

Generative Discovery of Partial Differential Equations by Learning from Math Handbooks

Corresponding Author: Professor Dongxiao Zhang

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The authors propose a novel approach for the data-driven discovery of PDEs based on the use of pre-trained transformers. I have found the use of a "generation-evaluation-optimization" loop very interesting.

Still, while the idea is intriguing and potentially very powerful, I have some concerns that, in my view, currently prevent the paper from being suitable for immediate publication.

1. The comparison with existing techniques is somewhat limited and requires some clarifications. For instance, in Figure 3:

1.1 Many techniques are marked as "not applicable", but the motivation is not clearly explained. For instance, in W-SINDy, I suspect the non applicability is associated with the PDEs being framed in complex non-squared domains, which makes it difficult to rely upon algorithms such as the FFT. Still, this is not clearly stated anywhere;

1.2 What is the precise meaning of "Discovered with redundant terms" and "Incorrectly discovered"? How many terms need to be captured for a model to fall in a category or the other? Does the accuracy on the coefficients matter in this classification?

2. Related to my previous questions: the authors use a NN-surrogate to circumvent the problem of having to estimate derivatives from noisy data. This strategy is very common in the literature of equation discovery, and has been integrated, e.g., with methods like SINDy: see the recent work by Meng and Qiu on the integration of Gaussian Processes within the SINDy framework (2025), or the paper by Chen, Liu and Sun (2021, published on this same journal) on the use of PINNs+SINDy. I believe the authors should include a comparison with another methods from the SINDy family, aside from W-SINDy (which is far too limited in the considered framework), to better motivate the improvements given by their proposed strategy. I suspect, in fact, that combining an NN-surrogate (as done here) with plain SINDy could provide good results (provided that the penalization coefficients are chosen accordingly and the library is broad enough).

3. The authors claim that their method is knowledge-based, but:

3.1 The generation process is constrained to ensure mathematically valid expressions, not physically consistent ones (although the authors state that "This enables [...] filtering out physically implausible expressions");

3.2 EqGPT only "sees" the data through the lens of the generation-evaluation-optimization loop, and cannot (as of now) integrate external knowledge of a given system. Considering the extreme flexibility of transformers in handling sequences, symbols and sentences, an exciting direction of future research could be to construct EqGPT so that, aside from the data, one could condition the generation process with actual words (e.g. "fluid", "turbulence", "force", "stable", "conservation", etc.). This would make the model truly domain-adaptive and would allow EqGPT to stand out. Currently, this is not possible, reason for which, regardless of the data at hand, EqGPT always begins its generation process with the "start" token alone.

These final remarks are not meant to urge the authors to develop a more general or powerful version of EqGPT, but rather to encourage them to moderate some of their claims.

(Remarks on code availability)

Reviewer #2

(Remarks to the Author)

The paper presents a knowledge-guided generative framework for discovering partial differential equations (PDEs) from data, introducing a model called EqGPT. This model integrates structured domain knowledge - drawn from mathematical handbooks—with generative modeling techniques, aiming to improve both the scope and efficiency of data-driven PDE discovery.

In particular, PDEs are encoded as sequences of tokens (representing operators and physical terms), thereby transforming equations into sentence-like structures that are amenable to sequence modeling. EqGPT is pretrained on 221 canonical PDEs extracted from Debnath's Nonlinear Partial Differential Equations, and augmented via permutation and reordering to produce 7,072 training instances. The framework operates through a "generation—evaluation—optimization" loop, where EqGPT proposes candidate PDEs, evaluates their data fidelity using a reward function, and fine-tunes its generative process based on top-performing equations.

The framework is tested on rediscovering canonical PDEs, including the Poisson equation in 2D and 3D irregular domains, and is compared with several baseline algorithms. Finally, the model is applied to real experimental data, where it successfully discovers a governing PDE that aligns well with observed phenomena - a particularly compelling result that demonstrates the method's practical utility and is highly appreciated by the reviewer.

Major Concerns: Nature and Utility of the Generative Model

From the description of the generation process (lines 174–181), it appears that EqGPT is effectively sampling combinations of terms to construct PDE forms, which are subsequently evaluated using a linear regression model. For instance, using the Fig. 1c as an example, if a PDE form $u_{u_x} + u_t + u_{\{xxx\}}$ is generated by the EqGPT, then the discovered PDE is a linear combination of u_{u_x} , u_t , $u_{\{xxx\}}$, the equation is treated as a linear combination of those terms, with coefficients determined via least squares fitting.

This raises an important conceptual question: Is EqGPT essentially performing stochastic enumeration? It seems to me that the method is like an exhaustive enumeration that as many as possible PDE forms are enumerated and then tested to find best ones. Since the space of possible token sequences is finite (given a finite "vocabularies" and training instances), the sampling process can be viewed as drawing from a discrete distribution. In this case, why is a generative model necessary, especially one based on autoregressive language modeling typically suited for output spaces of infinite dimensionality?

Moreover, after selecting the top 10 equations based on reward, these are used to fine-tune EqGPT. However, the purpose of this fine-tuning remains unclear. If the top equations are already selected based on their evaluation scores, why adjust the model to sample them more often? This could reinforce generation of those specific forms, but the next round of evaluation will still depend on reward ranking rather than generation probability. As such, the value of model fine-tuning within this loop could be more explicitly justified.

Minor Concerns and Suggestions:

1. The definition of the reward function (Equation 1) is unclear and requires improvement. I recommend rewriting this section to more clearly and precisely define all variables involved. For example, the definition of θ_{LHS} is ambiguous - it is described as "the negative of the first term of the structure," yet the first term refers to a symbolic PDE term, not a numerical value. It is unclear how this symbolic expression is converted into a numerical vector or quantity for use in the reward calculation. A more rigorous explanation, including the mathematical meaning of each component and how these are derived from the PDE structure and surrogate model output, would greatly improve clarity.

2. Explicit Vocabulary List

The paper mentions 56 tokens used for constructing PDEs, but these are not explicitly listed. Including a table or appendix with the full list would greatly improve transparency and reproducibility. Currently, the reader must infer this set from Fig. 1a.

3. Handling of Non-Permutational Equivalences

The paper notes that "the semantic meaning of a PDE remains invariant under permutation of additive terms." However, many mathematical equivalences are more subtle. For instance, $(u_{u_x})_x$ is considered as a vocabulary, and it is in fact equivalent to $u_x * u_x + u * u_{\{xx\}}$. Another example would be $(u^2)_x$ is equivalent to $2 u * u_x$. These are common in conservation laws and other physically meaningful transformations. How the framework handles - or fails to handle - such equivalences deserves further discussion.

