

Article

Evaluation of a User Interface Extension Integrating an LLM-Based AI Assistant into an Interactive Visual Equation Editor for Solving High School Mathematics Problems

Agnieszka Bier  and Zdzisław Sroczyński * 

Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland

* Correspondence: zdzislaw.sroczyński@polsl.pl

Abstract

This paper presents an evaluation of AI-assisted human–computer interaction for mathematical problem solving within a multimodal mathematical editing environment. The proposed architecture integrates a visual equation editor, voice-based interaction, REST communication services, and externally hosted generative AI large language models (LLMs) to support the creation, interpretation, and solution of mathematical expressions. The study was conducted using the Equation Wizard environment, which provides multimodal mathematical content editing based on both proprietary and standard formula representations, including MathML and $\text{\LaTeX}$. A conversational AI interaction layer enables users to communicate with selected LLMs using natural language voice commands. The main objective of the study was to determine whether contemporary LLM-based services can reliably support mathematical problem solving within an AI-enhanced equation editing environment. To address this objective, seven contemporary GenAI LLMs were evaluated using a benchmark consisting of representative high school mathematics problems covering algebra, limits, trigonometry, logarithms, inequalities, and parameterized expressions. The evaluation focused on mathematical correctness, output interpretability and visualization quality, response latency, and compliance with mathematical encoding formats within the complete interaction workflow. The study also compares representative model families with respect to correctness, syntactic compliance, and responsiveness within the complete interaction workflow. Unlike conventional LLM benchmarks that assess models in isolation, this work evaluates end-to-end AI-assisted mathematical interaction involving the editor, communication infrastructure, and language models. The study demonstrates a practical approach to assessing the usefulness of multimodal AI-enhanced mathematical problem solving in realistic usage scenarios. The results show that, within the evaluated interaction workflow, current LLMs achieve high levels of mathematical problem-solving performance, although substantial differences were observed in response latency and output-format compliance. An important finding is the discrepancy between the models' strong semantic understanding of custom-encoded mathematical input and their weaker ability to generate syntactically compliant encoded outputs, highlighting a key challenge in the integration of LLMs with structured mathematical software environments.



Academic Editors: Radoslava Kraveva and Velin Kravev

Received: 21 May 2026

Revised: 5 July 2026

Accepted: 9 July 2026

Published: 16 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)

Attribution (CC BY) license.

Keywords: multimodal formula editing; natural language processing; voice user interface; large language models (LLMs); AI-assisted mathematical problem solving; performance of HCI in systems processing mathematical content

1. Introduction

The rapid expansion of generative AI and large language models (LLMs) observed in recent years has already changed the way humans interact with computers, search for knowledge, or seek solutions to problems of any kind. Among the variety of areas and disciplines, mathematical notation and mathematics problems have always been a challenge to automated systems such as educational tools or search engines [1,2], and in the era of LLMs, it is no different [3,4]. Solving mathematics problems and mathematical formal reasoning remains one of the most challenging to handle with LLMs. Nevertheless, the improvement observed in this area over the years has led us to a point where utilizing LLMs as solvers or educators in mathematics is a fact. Given the expansion, wide implementation, and high level of integration with classical software tools, the control over correctness and reliability of answers provided by the models becomes critical. Whenever it is a student searching assistance with their studies, or an engineer seeking for problem formalization and solutions, the LLM-based assistants are trusted to deliver correct and relevant content. Unlike knowledge-seeking outputs, the relevance and correctness of generated mathematical content often cannot be verified by users on their own or provided links. Therefore, the more significant it becomes for the experts to monitor, benchmark, and report on the quality of LLM-generated answers to stated mathematics problems.

In this study, we investigate the performance of selected LLMs on tasks related to solving mathematics problems of difficulty comparable to those appearing in the Polish high school final examination (“matura”). The exam includes both basic and advanced levels, and it provides a useful benchmark for evaluating mathematical reasoning skills in automated systems. Our goal is to evaluate the practical usefulness of AI-assisted mathematical problem solving within the Equation Wizard environment, while comparing the behavior of several representative LLM services integrated into this architecture with two main criteria: (1) correctness and formal precision of the produced solutions in terms of required data format, and (2) response time.

To support this analysis, we use a dedicated Windows application Equation Wizard [5]. The software provides a visual interface for displaying and editing mathematical expressions with menus, palettes and keyboard shortcuts. Additionally, users can load formulas in standard $\text{\LaTeX}$, MathML, or in a simplified internal format called EQED. EQED follows the structure of $\text{\LaTeX}$ commands but restricts all arguments to curly brackets. This constraint reduces ambiguity and prevents errors during parsing.

The Equation Wizard includes a built-in module for interacting with LLMs via a REST API [6]. When the user enters a prompt, the program sends both the currently selected formula and the prompt text to the chosen model. The additional context with EQED specification and set of rules is transferred as well. The model is supposed to respond with a new expression altered according to the user’s prompt, typically encoded in EQED, which the editor then renders. The system also allows users to save full conversation logs and measure the time required to generate each response. These features enable controlled experiments under repeatable conditions. The capability of the tested models to properly read the problem statement and provide its solution in the custom EQED format is an important aspect of this work, which addresses the problem of a proper formatting and interpretation of the results in LLM’s reasoning.

The main objective of the study is to evaluate the practical usefulness of AI-assisted mathematical problem solving within the Equation Wizard environment and to determine whether contemporary LLM services can reliably support such interaction workflows. In addition, the study seeks to identify which model families provide the most favorable balance between mathematical correctness, syntactic compliance, and perceived responsiveness when solving representative “matura”-level mathematical problems. The results

can support further development of the Equation Wizard environment and its integration into educational and automated assessment scenarios.

Unlike many studies focused on benchmarking the intrinsic mathematical capabilities of LLMs, the objective of the present work is to evaluate an AI-enhanced human–computer interaction architecture integrated into an existing multimodal mathematical editor. The investigated system combines a visual equation editor, a voice interface, a prompt-construction layer, REST-based communication services, and externally hosted GenAI models. Consequently, the study evaluates the practical usefulness of complete AI-assisted workflows rather than isolated model performance.

