SIMAGENTS: Bridging Literature and the Universe Via A Multi-Agent Large Language Model System

Xiaowen Zhang[♠*], **Zhenyu Bi**[♡*], **Patrick Lachance**[♠],
Xuan Wang[♡], **Tiziana Di Matteo**[♠], **Rupert A. C. Croft**[♠†]
[♠]Department of Physics, Carnegie Mellon University, Pittsburgh, PA, USA
[♡]Department of Computer Science, Virginia Tech, Blacksburg, VA, USA
[♠](xiaowen4,plachanc,tizianad)@andrew.cmu.edu, rcroft@cmu.edu,
[♡](zhenyub,xuanw)@vt.edu

Abstract

As cosmological simulations and their associated software become increasingly complex, physicists face the challenge of searching through vast amounts of literature and user manuals to extract simulation parameters from dense academic papers, each using different models and formats. Translating these parameters into executable scripts remains a time-consuming and error-prone process. To improve efficiency in physics research and accelerate the cosmological simulation process, we introduce SIMAGENTS, a multi-agent system designed to automate both parameter configuration from the literature and preliminary analysis for cosmology research. SIMAGENTS is powered by specialized LLM agents capable of physics reasoning, simulation software validation, and tool execution. These agents collaborate through structured communication, ensuring that extracted parameters are physically meaningful, internally consistent, and software-compliant. We also construct a cosmological parameter extraction evaluation dataset by collecting over 40 simulations in published papers from Arxiv and leading journals that cover diverse simulation types. Experiments on the dataset demonstrate a strong performance of SIMAGENTS, highlighting its effectiveness and potential to accelerate scientific research for physicists. Our demonstration video is available at: https://youtu.be/w1zLpm_CaWA. The complete system and dataset are publicly available at https://github.com/xwzhang98/SimAgents.

1 Introduction

Modern cosmological simulations are essential tools for advancing our understanding of the universe, enabling researchers to study the formation of galaxies and the evolution of structures. Setting up such simulations is a highly manual, time-consuming, and error-prone process. Researchers must extract parameters from dense scientific papers, convert values between units, interpret context-specific model assumptions, and then format them into executable scripts compatible with domain-specific software such as MP-GADGET (Feng et al., 2018), GADGET-4 (Springel et al., 2022), Arepo (Springel et al., 2019), GIZMO code (Hopkins, 2015) and ENZO (Bryan et al., 2014). In addition to the diversity of the simulations themselves, the complexity of using the software adds another layer of difficulty. Software user manuals are often dozens of pages long and filled with intricate rules about parameter dependencies, default settings, and strict formatting requirements.

As a result, even experienced physics researchers face a steep learning curve when trying to adopt a new simulation tool. For example, when given a cosmology paper covering several simulations, the average time cost for a human researcher to formulate the correct parameter files is in the range of hours to days, depending on the familiarity with the software. Ideally, we want this labor-intensive process to be done in minutes. The above challenges raise a crucial question: **How can we design a highly professional automated toolkit to assist cosmologists with the lengthy and complex task of setting up simulations?**

Large Language Model (LLM) agents have demonstrated significant potential on many scientific tasks (Zhao et al., 2023). Recently, researchers have proposed multi-agent reasoning frameworks that enable collaborative debates among multiple LLM agents to enhance their problem-solving abilities (Wu et al., 2023; Liang et al., 2024; Zhuge et al., 2024; Bi et al., 2025). Following this path, researchers have explored LLM-agent-based workflows on several highly professional scientific and technical applications, such as biomedical tasks and clinical tasks (Bi et al., 2024; Lu et al., 2024).

In the field of cosmology, researchers have ex-

[*]Equal contribution
[†]Corresponding author.

Proceedings of The 14th International Joint Conference on Natural Language Processing and The 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations, pages 55–66
December 20-24, 2025

