LLM and Simulation as Bilevel Optimizers: A New Paradigm to Advance Physical Scientific Discovery

Pingchuan Ma[1], **Tsun-Hsuan Wang**[1], **Minghao Guo**[1], **Zhiqing Sun**[2],
Joshua B. Tenenbaum[1 3 4], **Daniela Rus**[1], **Chuang Gan**[5 6], **Wojciech Matusik**[1]
[1]MIT CSAIL, [2]CMU LTI, [3]MIT BCS, [4]CBMM, [5]UMass Amherst, [6]MIT-IBM Watson AI Lab

Abstract

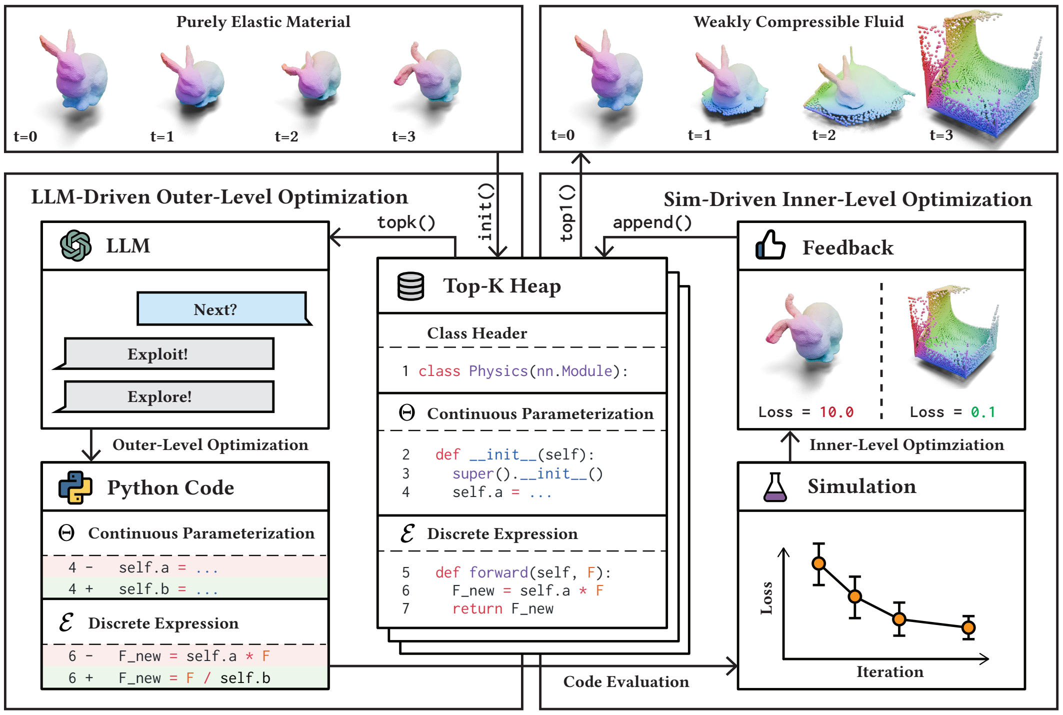
Large Language Models have recently gained significant attention in scientific discovery for their extensive knowledge and advanced reasoning capabilities. However, they encounter challenges in effectively simulating observational feedback and grounding it with language to propel advancements in physical scientific discovery. Conversely, human scientists undertake scientific discovery by formulating hypotheses, conducting experiments, and revising theories through observational analysis. Inspired by this, we propose to enhance the knowledge-driven, abstract reasoning abilities of LLMs with the computational strength of simulations. We introduce *Scientific Generative Agent* (SGA), a bilevel optimization framework: LLMs act as knowledgeable and versatile thinkers, proposing scientific hypotheses and reason about discrete components, such as physics equations or molecule structures; meanwhile, simulations function as experimental platforms, providing observational feedback and optimizing via differentiability for continuous parts, such as physical parameters. We conduct extensive experiments to demonstrate our framework's efficacy in constitutive law discovery and molecular design, unveiling novel solutions that differ from conventional human expectations yet remain coherent upon analysis.

1 Introduction

Physical science automation aims to accelerate discovery [55]. Key aspects of human scientific process include: iterative hypothesis testing [40], discrete and continuous solution components [55], knowledge exploitation with occasional exploration [58], and universal principles with discipline-specific nuances [44]. LLMs excel as generalist tools with vast knowledge [1], aiding scientific discovery through reasoning and natural language interfaces. However, they lack computational capabilities crucial for physical sciences that requires specific domain knowledge.

To this end, inspired by the overarching philosophy of human scientists, we introduce **Scientific Generative Agent (SGA)**, a bilevel optimization approach wherein the outer-level engages LLMs as knowledgeable and versatile thinkers for generating and revising scientific hypothesis, while the inner-level involves simulations as experimental platforms for providing observational feedback. Overall, our contributions are concluded as:

- We present a generic framework for physical scientific discovery that combines LLMs with physical simulations.
- We propose a bilevel optimization with LLMs for discrete-space search-based optimization and differentiable simulations for continuous-space gradient-based optimization.
- We conduct extensive experiments to demonstrate the effectiveness and generality of the proposed framework in physics law discovery and molecular design.



```
class Physics(nn.Module):
    def __init__(self):
        super().__init__()
        self.a = ...
    def forward(self, F):
        F_new = self.a * F
        return F_new
```

```
-   self.a = ...
+   self.b = ...
```

```
-   F_new = self.a * F
+   F_new = F / self.b
```

Figure 1: **The overall pipeline of Scientific Generative Agent (SGA).** Taking the constitutive law searching problem as an example, the input is an initial guess (a purely elastic material), and the output is another constitutive law optimized towards the ground truth (weakly compressible fluid).