4. Dimensional Consistency of Generated Terms

In practice, PDEs must match the dimensionality of the physical problem. For instance, in a 1D time-dependent system, it would be incorrect to include terms like u_y or u_{xy} . It is unclear how EqGPT enforces this constraint during generation. Clarifying how dimensional appropriateness is maintained (e.g., through token masking or probability constraints) would strengthen the methodology section.

5. Relation to PINN-Symbolic Regression Hybrids

There is a growing body of literature combining physics-informed neural networks (PINNs) with symbolic regression techniques. It would be helpful for the authors to briefly differentiate their approach from such methods. For example: How does the generative-symbolic mechanism of EqGPT compare in flexibility, robustness, or interpretability to hybrid PINN-symbolic strategies?

(Remarks on code availability)

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The authors have thoroughly addressed all my remarks. I have no additional comments.

(Remarks on code availability)

Reviewer #2

(Remarks to the Author)

Thank you to the authors for thoroughly addressing my comments and concerns; I am satisfied that all have been properly resolved.

(Remarks on code availability)

Open Access This Peer Review File is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

In cases where reviewers are anonymous, credit should be given to 'Anonymous Referee' and the source.

The images or other third party material in this Peer Review File are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons

license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

Response to Reviewers

We would like to sincerely express our deep gratitude to the editors and reviewers for their insightful comments and suggestions on the manuscript, which are fair, encouraging, and constructive. After carefully studying the comments, we have made corresponding changes in the revised manuscript. The specific responses to the comments are listed below. In the revised manuscript, the text in **red** is the revision for Reviewer #1, **blue** for Reviewer #2. In this response, the corresponding revisions in the manuscript are quoted within the gray box.

Contents

Response to Reviewer #1	1
Response to Reviewer #2.....	10

Response to Reviewer #1

Reviewer Point P 1.1—*The authors propose a novel approach for the data-driven discovery of PDEs based on the use of pre-trained transformers. I have found the use of a “generation-evaluation-optimization” loop very interesting. Still, while the idea is intriguing and potentially very powerful, I have some concerns that, in my view, currently prevent the paper from being suitable for immediate publication.*

Reply:

We are very grateful for your positive assessment of our study and for the thoughtful suggestions provided. Your comments are highly motivating and have guided us in refining our manuscript. In the revised version, we have carefully incorporated additional comparisons with existing methods and clarified key aspects of our approach through more explicit explanations and discussion. We truly appreciate the time and effort you devoted to reviewing our work, which has significantly improved the quality of the paper and also inspired potential directions for our future research.

Reviewer Point P 1.2—*The comparison with existing techniques is somewhat limited and requires some clarifications. For instance, in Figure 3: 1.1 Many techniques are marked as "not applicable", but the motivation is not clearly explained. For instance, in W-SINDy, I suspect the non applicability is associated with the PDEs being framed*

in complex non-squared domains, which makes it difficult to rely upon algorithms such as the FFT. Still, this is not clearly stated anywhere.

Reply:

We sincerely thank the reviewer for these valuable comments. In the revised manuscript, the section *Comparison against existing PDE discovery algorithms* has been thoroughly revised according to the reviewer’s suggestions. We have added a clear description of the four categories used to classify the discovery outcomes, which makes it easier to be understand.

Revised Comparison against existing PDE discovery algorithms on page 9.

The ability of existing methods and our proposed framework to discover these PDEs and corresponding time overhead are provided in Fig. 3a. The discovery outcomes are categorized into four classes. Results are deemed correctly discovered when the identified equation exactly matches the true form. If the correct terms are present but accompanied by additional terms, the result is classified as discovered with redundant terms. When the identified equation fails to include all of the true terms, it is considered incorrectly discovered. Finally, if a method is technically unable to address a given type of equation, the case is marked as not applicable.

In addition, we have explicitly explained the reasons why certain methods are marked as “not applicable.” In particular, we agree with the reviewer’s assessment regarding W-SINDy, and we have clarified that its non-applicability arises from difficulties in handling PDEs defined on complex, non-rectangular domains where FFT-based techniques cannot be directly applied.

Revised Comparison against existing PDE discovery algorithms on pages 10~11.

While W-SINDy and WeakIdent employ weak-form formulations to enhance robustness and accuracy, their reliance on FFT-based techniques makes them difficult to apply when PDEs are posed on complex or non-squared domains.

Moreover, we have also explained why other methods are “not applicable” in certain circumstances. For example, sparse-regression based methods is usually unable to handle free-form terms like xu_{tt} and u_x/x , since their candidate library is technically defined by polynomials of derivatives. Meanwhile, most existing approaches that fix the temporal derivative u_t as the left-hand-side (LHS) term struggle to accommodate mixed partial derivatives such as u_{xt} and u_{xxt} .

Revised Comparison against existing PDE discovery algorithms on page 9.

However, we find that they are usually not applicable to free-form terms, since

the candidate library is practically restricted to a finite set of polynomial derivative combinations and cannot accommodate infinite free-form expressions, such as xu_{tt} and u_x/x . Expanding the candidate library to include broader forms would simultaneously increase the difficulty of sparse optimization and hyperparameter tuning, making it challenging to preserve parsimony and leading to the appearance of redundant terms.

Generally, most existing approaches that fix the temporal derivative u_t as the LHS term struggle to accommodate mixed partial derivatives such as u_{xt} and u_{xxt} . In contrast, EqGPT does not require a predetermined LHS term, which can adaptively adjust the equation structure during optimization.

Reviewer Point P 1.3—1.2 *What is the precise meaning of "Discovered with redundant terms" and "Incorrectly discovered"? How many terms need to be captured for a model to fall in a category or the other? Does the accuracy on the coefficients matter in this classification?*

Reply:

Thank you for pointing out this issue to us. In the revised manuscript, we have added the precise meaning of “Discovered with redundant terms” and “Incorrectly discovered”? The essential distinction lies in whether all correct terms are present in the identified equation. *Discovered with redundant terms* indicates that all correct terms are included but accompanied by additional terms, whereas *incorrectly discovered* denotes that not all correct terms are captured. For example, for Burgers’ equation where the true terms are u_t, uu_x, u_{xx} , if the discovered structure is $u_t, uu_x, u_{xx}, u^2u_x$, it is *Discovered with redundant terms*. If the discovered structure is u_t, uu_x, u_{xxx} , it is *incorrectly discovered*.

Revised *Comparison against existing PDE discovery algorithms* on page 9.