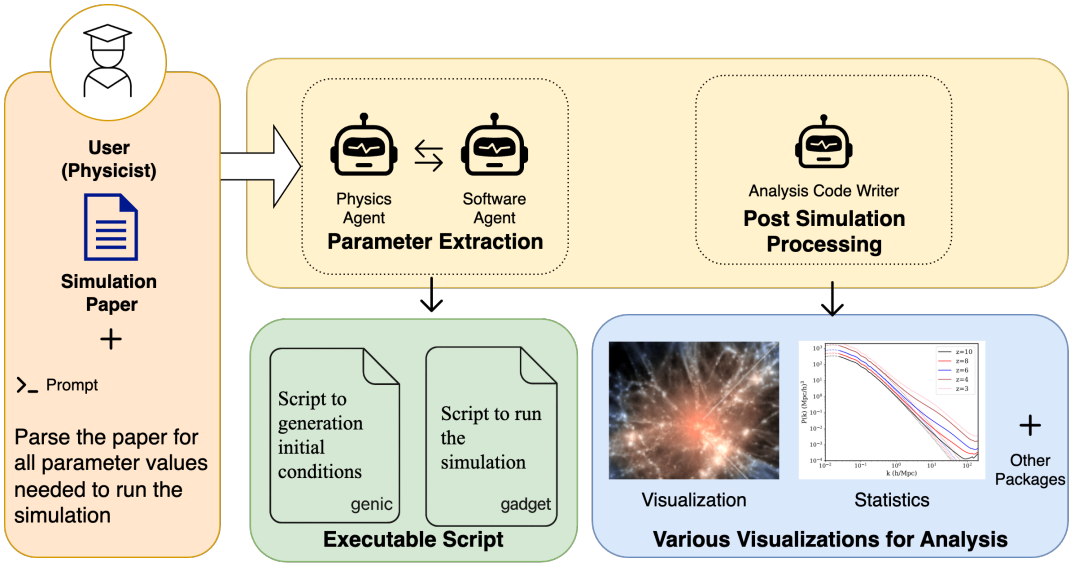

Figure 1: The workflow of our proposed multi-agent system, SIMAGENTS.

plored various LLM agent tools to provide assistance to researchers, targeting several tasks, such as a programming assistant specialized in different cosmology tasks. For example, CLAPP[1] is a single LLM agent that specializes in the CLASS cosmology code. CAMEL agents [2] provide a suite of AI-powered agents designed specifically to navigate and analyze the extensive CAMELS cosmological simulation dataset, automating tasks such as data exploration and code generation. In addition, CMBAgent (Laverick et al., 2024) and Mephisto (Sun et al., 2024) utilize a multi-agent LLM system to aid physicists in cosmological parameter analysis. Each of these systems focuses on a different scope of research, ranging from coding support to data analysis research directions. **However, to our knowledge, no prior LLM agent system automates the whole workflow from parameter configuration from the literature to initial simulation output analysis on cosmology simulation software.**

Toward this end, we introduce SIMAGENTS, a multi-agent system that automates parameter extraction, validation and configuration for cosmological simulations. The system is composed of specialized LLM agents with different distinct roles:

- **Physics Agent** that reads and interprets simulation papers using domain knowledge
- **Software Agent** that parses and enforces the constraints specified in the software user manual
- **Analysis Code Writer** that provides codes for result visualization and produces preliminary analysis (e.g. power spectra and density fields plot)

These agents collaborate through structured communication, ensuring that extracted parameters are physically meaningful, internally consistent, and software-complaint. To assess the effectiveness of SIMAGENTS, we construct a benchmark dataset of 41 simulations and evaluate the system's performance using metrics such as precision and recall, and error-specific breakdowns (e.g. Value Error, Type Error and Hallucinations). Our results show that SIMAGENTS achieves high accuracy while significantly reducing the manual workload typically required for simulation setup.

2 SIMAGENTS

In this section, we present the structure and implementations of SIMAGENTS. As illustrated in Figure 1, SIMAGENTS is composed of the following key components:

- **Parameter extraction:** This module automates the process of generating simulation scripts by extracting relevant parameters from user-uploaded papers and formatting them according to the internal requirements of the target simulation software. The extraction is performed through iterative communication between a dual-agent setup, ensuring accuracy and consistency.

[1] https://github.com/santiagocasas/clapp/
[2] https://github.com/franciscovillaescusa/CAMELS_Agents/

- **Post Simulation Processing:** This module handles code generation and execution for preliminary simulation analysis, including power spectra and density field plotting.

Together, the simulation preparation and preliminary analysis step allows users to move quickly from a published paper to actionable simulation output, closing the loop from literature reading to research insight.

2.1 Parameter Extraction

The parameter extraction module is responsible for transforming scientific papers into structured simulation-ready configurations. Given a user-uploaded paper, the system initiates a dual-agent collaboration between **Physics Agent** and **Software Agent**. The **Physics Agent** reads the input paper using domain knowledge in cosmology to identify relevant parameters such as cosmological constants, simulation box size, redshift and simulation types (dark matter, gas, stars and neutrinos). The **Software Agent** utilizes the simulation software's official user manual to query all required and optional parameters, including their default values, units, and inter-parameter dependencies.

These two agents collaborate through participation in multiple rounds of discussions based on the provided material, including a research paper and the software manual, to refine the parameter extraction process. Specifically, after Physics Agent reads the paper and extracts the parameters, the results will be sent to the Software Agent, which will then use the software user manual to check the coverage and validity of the extracted parameters. Then Software Agent will generate the parameter file following the required format and constraints. The generated file will then be sent to Physics Agent for another round of refinement. This iterative process, together with specialized task assignment on each agent, ensures:

- **High accuracy**, including scientific parameter accuracy and software requirement compliance, through task-specific expertise;
- **Modular adaptability**, as the formatting agent can be extended to support different simulation software by referencing alternative user manuals without altering the extraction logic.