The evaluation was conducted from the perspective of a typical “bring your own key” user who accesses commercial GenAI services through provider-hosted infrastructure. Therefore, both the correctness of generated solutions and the perceived responsiveness of the complete interaction process contribute to the overall assessment.

The study addresses the following research questions:

- RQ1: Can contemporary LLM-based services be effectively integrated into a multimodal mathematical editing environment to support solving secondary school mathematics problems?
- RQ2: To what extent do different LLM services provide mathematically correct and syntactically compliant outputs when operating within a constrained mathematical encoding scheme (EQED)?
- RQ3: Which model families appear most suitable for practical deployment in AI-assisted mathematical editing workflows when correctness, syntactic compliance, and responsiveness are considered jointly?

Based on previous work on multimodal mathematical editing, we hypothesize that advanced frontier models will demonstrate strong semantic understanding of mathematical expressions, but their ability to generate outputs fully compliant with a custom formal notation will remain less reliable, in particular the following hypotheses are investigated:

Hypothesis 1. *At least some contemporary frontier LLM services are capable of solving representative high school mathematical tasks with a level of correctness sufficient for practical integration into mathematical editing environments.*

Hypothesis 2. *Mathematical correctness and syntactic compliance constitute partially independent dimensions of performance, and mathematically correct answers do not necessarily guarantee successful integration with downstream software tools.*

Hypothesis 3. *Models offering the best practical user experience are not necessarily those with the shortest response times, because correctness and reliability dominate perceived usefulness within AI-assisted mathematical problem-solving environments.*

2. Related Work

The problem of efficient and consistent processing of mathematical notation in automated systems emerges in several areas of application. First, mathematical notation processors enable standardized formula visualization. Secondly, it is critical in tasks related to comparison between documents with mathematical content such as phrase search or plagiarism detection. Last but not least, efficient math-processing engines enable math verbalization in voice interfaces, both for transcribing spoken formulas as well as synthesizing math content as spoken output. As such, it has been widely addressed in the bibliography [1,7–19].

With LLMs, verbalization comes naturally by their design. However, mathematical content read or transcribed in the natural language is hard to comprehend by automated systems. Hence, the LLM's capabilities of "understanding" the symbolic language, evaluating math problems or even mathematical reasoning or theorem proving, have drawn the attention of many researchers [20–27] and are a valid research question on its own. Nevertheless, our work goes one step further—we are interested not only in the evaluation of the mathematical skills, but also in consistency and reliability of generated output in a structured way that is relevant for integration with other software components.

Our previous research on human–computer interaction for mathematical expression editing provides the foundation for the present study. The earlier work focused on enabling LLM-based systems to support voice-driven composition and edition of mathematical formulas. In the first stage of this research, we explored how interactive voice commands can extend the functionality of a math formula editor and improve user experience when entering complex expressions [28,29].

More recent research addressed the integration of a GenAI assistant directly into Equation Wizard, providing a natural language interface in Polish language [6]. We examined the ability of LLMs to verbalize mathematical expressions from the perspective of end users. This case study assessed how well different models interpret natural language descriptions of formulas and how reliably they produce correct symbolic output. The findings highlighted both the strengths and common failure modes of LLMs when dealing with ambiguous or multi-step verbal instructions in Polish. This work extended the earlier concepts by embedding the LLM within a full editing environment, allowing continuous interaction, command-based transformations, and real-time validation. Together, these studies show that LLMs are increasingly capable of supporting the input and modification of mathematical notation.

However, prior research has focused primarily on interpreting, verbalizing, or editing formulas rather than solving mathematical problems expressed by those formulas. The present study addresses this gap. We shift attention from formula processing to problem solving and evaluate the capability of various LLMs to handle tasks at the difficulty level of a Polish high school final exam. This direction follows naturally from earlier observations and opens new possibilities for applying AI systems in mathematical education and assessment.

3. Experiment Setup and Methodology

3.1. Models Selected for the Test

The study evaluates the ability of modern large language models to solve math problems at the level of a Polish high school final exam. Based on prior experience with symbolic editing and on the requirements of multi-step reasoning, we selected seven models for detailed testing. Processing cost was intentionally excluded from the selection criteria. Instead, we focused on mathematical competence, inference reliability, and response latency in interactive workflows. In addition to the aforementioned aspects, the model's ability to parse input and generate output in accordance with the custom mathematical encoding rules was evaluated as a necessary intermediate step toward producing human-readable answers and providing comfortable human–computer interaction.

The selected models are:

1. gpt-5: The strongest general-purpose model available at the moment of experiments in the GPT family and a natural baseline for high-accuracy reasoning. GPT-5 demonstrates excellent performance on algebra, geometry, and multi-step tasks and reliably generates well-formatted L^AT_EX.

2. gpt-4o: This model is optimized for structured reasoning, including symbolic manipulation and equation solving. It offers high stability, consistent formatting, and very fast inference compared to previous generations.
3. gpt-4o-mini: A reduced-size variant designed for low latency. Although less capable than gpt-4o, it preserves much of its mathematical reasoning ability and is especially valuable in real-time editing scenarios where response time is critical.
4. claude-opus-4: Claude Opus provides strong analytical reasoning and coherent step-by-step explanations. It performs well on exam-style problems that require structured justification. It also produces clean, readable L^AT_EX output, which is essential for formula visualization.
5. grok-4-latest: Grok-4 offers competitive reasoning speed and improved mathematical reliability compared to earlier versions. Its fast inference makes it suitable for experiments requiring large batches of queries or interactive command sequences.
6. mistral-large-latest: Mistral Large combines solid mathematical competence with efficient inference. It tends to produce concise outputs and handles symbolic transformations well. Its inclusion broadens the diversity of model architectures examined in this study.
7. gemini-flash-latest: Gemini Flash is optimized for fast inference while maintaining competitive reasoning performance. It demonstrates strong symbolic manipulation capabilities and produces structurally consistent formula output. Its inclusion allows for comparison between latency-oriented architectures and more computation-heavy reasoning models, particularly in interactive mathematical editing environments.

