# 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 how the Universe evolved from its earliest moments to today relies on the existence of dark energy and dark matter—mysterious components detectable only through their gravitational effects, despite accounting for 95% of the Universe. Recent surveys reveal systematic discrepancies in the temporal evolution of dark energy, potentially pointing toward new physics. Given the success of Large Language Models at completing research-level tasks such as coding and mathematical reasoning, we investigate the capability of LLMs to autonomously propose, implement, and test different cosmological theories. We leverage our framework to challenge Claude Code in three experimental settings: (1) implementing alternative models from curated descriptions by modifying a physics simulation codebase, (2) performing the same implementation directly from research papers, and (3) generating novel hypotheses for dark energy evolution to better explain recent observations. Across two benchmark models with ground-truth implementations, Claude Code successfully implemented a "Thawing Quintessence" dark energy model. However, it failed to generate correct observables for a more complex "Early Dark Energy" model despite successful code compilation. When working directly from papers rather than curated descriptions, numerical accuracy degraded significantly though qualitative behavior remains correct. Most remarkably, Claude Code's self-proposed dark energy model achieved a better statistical fit to observations than the standard model, though at the cost of additional parameters.

## 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 ($\Lambda$CDM) model, our current standard model of cosmic evolution, has achieved remarkable success in predicting observations from the formation of the first atoms to large-scale galaxy clustering. However, this success relies on "dark matter" and "dark energy"—mysterious components detectable only through their gravitational effects, and whose true nature remains elusive, despite accounting for 95% of the Universe. Beyond these theoretical issues lies also an empirical problem: even accepting dark matter and dark energy as given, $\Lambda$CDM increasingly struggles to fit high-precision data. Recent surveys reveal systematic discrepancies that may hold the key to solving both puzzles—the specific ways $\Lambda$CDM fails could reveal the true nature of these dark components (DESI Collaboration et al., 2025; Lodha et al., 2025). Exploring alternative models becomes now an urgent priority.

Resolving these cosmological 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, dynamical dark energy models etc. Each theoretical framework introduces new parameters, functional forms, and physical assumptions, creating a high-dimensional space where most regions remain unexplored due to computational and human limitations. Large language models, trained on the vast literature of theoretical physics, could potentially sample from this space of hypothesis systematically, generating novel combinations and extensions that human researchers might not consider.

Testing cosmological theories requires specialized simulation codes, such as the Cosmic Linear Anisotropy Solving System (CLASS) and Code for Anisotropies in the Microwave Background (CAMB), that take theoretical parameters as input and produce synthetic observational data as output. Implementing an alternative cosmological model involves a canonical workflow in which these standard codebases are modified to accommodate the new physics. As a next step, researchers evaluate Bayesian posteriors on the observables and assess whether the proposed model can explain the data better than $\Lambda$CDM does.

Model building in cosmology makes for a compelling yet challenging LLM application for several reasons. Firstly, almost all papers implementing an alternate cosmological model rely on 2-4 codebases. Foremost among these is the CLASS codebase. Few subfields in the sciences have such a *unified workflow*. Secondly, these alternate theories need to pass a set of common tests that evaluate their ability to reproduce observations. From a machine-learning perspective, this means that cosmological model discovery can be reformulated as a setting in which all models are tested against and optimized against a set of *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 that establishes a systematic test bed in which to measure the ability of an LLM-driven agent to autonomously propose, implement and explore cosmological models. We measure both the potential of the agentic framework to correctly implement a given cosmological model, as well as propose its own theoretical model.

## 2 Environment Design

The agent performs its tasks in a containerized code execution environment using a Podman-based emulator for Docker. Containerized code execution enables us to control which files, directories and packages the agent has access to and / or can modify. The environment is designed to eliminate the possibility of the agent accessing the internet to look up information or code (see Appendix B for details).

