Can LLMs Solve Ordinary Differential Equations? A Comparative Analysis

Domingos Samanjata[1,2], **René Vieira Santin**[1], **Ricardo Marcondes Marcacini**[1], **Solange Oliveira Rezende** [1]

[1] Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP), Avenida Trabalhador são-carlense, São Carlos, 400 - Centro CEP: 13566-590, São Paulo, Brasil.

[2]Instituto Politécnico da Universidade José Eduardo dos Santos (IP-UJES), Bairro Académico, Rua Comandante Nzage, Huambo-Angola.

{kasams,renevs,ricardo.marcacini}@usp.br, solange@icmc.usp.br

Abstract. *Solving Ordinary Differential Equations (ODEs) requires mathematical modeling and reasoning skills that remain challenging for large language models (LLMs). This paper evaluates five LLMs, DeepSeek-R1, Claude-Haiku-4.5, Llama-3.3-70B-Instruct, Gemini-2.5-Pro, and GPT-5.2, using ten first-order ODE application problems and a task involving analogous problem generation. We propose a multi-dimensional framework assessing mathematical modeling, Python code generation, solution accuracy, and the generation of analogous problems. Results show strong modeling performance across all models (96%–98%), demonstrating their ability to translate natural language descriptions into mathematical formulations; when generating analogous problems, scores ranged from 85% to 100%, enabling direct performance comparisons against each model's own baseline. However, performance declines in code generation tasks, exposing limitations in multi-step reasoning and execution. Claude-Haiku-4.5 achieves the highest overall score (84%), while Llama-3.3-70B-Instruct performs best in code generation. GPT-5.2 records the lowest overall performance. These findings reveal a gap between conceptual understanding and reliable end-to-end problem solving, highlighting the importance of human oversight and hybrid symbolic-numerical approaches.*

1. Introduction

Recent advances in Large Language Models (LLMs) have demonstrated promising results in mathematical problem-solving. However, despite strong performance on general benchmarks, it remains uncertain whether LLMs genuinely engage in mathematical reasoning or primarily depend on memorization and heuristic patterns. Prior studies suggest that LLMs may fail when problem formulations are slightly modified, which suggests limited robustness in their reasoning processes [Patel et al. 2021, Li et al. 2023]. Furthermore, evaluating the correctness of their solutions and intermediate reasoning steps is increasingly challenging as model complexity grows [Azerbayev et al. 2024, Cobbe et al. 2021]. Therefore, assessing LLMs' mathematical reasoning and problem-solving capabilities requires a rigorous, structured evaluation process that thoroughly examines the depth, consistency, and robustness of their performance. Although recent research increasingly explores LLMs' ability to solve math problems, comprehensive reviews of their

performance in specific domains are still needed. General assessments exist, but systematic evaluations in specialized areas, especially higher-level mathematics such as calculus, are rare. This gap presents a challenge for researchers seeking evidence-based approaches to integrating these tools into instructional practice.

While large language models (LLMs) have shown promise in mathematical reasoning, their application to ordinary differential equations (ODEs) has received surprisingly little attention, even though ODEs are fundamental to most engineering and scientific disciplines. Existing work remains limited in scope and lacks the scale and targeted evaluation needed for advanced problems [Hendrycks et al. 2021, Lee et al. 2024, Huang et al. 2025]. LLMs offer advantages by reducing reliance on expert-driven methods, supporting automation and broader access to symbolic mathematical modeling.

The primary contributions of this paper are as follows:

- Development of a comprehensive evaluation framework to assess the mathematical modeling abilities of state-of-the-art large language models (LLMs) on open-ended first-order ordinary differential equation (ODE) problems.
- An evaluation tool for open-ended first order ODE problems, which enable reproducible comparisons of LLM performance, and provide both quantitative and qualitative insights to guide future advancements in mathematical reasoning.
- Evaluation of LLMs for generating analogous problems aimed at expanding the currently under-development ODE benchmark[1]

2. Related Work

Recent benchmarks have advanced the evaluation of mathematical reasoning in large language models by emphasizing both scale and diversity. Datasets such as Omni-MATH [Gao et al. 2023], FrontierMath [Glazer et al. 2024], OMEGA [Sun et al. 2025], and RealMath [Zhang et al. 2023] include problems spanning Olympiad to research levels and cover domains including number theory, real analysis, and algebraic geometry. These initiatives demonstrate an increasing emphasis on comprehensive and multi-level assessment.