Together, these seven models represent multiple model families—GPT, Claude, Grok, Gemini and Mistral—which allows for a broader assessment of architectural differences and reasoning strategies.

3.2. Alternative Models Considered

Several available models were excluded from primary testing for the following reasons:

- gpt-5-mini and gpt-5-nano: These compact versions provide very fast responses but lack the reasoning depth needed for nontrivial exam problems. They remain useful for latency benchmarks but are not suitable as core evaluation targets.
- mistral-small-latest: Although efficient, it performs inconsistently on tasks requiring multi-step symbolic reasoning.
- claude-sonnet variants (3.7 and 3.5 families): These models display strong general reasoning, but Claude Opus consistently achieves higher accuracy in mathematical tasks, making it the more appropriate representative of the Claude family.
- o-series models (including o3): These models offer good general reasoning, but the 4o family provides superior performance on symbolic tasks and more stable L^AT_EX formatting. Therefore, o-series models were not prioritized.

The selection process emphasizes consistency, mathematical reliability, and suitability for exam-level tasks, rather than broad coverage of all available architectures.

3.3. Future Extensions

To enable a more comprehensive benchmark, the study may be extended in the future with additional models that are known for strong performance on mathematical tasks or low-latency reasoning:

- DeepSeek-R1 and DeepSeek-Math series: These models show competitive results on math-oriented benchmarks and have been optimized for symbolic reasoning and structured inference.

- Qwen2.5 Large and Qwen2.5-Math: Qwen models have demonstrated strong performance on multilingual math datasets and could provide valuable comparisons with Western LLM architectures.
- Gemini 2.0 Pro/Flash: If integration becomes available, these models could be evaluated for their efficiency and reasoning performance, especially given their strong results on broader reasoning benchmarks.
- Open-source math-specialized models: Models such as Llemma or MathCoder may also be useful for targeted experiments in symbolic transformation and step-by-step justification.
- Local models: Models available and run at the local machine can be evaluated to decide if they are suitable for tasks solving despite their reduced size and other parameters.

Expanding the model pool in future research will allow for a deeper analysis of the relationship between model size, architecture, and symbolic reasoning performance. Moreover, the implementation of local models can open the possibility of developing an independent educational application that would run regardless of the network connectivity or the financial resources needed for publicly available solutions.

3.4. Software Environment and Configuration

All experiments were conducted using the Equation Wizard, a Windows application capable of rendering mathematical expressions and interacting with LLMs through a configurable REST interface. Model configurations are stored in a JSON configuration file, encrypted and loaded either locally or remotely using a username- and password-based access control. This mechanism safeguards access to paid models and ensures that only authorized participants can perform experiments.

The editor supports input and output in MathML, $\LaTeX$ and EQED, a simplified internal format that mirrors $\LaTeX$ syntax but restricts all arguments to curly braces. This simplification reduces parsing ambiguity and facilitates automated processing. The interaction between a user and graphical interface of the editor is possible with the use of classic menus, palletes, keyboard shortcuts, copy–paste commands and selection with the mouse pointer. Moreover, there is a mobile assistant application, which enables actual voice-controlled interaction, where the description of the formula along with editing commands are transmitted in the form of natural language. The mobile assistant receives the visualization of the resulting formula in the buffer of the editor, so it can be presented or spelled in the handheld device. This approach opens the variety of possible applications, including educational ones and supporting people with disabilities. The method of entering and manipulating the mathematical notation is transparent and can be changed even in the middle of the process, creating a fully multimodal environment. The example of the Equation Wizard editor connected with the mobile assistant is shown in Figure 1.

Communication between the Equation Wizard and external LLM services is performed through JSON messages exchanged via the REST interface. Each request contains three principal components. The first component is a system-level instruction defining the EQED notation, including its $\LaTeX$ -based macro specification and the rules governing mathematical expression encoding. The second component contains the mathematical expression currently stored in the editor buffer, represented in EQED format and inserted into the prompt dynamically. The third component is a user instruction describing the requested operation, such as solving an equation, computing a limit, or evaluating an expression.

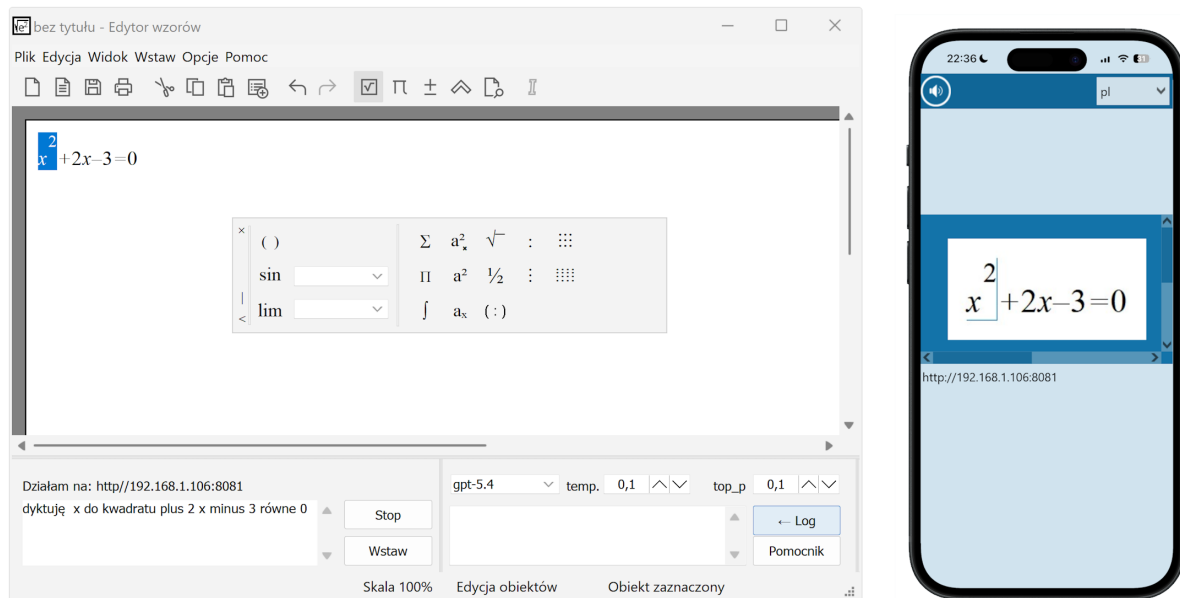


Figure 1. Graphic user interface of the Equation Wizard environment supporting the editing of mathematical expressions using a voice interface (main application on the left, mobile assistant application on the right).