2.2 Post-Simulation Processing

Once parameter extraction is completed, the generated script is passed to the simulation software for execution. After obtaining the output, the system transitions to the post-simulation processing stage, where an **Analysis Code Writer** automatically generates Python scripts to assist users with early-stage analysis of the simulation output. These generated scripts support:

- **Visualization:** Generating 2D/3D density plots from slices of the simulation box.
- **Statistical Analysis:** Generating code to plot summary statistics like matter power spectrum.
- **Custom Post-Processing:** Capability to use user-provided custom packages

The scripts are designed to be executable with minimal modification and make use of standard Python libraries such as NumPy, Matplotlib. For specialized cosmology packages, the system generates code based on example usage provided to the agent. This stage helps researchers validate simulations, identify issues early and prepare for deeper scientific investigation.

3 Experimental Setup

Our experiments are conducted in two parts: the first focuses on parameter extraction, where we evaluate quantitative accuracy; the second addresses simulation post-processing, which is more subjective and demonstrated through a representative pipeline. In our paper, we use MP-GADGET as our simulation software. In the following, we describe the experimental setup for parameter extractions.

Dataset We construct a dataset for the evaluation of cosmological parameter extraction by collecting more than 40 different simulations from published articles from ArXiv and leading journals (e.g. *ApJ*, *MNRAS*). To run MP-GADGET, two input files are required: a .genic file and a .gadget file. The .genic file generates the initial positions and velocities of particles, along with essential simulation metadata. The .gadget file evolves the initial particle distribution over time and contains numerous configuration options for selecting and enabling various physical models. Each paper is manually annotated with all MP-GADGET relevant parameter value pairs, covering cosmological parameters (Ω_m, Ω_b, Ω_Λ, h, σ_8, n_s), initial-condition settings (`BoxSize`, `Ngrid`, `Redshift`), and key model switches (e.g. `StarformationOn`, `WindOn`). To our knowledge, this is the first publicly released dataset of cos-

mological simulations with parameters derived directly from published text.

Implementation We use OpenAI GPT-4 (OpenAI et al., 2023) for our zero-shot extraction experiments. Our `SimAgents` framework utilizes the publicly available Autogen framework[3]. We also conduct an ablation study of our SIMAGENTS using the Qwen3-4B model (Yang et al., 2025). We set the temperature to 0.01 and top_p to 0.1. For the simulation software, we use MP-GADGET as an example in this paper. All outputs are formatted directly in MP-GADGET configuration syntax. We conduct all the experiments with user manual since the LLM does not have sufficient knowledge of current simulation software.

Baselines We compare our methods against two baseline methods.

- **Chain-of-thought (CoT)** (Kojima et al., 2022) We implement zero-shot CoT prompting with a single LLM agent. The agent is provided with both the literature and the manual.
- **Exchange-of-thought (EoT)** (Yin et al., 2023) We implement EoT using two agents with the same initialization, and provide them both with the literature and the manual. The agents engage in a discussion with one another.
- **SIMAGENTS** Our approach employs two task-specific agents: Physics Agent and Software Agent, each with role-specialized profiling. We provide Physics Agent with only the literature and Software Agent with only the manual. The agents engage in a discussion with one another.

We recognize that there are other LLM-based retrieval augmented generation frameworks (Gao et al., 2023). However, these RAG methods are unnecessary for our current work, as the information we provide is straightforward and does not need special design on the RAG techniques. Other LLM-based multi-agent tools in the field of cosmology (Laverick et al., 2024; Sun et al., 2024) do not fit into the scope of the current work. Thus, we do not compare with these methods in our baselines.

Evaluation We evaluate our framework using F1-score and different error metrics and provide the details of these metrics in Appendix B. Due to time constraints, we only annotated one version of the executable files. For each simulation, there exist multiple variants that contain parameters not covered in the original paper, but which could still yield the same output. To facilitate a fair comparison with the baselines, we conduct a human evaluation covering as many variants as possible and report the results in Table 1 and Table 2. The automated evaluation against the annotated dataset is reported in Section C.

[3] https://microsoft.github.io/autogen/

Method	Micro-F1	Precision	Recall
CoT (1-Agent)	93.64	92.46	94.84
EoT (2-Agent)	<u>94.95</u>	<u>93.87</u>	<u>96.05</u>
Ours (2-Agent)	**98.67**	**97.80**	**99.55**

Table 1: Performance comparison of SIMAGENTS with baseline methods on the cosmological simulation dataset. We report Micro-F1 score, Precision, and Recall as percentages. **Higher values indicate better performance.** The best-performing methods are bolded, and the second-best are underlined.

Method	Value Error	Type Error	Hallucination
CoT (1-Agent)	<u>0.97</u>	0.51	**0.21**
EoT (2-Agent)	1.21	<u>0.21</u>	0.34
Ours (2-Agent)	**0.46**	**0.02**	<u>0.30</u>

Table 2: Performance comparison of SIMAGENTS with baseline methods on the cosmological simulation dataset, in terms of average number of errors made per simulation. Each error type is reported as the average number of errors per case. **Lower values indicate better performance.** The best-performing methods are bolded, and the second-best are underlined.

4 Results

In this section, we first present the quantitative results, which contains baseline comparisons, detailed error analysis, ablation studies, and cost analysis. We then present a brief overview of the post-simulation processing capabilities of our system.