2 Scientific Generative Agent

SGA is a bilevel optimization framework where the upper level features LLMs as proposers of scientific solutions, and the lower level utilizes simulations as experimental platforms for validation. We illustrate the overall pipeline in Fig. 1.

2.1 Bilevel Optimization Pipeline

First, we describe the simulation as a process where a simulator takes in a scientific expression and continuous components as inputs and gives simulated physical phenomenon and additional observational feedback as outputs. Next, the LLM acts as a thinker to propose expressions based on past experimental results from simulation. This process involves the LLM taking in a set of past simulation results containing an evaluation of the scientific problem, other physical feedback, and past proposals, along with a prompt. The LLM then outputs proposed expressions and continuous parameterization for the decision variables. With these elements, we define a bilevel optimization problem: the objective is to minimize the evaluation of the simulated physical phenomenon, which depends on the proposed expression, continuous parameterization, and optimal continuous parameters. The optimization problem has two levels: (i) the outer optimization searches for an expression that defines what experiments to be conducted and continuous parameterization that defines the search space of the inner continuous optimization; (ii) the inner optimization, which depends on the outer-level variables, searches for the optimal continuous parameters given the proposed expression via differentiable simulation. We detail the complete algorithm with a python-like pseudo-code in Alg. 1.

2.2 LLM-Driven Outer-Level Search

LLM-driven Optimization LLMs are effective for generic optimization through prompting and context [60, 43]. Inspired by [31], we use evolutionary search with multiple offspring per iteration. Our approach selects several high-performing candidates, enhancing hypothesis feasibility and facilitating crossover, with LLMs generating new hypotheses from past experiments [43].

Table 1: **Benchmark**. We use column **#Iter.** as the number of iterations, **#Hist.** as the K value for the top-k retrieval, $\frac{\textbf{\#Exploit}}{\textbf{\#Explore}}$ as the number of offspring for exploitation versus exploration, **Bilevel** as if bilevel optimization is enabled. The best method with the lowest loss is highlighted in **bold** text.

Method	#Iter.	#Hist.	$\frac{\textbf{\#Exploit}}{\textbf{\#Explore}}$	Bilevel	Constitutive Law Search				Molecule Design			
					(a) ↓	**(b)** ↓	**(c)** ↓	**(d)** ↓	**(e)** ↓	**(f)** ↓	**(g)** ↓	**(h)** ↓
CoT	1	5	N/A	✗	298.5	1462.3	150.0	384.1	3.0	32.1	18.6	6.0
FunSearch	20	2	0 / 4	✗	210.3	872.2	82.8	139.5	1.1	7.1	8.3	1.1
Eureka	5	1	0 / 16	✗	128.0	531.0	101.7	150.1	4.3	9.8	3.3	9.7e-1
OPRO	5	5	0 / 16	✗	136.2	508.3	99.2	128.8	2.4	9.4	3.1	1.3
Ours (no bilevel)	5	5	4 / 12	✗	90.2	517.0	83.6	68.4	8.6e-1	9.1	1.8	1.4
Ours (no exploit)	5	5	0 / 16	✓	3.0e-3	3.9e-1	6.6e-2	**1.4e-12**	4.0e-4	1.5e-1	6.1e-1	**2.8e-5**
Ours	5	5	4 / 12	✓	**5.2e-5**	**2.1e-1**	**6.0e-2**	**1.4e-12**	**1.3e-4**	**1.1e-1**	**5.4e-1**	3.6e-5

Interfacing with Simulation Integrating LLMs with simulation requires efficient, structured communication. We use equation searching and entity searching for LLM-to-simulation communication, unified as an abstraction. Equation searching allows LLMs to propose equations and search spaces, while entity searching focuses on structural descriptions. For simulation-to-LLM communication, we use expert knowledge to extract relevant information as feedback, similar to a senior scientist guiding a junior colleague. The inner optimization results also serve as feedback from simulation to LLMs, detailed in the next section.

Exploitation and Exploration We employ an exploit-and-explore strategy by adjusting LLMs' decoding temperature [60], mimicking human scientists' approach to breakthroughs. When generating offspring, we divide them into two groups: cautious followers (exploit) and daring adventurers (explore). We observed that the exploit group often repeats previous solutions, while the explore group tends to yield overly random or invalid solutions. A 1:3 ratio between exploit and explore groups has proven effective emperically based our experiments.

2.3 Differentiable Inner-Level Optimization

Inner optimization uses gradient-based methods to find optimal parameters within the search space defined by the outer level. Domain-specific knowledge is distilled through gradients from the simulation to intermediate optimization results. These results, along with the final output, are fed back to LLMs for solution refinement. The feedback may include loss curves and auxiliary recordings, providing information on various aspects of improvement.