The discovery outcomes are categorized into four classes. Results are deemed correctly discovered when the identified equation exactly matches the true form. If the correct terms are present but accompanied by additional terms, the result is classified as discovered with redundant terms. When the identified equation fails to include all of the true terms, it is considered incorrectly discovered. Finally, if a method is technically unable to address a given type of equation, the case is marked as not applicable.

In this classification, the accuracy of the recovered coefficients is not of primary importance, as all experiments were conducted on clean data. The main objective was to assess the capability and applicability of different methods in discovering various

PDE forms. In the revised manuscript, we have emphasized this point. At the same time, we acknowledge that comparing the stability and accuracy of PDE discovery under sparse and noisy data is also essential. Following the reviewer’s suggestion, we have added such comparisons, as detailed in our response to Reviewer Point P 1.4.

Revised Comparison against existing PDE discovery algorithms on page 9.

First, we focus on comparing the discovery scope of PDEs and discovery efficiency.

Eight canonical PDEs are examined, which involves typically-used baselines as well as complex-form PDEs recorded in the math handbook²⁶. For each case, 10,000 discrete data with no noise are sampled for PDE discovery.

Reviewer Point P 1.4—*Related to my previous questions: the authors use a NN-surrogate to circumvent the problem of having to estimate derivatives from noisy data. This strategy is very common in the literature of equation discovery, and has been integrated, e.g., with methods like SINDy: see the recent work by Meng and Qiu on the integration of Gaussian Processes within the SINDy framework (2025), or the paper by Chen, Liu and Sun (2021, published on this same journal) on the use of PINNs+SINDy. I believe the authors should include a comparison with another methods from the SINDy family, aside from W-SINDy (which is far too limited in the considered framework), to better motivate the improvements given by their proposed strategy. I suspect, in fact, that combining an NN-surrogate (as done here) with plain SINDy could provide good results (provided that the penalization coefficients are chosen accordingly and the library is broad enough).*

Reply:

Thank you for these constructive comments! According to the reviewer’s suggestion, we have included the comparison with more relevant methods, including the PINN-SR (Chen, Liu and Sun, 2021) and GP-SINDy (Meng and Qiu, 2025) in the revised manuscript. The results are shown in the revised Fig. 3.

Revised Fig.3.

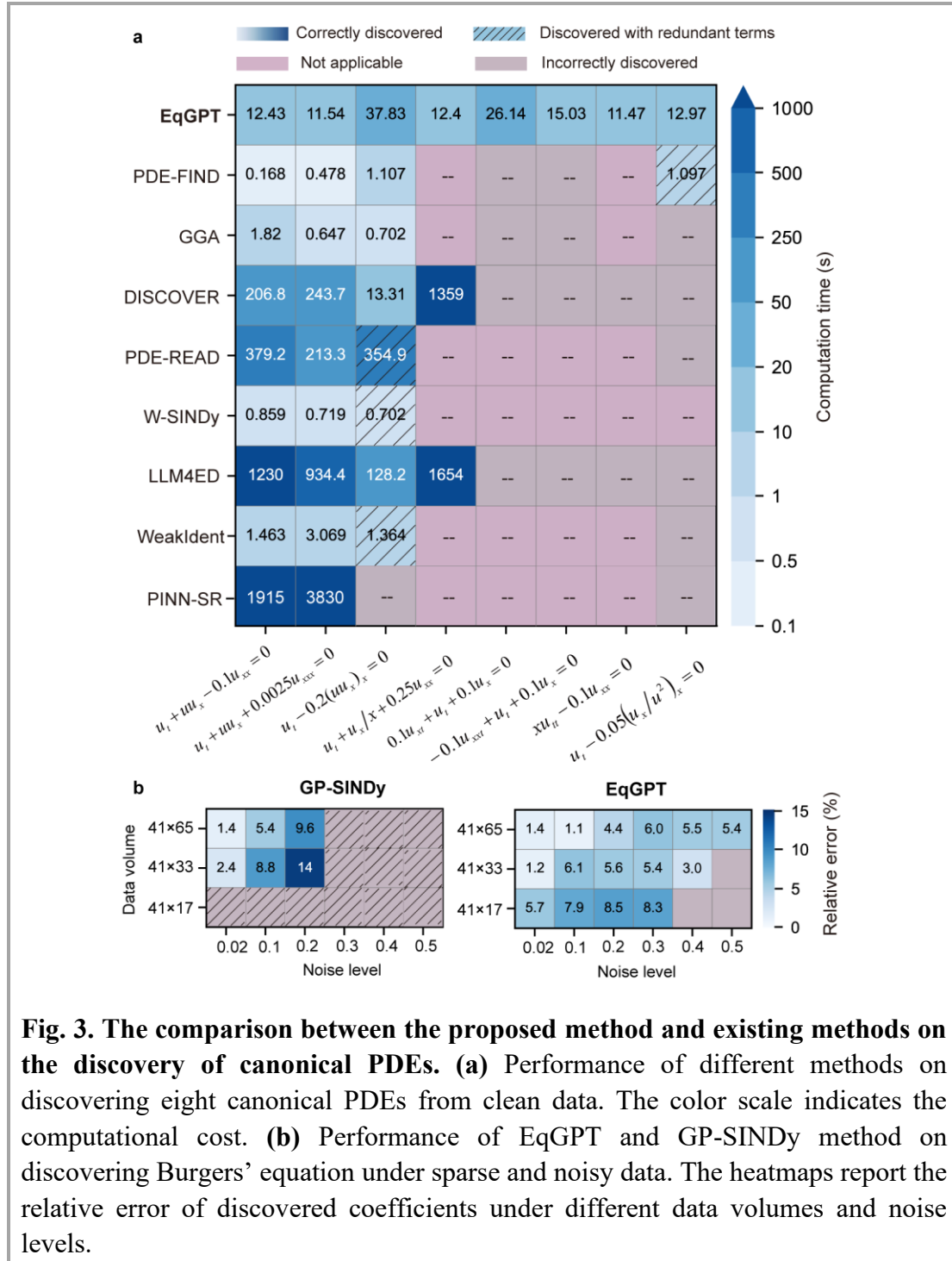


Fig. 3. The comparison between the proposed method and existing methods on the discovery of canonical PDEs. (a) Performance of different methods on discovering eight canonical PDEs from clean data. The color scale indicates the computational cost. **(b)** Performance of EqGPT and GP-SINDy method on discovering Burgers' equation under sparse and noisy data. The heatmaps report the relative error of discovered coefficients under different data volumes and noise levels.

