp.symbols("G m1 m2 r1 r2")
f = G*m1*m2*(1/r2) - G*m1*m2*(1/r1)
print(ComputationTree(f, [m1,m2,r1,r2]))
```

```
f=f0 + f1
f0=G*f00*m1*m2
f00=1/r2
f1=-G*f10*m1*m2
f10=1/r1
```

Based on the expression tree extracted from the representation in `sympy`, for each variable, we calculate a "variable queue". Let us consider the variable r_1 in the example above. We start from the root. $f = f_0 + f_1$. We check that only one of the two intermediate variables depends on r_1, namely f_1. Thus by Proposition 2 we know that $\frac{\circledast_1 f}{\circledast_2 r_1} = \frac{\circledast_1 f}{+ f_1} \otimes \frac{+ f_1}{\circledast_2 r_1}$. If more than one intermediate variable had depended on r_1, the algorithm would have stopped traversing the computation tree. In our case, the algorithm continues. As only one of the variables in $f_1 = -G f_{10} m_1 m_2$ depends on r_1, namely f_{10}, we add it to the queue. At the end, our queue contains the following variables: f, f_1, f_{10}, r_1. Thus we want to represent the interaction function as $\frac{\circledast_1 f}{\circledast_2 f_1} \otimes \frac{\circledast_2 f_1}{\circledast_3 f_{10}} \otimes \frac{\circledast_3 f_{10}}{\circledast_4 r_1}$. We then test all possible combinations of $(\circledast_1, \circledast_2, \circledast_3, \circledast_4) \in \{+, \times\}^4$ and for each of those, we calculate the degree of each of the constituent interaction functions. Thus, to each sequence $(\circledast_1, \circledast_2, \circledast_3, \circledast_4)$ we associate a vector $\left(\deg\left(\frac{\circledast_1 f}{\circledast_2 f_1}\right), \deg\left(\frac{\circledast_2 f_1}{\circledast_3 f_{10}}\right), \deg\left(\frac{\circledast_3 f_{10}}{\circledast_4 r_1}\right)\right)$. Then, we process each vector by replacing any sequence of consecutive 0s with a single 0 (justified by Proposition 3). Then, among all the sequences, we filter the ones with the smallest maximum degree (among the intermediate interaction functions). Finally, we return the shortest sequence among those.

```
from sgpa import complexity_2
print(complexity_2(f,[m1,m2,r1,r2]))
print(complexity_2(f,[m1,m2,r1,r2], reduce_strategy="max_all"))
```

```
{m1: [0], m2: [0], r1: [1, 0], r2: [1, 0]}
1
```

We also implemented the three constraints described in the paper.

```
from sgpa import constraint_1
print(constraint_1(y, [r,F,theta])) # False
print(constraint_1(y, [r,F])) # True
print(constraint_1(f,[m1,m2,r1,r2])) # False
```

```
False
True
False
```

```
from sgpa import constraint_2
print(constraint_2(y, [r,F,theta]))
print(constraint_2(y, [r,F]))
print(constraint_2(f,[m1,m2,r1,r2]))
```

```
True
True
False
```

```
from sgpa import constraint_3
print(constraint_3(y, [r,F,theta]))
print(constraint_3(y, [r,F]))
print(constraint_3(f,[m1,m2,r1,r2]))
```

```
True
True
True
```

Empirical Computation of the Degree As sympy's simplification process can be time-intensive, we also implemented the option to evaluate the degree empirically. We do it by evaluating the function on randomly sampled points and checking whether the value of the function changes. This is what we use in our extension of gplearn.

Greedy Choice of the Natural Perturbation Operator We have implemented a "greedy" version of c_2 where the natural perturbation operator is constructed greedily by choosing the next operator that gives the smallest degree of the current interaction function.