4.1 Main Results

The performance of our system compared to the baseline methods is shown in Table 1. Our proposed method, SIMAGENTS, outperforms CoT and EoT, achieving improvements of 5.03% and 3.72% in Micro-F1 score, respectively. Reduces the overall error rate by 80% compared to CoT and 70% compared to EoT, demonstrating significantly improved reliability in parameter extraction.

Comparison with Baseline Methods We examine the reasoning process of the CoT method and find that it struggles to handle excessive task instructions and information at input time, consistently making errors, and is unable to complete any tasks effectively. Simply involving multiple agents is not sufficient for optimal performance: EoT benefits from multi-agent interaction, but its lack of specialized task decomposition and clear communication structures results in imprecise outcomes. In contrast, SIMAGENTS incorporates task-specialized agents with specialized inputs, significantly reducing critical error types and leading to more accurate and robust parameter extraction.

Error Analysis In detailed error analysis, we observe that both CoT-based extraction and EoT-based extraction exhibits a higher frequency of both value error and type errors as shown in Table 2. Although our system exhibits slightly higher hallucination per case than the other baselines, these hallucinated parameters are easier to detect and filter (we provide an example in Appendix A). In contrast, value errors involve plausible-looking parameters whose values or units are subtly incorrect, often bypassing sanity checks and undetected during the simulation stage. Figure 2 shows that a single value error leads to drastically different structures, due to the different unit convention between the literature and simulation software.

4.2 Ablation study

Rounds of Discussion We conduct an ablation study to investigate the optimal number of discussion rounds between Physics Agent and Software Agent. In our parameter extraction module, each agent contributes domain-specific expertise to achieve high extraction accuracy while maintaining computational efficiency. By varying the number of discussions between these agents, we observe that two iterations yield the highest Micro-F1 score, as shown in Figure 3.

Smaller Backbone Model We also conduct experiments using Qwen3-4B as the backbone model to examine the generalizability of SIMAGENTS on Small Language Models. We provide the detailed results in Table 5 and Table 6 in Appendix C. Compared with GPT-4 which is significantly larger in model size, Qwen3-4B has inferior reasoning ability, leading to a decreased performance of an 81.23 F1 score, and an average of 3.05 value errors and 2.59 type errors per simulation.

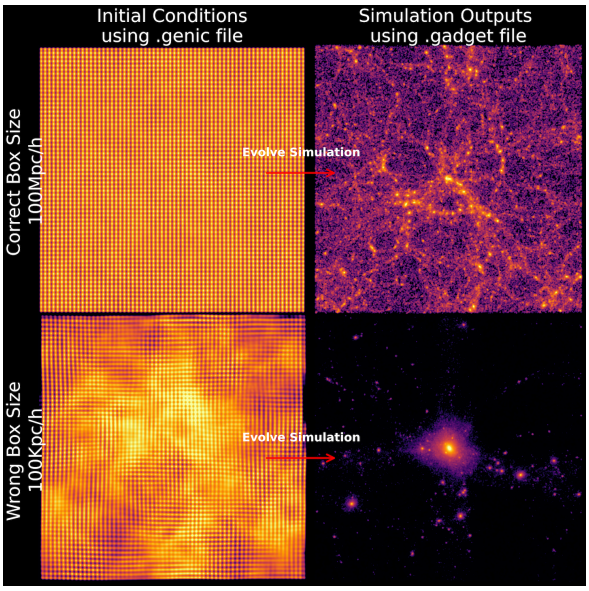

Figure 2: Impact of incorrect parameters (Value Error) on cosmological simulation outputs. Varying a single parameter, such as box size (correct top row: 100 Mpc/h; incorrect bottom row: 100 Kpc/h), while keeping all others fixed, can result in drastically different structures.

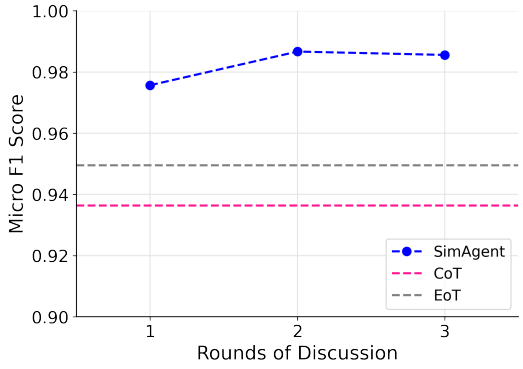

Figure 3: Results for the ablation study on the number of rounds of discussion.

4.3 Time and Cost Analysis

We conducted a survey of researchers to estimate the time cost of using simulation software. Results show that first-time users require an average of 166 minutes to replicate experiments, while experienced users average 44.4 minutes. In contrast, SIMAGENTS completes the same step in about 2 min per simulation, giving an $83\times$ speedup (-98.8%) for first-time users and $22.2\times$ (-95.5%) speed up for familiar users. At current GPT-4 API rates, a full extraction consumes around $0.25 per paper. Additionally, SIMAGENTS can run on smaller, locally executable language models with no monetary cost and an increased time cost. We report detailed numbers in Table 7 and 8 in Appendix D.