After the comparison with PINN-SR, we have expanded the discussion to cover PINN-based PDE discovery frameworks more broadly. In the revised manuscript, we highlight their advantages, such as strong robustness and flexibility, while also noting their limitations, including high computational cost and sensitivity to initialization. We further contrast these characteristics with EqGPT, which achieves efficient and stable discovery by directly generating free-form PDE structures through a fine-tuned generative model.

Revised Comparison against existing PDE discovery algorithms on page 11.

Meanwhile, the integration of PINNs with PDE discovery has emerged as a promising direction, demonstrating notable robustness and flexibility^{32,35,36}. However, such approaches typically rely on an alternating direction optimization (ADO) scheme, as in PINN-SR, which leads to a high computational cost. Meanwhile, the success of the procedure depends on favorable initialization, which may fail to converge in complex problems. In contrast, EqGPT leverages a generative model fine-tuned on mathematical handbooks to directly propose free-form PDE structures, which allows it to generate plausible candidates efficiently and to converge rapidly, thereby offering superior flexibility and robustness.

On the other hand, we have added a comparison of PDE discovery performance under sparse and noisy data. As suggested by the reviewer, GP-SINDy is selected as the baseline. As one of the most recent approaches, it employs Gaussian processes to mitigate the influence of noise and has demonstrated greater robustness and accuracy than many existing SINDy-based methods. It is worth noting that under clean data conditions, GP-SINDy is essentially similar to PDE-FIND (Fig. 3a); therefore, we focus our comparison on its performance under sparse and noisy data. Our results show that EqGPT achieves even higher accuracy and stability in the sparse and noisy setting. Notably, in this experiment we used the Burgers' equation as an example, with GP-SINDy including derivatives up to third order and polynomial terms up to second order, yielding 15 candidate terms in the library. Restricting the library to second-order derivatives, as reported in the original GP-SINDy paper, leads to improved performance, but this comes at the cost of limiting its practical applicability. These aspects are discussed in detail in the revised manuscript, with further experimental details provided in the Materials and Methods section.

Revised Comparison against existing PDE discovery algorithms on pages 11~12.

Next, we focus on the stability and accuracy of EqGPT compared with existing methods under sparse and noisy data. For comparison, we select GP-SINDy³⁷, which combines Gaussian processes with SINDy and has been shown to achieve higher robustness than most existing methods. We consider Burgers' equation with a diffusion coefficient of 0.5 as an example. Since GP-SINDy requires grid data, we construct datasets by subsampling to obtain grids of size 41×65 , 41×33 , and 41×17 (denoted as $n_t \times n_x$), corresponding to 2,665, 1,353, and 697 data points, respectively. For each dataset, Gaussian noise with noise levels=0.02, 0.1, 0.2, 0.3, 0.4, and 0.5 is added. In the GP-SINDy experiments, we include derivatives up to third order and polynomial terms up to second order, resulting in 15 candidate terms in the library. Detailed experimental settings are provided in Materials and Methods. Accuracy is evaluated for correctly discovered equations by the mean relative error of the

recovered coefficients defined in Eq. (3). The results are presented in Fig. 3b. EqGPT demonstrates high stability and accuracy across sparse and noisy scenarios even with only 697 data points, which remains robust to noise levels up to 0.3. By contrast, GP-SINDy exhibits robustness to noise levels up to 0.2 when using 2,665 or 1,353 data points, which already exceeds many comparable algorithms^{5,38}. However, it fails to maintain sparsity under higher noise, leading to incorrect results. With as few as 697 data points, GP-SINDy cannot tolerate any noise. Furthermore, even at the same noise levels, EqGPT achieves lower relative coefficient errors than GP-SINDy, owing to its flexible regression process and the denoising and optimization capability of physics-informed neural network. Notably, GP-SINDy can achieve improved performance with a smaller candidate library, when limited to second-order derivatives. However, this restriction reduces its ability to address diverse PDE structures and limits its practical applicability. In contrast, EqGPT is able to operate within a broad search space and still accurately recover equations from sparse and noisy data.

Revised *Materials and Methods* on page 26.

In the comparison with GP-SINDy, we include derivatives up to third order and polynomial terms up to second order to establish the candidate library, which is denoted as:

$$\Phi_{\text{GP-SINDy}} = \left[1, u, u_x, u_{xx}, u_{xxx}, u^2, uu_x, uu_{xx}, uu_{xxx}, u_x^2, u_x u_{xx}, \right. \\ \left. u_x u_{xxx}, u_{xx}^2, u_{xx} u_{xxx}, u_{xxx}^2 \right] \quad (18)$$

The noise is added as: $\sigma = \sigma_{\text{NR}} \times \|U\|_F / \sqrt{N_{\text{data}}}$, where σ_{NR} is the noise level. $\|U\|_F$ represents the Frobenius norm of the clean data matrix U . N_{data} is the number of observation data. For EqGPT, the penalty strength α_0 is 0.3.

The reviewer raises a very interesting point that “combining an NN-surrogate with plain SINDy could provide good results”. In fact, our earlier work on DL-PDE (Xu, Chang, Zhang, 2019) adopted exactly this idea by combining a neural network surrogate with SINDy. The results demonstrated that, compared with plain SINDy, this approach is more robust when dealing with limited data and higher noise levels. Nevertheless, it should be emphasized that the major limitation of SINDy-based methods lies in the selection of the candidate library. As we have shown in our previous work (Xu, Chang, Zhang, 2020), enlarging the candidate library increases the difficulty of sparse optimization and hyperparameter tuning, which can ultimately cause the discovery process to fail. Moreover, in practical applications, it is rarely possible to guarantee that the proposed library is complete, which has long been a dilemma for sparse-regression methods. Therefore, EqGPT is essentially designed to address this issue: by training a generative model on mathematical handbooks, EqGPT can generate and optimize free-form PDEs without being constrained by a predefined candidate

library.

Reviewer Point P 1.5—3. *The authors claim that their method is knowledge-based, but: 3.1 The generation process is constrained to ensure mathematically valid expressions, not physically consistent ones (although the authors state that "This enables [...] filtering out physically implausible expressions");*

Reply:

We thank the reviewer for pointing this out. In our work, the term knowledge-based specifically refers to the fact that EqGPT is trained on the PDE dataset obtained from mathematical handbooks, from which it learns tokens and their associated contribution probabilities. As these tokens correspond to mathematically meaningful and physically interpretable terms, the generation process is guided toward producing plausible structures. We agree that this does not enforce full physical consistency, but it ensures that the generated expressions are composed of meaningful building blocks rather than arbitrary symbols. We have added some clarifications in the revised manuscript.

