

A Neural Symbolic Model for Space Physics

Jie Ying^{1,†}, Haowei Lin^{2,†}, Chao Yue^{3,†}, Yajie Chen⁴, Chao Xiao⁵, Quanqi Shi⁵, Yitao Liang², Shing-Tung Yau^{6,7,*}, Yuan Zhou^{6,7,8,*}, and Jianzhu Ma^{9,10,*}

¹Qiuzhen College, Tsinghua University, Beijing, China.

²Institute for Artificial Intelligence, Peking University, Beijing, China.

³School of Earth and Space Sciences, Peking University, Beijing, China

⁴Max-Planck Institute for Solar System Research, Göttingen, Germany.

⁵Institute of Space Sciences, Shandong University, Weihai, China

⁶Yau Mathematical Sciences Center, Tsinghua University, Beijing, China.

⁷Beijing Institute of Mathematical Sciences and Applications, Beijing, China.

⁸Department of Mathematical Sciences, Tsinghua University, Beijing, China.

⁹Department of Electronic Engineering, Tsinghua University, Beijing, China.

¹⁰Institute for AI Industry Research, Tsinghua University, Beijing, China.

[†]Equal contribution.

*Correspondence should be addressed to: styau@tsinghua.edu.cn, yuan-zhou@tsinghua.edu.cn, majianzhu@tsinghua.edu.cn.

ABSTRACT

In this study, we unveil a new AI model, termed PhyE2E, to discover physical formulas through symbolic regression. PhyE2E simplifies symbolic regression by decomposing it into sub-problems using the second-order derivatives of an oracle neural network, and employs a transformer model to translate data into symbolic formulas in an end-to-end manner. The resulting formulas are refined through Monte-Carlo Tree Search (MCTS) and Genetic Programming (GP). We leverage a large language model (LLM) to synthesize extensive symbolic expressions resembling real physics, and train the model to recover these formulas directly from data. A comprehensive evaluation reveals that PhyE2E outperforms existing state-of-the-art approaches, delivering superior symbolic accuracy, precision in data fitting, and consistency in physical units. We deployed PhyE2E to five critical applications in space physics, including the prediction of sunspot numbers, solar rotational angular velocity, emission line contribution functions, near-Earth plasma pressure, and lunar-tide plasma signals. The physical formulas generated by AI demonstrate a high degree of accuracy in fitting the experimental data from satellites and astronomical telescopes. We have successfully upgraded the formula proposed by NASA in 1993 regarding solar activity, and for the first time, provided the explanations for the long cycle of solar activity in an explicit form. We also found that the decay of near-Earth plasma pressure is proportional to r^2 to Earth, where subsequent mathematical derivations are consistent with satellite data from another independent study. Moreover, we found physical formulas that can describe the relationships between emission lines in the extreme ultraviolet spectrum of the Sun, temperatures, electron densities, and magnetic fields. The formula obtained is consistent with the properties that physicists had previously hypothesized it should possess.

1 Introduction

The primary distinction between physics and other data sciences is the pursuit of the discovery of fundamental laws behind the world using concise symbolic formulas. The discovery of physical formulas is a lengthy process of trial and error. For instance, after about 40 attempts to match Mars data with various elliptical shapes, Kepler discovered that Mars' orbit was elliptical and proposed the three laws of Kepler. Similarly, through meticulous experimentation of electric and

magnetic phenomena, Faraday unveiled the laws of electromagnetic induction. He demonstrated that the intricate relationship between electricity and magnetism could be articulated through elegant, fundamental principles derived from empirical data.

Since AI has achieved tremendous success in multiple domains, a naturally arising question is whether we can leverage AI to automatically extract physical laws from experimental observation data to better understand our physical world. This task is known as symbolic regression (SR) within the AI field and has received widespread attention in recent years. Genetic algorithms (GAs) were first adopted to address SR problems by evolving a population of candidate symbolic expressions (typically represented as trees) to minimize a fitness function, which measures how well the expression fits the data^[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Monte Carlo Tree Search (MCTS) predicts the expression by exploring a search tree of possible expressions, simulating paths down the tree by randomly selecting mathematical operations to extend expressions. For each path, it generates complete expressions and assesses fitness based on how well they approximate the target function^[11, 12, 13, 14, 15]. Compared to GAs, MCTS offers more dynamic and fine-grained control over reward signals. Rewards can be customized to reflect the quality of partial solutions during tree exploration, facilitating a more efficient search for the optimal solution. Recent advances in deep neural networks have paved the way for the development of end-to-end methodologies. Unlike Genetic Algorithms (GAs) and Monte Carlo Tree Search (MCTS), which are symbolic search techniques that frequently struggle with the expansive search space, the end-to-end approach streamlines the process by eliminating the need for iterative searching and refinement. It predicts mathematical expressions as sequences of symbols (operators, variables, constants) from data, allowing symbolic expressions to be generated in a single forward pass through the neural network, which significantly improves speed, especially for large datasets and complex functions^[16, 17, 18, 19, 20, 21]. End-to-end approaches require large datasets for training, yet the number of physical formulas discovered by humanity remains relatively limited (fewer than 300). Therefore, current data-driven end-to-end algorithms are limited in reasoning simple mathematics equations and have not been applied to real physical data. Additionally, symbols in physical law have physical dimension systems, and the entire expression should ensure the correctness of these dimensions, which is not yet considered by current computational methods.

To address these limitations, we propose a new framework, named PhyE2E, to achieve accurate symbolic regression for space physics. The PhyE2E framework includes the following key components. First, we fine-tuned a large language model (LLM) with existing physical formulas, enabling it to generate a diverse array of formulas that align with the statistical distribution of physical formulas. By harnessing the common knowledge acquired by LLMs from the internet, the generative model efficiently learns the underlying distribution from a small set of seed formulas, thereby overcoming the challenge of data sparsity in the training process. Second, we trained an end-to-end formula regression model based on the transformer model, which directly converts data matrices into symbolic formulas encoded in reverse Polish representation. With the aid of the LLM, our end-to-end model is trained with a large volume of formulas that “look physical” and adhere to consistent dimensional systems. Third, to reduce the search complexity, we developed a formula-splitting technique that is able to group variables without nonlinear (or logarithmically nonlinear) relationships, producing a series of simpler sub-formulas for a nested and more simplified symbolic regression task. This technique uses an oracle neural network to fit the data and then analyzes its second-order derivatives to uncover relationships among the input variables. Finally, we leveraged the state-of-the-art GAs and MCTS methods to further refine the predicted formulas. To evaluate our performance, we conducted comparisons on both an LLM-synthesized dataset and another real-world physics dataset AI-Feynman against GA-based, LLM-based, and NN-based methods. Experimental results indicate

that our method outperforms all others in terms of data fitting accuracy, the correctness of the mathematical form of the formulas, and dimensional accuracy. We further applied our model to various important applications in space physics, including predicting sunspot numbers, plasma sheet pressure, solar rotational angular velocity, emission line contribution functions from the Sun, and lunar tidal signals in the plasma layer. Compared to formulas proposed by physicists, the formulas derived from PhyE2E exhibit better generalizability, a more concise mathematical form, more precise physical units, and more importantly, provide physically meaningful insights and explanations for instrumental observations.

2 Results

Overview of PhyE2E. PhyE2E comprises an end-to-end transformer-based physical model designed to take observed data points as input and predict both the operators and the physical units involved in a formula directly (Fig. 1). A set of 264,000 synthetic formulas were generated by an LLM (OpenLLaMA2-3B) that had been fine-tuned using real physics formulas from the Feynman Symbolic Regression Database (FSReD)^[22] (Fig. 1 top). We randomly selected 180,000 formulas from the synthetic dataset for training, reserving the remaining data for testing. The inference process of PhyE2E involves three stages: a variable interaction detection method to decompose the target formula into sub-formulas (Fig. 1 bottom left), end-to-end prediction of each sub-formula using the trained model (Fig. 1 middle), and GA and MCTS approaches to refine the predicted formulas (Fig. 1 bottom right). We first fit the data using a standard multi-layer neural network, and used its Hessian matrix to determine the nonlinear interactions between all pairs of variables. The core of our model leverages a transformer architecture, synthesizing formulas as a prefix sequence with attached physical units. This approach incorporates both observed data points and physical prior knowledge about the target formula. In the final stage, we utilized GA and MCTS approaches to minimize the root-mean-square error (RMSE) of the predicted formula. This was achieved by using a grammar pool of context-free rules that incorporates basic operators like exp, as well as the most promising sub-formulas, which were automatically constructed within the search tree by PhyE2E.

Performance on the synthetic and AI Feynman datasets. We divided the synthetic dataset containing 264,000 formulas into training, validation, and test sets with a ratio of 80%, 10%, and 10%. We ensure that all validation and test formulas are unseen during training by removing formulas that are either identical, or become completely equivalent to the formulas in the training set after simple mathematical transformations (Sec. 4.5 in Methods). We compared PhyE2E with 13 state-of-the-art symbolic regression baseline models, including 4 GA-based models (PySR^[23], GP-GOMEA^[5], Operon^[24], and GPLEarn^[6]), 3 LLM-based models (TPSR^[21], EndToEnd^[19], NeSymReS^[20]), and 6 NN-based models (AIFeynman^[22, 25], uDSR^[26], PhySR^[27], ParFam^[28], KAN^[29, 30], BSR^[31]). The technical details of running these models are provided in Supplementary Notes 3. We also included two variants of PhyE2E in our comparison, including versions without using the formula decomposition module (D&C) and the MCTS refinement module (MCTS). We evaluated the performance of all the models on 6 metrics^[32] including symbolic accuracy, average accuracy ($R^2 > 0.99$), unit accuracy, complexity, relative complexity to the ground truth formulas and elapsed times (Sec. 4.6 in Methods).

First, we evaluated whether the physical formulas synthesized by the LLM were consistent with the real physical formulas from the Feynman Symbolic Regression Database (FSReD). It can be observed that the formulas generated by the LLM closely match the distribution of real

physical formulas in terms of the number of variables, formula complexity, depth, and operator types measured by the Jensen-Shannon divergence (D_{JS}) (Sec. 4.6 in Methods, Fig. 2a). Then, we studied the symbolic accuracy of the formulas generated by the model, that is, whether the mathematical forms of the formulas correspond to the true formulas used to generate the data. PhyE2E(D&C+MCTS) exceeds the second-best model uDSR by 31.75%, outperforms the best GA-based model, PySR, by 37.08% and the best LLM-based model, TPSR, by 39.83% (Fig. 2b). The generated formulas by PhyE2E also demonstrate a more powerful ability to fit data compared to other methods. PhyE2E(D&C+MCTS) outperforms all the state-of-the-art approaches by at least 30.00% in terms of Avg.Acc. ($R^2 > 0.99$). Regarding the accuracy of (physics) units, we observe that PhyE2E achieves 99.27% of accuracy, leading all the other approaches. The performance drops to 93.30% by including the D&C and MCTS modules. This decline is due to the absence of unit constraints during the D&C and MCTS refinement stage. The most compatible baseline of unit accuracy is PhySR, which reaches a comparable 89.70% with a strictly unit constraint during its search process^[27].

The data-fitting capabilities of the formulas can be enhanced by increasing their complexity. According to Occam’s Razor, complex formulas tend to have weaker generalizations compared to simpler ones and lose their interpretability in a physical context. Therefore, in addition to studying the formulas’ ability to fit data, we also focus on the complexity of the formulas generated by different models. Our model produces formulas with lower model complexity compared to the formulas generated by other models. For 42.17% of the test formulas, their depth is less than 3, suggesting that they predominantly represent linear relationships. For these low complexity formulas, our model successfully recovers the mathematical form of 98.02% of them (Fig. 2d). The relative complexity to the ground truth formula of PhyE2E is 2, which is 31.86% better than the second-best model PySR. By introducing the D&C and MCTS modules, PhyE2E increases the complexity of formulas by 6.79%, while still maintaining a similar complexity with PySR. Only a handful of other baseline models demonstrate the capability to predict formulas of appropriate complexity (e.g., PySR, GP-GOMEA, AIFeynman, PhySR), primarily because of their constraints on complexity or the inclusion of physics units. Others either generate formulas with high complexity (complexity > 50) or formulas deviate far away from the target formula (relative complexity > 10), making it difficult to interpret in practice (Fig. 2b).

Next, we evaluated the performance of different models on the formulas collected from the Feynman Symbolic Regression Database (FSReD), referred to as the Feynman Dataset for brevity. The Feynman dataset contains 100 real-world physics formulas sampled from the seminal Feynman Lectures on Physics^[33, 34, 35] covering core physics topics like classical mechanics, electromagnetism, and quantum mechanics. Although our model was not directly trained on formulas from the Feynman dataset, our training dataset was generated by LLM based on formulas from the Feynman dataset. To inhibit data leakage, we removed the formulas in the training dataset that were either identical, or became completely equivalent after simple mathematical transformations compared to those in the Feynman dataset. A comprehensive list of the test formulas is provided in Supplementary Table S1.

First, we observe that all computational methods, including ours, show similar performance on the Feynman dataset and on our synthetic data, which demonstrates that the distribution of formulas generated by the LLM is essentially consistent with those from the Feynman dataset. In terms of symbolic accuracy, PhyE2E(D&C+MCTS) exceeds the best classical model, PySR, by 17.09%, the best LLM-based model, TPSR, by 21.77%, and the best NN-based model, uDSR, by 18.49%. Although PySR outperforms standard PhyE2E in numerical precision ($R^2 > 0.99$) with

an accuracy of 80.80%, it is still surpassed by PhyE2E with the D&C and MCTS modules. The complexity of the models generated by PhyE2E remains low. The relative complexity to the ground truth formula of PhyE2E is 2.98, 2.83% higher than the best model PySR. The relative complexity for PhyE2E is lower than 5.75 and the complexity is lower than 16.85, which is essentially the best among all the baseline models (Fig. 2c). To further investigate the relationship between performance and formula complexity, we calculated the symbolic accuracy as a function of the number of variables, complexity, and the number of unary and binary operations. We found that our method had a significant advantage over other methods for formulas with high complexity. When the complexity is larger than 20, our model outperformed the second-best methods 67.11% and 32.73% on the two datasets, respectively (Fig. 2d). We divided both datasets into three subsets of varying difficulty based on the similarity of mathematical formulas compared to those in training datasets (0.95-1.0 as easy, 0.80-0.95 as medium, 0.00-0.80 as hard), and then systematically calculated the symbolic accuracy of each method. We found that our method had a significant advantage on the medium and hard datasets (Sec. 4.5 in Methods, Fig. 2d).

Among all the modules, the divide-and-conquer (D&C) module plays a crucial role in simplifying the search space in our framework. Consider a formula $f = m\sqrt{B_1^2 + B_2^2 + B_3^2}$ from the Feynman dataset, our D&C module first determined that the three variables B_1 , B_2 and B_3 under the square root do not interact, which indicates that the operators between them can only be addition or subtraction (Supplementary Fig. S1a). Therefore, the original symbolic regression task decomposes into three parts $g_i = m\sqrt{B_i^2 + C_i}$ ($i = 1, 2, 3$), each of which is processed individually by the end-to-end module. The predicted formulas for g_1 , g_2 and g_3 are later aggregated into one complete formula (Supplementary Fig. S1a). The D&C module reduces the complexity of a formula by splitting a set of variables that have no nonlinear interactions locally within a certain formula. This also explains why we find that our method has a considerable advantage over other methods as the complexity of the formulas increases.

However, the risk associated with this methodology is that if the decomposition is incorrect, some of the segments will contain incorrect variables. Therefore, we carefully studied the performance of different D&C strategies. We implemented 4 different strategies, including Single Pattern (detects the interaction of additive and multiplicative), Multi-Pattern (identifies all interactions such as additive terms under sine functions), Fixed Threshold and Adaptive Threshold (Sec. 4.2 in Methods). We evaluated the performance of different strategies by assessing how many formulas could be correctly decomposed, partially correctly decomposed, and completely incorrectly decomposed on the test set of the synthetic dataset. We found that the Adaptive Threshold strategy provided a flexible approach for interaction detection, resulting in a substantial improvement in complete accuracy by 50.61%. The Multi-Pattern strategy facilitates diverse types of interactions and effectively reduces the proportion of absolutely wrong formulas from 6.50% to 5.03% (Supplementary Fig. S1b). Overall, adopting the Multi-Pattern and Adaptive Threshold strategies results in the highest number of accurate formula decompositions, which was also the strategy we used in our framework.

Performance of sunspot number prediction. Next, we applied our trained model to multiple applications in space physics. Our goal is to find formulas that are more accurate in prediction and simpler in mathematical form than existing physical formulas. We directly applied our trained model to these real-world applications instead of performing any fine-tuning operations. We started by studying the pattern of changes in the sunspot number (SSN) over time. The Sun serves as the primary source of energy for the entire Earth system. Predicting sunspot numbers is essential for forecasting space weather events that can impact both satellite operations and terrestrial

communication systems. These forecasts are also pivotal in climate studies, aiding in modeling the impact of solar variability on the Earth’s climate. The accurate prediction of sunspot activity is also crucial for managing the effects of geomagnetic storms on the technological infrastructure, which can lead to significant disruptions and damage. Although the 11-year solar cycle is determined through direct observations of sunspot numbers for the past four centuries, scientists want to know whether there is a longer cycle in solar activity which could influence the climate of Earth (Fig. 3a). Therefore, in this task, in addition to predicting the sunspot number, we also focus on how to derive the long-term cyclical variations of solar activity from the predicted physics formulas.

We first collected the SSN data from the Sunspot Index and Long-term Solar Observations (SILSO) over the last 400 years^[36]. The most widely adopted formula is the one proposed by Hathaway et al.^[37], which is still being used in recent studies to analyze data from the last 30 years^[38]. This formula modeled the sunspot numbers $R(t)$ using different sets of parameters for different cycles (Fig. 3b). Although it can accurately fit the data for each cycle, it cannot be generalized from one cycle to another, making it incapable of predicting future SSNs or revealing the longer cycle of solar activity. To summarize a symbolic formula for SSN, we selected the SSN data from year 1855 to 1976 which containing 11 cycles and 1,450 data points as input of our model. The main components of the denominator in our formula are a squared sine term and a squared cosine term, while the numerator consists of a squared sine term (Fig. 3c top). The Pearson correlation between the predicted SSN and the measured ones for the next four complete cycles (year 1976 to 2019) reaches 0.72. For the upcoming cycle (2019-present), PhyE2E predicts the peak value to be 177.40 and occurs on October 10, 2024 (Fig. 3c top). It is worth noting that no data were used to fine-tune or retrain our model. We did not use all the data to generate the formula because we needed to examine the generalizability of the formula obtained by the model. The generalizability of the model and the generalizability of the formulas predicted by our model should be evaluated separately. We tried generating formulas with different amounts of data, and the resulting formula forms remained largely consistent (Supplementary Tables S7, S8).

We compared the formula generated by PhyE2E with those generated by other SR models. Among all the models, only our model can be directly applied to the data for formula inference without any retraining or fine-tuning. Therefore, we retrained all the other models to be compared on data from year 1855 to 1976 and tested their performance on data after year 1976 to fairly compare with our model. We first examined the formulas generated by different SR models. We observed that, except for AIF and GP-GOMEA, all other methods produced formulas containing trigonometric functions capable of generating periodic outputs. Formulas from BSR, EndtoEnd, KAN, NeSymReS, PhySR, and TPSR all yield identical values for each cycle, which clearly contradicts our observations and common sense. The formulas generated by GPLEarn, Operon, ParFam, uDSR, and our method are capable of generating periodic outputs with different values for each cycle (Fig. 3c bottom). However, the four other models generated excessively complex formulas, making it impossible to parse their physical meaning or ascertain the long-term cyclical patterns of solar activity (Supplementary Tables S9, S10). Next, we quantified the performance of the formulas generated by these models on the test sets from year 1976 to 2019 using two metrics: 1) the correlation between the formula’s predictions and the measured data within each individual cycle, then averages these correlations across multiple cycles (avg-R); 2) the correlation across multi-cycles (multi-R). Our simpler formulas significantly outperformed these complex formulas for both metrics. Specifically, the avg-R and multi-R of our method outperform the second-best methods by at least 59.75% and 33.74%, respectively (Fig. 3d).