Licenses Our code uses the following dependencies:

- jupyter 1.0.0 BSD-3-Clause license
- numpy 1.26.40 BSD license
- python 3.12.2 PSF license agreement
- sympy 1.12 BSD license
- gplearn 0.42 BSD-3-Clause license

E DEMONSTRATIONS

E.1 Reconstruction

In this section, we show how Theorem 4 looks in practice. Consider

$$U(m_1, m_2, r_1, r_2) = Gm_1m_2(\tfrac{1}{r_2} - \tfrac{1}{r_1}) \tag{97}$$

We have already calculated interaction functions of U in Section 6. Let us reiterate those:

1. $\frac{\times U}{\times m_1}[\lambda] = \lambda$
2. $\frac{\times U}{\times m_2}[\lambda] = \lambda$
3. $\frac{+U}{\times r_1}[\lambda] = Gm_1m_2(\frac{1}{r_1} - \frac{1}{\lambda r_1})$
4. $\frac{+U}{\times r_2}[\lambda] = Gm_1m_2(\frac{1}{\lambda r_2} - \frac{1}{r_2})$

To reconstruct U from its interaction function, we use Theorem 4 with $\circledast_1 = \times$, $\circledast_1' = \times$, $\circledast_2 = \times$, $\circledast_2' = \times$, $\circledast_3 = +$, $\circledast_3' = \times$, $\circledast_4 = +$, $\circledast_4' = \times$. Consequently $e_{\circledast_1'} = 1$, $e_{\circledast_2'} = 1$, $e_{\circledast_3'} = 1$, $e_{\circledast_4'} = 1$. The result of the theorem is the following:

$$\left(\times [m_1] \circ \times[m_2] \circ +[G \times 1 \times 1(\tfrac{1}{1} - \tfrac{1}{r_1 \times 1})] \circ +[G \times 1 \times 1(\tfrac{1}{r_2 \times 1} - \tfrac{1}{1})] \right)(G \times 1 \times 1(\tfrac{1}{1} - \tfrac{1}{1})) \tag{98}$$

After simplification:

$$\left(\times [m_1] \circ \times[m_2] \circ +[G(1 - \tfrac{1}{r_1})] \circ +[G(\tfrac{1}{r_2} - 1)] \right)(0) \tag{99}$$

We can rewrite using infix notation.

$$\begin{aligned} m_1 \times (m_2 \times (G(1 - \tfrac{1}{r_1}) + (G(\tfrac{1}{r_2} - 1) + 0))) &= m_1 \times (m_2 \times (G(1 - \tfrac{1}{r_1}) + (G(\tfrac{1}{r_2} - 1)))) \\ &= m_1 \times (m_2 \times (G - G\tfrac{1}{r_1} + G\tfrac{1}{r_2} - G)) \\ &= m_1 \times (m_2 \times (G\tfrac{1}{r_2} - G\tfrac{1}{r_1})) \\ &= Gm_1m_2(\tfrac{1}{r_2} - \tfrac{1}{r_1}) \\ &= U \end{aligned} \tag{100}$$

In this case, we ordered variables as (m_1, m_2, r_1, r_2), but different ordering should give the same result. For instance, consider ordering (r_1, m_1, r_2, m_2). Then, the reconstruction is performed as follows.

$$\left(+ [Gm_1m_2(\tfrac{1}{1} - \tfrac{1}{r_1 \times 1})] \circ \times[m_1] \circ +[G \times 1 \times m_2(\tfrac{1}{r_2 \times 1} - \tfrac{1}{1})] \circ \times[m_2] \right)(0) \tag{101}$$

After simplification and rewriting using infix notation, we obtain:

$$\begin{aligned} Gm_1m_2(1 - \tfrac{1}{r_1}) + (m_1 \times (Gm_2(\tfrac{1}{r_2} - 1) + (m_2 \times 0))) &= Gm_1m_2(1 - \tfrac{1}{r_1}) + (m_1 \times (Gm_2(\tfrac{1}{r_2} - 1))) \\ &= Gm_1m_2(1 - \tfrac{1}{r_1}) + (Gm_1m_2(\tfrac{1}{r_2} - 1)) \\ &= Gm_1m_2(\tfrac{1}{r_2} - \tfrac{1}{r_1}) \\ &= U \end{aligned} \tag{102}$$

E.2 Case Studies: Comparison With Other Complexity Metrics

To better demonstrate the added value of our framework and its effectiveness, we have added a few case studies of different equations where we quantify how complex they are with respect to three different metrics: the number of nodes (size-based metric) denoted as N, the metric proposed by Kommenda et al. (2015) (semantic metric) denoted K and c_1 (one of the metrics we propose). We chose equations with a similar number of terms (10 or 11) so that it is easier to compare them. The results are summarized in Table 6.