In addition, smaller and more targeted benchmarks investigate specific reasoning skills. Combi-Puzzles [Nikolaiev et al. 2024] examines combinatorial reasoning through structured variations, whereas TriMaster100 [Zhao et al. 2024] addresses multi-step trigonometric reasoning using stepwise chain-of-thought approaches. Although these datasets provide valuable insights into structured reasoning, they remain domain-specific and do not prioritize mathematical modeling.

Recent studies have investigated specialized domains such as differential equations and optimization. Mamo [Huang et al. 2025] provides a dataset containing over one thousand problems, encompassing linear programming, mixed-integer programming, and ordinary differential equations. However, these benchmarks are multi-domain and generally do not focus on specific problem types. Samanjata et al. (2025) assessed both proprietary and open-source large language models (LLMs) on a benchmark of ordinary differential equation (ODE) problems, highlighting persistent shortcomings in their capacity for open-ended mathematical reasoning in this domain [Samanjata et al. 2025].

[1]Available at: `https://github.com/Domingos-hub/ODE-Benchmark-Dataset`

In contrast, this study examines open-ended ordinary differential equation problems, emphasizing the derivation of mathematical models from natural language descriptions and the evaluation of LLMs in generating analogous open-ended first-order ODE problems. While previous benchmarks have improved diversity and performance through scaling and tool integration, they remain limited, as their broad topical coverage prevents precise identification of the domains in which LLMs perform well or struggle. Previous research frequently incorporates external tools to enhance performance [Chen and Goodman 1999, Gao et al. 2023, Yue 2025], which can obscure the distinction between reasoning and computation and hinder the assessment of intrinsic reasoning capabilities. Although earlier benchmarks have improved diversity and performance through scaling and tool integration, they often overlook the evaluation of end-to-end reasoning in mathematical word problem modeling tasks. The present study addresses this gap by isolating intrinsic reasoning and emphasizing the complete modeling-to-solution process. Table 1 illustrates the primary distinctions between existing benchmarks, prior research, and the proposed framework.

3. Methodology

We introduce a structured evaluation framework for assessing the ability of large language models (LLMs) to solve open-ended first-order ordinary differential equation (ODE) problems and generate analogous ODE problems. The framework enables a detailed analysis of model behavior, capturing not only the correctness of solutions but also the complete problem-solving process, from natural language interpretation to symbolic ODE modeling and executable code generation. We first present and classify the ODE problems used in this paper by category. We then describe the prompting strategy employed for the LLMs and, finally, the evaluation criteria used to assess their performance.

3.1. Problem Selection and Classification

A set of 10 open-ended first-order ordinary differential equation (ODE) problems was carefully selected to represent fundamental application domains in engineering, physics, biology, and economics. Rather than being randomly chosen, these problems were systematically designed to reflect real-world scenarios commonly addressed in university-level differential equations courses. Each problem is carefully selected to be representative of its category and maintains a level of difficulty consistent with university-level curricula. These problems target specific skills and conceptual understanding, with complexity aligned to standard calculus curricula and a focus on core applications of first-order ODEs. The problems are as follows:

1. **Newton's Law of Cooling:** Models heat transfer between an object and its environment using $\frac{dT}{dt} = -k(T - T_{amb})$. This problem evaluates the ability to identify physical laws and correctly formulate temperature dynamics.
2. **Mixing Problems:** Describes solute concentration changes with $\frac{dQ}{dt} = inflow - outflow$. This problem assesses the translation of flow processes into first-order differential equations.
3. **Electrical Circuits (RL):** Models current dynamics using $L\frac{dI}{dt} + RI = V(t)$. This problem evaluates the formulation of circuit-based differential equations.
4. **Exponential Population Decay:** Represents decay processes with $\frac{dy}{dt} = -ky$. This problem assesses the identification and modeling of exponential decay behavior.