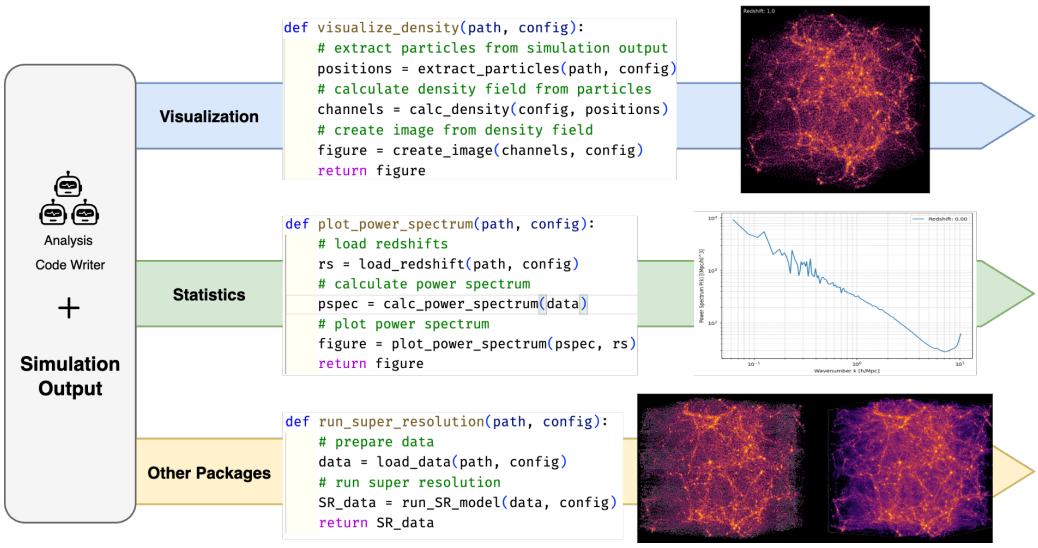

Figure 4: Illustration of post-simulation processing pipeline

4.4 Post-Simulation Processing

The output of simulation software typically consists of particle data, including positions, velocities, masses and optional quantities such as internal energy and star formation rate depending on the physical models enabled. These particles represent matter components in the universe, and the evolution over cosmic time encodes the formation of large-scale structures such as filaments, voids and halos. Some preliminary analysis are crucial for validating and interpreting simulation results:

- **Matter Power Spectrum:** Quantifies the statistical distribution of matter at different scales, sensitive to cosmological parameters such as Ω_m, σ_8, and n_s. Comparing the measured power spectrum with theoretical expectations helps to assess whether the simulation correctly reproduces the output we want.
- **Density Visualization:** Provides intuitive insight into particle distribution, particularly useful for identifying issues like incorrect box sizes or physical model settings.
- **Specialized Packages:** Generates code for specialized cosmology tools using sample code or minimal user input.

The Analysis Code Writer agent automatically provides the user with Python scripts designed to facilitate preliminary analysis of the complex and non-straightforward simulation output. As shown in Figure 4, the generated code processes the simulation output using various packages to produce the figures described above. Due to the lengthy running time of the simulation software, we were only able to perform visualization analysis and evaluation on a subset of our annotate dataset. The code provided by the Analysis Code Writer agent is highly reliable, with an execution rate of 100% in the evaluation subset.

5 Conclusions

In this paper, we propose SIMAGENTS, a multi-agent system that could accelerate physicist research for cosmological research by automatically performing parameter extraction from user-uploaded paper and simulation setup with preliminary analysis. We demonstrate the system's ability to accurately extract parameters from various simulations and translate them into valid software configuration files. Through benchmark evaluations, SIMAGENTS achieves F1 score of 98%, showing its utility in improving reproducibility, reducing human workload and accelerating the research pipeline. We envision extending SIMAGENTS to support additional simulation engines, incorporating more advanced reasoning techniques to interactively assist the researcher during post-simulation analysis. Our system and dataset are released to support further development.

Limitations

Due to time constraints, we annotated only one executable variant per paper. SIMAGENTS currently supports a small set of pretrained models and simulation codes, we will expand both datasets and coverage in the future.

Acknowledgments

This work was supported by The Block Center for Technology and Society at Carnegie Mellon University, the NSF NAIRR Pilot with PSC Neocortex and NCSA Delta, Commonwealth Cyber Initiative, Children's National Hospital, Fralin Biomedical Research Institute (Virginia Tech), Sanghani Center for AI and Data Analytics (Virginia Tech), Virginia Tech Innovation Campus, and generous gifts from Nivida, Cisco, and the Amazon + Virginia Tech Center for Efficient and Robust Machine Learning.

Ethics Statement

All models used in our system are commercially available and operated via the OpenAI API under their usage policies. No private, sensitive data were used in this paper. To ensure reproducibility and transparency, we use only publicly available papers, software and user manuals.