In all equations below, x_1, x_2, x_3 are the independent variables, whereas a, b, c denote constants.

Even though c_1 is the most straightforward possible metric in our framework, we justify below why it already better characterizes the difficulty of performing SGPA than the two alternatives.

Three equations can be classified as the simplest according to c_1 ($c_1(y) = (0, 0, 0)$): equations 1, 3, and 5. We believe these are indeed easy to analyze because they correspond to the following simple relationships:

Table 6: Complexity of selected equations with respect to three metrics: the number of nodes (size-based metric) denoted as N, the metric proposed by Kommenda et al. (2015) (semantic metric) denoted K and c_1 (one of the metrics we propose).

Number	Equation	$N(y)$	$K(y)$	$c_1(y)$
1	$y = a_1x_1 + a_2x_2 + a_3x_3$	11	18	(0,0,0)
2	$y = a\sqrt{x_3} + b\sqrt{x_1x_2}$	11	1478	(0,2,2)
3	$y = (x_1x_2x_3)^2$	11	729	(0,0,0)
4	$y = a\sin(x_1) + b\cos(x_1^2)$	11	1036	(1,0,0)
5	$y = e^{ax_1+bx_2+x_3}$	10	16384	(0,0,0)
6	$y = \sin\left(\sqrt{x_1}\right)^2 + x_2 + x_3$	11	65540	(1,0,0)
7	$y = e^{\sin(\sqrt{x_1})} + bx_2 + cx_3$	10	1.15e+77	(1,0,0)
8	$y = \sin(\cos(x_1)) + a\sin(x_2) + \cos(x_3)$	11	30	(1,1,1)
9	$y = (x_1 + x_2 + a)^2$	11	25	(2,2,0)

- (1) y increases by a_i when x_i is increased by 1
- (3) y gets multiplied by λ^2 when x_i is multiplied by λ
- (5) y gets multiplied by e^{a_i} $(i \in 1, 2)$ or e $(i = 3)$ when x_i is increased by 1

The semantic metric K assigns wildly different complexities to these three equations, from 18 to 16384. We find this counterintuitive.

Equations 4, 6, and 7 require inspection of exactly one univariate function, and as univariate functions can be relatively straightforward to investigate by plotting them, we deem them slightly more challenging than the previous three equations. This perfectly aligns with $c_1(y)_1$ being equal to 1 instead of 0. As all equations have 10 or 11 terms, the number of terms does not differentiate between those two types of analysis. K complexity once again varies a lot: from 1036 to 1.15e+77.

Equation 8 constitutes an interesting example as it has one of the lowest K complexities ($K(y) = 30$)) and much lower than the simplest equation considered in the paragraph above (equation 4, where $K(y) = 1036$) even though equation 8 requires investigating not one (as in eq. 4) but three separate univariate functions. This is perfectly captured by $c_1(y) = (1, 1, 1)$.

Finally, equations 2 and 9 impede SGPA as there are strong interactions between x_1 and x_2. This is reflected in c_1 but is poorly captured by K, which assigns a complexity of only 25 to equation 9.

It is important to note that although these equations require different amounts of effort to perform SGPA, the number of terms (N) is basically the same. Thus, although a prevalent metric for simplicity (La Cava et al., 2021; de Franca et al., 2023), it remains a relatively poor indicator.

E.3 Case Study: gplearn

To illustrate the practical advantage of our complexity metrics, we have tested gplearn on the Concrete dataset (Yeh, 1998) (UCI dataset) with our second constraint ($\max(c_1(f)) \leq 1$) and the traditional constraints based on the length of the program $|f|$. The results can be seen in the table below (Table 7). We split the dataset into training and test subsets (0.8:0.2). We report the mean absolute error on the test dataset.

$$\begin{aligned}&\sin\left(\frac{X_0}{2} + \frac{X_7}{2} + \frac{\sin\left(X_1\right)}{2} - \frac{\sin\left(\sin\left(X_3\right) - \sin\left(X_4\right)\right)}{2}\right) + \\ &\sin\left(\sin\left(X_7 + \sin\left(0.21X_0 + 1.21X_7 + 0.21\sqrt{\left|\log\left(\left|\sin\left(X_4\right) + 0.21\right|\right)\right|}\right) + \sin\left(\sqrt[4]{\left|\sin\left(X_3\right)\right|}\right)\right)\right)\end{aligned} \tag{103}$$

Table 7: The results of gplearn on the Concrete dataset with different constraints. We report mean absolute error on the test dataset.