To further validate the accuracy of the formula we obtained, we focused on the SSN data before

the 1700s. Due to technological limitations, there are no SSN data directly observed from telescopes prior to 1749^[36]. Therefore, we collected solar modulation levels reconstructed from atmospheric ^{14}C concentrations from the annual rings of thousand-year-old trees as an approximation of the SSN measurements^[39]. We adopted the same smoothing strategy as reported in Brehm et al., 2021 and found that the Pearson correlation between solar modulation from tree rings and the SSN measured by the telescopes is 0.886 after the 1700s, which verifies their close relationship (Fig. 3f). Then, we compared the SSN generated by our formula and solar modulation data before the 1700s. Their Pearson correlation is 0.561 before 1300, 0.653 during 1300 to 1700 and 0.501 after 1700 (Fig. 3e), which further demonstrates the predictive capacity of our formulas on previously unseen data. Lastly, since there are three sine/cosine functions in our formula, it is natural for us to derive three cycles from these trigonometric functions. The shortest cycle is 10.91 years, which aligns with observations of solar activity widely accepted by the research community. The second longest cycle is 59.27 years, which coincides exactly with the 60-year cycle of the ancient Chinese astronomical calendar system of Heavenly Stems and Earthly Branches. The longest cycle is 204.93 years, which we speculate is the cycle of the solar system operating within a larger planetary system (Fig. 3e). Although this result requires further confirmation through additional astronomical observational data, it is the first conjecture directly derived from symbolic formulas. The constants of our formula are in Supplementary Table S2.

Performance of plasma pressure prediction. Plasma pressure is a macroscopic parameter that plays an important role in plasma dynamics and the generation of electric currents. Increasing plasma pressure gradients in the radial direction causes the stretching of magnetic field lines and enhances perpendicular currents flowing azimuthally. The azimuthal plasma pressure gradient generates field-aligned currents, resulting in the bending of magnetic field lines (Fig. 4a). Wang et al. proposed a formula that describes how equatorial plasma pressure varies with its position relative to Earth using Geotail and the NASA mission Time History of Events and Macroscale Interactions During Substorms (THEMIS) data. However, their formula involves exponential terms and 9 constants for night-side equatorial isotropic plasma pressure, making it difficult to derive meaningful physical interpretations^[40, 41](Fig. 4b). To derive a simpler formula, we divided the same equatorial plasma pressure data according to the azimuthal angle, using the data from the near-side of the Earth as input for our model and the data from the far-side to assess the performance of our formula. As we increased the number of input data points, feeding the PhyE2E model with progressively farther data from Earth, the average mean square error (MSE) decreased rapidly (Fig. 4c, Supplementary Table S11, S12). PhyE2E achieves an average MSE of 7.04×10^{-3} with only 10% of the data provided, demonstrating strong generalization capabilities with the small dataset as input. As more data was provided, the accuracy of our model continued to improve, eventually reaching 6.63×10^{-4} , resulting in more precise predictions for the far side of Earth regions (Fig. 4e). PhyE2E also outperforms all other baseline models in terms of fitting accuracy (MSE) and model complexity, delivering the most accurate and simplest prediction formula, while other models are unable to provide accurate predictions in certain areas. The formula by Wang et al. cannot accurately predict the plasma pressure for the far side of Earth regions, while the EndtoEnd method fails to provide accurate predictions for the near side of Earth regions (Figs. 4d). Note that this problem involves two variables, so there are two possible scenarios: one where r and θ can be decomposed into two sub-formulas, and another where decomposition is not possible. We generated one formula for each scenario, compared their MSE on the training dataset, and selected the formula with the lower MSE as the final prediction. In this case, the formula derived using the

decomposition method outperformed the alternative. The formula we predicted has a more concise mathematical form compared to the formula proposed by Wang et al. (Fig. 4b). It reveals to us that the decay of the near-Earth plasma pressure is proportional to the square of the distance r to the Earth's center, whereas in the formula proposed by Wang et al., the plasma pressure has an exponential relationship with r . More importantly, we can derive certain physical facts that align with the observational data from this new formula. Specifically, if the plasma pressure decays with the square of r and it is also known that the magnetic pressure decays with the sixth power of r . Then, according to the formula $\text{plasma beta} = \text{magnetic pressure (Pth)}/\text{plasma pressure (Pb)}$, we can infer that plasma beta increases with the fourth power of r , which can be confirmed by observational data from another independent study^[42]. Among the formulas obtained by the other methods, only the EndtoEnd approach^[19] produced a formula that is inversely proportional to the square of r . However, its mathematical form is more complex compared to our formula (Supplementary Table S11). The constants of our formula are in the Supplementary Table S3.

Performance of solar rotational angular velocity prediction. The Sun's magnetic field is generated by the plasma motion within its interior. Angular velocity of solar rotation varies at different latitudes, and the magnetic field lines are stretched and twisted (Fig. 4f). Differential rotation is a significant factor in the solar cycle for the prediction of magnetic fields and sunspots. Understanding solar differential rotation helps to improve the prediction of solar activities, which is important for predicting and mitigating the impact of space weather events on satellites and human activities in space. Differential rotation also provides key insights into the structure and dynamics of the solar interior by comparing rotation speeds at different latitudes to those predicted by comprehensive numerical models^[43, 44]. One of the most widely adopted formulas describing the relationship between the solar differential rotation and solar latitude was derived by Snodgrass et al.^[45]. In this work, they assumed that solar differential rotation was symmetric with respect to the equator. Such an assumption often fails especially during periods of high levels of solar activity. In addition, this formula was fitted by using the measurements at low latitudes of the Sun, but observations at high latitudes are still missing, which limits the suitability of the model near the solar poles. For this task, the challenge for other AI models is that the limited amount of data makes it impossible for them to train the model and predict the formulas.

PhyE2E derived a simpler and more accurate formula with a simple trigonometric term using the data from Snodgrass et al. in Magnetic Rotation of the Solar Photosphere^[45]. The largest difference between this formula and the one generated by PhyE2E lies in their difference in the trigonometric periodicity, leading to a steeper prediction for polar regions, rather than a flat one (Fig. 4g). We further reduced the number of training data points and found that PhyE2E could predict the same formula with only 14 data points as input, rapidly achieving an MSE of 1.31×10^{-4} , which outperforms other models (Fig. 4g, Supplementary Table S13, S14). In contrast to the oscillations seen in other models, PhyE2E exhibits exceptional consistency and robustness, providing stable predictions with varying amounts of input data. The constants of our model can be found at the Supplementary Table S4. The predictions for all the baseline models are quite similar in non-polar regions, but they start to diverge in the polar regions (Fig. 4h). Regarding predictions at high latitudes, Hotta et al.^[43] overcame the “convective conundrum” through the supercomputer Fugaku and successfully reproduced solar-like differential rotation. We selected the simulation data from the north and south polar regions and compared them with the baseline models. The formula derived from PhyE2E performed the best in both polar regions, with correlations of 0.9814 and 0.9740, outperforming all baseline models including the formula proposed by Snodgrass et al.^[45]. In

addition, we applied our model to different heights in the solar atmosphere, using data from various spectral lines^[46] in the photosphere (Si I and Fe I) and the chromosphere ($H\alpha$) (Fig. 4i). Similar formulas are derived across all the spectral lines with remarkable consistency, suggesting that the differential rotation speed within the Sun follows a regular and predictable pattern (Fig. 4j).

Performance of contribution function of emission lines. Emission lines in the extreme ultraviolet spectrum of the Sun, such as Fe X lines, are often used to observe the solar corona (Hinode/EIS, Solar Orbiter/EUI) (Fig. 5a). Predicting the contribution functions of these lines helps in plasma diagnostics such as temperatures, densities, and magnetic fields, which is essential to understand solar phenomena such as flares and coronal mass ejections. The EUV emission lines can also be formed in other types of astrophysical targets, including stellar coronae, galactic nuclei, and supernovae. Understanding the formation of the emission lines can provide insight into the physical conditions and processes of these targets. Given its role as a critical component of fundamental atomic databases applicable to a wide range of studies, considerable efforts have been made to calculate the contribution functions of emission lines (CHIANTI^[47, 48] and NIST atomic database^[49]). The contribution functions could be approximated by solving complex quantum mechanical equations involving detailed calculations of electron transitions, collisions, and radiative transfer, which is usually a computationally expensive process. Therefore, a challenge in physics is whether we can accurately estimate the contribution function using easily observable data, including the temperature and electron density around the Sun. Currently, there is no physical formula that accurately reveals how the temperature and electron density around the Sun influence the contribution levels of the emitted spectral lines.

To address this problem, we downloaded the Fe X 174 and Fe X 175 line data from the CHIANTI database^[47, 48], and uniformly sampled 2,500 data points in an electron density range of $10^8 - 10^{10}$ and a temperature range of $5 \times 10^5 - 5 \times 10^6$ °C. Data were segmented into low and high temperature regions using a cutoff of 2.8×10^6 °C. Instead of fine-tuning or retraining our model, we took the data from the low-temperature region as input to generate the formula and test the prediction performance of the formula in the high-temperature region. In the high-temperature region, PhyE2E achieved a significantly lower MSE, with the magnitude of the MSE being two orders of magnitude smaller than the other baseline models (Fig. 5b left). Both the Fe X 174 and 175 emission lines are highly temperature-dependent, with a relatively smaller influence from the electron density. To address this issue, we further investigated the ratio of these two lines, which serves as a powerful diagnostic tool for probing the effects of electron density on the intensity. Compared to the prediction of the two spectral lines individually, the prediction of the ratio of the two lines carries more physical significance and is also more challenging. Accurate predictions of the individual spectral lines do not guarantee that their ratio can be predicted accurately (Figs. 5b,c,d). In this task, our formula achieves an MSE of 3.10×10^{-3} , which is three orders of magnitude lower than the second best method EndtoEnd (Fig. 5b, middle).

Next, we examined the physical significance of the formulas generated by different methods. First, the complexity of our formula is not the lowest, but it is the only formula that has the correct physical units among all the baseline methods (Figs. 5b right, Supplementary Table S15, S16). In the solar corona, the processes of ionization and recombination, as well as collision and excitation, can be considered decoupled^[50]. In our formula, the electron density and temperature also exist in a decoupled form. The dependence of the electron density following the mathematical form $(n + c_1)/(n + c_2)$ is widely accepted by the space physics community^[51]. It captures the behavior where the intensity increases with electron density at low values but saturates at higher

densities because of collisional de-excitation. The temperature term of our formula is composed of the combination of a power-law term and an exponential term. The power-law term dominates at low temperatures, capturing the increase in intensity as more electrons gain sufficient energy to excite the ions; the exponential decay term dominates at high temperatures, reflecting the rapid decrease in intensity due to ionization to higher states. This combination of a power-law term and an exponential term was also adopted by Raymond et al.^[52] The constants of our formula are in the Supplementary Table S5. For the formulas generated by other methods, some have excessive complexity, making them difficult to interpret, while others have incorrect dimensions or are overly simplistic. For instance, the formula produced by PhySR is simple, but does not include the electron density term (Supplementary Table S15).

Performance of lunar tide signal of plasma layer. The Earth’s magnetospheric electric fields, including corotation and convection electric fields, are crucial for understanding the behavior of charged particles and maintaining the stability of the magnetosphere (Fig. 5e). These fields are responsible for the movement and energization of charged particles, which in turn affect space weather and the interaction between the solar wind and Earth’s magnetic field. Prior to 2023, it was commonly accepted in the scientific community that the electric fields at a specific near-Earth location were solely influenced by the Earth’s position relative to the Sun and the distance to the Earth’s center. A recent work indicated that due to the effects of lunar tidal forces, the electric fields were also related to the Earth’s position relative to the Moon^[53]. However, due to the complexity of the problem, physicists have been unable to provide a physical formula that links these three important factors: electric fields with the distance to the Earth’s center, the relative positions of the Earth and the Moon, and the Earth’s position relative to the Sun, which can be represented as Lunar Local Time (LLT), Magnetic Local Time (MLT) and L-shell, respectively.

To address this problem, we collected $\sim 20,000,000$ data measured by the Van Allen Probes satellites between L values of 3–6 from January 2013 to May 2019 from the RBSP/EFW official website¹, and used PhyE2E to generate a formula to predict Radial Electric Field, denoted as E_r , from LLT, MLT and L-shell values. Due to the large volume and high redundancy of data, we divided the entire three-dimensional space near the Earth into $50 \times 50 \times 50$ grids, and then calculated the average value of the Radial Electric Field within each grid. We randomly sampled 80% of the grids as input for our model and also used them as training data for other baseline models. The remaining data were adopted as test data to evaluate the performance of all models. Based on the adaptive threshold for the decomposition of the formula, we found that there are no coupling relationships among these three variables. Therefore, we decomposed the original symbolic regression problem into three sub-problems, each containing only one variable. Then, we predicted each uni-variate function using the end-to-end model (Sec. 4.3 in Methods). Without fine-tuning or re-training of the model, the formula generated by our model has an MSE lower than the second-best method by 53.37%, with complexity reduced by 75.91% (Fig. 5f). We examined the effects of LLT and MLT on the Radial Electric Field separately and found that our formula provides a good approximation for the original data. The prediction of our formula is much smoother than the measured data, due to the data smoothing applied within each grid (Fig. 5g). Compared to the formulas generated by other methods, the formula produced by our model captures multiple physical principles, making it more physically meaningful. First, among all the models, only our model provides an asymmetric prediction between the dayside and nightside of the Earth, suggesting that the radial electric field (E_r) on the dayside decays more rapidly in the radial direction (L-shell), while on the nightside, the

¹<http://www.space.umn.edu/rbspew-data/>

radial electric field(E_r) decays faster in the non-radial direction (MLT). Second, since the radial electric field (positive direction towards Earth) is derived from the calculation of the electric field's y-component, it exhibits periodic variation with Magnetic Local Time (MLT), and the period is 12 hours^[54], which is consistent with the periodicity of the cosine function of MLT in our formula (12.13 hours). Third, our formula indicates that E_r decays with the square of the L-shell, which is consistent with theoretical calculations. According to the ideal magneto-hydrodynamic (MHD), the corotational electric field E could be derived as $E = -\Omega_E B_0 / L_{shell}^2$, where Ω_E and B_0 are Earth's rotational angular velocity and Earth's surface magnetic field, respectively. The constants of our model could be found in Supplementary Table S6. The complexity of our formula is not the lowest among all the models because the pattern of this physics application is complicated (Fig. 5b, Supplementary Table S17, S18). Among methods with lower complexity, only the formulas from BSR and PySR exhibit periodicity in LLT or MLT, and only PySR captures the inverse relationship between E_r and L-shell. However, the BSR formula does not include the crucial L-shell variable, and the PySR formula lacks the LLT variable (Supplementary Table S17).

3 Discussion

Existing symbolic regression research primarily employs search methods based on Monte Carlo Tree Search (MCTS) and Reinforcement Learning (RL). These methods often struggle to accurately predict formulas with a large number of variables or complex operational relationships between variables. To discover the correct formulas within a limited time, most of the MCTS approaches require prior knowledge to achieve an initialization close to the true solution. In contrast, our method can decompose formulas without knowing their specific forms, significantly reducing the complexity of symbolic regression. For each decomposed sub-problem, we utilized an end-to-end approach, tokenizing the data and directly translating it into formula strings using a transformer. Among all SR methods, our approach offers a ready-to-use model and is the only one that does not require retraining or fine-tuning on physical data.

One characteristic of space physics is that there is no data noise on the planet scales or on the microscopic particle scale. However, if the model is to be generalized to other areas of physics, such as condensed matter physics and fluid dynamics, the effects of noise inherent in the data must be considered. Since the data is free from noise, the model must learn to deduce the operational relationships between physical variables based solely on the data provided. Therefore, we can still leverage large language models using this method of simulating the generation of physical formulas for data augmentation. This principle is similar to that of AlphaZero, which does not require human game records but can learn to play Go through AI-versus-AI games. This is because AI only needs to learn optimal strategies in various complex situations, rather than necessarily mimicking human players' thought processes and habits. Currently, our current model cannot handle operations such as integration and differentiation, which means that a significant portion of physics formulas based on partial differential equations cannot be resolved. We believe that the data augmentation and formula decomposition techniques are still applicable in partial differential equations.

4 Methods

We now detail the components of our system, starting with the generative model for synthetic physics formulas. Next, we present the core framework, which includes an end-to-end symbolic regression model integrated with a Divide-and-Conquer (D&C) strategy for decomposing complex formulas into simpler sub-formulas. We then describe the process to refine the formula predicted by

the end-to-end model, encompassing Monte Carlo Tree Search (MCTS) and Genetic Programming (GP) refinement. Finally, we discuss the details of how to construct test data and evaluation metrics.

4.1 Generative model for synthetic physics formulas

To generate synthetic formulas resembling real physical formulas, we fine-tuned the pre-trained OpenLLAMA-2-3B language model^[55] using the AI Feynman dataset^[22], which is a benchmark collection of mathematical formulas representing real-world physical laws and relationships, consisting of 100 formulas. A two-stage fine-tuning strategy was devised to address the challenge of limited training data and to integrate prior knowledge of physical dimensional systems into the training process. In the first stage, the OpenLLAMA-2-3B model was fine-tuned on the AI Feynman training set. The fine-tuned model was then employed to generate 50,000 synthetic formulas. These formulas were evaluated for consistency with physical dimensional systems, resulting in approximately 8,000 formulas that adhered to dimensional consistency. The second stage involved reassigning weights to the 8,000 dimensionally consistent formulas generated in the first stage. This weighting was designed to ensure that the statistical distribution of the synthetic formulas, such as the number of variables, formula depth, and operator frequency, aligned with those of the real physical formulas in the AI Feynman dataset. Finally, the language model was further fine-tuned using the weighted distribution of the 8,000 formulas.

Specifically, the fine-tuning was performed using the DeepSpeed ZeRO Stage 2 optimizer within the HuggingFace Transformer framework^[56], with a learning rate of 3×10^{-5} . The training prompt followed the format: “Generate a physical formula: {formula}”. Mathematical formulas in the prompt were represented in plain text using their natural mathematical forms. To generate synthetic formulas, the model was prompted with the same instruction. The hyperparameters of the generative model were configured to balance diversity and quality in the generated formulas: the temperature was set to 2.0, efficient sampling was enabled (`do_sample = True`), and the maximum length of the generated sequences was restricted to 64 tokens. To evaluate the consistency of a formula with physical dimensional systems (in the first stage), the formula’s expression tree^[57] was constructed, and the dimensions of each subtree were computed in a bottom-up manner, starting from the leaves and moving toward the root. During this process, the following checks were performed: (1) for any sub-tree in the form of $A + B$ or $A - B$, the dimensions of A and B were required to be the same, (2) for any sub-tree in the form of $\sin(A)$, $\cos(A)$, or $\exp(A)$, the dimension of A was required to be null. To reassign weights to a set of formulas (in the second stage), a linear program (LP) was formulated. The LP variables represent the weights assigned to each formula, subject to the constraints that the weights must be non-negative and sum up to 1. The objective was to minimize the total variation distances between the statistical distributions (e.g., number of variables, formula depth, operator frequency) of the weighted synthetic formulas and those of the AI Feynman dataset. These total variation distances were expressed as linear combinations of the LP variables. Additionally, a regularization term was included in the LP objective to ensure that the weighted distribution did not deviate excessively from the original uniform distribution.

4.2 The divide-and-conquer strategy

Many physical formulas exhibit intrinsic simplicity and symmetry, with variables often interconnected through straightforward addition or multiplication. Inspired by this observation, we proposed a divide-and-conquer strategy to decompose the target formula into a summation (or multiplication)

of simpler sub-formulas by estimating inter-variable relationships. These sub-formulas are then individually predicted and subsequently integrated to reconstruct the target formula. Below, we first introduce the inner-variable relationships, followed by the decomposition strategy for sub-formulas and the resampling technique for data points. Finally, we present the aggregation theorem for the back aggregation step.

4.2.1 The oracle neural network and estimation of inter-variable relationships

Given the data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with N evaluations of the target formula $f(\mathbf{x})$ ($N = 200$ in our experiments), an oracle neural network $\tilde{f}_\theta(\mathbf{x})$ was first trained to approximate $f(\mathbf{x})$ at any input point $\mathbf{x}$. Here, θ are the parameters of the oracle neural network, which consists of 5 hidden layers. The first 3 layers each contained 128 tanh neurons, while the last 2 layers each contained 64 tanh neurons. The network was trained for 400 epochs, with an initial learning rate of 0.1 that decayed tenfold every 100 epochs. The following definition characterizes the inter-variable relationship to be estimated for decomposing the target formula.

Definition 1. Let $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be a uni-variate operator. Two features $i, j \in \{1, 2, \dots, n\}$ of a target formula $f : \mathbb{R}^n \rightarrow \mathbb{R}$ are said to be σ -separable if there exist sub-formulas f_1 and f_2 such that f can be expressed as:

$$f(\mathbf{x}) \equiv \sigma(f_1(\mathbf{x}_{-i}) + f_2(\mathbf{x}_{-j})),$$

where $\mathbf{x}_{-i}$ is the $(n - 1)$ -dimensional vector obtained by removing x_i from $\mathbf{x}$, and $\mathbf{x}_{-j}$ is defined analogously.

When the operator σ is invertible, the following lemma, whose proof is provided in Supplementary Sec. 5.1, provides an equivalent condition to check whether two features are σ -separable in a twice-differentiable formula.