Table 1. Comparison between related work and our proposed evaluation framework

Work	Open-ended ODE focus	Natural Language to ODE modeling	Step-by-step reasoning	External tools / solvers	Solution Verifiers	Multi-step reasoning evaluation
Omni-MATH [Gao et al. 2023]	–	–	✓	–	–	–
FrontierMath [Glazer et al. 2024]	–	–	✓	–	–	–
RealMath [Zhang et al. 2023]	–	–	✓	–	–	–
Huang et al. (2025) Mamo ODE benchmark	✓	✓	✓	✓	–	–
Shen et al. (2021) GSM8K Dataset	–	–	✓	–	✓	–
Ours	✓	✓	✓	✓	✓	✓

5. **Electrical Circuits (RC):** Describes charge dynamics using $R\frac{dQ}{dt} + \frac{Q}{C} = V(t)$. This problem evaluates the correct modeling of capacitor-based systems.
6. **Exponential Population Growth:** Models exponential growth using $\frac{dy}{dt} = ky$. This problem assesses the ability to distinguish and formulate growth processes.
7. **Motion with Air Resistance:** Models motion under gravity and drag with $m\frac{dv}{dt} = mg - kv$. This problem evaluates the formulation of physical dynamics from force descriptions.
8. **Economic Decay (Inflation):** Represents economic decay processes using $\frac{dy}{dt} = -ky$. This problem evaluates the translation of economic scenarios into mathematical models.
9. **Chemical Kinetics Decay:** Models decay processes using $\frac{dy}{dt} = -ky$. This problem assesses the identification of exponential decay in physical systems.
10. **Economic (Compound Interest Growth):** Models financial growth using $\frac{dy}{dt} = ky$. This problem evaluates the formulation of exponential growth in economic contexts.

This classification framework enables a systematic and comprehensive evaluation of diverse open-ended first-order ODE problems, allowing for a detailed analysis of the strengths and limitations of each model's mathematical reasoning capabilities.

3.2. LLM Prompting Strategies

We define open-ended ODE problem prompt using three-shot prompting, where each prompt p consists of three example pairs: p = [(x_1, y_1),(x_2, y_2),(x_3, y_3)], as illustrated in Figure 1. Each x_i represents an ODE problem stated in natural language, and each y_i is the corresponding step-by-step solution implemented using SCoT. The three illustrative examples are chosen specifically for the ODE category under evaluation, and are based on prior experiments that showed low performance on these problem types. The examples used are an exponential growth application, a Newton's Law of Cooling scenario, and an electric circuit problem. Each example includes both the natural language statement and the corresponding full solution, covering everything from mathematical model formulation to the final Python code, along with two new analogous problems of the same type.

You are Dr. ODELA – an expert in Ordinary Differential Equations (ODEs), with vast experience in formulating ODE models from natural language descriptions and in outputting the ODE model as Python code using either 'solve_ivp' or 'odeint' from SciPy.

Task 01:
Model the following problem using an ordinary differential equation (ODE).
Instructions:
1. Present the reasoning in a clear, step-by-step textual format.
2. Structure your response in natural language, suitable for conversion into Python code.
3. Convert all LaTeX-style math expressions into valid, readable mathematical expressions in plain text.
4. Do not compute final numerical answers; focus only on formulation and symbolic steps.

Task 02:
Convert the given ODE model (from Task 01) into a complete and executable Python code that returns the final numerical answer. Use the most appropriate solver – either solve_ivp or odeint from SciPy – based on the characteristics of the ODE problem.
The Python code must:
1. Output only valid Python code, without explanations or comments.
2. Use either solve_ivp or odeint appropriately.
3. Treat all dependent variable inputs as one-element arrays when using solve_ivp to avoid shape mismatch errors.
4. Return only the final computed value; do not include intermediate steps.
5. Round the final result to exactly 5 decimal places.
6. Ensure the code runs without requiring any modifications.

Task 03:
Generate two (2) new problems analogous to the one you just solved. Each problem must be valid, self-explanatory, and adhere to the same specifications and constraints as the original.

Figura 1. Modeling Prompt

3.3. Evaluation Criteria

To rigorously evaluate the capabilities of large language models (LLMs) in modeling open-ended ordinary differential equation (ODE) problems and generating valid Python code, we adopt a pipeline-based framework, illustrated in Figure 2.

We denote s a fixed system prompt containing role instructions and output constraints. Given a new test problem x_{test}, the language model f_θ receives the concatenation of the system prompt, the demonstration prompt, and the test problem as input, and produces the predicted output

$$\hat{y} = f_\theta\left(s, p, x_{test}\right) \tag{1}$$

where f_θ denotes the language model parameterized by θ, with θ representing the model parameters. The generated output $\hat{y}$ consists of a structured ODE formulation, the asso-