3 Experiments

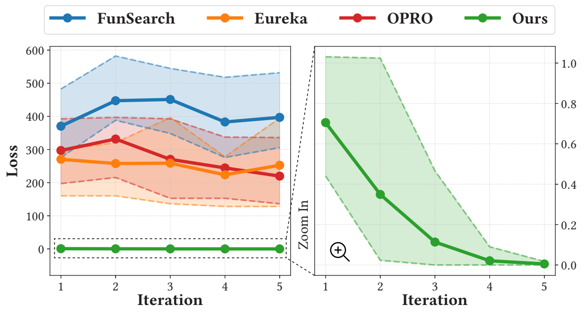
3.1 Problem Definitions

Constitutive Law Discovery Identifying the constitutive law from motion observations stands as one of the most difficult challenges in fields such as physics, material science, and mechanical engineering. Here we follow the recent advances in physical simulation and formulate the constitutive law discovery task as an optimization problem [29].

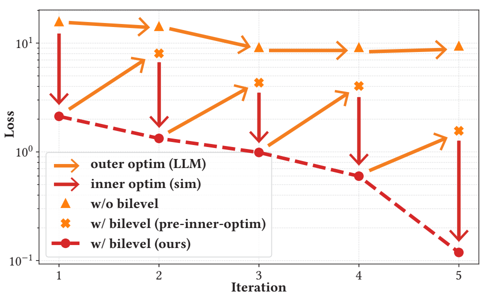
Molecule Design We focus on a prevalent task in molecule design: discovering molecules with specific quantum mechanical properties. Our objective is to determine the optimal molecular structure and its 3D conformation to match a predefined target quantum mechanical property. The design process involves both the discrete expression – the molecular structure represented by SMILES strings [57], and the continuous parameters – the 3D coordinates of each atom in the molecule.

3.2 Experiment Setup

We design a diverse set of challenging tasks for evaluation. For constitutive law discovery, we propose 4 tasks including: **(a)** fitting the non-linear elastic material starting from a linear elastic material, **(b)** fitting the von Mises plastic material starting from a purely elastic material, **(c)** fitting the granular material starting from a purely elastic material, and **(d)** fitting the weakly compressible fluid starting



(a) **Loss trends comparison.**



(b) **Bilevel optimization.**

from a purely elastic material. For molecular design task, we consider 4 popular tasks, centering on 3 commonly evaluated quantum mechanical properties [13, 65]: **(e)** HOMO (Highest Occupied Molecular Orbital) set to 0, **(f)** LUMO (Lowest Unoccupied Molecular Orbital) set to 0, **(g)** the HOMO-LUMO energy gap set to 0, and **(h)** the HOMO-LUMO energy gap set to -2.

3.3 Physical Scientific Discovery

We consider 6 strong baselines for evaluation: (i) **Chain-of-Thoughts (CoT)** prompting [56] solves the problem by looking at step-by-step solutions from examples. We provide 5 examples with an explanation to CoT as the initial solution. (ii) **FunSearch** [43] utilizes evolutionary strategy to avoid local optimum. We adopt the given hyperparameters from the original implementation with 2 optimization histories and 4 explorers. We set the number of iterations to 20, yielding the same number of solutions evaluated, for a fair comparison to other methods. (iii) **Eureka** [31] generates multiple solutions in each iteration to improve the success rate of the generated code. We keep the hyperparameters from the original implementation. (iv) **Optimization by PROmpting (OPRO)** [60] highlights the advantages of involving a sorted optimization trajectory. We set the hyperparameters to be equal to **Eureka** except for the number of historical optimization steps. In all these works (i-iv), we notice the temperatures for LLM inference are all 1.0, which is equal to the exploring temperature in our method, so we denote them with 0 exploiter. We also consider 2 variants of our method: (v) **Ours (no bilevel)** removes the bilevel optimization by only searching with LLM. (vi) **Ours (no exploit)** removes the exploitation by setting the temperature to 1.0 all the time.

We present our experiments against the 8 designed tasks and show the results in Table 1. Compared to baselines (i-iv), our method is significantly better by a number of magnitudes. When the bilevel optimization is removed from our method, the performance drops dramatically, but still statistically better than baselines (i-iv), indicating the choice of hyperparameters and the integration of exploitation is helpful for the task. When we remove the exploitation but restore the bilevel optimization, we notice the performance grows back. It has comparable performance compared to our method in **(d)** or even better results in **(h)**. However, in some tasks, especially hard ones (e.g., **(b)** and **(f)**) that we care more in reality, the performance gap is over 50%, indicating the effectiveness of our exploit-and-explore strategy. We also present the loss trend in task **(a)** in Figure 2a, our method outstands with a much lower loss and a converging trend. We present more experiments in Sec. C.

3.4 Bilevel Optimization

Here we evaluate the importance of bilevel optimization in Figure 2b using the task **(h)**. Comparing the blue triangle curve and the red dot curve, which represent the LLM-driven outer-level optimization and the simulation-driven inner-level optimization, it is easy to conclude that the loss performance with bilevel optimization is better. Nevertheless, we are also interested in how bilevel optimization works inside each optimization step and how much LLMs and simulations help respectively. As shown as a zigzag curve, we found that LLMs and simulations help each other over all optimization steps: the next proposal from LLMs will be better with simulation-optimized results, and vice versa. We argue that LLMs and simulations have different expertise: LLMs are generalist scientists who have cross-discipline knowledge, while simulations are domain experts who have specialized knowledge.

References

[1] Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*, 2023.

[2] Anthropic. Introducing the next generation of claude, 2024.

[3] Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. Multiple regression genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 879–886, 2014.