The prompt is designed to enforce a strict output format suitable for automated processing. In addition to the EQED macro definitions, it specifies encoding rules for numbers, variables, operators, and structural mathematical constructs. The model is instructed to interpret the source expression according to standard mathematical conventions, perform the requested transformation, and return exclusively the resulting expression encoded in EQED. Any explanatory text, comments, formatting markers, or repeated macro definitions are explicitly prohibited. This approach enables direct integration of model responses with the rendering and editing mechanisms of the Equation Wizard while reducing ambiguity in the interpretation of generated mathematical content.

All experiments were performed using the official REST APIs provided by the respective model vendors. GPT models were accessed through the OpenAI API, Claude through the Anthropic API, Mistral through the Mistral AI API, Grok through the xAI API, and Gemini through the Google Generative Language API using its OpenAI-compatible interface. The experiments were conducted during several experimental sessions carried out over several weeks at the turn of 2025 and 2026, using the model versions publicly available through the official APIs of their respective providers. The evaluated services were identified by their publicly available model names; more detailed internal revision identifiers were not available in a consistent form across all evaluated providers and therefore could not be reported uniformly. Unless required by a specific provider, the default model parameters were retained, with the exception of the temperature parameter, which was set to the lowest supported value in order to reduce response variability and improve determinism. The prompt structure and system-level instructions used throughout the experiments are described in this section and were applied consistently across all evaluated models.

The main software components and the flow of information exchanged during AI-assisted operations are presented in Figure 2. Particular attention is given to the AI integration layer, which is responsible for prompt construction, communication with external LLM services, and interpretation of generated responses.

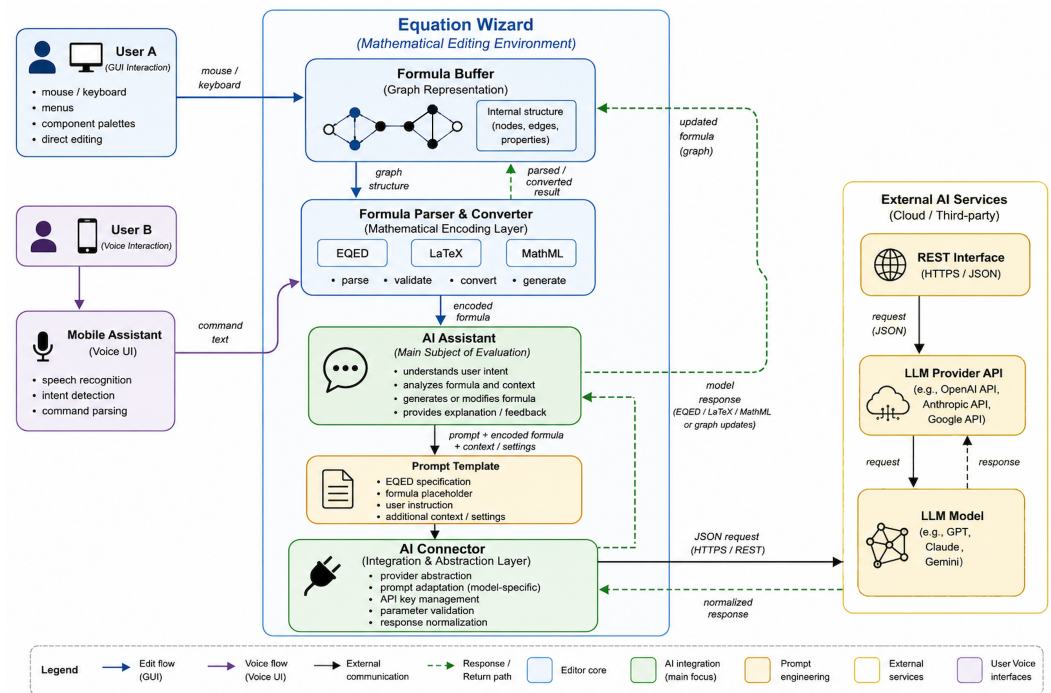


Figure 2. Architecture of the AI-enhanced Equation Wizard environment. The system combines GUI-based and voice-based interaction with a visual equation editor and an AI integration layer consisting of the AI assistant, Prompt Template, and AI Connector modules. The integration layer communicates with external large language models through provider-independent REST interfaces and enables automated processing of mathematical expressions encoded in EQED, LaTeX , and MathML formats.

3.5. Repeated Executions and Response-Time Evaluation

Each experiment was executed multiple times (at least five repetitions per model and task) under comparable network and infrastructure conditions. The repeated executions produced nearly identical mathematical outputs and only minor latency fluctuations, typically below one second.

Because the objective of the study is not infrastructure benchmarking but evaluation of practical user experience within an AI-enhanced mathematical editor, exact latency measurements were not treated as primary quantitative variables. Instead, response times were grouped into qualitative responsiveness categories reflecting perceived user experience.

Occasional network failures, temporary service unavailability, or unusually long delays were observed during testing. Such incidents were not reproducible and disappeared when the request was repeated after a short interval. Since the experiments intentionally reflect the perspective of a real end user accessing externally hosted GenAI services, these events were treated as part of the overall interaction environment rather than as model-specific characteristics.