Lemma 1. Let the uni-variate operator $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ and the target formula $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be twice differentiable. Suppose σ is strictly monotonic, then two features $i, j \in \{1, 2, \dots, n\}$ are σ -separable if and only if for all $\mathbf{x} \in \mathbb{R}^n$,

$$\frac{\partial^2 \sigma^{-1} \circ f}{\partial x_i \partial x_j}(\mathbf{x}) = (\sigma^{-1})''(f(\mathbf{x})) \cdot \frac{\partial f}{\partial x_i}(\mathbf{x}) \cdot \frac{\partial f}{\partial x_j}(\mathbf{x}) + (\sigma^{-1})'(f(\mathbf{x})) \cdot \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) = 0. \quad (1)$$

Practical implementation. In our experiment, we tried the uni-variate operators $\sigma \in \{\text{id}, \text{sqrt}, \text{inv}, \text{arcsin}, \text{arccos}, \text{log}, \text{sqrt} \circ \text{log}, \text{inv} \circ \text{log}, \text{arcsin} \circ \text{log}, \text{arccos} \circ \text{log}\}$ to perform the divide-and-conquer strategy and predict the target formula via the end-to-end model (Sec. 4.3). Note that the operators that involve logarithm effectively separate the target formula into the multiplication of two sub-formulas according to Definition 1. The prediction based on different uni-variate operators were collected and fed into the MCTS and GP module for further refinement (Sec. 4.4).

However, even with access to the oracle neural network $\tilde{f}_\theta$, we cannot directly verify the condition in Lemma 1 because it required evaluating all the points in $\mathbf{x} \in \mathbb{R}^n$ and the second-order derivative of the approximate function $\tilde{f}_\theta(\mathbf{x})$ tends to be noisy. In our algorithm, this condition was verified approximately by sampling a subset of points and employing a majority rule to mitigate the effects of approximation noise. Specifically, for the set of training data points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ($N = 200$), two features i and j are determined to be σ -separable if, for a threshold parameter $\epsilon > 0$, it holds that

$$J_{i,j}(\sigma, \tilde{f}_\theta) \stackrel{\text{def}}{=} \text{median}_{1 \leq k \leq N} \left\{ \left| \frac{\partial^2 \sigma^{-1} \circ \tilde{f}_\theta}{\partial x_i \partial x_j}(\mathbf{x}_k) \right| \right\} \leq \epsilon. \quad (2)$$

The choice of the threshold parameter ϵ is crucial to the accurate identification of σ -separable pairs. We first employ the *fixed thresholding* strategy, treating $\epsilon = \epsilon(\sigma)$ as a constant for each uni-variate operator σ . This fixed constant is determined using a data-driven approach for each σ . More specifically, we randomly sampled 1,000 target formulas h_k and trained the corresponding oracle neural networks $\tilde{h}_k(\mathbf{x}_k|\theta_k)$. For each feature pair (i, j) , we identified their σ -separability by Lemma 1 and calculated the corresponding $J_{i,j}(\sigma, \tilde{h}_k)$. The threshold was then chosen to maximize the number of feature pairs among the 1,000 additional target formulas that were classified correctly in terms of σ -separability.

On the other hand, a single constant value for a fixed ϵ may not work well for all target formulas f , as different functions can exhibit vastly different scales of derivatives. This scale can even vary if f is multiplied by a large constant factor. To address this issue, the technique of *adaptive thresholding* was also proposed to automatically select the threshold based on the derivative scale, thereby improving the numerical stability of the estimation of σ -separable pairs. More specifically, let $\epsilon_0 = \min_{1 \leq i < j \leq n} J_{i,j}(\sigma, \tilde{f}_\theta)$. We then define $\epsilon_1 = \alpha_{\min} \epsilon_0$ and $\epsilon_2 = \alpha_{\max} \epsilon_0$ as the minimum and maximum thresholds, respectively. In our algorithm, we set $\alpha_{\min} = 2$ and $\alpha_{\max} = 10$. For each $\epsilon \in [\alpha_{\min}, \alpha_{\max}]$, the set of σ -separable pairs (hereafter referred to as the σ -separable set) was estimated as

$$Q_\epsilon(\sigma, \tilde{f}_\theta) \stackrel{\text{def}}{=} \left\{ (i, j) \mid 1 \leq i < j \leq n, J_{i,j}(\sigma, \tilde{f}_\theta) \leq \epsilon \right\}. \quad (3)$$

Finally, the class of σ -separable sets was defined as

$$\mathcal{Q} \stackrel{\text{def}}{=} \{Q_\epsilon(\sigma, \tilde{f}_\theta) \mid \epsilon \in [\alpha_{\min}, \alpha_{\max}]\}. \quad (4)$$

It is straightforward to verify that $|\mathcal{Q}| \leq n(n-1)/2$. Therefore, applying the divide-and-conquer strategy based on each $Q \in \mathcal{Q}$ is computationally feasible. Each application of this strategy derived a prediction of the target formula. These formulas were evaluated and the best one was selected. In the following subsections, we will explain how the divide-and-conquer strategy operates for any estimated σ -separable set Q : it begins by inducing a set of feature sets, followed by predicting the sub-formulas for these feature sets, and ultimately integrates them into the final prediction of the target formula.

4.2.2 The division of the target formula

We have discussed the σ -separable relationship between pairs of features. The following definition about the global division of the target formula and features is crucial to our divide-and-conquer strategy.

Definition 2. Let $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be a uni-variate operator. For any feature subset A_i of $A = \{1, 2, \dots, n\}$, denote by $\mathbf{x}_{A_i}$ the vector obtained by restricting $\mathbf{x}$ to A_i , i.e., if $A_i = \{j_1, j_2, \dots, j_k\}$, then $\mathbf{x}_{A_i} = (x_{j_1}, x_{j_2}, \dots, x_{j_k})$. A class of subsets $\{A_i\}_{i=1}^m$ is said to be a σ -division of f if none of the subsets is contained in another subset and there exist m sub-formulas $f_1, f_2, \dots, f_m$ such that f can be expressed as:

$$f(\mathbf{x}) \equiv \sigma(f_1(\mathbf{x}_{A_1}) + f_2(\mathbf{x}_{A_2}) + \dots + f_m(\mathbf{x}_{A_m})), \quad (5)$$

where each f_i is a function of $\mathbf{x}_{A_i}$.

It is quite straightforward to derive a σ -division from σ -separable pairs, detailed as follows. First, the σ -separable relationships between all features are identified as described in Sec. 4.2.1.

Then, the algorithm starts with an initial (and trivial) σ -division $\mathcal{A} = \{\{1, 2, \dots, n\}\}$ and iteratively refines it. At each iteration, the algorithm selects a σ -separable pair of features j_1 and j_2 that has not been considered. For each feature subset $A \in \mathcal{A}$ such that $\{j_1, j_2\} \subseteq A$, $\mathcal{A}$ is updated as follows:

$$\mathcal{A} \leftarrow (\mathcal{A} - \{A\}) \cup \{A - \{j_1\}, A - \{j_2\}\}.$$

This process is done after all σ -separable feature pairs have been considered. By the following Lemma 2, whose proof can be found in Supplementary Sec. 5.1, this iterative procedure guarantees to yield a valid σ -division.

Lemma 2. *Let uni-variate operator $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be strictly monotonic and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be the target formula that is twice differentiable. Suppose we have accurately obtained the set of all σ -separable feature pairs of f . Then, for each iteration number $\ell \geq 1$, the division obtained by the above algorithm after the ℓ -th iteration, denoted by $\{A_k^\ell\}_{k=1}^{m_\ell}$, is a σ -division of f .*

Algorithm 1 is provided in Supplementary Sec. 4 to formally describe the above procedure to derive σ -separable feature pairs via the fixed and adaptive thresholding techniques, as well as to construct a σ -division based on the σ -separable pairs.

4.2.3 Evaluation of surrogate sub-formulas and back aggregation for the target formula

Once a σ -division $\{A_i\}_{i=1}^m$ is derived, it would be most natural to recursively perform symbolic regression to predict the sub-formulas $\{f_i\}$ and reconstruct the target formula f according to Eq. (5). However, there are two challenges to this approach. First, we do not have evaluation data ($\{(\mathbf{x}, y = f_i(\mathbf{x}))\}$) for each sub-formula f_i . Moreover, the sub-formula f_i are not even unique. For example, if $f(x_1, x_2, x_3) = \sigma(f_1(x_1, x_2) + f_2(x_1, x_3))$, then it can also be expressed as $f(x_1, x_2, x_3) = \sigma((f_1(x_1, x_2) - x_1) + (f_2(x_1, x_3) + x_1))$, where $f'_1 = f_1 - x_1$ and $f'_2 = f_2 + x_1$ are also a set of valid sub-formulas for the σ -division $\{\{x_1, x_2\}, \{x_1, x_3\}\}$.

To address the above challenges, we turn to predict the *surrogate sub-formulas* $\{g_i\}_{i=1}^m$, defined as follows. First, an arbitrary $\mathbf{z} \in \mathbb{R}^n$ is chosen, where in the experiment, $\mathbf{z}$ was sampled from the standard multivariate Gaussian distribution. Then, given the σ -division $\{A_i\}_{i=1}^m$, for each $i \in \{1, 2, \dots, m\}$, we define $g_i : \mathbb{R}^{A_i} \rightarrow \mathbb{R}$ as

$$g_i(\mathbf{x}_{A_i}) = f(\mathbf{x}_{A_i}, \mathbf{x}_{\overline{A_i}} = \mathbf{z}_{\overline{A_i}}), \quad (6)$$

where $\overline{A_i} = [n] - A_i$.

Each surrogate sub-formula g_i might be quite different from the corresponding sub-formula f_i . For instance, consider $f(x_1, x_2, x_3) = f_1(x_1, x_3) + f_2(x_2, x_3)$ where $f_1(x_1, x_3) = \sin(x_1 x_3)$ and $f_2(x_2, x_3) = \cos(x_2 x_3)$. According to the definition, we have $g_1(x_1, x_3) = \sin(x_1 x_3) + \cos(c_2 x_3)$, and $g_2(x_2, x_3) = \cos(x_2 x_3) + \sin(c_1 x_3)$. Note that each g_i not only contains f_i , but also introduces extra non-trivial terms. The following theorem, the proof of which can be found in Supplementary Sec. 5.2, provides a way to eliminate the extra terms and aggregate the surrogate sub-formulas $\{g_i\}_{i=1}^m$ to reconstruct the target formula f .

Theorem 3. *Suppose the uni-variate operator $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is strictly monotonic. Let $\mathcal{A} = \{A_i\}_{i=1}^m$ be a σ -division of the target formula f and $\{g_i\}_{i=1}^m$ be defined as in Eq. (6) based on a vector $\mathbf{z}$. For each $I \subseteq [m]$, denote*

$$\mathcal{A}_I = \cap_{i \in I} A_i.$$

Then, it holds that

$$f(\mathbf{x}) = \sigma \left(\sum_{\emptyset \neq I \subseteq [m]} \frac{(-1)^{|I|-1}}{|I|} \sum_{i \in I} \sigma^{-1} \circ g_i(\mathbf{x}_{\mathcal{A}_I}, \mathbf{x}_{\mathcal{A}_i - \mathcal{A}_I} = \mathbf{z}_{\mathcal{A}_i - \mathcal{A}_I}) \right). \quad (7)$$

Practical implementation. For the i -th surrogate sub-formula g_i , the data $D^{(i)} = \{\mathbf{x}_k^{(i)}, y_k^{(i)}\}_{k=1}^N$ were constructed as follows:

$$\mathbf{x}_k^{(i)} = (\mathbf{x}_k)_{\mathcal{A}_i}, \quad y_k^{(i)} = \tilde{f}_\theta(\mathbf{x}_{\mathcal{A}_i} = \mathbf{x}_k^{(i)}, \mathbf{x}_{\overline{\mathcal{A}_i}} = \mathbf{c}_{\overline{\mathcal{A}_i}}),$$

where $\mathbf{x}_k$ is the k -th data point in the original data D . Then, each g_i were predicted by the end-to-end model (as will be described in Sec. 4.3). Finally, the target formula f was predicted by aggregating the surrogate sub-formulas according to Eq. (7).

4.3 The end-to-end model

Architecture. Kamienny et al.^[19] established a transformer-based end-to-end model for symbolic regression. We designed a similar transformer that includes 4 encoder layers and 16 decoder layers, forming an asymmetric architecture^[58]. Each layer consists of 16 attention heads and an embedding dimension of 512. We removed positional embeddings to ensure permutation invariance of the input data.

Physical priors (e.g., physics units) play an important role in physical formulas^[27, 59]. We integrated physical hints, including the physical units for both input and output variables, formula complexity, possible operators and possible constants, into the input of our model for predictions. Note that, the input of our model is a set containing the evaluations at N data points and h possible physics hints. Each data point was encoded into $(n + 1) \times 3$ tokens, where each of the n features and the evaluation was encoded into 3 tokens, according to its sign, mantissa (between 0 and 9999) and exponent (from E-100 to E+100)^[58]. Each of the h physical hints was encoded into specialized tokens. These encoded token strings were padded with the <pad> token so that each data point and each hint span exactly $L = 33$ tokens. They are finally concatenated together to form the input of our model.

As for the output of our model, we additionally incorporated the physical units of each operator within the formulas as output sequence, making it compatible for the model to automatically infer dimensions of the target formula. While a formula can be encoded as a prefix expression consisting of operators and variables, we enhance it by associating physical units to each operator, providing intrinsic insights into the structure of physics formulas. Specifically, our model generates a sequence of (operator/variable, physical unit) as output. We utilize five of the most commonly used physical units as our bases, slightly different from the International System of Units (SI), including the Meter (m), Second (s), Kilogram (kg), Kelvin (K), and Volt (V)^[22]. In this way, a physical unit can be encoded into five tokens. For dimensionless formulas, we simply assign each base unit with value 0.

Training details. The training data consisted of the data evaluated by 200,000 synthetic physical formulas generated in Sec. 4.1. Each formula was evaluated at $N = 200$ points sampled from the standard multivariate Gaussian distribution, conditioned on the fact that the formula is properly defined at the point, which follows the previous work^[19].

We used the cross-entropy loss on the next-token prediction with the Adam optimizer, starting with a learning rate of 10^{-7} , gradually increased to 2×10^{-4} during the first 10,000 steps, and

subsequently decayed proportional to the inverse square root of the step count^[60]. A validation set comprising 20,000 synthetic formulas is held out, and our model is trained for 100 epochs, processing 500 million formulas in total, until the validation accuracy stabilizes. Each batch consists of 10,000 tokens, with formulas grouped by similar lengths to minimize padding overhead. The training was performed on a single NVIDIA A100 GPU with 80GB of memory over about one day.

Constant optimization. The end-to-end model only returned a “formula skeleton” where the constant parameters in the formula remained unoptimized. We adopted the BFGS algorithm^[61] to further refine the constants in the formula skeleton. Specifically, given a formula f , we denote by $\mathbf{c}$ the constant parameters in the formula. Based on the data $\{(\mathbf{x}_k, y_k)\}_{k=1}^N$, the BFGS algorithm was invoked to find

$$\arg \min_{\mathbf{c}} \sum_{k=1}^N [f(\mathbf{x}_k; \mathbf{c}) - y_k]^2.$$

4.4 MCTS and GP refinement

MCTS. The standard Monte Carlo Tree Search (MCTS) process consists of four steps:^[15] 1) selection, where the best candidate formulas are identified based on their performance; 2) expansion, where new symbolic formulas (which may be incomplete) are generated by adding one more operator from the incomplete formula; 3) simulation, where the incomplete formulas are randomly simulated and then evaluated based on their predictive accuracy; and 4) back-propagation, where the results are propagated back to update rewards and visit times of each MCTS node. We employed MCTS to refine the target formula obtained by the end-to-end model. Specifically, random sub-trees were removed from the expression tree of the target formula, and MCTS was invoked with the resulting incomplete formula to search for a better target formula. Moreover, following the work of Sun et al.^[15], we constructed a grammar pool to reduce the search space of MCTS. Specifically, this grammar pool consisted of basic operators (+, −, ×, /, sin(x), cos(x), exp(x)) and the operators extracted from the predicted results of the end-to-end model. During the expansion steps, MCTS was restricted to select operators only from the grammar pool to construct a complete symbolic formula.

Genetic programming. Our genetic programming (GP) refinement module followed the work of PySR^[23], where we used the results from the end-to-end model and MCTS as the initial populations for the GP algorithm. During each round of evolution, each formula in the population undergoes random mutations, including appending, changing, or deleting specific operators in the formula. Crossover operations were also performed between individuals to combine their expressions and create new formulas. The entire GP algorithm optimizes the formulas for 40 rounds. The optimization process stops early if the MSE of one of the generated formulas achieves 10^{-6} during the process. Finally, a Pareto front of the formulas was generated. The best formula from this set was selected based on the criterion of $\text{MSE} \times 0.99^{\text{complexity}}$ to serve as the final solution, considering a trade-off between both accuracy and complexity.

4.5 Details of test data

To ensure that no data leakage occurred, we strictly split the training and test datasets. While it is not straightforward to identify mathematically equivalent formulas via symbolic derivations, we chose to measure the similarity between two formulas based on numerical evaluations. Specifically, we define the similarity between two formulas, f_i and f_j , with the same number of features, using

the averaged R^2 score:

$$\text{sim}(f_i, f_j) \stackrel{\text{def}}{=} 1 - \frac{1}{2} \left[\frac{\sum_{k=1}^{N'} (f_i(\mathbf{x}_k) - f_j(\mathbf{x}_k))^2}{\sum_{k=1}^{N'} (f_i(\mathbf{x}_k) - \hat{f}_i)^2} + \frac{\sum_{k=1}^{N'} (f_i(\mathbf{x}_k) - f_j(\mathbf{x}_k))^2}{\sum_{k=1}^{N'} (f_i(\mathbf{x}_k) - \hat{f}_j)^2} \right],$$

where $\hat{f}_i \stackrel{\text{def}}{=} [\sum_{k=1}^{N'} f_i(\mathbf{x}_k)]/N'$, $N' = 40$, and the $\mathbf{x}_k$'s are independently sampled from the standard multivariate Gaussian distribution. The similarity is set to 0 if there exists an $\mathbf{x}_k$ such that exactly one of $f_i(\mathbf{x}_k)$ and $f_j(\mathbf{x}_k)$ is undefined. The similarity between two formulas with a different number of features is also set to 0. Observe that two mathematically equivalent formulas achieve the maximum possible similarity 1.

We selected 3,000 formulas such that the pairwise similarity is less than 0.99 to form our test set. For the training dataset, we only removed the symbolically identical formulas while not requiring the pairwise similarity to be away from 1. This is because formulas that are mathematically equivalent but with different symbolic forms may help the model understand the mathematical equivalences. Moreover, such formulas might originate from different physical scenarios and have different physical dimensions, thus still represent distinct entities. The formulas in the test set were further divided into three difficulty levels based on their maximum similarity with the formulas in the training dataset. Formulas with a similarity greater than 0.95 were defined as the ones that were easy to predict, those with a similarity between 0.8 and 0.95 were considered as medium difficulty, and formulas with a similarity less than 0.8 were considered as hard.

4.6 Evaluation metrics

We evaluate the performance of our model and other baseline models using the following metrics.

- *Symbolic accuracy.* This evaluation metric was introduced by Cava et al.^[32] Let the target formula be f . The symbolic accuracy of a predicted formula g is 1 if there exists a constant such that $f - g \equiv c$ or $f = c \cdot g$ ($c \neq 0$ in the second case); otherwise, the symbolic accuracy is 0.
- *Numerical precision.* This metric is used to evaluate the numerical precision of data points given by predicted formula. We utilize R^2 -score for the predicted formula g on testing data points $\{\mathbf{x}_k^{\text{test}}, y_k^{\text{test}}\}_{k=1}^N$ ($N = 200$) which are sampled from the same distribution as the training data points:

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i^{\text{test}} - g(\mathbf{x}_i^{\text{test}}))^2}{\sum_{i=1}^N (y_i^{\text{test}} - \bar{y})^2}$$

where $\bar{y} \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N y_i^{\text{test}}$.

- *Accuracy of physical units.* This metric aims to assess the capability of a model to generate formulas with consistent physical units. The accuracy is 1 if all operations in the formula work with compatible physical units and the physical unit of the result is the same as the target formula; otherwise, the accuracy of physical units is 0.
- *Formula complexity.* The complexity of a formula f is defined to be the number of operators, variables and constant parameters in the formula. We also define the relative complexity of a prediction g is the difference (in absolute value) of the complexity of g and that of the target formula f .
- *Formula depth.* The depth of a formula f is the maximum depth of its expression tree.