[4] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International conference on machine learning*, pages 550–559. PMLR, 2018.

[5] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pages 936–945. Pmlr, 2021.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

[8] Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations*, 2024.

[9] William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason Moore. Learning concise representations for regression by evolving networks of trees. In *International Conference on Learning Representations*, 2019.

[10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[11] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153:235–256, 2007.

[12] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (TOG)*, 41(2):1–21, 2021.

[13] Xiaomin Fang, Lihang Liu, Jieqiong Lei, Donglong He, Shanzhuo Zhang, Jingbo Zhou, Fan Wang, Hua Wu, and Haifeng Wang. Geometry-enhanced molecular representation learning for property prediction. *Nature Machine Intelligence*, 4(2):127–134, 2022.

[14] Thomas A Halgren. Merck molecular force field. i. basis, form, scope, parameterization, and performance of mmff94. *Journal of computational chemistry*, 17(5-6):490–519, 1996.

[15] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pages 75–102, 2006.

[16] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[17] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[18] Ying Jin, Weilin Fu, Jian Kang, Jiadong Guo, and Jian Guo. Bayesian symbolic regression. *arXiv preprint arXiv:1910.08892*, 2019.

[19] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, San Diega, CA, USA, 2015.

[21] Michael Kommenda, Bogdan Burlacu, Gabriel Kronberger, and Michael Affenzeller. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, 21(3):471–501, 2020.

[22] Stefan Kramer, Mattia Cerrato, Sašo Džeroski, and Ross King. Automated scientific discovery: From equation discovery to autonomous discovery systems. *arXiv preprint arXiv:2305.02251*, 2023.

[23] William La Cava, Thomas Helmuth, Lee Spector, and Jason H Moore. A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection. *Evolutionary Computation*, 27(3):377–402, 2019.

[24] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio de Franca, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason Moore. Contemporary symbolic regression methods and their relative performance. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[25] Greg Landrum et al. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 8:31, 2013.

[26] Wenqiang Li, Weijun Li, Linjun Sun, Min Wu, Lina Yu, Jingyi Liu, Yanjie Li, and Songsong Tian. Transformer-based model for symbolic regression via joint supervised learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[27] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[28] Risheng Liu, Pan Mu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A general descent aggregation framework for gradient-based bi-level optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):38–57, 2022.

[29] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*. PMLR, 2023.

[30] Pingchuan Ma, Tao Du, Joshua B Tenenbaum, Wojciech Matusik, and Chuang Gan. Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. In *International Conference on Learning Representations*, 2021.

[31] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024.

[32] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics, March 2022. NVIDIA GPU Technology Conference.

[33] Trent McConaghy. Ffx: Fast, scalable, deterministic symbolic regression technology. *Genetic Programming Theory and Practice IX*, pages 235–260, 2011.

[34] T. Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P. Santiago, Daniel M. Faissol, and Brenden K. Petersen. Symbolic regression via neural-guided genetic programming population seeding. In *Advances in Neural Information Processing Systems*, 2021.

[35] OpenAI. OpenAI: Introducing ChatGPT, 2022.

[36] OpenAI. OpenAI: GPT-4, 2023.

[37] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[39] Brenden K Petersen, Mikel Landajuela Larma, Terrell N Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2020.

[40] Karl Popper. *The logic of scientific discovery*. Routledge, 2005.

[41] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[42] Sereina Riniker and Gregory A Landrum. Better informed distance geometry: using what we know to improve conformation generation. *Journal of chemical information and modeling*, 55(12):2562–2574, 2015.

[43] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, pages 1–3, 2023.

[44] Alex Rosenberg and Lee McIntyre. *Philosophy of science: A contemporary introduction*. Routledge, 2019.

[45] Robert E Schapire. The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, pages 149–171, 2003.

[46] Gisbert Schneider. Automating drug discovery. *Nature reviews drug discovery*, 17(2):97–113, 2018.

[47] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research*, 257(2):395–411, 2017.

[48] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[49] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. Survey Certification.

[50] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[51] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *Advances in Neural Information Processing Systems*, 33:4860–4871, 2020.

[52] Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. Symbolicgpt: A generative transformer model for symbolic regression. *arXiv preprint arXiv:2106.14131*, 2021.

[53] Marco Virgolin, Tanja Alderliesten, and Peter AN Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In *Proceedings of the genetic and evolutionary computation conference*, pages 1084–1092, 2019.

[54] Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter AN Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary computation*, 29(2):211–237, 2021.

[55] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[56] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[57] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[58] Mignon Wuestman, Jarno Hoekman, and Koen Frenken. A typology of scientific breakthroughs. *Quantitative Science Studies*, 1(3):1203–1222, 2020.

[59] Chao Xue, Xiaoxing Wang, Junchi Yan, Yonggang Hu, Xiaokang Yang, and Kewei Sun. Rethinking bi-level optimization in neural architecture search: A gibbs sampling perspective. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 10551–10559, 2021.

[60] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

[61] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Conference on Neural Information Processing Systems*, 2023.

[62] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[63] Naruki Yoshikawa, Kei Terayama, Masato Sumita, Teruki Homma, Kenta Oono, and Koji Tsuda. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 47(11):1431–1434, 2018.

[64] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104*, 2023.

[65] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. In *International Conference on Learning Representations*, 2023.

[66] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):10752, 2019.