Revised *Comparison against existing PDE discovery algorithms* on page 4.

This enables the autonomous generation of free-form candidate equations directly from physically meaningful tokens, while implicitly filtering out mathematically implausible expressions, thereby improving both the efficiency and the relevance of the search process without relying on brute-force enumeration.

Reviewer Point P 1.6—EqGPT only "sees" the data through the lens of the generation-evaluation-optimization loop, and cannot (as of now) integrate external knowledge of a given system. Considering the extreme flexibility of transformers in handling sequences, symbols and sentences, an exciting direction of future research could be to construct EqGPT so that, aside from the data, one could condition the generation process with actual words (e.g. "fluid", "turbulence", "force", "stable", "conservation", etc.). This would make the model truly domain-adaptive and would allow EqGPT to stand out. Currently, this is not possible, reason for which, regardless of the data at hand, EqGPT always begins its generation process with the "start" token alone. These final remarks are not meant to urge the authors to develop a more general or powerful version of EqGPT, but rather to encourage them to moderate some of their claims.

Reply:

We greatly appreciate this insightful comment, which indeed highlights a promising direction for future research. We fully agree that conditioning the generation process on domain-specific cues would further enhance the adaptability of EqGPT. Inspired by this suggestion, we have added a corresponding outlook in the *Discussion* section to clarify our current scope and to outline this possibility as an avenue for future work. We sincerely thank the reviewer for this inspiring perspective.

Revised *Discussion* on page 21.

Beyond the current generation-evaluation-optimization loop, we wish to enable EqGPT to condition its generation on physical prompt with simple natural-language descriptors, such as “fluid”, “turbulence”, “force”, and “stable”, thereby moving from a generic start token toward genuinely domain-adaptive discovery. A promising way to incorporate domain knowledge is to extend EqGPT with multimodal information, aligning equations, data, language, and visual representations within a shared latent space.

Response to Reviewer #2

Reviewer Point P 2.1—*The paper presents a knowledge-guided generative framework for discovering partial differential equations (PDEs) from data, introducing a model called EqGPT. This model integrates structured domain knowledge - drawn from mathematical handbooks—with generative modeling techniques, aiming to improve both the scope and efficiency of data-driven PDE discovery.*

In particular, PDEs are encoded as sequences of tokens (representing operators and physical terms), thereby transforming equations into sentence-like structures that are amenable to sequence modeling. EqGPT is pretrained on 221 canonical PDEs extracted from Debnath's Nonlinear Partial Differential Equations, and augmented via permutation and reordering to produce 7,072 training instances. The framework operates through a "generation–evaluation–optimization" loop, where EqGPT proposes candidate PDEs, evaluates their data fidelity using a reward function, and fine-tunes its generative process based on top-performing equations.

The framework is tested on rediscovering canonical PDEs, including the Poisson equation in 2D and 3D irregular domains, and is compared with several baseline algorithms. Finally, the model is applied to real experimental data, where it successfully discovers a governing PDE that aligns well with observed phenomena - a particularly compelling result that demonstrates the method's practical utility and is highly appreciated by the reviewer.

Reply:

We sincerely thank the reviewer for the positive evaluation of our work, in particular the appreciation of the application to real experimental data. We are especially pleased by this recognition, as our long-term goal has been to advance PDE discovery toward the generation of new scientific knowledge in practical applications. We are also very grateful for the reviewer's constructive suggestions, which highlighted areas where our explanations were insufficient and guided us in improving the clarity and rigor of the manuscript. In the revised version, we have added additional experiments and detailed explanations to address these points. We greatly appreciate the reviewer's effort and insightful feedback, which have significantly strengthened the paper.

Reviewer Point P 2.2—*Major Concerns: Nature and Utility of the Generative Model*
From the description of the generation process (lines 174–181), it appears that EqGPT is effectively sampling combinations of terms to construct PDE forms, which are subsequently evaluated using a linear regression model. For instance, using the Fig. 1c

as an example, if a PDE form $u u_x + u_t + u_{xxx}$ is generated by the EqGPT, then the discovered PDE is a linear combination of $u u_x$, u_t , u_{xxx} , the equation is treated as a linear combination of those terms, with coefficients determined via least squares fitting. This raises an important conceptual question: Is EqGPT essentially performing stochastic enumeration? It seems to me that the method is like an exhaustive enumeration that as many as possible PDE forms are enumerated and then tested to find best ones. Since the space of possible token sequences is finite (given a finite "vocabularies" and training instances), the sampling process can be viewed as drawing from a discrete distribution. In this case, why is a generative model necessary, especially one based on autoregressive language modeling typically suited for output spaces of infinite dimensionality? Moreover, after selecting the top 10 equations based on reward, these are used to fine-tune EqGPT. However, the purpose of this fine-tuning remains unclear. If the top equations are already selected based on their evaluation scores, why adjust the model to sample them more often? This could reinforce generation of those specific forms, but the next round of evaluation will still depend on reward ranking rather than generation probability. As such, the value of model fine-tuning within this loop could be more explicitly justified.

Reply:

We sincerely thank the reviewer for raising this important point. Regarding the concern of whether EqGPT is essentially performing stochastic enumeration, we would like to clarify that the EqGPT can generate infinitely many equation structures by combining operators with physically meaningful terms from its token vocabulary, and thus cannot be explored through exhaustive enumeration to identify the correct form.

As illustrated in Fig. 1c, the generative process outputs equations token by token, where tokens include operators and basic terms. As a result, composite expressions (like uu_x , $(ux)^2$) must be dynamically constructed through the sequential generation of operators and terms. In the revised Table S1, we list all tokens used in EqGPT, showing that apart from operators, every token corresponds to a basic term and no compound terms (e.g., " uu_x ") are included. Consequently, EqGPT is capable of generating highly diverse free-form PDEs, including expressions such as $u/u_x + u_t + u \times u_x \times u_{xxx}$, which is far beyond any finite enumerated set. Faced with such an unbounded search space, exhaustive enumeration would be both inefficient and ineffective. To overcome this challenge, EqGPT leverages knowledge learned from an established PDE dataset extracted from mathematical handbooks, capturing co-occurrence probabilities among terms and thereby generating equations that are mathematically valid and physically meaningful. Additional explanations have been added in the revised manuscript to avoid possible misunderstanding.

Added explanations on page 4 and pages 5~6.