5 Data availability

AI Feynman data can be downloaded from Feynman Symbolic Regression Database (<https://space.mit.edu/home/tegmark/aifeynman.html>). The formula generated by the LLM, including the training and test datasets can be downloaded from https://drive.google.com/drive/folders/17rbDLb2ZBgK9DidJtb1nyBFmGtOokhYs?usp=share_link. The sunspot number data could be downloaded from the Sunspot Index and Long-term Solar Observations website (<https://www.sidc.be/SILSO/datafiles>). The plasma pressure data could be downloaded from Geotail and Time History of Events and Macroscale Interactions During Substorms (THEMIS) website (https://themis.igpp.ucla.edu/overview_data.shtml). The solar rotational angular velocity data could be found at table presented by Snodgrass et al. 1983 in Magnetic Rotation of the Solar Photosphere^[45]. We collected the contribution function of emission lines data from the CHIANTI website (<http://www.chiantidatabase.org>). Lunar tide signal data was downloaded from RBSP/EFW official website (<https://www.space.umn.edu/rbspefw-data/>).

6 Code availability

Codes for running PhyE2E including both training and test modules are accessible at <https://github.com/Jie0618/PhysicsRegression>. The pre-trained PhyE2E can be downloaded at <https://drive.google.com/drive/folders/14M0Ed0gvSKmtuTOornfEoup8l48IfEUW?usp=sharing>.

References

1. Schmidt, M. D. & Lipson, H. Age-fitness pareto optimization. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 543–544 (2010).
2. Schmidt, M. & Lipson, H. Distilling free-form natural laws from experimental data. *Science* **324**, 81–85 (2009).
3. La Cava, W., Helmuth, T., Spector, L. & Moore, J. H. A probabilistic and multi-objective analysis of lexicase selection and ε -lexicase selection. *Evolutionary Computation* **27**, 377–402 (2019).
4. La Cava, W., Singh, T. R., Taggart, J., Suri, S. & Moore, J. H. Learning concise representations for regression by evolving networks of trees. In *International Conference on Learning Representations* (2019).
5. Virgolin, M., Alderliesten, T., Witteveen, C. & Bosman, P. A. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary Computation* **29**, 211–237 (2021).
6. McCormick, T. gplearn: Genetic Programming in Python (2019).
7. de Franca, F. & Aldeia, G. Interaction–transformation evolutionary algorithm for symbolic regression. *Evolutionary Computation* **29** (2021).
8. Arnaldo, I., Krawiec, K. & O’Reilly, U.-M. Multiple regression genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 879–886 (2014).

9. Kommenda, M., Burlacu, B., Kronberger, G. & Affenzeller, M. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming Evolvable Machines* **21**, 471–501 (2020).
10. Virgolin, M., Alderliesten, T. & Bosman, P. A. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In *Proceedings of the genetic and evolutionary computation conference*, 1084–1092 (2019).
11. Kamienny, P.-A., Lample, G., Lamprier, S. & Virgolin, M. Deep generative symbolic regression with monte-carlo-tree-search. In *International Conference on Machine Learning*, 15655–15668 (PMLR, 2023).
12. Lu, Q., Tao, F., Zhou, S. & Wang, Z. Incorporating actor-critic in monte carlo tree search for symbolic regression. *Neural Computing Applications* **33**, 8495–8511 (2021).
13. Xu, Y., Liu, Y. & Sun, H. Rsrn: Reinforcement symbolic regression machine. *arXiv preprint arXiv:2305.14656* (2023).
14. Xie, Y. *et al.* An efficient and generalizable symbolic regression method for time series analysis. *arXiv preprint arXiv:2409.03986* (2024).
15. Sun, F., Liu, Y., Wang, J.-X. & Sun, H. Symbolic physics learner: Discovering governing equations via monte carlo tree search. *arXiv preprint arXiv:2205.13134* (2022).
16. Valipour, M., You, B., Panju, M. & Ghodsi, A. Symbolicgpt: A generative transformer model for symbolic regression. *arXiv preprint arXiv:2106.14131* (2021).
17. Chen, T., Xu, P. & Zheng, H. Bootstrapping ots-funcing pre-training model (botfip)—a comprehensive symbolic regression framework. *arXiv preprint arXiv:2401.09748* (2024).
18. Xing, H., Salleb-Aouissi, A. & Verma, N. Automated symbolic law discovery: A computer vision approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 660–668 (2021).
19. Kamienny, P.-A., d’Ascoli, S., Lample, G. & Charton, F. End-to-end symbolic regression with transformers. *Advances Neural Information Processing Systems* **35**, 10269–10281 (2022).
20. Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A. & Parascandolo, G. Neural symbolic regression that scales. In *International Conference on Machine Learning*, 936–945 (Pmlr, 2021).
21. Shojaei, P., Meidani, K., Barati Farimani, A. & Reddy, C. Transformer-based planning for symbolic regression. *Advances Neural Information Processing Systems* **36**, 45907–45919 (2023).
22. Udrescu, S.-M. & Tegmark, M. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances* **6**, eaay2631 (2020).
23. Cranmer, M. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582* (2023).
24. Burlacu, B., Kronberger, G. & Kommenda, M. Operon c++: An efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO ’20, 1562–1570, DOI: [10.1145/3377929.3398099](https://doi.org/10.1145/3377929.3398099) (Association for Computing Machinery, New York, NY, USA, 2020).
25. Udrescu, S.-M. *et al.* Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *Advances Neural Information Processing Systems* **33**, 4860–4871 (2020).

26. Landajuela, M. *et al.* A unified framework for deep symbolic regression. *Advances Neural Information Processing Systems* **35**, 33985–33998 (2022).
27. Tenachi, W., Ibata, R. & Diakogiannis, F. I. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. *The Astrophysical Journal* **959**, 99 (2023).
28. Scholl, P., Bieker, K., Hauger, H. & Kutyniok, G. Parfam–symbolic regression based on continuous global optimization. *arXiv preprint arXiv:2310.05537* (2023).
29. Liu, Z. *et al.* Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756* (2024).
30. Liu, Z., Ma, P., Wang, Y., Matusik, W. & Tegmark, M. Kan 2.0: Kolmogorov-arnold networks meet science. *arXiv preprint arXiv:2408.10205* (2024).
31. Jin, Y., Fu, W., Kang, J., Guo, J. & Guo, J. Bayesian symbolic regression. *arXiv preprint arXiv:1910.08892* (2019).
32. La Cava, W. *et al.* Contemporary symbolic regression methods and their relative performance. *arXiv preprint arXiv:2107.14351* (2021).
33. Feynman, R., Leighton, R. & Sands, M. *The Feynman Lectures on Physics: The New Millennium Edition: Mainly Mechanics, Radiation, and Heat*, vol. 1 (Basic Books, 1963).
34. Feynman, R., Leighton, R. & Sands, M. *The Feynman Lectures on Physics*, vol. 2 (Pearson/Addison-Wesley, 1963).
35. Feynman, R., Leighton, R. & Sands, M. *The Feynman Lectures on Physics*, vol. 3 (Pearson/Addison-Wesley, 1963).
36. SILSO World Data Center. The International Sunspot Number. *International Sunspot Number Monthly Bulletin online catalogue* (1749–2023).
37. Hathaway, D. H., Wilson, R. M. & Reichmann, E. J. The shape of the sunspot cycle. *Solar Physics* **151**, 177–190 (1994).
38. Solar cycle precursors and the outlook for cycle 25. *Journal Geophysical Research: Space Physics* **128**, e2023JA031681 (2023).
39. Brehm, N. *et al.* Eleven-year solar cycles over the last millennium revealed by radiocarbon in tree rings. *Nature Geoscience* **14**, 10–15 (2021).
40. Wang, C.-P. *et al.* Empirical modeling of plasma sheet pressure and three-dimensional force-balanced magnetospheric magnetic field structure: 1. observation. *Journal Geophysical Research: Space Physics* **118**, 6154–6165 (2013).
41. Yue, C., Wang, C.-P., Zaharia, S. G., Xing, X. & Lyons, L. Empirical modeling of plasma sheet pressure and three-dimensional force-balanced magnetospheric magnetic field structure: 2. modeling. *Journal Geophysical Research: Space Physics* **118**, 6166–6175 (2013).
42. Lui, A. T. & Hamilton, D. C. Radial profiles of quiet time magnetospheric parameters. *Journal Geophysical Research: Space Physics* **97**, 19325–19332 (1992).
43. Hotta, H. & Kusano, K. Solar differential rotation reproduced with high-resolution simulation. *Nature Astronomy* **5**, 1100–1102, DOI: [10.1038/s41550-021-01459-0](https://doi.org/10.1038/s41550-021-01459-0) (2021).
44. Vasil, G. M. *et al.* The solar dynamo begins near the surface. *Nature* **629**, 769–772, DOI: [10.1038/s41586-024-07315-1](https://doi.org/10.1038/s41586-024-07315-1) (2024).

45. Magnetic rotation of the solar photosphere. *Astrophysical Journal, Part 1 (ISSN 0004-637X)*, vol. 270, July 1, 1983, p. 288-299. **270**, 288–299 (1983).
46. Rao, S. *et al.* Height-dependent differential rotation of the solar atmosphere detected by chase. *Nature Astronomy* 1–8 (2024).
47. Dere, K. P., Landi, E., Mason, H. E., Monsignori Fossi, B. C. & Young, P. R. CHIANTI - an atomic database for emission lines. *Astronomy Astrophysics Supplement Series* **125**, 149–173, DOI: [10.1051/aas:1997368](https://doi.org/10.1051/aas:1997368) (1997).
48. Dufresne, R. P. *et al.* CHIANTI—An Atomic Database for Emission Lines—Paper. XVIII. Version 11, Advanced Ionization Equilibrium Models: Density and Charge Transfer Effects. *The Astrophysical Journal* **974**, 71, DOI: [10.3847/1538-4357/ad6765](https://doi.org/10.3847/1538-4357/ad6765) (2024). [2403.16922](https://arxiv.org/abs/2403.16922).
49. Kramida, A., Ralchenko, Y., Reader, J. & Team, N. A. Nist atomic spectra database (version 5.11). <https://physics.nist.gov/asd>, DOI: <https://doi.org/10.18434/T4W30F> (2023). [Online; accessed 20-Oct-2024].
50. Aschwanden, M. J. *Physics of the Solar Corona. An Introduction* (2004).
51. Mason, H. E. & Monsignori Fossi, B. C. Spectroscopic diagnostics in the vuv for solar and stellar plasmas. *The Astronomy Astrophysics Review* **6**, 123–179, DOI: [10.1007/BF01208253](https://doi.org/10.1007/BF01208253) (1994).
52. Raymond, J. C. & Smith, B. W. Soft X-ray spectrum of a hot plasma. *The Astrophysical Journal Supplement Series* **35**, 419–439, DOI: [10.1086/190486](https://doi.org/10.1086/190486) (1977).
53. Xiao, C. *et al.* Evidence for lunar tide effects in earth’s plasmasphere. *Nature Physics* **19**, 486–491 (2023).
54. Zhang, Z., Liu, W. L., Zhang, D. J. & Cao, J. B. Estimating the corotation lag of the plasmasphere based on the electric field measurements of the van allen probes. *Advances Space Research* **73**, 758–766 (2024).
55. Touvron, H. *et al.* Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
56. Maurya, A., Ye, J., Rafique, M. M., Cappello, F. & Nicolae, B. Deep optimizer states: Towards scalable training of transformer models using interleaved offloading. *arXiv preprint arXiv:2410.21316* (2024).
57. Lample, G. & Charton, F. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412* (2019).
58. Charton, F. Linear algebra with transformers. *arXiv preprint arXiv:2112.01898* (2021).
59. Bendinelli, T., Biggio, L. & Kamienny, P.-A. Controllable neural symbolic regression. In *International Conference on Machine Learning*, 2063–2077 (PMLR, 2023).
60. Vaswani, A. *et al.* Attention is all you need. In *Advances in Neural Information Processing Systems*, vol. 30, 5998–6008 (2017).
61. Fletcher, R. *Practical Methods of Optimization* (John Wiley & Sons, New York, 1987), 2nd edn.

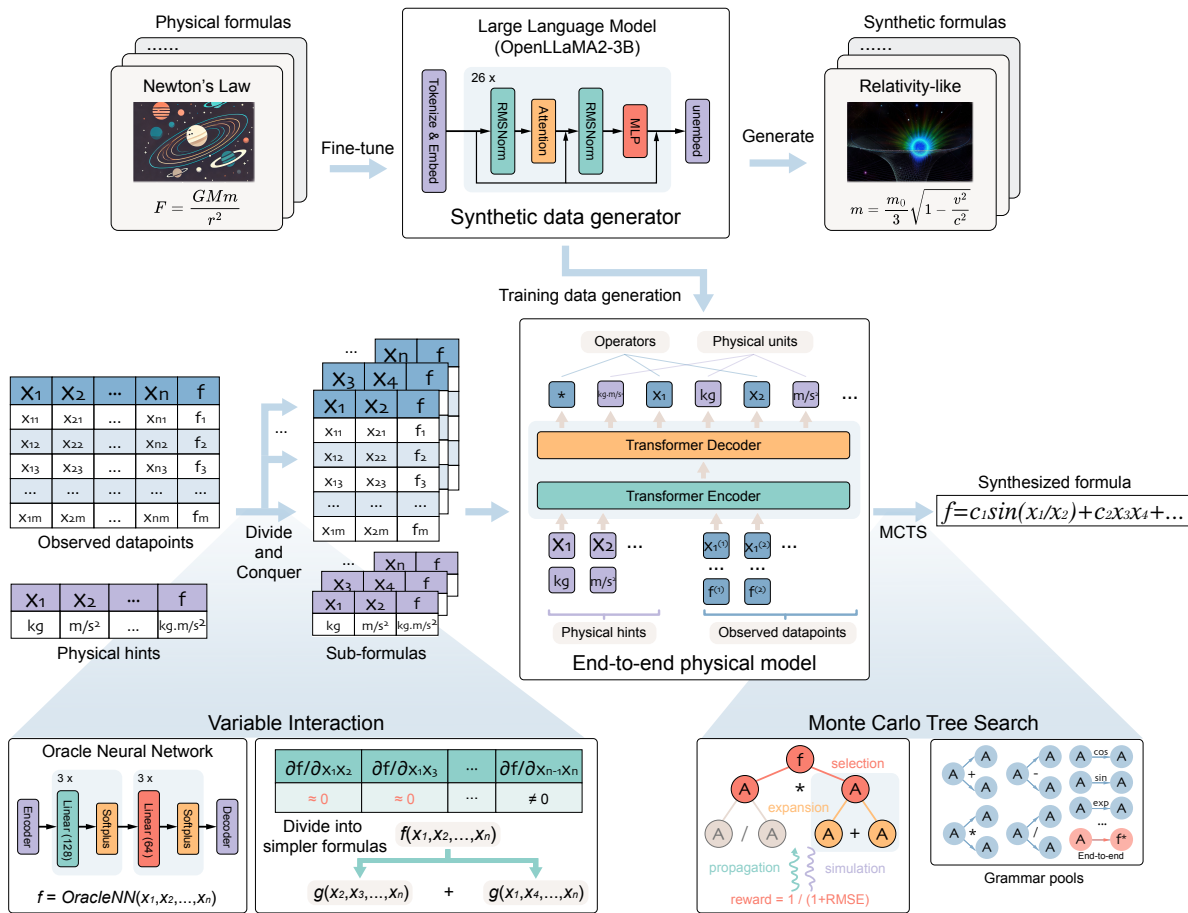


Figure 1. The overall PhyE2E framework. **Top.** The training dataset was augmented with a large-scale synthetic dataset generated by a large language model. **Middle.** A variable interaction technique was integrated to decompose the original symbolic regression problem into simpler sub-problems, referred to as Divide-and-Conquer(D&C). An end-to-end model was trained to predict the target formula using observed data points and prior physics knowledge (referred to as “physical hints”). **Bottom.** Monte Carlo Tree Search (MCTS) module was adopted to refine the generated formulas, using a context-free grammar pool that includes atomic formulas and the end-to-end generated formula.

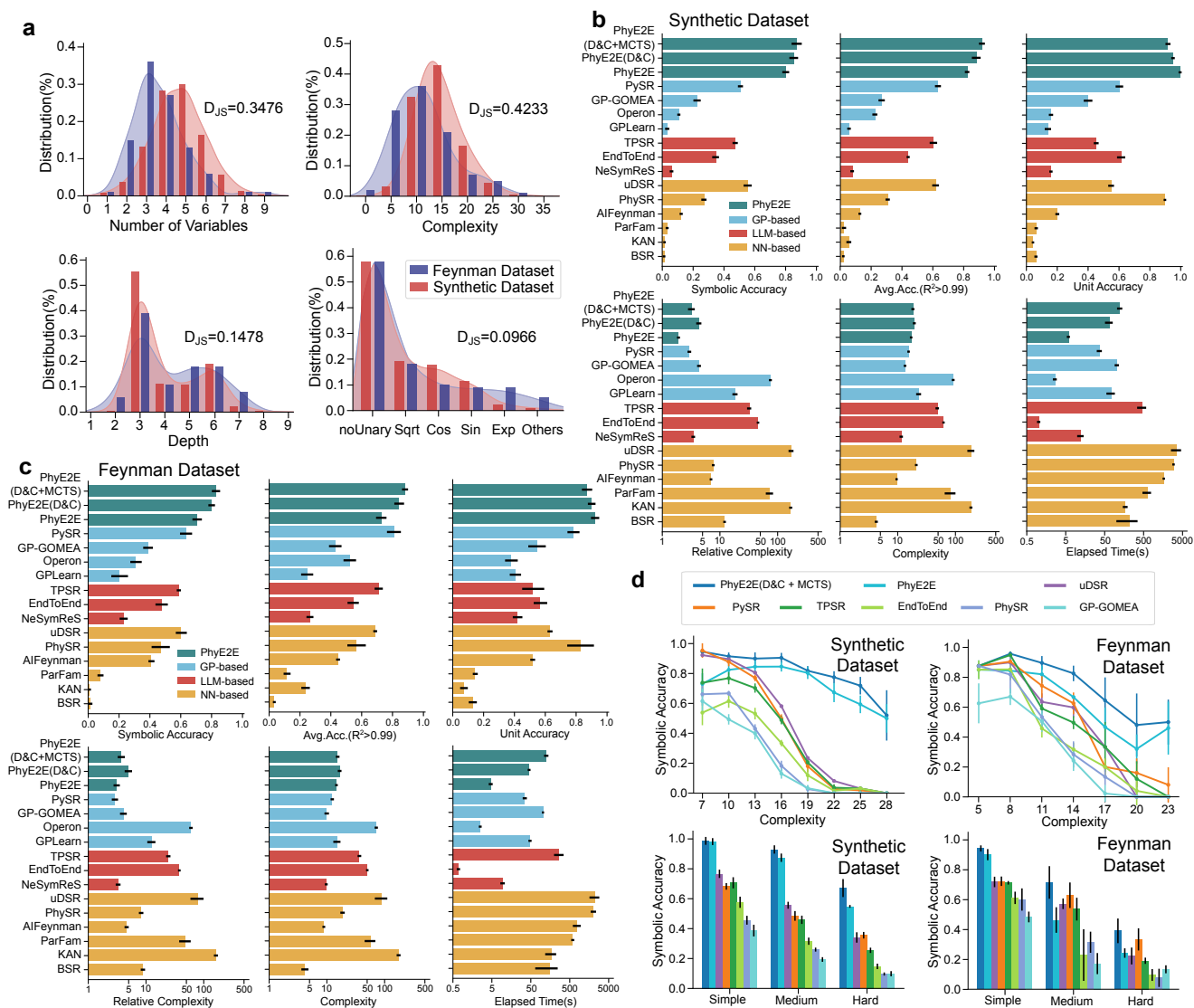


Figure 2. Performance on the synthetic and AI Feynman datasets. **a**, Comparison between the formulas generated from LLaMa2 and the Feynman Dataset. The distance between the distributions of different properties of the two sets of formulas is measured using the Jensen-Shannon divergence (D_{JS}). **b**, Evaluation results for Symbolic Regression methods on the test set of the synthetic dataset. **c**, Evaluation results for Symbolic Regression methods on the AI Feynman dataset. **d**, Evaluation results on formulas with different complexity (upper panels) and different difficulties (bottom panels) on the synthetic and AI Feynman datasets.