The agent will modify the physics simulation codebase of CLASS (Lesgourgues, 2011). The CLASS code is available at `https://github.com/lesgourg/class_public`. The CLASS machinery is implemented in C, although it can additionally be used to compute observables via a Python wrapper to the C code (Classy). The LLM agent is launched in a directory that contains the base CLASS repository, instantiated as a GitHub repository, in order to easily

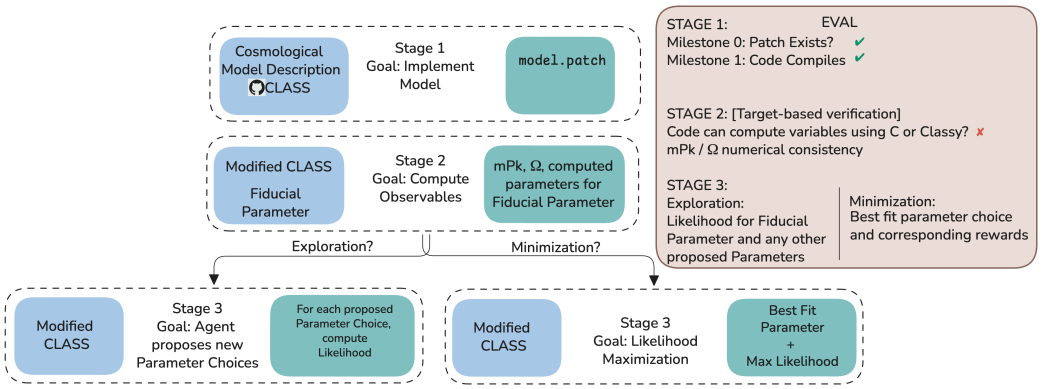


Figure 1: Flow diagram depicting the input state for each stage (blue) the output artifacts (green) and the task that the agent has to perform. Once the agent has completed its work, the outputs are automatically evaluated on the basis of both binary milestones that track initial progress as well as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

export the changes as a patch. The repository additionally contains contextual prompts with instructions for each stage, information about CLASS, the cosmological-model specific information as well as auxiliary helper and testing code. We work with the Claude Code (Anthropic, b) agentic framework.

### 2.1 Cosmological Model Specification

Cosmological observables, such as the matter power spectrum, which quantifies the clustering of matter at different scales, are determined by both the underlying theoretical model as well as the specific values of its characteristic parameters. We use $\mathcal{M}$ to denote a theoretical model and $\mathcal{P}$ to denote a specific point in parameter space for the relevant cosmological model.

A model instance typically consists of the following files:

- Model Description [`description.md`]: This is a file containing the description of the cosmological model of interest, $\mathcal{M}$. Appendix A.2.2 includes the description for one of the models we evaluated on.
- Fiducial Parameter Files: The fiducial parameter set is used to test whether the modified code can actually be used to compute cosmological observables. We refer to this parameter set as $\mathcal{P}_{Fiducial}$.
- Other problem-specific contextual variables [`problem_context.yaml`]: This is a YAML file that seeks to account for any other problem specific contextual variations and instructions such as the manner in which the agent should explore the models parameter space, as well as the observables that need to be computed and saved.

## 3 Stages

### 3.1 Stage 1: Model Implementation

In the first stage, the agent's objective is to implement the cosmological model description by modifying the CLASS code, to enable it to work with the fiducial parameter file. The agent is instructed to read the provided model description. It must then modify code across the CLASS code base and seek to compile the modified CLASS. The core CLASS machinery has over 43000 lines of C code, with another 10000 lines in auxiliary numerical files. To mitigate the challenge of navigating the code base, we added three 'filemap' files: high-level

summaries of the organization of the 'background', 'input' and 'perturbations' files, in addition to the model description and the fiducial parameter file. The desired state of the environment upon the completion of this stage is code that now successfully implements the alternate cosmological model.

### 3.2 Stage 2: Computing Observables

The agent must now use its modified code to compute observables for the fiducial cosmology, using the provided parameter files. The agent is instructed to use a helper script that computes observables using the modified code, that should now implement $\mathcal{M}$. These observables are relevant to the target artifact-based evaluation.

If the code fails to execute, another LLM (`o1-2024-12-17`) (OpenAI) is queried to provide feedback comparing the description of the cosmological model with the patch and identify any inconsistencies. This functions as an in-the-loop LLM-as-Judge setup. If the code computes observables successfully, a Pickle file containing these observables is saved, and this serves as the key output artifact relevant to Stage 2. The agent is also instructed to call a module that displays the present day values of key computed cosmological parameters for the $\mathcal{P}_{Fiducial}$ to confirm if the values seem reasonable, as a self-consistency test.

