An LLM-driven framework for cosmological model-building and exploration

Nayantara Mudur
Harvard University, Cambridge, MA, USA
The NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Harvard-Smithsonian Center for Astrophysics, Cambridge, MA, USA
nmudur@g.harvard.edu

Carolina Cuesta-Lazaro
The NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Massachusetts Institute of Technology, Cambridge, MA, USA
Harvard-Smithsonian Center for Astrophysics, Cambridge, MA, USA
cuestalz@mit.edu

Michael W. Toomey
Center for Theoretical Physics – a Leinweber Institute
Massachusetts Institute of Technology, Cambridge, MA, USA
mtoomey@mit.edu

Douglas P. Finkbeiner
Harvard University, Cambridge, MA, USA
The NSF AI Institute for Artificial Intelligence and Fundamental Interactions
Harvard-Smithsonian Center for Astrophysics, Cambridge, MA, USA
dfinkbeiner@cfa.harvard.edu

Abstract

Our understanding of cosmic evolution relies on dark energy and dark matter—mysterious components detectable only through gravitational effects despite comprising 95% of the Universe today. Recent surveys reveal systematic discrepancies in dark energy's temporal evolution, potentially indicating new physics. Given Large Language Models' success at research-level coding and mathematical reasoning, we investigate LLMs' capability to autonomously propose, implement, and test cosmological models. We challenge an agentic LLM (Claude Code) in three settings: (1) implementing alternative cosmological models from curated descriptions by modifying a physics simulation codebase, (2) implementing those from research papers directly, and (3) generating novel dark energy hypotheses to explain recent observations. When given curated descriptions with numerical implementation tips, the agent successfully implements both test models—"Thawing Quintessence" and "Early Dark Energy"—with different numerical accuracy levels reflecting the model complexity. Working from papers rather than curated descriptions significantly degrades numerical accuracy though qualitative behavior remains correct. Interestingly, the agent's self-proposed dark energy model achieved comparable observational fits to the standard model, though requiring additional parameters.

Preprint number: MIT-CTP/5945

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

1 Introduction

Cosmology—the study of how the Universe evolved from the Big Bang to today—faces a double crisis. The Lambda Cold Dark Matter (ΛCDM) model has achieved remarkable success in predicting observations, from the first atoms to the clustering of galaxies today, but relies on “dark matter" and “dark energy"—mysterious components detectable only through gravity despite comprising 95% of the Universe today. Beyond this theoretical issue also lies an empirical problem: even accepting dark matter and dark energy as given, ΛCDM appears to increasingly struggle to fit high-precision data. With recent surveys hinting at further discrepancies [1, 2], revealing the true nature of these dark components by exploring alternative models is an urgent priority.

Resolving these tensions requires exploring a vast landscape of theoretical models that propose alternative forms of dark matter and dark energy, such as modified gravity theories, interacting dark sectors, and dynamical dark energy models, each introducing new parameters and assumptions that create a high-dimensional space largely unexplored due to computational and human limitations. Large language models, trained on vast theoretical physics literature, could systematically sample this hypothesis space, generating novel combinations human researchers might not consider.

In practice, testing cosmological models requires specialized simulation codes, like CLASS [3] and CAMB [4], that take theoretical parameters as input and produce synthetic observational data as output. The canonical workflow modifies these codebases for new physics, then evaluates whether proposed models explain data better than ΛCDM through a Bayesian analysis.

Model building in cosmology makes for a compelling yet challenging LLM application. Firstly, almost all papers implementing an alternate cosmological model rely on 2-4 codebases. Few subfields in the sciences have such a *unified workflow*. Secondly, these alternate theories must pass a set of unified tests that evaluate their ability to reproduce observations. From a machine-learning perspective, this means that cosmological model discovery can be reformulated as an optimization problem with *well-defined metrics*. Lastly, beyond implementation, this field offers an intriguing test bed to ask whether recent developments in artificial intelligence could help us discover *new physics*.

In this work, we introduce a multi-stage framework to evaluate the ability of an LLM-driven agent to autonomously propose, implement and explore cosmological models.

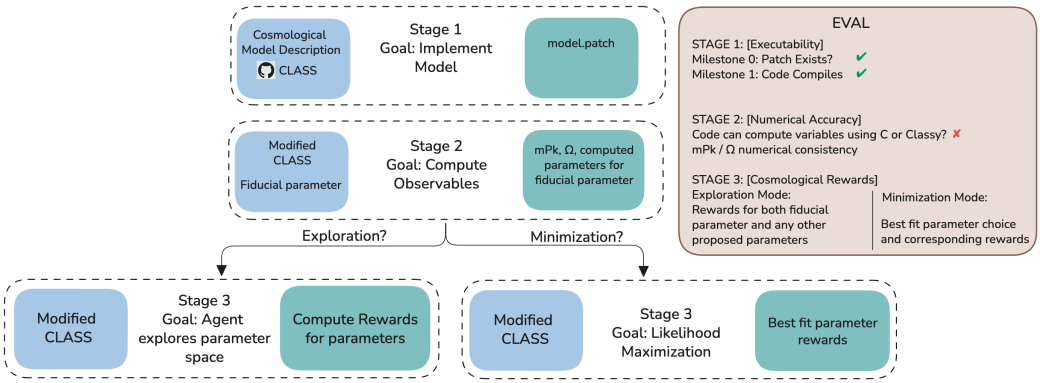

Figure 1: Flow diagram depicting the input state for each stage (blue) the output artifacts (green) and the task that the agent has to perform. Once the agent has completed its work, the outputs are automatically evaluated on the basis of both binary milestones that track initial progress as well as numerical metrics. The evaluation box (brown) depicts a template evaluation report for the agent’s ’submission’.

2 Framework and Evaluation setup

The agent is entasked with modifying the CLASS codebase [3] for cosmological simulations[1]. CLASS is implemented in C with a Python wrapper (Classy) for computing observables. The agent

[1]Available at `https://github.com/lesgourg/class_public`, see also interesting agent developments for CLASS usage at [5]

operates in a containerized environment containing the CLASS repository (as a GitHub repo to enable the modified code to be represented and exported as a patch), stage-specific instruction prompts, CLASS documentation, model descriptions, and auxiliary testing code. We use the Claude Code agentic framework [6] with the `claude-sonnet-4-20250514` model [7]. See Appendix A.2for a detailed description. Given CLASS's complexity (over 43,000 lines of C code plus 10,000 lines of numerical files), we provide three file maps summarizing the organization of the background, input, and perturbations modules.

2.1 Cosmological Model Specification

We use $\mathcal{M}$ to denote a theoretical model and $\mathcal{P}$ a point in parameter space for the specific model. Each model instance consists of: (1) a **Model Description** in either of two formats—curated `description.md` files with a detailed description of the model and in depth implementation tips (see Appendix B.2.1for an example) or original research papers that require extraction of relevant information from the full text; (2) **Fiducial Parameters** $\mathcal{P}_{\text{Fiducial}}$ used to test whether the modified code can actually be used to compute cosmological observables and to facilitate quantitative comparisons with target implementations of the model (when one exists); and (3) a **Context File** [`problem_context.yaml`] with problem-specific contextual instructions or variables.

2.2 Defining the Agent's Task Sequence

We split the agent's tasks into three sequential stages. In **Stage 1 (Model Implementation)**, the agent is entasked with reading the model description and modifying the CLASS codebase to implement the cosmological model $\mathcal{M}$. The stage is successful when the code compiles and implements the target model.

Stage 2 (Computing Observables) requires the agent to use its modified code to compute cosmological observables (e.g., power spectra) for $\mathcal{P}_{\text{fiducial}}$ using a provided helper script. If execution fails, we query another LLM (`o1-2024-12-17`) [8] to compare the model description with the generated patch and identify inconsistencies, functioning as an LLM-as-Judge setup. Success produces a file containing the observables.

In **Stage 3 (Parameter Exploration and Rewards)**, we evaluate the implemented cosmological model's agreement with observations and potential to resolve cosmological tensions through a set of rewards. These rewards include the Hubble constant discrepancy between early-Universe (*Planck* CMB [9]) and late-Universe (supernovae) measurements, and recent DESI findings suggesting evolving dark energy that deviates from the cosmological constant of ΛCDM. Our data likelihood combines three components: DESI baryonic acoustic oscillation (BAO) measurements [1], an early Universe (CMB) 3-parameter prior [1], and Pantheon+ supernovae data [10] with $z > 0.01$ (see Appendix C.2.1 for an in depth description). Rather than sampling the full posterior over the model parameters, we adopt two approaches depending on model complexity. For models amenable to optimization, we use the *Minimization* mode to find maximum likelihood parameters $\mathcal{P}_{\text{Best-Fit}}$ via numerical minimization. For models with challenging parameter degeneracies (e.g., Early Dark Energy), we use *Exploration*, where the agent manually proposes and tests alternative parameter combinations.

For any given parameter set, we compute the following rewards: (1) **Data likelihood** - how well the model explains the observations, together with the Bayesian Information Criterion (BIC), to penalize overfitting via extra parameters; (2) **Hubble tension** - whether the model alleviates the tension by comparing the predicted H_0 to SH0ES measurements [11], or alternatively comparing M_B values as a more robust late-time diagnostic (see Appendix C.2.2 for a more in depth explanation of the two metrics); (3) **CMB power spectrum agreement** - maximum deviation of predicted CMB power spectrum, C_l, from the Planck2018 C_ls in cosmic variance units, serving as a consistency check.

2.3 Evaluating the agent's capabilities to implement new physics in CLASS

After passing 3 binary execution milestones (it generated a patch, that compiles, and that can be used to compute observables), we compare the agent's implementation to those generated by a ground truth repository implementing the same cosmological model, focusing on *correctness*. We numerically compare computed physical quantities (the power spectra and the evolution of the dark energy field)

and report the maximum fractional deviation with respect to the target repository in Table 1. We also report the Stage 3 rewards for each of the parameters.

3 Results

3.1 Assessing the LLMs capabilities to implement new physics in CLASS

In Table 1, we examine the performance of Claude Code on two cosmological models: in increasing order of difficulty, a phenomenological *Thawing Quintessence* (TQ)[12] model that mimics the qualitative behavior of (late-time) dark energy, and (2) an *Early Dark Energy* (EDE) model [13–17], a particle physics inspired extension to standard cosmology. Both models have ground truth implementations available[2]. Note that the implementation of *EDE* is considerably more complex than that of *Thawing Quintessence* as reflected by the patch lengths.