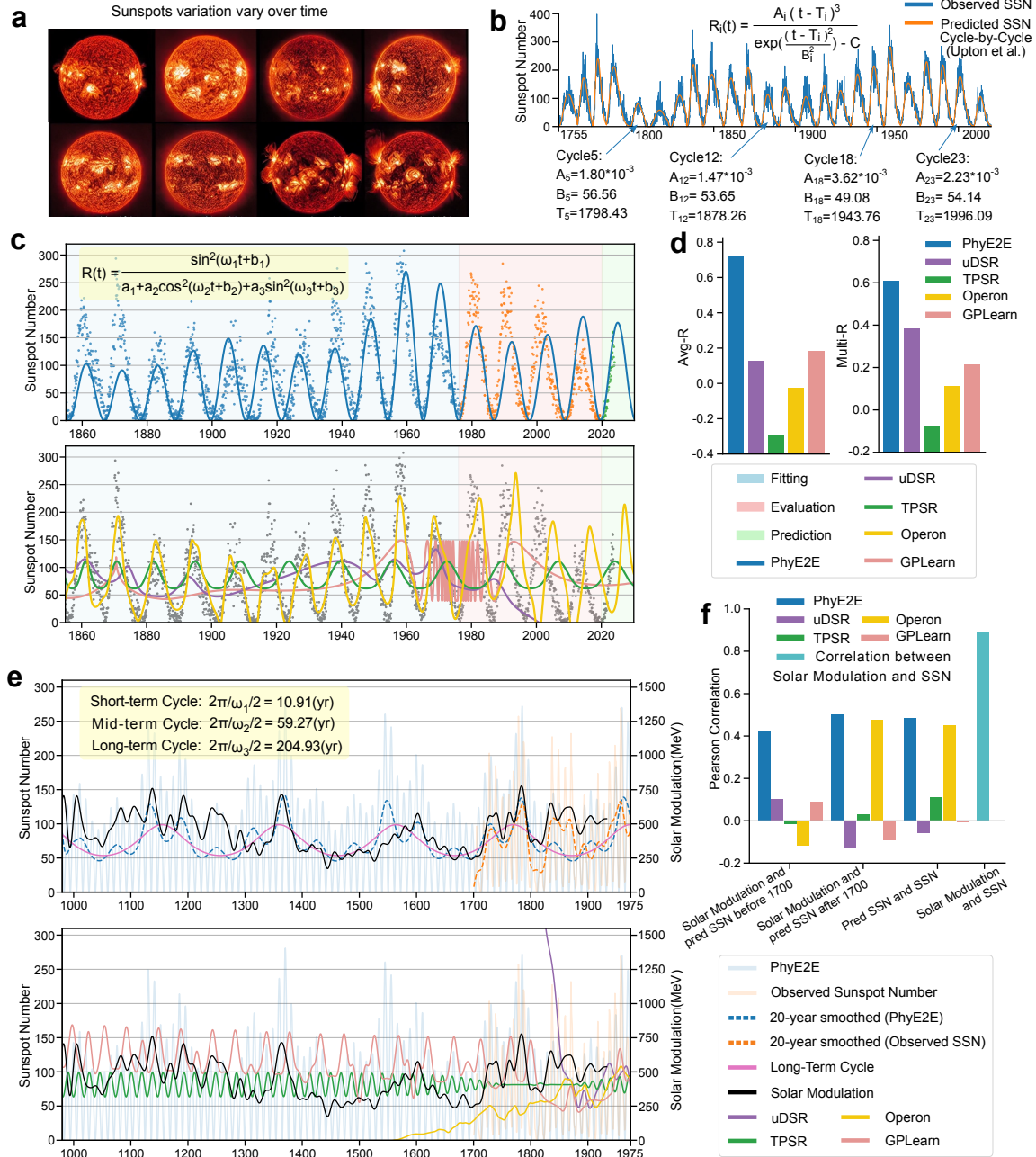


Figure 3. Performance of sunspot intensity prediction **a**, Sunspot variation over time. **b**, Variations in SSN observed through telescopes from 1755 to 2020 and the formula derived by Hathaway et al., 1994. **c**, The PhyE2E formula and the variations in SSN yielded by the formula from 1855 to 1976 (top). The formulas generated by other baseline models and the variations in SSN yielded by these formulas from 1855 to 1976 (bottom). **d**, Avg-R (left) and Multi-R (right) on the test data from 1976 to 2019 for different baseline models. **e**, Solar modulation level and smoothed SSN from different baseline models over longer time frame from 980 to 1976. **f**, Pearson Correlation between the SSN observed by telescopes, SSN predicted by the generated formulas, and Solar Modulation level from 980 to 1932.

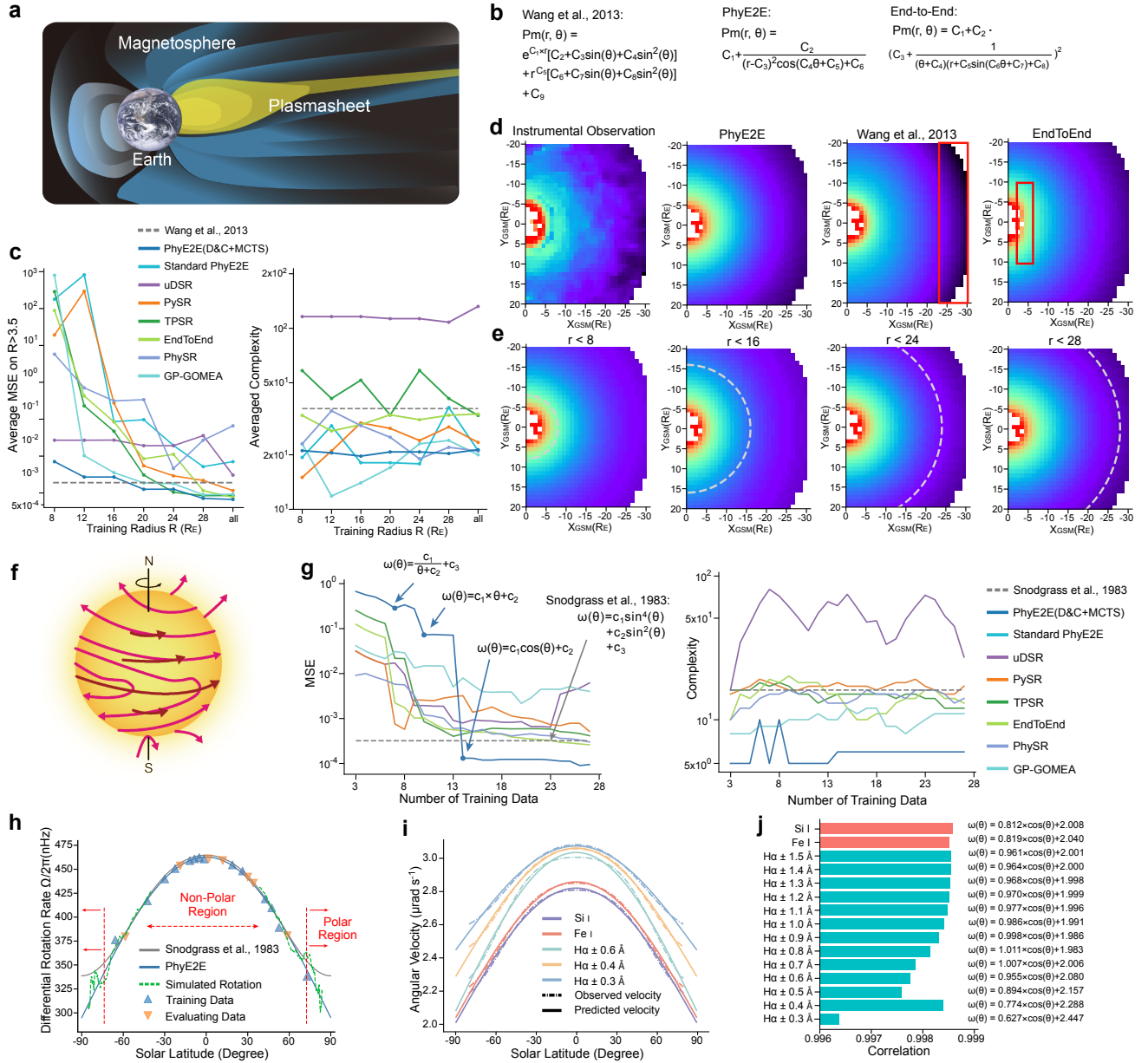


Figure 4. Performance of plasma sheet pressure prediction and solar differential rotation prediction **a**, The distribution of near-Earth magnetosphere and plasmasheet. **b**, Average Mean Square Error (left) and complexity (right) when utilizing data from different radius for models to be compared. **c**, Instrumental observations and formula predictions for plasma sheet pressure using different models. **d**, Predictions for plasma sheet pressure using data from different radius by PhyE2E. **e**, Symbolic formulas of Wang et al., 2013 and PhyE2E. **f**, Solar rotation varies at different latitudes, making magnetic field lines stretched and twisted. **g**, Predictions from Snodgrass et al., 1993 and PhyE2E across all the latitudes. **h**, MSE and complexity from different models using different numbers of training data. **i**, PhyE2E predicts consistent formulas with high robustness across various spectral lines.

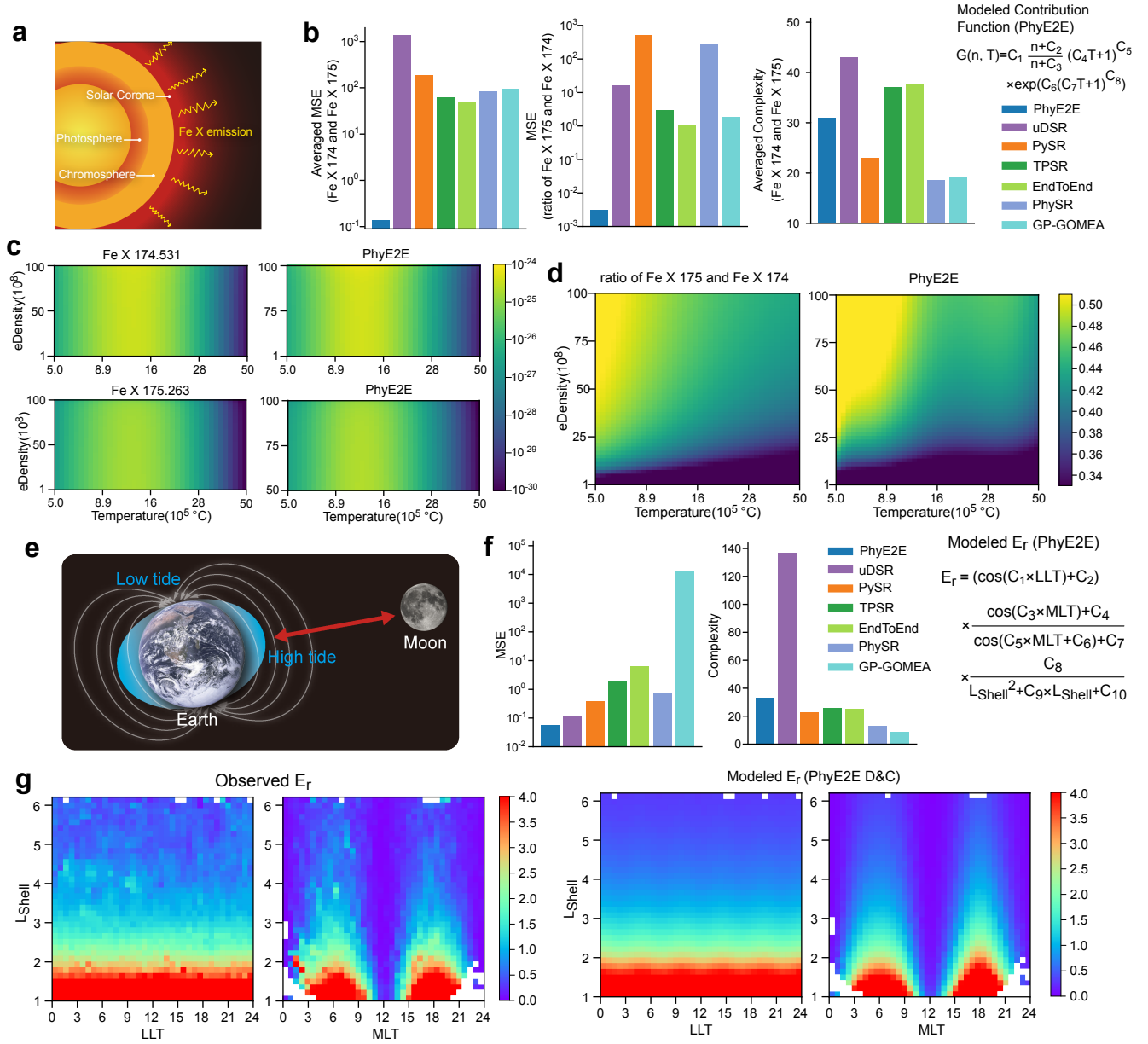


Figure 5. Performance of contribution function of emission lines predictions and lunar tide signal of plasma layer predictions **a**, Emission lines in the extreme ultraviolet spectrum of the Sun. **b**, Average MSE for Fe X 174 and Fe X 175 (left), MSE of the ratio between the two emission lines (middle), and the complexity (right) of the formulas generated by different models to be compared. **c**, Instrumental measured contribution function and PhyE2E predictions for Fe X 174 and Fe X 175. **d**, Instrumental measured ratio of the two emission lines and PhyE2E predictions. **e**, Tidal radial electric fields influences the Earth's magnetospheric electric fields. **f**, MSE and complexity for different models to be compared. **g**, Instrumental measured radial electric field (E_r) (left) and PhyE2E predictions for dayside and nightside of the Earth.

A Supplementary Figures

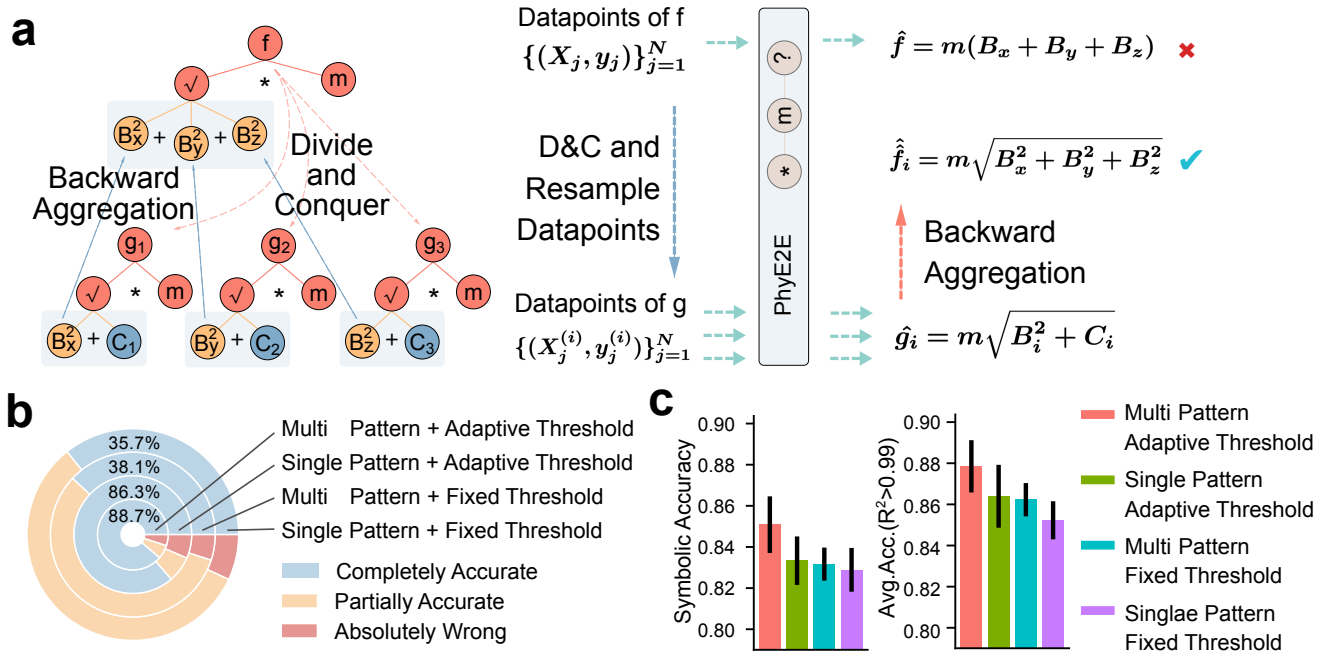


Figure S1. Performance on Synthetic and AI Feynman datasets. **a**, An example of D&C procedure, including Divide-and-Conquer and Backward Aggregation step. **b**, Decomposition accuracy for different D&C strategies. **c**, Symbolic accuracy and average accuracy of $R^2 > 0.99$ under different D&C strategies.

B Supplementary Tables

Table S1. Real-world physics formulas from Feynman Datasets

Eq.	Formula	Eq.	Formula	Eq.	Formula
I.6.2a	$\exp(-\theta^2/2)/\sqrt{2\pi}$	I.6.2b	$\exp(-(\theta/\sigma)^2/2)/(\sqrt{2\pi}\sigma)$	I.6.2	$\exp(-((\theta - \theta_1)/\sigma)^2/2)/(\sqrt{2\pi}\sigma)$
I.8.14	$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	I.9.18	$\frac{Gm_1m_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$	I.10.7	$\frac{m_0}{\sqrt{1-v^2/c^2}}$
I.11.19	$x_1y_1 + x_2y_2 + x_3y_3$	I.12.1	μN_n	I.12.2	$\frac{q_1q_2\epsilon}{4\pi\epsilon r^3}$
I.12.4	$\frac{q_1r}{4\pi\epsilon r^3}$	I.12.5	$q_2 \cdot E_f$	I.12.11	$q \cdot (E_f + Bv \sin(\theta))$
I.13.4	$\frac{1}{2}m(v^2 + u^2 + w^2)$	I.13.12	$Gm_1m_2 \left(\frac{1}{r_2} - \frac{1}{r_1} \right)$	I.14.3	mgz
I.14.4	$\frac{1}{2}k_{\text{spring}}x^2$	I.15.3x	$\frac{x-ut}{\sqrt{1-u^2/c^2}}$	I.15.3t	$\frac{t-ux/c^2}{\sqrt{1-u^2/c^2}}$
I.15.1	$\frac{m_0v}{\sqrt{1-v^2/c^2}}$	I.16.6	$\frac{u+v}{1+uv/c^2}$	I.18.4	$\frac{m_1r_1+m_2r_2}{m_1+m_2}$
I.18.12	$rF \sin(\theta)$	I.18.14	$mr v \sin(\theta)$	I.24.6	$\frac{1}{2}m(\omega^2 + \omega_0^2)\frac{1}{2}x^2$
I.25.13	q/C	I.26.2	$\arcsin(n \sin(\theta_2))$	I.27.6	$\frac{1}{1/d_1+n/d_2}$
I.29.4	ω/c	I.29.16	$\sqrt{x_1^2 + x_2^2 - 2x_1x_2 \cos(\theta_1 - \theta_2)}$	I.30.3	$\text{Int}_0 \cdot \sin(n\theta/2)^2 / \sin(\theta/2)^2$
I.30.5	$\arcsin(\lambda/(nd))$	I.32.5	$\frac{q^2a^2}{6\pi\epsilon c^3}$	I.32.17	$\left(\frac{1}{2}\epsilon c E_f^2\right) \left(\frac{8\pi r^2}{3}\right) \frac{\omega^4}{(\omega^2 - \omega_0^2)^2}$
I.34.8	qvB/p	I.34.1	$\frac{\omega_0}{1-v/c}$	I.34.14	$\frac{1+v/c}{\sqrt{1-v^2/c^2}}\omega_0$
I.34.27	$\frac{h}{2\pi}\omega$	I.37.4	$I_1 + I_2 + 2\sqrt{I_1I_2} \cos(\delta)$	I.38.12	$4\pi\epsilon \left(\frac{h}{2\pi}\right)^2 / (mq^2)$
I.39.1	$\frac{3}{2}prV$	I.39.11	$\frac{1}{\gamma-1}prV$	I.39.22	$n \cdot kb \cdot T/V$
I.40.1	$n_0 \exp(-mgx/(kb \cdot T))$	I.41.16	$\frac{h}{2\pi}\omega^3 / (\pi^2c^2 (\exp(\frac{h}{2\pi}\omega/(k_B T)) - 1))$	I.43.16	$\mu_{\text{drift}} \cdot q \cdot \text{Volt}/d$
I.43.31	$\text{mob} \cdot kb \cdot T$	I.43.43	$\frac{1}{\gamma-1} \cdot kb \cdot v/A$	I.44.4	$n \cdot kb \cdot T \ln(V_2/V_1)$
I.47.23	$\sqrt{\gamma p_r/\rho}$	I.48.2	$\frac{mc^2}{\sqrt{1-v^2/c^2}}$	I.50.26	$x_1(\cos(\omega t) + \alpha \cos^2(\omega t))$
II.2.42	$\kappa(T_2 - T_1)A/d$	II.3.24	$P_{\text{wr}}/(4\pi r^2)$	II.4.23	$q/(4\pi\epsilon r)$
II.6.11	$\frac{1}{4\pi\epsilon}p_d \cos(\theta)/r^2$	II.6.15a	$\frac{p_d}{4\pi\epsilon} \frac{3z}{r^3} \sqrt{x^2 + y^2}$	II.6.15b	$\frac{p_d}{4\pi\epsilon} \frac{3 \cos(\theta) \sin(\theta)}{r^3}$
II.8.7	$\frac{3}{5}q^2/(4\pi\epsilon d)$	II.8.31	$\epsilon E_f^2/2$	II.10.9	$\frac{\sigma_{\text{den}}}{\epsilon} \frac{1}{1+\chi}$
II.11.3	$\frac{qE_f}{m(\omega_0^2 - \omega^2)}$	II.11.17	$n_0(1 + p_d E_f \cos(\theta)/(kb \cdot T))$	II.11.20	$n_\rho p_d^2 E_f / (3kb \cdot T)$
II.11.27	$n\alpha / (1 - n\alpha/3) \epsilon E_f$	II.11.28	$1 + n\alpha / (1 - n\alpha/3)$	II.13.17	$\frac{1}{4\pi\epsilon c^2} \frac{2I}{r}$
II.13.23	$\rho_{c0}/\sqrt{1-v^2/c^2}$	II.13.34	$\frac{\rho_{c0}v}{\sqrt{1-v^2/c^2}}$	II.15.4	$-momB \cos(\theta)$
II.15.5	$-p_d \cdot E_f \cdot \cos(\theta)$	II.21.32	$\frac{q}{4\pi\epsilon r(1-v/c)}$	II.24.17	$\sqrt{\omega^2/c^2 - \pi^2/d^2}$
II.27.16	$\epsilon c E_f^2$	II.27.18	ϵE_f^2	II.34.2a	$\frac{qv}{2\pi r}$
II.34.2	$qvr/2$	II.34.11	$\frac{q_s q B}{2m}$	II.34.29a	$\frac{qh}{4\pi m}$
II.34.29b	$\frac{g_s om B J_\pm}{h/(2\pi)}$	II.35.18	$\frac{n_0}{\exp\left(\frac{momB}{k_B T}\right) + \exp\left(-\frac{momB}{k_B T}\right)}$	II.35.21	$n_\rho mom \tanh\left(\frac{momB}{k_B T}\right)$
II.36.38	$mom \frac{H}{k_B T} + \frac{mom\alpha}{\epsilon c^2 k_B T} M$	II.37.1	$mom(1 + \chi)B$	II.38.3	$Y Ax/d$
II.38.14	$\frac{Y}{2(1+\sigma)}$	III.4.32	$\frac{1}{\exp\left(\frac{h}{2\pi} \frac{\omega}{k_B T}\right) - 1}$	III.4.33	$\frac{h}{2\pi} \frac{\omega}{\exp\left(\frac{h}{2\pi} \frac{\omega}{k_B T}\right) - 1}$
III.7.38	$\frac{2momB}{h/(2\pi)}$	III.8.54	$\sin\left(\frac{E_n t}{h/(2\pi)}\right)^2$	III.9.52	$\frac{p_d E_f t}{h/(2\pi)} \frac{\sin((\omega - \omega_0)t/2)^2}{((\omega - \omega_0)t/2)^2}$
III.10.19	$mom\sqrt{B_x^2 + B_y^2 + B_z^2}$	III.12.43	$n\left(\frac{h}{2\pi}\right)$	III.13.18	$2E_n d^2 k / (h/(2\pi))$
III.14.14	$I_0 \left(\exp\left(\frac{q\text{Volt}}{k_B T}\right) - 1 \right)$	III.15.12	$2U(1 - \cos(kd))$	III.15.14	$\frac{(h/(2\pi))^2}{2E_n d^2}$
III.15.27	$\frac{2\pi\alpha}{nd}$	III.17.37	$\beta(1 + \alpha \cos(\theta))$	III.19.51	$-mq^4/(2(4\pi\epsilon)^2(h/(2\pi))^2)$
III.21.20	$-rho_{c0} \cdot q \cdot A_{\text{vec}}/m$				