3.6. Evaluation Criteria

The evaluated models were assessed using two complementary criteria: correctness of the generated solution and perceived responsiveness of the interaction. Correctness was classified using a four-level scale ranging from fully correct solutions ☆☆☆ to mathematically incorrect results ☹.

The assigned rating represents a joint qualitative assessment of mathematical correctness and syntactic compliance with the EQED representation. Mathematical correctness was treated as the primary criterion, while syntactic compliance determined whether a mathematically correct response could be directly used within the Equation Wizard environment. The rating was performed in the following scale:

- Mathematically correct solutions represented in a form directly usable within the Equation Wizard environment ☆☆☆;
- Mathematically correct results containing only minor formatting or presentation deficiencies that did not prevent correct interpretation ☆☆;
- Correct mathematical idea or intermediate result but incorrect or syntactically unusable final representation ☆;
- Mathematically incorrect result or output that could not be interpreted as valid solutions ☹.

Because the observed response times depend not only on the model itself but also on network conditions, provider-side infrastructure, and temporary system load, exact latency values were not treated as a primary evaluation metric. Instead, response times were grouped into four qualitative responsiveness categories reflecting the user experience during interaction with the system:

- Instant (up to 5 s) ⚡;
- Fast (6–15 s) ⚡⚡;
- Moderate (16–30 s) ⚡⚡⚡;
- Slow (more than 30 s) ⚡⚡⚡⚡;
- no response received or processing failed [!].

This categorization reduces the impact of minor timing fluctuations while preserving the practical distinction between models that respond immediately and those requiring noticeable waiting time.

4. Experiments

In this section, we present a series of experiments designed to evaluate how selected large language models handle mathematical problems taken from a Polish high school final examination (“matura”). Based on the model selection criteria discussed earlier, we test seven LLMs using the same set of tasks. Each experiment begins with the Equation Wizard initialized with an expression representing the problem, followed by a prompt instructing the model to perform a specific mathematical operation and provide the output in the expected internal format—EQED. The editor logs all information on LLM interactions, including raw prompts, model outputs, processing times, and editing commands. These logs provide all necessary data for comparing accuracy, performance, and responsiveness across different models. The user interface of Equation Wizard with LLM interactive dialog during experiments is shown in Figure 3.

For clarity and conciseness, we provide a full, detailed description only for the first model tested in each task, including the complete log output, timestamps, and analysis of the returned EQED structure. For the remaining models, we summarize the most relevant observations, highlight differences in execution time, output format, and correctness, and report any recurring issues identified during evaluation. This approach keeps the narrative focused while preserving all necessary comparative insights.

4.1. Experimental Reproducibility and Response Times

All response times reported in the following experiments represent average values obtained from multiple executions of the same task. Since the measured latency depends not only on the language model itself but also on network conditions, API communication overhead, provider-side infrastructure, and temporary service load, exact timing values should be interpreted as indicative rather than absolute measurements. For this reason, the comparative analysis presented later in the paper primarily relies on qualitative responsiveness categories rather than precise latency measurements.

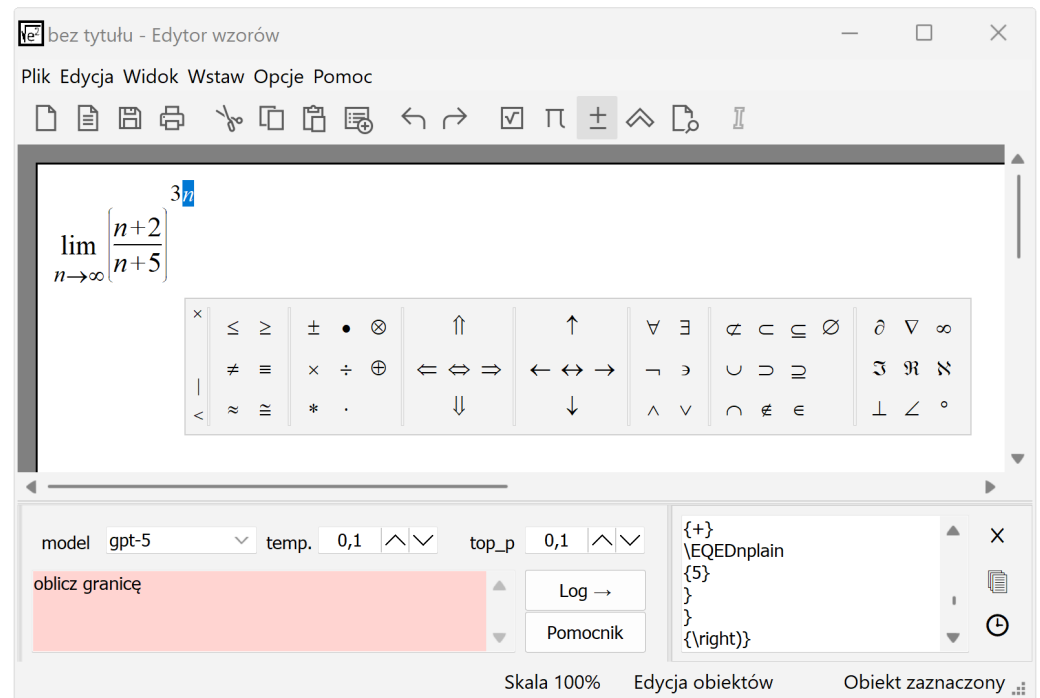


Figure 3. Equation Wizard during experiments. There is main editor with the visualization of the equation, menu and one of the palettes of ready-to-insert mathematical symbols in the upper part, as well as AI assistant dialog in the lower part with user prompt marked with light red color.

An additional observation concerns the reproducibility of generated outputs. Most evaluated models produced either consistently correct or consistently incorrect results across repeated executions. However, the gpt-4o and gpt-4o-mini models exhibited noticeably higher output variability. In several experiments, repeated executions of the same task resulted in different incorrect answers, typically differing by individual symbols, numerical values, or minor algebraic transformations. Consequently, the incorrect outputs reported for these models should be treated as representative examples of the observed behavior rather than as unique deterministic responses. No comparable variability was observed for the remaining evaluated models.