This enables the autonomous generation of free-form candidate equations directly from physically meaningful tokens, while implicitly filtering out mathematically implausible expressions, thereby improving both the efficiency and the relevance of the search process without relying on brute-force enumeration. (page 4)

Therefore, EqGPT can generate infinitely many equation structures by combining operators with physically meaningful terms from its token vocabulary. (page 5~6)

The reviewer also noted the need to better explain the role of fine-tuning on the top-10 equations. In the revised manuscript, we have added detailed descriptions (page 6) together with illustrative examples (see revised Table S2). Conceptually, this fine-tuning strategy is inspired by reinforcement learning, but implemented in a computationally more efficient and stable manner. The key idea is that high-reward equations are more likely to contain correct terms; by training EqGPT on these equations, the model becomes more inclined to generate such terms in subsequent iterations. As shown in revised Table S2 for the case of rediscovering the KdV equation with 1,000 data points, the top-5 equations from the first generation already included some correct components but still deviated from the true form. Without fine-tuning, relying solely on random generation is difficult to recover the exact equation. With fine-tuning, EqGPT increasingly favors high-reward components, enabling the correct PDE to be identified by the third generation.

Added explanations on page 6.

Initially, EqGPT is only trained on mathematical handbooks, which ensures that its generations are mathematically valid but not necessarily consistent with the observed data. The fine-tuning strategy serves to adapt the generative model by learning from the top-ranked equations that best fit the data. In this way, EqGPT increases the likelihood of sampling terms that are more consistent with the underlying dynamics, since equations with higher rewards are more likely to contain the correct components. This mechanism is illustrated in Table S2 by an example for better understanding. In the first epoch, the top-ranked equations may already include some correct terms but still deviate from the true structure, whereas after fine-tuning, EqGPT increasingly favors higher-reward terms and successfully recovers the correct PDE by the third epoch.

Table S2. The top-5 equations in each epoch of the optimization process when

discovering KdV equation with 1000 data

	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
Epoch 1	$uu_x, u_t, u^2u_x, u_{xxx}$	$uu_x, u_t, u/t, u_{xxx}$	$u_{xxx}, u_xu, (u_x)^2u^2, u_t$	$(uu_x)_x, u_{tt}, u_{xxxx}, u_{xx}$	$u_{xxx}, uu_x, u^2u_x, (u(u^2)_{xx})_{xx}, u_{xx}, u_t$
Epoch 2	$uu_x, u_t, u^2u_x, u_{xxx}$	$uu_x, u_{xxx}, u_t, (u_x)^2$	$u_xu, u_{xxx}, u_{xx}, u_t$	$uu_x, u_t, u/t, u_{xxx}$	$u_{xxx}, u_xu, (u_x)^2u^2, u_t$
Epoch 3	uu_x, u_t, u_{xxx}	$u_{xxx}, u_t, u^2u_x, u_{xxx}$	$u_xu, u_{xxx}, u_t, (u_x)^2$	$u_xu, u_{xxx}, u_{xx}, u_t$	$uu_x, u_t, u/t, u_{xxx}$
Epoch 4	uu_x, u_t, u_{xxx}	$uu_x, u_t, u^2u_x, u_{xxx}$	$u_{xxx}, u_xu^2, u_t, u_xu$	$u_xu, u_xu_{xxxxx}, u_t, u_{xxx}$	uu_x, u_t, u_{xxx}, u_x
Epoch 5	uu_x, u_t, u_{xxx}	u_{xxx}, uu_x, u_t	$u_xu, u_t, u^2u_x, u_{xxx}$	$u_{xxx}, (uu_x)_x, uu_x, u_t$	$u_{xxx}, u_xu^2, u_t, uu_x$

In the revised manuscript, we have expanded the Discussion section with a more in-depth analysis. Starting from the core challenges of PDE discovery, we explain why a generative model is adopted and highlight its differences and advantages compared with traditional sparse regression and tree-based symbolic regression approaches. We hope these revisions will help the reviewer and potential readers better understand the motivation and significance of our work.

Added discussions in *Discussion* on page 21.

In the context of PDE discovery, the generation and optimization of potential equations are central aspects. Sparse regression restricts the search space to a predefined candidate library to improve search efficiency, limiting the diversity of admissible terms, whereas tree-based symbolic regression expands expressiveness by constructing free-form structures, but at the expense of optimization efficiency. To balance flexibility and efficiency, we adopt a generative approach in this work. Despite the structural diversity of PDEs, their constituent elements exhibit quantifiable regularities. Building on this observation, we introduce the generative representation of equations (GRE) strategy, which reformulates PDE structures as sentence-like sequences of operators and basic terms. By embedding known equation forms into the learning process, EqGPT not only identifies plausible terms but also captures their co-occurrence patterns, thereby guiding discovery in a more structured manner. Notably, through combinations of operators and basic terms, EqGPT can generate infinitely many PDEs while preserving mathematical validity. Traditional optimization approaches include sparse optimization, which offers computational efficiency but relies on the predetermined candidate library, and genetic programming, which provides greater flexibility but incurs substantial computational cost. In contrast, EqGPT leverages the strengths of generative modeling by fine-tuning on the highest-rewarded sequences from each optimization epoch, thereby steering subsequent sampling toward equations that better match the data. Conceptually, this fine-tuning strategy is inspired by reinforcement learning, but it

achieves superior computational efficiency and stability. Notably, although previous studies have also attempted to construct generative models for symbolic regression^{12,21,47}, these models largely focused on algebraic expressions and lacked the capacity to handle differential operators, making them unsuitable for PDE discovery.

Reviewer Point P 2.3—*The definition of the reward function (Equation 1) is unclear and requires improvement. I recommend rewriting this section to more clearly and precisely define all variables involved. For example, the definition of θ_{LHS} is ambiguous - it is described as "the negative of the first term of the structure," yet the first term refers to a symbolic PDE term, not a numerical value. It is unclear how this symbolic expression is converted into a numerical vector or quantity for use in the reward calculation. A more rigorous explanation, including the mathematical meaning of each component and how these are derived from the PDE structure and surrogate model output, would greatly improve clarity.*

Reply:

We thank the reviewer for pointing out this important issue. In the revised manuscript, we have provided a more rigorous and detailed explanation of the reward function, including precise definitions of all variables involved and clarification of how symbolic PDE terms are converted into numerical vectors using the surrogate model. These additions are intended to improve clarity and allow readers to better understand the formulation and role of the reward function. In fact, EqGPT generates the structural form of an equation (e.g., $u_x + u_t + u \times u_x = 0$), while the coefficients of terms are determined through regression. To perform least-squares regression, we move the first term to the left-hand side (LHS) and treat the remaining terms as the right-hand side (RHS) terms, for example, the equation is converted to $-u_x = u_t + u \times u_x$. This is the reason why the LHS is described as the “negative of the first term.”