Table 1: Target Artifact-based Evaluation Results

Model	EDE [curated]	TQ [curated]	TQ [paper]
Execution milestones passed	✓	✓	✓
Max frac. dev: Ω_X/mPk(0)/mPk(1)	0.026/0.116/0.110	$2.2e-7$/$1.1e-4$/$7.6e-5$	0.248/0.100/0.027
$\ln\mathcal{L}\uparrow$	558	830	830
BIC $\downarrow$	N/A	-1616	-1616
H_0 tension $\downarrow$	4.12σ [67.94]	4.73σ [67.19]	3.57σ [68.61]
M_B tension $\downarrow$	N/A	5.01σ [-19.42]	5.03σ [-19.42]

Table 2: LLM-proposed Model Results

Model Context	LCDM Reference	EDE-RC One-Shot	ODEM +Summary
Execution milestones passed	N/A	✓	✓
$\ln\mathcal{L}\uparrow$	831	828	828
BIC $\downarrow$	-1640	-1604	-1612
H_0 tension $\downarrow$	4.22σ [67.82]	2.75σ [69.63]	3.70σ [68.46]
M_B tension $\downarrow$	5.25σ [-19.43]	4.69σ [-19.41]	4.70σ [-19.41]

- **Starting from a curated model description** (`description.md`). We find that the model successfully implements the *Thawing Quintessence* model with a high numerical precision, whereas it reproduces the EDE results with lower accuracy. Note that the agent required guidance on unit conversion and numerical tips for implementing the EDE model. In future work we will test specific ablations of the curated model descriptions.
- **'Paper to Code'**. For *Thawing Quintessence*, we ran the same experiment after swapping the description.md file with the tex file of the corresponding paper [12], to examine whether the agent is still able to generate working code given just the paper, as opposed to a well-specified problem description. The agent is still able to generate executable code. However, the comparison against the target artifacts reveals significantly higher numerical inconsistencies relative to the model description. This is in part because the model omits a change to the code where it needs to explicitly rewrite the numerical integration prescription for this model for increased numeral precision. These differences do not seem to impact the best fit likelihood values of the models (Table 1).

3.2 Hypothesis Generation

Thus far, our experiments measured the ability of the agent to implement a *given* cosmological model $\mathcal{M}$. In this section, we further examine whether it is capable of developing its *own* theoretical model, and implement it by modifying CLASS.

[2]But known to be public only for the EDE model.

In the first experiment *One-Shot*, the model is instructed to read an auxiliary prompt and propose its *own novel* cosmological model, prioritizing (1) novelty, (2) the potential to alleviate cosmological tensions and provide a good fit to data, and (3) ease of implementation in CLASS. We additionally provide it with an example description and example parameter file from the *Thawing Quintessence* problem. It must first create its own `description.md` file describing its proposed cosmological model, $\mathcal{M}$ (see Appendix D.3for examples of generated model descriptions), and a corresponding $\mathcal{P}_{Fiducial}$, and then proceed with the remaining stages. In the second experiment, *One-Shot + Summary*, we additionally provide the agent an LLM (`gemini-2.5-pro`) generated summary of five recent papers [18–25] from the literature that explored alternate cosmological models that may explain recent DESI findings.

In the first experiment, the agent proposed an *Early Dark Energy with Running Coupling (EDE-RC)* model with

$$\rho_{\text{ede-rc}}(a) = \rho_{\text{ede,0}} \left(\frac{a_c}{a}\right)^{3(1+w_0)} \exp\left[-\alpha_{\text{run}} \ln^2\left(\frac{a}{a_c}\right)\right] \tag{1}$$

where a_c is the characteristic scale factor at field peak, w_0 the base equation of state, α_{run} controls transition sharpness, and $\rho_{\text{ede,0}}$ sets normalization.

In the second experiment, the agent proposed an *Oscillatory Dark Energy with Memory (ODEM)* model, with

$$w(a) = -1 + A(a) \sin\left(\frac{2\pi \ln(a)}{\Delta \ln a}\right) e^{-\frac{\ln(a/a_m)}{\tau_m}} \tag{2}$$

where $A(a) = A_0(a/a_0)^\alpha$ is the evolving oscillation amplitude, $\Delta \ln a$ controls oscillation period, a_m is the memory scale factor, and τ_m the memory decay timescale. We note that the hypothesis generated are of an empirical nature, and not related to the physical field properties of dark energy. In future work, we will incorporate mechanisms that ensure the model avoids empirical parametrisations.

In two other hypothesis generation experiments with different contextual information / literature available the agent was unable to generate code that crossed all binary execution milestones successfully.

4 Conclusions

Our results demonstrate both the promise and limitations of LLM-driven scientific model discovery, providing a concrete framework for evaluating AI agents on complex scientific workflows that combine domain knowledge [26], mathematical reasoning [27], coding [28], and empirical validation. We found that while the agent produced reasonable implementations when provided broad context, numerical accuracy was challenging to attain without expert-crafted prompts.

Moving forward, we plan to expand our approach in two key directions to enable more systematic and principled exploration of the theoretical landscape. First, we will implement evolutionary algorithm-inspired approaches that use the computed BIC to iteratively guide the discovery of improved dark energy models in a sequential manner. Rather than exploring fixed parameterizations, this approach will enable the LLM to evolve functional forms for scalar field potentials and dark energy evolution, using reward signals to steer toward models that better explain observations. We will extend this framework to incorporate interactions between dark energy and dark matter sectors—a theoretically motivated class of models that could address multiple cosmological tensions simultaneously.

Acknowledgments and Disclosure of Funding

This work was supported by the National Science Foundation under Cooperative Agreement PHY2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions). This research used data obtained with the Dark Energy Spectroscopic Instrument (DESI). DESI construction and operations is managed by the Lawrence Berkeley National Laboratory. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of High-Energy Physics, under Contract No. DE–AC02–05CH11231, and by the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility under the same contract. Additional support for DESI was provided by the U.S. National Science Foundation (NSF), Division of Astronomical Sciences under Contract No. AST-0950945 to the NSF's National Optical-Infrared Astronomy Research Laboratory; the Science and Technology Facilities Council of the United Kingdom; the Gordon

and Betty Moore Foundation; the Heising-Simons Foundation; the French Alternative Energies and Atomic Energy Commission (CEA); the National Council of Humanities, Science and Technology of Mexico (CONAHCYT); the Ministry of Science and Innovation of Spain (MICINN), and by the DESI Member Institutions: www.desi.lbl.gov/collaborating-institutions. The DESI collaboration is honored to be permitted to conduct scientific research on I'oligam Du'ag (Kitt Peak), a mountain with particular significance to the Tohono O'odham Nation. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics of U.S. Department of Energy under grant Contract Number DE-SC0012567. M. W. T. acknowledges financial support from the Simons Foundation (Grant Number 929255). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. National Science Foundation, the U.S. Department of Energy, or any of the listed funding agencies.

References

[1] DESI Collaboration, M. Abdul-Karim, J. Aguilar, S. Ahlen, S. Alam, L. Allen, C. Allende Prieto, O. Alves, A. Anand, U. Andrade, E. Armengaud, A. Aviles, S. Bailey, C. Baltay, P. Bansal, A. Bault, J. Behera, S. BenZvi, D. Bianchi, C. Blake, S. Brieden, A. Brodzeller, D. Brooks, E. Buckley-Geer, E. Burtin, R. Calderon, R. Canning, A. Carnero Rosell, P. Carrilho, L. Casas, F. J. Castander, R. Cereskaite, M. Charles, E. Chaussidon, J. Chaves-Montero, D. Chebat, X. Chen, T. Claybaugh, S. Cole, A. P. Cooper, A. Cuceu, K. S. Dawson, A. de la Macorra, A. de Mattia, N. Deiosso, J. Della Costa, R. Demina, A. Dey, B. Dey, Z. Ding, P. Doel, J. Edelstein, D. J. Eisenstein, W. Elbers, P. Fagrelius, K. Fanning, E. Fernández-García, S. Ferraro, A. Font-Ribera, J. E. Forero-Romero, C. S. Frenk, C. Garcia-Quintero, L. H. Garrison, E. Gaztañaga, H. Gil-Marín, S. Gontcho A Gontcho, D. Gonzalez, A. X. Gonzalez-Morales, C. Gordon, D. Green, G. Gutierrez, J. Guy, B. Hadzhiyska, C. Hahn, S. He, M. Herbold, H. K. Herrera-Alcantar, M. Ho, K. Honscheid, C. Howlett, D. Huterer, M. Ishak, S. Juneau, N. V. Kamble, N. G. Karaçaylı, R. Kehoe, S. Kent, A. G. Kim, D. Kirkby, T. Kisner, S. E. Koposov, A. Kremin, A. Krolewski, O. Lahav, C. Lamman, M. Landriau, D. Lang, J. Lasker, J. M. Le Goff, L. Le Guillou, A. Leauthaud, M. E. Levi, Q. Li, T. S. Li, K. Lodha, M. Lokken, F. Lozano-Rodríguez, C. Magneville, M. Manera, P. Martini, W. L. Matthewson, A. Meisner, J. Mena-Fernández, A. Menegas, T. Mergulhão, R. Miquel, J. Moustakas, A. Muñoz-Gutiérrez, D. Muñoz-Santos, A. D. Myers, S. Nadathur, K. Naidoo, L. Napolitano, J. A. Newman, G. Niz, H. E. Noriega, E. Paillas, N. Palanque-Delabrouille, J. Pan, J. Peacock, Marcos Pellejero Ibanez, W. J. Percival, A. Pérez-Fernández, I. Pérez-Ràfols, M. M. Pieri, C. Poppett, F. Prada, D. Rabinowitz, A. Raichoor, C. Ramírez-Pérez, M. Rashkovetskyi, C. Ravoux, J. Rich, A. Rocher, C. Rockosi, J. Rohlf, J. O. Román-Herrera, A. J. Ross, G. Rossi, R. Ruggeri, V. Ruhlmann-Kleider, L. Samushia, E. Sanchez, N. Sanders, D. Schlegel, M. Schubnell, H. Seo, A. Shafieloo, R. Sharples, J. Silber, F. Sinigaglia, D. Sprayberry, T. Tan, G. Tarlé, P. Taylor, W. Turner, L. A. Ureña-López, R. Vaisakh, F. Valdes, G. Valogiannis, M. Vargas-Magaña, L. Verde, M. Walther, B. A. Weaver, D. H. Weinberg, M. White, M. Wolfson, C. Yèche, J. Yu, E. A. Zaborowski, P. Zarrouk, Z. Zhai, H. Zhang, C. Zhao, G. B. Zhao, R. Zhou, and H. Zou. Desi dr2 results ii: Measurements of baryon acoustic oscillations and cosmological constraints, 2025. URL `https://arxiv.org/abs/2503.14738`.