### 3.3 Stage 3: Parameter Exploration

In this stage, the agent is instructed to explore other parameters and compute the log likelihood with respect to observational data for both any new parameters and $\mathcal{P}_{Fiducial}$. In this case, we will compare the theoretical models to observations on how galaxies cluster on large scales at late times. In particular, we use data from the DESI collaboration (DESI Collaboration et al., 2025), and combine its likelihood with the early Universe priors described in Appendix A of (DESI Collaboration et al., 2025).

We consider two pathways in Stage 3. The '*Exploration*' pathway seeks to mirror the model exploration sections of several theoretical cosmology papers, in which cosmologists manually vary key parameter values to preliminarily assess the model's behavior in different physical limits and examine its merits. This pathway is also a less prohibitive option if the theoretical model happens to be one that is challenging to optimize, as in the *Early Dark Energy* model. In this pathway, we essentially ask the agent to propose and explore alternate parameter choices, $\{\mathcal{P}\}$ 'manually'.

In the '*Minimization*' pathway, the agent must algorithmically find the subset of parameters that best fits the observations. Here, we optimize the parameter file with respect to the likelihood. The best-fit parameter, that maximizes this combined likelihood is subsequently referred to as $\mathcal{P}_{Best-Fit}$.

## 4 Evaluation

Our setup is designed so that once the agent has completed its tasks its changes are saved in a GitHub patch in a 'submission' folder on the host system. We then separately launch another Docker container and run an evaluation script that applies the patch containing the changes the agent made and evaluates its progress on both binary milestones and continuous evaluation metrics. The first 2 binary milestones examine whether it was able to generate a `model.patch` file that produces code that can compile. The next binary milestone checks whether the compiled code can be used to compute observables with $\mathcal{P}_{Fiducial}$.

The evaluation for this stage focuses on *correctness*, and seeks to assess whether the agent correctly implemented the given cosmological model. We numerically compares certain key computed physical quantities using the agent's codebase implementation with those generated by a ground truth repository implementing the same cosmological model. The quantities we consider are the matter power spectrum at two different redshifts[1], $z = 0$ (present-day) and $z = 1$, and the evolution of the energy density of the relevant dark energy

[1]The redshift denoted by $z$, is a cosmological proxy for time.

component as a function of redshift, denoted by $\Omega_X$ where X is a subscript determining the specific kind of dark energy field (fld for fluid, in Figure 2). We report the maximum fractional deviation with respect to the target repository in Table 1. We compute this as follows: $\max \frac{Q_{LM} - Q_{Target}}{Q_{Target}}$, where the maximum is taken over the independent variable ($k$ for the power spectra and $z$ for the density evolution). For the evaluation of Stage 3, we compute how well the agent implemented theoretical framework $(\mathcal{M}, \mathcal{P})$ fits data, for all $\mathcal{P}$ explored / derived via optimization.

## 5 Results

Unless otherwise specified, we used the `claude-sonnet-4-20250514` model (Anthropic, a) in all experiments.

### 5.1 Description of Physical Models

We examined the performance of Claude Code on two cosmological models: in increasing order of difficulty, a phenomenological *Thawing Quintessence* (Payeur et al., 2025) model that mimics the qualitative behavior of (late-time) dark energy, and (2) an *Early Dark Energy* (EDE) model (Karwal & Kamionkowski, 2016; Poulin et al., 2019; McDonough et al., 2024; Kamionkowski & Riess, 2023; Poulin et al., 2023), a particle physics inspired extension to standard cosmology. Both models have ground truth implementations available (but known to be public only for EDE).

For a successful implementation of *Thawing Quintessence*, the model needs to introduce the correct evolution of the dark energy density as a function of time, denoted as $w(z)$, together with its additional parameters and analytical derivative. In this case, dark energy behaves as a homogeneous fluid and to compute the dark energy evolution the model can use the existing fluid evolution framework in CLASS.

While fluid models with $w(z)$ profiles prescribe how dark energy's pressure-to-density ratio evolves over time, EDE requires solving for a fundamental dynamical variable that must achieve a specific behavior—rapidly rising to 10% of the universe's energy at a precise epoch before quickly decaying—through solving a new set of differential equations rather than simply specifying this evolution. This dynamical variable creates its own fluctuations that interact with matter fluctuations at all scales, requiring coupled equations where the variable's evolution both depends on and influences the universe's expansion rate, unlike fluid models where the energy evolution is given as an input. The implementation complexity arises because we must compute from first principles how this dynamical component evolves and gravitationally affects all other components at every scale, rather than treating EDE as having only aggregate properties that affect the background expansion—making the computational framework substantially more involved than the prescriptive fluid approach used for *Thawing Quintessence*.