Added explanations on page 6.

Benefitted from the proposed GRE technique for PDEs, EqGPT can generate diversified free-form PDEs. A reward function is defined to evaluate the generated PDEs:

$$\text{Reward} = \alpha \cdot R^2 = (1 - \alpha_0 \log_{10} N_{term}) \cdot \left(1 - \frac{\sum (\theta_{LHS} - \theta_{RHS} \cdot \vec{\xi})^2}{\sum (\theta_{LHS} - \bar{\theta}_{LHS})^2} \right). \quad (1)$$

where the reward is composed of two components: the penalty term α and the determination coefficient R^2 . The former is defined to measure the sparsity, which penalizes the number of terms in the structure to obtain a parsimonious PDE, and the latter is employed to measure the consistency with observation data through the R^2

of the regression. Notably, EqGPT generates the structural form of an equation (e.g., $u_x + u_t + u \times u_x = 0$), while the coefficients of terms are determined through regression. To perform least-squares regression, we move the first term to the left-hand side (LHS) and treat the remaining terms as the right-hand side (RHS) terms (e.g., $-u_x = u_t + u \times u_x$). Therefore, the LHS term is designated as the negative of the first term. Using the surrogate model trained on observation data, automatic differentiation is employed to compute the numerical values of each PDE term at the N_{meta} spatiotemporal meta-data points. Thus, the value of LHS term is represented as a vector θ_{LHS} with the size of $N_{\text{meta}} \times 1$, while the values of RHS term is represented as a matrix Θ_{RHS} of size $N_{\text{meta}} \times (N_{\text{term}} - 1)$. N_{term} is the total number of terms in the structure. We then solve a least-squares regression problem, $\theta_{LHS} \approx \Theta_{RHS} \tilde{\xi}$, where $\tilde{\xi}$ is the coefficient vector obtained from regression with the size of $(N_{\text{term}} - 1) \times 1$. Through the observed and fitted LHS term, the R^2 can be calculated to represent their agreement. In the equation, $\bar{\theta}_{LHS}$ is the mean value of θ_{LHS} . Overall, the higher is the reward, the better is the generated PDE.

In addition, we also added further clarifications in the Materials and Methods section.

Added clarifications in *Materials and Methods* on page 25.

The reward is calculated via Eq. (1). For a candidate PDE structure, for example $u_{xx} + u_t + uu_x = 0$, we designate the negative of the first term ($-u_{xx}$) as the LHS, while the remaining terms (u_t, uu_x) form the RHS. The coefficients are obtained from the least squares regression on the values of LHS and RHS. Importantly, EqGPT generates equation structures with diverse LHS terms (rather than a pre-fixed u_t), thereby increasing the degrees of freedom available to the regression.

Reviewer Point P 2.4—Explicit Vocabulary List

The paper mentions 56 tokens used for constructing PDEs, but these are not explicitly listed. Including a table or appendix with the full list would greatly improve transparency and reproducibility. Currently, the reader must infer this set from Fig. 1a.

Reply:

We thank the reviewer for this helpful suggestion. In the revised manuscript, we have explicitly provided the complete vocabulary of 56 tokens in revised Table S1 to ensure greater transparency and reproducibility.

Table S1. The tokens used for constructing PDEs in the EqGPT framework

Index	Token	Index	Token	Index	Token
1	E	20	u_{xxx}	39	u_z
2	+	21	u^3	40	u_{zz}
3	*	22	x	41	Δu_{tt}
4	/	23	u_{yy}	42	$x+y$
5	S	24	$(1/u)_{xx}$	43	e^x
6	u_t	25	$(u_x/u^2)_x$	44	u_{yyt}
7	u	26	u_{xxxxx}	45	$\sin(t)$
8	u_x	27	$(u_t)^3$	46	$\sin(x)$
9	u_{xx}	28	\sqrt{u}	47	$(u_{xx}+u_x/2)^2$
10	u_{xxt}	29	$\sin(u)$	48	x^4
11	$(u_t)^2$	30	$\sinh(u)$	49	\sqrt{x}
12	u_{xt}	31	$\Delta^2 u$	50	e^{-y}
13	u_{tt}	32	u_y	51	t
14	$(u_x)^2$	33	Δu	52	u_{yyy}
15	u_{xxx}	34	y	53	$(uu_x)_t$
16	$(uu_x)_x$	35	x^2	54	$(uu_x)_{xx}$
17	u_{xxtt}	36	y^2	55	$(u^3)_{xx}$
18	$(u^4)_{xx}$	37	$(u_y)^2$	56	$(u(u^2)_{xx})_{xx}$
19	u^2	38	u_{xy}		

Reviewer Point P 2.5—Handling of Non-Permutational Equivalences

*The paper notes that "the semantic meaning of a PDE remains invariant under permutation of additive terms." However, many mathematical equivalences are more subtle. For instance, $(u u_x)_x$ is considered as a vocabulary, and it is in fact equivalent to $u_x * u_x + u * u_{xx}$. Another example would be $(u^2)_x$ is equivalent to $2 u * u_x$. These are common in conservation laws and other physically meaningful transformations. How the framework handles - or fails to handle - such equivalences deserves further discussion.*

Reply:

We thank the reviewer for raising this important point. In the revised manuscript, we have added an illustrative example and corresponding discussion to clarify how EqGPT addresses such non-permutational equivalences (Supplementary Information S1.4). Benefiting from its free-form generation ability, EqGPT is not restricted to predefined compound tokens but can construct mathematically equivalent forms from basic terms and operators. As demonstrated in the Fig.S4, for $(uu_x)_x$, EqGPT is able to recover both the compound form and its expanded equivalent expression composed of elementary terms. These equivalent structures can appear among the top-ranked discovered equations, which highlights the framework's flexibility in handling

mathematically equivalent formulations.

Added experiments in *Supplementary Information S1.4* on pages 35~36.