[2] K. Lodha, R. Calderon, W. L. Matthewson, A. Shafieloo, M. Ishak, J. Pan, C. Garcia-Quintero, D. Huterer, G. Valogiannis, L. A. Ureña-López, N. V. Kamble, D. Parkinson, A. G. Kim, G. B. Zhao, J. L. Cervantes-Cota, J. Rohlf, F. Lozano-Rodríguez, J. O. Román-Herrera, M. Abdul-Karim, J. Aguilar, S. Ahlen, O. Alves, U. Andrade, E. Armengaud, A. Aviles, S. BenZvi, D. Bianchi, A. Brodzeller, D. Brooks, E. Burtin, R. Canning, A. Carnero Rosell, L. Casas, F. J. Castander, M. Charles, E. Chaussidon, J. Chaves-Montero, D. Chebat, T. Claybaugh, S. Cole, A. Cuceu, K. S. Dawson, A. de la Macorra, A. de Mattia, N. Deiosso, R. Demina, Arjun Dey, Biprateep Dey, Z. Ding, P. Doel, D. J. Eisenstein, W. Elbers, S. Ferraro, A. Font-Ribera, J. E. Forero-Romero, Lehman H. Garrison, E. Gaztañaga, H. Gil-Marín, S. Gontcho A Gontcho, A. X. Gonzalez-Morales, G. Gutierrez, J. Guy, C. Hahn, M. Herbold, H. K. Herrera-Alcantar, K. Honscheid, C. Howlett, S. Juneau, R. Kehoe, D. Kirkby, T. Kisner, A. Kremin, O. Lahav, C. Lamman, M. Landriau, L. Le Guillou, A. Leauthaud, M. E. Levi, Q. Li, C. Magneville, M. Manera, P. Martini, A. Meisner, J. Mena-Fernández, R. Miquel, J. Moustakas, D. Muñoz Santos, A. Muñoz-Gutiérrez, A. D. Myers, S. Nadathur, G. Niz, H. E. Noriega, E. Paillas, N. Palanque-Delabrouille,

W. J. Percival, Matthew M. Pieri, C. Poppett, F. Prada, A. Pérez-Fernández, I. Pérez-Ràfols, C. Ramírez-Pérez, M. Rashkovetskyi, C. Ravoux, A. J. Ross, G. Rossi, V. Ruhlmann-Kleider, L. Samushia, E. Sanchez, D. Schlegel, M. Schubnell, H. Seo, F. Sinigaglia, D. Sprayberry, T. Tan, G. Tarlé, P. Taylor, W. Turner, M. Vargas-Magaña, M. Walther, B. A. Weaver, M. Wolfson, C. Yèche, P. Zarrouk, R. Zhou, and H. Zou. Extended dark energy analysis using desi dr2 bao measurements, 2025. URL https://arxiv.org/abs/2503.14743.

[3] Julien Lesgourgues. The cosmic linear anisotropy solving system (class) i: Overview, 2011. URL https://arxiv.org/abs/1104.2932.

[4] Antony Lewis and Anthony Challinor. CAMB: Code for Anisotropies in the Microwave Background. Astrophysics Source Code Library, record ascl:1102.026, February 2011.

[5] Andrew Laverick, Kristen Surrao, Inigo Zubeldia, Boris Bolliet, Miles Cranmer, Antony Lewis, Blake Sherwin, and Julien Lesgourgues. Multi-Agent System for Cosmological Parameter Analysis, 11 2024.

[6] Anthropic. Claude code, . https://docs.anthropic.com/en/docs/claude-code/overview.

[7] Anthropic. Claude sonnet 4, . https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf.

[8] OpenAI. o1. https://openai.com/index/openai-o1-system-card/.

[9] N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, R. B. Barreiro, N. Bartolo, S. Basak, R. Battye, K. Benabed, J.-P. Bernard, M. Bersanelli, P. Bielewicz, J. J. Bock, J. R. Bond, J. Borrill, F. R. Bouchet, F. Boulanger, M. Bucher, C. Burigana, R. C. Butler, E. Calabrese, J.-F. Cardoso, J. Carron, A. Challinor, H. C. Chiang, J. Chluba, L. P. L. Colombo, C. Combet, D. Contreras, B. P. Crill, F. Cuttaia, P. de Bernardis, G. de Zotti, J. Delabrouille, J.-M. Delouis, E. Di Valentino, J. M. Diego, O. Doré, M. Douspis, A. Ducout, X. Dupac, S. Dusini, G. Efstathiou, F. Elsner, T. A. Enßlin, H. K. Eriksen, Y. Fantaye, M. Farhang, J. Fergusson, R. Fernandez-Cobos, F. Finelli, F. Forastieri, M. Frailis, A. A. Fraisse, E. Franceschi, A. Frolov, S. Galeotta, S. Galli, K. Ganga, R. T. Génova-Santos, M. Gerbino, T. Ghosh, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gruppuso, J. E. Gudmundsson, J. Hamann, W. Handley, F. K. Hansen, D. Herranz, S. R. Hildebrandt, E. Hivon, Z. Huang, A. H. Jaffe, W. C. Jones, A. Karakci, E. Keihänen, R. Keskitalo, K. Kiiveri, J. Kim, T. S. Kisner, L. Knox, N. Krachmalnicoff, M. Kunz, H. Kurki-Suonio, G. Lagache, J.-M. Lamarre, A. Lasenby, M. Lattanzi, C. R. Lawrence, M. Le Jeune, P. Lemos, J. Lesgourgues, F. Levrier, A. Lewis, M. Liguori, P. B. Lilje, M. Lilley, V. Lindholm, M. López-Caniego, P. M. Lubin, Y.-Z. Ma, J. F. Macías-Pérez, G. Maggio, D. Maino, N. Mandolesi, A. Mangilli, A. Marcos-Caballero, M. Maris, P. G. Martin, M. Martinelli, E. Martínez-González, S. Matarrese, N. Mauri, J. D. McEwen, P. R. Meinhold, A. Melchiorri, A. Mennella, M. Migliaccio, M. Millea, S. Mitra, M.-A. Miville-Deschênes, D. Molinari, L. Montier, G. Morgante, A. Moss, P. Natoli, H. U. Nørgaard-Nielsen, L. Pagano, D. Paoletti, B. Partridge, G. Patanchon, H. V. Peiris, F. Perrotta, V. Pettorino, F. Piacentini, L. Polastri, G. Polenta, J.-L. Puget, J. P. Rachen, M. Reinecke, M. Remazeilles, A. Renzi, G. Rocha, C. Rosset, G. Roudier, J. A. Rubiño-Martín, B. Ruiz-Granados, L. Salvati, M. Sandri, M. Savelainen, D. Scott, E. P. S. Shellard, C. Sirignano, G. Sirri, L. D. Spencer, R. Sunyaev, A.-S. Suur-Uski, J. A. Tauber, D. Tavagnacco, M. Tenti, L. Toffolatti, M. Tomasi, T. Trombetti, L. Valenziano, J. Valiviita, B. Van Tent, L. Vibert, P. Vielva, F. Villa, N. Vittorio, B. D. Wandelt, I. K. Wehus, M. White, S. D. M. White, A. Zacchei, and A. Zonca. Planck2018 results: Vi. cosmological parameters. *Astronomy amp; Astrophysics*, 641:A6, September 2020. ISSN 1432-0746. doi: 10.1051/0004-6361/201833910. URL http://dx.doi.org/10.1051/0004-6361/201833910.

[10] Dillon Brout, Dan Scolnic, Brodie Popovic, Adam G Riess, Anthony Carr, Joe Zuntz, Rick Kessler, Tamara M Davis, Samuel Hinton, David Jones, et al. The pantheon+ analysis: cosmological constraints. *The Astrophysical Journal*, 938(2):110, 2022.

[11] Adam G Riess, Wenlong Yuan, Lucas M Macri, Dan Scolnic, Dillon Brout, Stefano Casertano, David O Jones, Yukei Murakami, Gagandeep S Anand, Louise Breuval, et al. A comprehensive measurement of the local value of the hubble constant with 1 km s- 1 mpc- 1 uncertainty from

the hubble space telescope and the sh0es team. *The Astrophysical journal letters*, 934(1):L7, 2022.

[12] Guillaume Payeur, Evan McDonough, and Robert Brandenberger. Do observations prefer thawing quintessence?, 2025. URL https://arxiv.org/abs/2411.13637.

[13] Tanvi Karwal and Marc Kamionkowski. Dark energy at early times, the Hubble parameter, and the string axiverse. *Phys. Rev. D*, 94(10):103523, 2016. doi: 10.1103/PhysRevD.94.103523.

[14] Vivian Poulin, Tristan L. Smith, Tanvi Karwal, and Marc Kamionkowski. Early Dark Energy Can Resolve The Hubble Tension. *Phys. Rev. Lett.*, 122(22):221301, 2019. doi: 10.1103/PhysRevLett.122.221301.

[15] Evan McDonough, J. Colin Hill, Mikhail M. Ivanov, Adrien La Posta, and Michael W. Toomey. Observational constraints on early dark energy. *International Journal of Modern Physics D*, 33 (11):2430003, 2024. doi: 10.1142/S0218271824300039. URL https://doi.org/10.1142/S0218271824300039.

[16] Marc Kamionkowski and Adam G. Riess. The Hubble Tension and Early Dark Energy. *Ann. Rev. Nucl. Part. Sci.*, 73:153–180, 2023. doi: 10.1146/annurev-nucl-111422-024107.

[17] Vivian Poulin, Tristan L. Smith, and Tanvi Karwal. The Ups and Downs of Early Dark Energy solutions to the Hubble tension: A review of models, hints and constraints circa 2023. *Phys. Dark Univ.*, 42:101348, 2023. doi: 10.1016/j.dark.2023.101348.

[18] Jia-Qi Wang, Rong-Gen Cai, Zong-Kuan Guo, and Shao-Jiang Wang. Resolving the planck-desi tension by non-minimally coupled quintessence, 2025. URL https://arxiv.org/abs/2508.01759.

[19] Yashar Akrami, George Alestas, and Savvas Nesseris. Has desi detected exponential quintessence?, 2025. URL https://arxiv.org/abs/2504.04226.

[20] Tian-Nuo Li, Yun-He Li, Guo-Hong Du, Peng-Ju Wu, Lu Feng, Jing-Fei Zhang, and Xin Zhang. Revisiting holographic dark energy after desi 2024, 2025. URL https://arxiv.org/abs/2411.08639.

[21] Yuichiro Tada and Takahiro Terada. Quintessential interpretation of the evolving dark energy in light of desi observations. *Physical Review D*, 109(12), June 2024. ISSN 2470-0029. doi: 10.1103/physrevd.109.l121305. URL http://dx.doi.org/10.1103/PhysRevD.109.L121305.

[22] Bikash R Dinda and Roy Maartens. Physical versus phantom dark energy after desi: thawing quintessence in a curved background. *Monthly Notices of the Royal Astronomical Society: Letters*, 542(1):L31–L35, June 2025. ISSN 1745-3933. doi: 10.1093/mnrasl/slaf063. URL http://dx.doi.org/10.1093/mnrasl/slaf063.

[23] William Giarè, Miguel A. Sabogal, Rafael C. Nunes, and Eleonora Di Valentino. Interacting dark energy after desi baryon acoustic oscillation measurements. *Physical Review Letters*, 133(25), December 2024. ISSN 1079-7114. doi: 10.1103/physrevlett.133.251003. URL http://dx.doi.org/10.1103/PhysRevLett.133.251003.

[24] Ioannis D Gialamas, Gert Hütsi, Martti Raidal, Juan Urrutia, Martin Vasar, and Hardi Veermäe. Quintessence and phantoms in light of desi 2025. *arXiv preprint arXiv:2506.21542*, 2025.