References

Zhenyu Bi, Sajib Acharjee Dip, Daniel Hajialigol, Sindhura Kommu, Hanwen Liu, Meng Lu, and Xuan Wang. 2024. Ai for biomedicine in the era of large language models.

Zhenyu Bi, Daniel Hajialigol, Zhongkai Sun, Jie Hao, and Xuan Wang. 2025. StoC-TOT: Stochastic tree-of-thought with constrained decoding for complex reasoning in multi-hop question answering. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 141–151, Albuquerque, New Mexico, USA. Association for Computational Linguistics.

G. L. Bryan, M. L. Norman, B. W. O'Shea, T. Abel, J. H. Wise, M. J. Turk, D. R. Reynolds, D. C. Collins, P. Wang, S. W. Skillman, B. Smith, R. P. Harkness, J. Bordner, J.-h. Kim, M. Kuhlen, H. Xu, N. Goldbaum, C. Hummels, A. G. Kritsuk, E. Tasker, S. Skory, C. M. Simpson, O. Hahn, J. S. Oishi, G. C. So, F. Zhao, R. Cen, Y. Li, and The Enzo Collaboration. 2014. ENZO: An Adaptive Mesh Refinement Code for Astrophysics. *apjs*, 211:19.

Yu Feng, Simeon Bird, Lauren Anderson, Andreu Font-Ribera, and Chris Pedersen. 2018. Mp-gadget/mp-gadget: A tag for getting a doi.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997.

Philip F. Hopkins. 2015. A new class of accurate, mesh-free hydrodynamic simulation methods. *mnras*, 450(1):53–110.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. *arXiv e-prints*, page arXiv:2205.11916.

Andrew Laverick, Kristen Surrao, Inigo Zubeldia, Boris Bolliet, Miles Cranmer, Antony Lewis, Blake Sherwin, and Julien Lesgourgues. 2024. Multi-Agent System for Cosmological Parameter Analysis. *arXiv e-prints*, page arXiv:2412.00431.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Meng Lu, Brandon Ho, Dennis Ren, and Xuan Wang. 2024. TriageAgent: Towards better multi-agents collaborations for large language model-based clinical triage. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5747–5764, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo,

Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, and Alec Radford. 2023. GPT-4 Technical Report. *arXiv e-prints*, page arXiv:2303.08774.

Volker Springel, Rüdiger Pakmor, and Rainer Weinberger. 2019. AREPO: Cosmological magnetohydrodynamical moving-mesh simulation code. Astrophysics Source Code Library, record ascl:1909.010.

Volker Springel, Rüdiger Pakmor, Oliver Zier, and Martin Reinecke. 2022. GADGET-4: Parallel cosmological N-body and SPH code. Astrophysics Source Code Library, record ascl:2204.014.

Zechang Sun, Yuan-Sen Ting, Yaobo Liang, Nan Duan, Song Huang, and Zheng Cai. 2024. Interpreting Multi-band Galaxy Observations with Large Language Model-Based Agents. *arXiv e-prints*, page arXiv:2409.14807.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-Thought: Enhancing Large Language Model Capabilities through Cross-Model Communication. *arXiv e-prints*, page arXiv:2312.01823.

Xiaowen Zhang, Patrick Lachance, Yueying Ni, Yin Li, Rupert A. C. Croft, Tiziana Di Matteo, Simeon Bird, and Yu Feng. 2024. AI-assisted super-resolution cosmological simulations III: time evolution. *mnras*, 528(1):281–293.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *arXiv e-prints*, page arXiv:2303.18223.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Language agents as optimizable graphs. *ArXiv*, abs/2402.16823.

A Example script and type of errors

A correct version of the MP-GADGET simulation script to match the low-resolution simulation in Zhang et al., 2024.

```
"genic": {
      "OutputDir": "./ICs/",
      "FileBase": "LR_100Mpc_64",
      "BoxSize": 100000.0,
      "Ngrid": 64,
      "WhichSpectrum": 2,
      "FileWithInputSpectrum": "./
    WMAP9_CAMB_matterpower.dat",
      "Omega0": 0.2814,
      "OmegaBaryon": 0.0464,
      "OmegaLambda": 0.7186,
      "HubbleParam": 0.697,
      "ProduceGas": 0,
      "Redshift": 99,
      "Seed": 12345
    }

"gadget": {
      "InitCondFile": "./ICs/
    LR_100Mpc_64",
      "OutputDir": "./output/",
      "OutputList": "0.333,1.0",
      "TimeLimitCPU": 86400,
      "MetalReturnOn": 0,
      "CoolingOn": 0,
      "SnapshotWithFOF": 0,
      "BlackHoleOn": 0,
      "StarformationOn": 0,
      "WindOn": 0,
      "MassiveNuLinRespOn": 0,
      "DensityIndependentSphOn": 0,
      "Omega0": 0.2814
    }
```