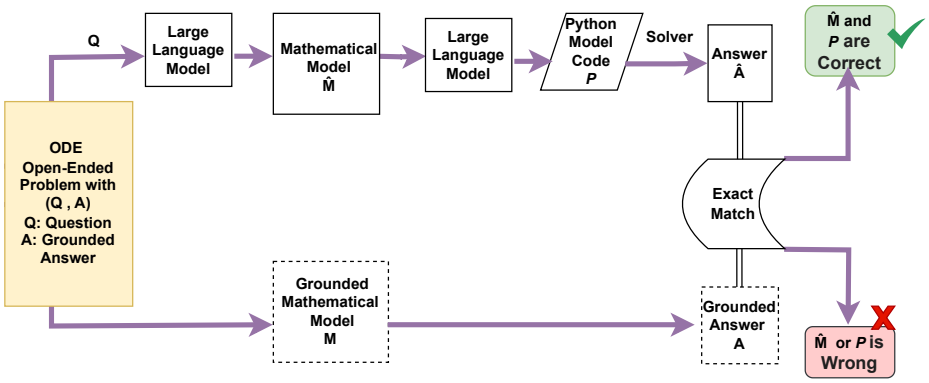

Figura 2. Pipeline for generating ODE models with valid Python code and answer verification

ciated executable Python implementation and the two analogous ODEs problems, which we denote as

$$\hat{y} = (\hat{M}, \hat{P}, \hat{G}) \tag{2}$$

where $\hat{M}$ denotes the mathematical formulation generated for the ODE model, $\hat{P}$ refers to the corresponding executable Python code, and $\hat{G}$ indicates the generated analogue problems. The Python code $\hat{P}$ is executed in a Python environment to produce a numerical result $\hat{A}$. To assess correctness, $\hat{A}$ is compared with the ground truth answer A using two tolerance criteria, as described below.

- Absolute tolerance: $ABS\_TOL = 0.1$
- Relative tolerance: $REL\_TOL = 0.05$ (5%)

The evaluation compares numerical results obtained from executing the generated Python code against the ground-truth solution. If numerical results obtained from executing the generated Python code match the ground-truth solution, then we consider that both the generated ODE model and the Python code are correct. However, if the numerical result does not align with the ground-truth solution, an additional verification step is triggered. . In this case, a dedicated prompt assesses the correctness of the generated ODE model independently of the numerical output, as shown in Figure 3. The quality of the generated analogous problems is evaluated separately. A specific instruction prompt checks the validity and consistency of these problems, as illustrated in Figure 4.

You are Dr. JUDGER – an expert in comparing texts describing ordinary differential equation (ODE) models in natural language. Compare the following two ODE models.

Inputs:
- Proposed solution: {proposed_solution}
- Model-generated solution: {model_solution}

Instructions:
1. Provide a step-by-step textual comparison of the reasoning in both models.
2. Analyze and compare the structure of the ODEs.
3. Verify numerical values, coefficients, and equations.
4. Summarize differences and similarities clearly.
5. Output a similarity score between 0 and 1, where 1 indicates identical correctness and reasoning.
6. Handle carefully texts in LaTeX form.

Output format:
1. Step-by-step comparison:
2. Structural comparison:
3. Numerical comparison:
4. Summary of differences:
5. Similarity score (0-1):

Figura 3. LLM modeling verifier Prompt

We selected 10 open-ended first-order ODE problems and implemented a systematic evaluation framework. Each LLM modeled a problem, generated valid Python code, and created two analogous problems of the same type. This process produced 100 generated problems (10 problems × 5 LLMs × 2 generated problems).

Verification was conducted in two stages. In the first stage, the predicted numerical result $\hat{A}$ was automatically compared with the ground-truth solution A.If the results matched within predefined tolerance thresholds, the solution was accepted. Otherwise, a

```
You are **Dr. JUDGER**, an expert in Ordinary Differential Equations (ODEs), with extensive experience in:
- Translating natural language problems into ODE models
- Verifying mathematical problem quality
- Evaluating whether a problem is well-posed and solvable using ODEs
Your task is to evaluate the given problem.
======================================================================
INPUT:Question:
\"\"\"{Problem}\"\"\"
======================================================================
EVALUATION CRITERIA:
1. ODE Relevance:
- Does the problem describe a process typically modeled by ODEs (e.g., growth, decay, cooling, circuits, mixing)?
2. Structural Clarity:
- Is the problem logically structured and understandable?
- Are relationships between variables clearly defined?
3. Numerical Consistency:
- Are the given values sufficient, coherent, and realistic?
- Are there contradictions or missing data?
4. Solvability:
- Can the problem be solved using standard ODE techniques?
======================================================================
SCORING RULE:
Provide a **correctness score between 0 and 1** using this guideline:
- 1.0 → Perfect ODE problem (clear, consistent, solvable)
- 0.7–0.9 → Minor issues, still solvable
- 0.4–0.6 → Moderate issues, partially unclear or incomplete
- 0.1–0.3 → Major flaws, difficult to solve
- 0.0 → Not an ODE problem or completely invalid
======================================================================
OUTPUT FORMAT (STRICT JSON):
{{
"question_type": "<e.g., exponential growth, Newton cooling, RC circuit, mixing problem, not ODE>",
"ode_relevance": "<yes/no with brief justification>",
"structural_analysis": "<brief analysis>",
"numerical_analysis": "<brief analysis>",
"solvability": "<yes/no with justification>",
"correctness_score": <float between 0 and 1>
}}
```