[25] L. A. Ureña-López, F. Lozano-Rodríguez, J. O. Román-Herrera, J. Aguilar, S. Ahlen, D. Bianchi, D. Brooks, T. Claybaugh, A. de la Macorra, Arjun Dey, S. Ferraro, J. E. Forero-Romero, E. Gaztañaga, S. Gontcho A Gontcho, G. Gutierrez, K. Honscheid, C. Howlett, M. Ishak, R. Kehoe, D. Kirkby, T. Kisner, A. Lambert, M. Landriau, L. Le Guillou, M. Manera, A. Meisner, R. Miquel, J. Moustakas, F. Prada, I. Pérez-Ràfols, G. Rossi, E. Sanchez, M. Schubnell, J. Silber, D. Sprayberry, G. Tarlé, B. A. Weaver, and H. Zou. Updated cosmological constraints on axion dark energy with desi, 2025. URL https://arxiv.org/abs/2503.20178.

[26] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

[27] Daniel J. H. Chung, Zhiqi Gao, Yurii Kvasiuk, Tianyi Li, Moritz Münchmeyer, Maja Rudolph, Frederic Sala, and Sai Chaitanya Tadepalli. Theoretical physics benchmark (tpbench) – a dataset and study of ai reasoning capabilities in theoretical physics, 2025. URL https://arxiv.org/abs/2502.15815.

[28] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

[29] Google LLC. Gemini 2.5 pro model card. https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro.pdf.

[30] Nils Schöneberg, Guillermo Franco Abellán, Andrea Pérez Sánchez, Samuel J Witte, Vivian Poulin, and Julien Lesgourgues. The h0 olympics: A fair ranking of proposed models. *Physics Reports*, 984:1–55, 2022.

Appendix

A Environment Details

The agent performs its tasks in a Podman (rootless) emulator for Docker. The Dockerfile that builds the image pre-installs CLASS (version 3.3.0), as well as the code needed to run a Claude Code session. The script that launches the experiment by creating a new container mounts certain host system folders to the containerized environment. This ensures that the agent has access to the necessary auxiliary code and that its submission folder remains on the host system once its experiment ends. This submission folder contains the patch and is used for evaluation.

A.1 Auxiliary Modules and Files

The files that are available in the directory, in addition to the base /class code files include prompt files, high-level summaries of key CLASS modules and auxiliary code modules relevant to Stages 2 and 3.

Prompts The instructions specific to each stage are separated into three prompt files: `stage1_prompt.md` `stage2_prompt.md`, `stage3_prompt.md`.

Filemaps The CLASS code base is considerably long: the `perturbations.c` module ,for example, has over 10000 lines of code. To facilitate navigation across such a codebase, we presave LLM-generated summaries of the the background.c, perturbations.c and input.c modules. We do so by giving the LLM, `Gemini-2.5-pro` [29] a prompt along with the saved module with corresponding line numbers.

A.2 Agent Design

Web Access The command that launches the agentic experiment disallows the 'WebSearch' and 'WebFetch' tools so the agent cannot use its intrinsic web access tools. The Dockerfile that constructs the environment further contains code that overwrites common web-access relevant commands such as ping and curl to ensure that they pass through a network filter that only allows "permitted" domains, so as to eliminate the LLM from circumventing the blocked tools by executing bash commands that query the Internet.

Claude Code We chose Claude Code as the agentic framework since modifications required to implement an arbitrary cosmological model necessitate changes to several parts of the code, to fully incorporate the extension into the code base. Claude Code is a terminal-based agentic framework

that has the ability to read files, navigate large code bases and execute code. The npm Claude Code release version we used was 1.0.31. Note, regardless of the version, all experiments we report here used `claude-sonnet-4` as the main model, although the agentic framework relegates easier tasks to a smaller model. By leveraging the 'allowedTools' flag while launching a new experiment, we allow most necessary tools, such as bash commands, python commands among others to run without requiring permission from the user. In spite of doing so, the agent will still prompt the human user for permission to run certain risky or new commands (such as a `make` command). This shows up as a multiple-choice selection on the user's interface. Across experiments we followed the same level of human interaction: we grant it permission for all commands, and do not interact with the agent after the initial experiment-launching prompt.

B Cosmological Models

B.1 Problem Specification

We provide $\mathcal{P}_{Fiducial}$ as a JSON dictionary that can be used with the Python wrapper to C, Classy. In addition to the JSON, we include an optional 'ini' version of $\mathcal{P}_{Fiducial}$ that may be used to compute observables using the modified CLASS executable directly, in case errors were encountered with the Classy wrapper. In addition to describing the theory, and added parameters, the standard description.md also includes implementation details to align the agents assumptions on the codebase and the input parameter with that of the target implementation. This is done so that we can consistently use the same $\mathcal{P}_{Fiducial}$ value with both the LLM's implementation, as well as the target repository (when we pre-save observables for comparison in Stage 2).

We additionally include a `problem_context.yaml` file with the following structure and fields.

- stage2:
  - type: This entry confers our framework with the ability to accommodate problems with and without a target / ground truth implementation. `Options: artifact-based | none`
  - `presave_in`: If 'type' is artifact-based, this entry determines whether the target implementation's observables were presaved using the C executable or Classy. `Options: Classy | C_exec`
  - `presaved_observables`: If 'type' is artifact-based, this entry determines which observables should be presaved using the target repository. `Eg: ['mPk', 'Omega']`
  - `save_observables`: Which observables to save during Stage 2 of the agentic experiment. `Eg: ['cls', 'mPk', 'Omega', 'H0', 'sigma8', 'S8']`
  - `model_specific_tests`:
    - `comparison_qty`: Which observables to compare between the target implementation and the agent's implementation.
    - `field`: Which dark energy density parameter is relevant to the problem. `Eg:` $\Omega_{scf}, \Omega_{fld}$
- stage3:
  - `type`: Whether to find parameters that minimize the negative log-likelihood (`minimization`) or ask the agent to manually propose and explore the model's behavior in different limits on its own (`exploration`).
  - `vary_params`: Single-line instruction explaining which parameters need to be minimized / explored.
- extra_context_files: This allows us the flexibility to include additional problem-specific contextual prompts for an experiment. `Eg: ["propose_model_prompt.md"] for` *Hypothesis Generation One-Shot*.

B.2 Model Descriptions

B.2.1 Thawing Quintessence

The agent is instructed to minimize Ω_{cdm}, h and the added parameters. The density parameter of interest here is Ω_{fld}. The `description.md` we used is below. The body consists of the three sections: the description of the theory, added parameters and implementation details.

The Thawing Quintessence (Tanh Parametrization) Model
This is a thawing quintessence model characterized by a specific parametrization of the dark energy equation of state, $w(z)$. Thawing quintessence models propose that the dark energy is a scalar field that was initially frozen (mimicking a cosmological constant with $w \approx -1$) due to Hubble friction at high redshifts. As the universe expands and Hubble friction decreases, ϕ eventually unfreezes. At that point, w departs from -1, and the ensuing evolution of ϕ and w depends on the scalar potential V (ϕ). the field begins to "thaw" and evolve, causing w to deviate from -1.
The core of this model is a phenomenological parametrization for the equation of state $w(z)$ designed to capture the qualitative behavior of thawing quintessence in a model-independent way:

$$w(z) = \frac{\Delta w}{2}\left(1 - \tanh\left(\frac{z - z_c}{\Delta z}\right)\right) - 1$$

where:

- z is the cosmological redshift.
- Δw controls the magnitude of the transition in w. The equation of state transitions from $w \approx -1$ at high z to $w \approx -1 + \Delta w$ at low z (specifically, for $z \ll z_c$). Consider Δw in the range $[0, 2]$, ensuring w stays between -1 and 1.
- z_c is the redshift at which the transition is centered. The prior on z_c is [0, 0.25] and forces the transition to occur in the near past.
- Δz determines the width (in redshift) of the transition. $\log_{10} \Delta z$ is in $[-1.5, 0.5]$.

This parametrization ensures $w(z) \to -1$ at high redshifts ($z \gg z_c$). The model reduces to a cosmological constant (ΛCDM) if $\Delta w = 0$. It can also appear indistinguishable from ΛCDM if the transition occurs very far in the past ($z_c \gg$ relevant redshifts) or entirely in the future ($z_c \ll 0$ with small Δz).
Parameters / .ini Structure for Added Parameters: This model introduces **3** new parameters beyond the standard ΛCDM parameters. For implementation in CLASS, these parameters should be added to the input `.ini` file. Names and descriptions are:

Parameter in .ini	Description	Typical Range (Priors)
`delta_w`	The amplitude of the transition in $w(z)$.	$0 \dots 2$
`log10_delta_z`	Base-10 logarithm of the width of the transition, $\log_{10}(\Delta z)$.	Fixed to -1.5
`zc`	The redshift z_c at the center of the transition.	$-2 \dots 2$

Additional background field variables: When running CLASS with this model, the following variables related to the thawing quintessence component should be accessible in the `background.dat` output file:

- `w`: The equation of state $w(z) = p_{tq}/\rho_{tq}$ of the thawing quintessence fluid.

Implementation Details

- In `background_initial_conditions` you should compute numerically the simple 1d integral $[int_{a_{ini}}^{a_0} 3[(1 + w_{fld})/a]da]$ (e.g. with the Romberg method) instead of calling `background_w_fld` to get `integral_fld` (this is left to 0 in calculations done is `background_w_fld`).
- Use the already implemented minimal Parametrized Post-Friedmann (PPF) approach via the `perturbations.c` module.

B.2.2 Early Dark Energy

The parameter of interest here is Ω_{scf}, the fractional density of the EDE field in the Universe. The `description.md` is provided below. Note that the EDE description contains a lot of additional information required to compare to the target repository, such as units and intended usage of input parameters. Additionally, we found the model struggled with unit conversions and found it necessary to provide guidelines on unit conversions. In future work we will test different ablations of this model description.