Each experiment was repeated at least five times for every model. The repeated executions were intended to verify the stability of the observed behavior rather than to perform a statistical comparison between models. Individual runs were assessed using the same qualitative criteria adopted throughout this study, and the reported ratings reflect the overall behavior consistently observed across the repeated executions. Occasional variations, when observed, were limited to minor differences that did not alter the assigned qualitative rating. With the exception of GPT-4o and GPT-4o-mini, which occasionally produced minor variations in incorrect outputs, the tested models demonstrated highly repeatable behavior.

4.2. Experiment 1: Solve Quadratic Equation

In the first experiment, we evaluated the behavior of the GPT-5 model when solving a basic quadratic equation inserted into the Equation Wizard buffer. The editor was initialized with the following $\text{\LaTeX}$ expression:

$$x^2 + 2x - 3 = 0.$$

After loading the expression, the user issued the following instruction in Polish (“solve the equation” in English):

```
rozwiąż równanie
```

The editor transmitted both the buffer content and the prompt to the GPT-5 model through the REST interface. The full interaction was recorded in the system log. The relevant piece of this log is shown below:

```
Model: gpt-5
START: 2025-11-28 17:09:26
temp. i top_p ignorowane!
\EQEDhi
{\EQEDnplain
{2}
}
{\EQEDplain
{x}
}
\EQEDplain
{+}
\EQEDnplain
{2}
\EQEDplain
{x}
\EQEDplain
{-}
\EQEDnplain
{3}
\EQEDplain
{=}
\EQEDnplain
{0}

rozwiąż równanie
\EQEDplain{x}\in\{\EQEDnplain{1}\EQEDplain{,}\EQEDplain{-}\EQEDnplain{3}\}
STOP: 2025-11-28 17:10:07
```

The EQED-formatted block at the beginning represents the expression stored in the buffer. This is followed by the user prompt, and finally by the result generated by the model. Generally, EQED commands are identifiable by the prefix EQED, and each tokenized element of the expression is enclosed in curly-bracket arguments. The total processing time for this request was 37 s. The model returned a solution, identifying the set of roots as:

$$x \in \{1, -3\}.$$

In this experiment, the GPT-5 model produced the mathematically correct solution to the quadratic equation. However, the returned representation mixed EQED commands with raw $\text{L}^{\text{T}}\text{E}^{\text{X}}$ parentheses. Although EQED tokens were used for numbers and operators, the model inserted set brackets using standard $\{ \}$ notation rather than the dedicated $\backslash\text{EQEDbrackets}$ command.

This inconsistency led to a rendering issue in the formula editor. Because the wizard expects all structural elements to follow EQED conventions, the raw $\text{\LaTeX}$ parentheses were not visualized properly. As a result, the displayed form of the solution was unclear, even though the user could infer the intended meaning from context.

The above distinction is important for assessing model performance. The output of GPT-5 in this experiment should be classified as *mathematically correct* but *syntactically insufficient*. The answer cannot be processed reliably in automated workflows, since it lacks a fully valid EQED structure.

The inconsistency observed here highlights a broader challenge in evaluating LLM behavior in structured mathematical tasks: a correct numerical or symbolic result does not guarantee a correct formal representation compatible with downstream tools. Subsequent experiments will track both correctness dimensions—mathematical and syntactic—to enable a more detailed comparison across models.

This confirms that GPT-5 was able to interpret the equation and carry out the required algebraic steps, but it did not produce a fully valid EQED representation of the final answer.

The result reported for gpt-4o was $x = \frac{3}{2}$. This value is incorrect. This discrepancy should be recorded in the experiment log as a *mathematical error*. The processing took just 2 s, but the result is useless.

The model claude-opus-4 produced not only a correct solution to the equation

$$x^2 + 2x - 3 = 0,$$

but also presented the full computation in a compact and readable single-line derivation in 7 s only:

$$x = \frac{-2 \pm \sqrt{4 + 12}}{2} = \frac{-2 \pm 4}{2} = 1 \vee x = -3.$$

This response is mathematically valid and demonstrates a clear and well-structured reasoning path. The transformation from the quadratic formula to the final numeric roots is correct, and each step follows standard algebraic conventions. In terms of mathematical clarity, this model provided one of the most coherent outputs in this task.

The model grok-4 required a relatively long processing time (66 s) to return its solution. The final answer was expressed as:

$$x = -1 \pm 2,$$

which is mathematically equivalent to the correct roots $x = 1$ and $x = -3$.

Although the expression is formally valid, its presentation is less transparent than the conventional explicit listing of the two solutions.

The model mistral-large produced a mathematically correct solution to the equation, but the syntax of the output was almost entirely unreadable. Only a careful user could extract the actual roots from the generated output.

Although the model returned the result quickly (in approximately 5 s), the poor readability renders the output unsuitable for direct use within the investigated mathematical editing workflow. This example highlights that a correct numeric answer alone does not guarantee a practically usable solution within the investigated environment.

The model Gemini gave the correct answer after rather long processing time, i.e., in 36 s.

4.3. Experiment 2: Calculate Limit of a Sequence

The second task requests the model to compute the limit of a sequence. The Equation Wizard buffer is initialized with the following L^AT_EX expression and the prompt given in Polish.

Buffer (L^AT_EX)

$$\lim_{n \rightarrow +\infty} \frac{15n + 24}{2\sqrt{3n^2 + 4n} - 2}$$

Prompt (Polish)

oblicz granicę

Reference solution

For this task, the limit can be evaluated by factoring out n from both the numerator and the square root in the denominator:

$$\lim_{n \rightarrow +\infty} \frac{15n + 24}{2\sqrt{3n^2 + 4n} - 2} = \lim_{n \rightarrow +\infty} \frac{n \left(15 + \frac{24}{n}\right)}{2n \sqrt{3 + \frac{4}{n} - \frac{2}{n^2}}} = \frac{15}{2\sqrt{3}}.$$

After rational simplification, the expected final answer is

$$\boxed{\frac{5\sqrt{3}}{2}}.$$

This value should be accepted as the correct mathematical result, and any algebraically equivalent expression (for example, $\frac{15}{2\sqrt{3}}$ or numerically equivalent rationalizations) is treated as correct in the evaluation.