Constraint	Equation	MAE
$\lvert f\rvert \leq 10$	$0.46X_0 + 0.46\sin(X_4) + \sin(X_7)$	0.517
$\lvert f\rvert \leq 20$	$\sin(X_7 + \sin(X_7) + 0.71) + \sin(0.41(X_0 + X_4 + X_7 + \sin(X_1)))$	0.422
$\lvert f\rvert \leq 50$	Equation (103)	0.328
$\lvert f\rvert \leq 90$	Equation (104)	0.320
$max(c_1(f)) \leq 1$	Equation (105)	0.289

$$\begin{aligned}&\sin\left(\frac{X_0}{2} + \frac{X_7}{2} + \frac{\sin(X_1)}{2} + \frac{\sin(X_4)}{2}\right)\\&+ \sin\Big(\sin\Big(X_7 + \sin\Big(0.29X_0 + 0.09X_1 + 0.09X_2 + 1.29X_7\\&- 0.21\sin(X_3) - 0.21\sin\Big(X_4 + X_7 + \sin\Big(-e^{X_0+\sin(\sin(X_4))} + \sin(X_3) + \sin(X_4)\Big)\Big)\\&- 0.09\sin(\sin(X_3 - X_7)) - 0.21\sin\Big(\sin\Big(X_7 + \sin(X_4)\\&+ \sin(\sin(X_4))\Big)\Big) + 0.09\sqrt{|\sin(\sin(X_3))|}\Big) + 0.76\Big)\Big)\end{aligned} \tag{104}$$

$$0.61X_0 + g_1(X_1) + 0.08X_2 - 0.44\sin(0.77X_3) + g_4(X_4) + g_7(X_7) \tag{105}$$

where

$$\begin{aligned}g_1(X_1) &= 0.44\sin(\log(|X_1 + 0.99|))\\g_4(X_4) &= 0.19\sin(\sin(X_4 + \sin(X_4) + 0.44)) + 0.08\\g_7(X_7) &= 0.44\log\left(\left|\sin\left(\log(|X_7 + 0.99|) + 0.19\sqrt{|\log(|\sin(2\log(|X_7 + 0.99|)) + 0.99|)|}\right) + 0.99\right|\right)\\&+ 0.44\sin(X_7 + 0.99) + 0.44\sin(\log(|X_7 + 0.99|))\end{aligned} \tag{106}$$

We can see that not only does the equation found with our constraint have a better performance, but, more importantly, SGPA is easier to perform. Each of the univariate functions g_1, g_4, g_7 can be analyzed by their line plot, so their intricate symbolic form is not a problem. In contrast, Equations (103) and (104) lack such modular structure, which makes performing SGPA on them much more challenging. The interaction functions have much higher degrees, and there is no decomposition that can lower the degrees significantly. This can also be observed in the second row of the table, even though the equation is very short.

E.4 SimplEq

Table 8 shows results of running SimplEq on FSReD. We split the dataset into training and test subsets (0.8:0.2). The R^2 score is calculated on the test dataset. The experiments were performed on 12th Gen Intel(R) Core i7-12700H with 64 GB of RAM.

E.5 Complexity and Constraints in FSReD

In Tables 9 to 11 we show all 100 equations from FSReD. We also calculate c_1 complexity measures and specify whether the equations satisfy the constraints defined in Section 7.

Table 8:

Property	Value
Number of correct equations	12/100
Number of equations with $R^2 > 0.99$	34/100
Number of equations with $R^2 > 0.9$	61/100
Median R^2 score	0.95
Average R^2 score	0.82
Minimum time to fit an equation	0.28 seconds
Median time to fit an equation	5.38 seconds
Average time to fit an equation	42.45 seconds

Table 9: This table lists equations from FSReD (Udrescu and Tegmark, 2020) along with their complexity c_1 and satisfiability of the three types of constraints (as defined in Section 7).