Table S2. Constants of the derived formula from PhyE2E for SSN prediction

ω_1	ω_2	ω_3	a_1	a_2	a_3	b_1	b_2	b_3
0.288	0.053	0.01533	0.00343	0.00327	0.00428	-0.432	-5.035	0.0184

Table S3. Constants of derived formulas from PhyE2E for plasma pressure prediction

	C_1	C_2	C_3	C_4	C_5	C_6
$K_p = 0 \quad P_{SW} = 1.5$	0.0368	0.502	1.312	0.0153	0.0240	-63.424
$K_p = 0 \quad P_{SW} = 3.0$	0.0617	0.122	0.666	0.00705	0.0111	-63.872
$K_p = 1 \quad P_{SW} = 1.5$	0.0280	4099.008	0.418	5.430	8.531	3897.984
$K_p = 1 \quad P_{SW} = 3.0$	0.0654	5233.664	-0.516	5.467	8.586	3447.424
$K_p = 2 \quad P_{SW} = 1.5$	0.0613	363.904	2.176	2.011	3.161	414.336
$K_p = 2 \quad P_{SW} = 3.0$	0.108	4480.448	1.568	3.507	5.515	3832.704
$K_p = 3 \quad P_{SW} = 1.5$	0.0459	1920.832	1.528	3.496	5.492	1876.800
$K_p = 3 \quad P_{SW} = 3.0$	0.0813	2835.456	0.725	3.753	5.896	1724.992
$K_p = 4 \quad P_{SW} = 1.5$	0.0538	1731.456	1.824	3.931	6.176	1645.632
$K_p = 4 \quad P_{SW} = 3.0$	0.0922	2260.416	0.952	4.006	6.299	1311.872
$K_p = 5 \quad P_{SW} = 1.5$	0.0648	945.792	2.568	4.116	6.469	959.808
$K_p = 5 \quad P_{SW} = 3.0$	0.112	1278.272	1.928	4.122	6.478	820.608

Table S4. Constants of derived formulas from PhyE2E for differential rotation prediction

C_1	C_2
1.104	1.807

Table S5. Constants of derived formulas from PhyE2E for the prediction of contribution functions

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
Fe X 174.531	1.561×10^{-5}	4.423×10^9	1.204×10^9	3.931×10^{-6}	-22.014	-216.711	2.071×10^{-6}	-2.058
Fe X 175.263	9.195×10^{-8}	3.342×10^9	8.094×10^9	3.186×10^{-6}	-22.023	-350.100	3.680×10^{-6}	-1.753

Table S6. Constants of derived formulas from PhyE2E for the prediction of lunar tide signal

C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
1.295	39.498	0.517	-1.079	0.829	1.525	12.621	-1.769	-0.583	0.746

Table S7. Formulas from PhyE2E for SSN prediction on different time frame

Training Years	Formula
1843-1976	$C_1 \sin(C_2 t - C_3)^2 / (C_4 \sin(C_5 t + C_6)^2 + C_7 \cos(C_8 t - C_9)^2 + C_{10})$
1833-1976	$C_1 (\cos(C_2 t - C_3)^2 + C_4)^{0.5}$
1822-1976	$C_1 \sin(C_2 t - C_3)^2 / (C_4 \sin(C_5 t - C_6)^2 + C_7 \cos(C_8 t - C_9)^2 + C_{10})$
1807-1976	$C_1 \sin(C_2 t - C_3)^2 / (C_4 \sin(C_5 t - C_6)^2 + C_7)$
1798-1976	$C_1 \sin(C_2 t - C_3)^2 / (C_4 - C_5 \sin(C_6 t - C_7)^2)$
1784-1976	$C_1 \sin(C_2 t - C_3)^2 / (C_4 \sin(C_5 t - C_6)^2 + C_7 \cos(C_8 t - C_9)^2 + C_{10})$
1775-1976	$C_1 \sin(C_2 t - C_3)^2 / (C_4 \sin(C_5 t - C_6)^2 + C_7)$
1766-1976	$C_1 \sin(C_2 t - C_3)^2 / (C_4 \sin(C_5 t - C_6)^2 - C_7 \cos(C_8 t - C_9)^2 + C_{10})$
1754-1976	$C_1 \cos(C_2 t - C_3)^2 + C_4$

Table S8. Constants of derived formulas from PhyE2E for SSN prediction on different time frame

Training Years	Constants									
1843-1976	C_1 497.433	C_2 0.287	C_3 658.997	C_4 0.847	C_5 0.0138	C_6 16.283	C_7 1.961	C_8 0.0362	C_9 179.001	C_{10} 1.912
1833-1976	C_1 137.219	C_2 0.286	C_3 532.729	C_4 0.00463						
1822-1976	C_1 156.474	C_2 0.287	C_3 538.283	C_4 -1.945	C_5 0.0284	C_6 47.570	C_7 1.667	C_8 0.258	C_9 845.447	C_{10} 2.324
1807-1976	C_1 286.012	C_2 0.0314	C_3 59.780	C_4 1.842	C_5 0.286	C_6 811.687	C_7 0.630			
1798-1976	C_1 248.281	C_2 0.0312	C_3 59.372	C_4 3.952	C_5 2.939	C_6 0.286	C_7 888.827			
1784-1976	C_1 273.327	C_2 0.287	C_3 544.136	C_4 0.602	C_5 0.0739	C_6 28.710	C_7 2.223	C_8 0.0330	C_9 106.837	C_{10} 0.853
1775-1976	C_1 150.131	C_2 0.315	C_3 594.433	C_4 1.165	C_5 0.0345	C_6 133.300	C_7 0.686			
1766-1976	C_1 96.270	C_2 0.283	C_3 534.797	C_4 4.130	C_5 0.0359	C_6 51.023	C_7 3.549	C_8 0.0368	C_9 98.159	C_{10} 0.490
1754-1976	C_1 91.322	C_2 0.282	C_3 522.318	C_4 34.675						

Table S9. Derived Formulas from all the baseline models for SSN prediction and its short-term cycle

Methods	Formulas	Short-term cycle (yr)
AlFeynman	$\arcsin (C_1 + C_2 \cdot e^{C_3 \cdot t})$	-
BSR	$-\sin (C_1 \cdot t - C_2)$	77.99
E2E	$C_1 \cdot (1 - C_2 \cdot \sin (C_3 \cdot t - C_4))^2 - C_5$	10.87
GP-GOMEA	$\frac{C_1 \cdot t - C_2}{C_3 \cdot t - C_4}$	-
GPLearn	$\left(C_1 - \cos \left(\sin \left(\frac{1}{-\sin (C_2 \cdot t - C_3) + \frac{1}{\log (C_4 \cdot \sqrt{C_5 \cdot t - 1})}} \right) \right) \right) \cdot ((C_6 - C_7 \cdot t) \cdot (\cos (\tan (\sin (\log (\log (C_8 \cdot t - C_9)))))) + C_{10}) + C_{11}) - C_{12} \cdot \sqrt{C_{13} \cdot t - 1}$	-
KAN	$C_1 \cdot \sin (C_2 \cdot t - C_3) + C_4$	98.78
NeSymReS	$C_1 \cdot \sin^4 (C_2 \cdot t - C_3)$	10.92
Operon	$C_1 \cdot \left(C_2 + \frac{(C_3 \cdot \cos (C_4 \cdot t + \cos (C_5 \cdot t - C_6) - C_7))}{\sqrt{(C_8 \cdot t + C_9 \cdot \cos (C_{10} \cdot t - C_{11}) + C_{12} \cdot \cos (C_{13} \cdot t - C_{14}) - 1)^2 + C_{15}}} \right) \cdot \left(C_{16} + \frac{C_{17} \cdot t - \sin (C_{18} \cdot t - C_{19}) + C_{20}}{\sqrt{\sin^2 (C_{21} \cdot t - C_{22}) + 1} \cdot \sqrt{\sin^2 (C_{23} \cdot t - C_{24}) + 1}} \right) + C_{25}$	11.33
ParFam	$(C_1 \cdot t + (C_2 \cdot t^2 + C_3 \cdot t \cdot \cos (C_4 \cdot t^2 + C_5 \cdot t + C_6) + C_7 \cdot t + C_8 \cdot \cos (C_9 \cdot t^2 + C_{10} \cdot t + C_{11}) + C_{12} \cdot e^{t \cdot (C_{13} \cdot t + C_{14})} + C_{15} \cdot \cos (C_{16} \cdot t^2 + C_{17} \cdot t + C_{18}) + C_{19} \cdot e^{t \cdot (C_{20} \cdot t + C_{21})})$	10.57
PhySR	$C_1 + C_2 \cdot e^{C_3 \cdot \sin (C_4 \cdot t - C_5)}$	10.58
PySR	$C_1 \cdot \sin (C_2 \cdot x_0 - C_3)^2$	10.94
TPSR	$C_1 \cdot \left(\sin \left(C_2 \cdot \sin \left(C_3 \cdot t + C_4 + \frac{C_5}{C_6 \cdot t - C_7} \right) + C_8 \right) - C_9 \right)^3 - C_{10}$	18.75
uDSR	$\frac{C_1 \cdot t - C_2 \cdot (C_3 \cdot t - 1)^3 + C_4 \cdot (C_5 \cdot t - 1)^2 + C_6}{\cos (\cos (e^{\sin (C_7 \cdot t - C_8)} \cdot \cos (\sin (C_9 \cdot t + e^{\sin (C_{10} \cdot t - C_{11})} - C_{12}))))}$	19.86

Table S10. Constants of derived formulas from all the baseline models for SSN prediction

Methods	Constants									
AIF	C_1	C_2	C_3							
	-0.0108	2.483×10^{53}	-0.0662							
BSR	C_1	C_2								
	0.0805	148.657								
E2E	C_1	C_2	C_3	C_4	C_5					
	1375.295	0.0232	0.578	1070.873	1294.074					
GP-GOMEA	C_1	C_2	C_3	C_4						
	4.790	9378.746	0.0662	129.714						
GPLearn	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
	1.178	0.0662	122.946	11.132	0.000534	122.946	0.0662	0.0662	123.936	0.977
	C_{11}	C_{12}	C_{13}							
	250	11.088	0.000539							
KAN	C_1	C_2	C_3	C_4						
	28.35	0.06360	122.658	79.02						
NeSymReS	C_1	C_2	C_3							
	171.798	0.288	533.667							
Operon	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
	0.995	-0.569	0.911	0.570	0.0573	106.324	1058.206	0.00124	0.911	0.576
	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
	1069.457	0.911	0.712	1320.759	0.831	0.869	-0.499	1.046	1941.547	780.735
	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}					
	0.0314	58.329	0.588	1090.857	0.770					
ParFam	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
	-0.270	0.192	13135.647	0.00539	0.509	945.212	0.522	1602.277	0.00539	965.415
	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
	39.222	0.522	1602.277	0.00539	965.415	309.784	0.0663	122.946	0.238	446.901
	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	C_{27}	C_{28}	C_{29}	C_{30}
	8.358	0.522	1602.277	0.00539	965.415	513.766	446.901	0.238	0.0663	122.946
PhySR	C_1	C_2	C_3	C_4	C_5					
	40.064	30.808	-1.272	0.594	1101.664					
PySR	C_1	C_2	C_3							
	152.630	0.287	532.886							
TPSR	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
	-238.089	0.318	0.396	-0.339	611.865	97.683	0.116	209.877	1.045	14.487
uDSR	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
	-2.909	1.692×10^7	0.000538	186300.965	0.000539	5489.483	0.0663	122.946	0.0663	0.0663
	C_{11}	C_{12}								
	122.946	122.946								

Table S11. Derived formulas from all the baseline models for plasma pressure prediction

Methods	Formulas
AIF	$\arcsin\left(\frac{C_1}{r^{3/2}}\right)$
BSR	$\frac{C_1}{r(\theta + C_2) + r}$
E2E	$C_1 \left(C_2 - \frac{1}{(C_3\theta + C_4)(C_5r + C_6\sin(C_7\theta + C_8) + C_9)} \right)^2 + C_{10}$
GP-GOMEA	$\frac{C_1r + C_2}{r(C_3r + \log(\theta + C_4))}$
GPLearn	$\frac{C_1}{r^{5/2}} \sqrt{\cos\left(C_2 \log\left(C_3 \sqrt{C_4 + \frac{2}{r} \cdot \frac{1}{r^{5/2}}}\right)\right)}$
KAN	$C_1 + C_2 \exp\left(C_3(1 - C_4r)^2\right)$
NeSymReS	$\frac{C_1(C_2r - \sin(\theta + C_3) + C_4)}{r^2}$
Operon	$C_1 + C_2 \left(C_3r - \sin\left(\frac{C_4r}{\sqrt{(C_5\theta + 1)^2 + C_6}}\right) - C_7 \right) \exp(C_8 \exp(C_9r))$ $/(\sqrt{(C_{10}r + \exp(C_{11}r) - C_{12}\sin(\sin(C_{13}\theta - C_{14})))^2 + C_{15}}$ $\times \sqrt{\left(-r + C_{16}r \frac{C_{17}\theta + \cos(C_{18}\theta + C_{19}) + C_{20}}{\sqrt{1 + \exp(C_{21}r)}\sqrt{(C_{22}r - 1)^2 + C_{23}}}\right)^2 + C_{24}})$
ParFam	$C_1 \exp\left(\frac{C_2r + C_3\theta + C_4}{C_5r^2 + C_6r(\theta + C_7) + C_8r + C_9\theta + C_{10}(C_{11}\theta + 1)^2 - C_{12}}\right)$ $+ C_{13}r + C_{14}\theta + C_{15}$
PhySR	$C_1 \exp(C_2r(C_3r + \theta + C_4)) + C_5$
PySR	$\frac{C_1 \sqrt{\exp(C_2 \sin(C_3\theta + C_4)/r)}}{(r + C_5)^2}$
TPSR	$C_1 \left(1 + \frac{C_2}{C_3r + C_4 \sin(C_5r + C_6\theta + C_7) + C_8} \right)^2 - C_9$
uDSR	$(C_1r^3 + C_2r^2(\theta + C_3) + C_4r(C_5\theta + C_6)^2 + C_7r(\theta + C_8)$ $+ C_9\theta - C_{10}(C_{11}\theta + C_{12})^3 - C_{13}(C_{14}\theta + C_{15})^2 + C_{16})$ $/((C_{17}r + C_{18}(\theta + C_{19})\sin(C_{20}r - \sin(\exp(C_{21}/r) - \cos(C_{22}r\sin(\sin(C_{23}r)))))) / r)$ $(C_{24}r + \sin(\theta + C_{25}))(\sin(\exp(\sin(\theta + C_{26}))) + \sin(\sin(\theta + C_{27}))))$

Table S12. Constants of derived formulas from all the baseline models for plasma pressure prediction

Methods	Constants									
AIF	C_1 30.049									
BSR	C_1 8.000	C_2 1.571								
E2E	C_1 1653.845	C_2 0.0124	C_3 −0.457	C_4 2.143	C_5 8.662	C_6 0.447	C_7 9.246	C_8 6.952	C_9 18.695	C_{10} 0.107
GP-GOMEA	C_1 8.000	C_2 2.928	C_3 76.334	C_4 9.272	C_5 1.571					
GPLearn	C_1 471.275	C_2 3.202	C_3 622.772	C_4 0.000243						
KAN	C_1 0.294	C_2 2.853	C_3 −5.956	C_4 0.115						
NeSymReS	C_1 64.000	C_2 0.125	C_3 1.571	C_4 0.807						
Operon	C_1 0.364	C_2 0.000999	C_3 1.976	C_4 14.888	C_5 0.637	C_6 0.369	C_7 3.534	C_8 2.653	C_9 −0.122	C_{10} 0.120
	C_{11} 0.864	C_{12} 0.117	C_{13} 0.184	C_{14} 1.357	C_{15} −0.278	C_{16} 0.264	C_{17} 0.0138	C_{18} 0.000484	C_{19} −0.207	C_{20} 0.457
	C_{21} 0.110	C_{22} 1.049	C_{23} 3.290	C_{24} 0.0121						
ParFam	C_1 −0.0205	C_2 0.0360	C_3 0.00600	C_4 −0.406	C_5 0.697	C_6 0.321	C_7 −0.0147	C_8 0.0116	C_9 1.571	C_{10} 0.0328
	C_{11} 0.0180	C_{12} 0.0173	C_{13} 0.637	C_{14} 0.154	C_{15} 0.721					
PhySR	C_1 0.0292	C_2 0.125	C_3 −4.278	C_4 29.639	C_5 0.419					
PySR C_1	C_2 135.203	C_3 −8.000	C_4 1.012	C_5 1.590	C_6 0.005					
TPSR	C_1 0.0844	C_2 0.0131	C_3 0.000654	C_4 0.000624	C_5 −0.286	C_6 2.423	C_7 5.756	C_8 0.000878	C_9 0.0748	
uDSR	C_1 −0.00293	C_2 0.0121	C_3 1.571	C_4 0.182	C_5 0.496	C_6 0.637	C_7 1.207	C_8 1.571	C_9 2.995	C_{10} 15.680
	C_{11} 1.213	C_{12} 0.637	C_{13} 5.280	C_{14} 0.637	C_{15} 39.989	C_{16} 0.125	C_{17} 8.000	C_{18} 1.571	C_{19} 0.250	C_{20} 4.000
	C_{21} 0.125	C_{22} 0.125	C_{23} 0.125	C_{24} 1.571	C_{25} 1.571	C_{26} 1.571				