Figura 4. Problem Prompt verification

second stage was triggered, in which two independent LLM-based verifiers assessed the correctness of the generated ODE model against the ground truth model M. For this analysis, Gpt-5.3-codex and deepseek-r1-distill-llama-70b were utilized, both of which possess advanced linguistic capabilities and have demonstrated high performance in mathematical reasoning. The final verification score was calculated as the average of these two evaluations, thereby reducing individual model bias and improving robustness. An additional verification stage assessed the correctness of the generated analogous mathematical problems. Independent LLM-based verifiers evaluated the validity and consistency of the results, using DeepSeek-R1-Distill-Llama-70B, a model that has demonstrated strong performance in natural language tasks. In both cases, to ensure reliability, a subset of samples was validated by a human annotator. This strategy, commonly referred to in the scientific literature as "LLM as a judge," has demonstrated promising results when leveraging very large language models such as GPT-4.1 [Zheng et al. 2023, Gu et al. 2024].

The evaluation protocol assigns 1 point each for correct ODE modeling, valid Python code, and a correct numerical solution. Two additional points are awarded for correctly generating two analogous problems (1 point per valid problem), for a maximum of 5 points per original problem. Each LLM completes 50 tasks (10 problems × 5 evaluation components), yielding a total of 250 evaluated tasks.

4. Results

This paper evaluates five prominent large language models (LLMs): DeepSeek-R1, Claude-Haiku 4.5, Llama 3.3 70B Instruct, Gemini 2.5 Pro, and GPT-5.2 across a set of ten open-ended problems focused on applications of first-order ODEs. The following sections present detailed results for the selected problem types, along with an in-depth analysis of model behavior.

Newton's Law of Cooling: A body at an unknown temperature is placed in a room maintained at a constant temperature of $30°$F. After 10 minutes, the temperature of the body is $0°$F, and after 20 minutes, it is $15°$F. Determine the initial temperature of the body. In this question, all models failed to find the correct final answer, despite achieving high scores in ODE modeling. This indicates that the models correctly capture the physical law but struggle with parameter estimation and multi-step exponential reasoning. While most models generated at least one analogous problem, Llama-3.3-70B-Instruct failed to produce two valid ones, indicating weaker generalization. The performance results of the five LLMs on this problem are summarized in Table 2.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	1/1	0/1	0/1	1/2	2/5
Claude-Haiku-4.5	0.95/1	0/1	0/1	2/2	2.95/5
Llama-3.3-70B-Instruct	0.75/1	0/1	0/1	0/2	0.75/5
Gemini-2.5-Pro	1/1	0/1	0/1	1/2	2/5
GPT-5.2	0.98/1	0/1	0/1	1/2	1.98/5

Table 2. Model Performance Evaluation, Newton's Law of Cooling

Mixing Problem: A tank contains 1000 L of salt water with 15 kg of dissolved salt. Pure water enters the tank at a rate of 10 L/min. The solution is well-mixed and drains at the same rate. Determine the amount of salt remaining after 20 minutes. In this question, all models achieved perfect performance across all evaluation dimensions. This result shows that well-structured first-order linear ODEs with explicit rates and straightforward integration are reliably handled. The performance results of the five LLMs on this problem are summarized in Table 3.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	1/1	1/1	1/1	2/2	5/5
Claude-Haiku-4.5	1/1	1/1	1/1	2/2	5/5
Llama-3.3-70B-Instruct	1/1	1/1	1/1	2/2	5/5
Gemini-2.5-Pro	1/1	1/1	1/1	2/2	5/5
GPT-5.2	1/1	1/1	1/1	2/2	5/5