A Related Work

A.1 Automated Scientific Discovery

Automated scientific discovery, enhanced by machine learning methods, serves as a powerful accelerator for research, enabling scientists to generate hypotheses, design experiments, interpret vast datasets, and unearth insights that may elude traditional scientific methodologies [1, 22, 55]. This multifaceted process unfolds through two synergistically linked stages: hypothesis formation and the collection and analysis of experimental data. The integration of automated systems not only augments the scientific inquiry process but also streamlines the discovery pipeline, from conceptualization to empirical validation. This paper places a particular emphasis on, but is not limited to, constitutive

Algorithm 1 Scientific Generative Agent

```
Require: Discrete expression and continuous param ($\mathcal{E}$, $\theta \in \Theta$), Num of exploiting $M_l$, Num of exploring $M_h$, Exploiting temperature $T_l$, Exploring temperature $T_h$
 1: # Store ranked (solution,param) by heap
 2: H $\leftarrow$ heap()
 3: # Continuous optimization
 4: $\hat{\theta}$ $\leftarrow$ optim($\mathcal{E}$,$\theta$;$\Phi$)
 5: H.append(($\mathcal{E}$,$\hat{\theta}$))
 6: for $i = 1, \ldots, N$ do
 7:   # Generate $M_l$ solutions from LLM
 8:   ($\mathcal{E}$,$\Theta$)[:$M_l$] $\leftarrow$ LLM(H.topk($K$),$T_l$)
 9:   # Generate $M_h$ solutions from LLM
10:   ($\mathcal{E}$,$\Theta$)[$M_l$:$M_l$+$M_h$] $\leftarrow$ LLM(H.topk($K$),$T_h$)
11:   for $m = 1, \ldots, M_l + M_h$ do
12:     # Continuous optimization
13:     $\hat{\theta}$ $\leftarrow$ optim($\mathcal{E}$,$\theta \in \Theta$;$\Phi$)
14:     H.append(($\mathcal{E}$,$\hat{\theta}$))
15:   end for
16: end for
Ensure: H.topk(1) # Return the best
```

law discovery and molecular design. These areas exemplify the profound impact of automation in unraveling complex material behaviors and in the innovative design of molecules with tailored properties. Automatic identification of constitutive material models has been a long-standing problem and recent works utilizes differentiable simulation [12, 29, 30] to address it as a system identification problem. Leveraging machine learning and artificial intelligence, researchers are able to predict molecular behavior, optimize chemical structures for specific functions, and thus, rapidly accelerate the development of new drugs, materials, and chemicals [17, 66, 46].

A.2 Large Language Models and Agents

The advancement of Large Language Models (LLMs) such as ChatGPT and GPT-4 has sparked considerable interest in their potential as autonomous agents [6, 35, 36]. Recent developments have shown that LLMs can be enhanced to solve complex problems by creating and utilizing their own tools, as demonstrated in the LATM framework [49], and by acting as optimizers in the absence of gradients, as seen in the OPRO methodology [60]. These approaches signify a shift towards more independent and versatile LLM-based agents capable of generating solutions through self-crafted tools and optimization techniques [8, 62, 61], showcasing their evolving problem-solving capabilities. In the realm of scientific discovery, LLMs have begun to make significant contributions, particularly in mathematics and computational problems. The FunSearch method [43] pairs LLMs with evaluators to exceed known results in extremal combinatorics and online bin packing, illustrating LLMs' ability to discover new solutions to established problems. Similarly, AlphaGeometry's success [50] in solving olympiad-level geometry problems without human demonstrations highlights the potential of LLMs in automating complex reasoning tasks. These examples underline the transformative impact of LLMs in pushing the boundaries of scientific inquiry and automated reasoning.

A.3 Bilevel Optimization

Bilevel optimization involves a hierarchical structure with two levels of optimization problems, where the solution to the upper-level problem is contingent upon the outcome of the lower-level problem [11]. Bilevel optimization problems are inherently more complex than their single-level counterparts due to the nested nature of the optimization tasks and the intricate interdependencies between them. Recent advancements have focused on developing efficient algorithms, including evolutionary algorithms [48], gradient-based approaches [28], and approximation techniques [47], to tackle the computational challenges presented by the non-convex and non-differentiable characteristics of many bilevel problems. Among a wide span of application domains of bilevel optimization, neural architecture search (NAS) [27, 4, 7, 59] is prominent and close to the problem setting in this paper: the upper level optimizes the discrete neural network architecture while the lower level optimizes

Table 2: **Symbolic Regression.**

Method	R2 ↑	MSE ↓	MAE ↓	Symbolic
AIFeynman [51]	0.05105	22814675.8	2520.0	✓
DSR [39]	0.57527	10966411.0	2045.0	✓
BSR [18]	0.66526	8642965.0	1938.6	✓
AdaBoost [45]	0.75058	6439962.9	1777.7	✗
GP-GOMEA [54]	0.77734	5749076.4	1580.1	✓
SBP-GP [53]	0.81773	4706077.0	1367.5	✓
LightGBM [19]	0.83368	4294433.7	1129.9	✗
XGBoost [10]	0.87775	3156500.5	1109.2	✗
MRGP [3]	0.91074	2304682.5	950.5	✓
EPLEX [23]	0.91851	2104070.1	122.2	✓
FFX [33]	0.93124	1775263.7	801.7	✓
MLP	0.98240	454461.5	366.3	✗
FEAT [9]	0.98761	319800.6	336.1	✓
DSO [34]	0.99642	92374.9	168.6	✓
Operon [21]	0.99684	81577.9	92.4	✓
SymbolicGPT [52]	0.52333	6862154.7	1680.7	✓
NeSymReS [5]	N/A to >3 variables			✓
T-JSL [26]	N/A to >2 variables			✓
Ours	**0.99901**	**17424.6**	**86.4**	✓

the continuous weights of the neural network. However, typical NAS methods require a predefined search space, constraining the exploration of discrete network architectures to manually specified boundaries. Our framework distinguishes itself by employing LLM encoded with general knowledge and gets rid of the limitations imposed by manual design constraints.