1.4 Discovery of PDEs with equivalent forms

In practice, many conservation laws and other physically meaningful transformations lead to PDEs with mathematically equivalent formulations. To examine how the proposed EqGPT framework addresses the issue of such mathematical equivalences among PDE terms, we designed an experiment with a compound-form PDE, denoted as PDE_compound:

$$u_t - 0.2(uu_x)_x = 0. \quad (\text{S.2})$$

This PDE does not appear in the established PDE dataset, and thus poses a challenge for the discovery task. The observation dataset is obtained from the simulation via *Mathematica*. The initial condition is $u(0,x)=\sin(\pi x)$, and the boundary condition is $u(t,0)=u(t,1)=0$. The dataset contains the grid data of 200 spatial observation points in the domain $x \in [0,1]$ and 200 temporal observation points in the domain $t \in [0,1]$; thus, the data size is 40,000. Here, 10,000 data points with no noise are randomly selected to train the surrogate model. The EqGPT successfully identified the correct PDE form in the 3rd optimization epoch, though with relatively lower coefficient accuracy due to the rarity and complexity of this structure. Importantly, beyond the compound expression (Eq. S.2), the framework also discovered its mathematically expanded equivalent:

$$u_t - 0.2uu_{xx} - 0.2u_x^2 = 0. \quad (\text{S.3})$$

This result demonstrates that the proposed framework is capable of independently recognizing and recovering both compound and expanded representations of the same PDE, thereby exhibiting flexibility in handling mathematically equivalent formulations.

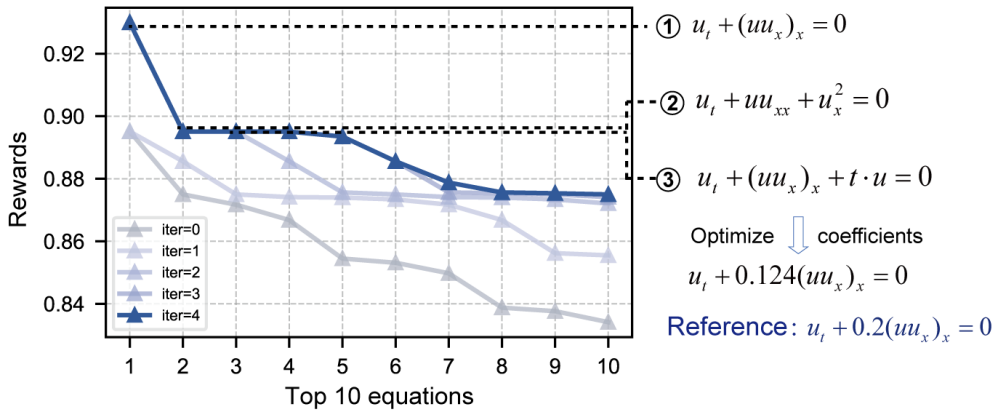


Fig. S4. Discovery of PDEs with equivalent forms. The rewards of the top 10 equations generated from the EqGPT model in each optimization epoch (**left**), and the discovered PDE and the true PDE (**right**). The best-discovered form and the suboptimal form are mathematically equivalent.

Reviewer Point P 2.6—Dimensional Consistency of Generated Terms

In practice, PDEs must match the dimensionality of the physical problem. For instance, in a 1D time-dependent system, it would be incorrect to include terms like u_y or u_{xy} . It is unclear how EqGPT enforces this constraint during generation. Clarifying how dimensional appropriateness is maintained (e.g., through token masking or probability constraints) would strengthen the methodology section.

Reply:

We thank the reviewer for this valuable suggestion. In the revised manuscript, we have added a detailed explanation of how dimensional consistency is ensured during generation. Specifically, EqGPT employs a token masking strategy: tokens that are incompatible with the dimensionality of the problem are masked out, and the probability distribution is renormalized over the remaining valid tokens. For example, in a one-dimensional system, all tokens involving y or z derivatives are excluded, while in two-dimensional systems only the z -related tokens are masked. Similarly, composite operators such as divergence and Laplacian are automatically adapted to the specified dimensionality. This guarantees that the generated PDE terms always match the dimensionality of the underlying physical problem.

Added explanations in *Materials and Methods* on page 24.

To ensure dimensional consistency with the underlying physical problem, EqGPT employs a token masking strategy during the generation process. Specifically, the vocabulary tokens that are incompatible with the problem's dimensionality are masked out, and the probability distribution is renormalized over the remaining valid tokens. For example, in a one-dimensional time-dependent system, all tokens involving y or z derivatives (e.g., u_y , u_{xy}) are masked, ensuring that only x and t -dependent terms can be generated. Similarly, for two-dimensional systems, all tokens involving z are excluded. In addition, composite operators such as divergence or Laplacian are automatically adapted to the specified dimensionality. This masking mechanism guarantees that the generated PDE structures remain physically meaningful with respect to the spatial and temporal dimensions of the problem.

Reviewer Point P 2.7—Relation to PINN-Symbolic Regression Hybrids

There is a growing body of literature combining physics-informed neural networks (PINNs) with symbolic regression techniques. It would be helpful for the authors to briefly differentiate their approach from such methods. For example: How does the generative-symbolic mechanism of EqGPT compare in flexibility, robustness, or

interpretability to hybrid PINN-symbolic strategies?

Reply:

Thank you for the insightful comment. In the revised manuscript, we have expanded the section *Comparison between existing methods* to include representative PINN–symbolic regression approaches, such as PINN-SR. We discuss their strengths, including robustness and flexibility, as well as their limitations, such as high computational cost and sensitivity to initialization. We further contrast these characteristics with EqGPT, which achieves efficient and stable discovery by directly generating and optimizing free-form PDE structures through a generative model.

Revised Comparison against existing PDE discovery algorithms on page 11.

Meanwhile, the integration of PINNs with PDE discovery has emerged as a promising direction, demonstrating notable robustness and flexibility^{32,35,36}. However, such approaches typically rely on an alternating direction optimization (ADO) scheme, as in PINN-SR, which leads to a high computational cost. Meanwhile, the success of the procedure depends on favorable initialization, which may fail to converge in complex problems. In contrast, EqGPT leverages a generative model fine-tuned on mathematical handbooks to directly propose free-form PDE structures, which allows it to generate plausible candidates efficiently and to converge rapidly, thereby offering superior flexibility and robustness.

Response to Reviewers

Response to Reviewer #1

Reviewer Point P 1.1—*The authors have thoroughly addressed all my remarks. I have no additional comments.*

Reply:

We sincerely thank the reviewer for the careful evaluation of our work and for confirming that all comments have been satisfactorily addressed. We greatly appreciate your time and constructive feedback, which have helped us improve the clarity and quality of the manuscript.

Response to Reviewer #2

Reviewer Point P 2.1—*Thank you to the authors for thoroughly addressing my comments and concerns; I am satisfied that all have been properly resolved.*

Reply:

We would like to express our sincere gratitude to the reviewer for the thorough assessment of our revisions and for acknowledging that all comments and concerns have been properly resolved. Your feedback has been invaluable in strengthening our work, and we truly appreciate your support.