Below, we summarize the behavior of each model.

GPT-5:

The model required 21 s to process the query and returned the correct value

$$\frac{5\sqrt{3}}{2}.$$

The answer was concise, fully correct, and syntactically well-formed.

claude-opus-4:

Despite fast execution (5 s), the model returned the value 0, which is mathematically incorrect. This indicates a failure in asymptotic reasoning for this task.

mistral-large:

The model responded very quickly (1 s) and produced the expression

$$\lim_{n \rightarrow +\infty} \frac{15}{2\sqrt{3}},$$

which contains the correct simplified value but redundantly repeats the limit symbol. Thus, the answer is mathematically correct but stylistically suboptimal.

grok-4:

After 42 s of processing, the model returned the correct result

$$\frac{5\sqrt{3}}{2}.$$

The form was clear and conventionally acceptable.

gpt-4o:

The model completed the task in 1 s but produced the incorrect answer 7. This represents a complete computational failure for the given limit.

gpt-4o-mini:

Processing took 5 s. The model provided a lengthy derivation containing the correct intermediate expression $\frac{15}{2\sqrt{3}}$, but ultimately concluded with the incorrect simplified form

$$\frac{15}{3},$$

which no longer corresponds to the correct limit. The result therefore must be classified as incorrect.

Gemini:

The model gave the correct results in 63 s on the second attempt, after supplementing the instruction with the note: “provide only the final result.”

Additional Note on gpt-5-mini

In preliminary tests, we also evaluated the gpt-5-mini model. For both tasks presented above, its responses were identical to those produced by the full gpt-5 model, and the execution times were similar. This suggests that gpt-5-mini may represent a viable alternative in production scenarios, especially when lower operational costs or reduced resource usage are important considerations.

4.4. Experiment 3: Limit of an Exponential Sequence

In the third experiment, we evaluate how selected LLM models handle a limit that involves both a rational expression and an exponent growing linearly with n . As before, the task is executed within the Equation Wizard, which provides the mathematical expression via its buffer and forwards a short natural language prompt to the selected model.

Task Description

The Equation Wizard buffer is initialized with the following $\text{\LaTeX}$ expression:

$$\lim_{n \rightarrow \infty} \left(\frac{n+2}{n+5} \right)^{3n}.$$

The prompt issued to the model is simply:

oblicz granicę

Reference solution

The limit can be evaluated by rewriting the base of the power in a form suitable for asymptotic analysis:

$$\frac{n+2}{n+5} = 1 - \frac{3}{n+5}.$$

Using the standard limit $\lim_{n \rightarrow \infty} \left(1 + \frac{a}{n}\right)^{bn} = e^{ab}$, we obtain

$$\lim_{n \rightarrow \infty} \left(1 - \frac{3}{n+5}\right)^{3n} = e^{-9}.$$

Thus, the correct value of the limit is

$$\boxed{e^{-9}}.$$

Below, we include a summary of results obtained in this experiment by each of the tested models.

GPT-5:

The model completed the task in 32 s. It returned exactly the expected EQED representation of e^{-9} , which is both mathematically correct and syntactically valid. The output is concise and directly usable in the Equation Wizard.

claude-opus-4:

The model responded quickly (10 s) but produced an incorrect reasoning chain. It transformed the expression into a limit that diverges to infinity and concluded with the value ∞ . This result is mathematically incorrect and indicates a failure in handling exponential limits of the form $(1 + a_n)^{b_n}$.

mistral-large:

The response time was very short (2 s). However, the model returned a malformed expression in which the original exponential structure was lost and replaced by a rational expression inside the limit. No valid final value was provided. The result is both mathematically incorrect and syntactically misleading.

grok-4:

After 43 s, the model returned the correct EQED expression corresponding to e^{-9} . The output is mathematically correct, syntactically valid, and clearly formatted, although the response time is relatively long.

gpt-4o:

The model completed the task in 4 s but returned the value 1, which is incorrect for this limit. This indicates a misinterpretation of the asymptotic behavior of the expression.

gpt-4o-mini:

The model responded in 4 s. It reproduced the original limit expression and then concluded with the value 3, which is mathematically incorrect. Additionally, the output mixed EQED with standard $\text{\LaTeX}$ constructs, reducing its syntactic consistency.

gemini-flash-latest:

The Gemini model completed the task in 6 s and returned the correct EQED representation of e^{-9} . The result is mathematically correct, syntactically valid, and clearly formatted. Compared to larger models, it achieved this with a relatively short execution time.

Overall, Experiment 3 highlights substantial differences between models in handling exponential limits. Only GPT-5, grok-4, and gemini-flash-latest produced fully correct and well-formed results, while other models either failed mathematically or produced outputs unsuitable for direct use in the Equation Wizard.

4.5. Experiment 4: Solving a Biquadratic Equation

In the fourth experiment, we examine how selected LLM models handle a polynomial equation of higher degree that can be reduced to a quadratic form. We follow the test procedure from previous experiments.

Task description

The Equation Wizard buffer is initialized with the following equation:

$$4x^4 - 13x^2 + 3 = 0.$$

The prompt issued to each model is:

rozwiąż równanie

Reference solution

By introducing the substitution $y = x^2$, the equation can be rewritten as a quadratic equation:

$$4y^2 - 13y + 3 = 0.$$

Solving this equation yields

$$y = 3 \quad \text{or} \quad y = \frac{1}{4}.$$

Reverting to the original variable x , we obtain the solution set:

$$x = \pm\sqrt{3} \quad \text{or} \quad x = \pm\frac{1}{2}.$$

Thus, the correct solution of the equation is

$$x \in \left\{ -\sqrt{3}, -\frac{1}{2}, \frac{1}{2}, \sqrt{3} \right\}.$$

Observed model behavior.