B More Explanations

B.1 Implementation Details

We run all our experiments 5 times with different random seeds following previous practices [31]. Due to the complexity of the task, we provide a simple bootstrapping example of a valid design to ensure the success rate. We use warp [32] for the differentiable MPM simulation, and we develop our inner-level optimization upon PyTorch [38]. In all our experiments, we use mean square error as the criteria and Adam optimizer [20]. We choose `gpt-4-turbo-preview` as the backbone model for LLM and tentatively set the exploiting temperature $T_l = 0.5$ and exploring temperature $T_h = 1.0$.

For the generation of 3D conformations, we utilize the ETKGD algorithm [42] followed by optimization using the Merck Molecular Force Field (MMFF) [14], both implemented within the RDKit [25]. To get the quantum mechanical property values, we employ UniMol [65], a pre-trained transformer-based large model, which has been fine-tuned on the QM9 dataset [41].

B.2 Algorithm

We attach the full python-like pseudo-code of Scientific Generative Agent pipeline in Alg. 1.

B.3 Data Workflow

The full input to LLM has 3 main parts: (i) system prompt, (ii) iteration information, and (iii) format prompt. For the system prompt, we insert it into the LLM at the beginning or input it as a special instruction depending on the type of LLM. For the iteration information, we first concatenate the code and its feedback and then simply stack the top K solutions. Finally, we append the format prompt at the end of the prompt to regularize the expected output. From our experiments, it is important to keep the order of prompts to ensure the performance and the successful parsing. More precisely, we show this process in the following python-like code:

Table 3: **Comparison with population-based molecule design**.

Method	**(e)** $\downarrow$	**(f)** $\downarrow$	**(g)** $\downarrow$	**(h)** $\downarrow$
GhemGE	4.8e-3	1.8	1.5	9.8e-5
Ours	**1.3e-4**	**1.1e-1**	**5.4e-1**	**3.6e-5**

Table 4: **Experiment in imaginary constitutive law**.

Method	**FunSearch**	**Eureka**	**OPRO**	**Ours**
Loss	105.0	89.1	98.0	**1.3e-3**

```
prompts = []
prompts.append(system_prompt)
for solution in reversed(solutions.topk()):
    iteration_prompt = solution.code + '\n' + solution.feedback
    prompts.append(iteration_prompt)
prompts.append(format_prompt)
full_prompt = '\n'.join(prompts)
```

B.4 Differences to Symbolic Regression Task

- Our problem focuses on loss-guided general scientific discovery, which is a super-set of regular regression problems. In the constitutive law search tasks, we do not directly feed the input/output pair to our method. Instead, we consider a much more challenging task: apply the generated constitutive law recursively and use the overall loss as the performance metric. Concretely, a classic SR methods solve $\arg\min_f \|f(X) - y\|$ given $< X, y >$ pairs, whereas our method solves $\arg\min_f \|g(f(X))\|$ given $< X, g(f(X)) >$ pairs and g is a complex function like physical simulation. It is easy to construct g to cover the former case using the later formulation, proving the generality of our problem setup. We formulate our problem as such to reflect a more realistic scenario in scientific discovery, where direct supervision is extremely sparse.
- Our method supports arbitrary number of input variables and output features, where most of SR methods [52] have limitation on the number of input and output. The input limitation strongly caps the complexity of tasks they can solve, and the output limitation forces them ignore the structural correlation between each output dimension. In a comparison, our method supports arbitrary problem settings thanks to the code-based representation, which enables multi-dimensional arrays and tensor operations.
- Our model adapts to multi-discipline application easily, while traditional SR methods typically incorporate with domain-experts' priors via hard-coded constraints and heuristic [51], which is limited, domain-specific, and difficult to customize. Our method is built upon LLMs pre-trained on internet-level data that contains multi-discipline natural languages, mathematical expressions, and codes. As a result, it is easy for users to customize it and adapt to their own diverse applications via natural language guidance.

C More Experiments

C.1 Symbolic Regression

We also compare our method with traditional methods in each specific area to demonstrate the generalizability of our method. First, we reformulate our constitutive law search task **(a)** into a symbolic regression task by (i) capture the ground-truth output (the stress tensors) as the supervision, and (ii) separate the 9 output dimension into 9 independent problems and ensemble them for evaluation. Note that these modifications dramatically simplified the original task: we removed back-propagation through time (BPTT) and directly discover the constitutive law without surrogate loss. We evaluate 14 traditional baselines in SRBench [24] and 3 data-driven pre-trained baselines. As shown in Table 2, our method topped on this task even with a much more challenging setting. Also, since our method depends on the in-context learning ability of LLMs, it has little constraint in the number of variables than the data-driven pre-trained baselines.