The EDE Model
Early dark energy refers to a model in which a scalar field is frozen-in at times prior to recombination, thus behaving during this epoch like a dark energy component. The idea that an anomalous era of expansion arising from EDE at such times might resolve the Hubble tension was first suggested in Karwal:2016vyq, where computation only at the level of the background was shown to partially alleviate the tension; it was later shown in Poulin:2018cxd,Lin:2019qug,Smith:2019ihp that Planck data is also strongly sensitive to the dynamics of the perturbations, favoring either a non-canonical kinetic term, whereby the equation of state w is approximately equal to the effective sound speed c_s^2 Lin:2019qug, or a potential that flattens close to the initial field value Smith:2019ihp.
Early dark energy can be realized with a scalar field of the following potential,

$$V(\phi) = (mf)^2 \left(1 - \cos\left(\frac{\phi}{f}\right)\right)^n + V_\Lambda$$

, where m is the axion mass, f the axion decay constant, ϕ the scalar field, and V_Λ is the cosmological constant. The evolution of the scalar field is governed by the Klein-Gordon equation, here in conformal time,

$$\phi'' + 2aH\phi' + a^2 \frac{dV}{d\phi} = 0$$

You must implement adiabatic initial conditions as follows. If `s2_squared` $= 1 - 3K/k^2$ and `ktau_three` $= (k\tau)^3$, $R_{\rm ini}$ is the comoving curvature perturbation and $\Omega_{\rm rad} = 4.15 \times 10^{-5} h^{-2}$:

$$\delta\phi(0) = \frac{dV_{\rm scf} \cdot H_0^2 \cdot (k\tau_3)^2 \cdot R_{\rm ini} \cdot s_2^2 \cdot \Omega_{\rm rad}}{210 \cdot k^4}$$

$$\delta\phi'(0) = \frac{a^2 \cdot dV_{\rm scf} \cdot k\tau_3 \cdot R_{\rm ini} \cdot s_2^2}{35 \cdot k} + \frac{dV_{\rm scf}^3 \cdot H_0^4 \cdot (k\tau_3)^3 \cdot R_{\rm ini} \cdot s_2^2 \cdot \Omega_{\rm rad}^2}{10500 \cdot k^7 \cdot V_{\rm scf}}$$

Added Parameters:
You should modify the CLASS codebase such that it works with the provided parameter file, which is described below. This model implements a 3-parameter extension of ΛCDM. The additional parameters provided in the parameter file are:

Parameter	Description
`m_scf`	Axion Mass (in eV)
`f_scf`	Axion decay constant (in eV)
`thetai_scf`	The initial axion misalignment angle θ_i

You will fix $n = 3$ and explore other parameters (although n may still be an input in the param file).
Additional background field variables:
These variables should be accessible in the background.dat file that is produced upon running the code:

- `V_e_scf` (the scalar field potential - i.e. without Λ)
- `phi_scf`

Implementation Details

- The following unit conversions need to be performed exactly as below. Note `_input` indicates the parameter value in terms of the units specified in the parameter file:

1. Construct the potential term using `m_axion_input` [eV] and `f_axion_input` [eV]
2. In the cosine term, (ϕ/f_{axion}) [dimensionless] = $\phi \cdot 2.435 \times 10^{27}/(f_{\text{axion,input}}$ [eV])
3. $(m_{\text{axion,input}} \cdot f_{\text{axion,input}})^2$ will thus have units of V [eV^4]. Multiply V [eV^4] by 4152.39. Thus, $3.1.V_{e,\text{scf}}$ [$\text{m}_{\text{pl}}^2/Mpc^2$] $= 4152.39 \cdot (m_{\text{axion,input}} \cdot f_{\text{axion,input}})^2 \cdot (1 - \cos(\phi \cdot 2.435 \times 10^{27}/f_{\text{axion,input [eV]}}))^n$
4. $V_{\text{scf}} = V_{e,\text{scf}}[m_{\text{pl}}^2/Mpc^2] + \text{CC}_{\text{scf,input}} \cdot 3.968 \times 10^{-8}$

- The user should take note that the implementation of dark energy, i.e. Λ, is done differently when using the EDE model. The cosmological constant has been included in the scalar field potential, thus the CLASS parameter Ω_Λ should be set to zero. This is done so that the shooting algorithm adjusts Λ to enforce the budget equation without touching the parameters of the EDE model. Thus, for the intended EDE behavior, in the parameter file:
  1. `Omega_Lambda` and `Omega_fld` must be 0. Do not specify a non-zero value for Ω_Λ as this will implement ΛCDM.
  2. `Omega_scf` must be set to a negative value (-1).
- The various parameters for the scalar field potential are provided via the `scf_parameters` entry in the parameter file. Specifically, the values for different indices of `scf_parameters` denote the corresponding parameters and should be accessed from the `scf_parameters` entry:
  - $0 \rightarrow$ `f_scf`
  - $1 \rightarrow$ `m_scf`
  - $2 \rightarrow$ `theta_i_scf`
  - $3 \rightarrow$ `CC_scf`
  - $4 \rightarrow$ `n_scf` (FIXED)
  - $5 \rightarrow$ Unused
- **Numerical Precision:** Increase `ntrial` to 50 in the Newton-Raphson step in the tools directory.
- Remember: $\phi_{\text{scf}} = \theta_{\text{scf}} \cdot f_{\text{axion,input}}$ [eV] / (2.435×10^{27})
- The adiabatic initial conditions can be found in the `perturbations.c` module, and `s2_squared` and `ktau_three` are already pre-defined.
- You need to appropriately partition the scalar field into the matter-like and radiation-like components:

$$p_{\text{scf}} = \frac{\phi'^2/(2a^2) - V_{e,\text{scf}}(\phi)}{3}$$

$$\rho_r = 3p_{\text{scf}}$$

$$\rho_m = \frac{\phi'^2/(2a^2) + V_{e,\text{scf}}(\phi)}{3} - 3p_{\text{scf}}$$

B.2.3 Hypothesis Generation

In this setting since the LLM has to create its own ($\mathcal{M}$, $\mathcal{P}_{Fiducial}$), there is an additional auxiliary prompt, `propose_model.md`. It thus modifies the empty description and $\mathcal{P}_{Fiducial}$ files before proceeding to the remaining stages.

C Additional pipeline information

An experiment is launched with the following command for the experiments with a provided input cosmological model:

```
claude "Complete the tasks by following the instructions in the three
    prompts stage1_prompt.md, stage2_prompt.md and stage3_prompt.md.
    Start by opening stage1_prompt.md, completing the tasks described
    for Stage 1 and then proceed to complete the tasks for the Stage
    2 and Stage 3 by reading the tasks in those prompts. Your final
    outputs will be evaluated by the eval_script.sh. You may NOT
    access the Internet at any point. Please continue until you have
    successfully completed all stages." --model $MODEL --verbose
    --disallowedTools="WebSearch,WebFetch"
```

For the *Hypothesis Generation* experiment, the following command launches the agentic experiment:

```
claude "You must first devise your own novel beyond-Lambda CDM
    cosmological model. Read the instructions in
    propose_model_prompt.md. Once you have described your
    cosmological model in description.md and param_base.json,
    complete the tasks by following the instructions in the three
    prompts stage1_prompt.md, stage2_prompt.md and stage3_prompt.md.
    Start by opening stage1_prompt.md, completing the tasks described
    for Stage 1 and then proceed to complete the tasks for the Stage
    2 and Stage 3 by reading the tasks in those prompts. Your final
    outputs will be evaluated by the eval_script.sh. You may NOT
    access the Internet at any point. Please continue until you have
    successfully completed all stages." --model $MODEL --verbose
    --disallowedTools="WebSearch,WebFetch"
```

C.1 Stage 2

In Stage 2, the LLM must compute observables using $\mathcal{P}_{Fiducial}$. We design the script that the LLM must use to compute observables so that it prioritizes computing the observables via the Python wrapper, Classy and subsequently falls back to the C executable if the former fails. The design choice of the using a predefined script that the LLM must use to compute observables enforces uniformity in how observables are computed and stored, which enables a straightforward comparison to the target observables.

C.2 Stage 3

In the *Minimization* variant, the agent must create a configuration file that specifies which parameters must be fixed and which should be kept free, and must additionally specify reasonable bounds for each parameter that is to be optimized. In our experiments, we instructed the agent (via the problem context file) to vary the added parameters, in addition to $\{\Omega_{cdm}, h\}$: the density parameter for cold dark matter and the expansion rate of the universe divided by 100. To optimize a different set of parameters, one would only need to modify the single line instruction relevant to this step in the `problem context.yaml`.

Since our setup has 2 phases – the agentic experiment followed by the evaluation of the resulting artifacts in the submission folder – the code and patch can potentially be evaluated on different rewards during evaluation than when the agent was computing rewards in Stage 3. In practice, in our experiments thus far, we do not expect this to affect the behavior of the agentic experiment since the reward computation stage is last and in none of the cases does the agent attempt to rectify its original model or code *after* computing rewards. We generally used the same code to compute rewards as that used during evaluation, however, there were a couple of differences: for the *TQ: Paper to Code* and the Novel One-Shot experiment, just the *Planck*+DESI likelihood was computed during the course of the agentic experiment, and the precision constants for the tensions were added during evaluation. Note, the numbers in all tables reported here correspond to the rewards computed during evaluation and were computed using the same rewards for all experiments.

C.2.1 Likelihood definition

We maximize a combined likelihood of the baryonic acoustic oscillations (BAO) data obtained from the DESI collaboration [1], and the supernovae data from the Pantheon+[10] dataset with $z > 0.01$. Additionally, we incorporate an early Universe 3 parameter prior described in Appendix A of [1], a compression of the full Cosmic Microwave Background (CMB) information into the quantities $\theta_\star$, $w_{\rm bc}$, and $w_{\rm b}$. This early Universe quantities are independent about the late time evolution assumptions and can therefore be safely used with models that only modified the late time expansion of the Universe. The parameters we fit are Ω_{cdm}, h, M_B and any extended model parameters from the beyond ΛCDM extensions.

C.2.2 Assessing the Hubble tension

We include two separate metrics for assessing the Hubble tension, one based on the H_0 measurements themselves, and one based on M_B, following [30]. In both cases we use the Gaussian tension definition, $\frac{x - x_{\rm SH0ES}}{\sigma^2 + \sigma^2_{\rm SH0ES}}$, where x is inferred in a model dependent way via maximising our likelihood, and σ is a constant precision that we assume to be 0.67 ($\sim 1\%$). Note that since we are not sampling full posteriors we cannot estimate σ directly.

Beyond quantifying the tension with respect to SH0ES in terms of the H_0 inference, we also check the supernovae intrinsic SNIa magnitude, M_b, tension to show whether the inferred values are consistent between Pantheon+ and SH0ES. This is important since SH0ES data does not constrain the H_0 parameter directly but calibrates the intrinsic magnitude of supernovae at higher redshifts needed to infer the current expansion rate. A model that resolves the H_0 tension could still predict supernovae magnitudes that do not agree with observed values.

D Experiment Results

D.1 Experiments cost

For the experiments we discuss here, the cost of a single experiment with Claude Sonnet-4 is around \$5: \$4.34 for EDE, \$4.53 for Thawing Quintessence, and \$5.32 and \$7.10 for the Hypothesis Generation experiments.