Number	Equation	Variables	c_1	$c_1(f)_n = 0$	$c_1(f)_n \le 1$	$\lvert c_2(f)_n \rvert \le 2$, $c_2(f)_{n,m} \le 1$
I.6.20a	$f = \frac{\exp(-\theta^2/2)}{\sqrt{2\pi}}$	θ	(0)	✗	✓	✓
I.6.20	$f = \frac{\exp(-(\theta/\sigma)^2/2)}{\sqrt{2\pi\sigma^2}}$	θ, σ	(2,2)	✗	✗	✗
I.6.20b	$f = \frac{\exp(-\frac{(\theta-\theta_1)^2}{2\sigma^2})}{\sqrt{2\pi\sigma^2}}$	θ, σ, θ_1	(3,3,3)	✗	✗	✗
I.8.14	$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	x_1, x_2, y_1, y_2	(4,4,4,4)	✗	✗	✗
I.9.18	$F = \frac{Gm1m2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$	$m_1, m_2, x_1, x_2, y_1, y_2, z_1, z_2$	(0,0,6,6, 6,6,6,6)	✗	✗	✗
I.10.7	$m = \frac{m_0}{\sqrt{1-v^2/c^2}}$	m_0, v	(0,1)	✗	✓	✓
I.11.19	$A = x_1y_1 + x_2y_2 + x_3y_3$	$x_1, x_2, x_3, y_1, y_2, y_3$	(1,1,1, 1,1,1)	✗	✓	✓
I.12.1	$F = \mu N_n$	μ, N_n	(0,0)	✓	✓	✓
I.12.2	$F = \frac{q_1q_2r}{4\pi\epsilon r^3}$	q_1, q_2, r	(0,0,0)	✓	✓	✓
I.12.4	$E_f = \frac{q_1r}{4\pi\epsilon r^3}$	q_1, r	(0,0)	✓	✓	✓
I.12.5	$F = q_2E_f$	q_2, E_f	(0,0)	✓	✓	✓
I.12.11	$F = q(E_f + Bv\sin(\theta))$	q, E_f, B, v, θ	(0,1,3,3,3)	✗	✗	✓
I.13.4	$K = 1/2m(v^2 + u^2 + w^2)$	m, v, u, w	(0,2,2,2)	✗	✗	✓
I.13.12	$U = Gm_1m_2(\frac{1}{r_2} - \frac{1}{r_1})$	m_1, m_2, r_1, r_2	(0,0,2,2)	✗	✗	✓
I.14.3	$U = mgz$	m, g, z	(0,0,0)	✓	✓	✓
I.14.4	$U = 1/2k_{spring}x^2$	k_{spring}, x	(0,0)	✓	✓	✓
I.15.3x	$x_1 = \frac{x-ut}{\sqrt{(1-u^2/c^2)}}$	x, u, t	(1,3,1)	✗	✗	✗
I.15.3t	$t_1 = \frac{t-ux/c^2}{\sqrt{(1-u^2/c^2)}}$	t, u, x	(1,3,1)	✗	✗	✗
I.15.10	$p = \frac{m_0v}{\sqrt{(1-v^2/c^2)}}$	m_0, v	(0,1)	✗	✓	✓
I.16.6	$v_1 = \frac{u+v}{1+uv/c^2}$	u, v	(2,2)	✗	✗	✗
I.18.4	$r = \frac{m_1r_1+m_2r_2}{m_1+m_2}$	r_1, r_2, m_1, m_2	(2,2,4,4)	✗	✗	✗
I.18.12	$\tau = rF\sin(\theta)$	r, F, θ	(0,0,1)	✗	✓	✓
I.18.14	$L = mrv\sin(\theta)$	m, r, v, θ	(0,0,0,1)	✗	✓	✓
I.24.6	$E_n = \frac{1}{4}m(\omega^2 + \omega_0^2)x^2$	m, ω, ω_0, x	(0,2,2,0)	✗	✗	✓
I.25.13	$V = \frac{q}{C}$	q, C	(0,0)	✓	✓	✓
I.26.2	$\theta_1 = \arcsin(n\sin(\theta_2))$	n, θ_2	(2,2)	✗	✗	✓
I.27.6	$f_f = \frac{1}{\frac{1}{d_1}+\frac{n}{d_2}}$	n, d_1, d_2	(3,3,3)	✗	✗	✗
I.29.4	$k = \frac{\omega}{c}$	ω	(0)	✓	✓	✓
I.29.16	$x = \sqrt{x_1^2 + x_2^2 - 2x_1x_2\cos(\theta_1 - \theta_2)}$	$x_1, x_2, \theta_1, \theta_2$	(4,4,4,4)	✗	✗	✗