Table 3. Model Performance Evaluation-Mixing Problem

Electric Circuits (RL): A resistor $R = 10\,\Omega$ and an inductor $L = 2\,H$ are connected in series with a DC voltage source of 20 V. If the circuit is switched on at $t = 0$, determine the initial current $I(0)$. In this question, only Claude-Haiku-4.5 and Llama-3.3-70B-Instruct correctly completed all tasks. The remaining models showed partial success, with reasonable ODE formulations but failure in code generation and final computation. All models generated analogous problems, indicating solid conceptual understanding. The main limitation lies in translating this understanding into correct procedural execution. The performance results are summarized in Table 4.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	0.85/1	0/1	0/1	2/2	2.85/5
Claude-Haiku-4.5	1/1	1/1	1/1	2/2	5/5
Llama-3.3-70B-Instruct	1/1	1/1	1/1	2/2	5/5
Gemini-2.5-Pro	0.85/1	0/1	0/1	2/2	2.85/5
GPT-5.2	0.75/1	0/1	0/1	2/2	2.75/5

Table 4. Model Performance Evaluation-Electric Circuits (RL)

Exponential Decay (Population): A village in Angola initially has a population of 10,000 people. Due to unfavorable living conditions, the population decreases at a rate proportional to its size. After 3 years, the population is reduced to 8,000. Determine the time required for the population to reach 5,000. In this question, all models correctly generated analogous problems and demonstrated strong modeling performance, none produced correct Python code, leading to incorrect final answers. This reveals a systematic weakness in solving inverse exponential problems that require parameter estimation followed by prediction. The performance results are summarized in Table 5.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	0.85/1	0/1	0/1	2/2	2.85/5
Claude-Haiku-4.5	0.94/1	0/1	0/1	2/2	2.94/5
Llama-3.3-70B-Instruct	0.95/1	0/1	0/1	2/2	2.95/5
Gemini-2.5-Pro	0.85/1	0/1	0/1	2/2	2.85/5
GPT-5.2	0.95/1	0/1	0/1	2/2	2.95/5

Table 5. Model Performance Evaluation-Exponential Decay (Population)

Electric Circuits (RC): A $5\ \mu F$ capacitor is charged through a $10\ k\Omega$ resistor from a 12 V battery. Determine the time required for the voltage across the capacitor to reach 8 V. In this question, all models except GPT-5.2 successfully completed all the tasks. GPT-5.2 showed near-perfect modeling but failed in Python code generation, highlighting a gap between formulation and computation. All models generated valid analogous problems, reinforcing strong conceptual understanding. The performance results are summarized in Table 6.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	1/1	1/1	1/1	2/2	5/5
Claude-Haiku-4.5	1/1	1/1	1/1	2/2	5/5
Llama-3.3-70B-Instruct	1/1	1/1	1/1	2/2	5/5
Gemini-2.5-Pro	1/1	1/1	1/1	2/2	5/5
GPT-5.2	0.97/1	0/1	0/1	2/2	2.97/5

Table 6. Model Performance Evaluation-Electric Circuits (RC)

Exponential Growth (Biological System): *Escherichia coli*, a bacterium found in the human intestine, divides into two cells every 20 minutes under ideal conditions. If the initial population is 60 cells, determine the time required for the population to reach 20,000 cells. In this question, all models failed to find produce correct Python code, despite producing accurate ODE formulations and valid analogous problems. This mirrors the exponential decay case and highlights consistent difficulty with logarithmic inversion and multi-step reasoning. The performance results are summarized in Table 7.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	0.95/1	0/1	0/1	2/2	2.95/5
Claude-Haiku-4.5	0.96/1	0/1	0/1	2/2	2.94/5
Llama-3.3-70B-Instruct	0.99/1	0/1	0/1	2/2	2.95/5
Gemini-2.5-Pro	0.95/1	0/1	0/1	2/2	2.85/5
GPT-5.2	0.97/1	0/1	0/1	2/2	2.95/5

Table 7. Model Performance Evaluation-Exponential Growth (Biological System)

Falling Objects with Air Resistance: A body weighing 8 lb falls from rest toward the Earth from a great height. Air resistance is assumed to be proportional to velocity, with magnitude $2v$ (in lb), where v is in ft/s. Determine the velocity after 10 seconds. In this question, all models successfully solved the problem in terms of modeling, code, and final answer. However, DeepSeek-R1 failed to generate valid analogous problems, indicating a limitation in creative generalization. The performance results are summarized in Table 8.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	1/1	1/1	1/1	0/2	5/5
Claude-Haiku-4.5	1/1	1/1	1/1	2/2	5/5
Llama-3.3-70B-Instruct	1/1	1/1	1/1	2/2	5/5
Gemini-2.5-Pro	1/1	1/1	1/1	2/2	5/5
GPT-5.2	1/1	1/1	1/1	2/2	5/5