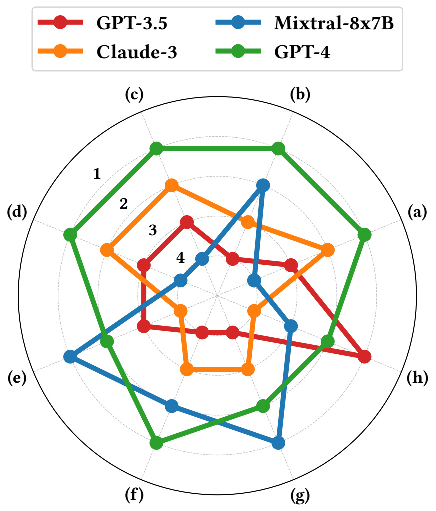


Figure 3: **Backbone LLM.**

C.2 Population-based Molecule Design

For molecule design tasks, we also compare our method with GhemGE [63], which employs a population-based molecule design algorithm. As shown in Table 3, our method presents a much lower loss, demonstrating the general effectiveness of our method.

C.3 Generalization or Memorization

In order to figure out if the improvement introduced by our method is merely because the LLM saw the solutions in its training phase, we design an experiment ablating it by making it invent an imaginary constitutive law that does not exist on the earth. We mix the constitutive law of von Mises plasticity, granular material, and weakly compressible fluid by 50%, 30%, and 20%, so that the new constitutive law represents an imaginary material whose behavior is extremely complex. We repeat our experiment setup as in Figure 1. We compare our method against the baselines and report the performances in Table 4. As shown in the table, our method can still discover the constitutive law with a low quantitative loss. From our observation, there is *very little visual difference* between the ground-truth material and the optimized constitutive law.

C.4 LLM Backbone

In addition to GPT-4 [36], we repeat the experiments in Table 1 using 3 additional LLM backbones: (i) GPT-3.5 [37], (ii) Claude-3-Sonnet [2], and (iii) Mixtral-8x7B [16], and report the rank of them in Figure 3. Indicated by the largest area, GPT-4, as our choice, statistically outperforms the other methods. Interestingly, we found Claude-3-Sonnet is the second top method on most of constitutive law search task, while Mixtral-8x7B even tops on 2 molecule design tasks. As a result, our workflow also works for other LLMs, however, our suggestion for practitioners is to try GPT-4 as the first choice but also consider open-source model (e.g., Mixtral-8x7B) for budget or customizability.

C.5 Exploitation v.s. Exploration

We visualize the statistics of the simulation execution status in Figure 4 (a) using the task **(b)**, which is one of the most challenging tasks in our experiments. When the exploitation is removed, the error rate dramatically increases, as shown by a decrease in green bars. It leads to a degeneration in the performance of the methods with exploitation as shown in Figure 4 (b). However, even though the success rate remains high, when exploration is removed, the optimization result is still worse than keeping them both. We argue that exploration is significant when the optimization problem is challenging, especially in our case, where the search space is highly non-linear and unstructured and resulting in numerous local optimum.

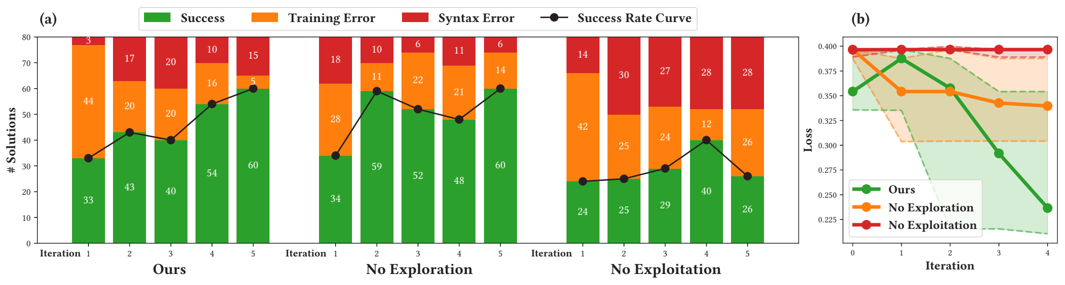


Figure 4: **Exploitation v.s. Exploration.**

Table 5: **Longer Iteration.**

#Iterations	**(a)** ↓	**(b)** ↓	**(c)** ↓	**(d)** ↓	**(e)** ↓	**(f)** ↓	**(g)** ↓	**(h)** ↓
5	5.2e-5	2.1e-1	6.0e-2	**1.4e-12**	**1.3e-4**	1.1e-1	5.4e-1	3.6e-5
20	**4.2e-6**	**4.0e-4**	**2.5e-3**	**1.4e-12**	**1.3e-4**	**6.5e-2**	**1.2e-1**	**5.6e-6**
Improvement	+1138.1%	+52400.0%	+2300.0%	0.0%	0.0%	+69.2%	+350.0%	+542.9%

C.6 Longer Iteration

In order to further investigate the potential of our method and ablate the hyper-parameters for practitioners, we add a new study in terms of the number of iterations (question-answering cycles). We repeat our experiment in Table 1 with a prolonged number of iterations to 20 and report the performance in Table 5.

As shown in the table, the number of iterations turns out to be a determining hyper-parameter with significant impart on the performance. While it has little affect on relatively easier tasks, it dramatically improves the performance of the most challenging tasks including **(b)** and **(c)**. For practitioners, the number of iteration should be first considered as the most important hyper-parameter when adapting our method to their own tasks.