I.30.3	$I{=}I_0\frac{\sin(n\theta/2)^2}{\sin(\theta/2)^2}$	I_0, n, θ	(0,2,2)	✗	✗	✗
I.30.5	$\theta = \arcsin(\frac{\lambda}{nd})$	λ, n, d	(3,3,3)	✗	✗	✓
I.32.5	$P = \frac{q^2a^2}{6\pi\epsilon c^3}$	$q, a\epsilon, c$	(0,0,0,0)	✓	✓	✓
I.32.17	$P = (\frac{1}{2}\epsilon cE_f^2)(8\pi r^2/3)\frac{\omega^4}{(\omega^2-\omega_0^2)^2}$	$\epsilon, E_f, r, \omega, \omega_0$	(0,0,0,2,2)	✗	✗	✓
I.34.8	$\omega = \frac{qvB}{p}$	q, v, B, p	(0,0,0,0)	✓	✓	✓
I.34.1	$\omega = \frac{\omega_0}{1-v/c}$	ω_0, v	(0,1)	✗	✓	✓
I.34.14	$\omega = \frac{1+v/c}{\sqrt{1-v^2/c^2}}\omega_0$	ω_0, v	(0,1)	✗	✓	✓
I.34.27	$E = \hbar\omega$	ω	(0)	✓	✓	✓
I.37.4	$I_* = I_1 + I_2 + 2\sqrt{I_1I_2}\cos(\delta)$	l_1, l_2, δ	(3,3,3)	✗	✗	✗
I.38.12	$r = \frac{4\pi\epsilon\hbar^2}{mq^2}$	ϵ, m, q	(0,0,0)	✓	✓	✓
I.39.1	$E = 3/2prV$	p_F, V	(0,0)	✓	✓	✓
I.39.11	$E = \frac{1}{\gamma-1}prV$	γ, p_F, V	(1,0,0)	✗	✓	✓
I.39.22	$p_F = \frac{nk_bT}{V}$	n, k_b, T, V	(0,0,0,0)	✓	✓	✓

Table 10: This table lists equations from FSReD (Udrescu and Tegmark, 2020) along with their complexity c_1 and satisfiability of the three types of constraints (as defined in Section 7).

Number	Equation	Variables	c_1	$c_1(f)_n = 0$	$c_1(f)_n \leq 1$	$\lvert c_2(f)_n \rvert \leq 2$, $c_2(f)_{n,m} \leq 1$
I.40.1	$n = n_0 \exp(-\frac{mgx}{k_b T})$	n_0, m, g, x, k_b, T	(0,5,5,5,5,5)	✗	✗	✓
I.41.16	$L_{rad} = \frac{\hbar\omega^3}{\pi^2 c^2(\exp(\frac{\hbar\omega}{k_b T})-1)}$	ω, k_b, T	(3,3,3)	✗	✗	✗
I.43.16	$v = \frac{\mu_{\text{drift}} q V_e}{d}$	$\mu_{\text{drift}}, q, V_e, d$	(0,0,0,0)	✓	✓	✓
I.43.31	$D = \mu_e k_b T$	μ_e, k_b, T	(0,0,0)	✓	✓	✓
I.43.43	$\kappa = \frac{1}{\gamma-1} \frac{k_b v}{A}$	γ, k_b, v, A	(1,0,0,0)	✗	✓	✓
I.44.4	$E = n k_b T \ln\left(\frac{V_2}{V_1}\right)$	n, k_b, T, V_1, V_2	(0,0,0,2,2)	✗	✗	✓
I.47.23	$c = \sqrt{\frac{\gamma p r}{\rho}}$	γ, p, r, ρ	(0,0,0,0)	✓	✓	✓
I.48.2	$E = \frac{mc^2}{\sqrt{1-v^2/c^2}}$	m, v	(0,1)	✗	✓	✓
I.50.26	$x = x_1(\cos(\omega t) + \alpha\cos(\omega t)^2)$	x_1, ω, t, α	(0,3,3,2)	✗	✗	✗
II.2.42	$P = \frac{\kappa(T_2-T_1)A}{d}$	κ, A, d, T_1, T_2	(0,0,0,2,2)	✗	✗	✓
II.3.24	$F_E = \frac{P}{4\pi r^2}$	P, r	(0,0)	✓	✓	✓
II.4.23	$V_e = \frac{q}{4\pi\epsilon r}$	q, ϵ, r	(0,0,0)	✓	✓	✓
II.6.11	$V_e = \frac{1}{4\pi\epsilon} \frac{p_d \cos(\theta)}{r^2}$	ϵ, p_d, θ, r	(0,0,1,0)	✗	✓	✓