Table S13. Derived formulas from all the baseline models for differential rotation prediction

Methods	Formulas
AIF	$C_1 + \frac{1}{\sqrt{\exp(\cos(\exp(\cos(\theta))))}}$
BSR	$C_1 (\exp(\cos(\theta)) + \sin(\sin(C_2\theta + C_3 \cos(\cos(\theta))) + C_4))^3 + C_5$
E2E	$C_1 - C_2 \left(-\theta + C_3 + \frac{C_4}{C_5\theta - C_6} \right)^2$
GP-GOMEA	$\log((\theta + C_1)(\theta + C_2)) + C_3$
GPLearn	$\cos(\theta) + C_1$
KAN	$C_1 + C_2 (\theta + C_3)^2$
NeSymReS	$\sin(C_1\theta + C_2) + C_3$
Operon	$C_1 \left(\tan \left(\frac{C_2 \sin(\sin(C_3\theta + C_4))}{\sqrt{C_5 + \log(C_6\theta + C_7)^2 / ((C_8\theta + C_9)^2 / ((-\theta - C_{10})^2 + C_{11}) + C_{12})}} \right) - C_{13} \right) \\ \sin \left(\frac{C_{14}(C_{15}\theta - C_{16})}{\sqrt{\left(\sin(\log(C_{17}\theta + C_{18})) + \cos \left(\frac{-C_{19}\theta - C_{20}}{\sqrt{\sin(C_{21}\theta + C_{22})^2 + C_{23}}} \right) \right)^2}} \right) \\ \frac{1}{\sqrt{(\theta + C_{24} \sin(C_{25}\theta + C_{26}) - C_{27} \sin(C_{28}\theta + C_{29}) + C_{30} \sin(C_{31}\theta + C_{32}) - C_{33})^2 + C_{34}}} - C_{35}$
ParFam	$C_1\theta + (C_2\theta + C_3) \exp \left(\frac{C_4\theta + C_5}{C_6\theta + C_7} \right) + C_8 (C_9\theta + 1)^2 \\ - (C_{10}\theta + C_{11}) \exp \left(\frac{C_{12}\theta + C_{13}}{C_{14}\theta + C_{15}} \right) + C_{16} \exp \left(\frac{C_{17}\theta + C_{18}}{C_{19}\theta + C_{20}} \right) \\ + C_{21} \exp \left(\frac{C_{22}\theta + C_{23}}{C_{24}\theta + C_{25}} \right) + C_{26} \exp \left(\frac{C_{27}\theta + C_{28}}{C_{29}\theta + C_{30}} - \frac{C_{31}\theta + C_{32}}{C_{33}\theta + C_{34}} \right) + C_{35}$
PhySR	$C_1 \frac{\cos(C_2\theta + C_3) + C_4}{\theta + C_5}$
PySR	$C_1 \sin(\theta + C_2) + C_3 \cos \left(\frac{C_4}{\theta + C_5} \right)^2 + \frac{C_6}{\theta + C_7} + C_8$
TPSR	$C_1 \sin((C_2\theta + C_3)(C_4\theta + C_5)) + C_6 - \frac{C_7}{\theta + C_8}$
uDSR	$\exp((C_1\theta^3 - C_2\theta^2 - C_3\theta - C_4)/(-\theta - C_5 + \exp((-\theta - C_6 + C_7 \exp(-\theta)) \\ \cdot \exp(\sin(\sin(\log(\theta + C_8)))))) * \log(\theta + C_9)/(-\theta - C_{10} + (\theta + C_{11} \exp(\theta \\ + \exp(\exp(\sin(\exp(\sin(\sin(\theta + \log(\log(\exp(C_{12} \exp(2\theta + \sin(\exp(-\sin((\theta + C_{13}) \\ \exp(\sin(\theta + \sin(\theta + C_{14} + \cos(\log(C_{15} \exp(\theta))))/\log(2\theta + C_{16})) + C_{17})))))))))) + C_{18})) \\ /(\theta + C_{19})))))) + C_{20})/(\theta + C_{21}))))$

Table S14. Constants of derived formulas from all the baseline models for differential rotation prediction

Methods	Constants									
AIF	C_1 1.331									
BSR	C_1 2.290	C_2 0.662	C_3 0.662	C_4 10.323	C_5 0.379					
E2E	C_1 2.901	C_2 0.488	C_3 0.00533	C_4 0.786	C_5 −56.308	C_6 375.424				
GP-GOMEA	C_1 1.638	C_2 1.571	C_3 1.966							
GPLearn	C_1 1.934									
KAN	C_1 2.910	C_2 0.492	C_3 0.00830							
NeSymReS	C_1 1.058	C_2 1.568	C_3 1.912							
Operon	C_1 1.000	C_2 0.244	C_3 0.268	C_4 0.421	C_5 0.0595	C_6 6.292	C_7 9.883	C_8 0.665	C_9 1.000	C_{10} 0.596
	C_{11} 3.064	C_{12} 0.000968	C_{13} 2.976	C_{14} 0.158	C_{15} 0.575	C_{16} 1.558	C_{17} 73.329	C_{18} 115.185	C_{19} −0.632	C_{20} 8.610
	C_{21} 0.707	C_{22} 1.111	C_{23} 1.000	C_{24} 0.158	C_{25} 7.637	C_{26} 12.000	C_{27} 0.158	C_{28} 37.297	C_{29} 58.586	C_{30} 0.158
	C_{31} 103.750	C_{32} 162.970	C_{33} 0.651	C_{34} 0.0250	C_{35} 0.000141					
ParFam	C_1 1.579	C_2 0.316	C_3 0.496	C_4 −1.328	C_5 0.973	C_6 0.516	C_7 1.667	C_8 0.967	C_9 0.637	C_{10} 2.264
	C_{11} 3.556	C_{12} −1.066	C_{13} 3.595	C_{14} 0.923	C_{15} 1.065	C_{16} 0.273	C_{17} −1.328	C_{18} 0.973	C_{19} 0.516	C_{20} 1.667
	C_{21} 0.291	C_{22} −1.066	C_{23} 3.595	C_{24} 0.923	C_{25} 1.065	C_{26} 1.362	C_{27} −1.328	C_{28} 0.973	C_{29} 0.516	C_{30} 1.667
	C_{31} 1.066	C_{32} 3.595	C_{33} 0.923	C_{34} 1.065	C_{35} 3.526					
PhySR	C_1 3.134	C_2 1.063	C_3 17.781	C_4 0.977	C_5 1.571					
PySR	C_1 1.104	C_2 1.571	C_3 1.104	C_4 0.059	C_5 1.571	C_6 0.704	C_7 0.00351	C_8 1.177		
TPSR	C_1 −2.938	C_2 8.593	C_3 0.147	C_4 −0.0731	C_5 0.183	C_6 0.0289	C_7 0.000181	C_8 1.147		
uDSR	C_1 0.208	C_2 0.295	C_3 −1.082	C_4 −1.678	C_5 1.571	C_6 1.571	C_7 0.208	C_8 1.571	C_9 1.571	C_{10} 1.571
	C_{11} 4.810	C_{12} 23.141	C_{13} 1.571	C_{14} 1.571	C_{15} 4.810	C_{16} 3.142	C_{17} 1.571	C_{18} 1.571	C_{19} 1.571	C_{20} 1.571
	C_{21} 1.571									

Table S15. Derived formulas from all the baseline models for the prediction of contribution functions

Methods	Formulas
AIF	$C_1 \sqrt{-\frac{1}{\pi - \exp(\exp(C_2 T)/\pi)}}$
BSR	$C_1 T$
E2E	$(C_1 + \frac{C_2}{C_3 Ne + C_4})(C_5 + C_6 \exp(C_7(C_8 T + 1 - \frac{C_9}{C_{10} Ne + C_{11}})^2))$
GP-GOMEA	$-\frac{T \log(\log(C_1 T))}{C_2 T^2 + C_3 T + C_4}$
GPLearn	$\tan(\cos(\sqrt{C_1 T - \frac{C_2}{T^2}}))$
KAN	$C_1 \arctan(C_2 Ne + C_3) + C_4 + C_5 \exp(C_6(1 - C_7 T)^2)$
NeSymReS	$C_1 \exp(C_2 T^2 + C_3 T + C_4 Ne)$
Operon	$C_1(C_2 + C_3(C_4 T - C_5 Ne/\sqrt{Ne^2 + C_6} + \sin(C_7 T))/\sqrt{(1 - C_8 T)^2 + C_9})$ $\cdot (C_{10} T + \frac{C_{11} T}{\sqrt{Ne^2 + C_{12}} \sqrt{C_{13}(C_{14} \frac{(\frac{C_{15} Ne}{Ne^2 + C_{17}} - \cos(C_{18} T))}{\sqrt{\frac{C_{16} T^2}{Ne^2 + C_{17}} + 1} - 1)^2 + 1}}})$ $\cdot \frac{1}{\sqrt{C_{22} \exp(C_{23} T) + 1}} - C_{24}$
ParFam	$(C_1 T^2 + C_2 T Ne + C_3 T \log(\frac{C_4 T + C_5 Ne + C_6}{C_7 T^2 + C_8 T Ne + C_9 T - C_{10} Ne + C_{11}} + C_{12}))$ $- C_{13} T + C_{14} Ne \log(\frac{C_{15} T + C_{16} Ne + C_{17}}{C_{18} T^2 + C_{19} T Ne + C_{20} T - C_{21} Ne + C_{22}} + C_{23})$ $- C_{24} Ne + C_{25} \log(\frac{C_{26} T + C_{27} Ne + C_{28}}{C_{29} T^2 + C_{30} T Ne + C_{31} T - C_{32} Ne + C_{33}} + C_{34}) - C_{35})$ $/(C_{36} T + C_{37} Ne - C_{38} \log(\frac{C_{39} T + C_{40} Ne + C_{41}}{C_{42} T^2 + C_{43} T Ne + C_{44} T - C_{45} Ne + C_{46}} + C_{47}) + C_{48})$
PhySR	$C_1 T \sin(C_2 \exp(C_3 T^2 + C_4 T))$
PySR	$C_1 \cos(\sin(\log(C_2 T) + C_3))^{18} \tan(\cos(\tan(C_4 \exp(C_5 \sqrt{Ne}))))$
TPSR	$(C_1 + C_2 \exp(C_3(C_4 T + 1 - \frac{C_5}{C_6 T + C_7} - \frac{C_8}{Ne})^2))(C_9 T + C_{10})$
uDSR	$\exp((C_1 T^3 + C_2 T^2 Ne - C_3 T^2 - C_4 T Ne^2 + C_5 T Ne + C_6 T + C_7 Ne^3$ $+ C_8 Ne^2 - C_9 Ne + \log(C_{10} Ne) - C_{11})/(C_{12} T + \cos(\sin(C_{13} T))))$

Table S16. Constants of derived formulas from all the baseline models for the prediction of contribution functions

Methods	Constants									
AIF	C_1 0.000689	C_2 10^{-6}								
BSR	C_1 2×10^{-6}									
E2E	C_1 −0.106	C_2 0.0413	C_3 -1.45×10^{10}	C_4 0.426	C_5 −0.0328	C_6 15.566	C_7 −9.861	C_8 −9.235	C_9 0.197	C_{10} 2.86×10^{11}
	C_{11} 26.889									
GP-GOMEA	C_1 10^{-6}	C_2 10^{-6}	C_3 −1.188	C_4 1.85×10^6						
GPLearn	C_1 10^{-6}	C_2 10^{12}								
KAN	C_1 0.120	C_2 2.2×10^{10}	C_3 −0.690	C_4 −0.180	C_5 1.260	C_6 −6.230	C_7 9.2×10^{-7}			
NeSymReS	C_1 0.215	C_2 -10^{-12}	C_3 2.17×10^{-6}	C_4 2.52×10^8						
Operon	C_1 1.00×10^{-4}	C_2 0.0106	C_3 0.447	C_4 -2.19×10^{-7}	C_5 0.936	C_6 3.27×10^{-24}	C_7 3.30×10^{-6}	C_8 8.84×10^{-7}	C_9 0.200	C_{10} -2.80×10^{-6}
	C_{11} 1.34×10^{-15}	C_{12} 1.84×10^{-19}	C_{13} 0.268	C_{14} 0.155	C_{15} -9.78×10^{11}	C_{16} 7.80×10^{-39}	C_{17} 3.44×10^{-26}	C_{18} 3.91×10^{-6}	C_{19} -5.32×10^{-7}	C_{20} 7.35×10^{-22}
	C_{21} 0.00647	C_{22} 0.000947	C_{23} 8.33×10^{-6}	C_{24} 7.20×10^{-5}						
ParFam	C_1 3.50×10^{-14}	C_2 50.0	C_3 3.00×10^{-9}	C_4 -4.35×10^{-7}	C_5 3.37×10^9	C_6 5.11	C_7 2.18×10^{-13}	C_8 130.0	C_9 6.96×10^{-7}	C_{10} 2.20×10^8
	C_{11} 0.684	C_{12} 1.00×10^{-6}	C_{13} 1.29×10^{-7}	C_{14} 8.00×10^7	C_{15} -4.35×10^{-7}	C_{16} 3.37×10^9	C_{17} 5.11	C_{18} 2.18×10^{-13}	C_{19} 130.0	C_{20} 6.96×10^{-7}
	C_{21} 2.20×10^8	C_{22} 0.684	C_{23} 1.00×10^{-6}	C_{24} 3.10×10^8	C_{25} 0.067	C_{26} -4.35×10^{-7}	C_{27} 3.37×10^9	C_{28} 5.11	C_{29} 2.18×10^{-13}	C_{30} 130.0
	C_{31} 6.96×10^{-7}	C_{32} 2.20×10^8	C_{33} 0.684	C_{34} 1.00×10^{-6}	C_{35} 0.119	C_{36} 4.97×10^{-7}	C_{37} 5×10^8	C_{38} 0.402	C_{39} -4.35×10^{-7}	C_{40} 3.37×10^9
	C_{41} 5.11	C_{42} 2.18×10^{-13}	C_{43} 130.0	C_{44} 6.96×10^{-7}	C_{45} 2.20×10^8	C_{46} 0.684	C_{47} 10^{-6}	C_{48} 0.767		
PhySR	C_1 1.46×10^{-6}	C_2 1.752	C_3 -2.58×10^{-12}	C_4 2.44×10^{-6}						
PySR	C_1 1.023	C_2 10^{-6}	C_3 0.0369	C_4 1.122	C_5 -10^5					
TPSR	C_1 −0.701	C_2 35429.186	C_3 −7.786	C_4 -7.59×10^{-7}	C_5 0.821	C_6 5.34×10^{-7}	C_7 0.189	C_8 1.15×10^{-12}	C_9 7.46×10^{-9}	C_{10} −0.0301
uDSR	C_1 2.18×10^{-18}	C_2 4.27×10^{-5}	C_3 2.23×10^{-11}	C_4 1.78×10^{12}	C_5 2158.980	C_6 3.94×10^{-5}	C_7 1.21×10^{27}	C_8 2.61×10^{18}	C_9 6.47×10^9	C_{10} 10^{10}
	C_{11} 18.467	C_{12} 10^{-6}	C_{12} 10^{-6}							

Table S17. Derived formulas from all the baseline models for the prediction of lunar tide signal

Methods	Formulas
AIF	-
BSR	$\exp(\cos(\cos(C_1 \cdot LLT^3 \cdot MLT^3)))$
E2E	$C_1(1 - \frac{1}{(C_2 - C_3 \cdot L_{Shell})(C_4 \cdot LLT + C_5 \cdot MLT + C_6(C_7 + \frac{1}{1.0 - \frac{C_8}{C_9 \cdot x_4 + C_{10}})})^2 - C_{11}})^3 + C_{12}$
GP-GOMEA	$\frac{C_1}{L_{Shell} + C_2 \cdot MLT + C_3}$
GPLearn	$\sin(L_{Shell}) + \tan(\sin(L_{Shell}))^{\frac{1}{4}}$
KAN	$C_1 \cdot \arctan(C_2 \cdot L_{Shell} + C_3) + C_4 \cdot \sin(C_5 \cdot MLT + C_6) + C_7 + C_8 \cdot \exp(C_9 \cdot (1 - C_{10} \cdot LLT)^2)$
NeSymReS	$\frac{C_1 \cdot \tan(C_2 \cdot LLT \cdot L_{Shell} \cdot MLT)^2}{L_{Shell} + C_3}$
Operon	$C_1(-C_2 \cdot L_{Shell}(C_3 \cdot L_{Shell} - C_4) \cdot \cos(C_5 \cdot MLT) / \sqrt{(C_6 \cdot L_{Shell} - C_7)^2 + C_8} + C_9 \cdot L_{Shell} \cdot \exp(-C_{10} \cdot L_{Shell} + C_{11} \cdot \cos(C_{12} \cdot MLT) / \sqrt{(1 - C_{13} \cdot L_{Shell})^2 + C_{14}}))$ $((-C_{15} \cdot LLT + C_{16} \cdot L_{Shell}) \cdot \cos(C_{17} \cdot MLT) + C_{18} + \frac{C_{19}}{\sqrt{(1 - C_{20} \cdot L_{Shell})^2 + C_{21}}})$ $/ \sqrt{(1 - C_{22} \cdot L_{Shell})^2 + C_{23}} + C_{24}$
ParFam	$C_1 - C_2 \sin\left(\frac{C_3 \cdot L_{Shell}^2 + C_4 \cdot L_{Shell} \cdot MLT - C_5 \cdot L_{Shell} - C_6 \cdot MLT^2 + C_7 \cdot MLT - C_8}{C_9 \cdot L_{Shell}^2 + C_{10} \cdot L_{Shell} + C_{11} \cdot MLT^2 - C_{12} \cdot MLT - C_{13}}\right)$
PhySR	$\frac{1}{(\frac{-L_{Shell}^2}{C_1 \cdot L_{Shell} + C_2}) + C_3}$
PySR	$C_1(C_2 \cdot \tan(\cos(C_3 \cdot MLT + \frac{C_4}{MLT^3})) - 1)^2 / L_{Shell}^2$
TPSR	$(x_4 - C_1)(C_2 \cdot LLT - C_3 \cdot L_{Shell} + C_4 \cdot MLT + C_5 + \frac{C_6}{C_7 + C_8 \cdot \exp(C_9 \cdot MLT)})$
uDSR	$(C_1 \cdot LLT^3 + C_2 \cdot LLT^2 \cdot L_{Shell} + C_3 \cdot LLT^2 \cdot MLT - C_4 \cdot LLT^2 - C_5 \cdot LLT \cdot L_{Shell}^2$ $- C_6 \cdot LLT \cdot L_{Shell} \cdot MLT + C_7 \cdot LLT \cdot L_{Shell} + C_8 \cdot LLT \cdot MLT^2 - C_9 \cdot LLT \cdot MLT$ $+ C_{10} \cdot LLT + C_{11} \cdot L_{Shell}^3 + C_{12} \cdot L_{Shell}^2 \cdot MLT - C_{13} \cdot L_{Shell}^2 + C_{14} \cdot L_{Shell} \cdot MLT^2$ $- C_{15} \cdot L_{Shell} \cdot MLT + C_{16} \cdot L_{Shell} + C_{17} \cdot MLT^3 - C_{18} \cdot MLT^2 + C_{19} \cdot MLT$ $+ (C_{20} \cdot LLT + \sin(C_{21} \cdot MLT))(C_{22} \cdot MLT + \log(C_{23} \cdot MLT)) - C_{24}) / L_{Shell}^2$

Table S18. Constants of derived formulas from all the baseline models for the prediction of lunar tide signal

Methods	Constants									
AIF	-									
BSR	C_1 0.00523									
E2E	C_1 0.729	C_2 0.010	C_3 0.060	C_4 3.75	C_5 3.75	C_6 32.4	C_7 0.00333	C_8 9.00	C_9 9.00	C_{10} 0.001
	C_{11} 0.003	C_{12} 0.002								
GP-GOMEA	C_1 1.466	C_2 0.121	C_3 0.859							
GPLearn	-									
KAN	C_1 −1.68	C_2 2.00	C_3 3.53	C_4 0.54	C_5 0.496	C_6 5.06	C_7 2.64	C_8 0.08	C_9 −0.493	C_{10} 0.758
NeSymReS	C_1 3.682	C_2 0.248	C_3 1.251							
Operon	C_1 0.201	C_2 −0.0111	C_3 16.797	C_4 10.148	C_5 0.524	C_6 0.783	C_7 1.000	C_8 0.0116	C_9 1.138	C_{10} −0.271
	C_{11} 0.0300	C_{12} 0.231	C_{13} 0.482	C_{14} 2.230	C_{15} −0.113	C_{16} 0.190	C_{17} 0.230	C_{18} 3.139	C_{19} 0.321	C_{20} 0.512
	C_{21} 0.00549	C_{22} 0.858	C_{23} 0.0405	C_{24} 0.00561						
ParFam	C_1 3.569	C_2 3.365	C_3 0.565	C_4 0.00083	C_5 1.046	C_6 0.00625	C_7 0.150	C_8 0.427	C_9 0.125	C_{10} 0.678
	C_1 0.00122	C_2 0.0271	C_3 0.722							
PhySR	C_1 −1.124	C_2 0.738	C_3 2.106							
PySR	C_1 6.288	C_2 0.399	C_3 0.536	C_4 6.936						
TPSR	C_1 6.0	C_2 0.000417	C_3 0.001	C_4 0.000417	C_5 0.0109	C_6 0.001	C_7 −0.01	C_8 0.985	C_9 −3.75	
uDSR	C_1 0.000570	C_2 0.00074	C_3 8.933	C_4 0.0214	C_5 0.00104	C_6 0.00155	C_7 0.0153	C_8 0.00303	C_9 0.286	C_{10} 0.388
	C_{11} 0.107	C_{12} 0.0209	C_{13} 1.106	C_{14} 0.000383	C_{15} 0.161	C_{16} 3.919	C_{17} 0.00367	C_{18} 0.189	C_{19} 2.550	C_{20} 0.417
	C_{21} 0.417	C_{22} 0.417	C_{23} 0.417	C_{24} 3.158						

C Details settings of baseline Symbolic Regression methods

AlFeynman The code for AI Feynman is available at <https://github.com/SJ001/AI-Feynman>. The parameters are set to be the same as their examples, allowing for 14 possible operators: $\{+, *, -, /, +1, -1, \text{neg}, \text{inv}, \text{sqrt}, \pi, \sin, \cos, \ln, \exp\}$, with a maximum time for brute-force search of 30 seconds, a maximum polynomial fitting degree of 3, and 500 training epochs for the neural network. We run the model using the command `run_aifeynman("../data/", "data.txt", 30, "14ops.txt")`.