The results of the evaluation are in Table 1. Whereas the model can successfully implement the *Thawing Quintessence* model, it fails to successfully compute observables for the EDE model. For the EDE model, the agent inconsistently integrates a density parameter for the EDE model with certain functions despite an implementation that is otherwise promising. This breaks a section of the code that adjusts the density parameter via a numerical shooting algorithm. The generated code patch for *Thawing Quintessence* can be found in Appendix D.

### 5.2 '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 (Payeur et al., 2025), to examine whether the agent is still able to generate working code given just the paper, as opposed to a well-specified problem description. We still include the fiducial cosmology parameter file in this setting so we can later perform the Stage 2 evaluation against numerical target artifacts. The agent is still able to generate executable code. However, the comparison against the target

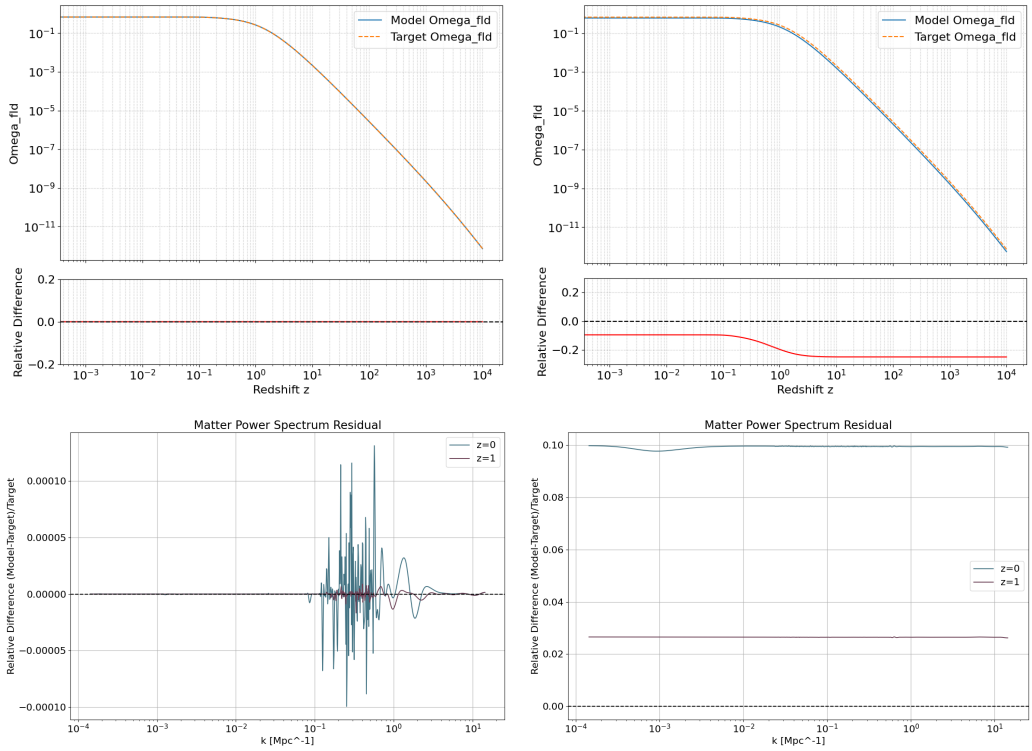


Figure 2: **Target Artifact-based evaluation:** The evaluation for Stage 2 compares the evolution of the relevant dark energy component ($\Omega_{fld}$ for *Thawing Quintessence*) and the matter power spectrum at two redshifts against their corresponding ground-truth values generated by a target implementation of the model. While the agent is able to generate executable code when provided with the *paper (right column)*, the numerical errors are much higher than when it is given a condensed, yet detailed *description (left column)*.

artifacts, in Figure 2 reveals significantly higher numerical inconsistencies relative to the condensed 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. These differences do not seem to impact the best fit likelihood values of the models (Table 1).