D.2 Code Implementation Patches

We provide the patch that the LLM generated for the *Thawing Quintessence* experiment below:

```
diff --git a/include/background.h b/include/background.h
index fe4d42d8..cf39df19 100755
--- a/include/background.h
+++ b/include/background.h
@@ -12,7 +12,7 @@

 /** list of possible parametrisations of the DE equation of
     state */

-enum equation_of_state {CLP,EDE};
+enum equation_of_state {CLP,EDE,TANH};


 /** list of possible parametrizations of the varying
     fundamental constants */
@@ -112,6 +112,9 @@ struct background
    double cs2_fld;  /**< \f$ c^2_{s~DE} \f$: sound speed of the
       fluid in the frame comoving with the fluid (so, this is
                        not [delta p/delta rho] in the
                            synchronous or newtonian gauge!) */
```

```
    double Omega_EDE;        /**< \f$ wa_{DE} \f$: Early Dark
       Energy density parameter */
+   double dw_tanh;          /**< \f$ \Delta w \f$: thawing
    quintessence transition amplitude */
+   double dz_tanh;          /**< \f$ \Delta z \f$: thawing
    quintessence transition width */
+   double z_tanh;           /**< \f$ z_c \f$: thawing
    quintessence transition center redshift */
    double * scf_parameters; /**< list of parameters describing
       the scalar field potential */
    short attractor_ic_scf;  /**< whether the scalar field has
       attractor initial conditions */
    int scf_tuning_index;    /**< index in scf_parameters used
       for tuning */
diff --git a/source/background.c b/source/background.c
index f24fc814..cbd51419 100755
--- a/source/background.c
+++ b/source/background.c
@@ -702,6 +702,15 @@ int background_w_fld(
      // w_ede(a) taken from eq. (11) in 1706.00730
      *w_fld = -
          dOmega_ede_over_da*a/Omega_ede/3./(1.-Omega_ede)+a_eq/3./(a+a_eq);
      break;
+   case TANH:
+     // Thawing quintessence tanh parametrization: w(z) =
    (dw/2)*(1-tanh((z-zc)/dz))-1
+     // Convert scale factor to redshift: z = 1/a - 1
+     {
+       double z = 1./a - 1.;
+       double tanh_arg = (z - pba->z_tanh) / pba->dz_tanh;
+       *w_fld = pba->dw_tanh/2. * (1. - tanh(tanh_arg)) - 1.;
+     }
+     break;
    }


@@ -721,6 +730,18 @@ int background_w_fld(
        +
            dOmega_ede_over_da*dOmega_ede_over_da*a/3./(1.-Omega_ede)/(1.-Omega_ede)/Ome
      + a_eq/3./(a+a_eq)/(a+a_eq);
      break;
+   case TANH:
+     // Derivative of w with respect to a for tanh
    parametrization
+     // dw/da = dw/dz * dz/da, where dz/da = -1/a^2
+     {
+       double z = 1./a - 1.;
+       double tanh_arg = (z - pba->z_tanh) / pba->dz_tanh;
+       double sech2_arg = 1. - tanh(tanh_arg) * tanh(tanh_arg);
    // sech^2(x) = 1 - tanh^2(x)
+       double dw_over_dz = -pba->dw_tanh/2. * sech2_arg /
    pba->dz_tanh;
+       double dz_over_da = -1./(a*a);
+       *dw_over_da_fld = dw_over_dz * dz_over_da;
+     }
+     break;
    }

    /** - finally, give the analytic solution of the following
       integral:
@@ -740,6 +761,10 @@ int background_w_fld(
    case EDE:
```

```
      class_stop(pba->error_message,"EDE implementation not
          finished: to finish it, read the comments in
          background.c just before this line\n");
      break;
+   case TANH:
+     // Numerical integration will be implemented in
    background_initial_conditions
+     *integral_fld = 0.;
+     break;
    }

    /** note: of course you can generalise these formulas to
        anything,
@@ -2251,6 +2276,58 @@ int background_initial_conditions(
          [(1+w_fld)/a] da] (e.g. with the Romberg method?)
              instead of
          calling background_w_fld */

+     /* Numerical integration for TANH equation of state */
+     if (pba->fluid_equation_of_state == TANH) {
+       int n_max = 20; // maximum number of Romberg iterations
+       double tolerance = 1.e-10;
+       double h, sum_trapz, sum_romberg;
+       double a_step, w_temp, dw_temp, integral_temp;
+       int n, i, k;
+       double romberg_table[21][21]; // Romberg table for
    n_max=20
+
+       // Romberg integration of int_{a_ini}^{1} 3(1+w_fld)/a da
+       h = 1. - a; // initial step size (from a_ini=a to a_0=1)
+
+       // First trapezoid rule approximation
+       class_call(background_w_fld(pba, a, &w_temp, &dw_temp,
    &integral_temp), pba->error_message, pba->error_message);
+       double f_start = 3.*(1. + w_temp)/a;
+       class_call(background_w_fld(pba, 1., &w_temp, &dw_temp,
    &integral_temp), pba->error_message, pba->error_message);
+       double f_end = 3.*(1. + w_temp)/1.;
+
+       romberg_table[0][0] = h * (f_start + f_end) / 2.;
+
+       // Romberg iterations
+       for (n = 1; n <= n_max; n++) {
+         h = h / 2.;
+         sum_trapz = 0.;
+
+         // Add intermediate points for trapezoid rule
+         for (i = 1; i <= (1 << (n-1)); i++) {
+           a_step = a + (2*i - 1) * h;
+           class_call(background_w_fld(pba, a_step, &w_temp,
    &dw_temp, &integral_temp), pba->error_message,
    pba->error_message);
+           sum_trapz += 3.*(1. + w_temp)/a_step;
+         }
+
+         romberg_table[n][0] = romberg_table[n-1][0] / 2. + h *
    sum_trapz;
+
+         // Richardson extrapolation
+         for (k = 1; k <= n; k++) {
+           double factor = pow(4., k);
```

```
+          romberg_table[n][k] = (factor *
    romberg_table[n][k-1] - romberg_table[n-1][k-1]) / (factor -
    1.);
+        }
+
+        // Check convergence
+        if (n >= 2 && fabs(romberg_table[n][n] -
    romberg_table[n-1][n-1]) < tolerance *
    fabs(romberg_table[n][n])) {
+          integral_fld = romberg_table[n][n];
+          break;
+        }
+
+        if (n == n_max) {
+          integral_fld = romberg_table[n][n];
+        }
+      }
+    }
+
     /* rho_fld at initial time */
     pvecback_integration[pba->index_bi_rho_fld] =
         rho_fld_today * exp(integral_fld);

diff --git a/source/input.c b/source/input.c
index 933e586c..da1cf0e7 100755
--- a/source/input.c
+++ b/source/input.c
@@ -3279,6 +3279,9 @@ int input_read_parameters_species(struct
    file_content * pfc,
       else if ((strstr(string1,"EDE") != NULL) ||
           (strstr(string1,"ede") != NULL)) {
         pba->fluid_equation_of_state = EDE;
       }
+      else if ((strstr(string1,"TANH") != NULL) ||
    (strstr(string1,"tanh") != NULL)) {
+        pba->fluid_equation_of_state = TANH;
+      }
       else {
         class_stop(errmsg,"incomprehensible input '%s' for the
             field 'fluid_equation_of_state'",string1);
       }
@@ -3298,6 +3301,14 @@ int input_read_parameters_species(struct
    file_content * pfc,
       class_read_double("Omega_EDE",pba->Omega_EDE);
       class_read_double("cs2_fld",pba->cs2_fld);
     }
+    if (pba->fluid_equation_of_state == TANH) {
+      /** 8.a.2.4) Equation of state of the fluid in 'TANH'
    case */
+      /* Read */
+      class_read_double("dw_tanh",pba->dw_tanh);
+      class_read_double("dz_tanh",pba->dz_tanh);
+      class_read_double("z_tanh",pba->z_tanh);
+      class_read_double("cs2_fld",pba->cs2_fld);
+    }
   }

   /** 8.b) If Omega scalar field (SCF) is different from 0 */
@@ -5903,6 +5914,10 @@ int input_default_params(struct
    background *pba,
   pba->wa_fld = 0.;
   /** 9.a.2.2) 'EDE' case */
   pba->Omega_EDE = 0.;
```

```
+  /** 9.a.2.3) 'TANH' case */
+  pba->dw_tanh = 0.;
+  pba->dz_tanh = 1.;
+  pba->z_tanh = 1.;
   /** 9.b) Omega scalar field */
   /** 9.b.1) Potential parameters and initial conditions */
   pba->scf_parameters = NULL;
```