Table 8. Model Performance Evaluation-Falling Objects with Air Resistance

Economic Decay (Inflation): In 2024, an investment of 15,000,000 kwanzas is made. Due to inflation in Angola, the value decays continuously at a rate of 10% per year. Determine the total value after 5 years. In this question, all models achieved perfect performance across all dimensions. The problem structure follows a standard exponential decay formulation, which is consistently well handled by all models. The performance results are summarized in Table 9.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	1/1	1/1	1/1	2/2	5/5
Claude-Haiku-4.5	1/1	1/1	1/1	2/2	5/5
Llama-3.3-70B-Instruct	1/1	1/1	1/1	2/2	5/5
Gemini-2.5-Pro	1/1	1/1	1/1	2/2	5/5
GPT-5.2	1/1	1/1	1/1	2/2	5/5

Table 9. Model Performance Evaluation-Economic Decay (Inflation)

Chemical Kinetics (Radioactive Decay): A radioactive substance decays at a rate proportional to the amount present. Initially, 50 mg is present, and after 2 hours, 10% of the original mass has decayed. Determine the remaining mass after 4 hours. In this question, only Llama-3.3-70B-Instruct and Gemini-2.5-Pro successfully completed all steps. The remaining models showed strong modeling and generalization but failed in Python code generation. This reinforces the observed gap between correct formulation and accurate computation. The performance results are summarized in Table 10.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	0.95/1	0/1	0/1	2/2	2.95/5
Claude-Haiku-4.5	0.92/1	0/1	0/1	2/2	2.92/5
Llama-3.3-70B-Instruct	1/1	1/1	1/1	2/2	5/5
Gemini-2.5-Pro	1/1	1/1	1/1	2/2	5/5
GPT-5.2	0.95/1	0/1	0/1	2/2	2.95/5

Table 10. Model Performance Evaluation-Chemical Kinetics (Radioactive Decay)

Economic Growth (Compound Interest): A depositor invests \$10,000 in a certificate of deposit at YETU Bank of Angola with an annual interest rate of 7.5%, compounded continuously. Determine the amount in the account after 8 years. In this question, all models correctly solved the problem, while only Llama-3.3-70B-Instruct failed to generate analogous problems. This suggests that procedural and computational capabilities are strong, but variability remains in creative generalization tasks. The performance results are summarized in Table 11.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	1/1	1/1	1/1	2/2	5/5
Claude-Haiku-4.5	1/1	1/1	1/1	2/2	5/5
Llama-3.3-70B-Instruct	1/1	1/1	1/1	0/2	3/5
Gemini-2.5-Pro	1/1	1/1	1/1	2/2	5/5
GPT-5.2	1/1	1/1	1/1	2/2	5/5

Table 11. Model Performance Evaluation-Economic Growth (Compound Interest)

5. Discussion

Near-perfect performance on Mixing Problems and Chemical Kinetics tasks indicates that current LLMs effectively capture procedural ODE modeling. In contrast, the weakest results occur for Newton's Law of Cooling, revealing difficulties in translating natural language descriptions into accurate ODE models. This challenge requires contextual interpretation and reasoning rather than simple procedural computation. The findings highlight a persistent gap between procedural competence and genuine mathematical understanding in AI systems.

The overall results presented in Table 12 reveal a consistent pattern across models. All LLMs demonstrate strong performance in ODE modeling, with scores exceeding 96%, which indicates a robust understanding of the underlying mathematical structures. However, performance declines in Python code generation, exposing a gap between problem formulation and computational execution.

Claude-Haiku-4.5 achieved the highest overall score, driven by strong modeling and perfect analogous problem generation. Llama-3.3-70B-Instruct excels in code generation and final-answer accuracy, though its generalization is weaker. Gemini-2.5-Pro shows balanced and robust performance across all dimensions, while DeepSeek-R1 remains consistent but limited in execution. GPT-5.2 records the lowest overall score, mainly due to weaker performance in coding and final answer derivation.