II.6.15a	$E_f = \frac{p_d}{4\pi\epsilon} \frac{3z}{r^5} \sqrt{x^2+y^2}$	$\epsilon, p_d, z, r, x, y$	(0,0,0,0,2,2)	✗	✗	✓
II.6.15b	$E_f = \frac{3}{4\pi\epsilon} \frac{p_d}{r^3} \cos(\theta)\sin(\theta)$	ϵ, p_d, r, θ	(0,0,0,1)	✗	✓	✓
II.8.7	$E = \frac{3}{5} \frac{q^2}{4\pi\epsilon d}$	q, ϵ, d	(0,0,0)	✓	✓	✓
II.8.31	$E_{den} = \frac{\epsilon E_f^2}{2}$	ϵ, E_f	(0,0)	✓	✓	✓
II.10.9	$E_f = \frac{\sigma_{den}}{\epsilon} \frac{1}{1+\chi}$	$\sigma_{den}, \epsilon, \chi$	(0,0,1)	✗	✓	✓
II.11.3	$x = \frac{q E_f}{m(\omega_0^2-\omega^2)}$	$q, E_f, m, \omega, \omega_0$	(0,0,0,2,2)	✗	✗	✓
II.11.17	$n = n_0(1 + \frac{p_d E_f \cos(\theta)}{k_b T})$	$n_0, p_d, E_f, \theta, k_b, T$	(0,5,5,5,5,5)	✗	✗	✓
II.11.20	$P_* = \frac{n_\rho p_d^2 E_f}{3 k_b T}$	n_ρ, p_d, E_f, k_b, T	(0,0,0,0,0)	✓	✓	✓
II.11.27	$P_* = \frac{n\alpha}{1-(n\alpha/3)} \epsilon E_f$	n, α, ϵ, E_f	(2,2,0,0)	✗	✗	✓
II.11.28	$\theta = 1 + \frac{n\alpha}{1-(n\alpha/3)}$	n, α	(2,2)	✗	✗	✓
II.13.17	$B = \frac{1}{4\pi\epsilon c^2} \frac{2I}{r}$	ϵ, l, r	(0,0,0)	✓	✓	✓
II.13.23	$\rho_c = \frac{\rho_{c_0}}{\sqrt{1-v^2/c^2}}$	ρ_{c_0}, v	(0,1)	✗	✓	✓
II.13.34	$j = \frac{rho_{c_0} v}{\sqrt{1-v^2/c^2}}$	ρ_{c_0}, v	(0,1)	✗	✓	✓
II.15.4	$E_n = -\mu_m B \cos(\theta)$	μ_M, B, θ	(0,0,1)	✗	✓	✓
II.15.5	$E_n = -p_d E_f \cos(\theta)$	p_d, E_f, θ	(0,0,1)	✗	✓	✓
II.21.32	$V_e = \frac{q}{4\pi\epsilon r(1-v/c)}$	q, ϵ, r, v	(0,0,0,1)	✗	✓	✓
II.24.17	$k = \sqrt{\frac{\omega^2}{c^2} - \frac{\pi^2}{d^2}}$	ω, d	(2,2)	✗	✗	✓
II.27.16	$F_E = \epsilon c E_f^2$	ϵ, E_f	(0,0)	✓	✓	✓
II.27.18	$E_{den} = \epsilon E_f^2$	ϵ, E_f	(0,0)	✓	✓	✓
II.34.2a	$I = \frac{qv}{2\pi r}$	q, v, r	(0,0,0)	✓	✓	✓
II.34.2	$\mu_M = \frac{qvr}{2}$	q, v, r	(0,0,0)	✓	✓	✓
II.34.11	$\omega = \frac{g_q B}{2m}$	g,q, B, m	(0,0,0,0)	✓	✓	✓
II.34.29a	$\mu_M = \frac{qh}{4\pi m}$	q, h, m	(0,0,0)	✓	✓	✓
II.34.29b	$E = \frac{g_- \mu_m B J_z}{\hbar}$	g_-, μ_M, B, J_z	(0,0,0,0)	✓	✓	✓
II.35.18	$n = \frac{n_0}{\exp(\mu_m B/(k_b T))+\exp(-\mu_m B/(k_b T))}$	n_0, μ_m, B, k_b, T	(0,4,4,4,4)	✗	✗	✓

Table 11: This table lists equations from FSReD (Udrescu and Tegmark, 2020) along with their complexity c_1 and satisfiability of the three types of constraints (as defined in Section 7).