An example script with an incorrect simulation box size (Value Error), caused by a mismatch between the units used in the paper and those expected by the simulation software.

```
"genic": {
      "OutputDir": "./ICs/",
      "FileBase": "LR_100Mpc_64",
      "BoxSize": 100.0,
      "Ngrid": 64,
      "WhichSpectrum": 2,
      "FileWithInputSpectrum": "./
    WMAP9_CAMB_matterpower.dat",
      "Omega0": 0.2814,
      "OmegaBaryon": 0.0464,
      "OmegaLambda": 0.7186,
      "HubbleParam": 0.697,
      "ProduceGas": 0,
      "Redshift": 99,
      "Seed": 12345
    }
......
```

An example script containing an incorrect option that enables gas production in a dark matter only simulation (Type Error), caused by a mismatch between the paper specifications and the generated script.

```
"genic": {
      "OutputDir": "./ICs/",
      "FileBase": "LR_100Mpc_64",
      "BoxSize": 100.0,
      "Ngrid": 64,
      "WhichSpectrum": 2,
      "FileWithInputSpectrum": "./
    WMAP9_CAMB_matterpower.dat",
      "Omega0": 0.2814,
      "OmegaBaryon": 0.0464,
      "OmegaLambda": 0.7186,
      "HubbleParam": 0.697,
      "ProduceGas": 1,
      "Redshift": 99,
      "Seed": 12345
    }
    ......
```

An example of script containing an incorrect variable name that mismatch with the one in software user manual. (Hallucination)

```
"genic": {
      "OutputDir": "./ICs/",
      "FileBase": "LR_100Mpc_64",
      "BoxSize": 100.0,
      "Ngrid": 64,
      "WhichSpectrum": 2,
      "FileWithInputSpectrum": "./
    WMAP9_CAMB_matterpower.dat",
      "Omega0": 0.2814,
      "OmegaBaryon": 0.0464,
      "OmegaLambda": 0.7186,
      "HubbleParam": 0.697,
      "ProduceGas": 1,
      "Redshift": 99,
      "Seed": 12345,
      "FinalRedshift": 0
    }
......
```

B Evaluation Protocol

We define our parameter-level metrics as follows:

- **True Positives (TP):** Number of extracted parameters whose names and values are exactly correct.
- **False Positives (FP):** Number of extracted parameters with incorrect values/settings .
- **False Negatives (FN):** Number of required parameters that are missing from the extraction out-

put.

Our primary evaluation metric is the $\mathbf{F_1}$ **Score**, which captures the overall balance between precision and recall in all extracted parameter instances.

$$\mathrm{F_1} = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}.$$

Precision and recall are defined as:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

A higher $\mathrm{F_1}$ indicates more accurate extractions with fewer missing or incorrect parameters.

We categorize error cases into the following types:

- **Value Error:** The extracted parameter exists but its numerical value is incorrect. This includes errors due to unit mismatch, incorrect scaling, or misinterpretation of scientific notation.
- **Type Error:** A parameter is extracted from an incompatible simulation context. (e.g. hydrodynamic settings mistakenly used in a dark matter only simulation)
- **Hallucination:** The system outputs parameters that do not appear in the user manual, inventing values or name unsupported by the source.

Each of these types of error is reported as the average number of errors per simulation.

C Additional Experiments and Results

We provide the automatic evaluation results on SIMAGENTS and the baselines in Table 3 and Table 4. The evaluation results are slightly worse for the baselines compared to the human evaluation, as the automatic evaluation does not consider all possible executable variations of the input file. We provide the automatic evaluation results on SIMAGENTS using different backbone models in Table 5 and Table 6.

D Time and Cost Analysis

We provide the average time and cost of SIMAGENTS using GPT-4 and Qwen3-4B as the backbone model, respectively. For GPT-4, we do direct API calling; for Qwen3-4B, we run experiments on a single NVIDIA A40 GPU and report the time cost.

Method	Micro-F1	Precision	Recall
CoT (1-Agent)	<u>91.27</u>	<u>85.84</u>	<u>97.44</u>
EoT (2-Agent)	90.69	84.94	97.27
Ours (2-Agent)	**98.13**	**97.77**	**98.50**

Table 3: Performance comparison of SIMAGENTS with baseline methods on the cosmological simulation dataset. We report Micro-F1 score, Precision, and Recall as percentages. Higher values indicate better performance. The best-performing methods are bolded, and the second-best are underlined.

Method	Value Error	Type Error
CoT (1-Agent)	<u>1.76</u>	1.00
EoT (2-Agent)	1.97	<u>0.95</u>
Ours (2-Agent)	**0.40**	**0.05**

Table 4: Performance comparison of SIMAGENTS with baseline methods on the cosmological simulation dataset, in terms of average number of errors made per simulation. Each error type is reported as the average number of errors per simulation. Lower values indicate better performance. The best-performing methods are bolded, and the second-best are underlined.

E Additional Discussion and Clarifications

In this appendix, we provide additional details and clarifications in response to reviewer questions.

E.1 Manual Inspection and Physical Equivalence

For simulations that can be completed within a few days on our available compute resources, we manually inspected the outputs, focusing primarily on the matter power spectrum and density fields. For huge simulations that would require months of computation, we did not rerun the full simulations from published results. Instead, we verified that the automatically generated configurations (e.g., cosmological parameters, resolution, and activated physical modules) match those described in the corresponding publications.