### 5.3 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 implementing it by modifying CLASS. The model is instructed to read an auxiliary prompt and propose its *own* cosmological model, prioritizing the following criteria (in order of importance): (1) novelty, (2) ability to alleviate cosmological tensions and simultaneously providing a good fit with respect to data, and (3) ease of implementation in CLASS. We additionally provide it with an example description derived from the description for *Thawing Quintessence* and an example parameter file for *Thawing Quintessence*. It must first create its own `description.md` file describing its proposed cosmological model, $\mathcal{M}$, and a corresponding $\mathcal{P}_{Fiducial}$, and then proceed with the remaining stages as before. Since there doesn't exist a target implementation in this setting, the Stage 2 target-based is skipped in the evaluation of the agent's submission.

We ran two experiments in this setting (summarized in Table 2). In the first experiment, the agent's proposed model implementation fails to cross the second milestone. In the second experiment, the agent succeeds at generating executable code for its model. The model proposes a phenomenological dark energy model with the following oscillatory equation of state

$$w(z) = w_0 + w_a \cdot \frac{z}{1+z} + A_{\text{osc}} \cdot \exp(-\alpha z) \cdot \sin\left(\omega \cdot \ln(1+z) + \phi\right), \tag{1}$$

where $z$ is the cosmological redshift, $w_0$ is the present-day equation of state , $w_a$ controls the linear evolution with redshift, $A_{\text{osc}}$ is the amplitude of oscillations, $\alpha$ is the damping parameter that causes oscillations to decay at high redshift, $\omega$ controls the frequency of oscillations in log(1+z) space, and $\phi$ is the phase offset of the oscillations.

In Stage 3, its $\mathcal{P}_{Best-Fit}$ value yields a likelihood of -12.59. For reference, the $\mathcal{P}_{Best-Fit}$ for $\Lambda$CDM, the canonical model, yields a likelihood of -12.88. However, the agent appears to achieve the superior fit by proposing a six parameter extension to $\Lambda$CDM (including $w_0$ and $w_a$). Adding more parameters to a model should in principle improve fits to data, since the theory now possesses more degrees of freedom.

Table 1: Results for Target Artifact-based Evaluation Experiments

| Model | Early Dark Energy | Thawing Quintessence | Thawing Quintessence [from paper] |
|---|---|---|---|
| Milestone 0 [Patch Exists?] | ✓ | ✓ | ✓ |
| Milestone 1 [Code compiles?] | ✓ | ✓ | ✓ |
| Milestone 2 [Observables can be computed?] | × | ✓ | ✓ |
| $\Omega_X(z)$ Max Fractional Deviation | - | $4.43e-10$ | 0.248 |
| $mPk(z=0)(k)$ Max Frac. Deviation | - | $1.31e-4$ | 0.0997 |
| $mPk(z=1)(k)$ Max Frac. Deviation | - | $1.67e-05$ | 0.0266 |
| Stage 3 Evaluation | - | | |
| DESI-*Planck* $\mathcal{P}_{Fiducial}$ $ln\mathcal{L}$ | | $-4.21e3$ | $-47.8$ |
| DESI-*Planck* $\mathcal{P}_{Best-Fit}$ $ln\mathcal{L}$ | | $-14.7$ | $-14.8$ |

Table 2: Results for LLM-proposed model experiments

| Model | Novel Run 1 | One-Shot | Novel Run 2 | One-Shot |
|---|---|---|---|---|
| Milestone 0 [Patch Exists?] | ✓ | | ✓ | |
| Milestone 1 [Code compiles?] | ✓ | | ✓ | |
| Milestone 2 [Observables can be computed?] | × | | ✓ | |
| Stage 3 Reward | - | | | |
| DESI-*Planck* $\mathcal{P}_{Fiducial}$ $ln\mathcal{L}$ | | | $-111$ | |
| DESI-*Planck* $\mathcal{P}_{Best-Fit}$ $ln\mathcal{L}$ | | | $-12.6$ | |

## 6 Conclusions

This work introduces an end-to-end framework that seeks to replicate the cosmological model-building research pipeline and examines the ability of an LLM-based agent, Claude Code, to execute multiple stages of the process. We first examine performance on two cosmological models for which ground truth codebase implementations are available: while the agent is able to generate code that compiles for both models, it only crosses all milestones and is able to complete Stage 3 for one. We further performed an ablation test by providing the agent with the text of the paper, instead of the well-specified problem description. This setting raises the challenge of the task since the agent must now also *extract* the most relevant information to modifying the codebase. Lastly, we took a step in the direction of automating theory development by additionally entasking the model to now devise and implement its own cosmological model. While the model succeeds at creating a model that attains a slightly better fit to data, it does so at the cost of introducing additional parameters.