And the code patch generated by Claude Code for the EDE implementation,

```
diff --git a/include/background.h b/include/background.h
index fe4d42d8..81aa3a4f 100755
--- a/include/background.h
+++ b/include/background.h
@@ -176,6 +176,7 @@ struct background
   int index_bg_phi_scf;       /**< scalar field value */
   int index_bg_phi_prime_scf; /**< scalar field derivative wrt
       conformal time */
   int index_bg_V_scf;         /**< scalar field potential V */
+  int index_bg_V_e_scf;       /**< scalar field potential V_e
    (without cosmological constant) */
   int index_bg_dV_scf;        /**< scalar field potential
       derivative V' */
   int index_bg_ddV_scf;       /**< scalar field potential
       second derivative V'' */
   int index_bg_rho_scf;       /**< scalar field energy density
       */
@@ -552,6 +553,21 @@ extern "C" {
                double phi
                );

+  double V_e_scf(
+                 struct background *pba,
+                 double phi
+                 );
+
+  double dV_e_scf(
+                  struct background *pba,
+                  double phi
+                  );
+
+  double ddV_e_scf(
+                   struct background *pba,
+                   double phi
+                   );
+
   double dV_scf(
                 struct background *pba,
                 double phi
diff --git a/source/background.c b/source/background.c
index f24fc814..4686321a 100755
--- a/source/background.c
+++ b/source/background.c
@@ -476,6 +476,7 @@ int background_functions(
     pvecback[pba->index_bg_phi_scf] = phi; // value of the
         scalar field phi
     pvecback[pba->index_bg_phi_prime_scf] = phi_prime; //
        value of the scalar field phi derivative wrt conformal
        time
```

```
      pvecback[pba->index_bg_V_scf] = V_scf(pba,phi);
          //V_scf(pba,phi); //write here potential as function of
          phi
+     pvecback[pba->index_bg_V_e_scf] = V_e_scf(pba,phi); //
    V_e_scf(pba,phi); //EDE potential without cosmological
    constant
      pvecback[pba->index_bg_dV_scf] = dV_scf(pba,phi); //
          dV_scf(pba,phi); //potential' as function of phi
      pvecback[pba->index_bg_ddV_scf] = ddV_scf(pba,phi); //
          ddV_scf(pba,phi); //potential'' as function of phi
      pvecback[pba->index_bg_rho_scf] =
          (phi_prime*phi_prime/(2*a*a) + V_scf(pba,phi))/3.; //
          energy of the scalar field. The field units are set
          automatically by setting the initial conditions
@@ -483,9 +484,18 @@ int background_functions(
      rho_tot += pvecback[pba->index_bg_rho_scf];
      p_tot += pvecback[pba->index_bg_p_scf];
      dp_dloga += 0.0; /** <-- This depends on a_prime_over_a,
          so we cannot add it now! */
-     //divide relativistic & nonrelativistic (not very
    meaningful for oscillatory models)
-     rho_r += 3.*pvecback[pba->index_bg_p_scf]; //field
    pressure contributes radiation
-     rho_m += pvecback[pba->index_bg_rho_scf] - 3.*
    pvecback[pba->index_bg_p_scf]; //the rest contributes matter
+
+     /* EDE model partitioning according to description.md */
+     /* p_scf = (phi'^2/(2*a^2) - V_e_scf(phi))/3 */
+     double p_scf_ede = (phi_prime*phi_prime/(2*a*a) -
    V_e_scf(pba,phi))/3.;
+     /* rho_r = 3*p_scf */
+     double rho_r_ede = 3.0 * p_scf_ede;
+     /* rho_m = (phi'^2/(2*a^2) + V_e_scf(phi))/3 - 3*p_scf */
+     double rho_m_ede = (phi_prime*phi_prime/(2*a*a) +
    V_e_scf(pba,phi))/3.0 - 3.0 * p_scf_ede;
+
+     //divide relativistic & nonrelativistic according to EDE
    model
+     rho_r += rho_r_ede; //field EDE radiation-like component
+     rho_m += rho_m_ede; //field EDE matter-like component
      //printf(" a= %e, Omega_scf = %f, \n ",a,
          pvecback[pba->index_bg_rho_scf]/rho_tot );
    }

@@ -1067,6 +1077,7 @@ int background_indices(
    class_define_index(pba->index_bg_phi_scf,pba->has_scf,index_bg,1);
    class_define_index(pba->index_bg_phi_prime_scf,pba->has_scf,index_bg,1);
    class_define_index(pba->index_bg_V_scf,pba->has_scf,index_bg,1);
+
    class_define_index(pba->index_bg_V_e_scf,pba->has_scf,index_bg,1);
    class_define_index(pba->index_bg_dV_scf,pba->has_scf,index_bg,1);
    class_define_index(pba->index_bg_ddV_scf,pba->has_scf,index_bg,1);
    class_define_index(pba->index_bg_rho_scf,pba->has_scf,index_bg,1);
@@ -2464,6 +2475,7 @@ int background_output_titles(
    class_store_columntitle(titles,"phi_scf",pba->has_scf);
    class_store_columntitle(titles,"phi'_scf",pba->has_scf);
    class_store_columntitle(titles,"V_scf",pba->has_scf);
+   class_store_columntitle(titles,"V_e_scf",pba->has_scf);
    class_store_columntitle(titles,"V'_scf",pba->has_scf);
    class_store_columntitle(titles,"V''_scf",pba->has_scf);

@@ -2537,6 +2549,7 @@ int background_output_data(
      class_store_double(dataptr,pvecback[pba->index_bg_phi_scf],pba->has_scf,storeidx);
```

```
     class_store_double(dataptr,pvecback[pba->index_bg_phi_prime_scf],pba->has_scf,stor
     class_store_double(dataptr,pvecback[pba->index_bg_V_scf],pba->has_scf,storeidx);
+
    class_store_double(dataptr,pvecback[pba->index_bg_V_e_scf],pba->has_scf,storeidx);
     class_store_double(dataptr,pvecback[pba->index_bg_dV_scf],pba->has_scf,storeidx);
     class_store_double(dataptr,pvecback[pba->index_bg_ddV_scf],pba->has_scf,storeidx);

@@ -2904,34 +2917,74 @@ int background_output_budget(
 double V_e_scf(struct background *pba,
                double phi
                ) {
-  double scf_lambda = pba->scf_parameters[0];
-  //  double scf_alpha  = pba->scf_parameters[1];
-  //  double scf_A      = pba->scf_parameters[2];
-  //  double scf_B      = pba->scf_parameters[3];
-
-  return  exp(-scf_lambda*phi);
+  double f_scf = pba->scf_parameters[0];  /* f_scf in eV */
+  double m_scf = pba->scf_parameters[1];  /* m_scf in eV */
+  double n_scf = pba->scf_parameters[4];  /* n_scf (fixed to
    3) */
+
+  /* Unit conversions as specified in description.md */
+  /* phi*2.435e27/f_scf gives dimensionless argument for
    cosine */
+  double phi_over_f = phi * 2.435e27 / f_scf;
+
+  /* (m_scf * f_scf)^2 in eV^4, multiplied by 4152.39 to get
    m_pl^2/Mpc^2 */
+  double potential_scale = 4152.39 * pow(m_scf * f_scf, 2);
+
+  /* V_e_scf = 4152.39 * (m*f)^2 * (1 - cos(phi/f))^n */
+  return potential_scale * pow(1.0 - cos(phi_over_f), n_scf);
 }

 double dV_e_scf(struct background *pba,
                 double phi
                 ) {
-  double scf_lambda = pba->scf_parameters[0];
-  //  double scf_alpha  = pba->scf_parameters[1];
-  //  double scf_A      = pba->scf_parameters[2];
-  //  double scf_B      = pba->scf_parameters[3];
-
-  return -scf_lambda*V_e_scf(pba,phi);
+  double f_scf = pba->scf_parameters[0];  /* f_scf in eV */
+  double m_scf = pba->scf_parameters[1];  /* m_scf in eV */
+  double n_scf = pba->scf_parameters[4];  /* n_scf (fixed to
    3) */
+
+  /* Unit conversions as specified in description.md */
+  double phi_over_f = phi * 2.435e27 / f_scf;
+  double potential_scale = 4152.39 * pow(m_scf * f_scf, 2);
+
+  /* dV_e_scf/dphi = potential_scale * n *
    (1-cos(phi/f))^(n-1) * sin(phi/f) * (2.435e27/f_scf) */
+  if (n_scf == 0.0) {
+    return 0.0;
+  } else if (n_scf == 1.0) {
+    return potential_scale * sin(phi_over_f) * (2.435e27 /
    f_scf);
+  } else {
```

```
+    return potential_scale * n_scf * pow(1.0 -
    cos(phi_over_f), n_scf - 1.0) * sin(phi_over_f) * (2.435e27
    / f_scf);
+  }
 }

 double ddV_e_scf(struct background *pba,
                  double phi
                  ) {
-  double scf_lambda = pba->scf_parameters[0];
-  //  double scf_alpha  = pba->scf_parameters[1];
-  //  double scf_A      = pba->scf_parameters[2];
-  //  double scf_B      = pba->scf_parameters[3];
-
-  return pow(-scf_lambda,2)*V_e_scf(pba,phi);
+  double f_scf = pba->scf_parameters[0];  /* f_scf in eV */
+  double m_scf = pba->scf_parameters[1];  /* m_scf in eV */
+  double n_scf = pba->scf_parameters[4];  /* n_scf (fixed to
    3) */
+
+  /* Unit conversions as specified in description.md */
+  double phi_over_f = phi * 2.435e27 / f_scf;
+  double potential_scale = 4152.39 * pow(m_scf * f_scf, 2);
+  double conversion_factor = 2.435e27 / f_scf;
+
+  /* Second derivative: d V_e_scf/dphi  */
+  if (n_scf == 0.0) {
+    return 0.0;
+  } else if (n_scf == 1.0) {
+    return potential_scale * cos(phi_over_f) *
    pow(conversion_factor, 2);
+  } else if (n_scf == 2.0) {
+    double one_minus_cos = 1.0 - cos(phi_over_f);
+    double sin_term = sin(phi_over_f);
+    return potential_scale * pow(conversion_factor, 2) * (2.0
    * sin_term * sin_term + one_minus_cos * cos(phi_over_f));
+  } else {
+    /* General case: d  /dphi [(1-cos(phi/f))^n] */
+    double one_minus_cos = 1.0 - cos(phi_over_f);
+    double sin_term = sin(phi_over_f);
+    double cos_term = cos(phi_over_f);
+
+    double term1 = n_scf * (n_scf - 1.0) * pow(one_minus_cos,
    n_scf - 2.0) * sin_term * sin_term;
+    double term2 = n_scf * pow(one_minus_cos, n_scf - 1.0) *
    cos_term;
+
+    return potential_scale * pow(conversion_factor, 2) *
    (term1 + term2);
+  }
 }


@@ -2985,17 +3038,22 @@ double ddV_p_scf(
 double V_scf(
              struct background *pba,
              double phi) {
-  return  V_e_scf(pba,phi)*V_p_scf(pba,phi);
+  double CC_scf = pba->scf_parameters[3];  /* CC_scf
    cosmological constant contribution */
+
+  /* V_scf = V_e_scf + CC_scf * 3.968e-8 */
+  return V_e_scf(pba, phi) + CC_scf * 3.968e-8;
```

```
 }

 double dV_scf(
               struct background *pba,
               double phi) {
-  return dV_e_scf(pba,phi)*V_p_scf(pba,phi) +
    V_e_scf(pba,phi)*dV_p_scf(pba,phi);
+  /* dV_scf/dphi = dV_e_scf/dphi (since CC_scf term is
    constant) */
+  return dV_e_scf(pba, phi);
 }

 double ddV_scf(
                struct background *pba,
                double phi) {
-  return ddV_e_scf(pba,phi)*V_p_scf(pba,phi) +
    2*dV_e_scf(pba,phi)*dV_p_scf(pba,phi) +
    V_e_scf(pba,phi)*ddV_p_scf(pba,phi);
+  /* ddV_scf/dphi  = ddV_e_scf/dphi  (since CC_scf term is
    constant) */
+  return ddV_e_scf(pba, phi);
 }
diff --git a/source/input.c b/source/input.c
index 933e586c..dc90298c 100755
--- a/source/input.c
+++ b/source/input.c
@@ -3329,12 +3329,14 @@ int
    input_read_parameters_species(struct file_content * pfc,
       }
       else {
         pba->attractor_ic_scf = _FALSE_;
-        /* Test */
-        class_test(pba->scf_parameters_size<2,
-                   errmsg,
-                   "Since you are not using attractor initial
    conditions, you must specify phi and its derivative phi' as
    the last two entries in scf_parameters. See explanatory.ini
    for more details.");
-        pba->phi_ini_scf =
    pba->scf_parameters[pba->scf_parameters_size-2];
-        pba->phi_prime_ini_scf =
    pba->scf_parameters[pba->scf_parameters_size-1];
+
+        /* For EDE model, calculate initial conditions from
    model parameters */
+        /* phi_scf = theta_i_scf * f_scf / 2.435e27 */
+        double f_scf = pba->scf_parameters[0];      /* f_scf
    in eV */
+        double theta_i_scf = pba->scf_parameters[2]; /*
    theta_i_scf (dimensionless) */
+
+        pba->phi_ini_scf = theta_i_scf * f_scf / 2.435e27;
+        pba->phi_prime_ini_scf = 0.0;  /* Set initial
    derivative to zero for EDE model */
       }
     }

diff --git a/source/perturbations.c b/source/perturbations.c
index c78fe437..44ecd9d8 100755
--- a/source/perturbations.c
+++ b/source/perturbations.c
@@ -5472,21 +5472,33 @@ int
    perturbations_initial_conditions(struct precision * ppr,
```

```
       }

       if (pba->has_scf == _TRUE_) {
-        /** - ---> Canonical field (solving for the
    perturbations):
-         *  initial perturbations set to zero, they should
    reach the attractor soon enough.
-         *  - --->  TODO: Incorporate the attractor IC from
    1004.5509.
-         *  delta_phi \f$ = -(a/k)^2/\phi'(\rho + p)\theta \f$,
-         *  delta_phi_prime \f$ = a^2/\phi' \f$ (delta_rho_phi
    + V'delta_phi),
-         *  and assume theta, delta_rho as for perfect fluid
-         *  with \f$ c_s^2 = 1 \f$ and w = 1/3 (ASSUMES
    radiation TRACKING)
+        /** - ---> EDE scalar field (solving for the
    perturbations):
+         *  Implement adiabatic initial conditions as
    specified in description.md
+         *      (0) = (dV_scf * H0   * ktau_three  * R_ini *
    s2_squared *  _rad ) / (210 *  k  )
+         *      '(0) = ( a   * dV_scf * ktau_three * R_ini *
    s2_squared) / (35* k) +
+         *           (dV_scf  *  H 0   * ktau_three  * R_ini
    * s2_squared *   _rad  ) / (10500 *  k  * V_scf)
          */

-        ppw->pv->y[ppw->pv->index_pt_phi_scf] = 0.;
-        /*
    a*a/k/k/ppw->pvecback[pba->index_bg_phi_prime_scf]*k*ktau_three/4.*1./(4.-6.*(1./3.
    * (ppw->pvecback[pba->index_bg_rho_scf] +
    ppw->pvecback[pba->index_bg_p_scf])* ppr->curvature_ini *
    s2_squared; */
-
-        ppw->pv->y[ppw->pv->index_pt_phi_prime_scf] = 0.;
-        /* delta_fld expression * rho_scf with the w = 1/3,
    c_s = 1
-           a*a/ppw->pvecback[pba->index_bg_phi_prime_scf]*( -
    ktau_two/4.*(1.+1./3.)*(4.-3.*1.)/(4.-6.*(1/3.)+3.*1.)*ppw->pvecback[pba->index_bg_
    -
    ppw->pvecback[pba->index_bg_dV_scf]*ppw->pv->y[ppw->pv->index_pt_phi_scf])*
    ppr->curvature_ini * s2_squared; */
+        double dV_scf_val =
    ppw->pvecback[pba->index_bg_dV_scf];
+        double V_scf_val = ppw->pvecback[pba->index_bg_V_scf];
+        double H0_squared = pba->H0 * pba->H0;
+        double H0_fourth = H0_squared * H0_squared;
+        double Omega_rad = 4.15e-5 / (pba->h * pba->h);  /*
     _rad  = 4.15e-5 h^(-2) */
+        double R_ini = ppr->curvature_ini;
+        double k_fourth = k * k * k * k;
+        double k_seventh = k_fourth * k * k * k;
+
+        /*     (0) = (dV_scf * H0   * ktau_three  * R_ini *
    s2_squared *  _rad ) / (210 *  k  ) */
+        ppw->pv->y[ppw->pv->index_pt_phi_scf] = (dV_scf_val *
    H0_squared * ktau_three * ktau_three * R_ini * s2_squared *
    Omega_rad) / (210.0 * k_fourth);
+
+        /*     '(0) = ( a   * dV_scf * ktau_three * R_ini *
    s2_squared) / (35* k) +
+                  (dV_scf  *  H 0   * ktau_three  * R_ini *
    s2_squared *   _rad  ) / (10500 *  k  * V_scf) */
```

```
+         double term1 = (a * a * dV_scf_val * ktau_three *
    R_ini * s2_squared) / (35.0 * k);
+         double term2 = 0.0;
+         if (V_scf_val != 0.0) {
+           term2 = (dV_scf_val * dV_scf_val * dV_scf_val *
    H0_fourth * ktau_three * ktau_three * ktau_three * R_ini *
    s2_squared * Omega_rad * Omega_rad) / (10500.0 * k_seventh *
    V_scf_val);
+         }
+         ppw->pv->y[ppw->pv->index_pt_phi_prime_scf] = term1 +
    term2;
       }

       /* all relativistic relics: ur, early ncdm, dr */
diff --git a/tools/evolver_ndf15.c b/tools/evolver_ndf15.c
index 0d293723..f92e6b0e 100755
--- a/tools/evolver_ndf15.c
+++ b/tools/evolver_ndf15.c
@@ -1084,7 +1084,7 @@ int fzero_Newton(int (*func)(double *x,
      take ntrial Newton-Raphson steps to improve the root.
      Stop if the root converges in either summed absolute
      variable increments tolx or summed absolute function
         values tolf.*/
-  int k,i,j,*indx, ntrial=20;
+  int k,i,j,*indx, ntrial=50;
   double errx,errf,d,*F0,*Fdel,**Fjac,*p, *lu_work;
   int has_converged = _FALSE_;
   int funcreturn;
```