BSR The Bayesian Symbolic Regression (BSR) is available at <https://github.com/ying531/MCMC-SymReg>, which includes a Bayesian framework and an efficient Markov Chain Monte Carlo (MCMC) algorithm. We use the default parameter, allowing for a total of 50 iterations and 3 output formula trees. We run the model using the command `bsr.fit(X, y)`.

EndtoEnd The code for EndtoEnd model can be downloaded from the official repository at <https://github.com/facebookresearch/symbolicregression>. We also download the pre-trained model from <https://dl.fbaipublicfiles.com/symbolicregression/model1.pt> and use the default parameters provided in the demo scripts. The model is allowed to search for operators from the following set: $\{\text{add}, \text{sub}, \text{mul}, \text{div}, \text{abs}, \text{inv}, \text{sqrt}, \log, \exp, \sin, \arcsin, \cos, \arccos, \tan, \arctan, \text{pow2}, \text{pow3}\}$. We only allow at most 200 datapoints in a single bag to fit for each problem, same as default setting, while the remaining of data are split into at most 10 bags, each of which is fitted independently. We run the model using the command `est.fit(X, y)`.

GP-GOMEA The Gene-pool Optimal Mixing Evolutionary Algorithm for Genetic Programming (GP-GOMEA) is available at <https://github.com/marcovirgolin/gpg>. The operators are allowed to search from the following set: $\{\text{add}, \text{sub}, \text{mul}, \text{div}, \text{sqrt}, \log\}$. Other default parameters follow their default configuration, including `popsiz=64`, `batchsize=64`, `time_limit=100`. We run the model using the command `gpg.fit(X, y)`.

GPLearn The GPLearn model is also a GP-based method available at <https://github.com/trevorstephens/gplearn>. We use a function set of $\{\text{add}, \text{sub}, \text{mul}, \text{div}, \sin, \cos, \tan, \text{sqrt}, \log, \exp\}$. Other parameters follow their default configuration, which include `population_size=5000`, `generations=20`, a crossover probability $p_{\text{crossover}} = 0.7$, a subtree mutation probability $p_{\text{subtree mutation}} = 0.1$, a hoist mutation probability $p_{\text{hoist mutation}} = 0.05$, a point mutation probability $p_{\text{point mutation}} = 0.1$, and a maximum sample fraction of 0.9. We run the model using the command `est_gp.fit(X, y)`.

KAN The Kolmogorov-Arnold Networks (KAN) model is available at <https://github.com/KindXiaoming/pykan>. The KAN trains a neural network with learnable activation functions, which is then fitted to the most likely operators selected from the following set: $\{x, x^2, x^3, x^4, 1/x, 1/x^2, 1/x^3, 1/x^4, \sqrt{x}, 1/\sqrt{x}, \exp(x), \log(x), \text{abs}(x), \sin(x), \tan(x), \tanh(x), \text{sigmoid}(x), \arcsin(x), \arctan(x), \text{arctanh}(x), 0, \cosh(x), \text{gaussian}(x)\}$ to derive its symbolic representation. Proper design on KAN's architecture is crucial when addressing practical problems. To balance its computation complexity and accuracy, we train the KAN model three times for each problem, using different network widths: $(nv, 1)$, $(nv, 5, 1)$, and $(nv, 3, 3, 1)$, respectively. These widths are among the most popular choices for solving Feynman equations. Other parameters follow their default configuration, including the number of grid intervals = 5, the order of piecewise polynomial = 3. We run the model using the command `formula = learning(kan, dataset, verbose)`.

NeSymRes The Neural Symbolic Regression that Scale (NeSymReS) is available at <https://github.com/SymposiumOrganization/NeuralSymbolicRegressionThatScales>. We use the 100M pre-trained model downloaded from <https://drive.google.com/drive/folders/1LTKUX-KhoUbW-WOx-ZJ8KitxK7Nov41G?usp=sharing>. The model is allowed to search for operators from the following set: {abs, acos, add, asin, atan, cos, cosh, coth, div, exp, ln, mul, pow, sin, sinh, sqrt, tan, tanh}, same as default configuration. We run the model using the command `output = fitfunc(X, y)`.

Operon The Operon is another GP-based symbolic regression methods available at <https://github.com/heal-research/operon>. We use a function set of {add, sub, mul, aq, sin, cos, tan, log, exp}. Other parameters follow their default configuration. We run the model using the command `reg.fit(X, y)`.

ParFam The ParFam model is available at <https://github.com/Philipp238/parfam>. We use a function set of {add, sub, mul, div, sin, cos, sqrt, log, exp} according to the default configuration, along with other parameters. We run the model using the command `model.fit(X, y)`.

PhySO The Physical Symbolic Optimizer (PhiSO) is available at <https://github.com/WassimTenachi/PhySO>. We use the default config0 for our experiment for the efficiency consideration. The operators are allowed from the following set: {add, sub, mul, div, inv, n2, sqrt, neg, log, exp, sin, cos, tan}. Each problem is assigned a fixed constant "1" and three free constants, all without physical units. The physical units of the input-output variables are also provided to the model. We run the model using the command `expression, logs = physo.SR(X, y, X_units=x_units, y_units=y_units, fixed_consts=[1.], fixed_consts_units = np.zeros((1, 5)), free_consts_units = np.zeros((3, 5)), run_config = config, op_names=["mul", "add", "sub", "div", "inv", "n2", "sqrt", "neg", "exp", "log", "sin", "cos", "tan"])`.

PySR The PySR model is available at <https://github.com/MilesCranmer/PySR>. We use a function set of {add, sub, mul, div, square, cube, exp, log, sin, cos, sqrt}, with maximum iteration of 400, as specified in the default configuration. We run the model using the command `model.fit(X, y)`.

TPSR The Transformer-based Planning for Symbolic Regression (TPSR) is available at <https://github.com/deep-symbolic-mathematics/TPSR>. We use the EndToEnd model as backbone model for MCTS, hence adopting the same function set as the EndToEnd model. Other parameters are set to their default configuration. We run the model using the command `python tpsr.py -backbone_model e2e -no_seq_cache True -no_prefix_cache True`.

uDSR The unified Deep Symbolic Regression (uDSO) is available at <https://github.com/dso-org/deep-symbolic-optimization>. We use a function set of {add, sub, mul, div, sin, cos, exp, log, poly}. Other parameters follow their default configuration. We run the model using the command `model.fit(X, y)`.

D Divide-and-Conquer Algorithm

Algorithm 1: Construction of the possible σ -divisions

Input: the oracle neural network $\tilde{f}_\theta(\mathbf{x})$, the uni-variate operator σ
Output: the set of possible σ -divisions $\mathcal{B} = \{\mathcal{A}_k\}_{k=1}^s$
// estimation of inner-variable relationship
for each distinct features i and j do
 | calculate $J_{i,j}(\tilde{f}_\theta, \sigma) = \text{median}_{1 \leq k \leq N}(|\frac{\partial^2 \sigma^{-1} \circ \tilde{f}_\theta}{\partial x_i \partial x_j}(\mathbf{x}_k)|)$;
// adaptive threshold strategy
Initialize the set of possible classes of σ -separable feature pairs $\mathcal{S} = \{\}$;
Initialize the class of σ -separable features pairs $S = \{\}$;
Sort $J_{i,j}(\tilde{f}_\theta, \sigma)$ in non-decreasing order and calculate $\epsilon_0, \epsilon_1, \epsilon_2$;
for each $J_{i,j}(\tilde{f}_\theta, \sigma)$ in the sorted order do
 | $S \leftarrow S \cup \{(i, j)\}$;
 | **if $\epsilon_1 \leq J_{i,j}(\tilde{f}_\theta, \sigma) \leq \epsilon_2$ then**
 | | $\mathcal{S} \leftarrow \mathcal{S} \cup \{S\}$;
// construction of the possible σ -divisions
Initialize the set of possible σ -divisions $\mathcal{B} = \{\}$;
for each set of σ -separable features $S \in \mathcal{S}$ do
 | Initialize σ -division $\mathcal{A} = \{\{1, 2, \dots, n\}\}$;
 | **for any feature pairs $(i, j) \in S$ do**
 | **for each $A \in \mathcal{A}$ do**
 | **if $i \in A$ and $j \in A$ then**
 | | $\mathcal{A} \leftarrow (\mathcal{A} - \{A\}) \cup \{A - \{i\}, A - \{j\}\}$;
 | $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{A}\}$;
for each $\mathcal{A}_k \in \mathcal{B}$ do
 | **for each two element $A_i, A_j \in \mathcal{A}$ do**
 | **if $A_i \subseteq A_j$ then**
 | | $\mathcal{A}_k \leftarrow \mathcal{A}_k - \{A_i\}$;
Return $\mathcal{B}$;

E Proofs for the divide-and-conquer strategy

E.1 Proofs of the decomposition step

Proof of Lemma 1. By the Inverse Function Theorem, the uni-variate operator σ has an inverse σ^{-1} , such that $\sigma \circ \sigma^{-1} = \text{Id}$ and $\sigma^{-1} \circ \sigma = \text{Id}$.

The “only if” part is straightforward. Suppose two features i and j are σ -separable. By the definition of being σ -separable, we have that

$$\sigma^{-1} \circ f(\mathbf{x}) = f_1(\mathbf{x}_{-i}) + f_2(\mathbf{x}_{-j}).$$

Straightforward calculation shows that for each $\mathbf{x} \in \mathbb{R}^n$,

$$\frac{\partial^2 \sigma^{-1} \circ f(\mathbf{x})}{\partial x_i \partial x_j} = 0.$$

Now we turn to the “if” part. Suppose we have that

$$\frac{\partial^2 \sigma^{-1} \circ f(\mathbf{x})}{\partial x_i \partial x_j} = 0.$$

Integrating both sides over x_j , we get that there exists $g_2(\mathbf{x}_{-j})$ such that

$$\frac{\partial \sigma^{-1} \circ f(\mathbf{x})}{\partial x_i} = g_2(\mathbf{x}_{-j}).$$

Now integrate both sides over x_i . We have that there exists $g_1(\mathbf{x}_{-i})$ such that

$$\sigma^{-1} \circ f(\mathbf{x}) = g_1(\mathbf{x}_{-i}) + \int g_2(\mathbf{x}_{-j}) dx_i.$$

Let $f_1(\mathbf{x}_{-i}) = g_1(\mathbf{x}_{-i})$ and $f_2(\mathbf{x}_{-j}) = \int g_2(\mathbf{x}_{-j}) dx_i$, we conclude that features i and j are σ -separable. $\square$

Proof of Lemma 2. We prove the lemma by induction on the number of iterations ℓ .

Induction basis. When $\ell = 1$, the lemma reduces to the definition of being σ -separable.

Inductive step. Suppose the induction hypothesis holds for any $\ell \geq 1$. Now we consider the iteration $(\ell + 1)$. By the induction hypothesis, we can express $f(\mathbf{x})$ as

$$f(\mathbf{x}) = \sigma \left(\sum_{k=1}^{m_\ell} f_k^{(\ell)}(\mathbf{x}_{A_k^\ell}) \right). \quad (8)$$

Suppose we select feature pair (i, j) that is σ -separable at the $(\ell + 1)$ -th iteration. By the definition of being σ -separable, we can also express $f(\mathbf{x})$ as

$$f(\mathbf{x}) = \sigma (f_1(\mathbf{x}_{-i}) + f_2(\mathbf{x}_{-j})), \quad (9)$$

where $\mathbf{x}_{-i}$ is the $(n - 1)$ -dimensional vector obtained by removing x_i from $\mathbf{x}$. We further define $\mathbf{x}_{(-i, -j)}$ as the $(n - 2)$ -dimensional vector obtained by removing x_i and x_j from $\mathbf{x}$.

Combining Eqs. (8,9) and using the fact that σ^{-1} exists, for any fixed α and β (e.g., $\alpha = \beta = 0$), we have

$$f_1(\mathbf{x}_{-i}) + f_2(\mathbf{x}_{(-i,-j)}, x_i = \alpha) = \sum_{k=1}^{m_\ell} f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{i\}}, x_i = \alpha)_{A_k^\ell} \right), \quad (10)$$

$$f_1(\mathbf{x}_{(-i,-j)}, x_j = \beta) + f_2(\mathbf{x}_{-j}) = \sum_{k=1}^{m_\ell} f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{j\}}, x_j = \beta)_{A_k^\ell} \right), \quad (11)$$

$$f_1(\mathbf{x}_{(-i,-j)}, x_j = \beta) + f_2(\mathbf{x}_{(-i,-j)}, x_i = \alpha) = \sum_{k=1}^{m_\ell} f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{i,j\}}, x_i = \alpha, x_j = \beta)_{A_k^\ell} \right). \quad (12)$$

Combining Eqs. (9,10,11,12), we further have

$$\begin{aligned} f(\mathbf{x}) &= \sigma(f_1(\mathbf{x}_{-i}) + f_2(\mathbf{x}_{-j})) \\ &= \sigma \left(\sum_{k=1}^{m_\ell} f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{i\}}, x_i = \alpha)_{A_k^\ell} \right) + f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{j\}}, x_j = \beta)_{A_k^\ell} \right) \right. \\ &\quad \left. - f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{i,j\}}, x_i = \alpha, x_j = \beta)_{A_k^\ell} \right) \right). \end{aligned} \quad (13)$$

Note that $f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{i\}}, x_i = \alpha)_{A_k^\ell} \right)$ is a function of $\mathbf{x}_{A_k^\ell - \{i\}}$ and $f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{j\}}, x_j = \beta)_{A_k^\ell} \right) - f_k^{(\ell)} \left((\mathbf{x}_{A_k^\ell - \{i,j\}}, x_i = \alpha, x_j = \beta)_{A_k^\ell} \right)$ is a function of $\mathbf{x}_{A_k^\ell - \{j\}}$. Therefore, Eq. (13) implies that $\{A_k^\ell - \{i\}, A_k^\ell - \{j\}\}$, after removing multiplicities in the class, forms a σ -division, which proves the lemma for iteration $(\ell + 1)$. $\square$

E.2 Proof of the aggregation theorem

Before proving Theorem 3, we first prove two useful lemmas as follows.

Lemma 4. *Suppose the uni-variate operator $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is strictly monotonic. If $\{A_i\}_{i=1}^m$ is a σ -partition of the target formula $f : \mathbb{R}^n \rightarrow \mathbb{R}$, i.e., f can be expressed as $f(\mathbf{x}) = \sigma(f_1(\mathbf{x}_{A_1}) + \dots + f_m(\mathbf{x}_{A_m}))$. Then for any $\chi \in \mathbb{R}^n$, and each $i \in \{1, 2, 3, \dots, m\}$, we have that*

$$f_i(\chi_{A_i}) = \sum_{\emptyset \subsetneq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}).$$

Proof. For any $\chi \in \mathbb{R}^n$, and each $i \in \{1, 2, 3, \dots, m\}$, we have that

$$\begin{aligned} &\sum_{\emptyset \subsetneq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \\ &= \sum_{\{i\} \subseteq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \\ &\quad + \sum_{\emptyset \subsetneq I \subseteq [m] - \{i\}} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}). \end{aligned}$$

Also note that

$$\begin{aligned}
& \sum_{\{i\} \subseteq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \\
&= f_i(\chi_{A_i}) + \sum_{\{i\} \subsetneq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \\
&= f_i(\chi_{A_i}) + \sum_{\{i\} \subsetneq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_{I-\{i\}}} = \chi_{A_i \cap A_{I-\{i\}}}, \mathbf{x}_{A_i - A_{I-\{i\}}} = \mathbf{z}_{A_i - A_{I-\{i\}}}) \\
&= f_i(\chi_{A_i}) - \sum_{\emptyset \subsetneq I \subseteq [m] - \{i\}} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}).
\end{aligned}$$

Altogether, we prove the desired equality . $\square$

Lemma 5. *Under the condition of Theorem 3, for each $I \subseteq [m]$ and every $k \in I$, we have that*

$$g_k(\mathbf{x}_{A_I}, \mathbf{x}_{A_k - A_I} = \mathbf{z}_{A_k - A_I}) = \sigma \left(\sum_{i=1}^m f_i(\mathbf{x}_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \right).$$

Proof. This lemma follows directly from the definition of $g_k(\mathbf{x}_{A_k})$:

$$g_k(\mathbf{x}_{A_k}) = f(\mathbf{x}_{A_k}, \overline{\mathbf{x}_{A_k}} = \overline{\mathbf{z}_{A_k}}) = \sigma \left(\sum_{i=1}^m f_i(\mathbf{x}_{A_i \cap A_k}, \mathbf{x}_{A_i - A_k} = \mathbf{z}_{A_i - A_k}) \right).$$

Thus, noting that $A_I \subseteq A_k$, we get

$$g_k(\mathbf{x}_{A_I}, \mathbf{x}_{A_k - A_I} = \mathbf{z}_{A_k - A_I}) = \sigma \left(\sum_{i=1}^m f_i(\mathbf{x}_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \right).$$

$\square$

Now, we are ready to prove Theorem 3.

Proof of Theorem 3. By Lemma 4, for any $\chi \in \mathbb{R}^n$, and each $i \in \{1, 2, 3, \dots, m\}$, we have

$$f_i(\chi_{A_i}) = \sum_{\emptyset \subsetneq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}).$$

Summing the above equality over $1 \leq i \leq m$, we have

$$\begin{aligned}
\sum_{i=1}^m f_i(\chi_{A_i}) &= \sum_{i=1}^m \sum_{\emptyset \subsetneq I \subseteq [m]} (-1)^{|I|-1} f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \\
&= \sum_{\emptyset \subsetneq I \subseteq [m]} \frac{(-1)^{|I|-1}}{|I|} \sum_{t \in I} \sum_{i=1}^m f_i(\mathbf{x}_{A_i \cap A_I} = \chi_{A_i \cap A_I}, \mathbf{x}_{A_i - A_I} = \mathbf{z}_{A_i - A_I}) \\
&= \sum_{\emptyset \subsetneq I \subseteq [m]} \frac{(-1)^{|I|-1}}{|I|} \sum_{t \in I} \sigma^{-1} \circ g_t(\mathbf{x}_{A_I} = \chi_{A_I}, \mathbf{x}_{A_t - A_I} = \mathbf{z}_{A_t - A_I}),
\end{aligned}$$

where the last equality is due to Lemma 5. Applying σ to both sides of the equality above, we prove the theorem. $\square$