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, mathematical implementation, and empirical validation. While our experiments measured agentic ability with minimal human interaction (only to grant permissions), the framework directly accommodates greater human interaction. By accelerating the pace of cosmological model exploration such pipelines may facilitate progress on one of the most pressing questions in modern physics: understanding the nature of dark energy.

Our current framework represents initial progress on the route to automated cosmological model discovery. 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 likelihood to iteratively guide the discovery of improved dark energy models. 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.

Second, we will expand our current likelihood-based reward function to include other rewards, such as the ability to alleviate other cosmological tensions such as the $H_0$ tension, and compute Bayesian evidence ratios comparing proposed models to $\Lambda$CDM. The inclusion of the latter addresses a critical limitation of our current approach: models with additional parameters can achieve better fits simply by overfitting the data. By using evidence ratios as rewards, we will naturally penalize models for their increased complexity while rewarding simpler extensions that can still explain the observations. This Bayesian framework will provide a more principled basis for model selection and better align our automated discovery process with standard practices in cosmological model comparison.

Together, these enhancements will create a more robust system for autonomous cosmological theory development, capable of both broader exploration and more rigorous evaluation of proposed alternatives to our current understanding of the cosmos.

## 7 Acknowledgements

This work was supported by the National Science Foundation under Cooperative Agreement PHY2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions). We are grateful to Mikhail Ivanov and Siddharth Mishra-Sharma for helpful conversations in the early stages of this work.

## References

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

Anthropic. Claude code, b. https://docs.anthropic.com/en/docs/claude-code/overview.

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.

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

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.

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.

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

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.

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.

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

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

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.

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.

# Appendix

## A Cosmological Models

### A.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 $\mathcal{P}_{Fiducial}$ dictionary 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*

### A.2 Problem Instances

#### *A.2.1 Early Dark Energy*

The dark energy density parameter of interest here is $\Omega_{scf}$.

#### *A.2.2 Thawing Quintessence*

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

**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, $\phi$ eventually unfreezes. At that point, w departs from -1, and the ensuing evolution of $\phi$ and w depends on the scalar potential V ($\phi$). 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.

- $\Delta w$ 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 $\Delta w$ 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.
- $\Delta z$ 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 ($\Lambda$CDM) if $\Delta w = 0$. It can also appear indistinguishable from $\Lambda$CDM 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 $\Delta z$).
**Parameters / .ini Structure for Added Parameters:** This model introduces **3** new parameters beyond the standard $\Lambda$CDM 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 $w(z)$. | $0 \ldots 2$ |
| `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. | $-2 \ldots 2$ |

**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.

### *A.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.

**Propose Model Prompt**

You are a cosmologist, seeking to propose a beyond-Lambda CDM model that will resolve current outstanding issues / tensions in cosmology. Specifically, you want to introduce a model that satisfies the following criteria (in order of importance):

1. Novelty: You should propose a new model that has * not * been implemented / explored in the literature.
2. Your model must optimally satisfy the following four criteria (rewards):

a) Minimizes the Hubble Tension (current discrepancies in the H0 value inferred from late-time model independent observations and "early-universe" measurements)
b) Minimizes the S8 tension

c) Has a high likelihood with respect to DESI BAO data and a Planck marginal likelihood
d) Is consistent with LCDM Cls

3. You will have to implement this model by refactoring the cosmological code CLASS (**Cosmic Linear Anisotropy Solving System**)** (you are currently in this codebase). So you may strategically choose to propose a model that you think would be easier for you to implement in CLASS.

Create a model description as in the example below:
EXAMPLE 1:
EXAMPLE DESCRIPTION.MD: `<Example description derived from>` *Thawing Quintessence*
EXAMPLE PARAM_BASE.JSON: $\mathcal{P}_{Fiducial}$ `dictionary for` *Thawing Quintessence*
===
You will need to create a description.md file and a param_base.json file. After doing this, proceed to complete the implementation related tasks in stage1_prompt.md, stage2_prompt.md and stage3_prompt.md. Note, even if the other prompts discourage you from modifying param_base.json, you **are** allowed to modify it to to get it to work with your implemented model in CLASS, if for example, it needs extra flags to work with Classy (the Python wrapper to CLASS) successfully.