Model	ODE modeling	Python code	Final Answer	G. Problems	G. Score
DeepSeek-R1	0.96	0.50	0.50	0.85	0.77
Claude-Haiku-4.5	**0.98**	0.60	0.60	**1.00**	**0.84**
Llama-3.3-70B-Instruct	0.97	**0.70**	**0.70**	0.80	0.79
Gemini-2.5-Pro	0.97	0.60	0.60	0.95	0.81
GPT-5.2	0.96	0.40	0.40	0.95	0.73

Table 12. Model Overall Performance

6. Conclusion

The results indicate that current large language models (LLMs) exhibit decent performance in ordinary differential equation (ODE) modeling, achieving high performance and effectively capturing underlying mathematical structures. However, these models encounter challenges during the transition from problem formulation to execution, often failing in Python code generation and consequently leading to incorrect final answers, particularly in parameter estimation and inverse exponential tasks. While structured problems are addressed reliably, open-ended and multi-step tasks expose substantial weaknesses in maintaining coherent reasoning. Among the evaluated models, Claude-Haiku-4.5 demonstrates the highest overall performance. In contrast, Llama-3.3-70B-Instruct performs well in ordinary differential equation (ODE) modeling and Python code generation but exhibits limitations in generating analogous problems. These findings underscore the necessity for evaluation frameworks that assess entire reasoning pipelines. Furthermore, integrating large language models (LLMs) with symbolic and numerical tools may enhance reliability. Current models lack the integrated conceptual understanding that typifies human reasoning. Future research should investigate self-evaluation capabilities and methods such as teacher-student learning, Low-Rank Adaptation (LoRA), and quantization to improve both performance and efficiency.

7. Supplementary Information

Supplementary materials accompanying this article include an ODE benchmark dataset comprising ten open-ended problems and their corresponding solutions, the source code files employed in the experimental evaluation, and representative output samples illustrating the performance spectrum of LLMs. All materials are available at: https://github.com/Domingos-hub/Can-LLMs-Solve-Ordinary-Differential-Equations-A-Comparative-Analysis

8. Acknowledgment

This study was funded by the São Paulo Research Foundation (FAPESP, Grant #2019/07665-4 and #23/10100-4), the National Council for Scientific and Technological Development (CNPq, Grant #309575/2021-4 and #307184/2025-0), CAPES, the University of São Paulo Support Foundation, project number 4148 and the Polytechnic Institute of the University of José Eduardo dos Santos (IP-UJES). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brasil, Finance Code 001.The project was supported by the Ministry of Science, Technology and Innovation, with resources of Law N. 8,248, of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex and published as Residence in TIC 13, DOU 01245.010222/2022-44. We express our gratitude to the anonymous reviewers for reviewing the paper and for providing helpful feedback.

Referências

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Gao, L. et al. (2023). PAL: Program-Aided Language Models. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Glazer, E. et al. (2024). Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*.

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., et al. (2024). A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. In *Proceedings of the Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS 2021), Datasets and Benchmarks Track*.

Huang, X., Shen, Q., Hu, Y., Gao, A., and Wang, B. (2025). Llms for mathematical modeling: Towards bridging the gap between natural and mathematical languages. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2678–2710. Association for Computational Linguistics, Albuquerque, New Mexico.

Lee, G.-G., Latif, E., Wu, X., Liu, N., and Zhai, X. (2024). Applying large language models and chain-of-thought for automatic scoring. *Computers and Education: Artificial Intelligence*, 6:100213.

Nikolaiev, A., Stathopoulos, Y., and Teufel, S. (2024). Can language models rival mathematics students? evaluating mathematical reasoning through textual manipulation and human experiments. *arXiv preprint arXiv:2412.11908*.

Samanjata, D., Santin, R. V., Marcacini, R. M., and Rezende, S. O. (2025). Benchmarking large language models for solving ordinary differential equations. In *Proceedings of the International Conference on Smart Computing and Machine Intelligence (ISCMI)*.

Sun, Y., Hu, S., Zhou, G., Zheng, K., Hajishirzi, H., Dziri, N., and Song, D. (2025). Omega: Can llms reason outside the box in math? evaluating exploratory, compositional, and transformative generalization. *Advances in Neural Information Processing Systems*, 38.

Yue, J. (2025). Benchmarking llms on advanced mathematical reasoning. Technical report, University of California, Berkeley.

Zhang, L., Wang, Y., and Lee, J. (2023). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Zhao, Z. et al. (2024). Stepwise self-consistent mathematical reasoning with large language models. *arXiv preprint arXiv:2402.17786*.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.