As a future validation goal, we plan to move toward more systematic checks of *physical equivalence* between SIMAGENTS-generated simulations and published benchmarks. This includes extending our current limited manual inspection on smaller runs to broader, human-verified comparisons on standardized benchmark setups, once additional compute resources are available.

Method	Micro-F1	Precision	Recall
SIMAGENTS (GPT-4)	98.13	97.77	98.50
SIMAGENTS (Qwen3-4B)	81.23	70.16	96.10

Table 5: Performance comparison of SIMAGENTS using Qwen3-4B as the backbone model and GPT-4 as the backbone model. Experiments are conducted on the cosmological simulation dataset. We report Micro-F1 score, Precision, and Recall as percentages.

Method	Value Error	Type Error
SIMAGENTS (GPT-4)	0.40	0.05
SIMAGENTS (Qwen3-4B)	3.05	2.59

Table 6: Error analysis of SIMAGENTS using Qwen3-4B as the backbone model and GPT-4 as the backbone model. Experiments are conducted on the cosmological simulation dataset. Each error type is reported as the average number of errors per simulation.

	Manual	Paper/rel.	Draft+dbg	Iter	Total
First-time	64	42	60	5.2	166
Familiar	12	19	13.4	2	44.4
SIMAGENTS		*2 (total only)*			2

Table 7: Average setup effort. Times are averages in minutes. **Manual** = reading the software manual; **Paper/rel.** = reading the paper or related materials and extracting needed parameters; **Draft+dbg** = drafting and debugging the configuration; **Iter** = iterations/debug cycles to first successful run. For SIMAGENTS, only the total time applies (no per-step times).

Backbone Model	Average Time (seconds)	Average Cost ($)
GPT-4	124	**0.25**
Qwen3-4B	406	-

Table 8: Average time and cost per paper of SIMAGENTS using GPT4 and Qwen3-4B, respectively

E.2 Definition of Simulation Success

In our evaluation, success requires both executability and basic physical consistency. First, the simulation script must run to completion without errors raised by the simulation software. Second, we perform lightweight checks of physical consistency, such as verifying units and ensuring that critical physical parameters and models are set in a way that is compatible with the problem specification. These considerations are reflected in the examples and analyses presented in the main paper.

E.3 Agent Decomposition and Coordination

We did consider alternative agent decompositions when designing SIMAGENTS. The current two-agent setup is chosen to balance information distribution and domain expertise: both agents are prompted to use physics knowledge, while the Software Agent focuses on interacting with the code and its manual. In a two-agent interaction, there is limited room for different coordination strategies and patterns in a two-agent interaction. We will explore this in future work as we implement more varieties of agent groups.

E.4 Generalizability to Other Simulation Codes

Our framework is designed to be largely adaptable to different simulation codes. Many widely used codes (e.g., Arepo, ENZO, GIZMO, GADGET-4) share similar high-level physical models and workflows (configuration $\rightarrow$ initial conditions $\rightarrow$ run $\rightarrow$ analysis), but differ in file structures, parameter names, units, and other conventions.

In principle, the overall multi-agent structure of SIMAGENTS can be reused across codes. Adapting to a new code primarily requires:

- Providing the Software Agent with the corresponding manuals and documentation.
- Adding code-specific guidance about file formats, execution commands, and key parameters.
- Performing modest prompt engineering to account for different naming conventions, error messages, and pipeline structures.

Thus, extending SIMAGENTS to other simulation environments is not a matter of re-engineering the entire framework, but of combining new documentation with few adaptations.

E.5 Model Dependency and Smaller Open-Source Models

We observe a substantial performance gap between GPT-4 and the smaller open-source model Qwen3-4B (F1 score dropping from 98.13% to 81.23%). A key limitation of Qwen3-4B in our setting is its weaker long-context reasoning ability: given very long inputs such as a $\sim$100-page software manual, it struggles to fully interpret and integrate the necessary information. Thus, giving it more examples

would not be that helpful, as we are feeding it with more contexts, which are usually dozens of pages of physics papers. Fine-tuning on the dataset could be beneficial, but it would be very time-consuming and dataset-specific. We will explore light fine-tuning in later works.

E.6 Error Analysis and Hallucination Errors

As noted in Section 3 of the main paper, all baselines, including SIMAGENTS, have access to the user manual, since current LLMs do not possess sufficient built-in knowledge of cosmological simulation software. In SIMAGENTS, we prompt the specialized Software Agent to focus particularly on parameter explanations rather than just parameter names, because type and value errors are especially critical from a physicist's perspective. As a result, many of the hallucination errors made are errors that a physicist can understand which physical parameter they correspond to. Still, the naming is different from that of the software. We expect these errors to be reducible by prompting the agents to pay closer attention to exact parameter names during inter-agent communication and by strengthening consistency checks between proposed configurations and the documentation. Such improvements are a natural direction for future iterations of the framework.