## B 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.

### B.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` (Google LLC) a prompt along with the saved module with corresponding line numbers.

### B.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.

## C Pipeline Details

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`.

## D Experiment Results

For the experiments we discuss here, the cost of a single experiment with Claude Sonnet-4 is less than $5: $3.75 for EDE, $1.91 for Thawing Quintessence ($1.92 with the paper as input), and $4.52 for the Hypothesis Generation experiment with the code that builds successfully.

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..308bf546 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$: amplitude of
    the transition in w(z) for tanh parametrization */
+  double dz_tanh;          /**< \f$ \Delta z \f$: width of the
    transition for tanh parametrization */
+  double z_tanh;           /**< \f$ z_c \f$: redshift at center
    of transition for tanh parametrization */
   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..ce472491 100755
--- a/source/background.c
+++ b/source/background.c
@@ -702,6 +702,14 @@ 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:
+    {
+      // w(z) = (Delta_w/2) * (1 - tanh((z - z_c)/Delta_z)) - 1
+      // Convert scale factor to redshift: z = 1/a - 1
```

```
+      double z = 1./a - 1.;
+      *w_fld = (pba->dw_tanh/2.) * (1. - tanh((z -
    pba->z_tanh)/pba->dz_tanh)) - 1.;
+    }
+    break;
   }


@@ -721,6 +729,18 @@ int background_w_fld(
       + dOmega_ede_over_da*dOmega_ede_over_da*a/3./(1.-Omega_ede)/
       (1.-Omega_ede)/Omega_ede
       + a_eq/3./(a+a_eq)/(a+a_eq);
     break;
+  case TANH:
+    {
+      // dw/da = dw/dz * dz/da = dw/dz * (-1/a^2)
+      // dw/dz = (Delta_w/2) * (-1/Delta_z) * sech^2((z -
    z_c)/Delta_z)
+      //       = -(Delta_w/2) * (1/Delta_z) * (1 - tanh^2((z -
    z_c)/Delta_z))
+      double z = 1./a - 1.;
+      double tanh_arg = (z - pba->z_tanh)/pba->dz_tanh;
+      double tanh_val = tanh(tanh_arg);
+      double dw_dz = -(pba->dw_tanh/2.) * (1. -
    tanh_val*tanh_val) / pba->dz_tanh;
+      *dw_over_da_fld = dw_dz * (-1./(a*a));
+    }
+    break;
   }

   /** - finally, give the analytic solution of the following
       integral:
@@ -740,6 +760,10 @@ 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:
+    // For TANH, we set integral_fld to 0 and handle it
    numerically in background_initial_conditions
+    *integral_fld = 0.;
+    break;
   }

   /** note: of course you can generalise these formulas to
       anything,
@@ -2251,6 +2275,28 @@ int background_initial_conditions(
        [(1+w_fld)/a] da] (e.g. with the Romberg method?) instead
            of
        calling background_w_fld */

+    /* For TANH parametrization, compute integral numerically if
    needed */
+    if (pba->fluid_equation_of_state == TANH && integral_fld ==
    0.) {
+      // Compute integral numerically: int_{a_ini}^{a_0}
    3(1+w_fld)/a da
+      // Using simple trapezoidal rule with sufficient steps
+      int n_steps = 1000;
+      double da_step = (1.0 - a) / n_steps;
+      double integral_sum = 0.0;
+      double a_temp, w_temp, dw_temp, integral_temp;
```

```
+
+      for (int i = 0; i <= n_steps; i++) {
+        a_temp = a + i * da_step;
+        class_call(background_w_fld(pba, a_temp, &w_temp,
    &dw_temp, &integral_temp), pba->error_message,
    pba->error_message);
+        double integrand = 3.0 * (1.0 + w_temp) / a_temp;
+        if (i == 0 || i == n_steps) {
+          integral_sum += 0.5 * integrand * da_step;
+        } else {
+          integral_sum += integrand * da_step;
+        }
+      }
+      integral_fld = integral_sum;
+    }
+
     /* 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..3e7f30e9 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 = 1.0;
+  pba->dz_tanh = 0.03162277660168379; // 10^(-1.5)
```

```
+  pba->z_tanh = 0.1;
   /** 9.b) Omega scalar field */
   /** 9.b.1) Potential parameters and initial conditions */
   pba->scf_parameters = NULL;
```