D.3 Model descriptions generated in the Hypothesis Generation phase

D.3.1 One shot experiment

The Early Dark Energy with Running Coupling (EDE-RC) Model
This is a novel early dark energy model that introduces a scalar field component that was significant in the early universe around recombination but becomes negligible today. The key innovation is a "running coupling" mechanism where the effective coupling strength of the scalar field to gravity varies with cosmic time, creating a natural transition from early dominance to late-time suppression.
The model is designed to address the Hubble tension by increasing the sound horizon at recombination while maintaining compatibility with late-time observations. The running coupling ensures the field naturally transitions from being dynamically important around recombination to being cosmologically negligible today.
Mathematical Framework
The scalar field energy density follows a modified exponential decay with running coupling:

$$\rho_{\text{ede-rc}}(a) = \rho_{\text{ede},0} \left(\frac{a_c}{a}\right)^{3(1+w_0)} \exp\left[-\alpha_{\text{run}} \ln^2\left(\frac{a}{a_c}\right)\right]$$

where:

- a is the scale factor
- a_c corresponds to the characteristic scale factor when the field peaks (around recombination)
- w_0 is the base equation of state parameter
- α_{run} is the running coupling parameter that controls the transition sharpness
- $\rho_{\text{ede},0}$ sets the normalization

The equation of state evolves as:

$$w_{\text{ede-rc}}(a) = w_0 - \frac{2\alpha_{\text{run}} \ln(a/a_c)}{3}$$

This ensures the field can have different effective equations of state during different epochs, transitioning from matter-like ($w \approx 0$) during early times to phantom-like ($w < -1$) during the transition, and then becoming negligible.

Model Behavior

Early Universe (z > 1000): The field energy density is subdominant but growing relative to radiation due to the modified scaling.

Around Recombination (z 1100): The field reaches its maximum fractional contribution, providing additional energy density that affects the sound horizon calculation and angular diameter distance.

Late Universe (z < 100): The exponential suppression term dominates, causing the field to become negligible compared to matter and dark energy, ensuring compatibility with late-time observations.

Parameters / .ini Structure for Added Parameters: This model introduces **4** new parameters beyond the standard ΛCDM parameters:

Parameter in .ini	Description	Typical Range (Priors)
`f_ede_rc`	Peak fractional energy density of EDE-RC at characteristic scale a_c	$0.01 \ldots 0.15$
`log10_a_c`	Base-10 logarithm of characteristic scale factor, $\log_{10}(a_c)$	$-3.5 \ldots -2.5$
`w_0_ede`	Base equation of state parameter	$-0.5 \ldots 0.5$
`alpha_run`	Running coupling parameter controlling transition sharpness	$0.5 \ldots 3.0$

Additional background field variables: When running CLASS with this model, the following variables related to the EDE-RC component should be accessible in the `background.dat` output file:

- `w_ede_rc`: The equation of state $w(z)$ of the early dark energy with running coupling fluid
- `rho_ede_rc`: The energy density of the EDE-RC component
- `f_ede_rc`: The fractional contribution $\rho_{\text{ede-rc}}/\rho_{\text{total}}$

Implementation Details

- The field should be implemented as a new fluid component in the background equations
- The energy density evolution should be computed directly from the analytical expression above
- The pressure is derived from $p = w\rho$ where w follows the time-dependent equation of state
- For perturbations, implement using the existing fluid framework with adiabatic sound speed $c_s^2 = 1$
- The field should be inactive ($f_{\text{ede-rc}} \to 0$) for $a \gg a_c$ to ensure late-time compatibility
- Use the parametrization $z_c = 1/a_c - 1$ for the characteristic redshift in user interfaces

Physical Motivation

This model is motivated by theories where scalar field couplings can evolve due to quantum corrections or environmental effects. The running coupling naturally provides a mechanism for early-time activity followed by late-time quiescence, addressing the "why now" problem in dark energy models by having the transition occur at a physically motivated epoch (recombination) rather than today.

D.3.2 One shot + Summary experiment

The Oscillatory Dark Energy with Memory (ODEM) Model
This novel model proposes that dark energy exhibits oscillatory behavior with a "memory" effect, where the current equation of state depends not only on the present cosmic conditions but also retains information about past evolutionary phases. This is physically motivated by quantum field theory in curved spacetime, where vacuum fluctuations can exhibit non-local correlations across cosmic time scales.
The core innovation lies in introducing a time-dependent equation of state that oscillates around -1 but with a decaying amplitude controlled by cosmic memory:

$$w(a) = -1 + A(a) \sin\left(\frac{2\pi \ln(a)}{\Delta \ln a}\right) e^{-\frac{\ln(a/a_m)}{\tau_m}}$$

where:

- a is the scale factor (with $a_0 = 1$ today)
- $A(a) = A_0 \left(\frac{a}{a_0}\right)^{\alpha}$ is the oscillation amplitude that evolves with cosmic expansion
- $\Delta \ln a$ controls the period of oscillations in logarithmic scale factor
- a_m is the memory scale factor (when memory effects become significant)
- τ_m is the memory decay timescale

Key Physics:

- At early times ($a \ll a_m$), the memory term $e^{-\frac{\ln(a/a_m)}{\tau_m}}$ is large, preserving oscillations
- At late times ($a \gg a_m$), memory decays and oscillations are suppressed
- The α parameter controls how amplitude evolves with expansion
- Oscillations can help resolve tensions by providing flexibility in expansion history

Parameters / .ini Structure for Added Parameters: This model introduces **3** new parameters beyond standard CDM:

Parameter in .ini	Description	Typical Range (Priors)
`A0_odem`	Initial oscillation amplitude A_0	$0.01 \ldots 0.5$
`alpha_odem`	Power law index for amplitude evolution α	$-2.0 \ldots 2.0$
`log10_period_odem`	Base-10 logarithm of oscillation period $\log_{10}(\Delta \ln a)$	$-1.0 \ldots 1.0$

Fixed Parameters:

- $a_m = 0.5$ (memory scale factor, fixed to transition epoch)
- $\tau_m = 0.3$ (memory decay timescale, fixed)

Additional background field variables: When running CLASS with this model, the following variables should be accessible in the `background.dat` output file:

- `w_odem`: The equation of state $w(a)$ of the oscillatory dark energy with memory
- `rho_odem`: Energy density of the ODEM component
- `p_odem`: Pressure of the ODEM component

Implementation Details

- The model requires modifications to `background.c` to implement the oscillatory equation of state evolution
- Memory effects are implemented through the exponential decay term in the $w(a)$ expression
- Use the existing fluid approach in CLASS with `fluid_equation_of_state = "odem"`
- The oscillatory nature provides rich phenomenology while the memory effect ensures physical regularity
- Integration should use adaptive step sizes to handle oscillatory behavior accurately