D Case Study

D.1 Constitutive Law Search

We provide a trimmed snippet of our searched constitutive law in Figure 5 (a) for task **(a)** where a highly non-linear material is provided as the trajectory to fit. We reformat the code slightly to fit into the text. Starting from a linear material, our method is able to automatically generate the constitutive law with a quadratic deviatoric term. Note that our method also provides a concrete implementation of `__init__` function that defines the continuous parameters in the computational graph for later inner-level optimization.

D.2 Molecule Design

When comparing the two molecules with respect to their HOMO-LUMO energy gap based on optimized results from the LLM as shown in Figure 5 (b), we observe distinct characteristics in each: (i) **Molecule A** (gap-0) includes sulfur and chlorine atoms attached to a ring, coupled with a trifluoromethyl group, introducing electron-withdrawing effects, and (ii) **Molecule B** (gap-2) includes oxygen (notably in ethers) and sulfur within the ring structures introducing localized non-bonding electron pairs. Furthermore, the overall structure of Molecule B is more complex than that of Molecule A, containing multiple rings. An intriguing aspect of Molecule B, which might initially defy expectations, is the presence of a single fluorine atom. The high electronegativity of fluorine typically leads to electron density withdrawal, influencing the gap value. However, due to the complexity of Molecule B's structure, the impact of the fluorine atom is somewhat localized, thereby not significantly altering the gap value.

(a)

```
class Physics(nn.Module):
  def __init__(self, youngs_modulus_log: float = 13.03,
                     poissons_ratio_sigmoid: float = -1.99):
    super().__init__()
    self.youngs_modulus_log = nn.Parameter(
      torch.tensor(youngs_modulus_log))  # Log of Young's modulus
    self.poissons_ratio_sigmoid = nn.Parameter(
      torch.tensor(poissons_ratio_sigmoid))  # Sigmoid of Poisson's ratio
  def forward(self, F: torch.Tensor) -> torch.Tensor:
    youngs_modulus = self.youngs_modulus_log.exp()
    poissons_ratio = torch.sigmoid(self.poissons_ratio_sigmoid) * 0.49
    mu = youngs_modulus / (2 * (1 + poissons_ratio))  # Shear modulus
    lam = youngs_modulus * poissons_ratio / (
      (1 + poissons_ratio) * (1 - 2 * poissons_ratio))
    # Deformation gradient determinant J
    J = F.det().view(-1, 1, 1)
    F_invT = F.inverse().transpose(1, 2)
    # Volumetric part
    P_vol = lam * (J - 1) * F_invT
    # Deviatoric part
    P_dev = mu * (F - (1 / J) * F_invT)
    # Compute Kirchhoff stress tensor
    kirchhoff_stress = P_vol + P_dev @ F.transpose(1, 2)
    return kirchhoff_stress
```

(b)

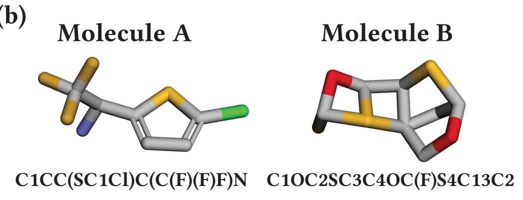


Figure 5: **Case Study.**

E Conclusion and Limitations

We consider a few limitations and future directions. (i) Although we prompt the LLM to generate pseudo-code plans and comments, it is generally hard to ensure the interpretability of LLM-generated solutions. (ii) Since the LLM-generated codes are executed directly without any filtering in our application, there exists potential AI safety risk that hazards the operating system. (iii) Our method only utilizes the internal knowledge of LLMs as the prior, where in reality people design manual constraints and rule to regularize and improve the optimization [51]. We leave these domain-specific applications and human feedback-based regularization methods as our future work. (iv) The performance our method highly depends on the differentiablity of the generated code. However, Zero-order optimizers [15] should also shine since the number of continuous parameters is relatively limited. (v) LLM inference requires large computational resources and thus increases expense. For example, it spends around $10 for our method to complete one task using GPT-4, which will be increasingly inacceptable when the number of iteration grows. (vi) Due to the reuse of previously generated solutions in our proposed top-k heap, the KV cache in LLM will be highly similar between neighbor iterations. It opens a gate for recent KV cache optimization methods [64] to speedup our method by KV cache reusing.

In conclution, we present Scientific Generative Agent, a bi-level optimization framework: LLMs serve as knowledgeable and adaptable thinkers, formulating scientific solutions like physics equations or molecule structures; concurrently, simulations operate as platforms for experimentation, offering observational feedback and optimizing continuous components like physical parameters. We focused on two scientific problems: constitutive law search and molecular design. Our approach outperforms other LLM-based benchmark methods, delivering consistent, robust, and nearly monotonic improvement. Furthermore, it shows exceptional ability in identifying unknown, true constitutive laws and molecular structures. Remarkably, our system generates innovative solutions that, despite being unconventional, are deemed reasonable after being thoroughly analyzed by experts in their respective domains. We view our process as a trailblazer, establishing a new paradigm for utilizing LLMs and simulations as bilevel optimization to further advancements in physical scientific discoveries.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Section 2 and Section 3.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix E.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release the code and data upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Table 1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors confirm that the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section E.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: See Section E.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Section 3.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This work does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.