Number	Equation	Variables	c_1	$c_1(f)_n = 0$	$c_1(f)_n \leq 1$	$\lvert c_2(f)_n \rvert \leq 2$, $c_2(f)_{n,m} \leq 1$
II.35.21	$M = n_\rho \mu_m \tanh(\frac{\mu_m B}{k_b T})$	$n_{rho}, \mu_M, B, k_b, T$	(0,4,4,4,4)	✗	✗	✗
II.36.38	$f = \frac{\mu_m H}{k_b T} + \frac{\mu_m \alpha}{\epsilon c^2 k_b T} M$	$\mu_m, k_b, T, B,$ α, M, ϵ	(0,0,0,3,4,4,4)	✗	✗	✓
II.37.1	$E = \mu_m (1 + \chi) B$	μ_m, χ, B	(0,1,0)	✗	✓	✓
II.38.3	$F = \frac{YAx}{d}$	Y, A, x, d	(0,0,0,0)	✓	✓	✓
II.38.14	$\mu_S = \frac{Y}{2(1+\sigma)}$	Y, σ	(0,1)	✗	✓	✓
III.4.32	$n = \frac{1}{\exp(\frac{\hbar\omega}{k_b T})-1}$	ω, k_b, T	(3,3,3)	✗	✗	✓
III.4.33	$E_n = \frac{\hbar\omega}{\exp(\frac{\hbar\omega}{k_b T})-1}$	ω, k_b, T	(3,3,3)	✗	✗	✗
III.7.38	$\omega = \frac{2\mu_m B}{\hbar}$	μ_M, B	(0,0)	✓	✓	✓
III.8.54	$p_\gamma = \sin(\frac{E_n t}{\hbar})^2$	E, t	(2,2)	✗	✗	✓
III.9.52	$p_\gamma = \frac{p_d E_f t}{\hbar} \frac{\sin((\omega-\omega_0)t/2)^2}{(\omega-\omega_0)t/2)^2}$	$p_d, E_f, t, \omega, \omega_0$	(0,0,3,3,3)	✗	✗	✗
III.10.19	$E = \mu_m \sqrt{B_x^2 + B_y^2 + B_z^2}$	mu_M, B_x, B_y, B_z	(0,3,3,3)	✗	✗	✓
III.12.43	$L = n\hbar$	n	(0)	✓	✓	✓
III.13.18	$v = \frac{2Ed^2k}{\hbar}$	E, d, k	(0,0,0)	✓	✓	✓
III.14.14	$I = I_0(\exp(\frac{qV_e}{k_b T}) - 1)$	I_0, q, V_e, k_b, T	(0,4,4,4,4)	✗	✗	✓
III.15.12	$E = 2U(1 - \cos(kd))$	U, k, d	(0,2,2)	✗	✗	✓
III.15.14	$m = \frac{\hbar^2}{2Ed^2}$	E, d	(0,0)	✓	✓	✓
III.15.27	$k = \frac{2\pi\alpha}{nd}$	α, n, d	(0,0,0)	✓	✓	✓
III.17.37	$f = \beta(1 + \alpha\cos(\theta))$	β, α, θ	(0,2,2)	✗	✗	✓
III.19.51	$E = \frac{-mq^4}{2(4\pi\epsilon)^2\hbar)^2}\frac{1}{n^2}$	m, q, ϵ, n	(0,0,0,0)	✓	✓	✓
III.21.20	$j = \frac{-\rho_{c_0} q A_{vec}}{m}$	$\rho_{c_0}, q, A_{vec}, m$	(0,0,0,0)	✓	✓	✓