- grok-4-latest returned the full and correct solution set in valid EQED notation.
- gpt-5 also returned the complete and correct solution set, properly formatted.
- gemini-flash-latest produced all four correct roots, though the presentation was less structured.
- gpt-4o produced an incorrect and incomplete result.
- gpt-4o-mini followed an invalid algebraic transformation, leading to an incorrect solution.
- claude-opus-4 applied an inappropriate quadratic formula directly to x , yielding incorrect results.

4.6. Experiment 5: Computation of a Sequence Term

Task. Compute the fifth term of the sequence defined by

$$a_n = (-1)^n \cdot \frac{2-n}{n^2},$$

with the instruction: “oblicz a_5 ”. All expressions were entered and evaluated using the Equation Wizard.

Reference solution. Substituting $n = 5$:

$$a_5 = (-1)^5 \cdot \frac{2-5}{5^2} = (-1) \cdot \frac{-3}{25} = \frac{3}{25}.$$

$$a_5 = \frac{3}{25}$$

Methodological note. For some models, the log contains two separate prompts:

- “oblicz a_5 ” (with subscript, 5A in the table).
- “oblicza a_5 ” (without subscript, 5B in the table).

Interestingly, in several cases the second phrasing produced better results.

Observed model behavior.

- gemini-flash-latest returned the correct value $\frac{3}{25}$ for both prompt variants.
- mistral-large-latest returned the correct result for the standard prompt, but produced a completely incorrect product-based construction for the variant without subscript.
- gpt-5 produced an incorrect value $\frac{3}{32}$ for the standard prompt, but returned the correct $\frac{3}{25}$ for the variant “oblicz a^5 ”.
- grok-4-latest consistently returned $\frac{3}{32}$, which is incorrect.
- claude-opus-4 returned $\frac{3}{32}$, systematically miscomputing the denominator.
- gpt-4o produced an incorrect value $-\frac{15}{32}$ for both prompt variants.
- gpt-4o-mini failed to complete the computation and did not produce a numerical result.

4.7. Experiment 6: Solving a Rational Equation

In this experiment, the models were asked to determine the value of the parameter a satisfying the equation entered into the Formula wizard:

$$2 + a = \frac{9a}{2a + 1}.$$

The prompt used in all tests was:

“dla jakiej wartości a równość jest prawdziwa”

After eliminating the denominator and simplifying, the equation becomes

$$(2 + a)(2a + 1) = 9a,$$

which leads to

$$2a^2 - 4a + 2 = 0,$$

and finally

$$(a - 1)^2 = 0.$$

Therefore, the correct answer is

$$a = 1.$$

GPT-5 returned the correct result directly after 26 s. The answer was concise, mathematically correct, and fully compatible with the expected EQED notation.

Claude Opus 4 also produced the correct result. Unlike GPT-5, it generated a complete derivation showing all intermediate algebraic transformations. Although some EQED constructs were slightly unusual, the reasoning remained mathematically valid and the final answer was correct. The response required 19 s.

Mistral Large failed to solve the task. Instead of transforming the equation, it returned a different expression,

$$2 + a = \frac{3}{a},$$

which is unrelated to the original problem. The output was therefore mathematically incorrect and unusable despite the very short response time of 2 s.

Grok-4 returned the correct solution $a = 1$ after 6 s. The answer was concise and correctly formatted.

GPT-4o produced the expression

$$a = \frac{2+1}{9-2},$$

which simplifies to $\frac{3}{7}$. Although this appears to be an attempt at symbolic manipulation, the result is incorrect and does not satisfy the original equation.

GPT-4o-mini did not perform any computation and merely repeated the original equation. Consequently, the task remained unsolved.

Gemini Flash returned the correct value $a = 1$ after 16 s. The answer was short, clear, and syntactically correct.

Overall, this experiment showed strong performance from GPT-5, Grok-4, Claude Opus 4, and Gemini Flash, all of which identified the unique solution correctly. In contrast, GPT-4o, GPT-4o-mini, and Mistral Large failed to produce a valid solution.

4.8. Experiment 7: Solving a Linear Inequality

In this experiment, the models were asked to determine the solution set of the inequality entered into the Formula wizard:

$$3 - x \geq \frac{5x - 1}{2}.$$

The prompt used in all tests was:

“podaj zbiór wszystkich rozwiązań nierówności”

Multiplying both sides by 2 and rearranging terms yields

$$6 - 2x \geq 5x - 1,$$

which simplifies to

$$7 \geq 7x,$$

and therefore

$$x \leq 1.$$

The solution set is

$$(-\infty, 1].$$

Unlike several previous experiments, this task exposed an interesting distinction between mathematical reasoning and output generation. The inequality itself is elementary and requires only a few algebraic transformations. Nevertheless, several models experienced difficulties not in the computation itself, but in expressing the final solution set using the expected notation.

GPT-5 required 71 s and returned the correct interval $(-\infty, 1]$. The answer was mathematically correct, although the response time was unusually high for a problem of this complexity.

Claude Opus 4 produced a complete derivation of the inequality and arrived at the correct solution set after 12 s. Both the reasoning and the final result were correct.

Mistral Large also solved the inequality correctly in only 5 s. However, the generated EQED notation contained several formatting artifacts and fragments of raw text. Despite these issues, the final interval could still be interpreted correctly.

Grok-4 returned the correct condition $x \leq 1$ after 8 s. The mathematical result was correct, although the answer was expressed as a set-builder expression rather than an interval. GPT-4o produced an entirely incorrect solution set,

$$(-\infty, -\frac{1}{2}] \cup [2, +\infty),$$

which appears unrelated to the original inequality.

GPT-4o-mini generated a detailed derivation and correctly obtained the condition $x \leq 1$. Although the final answer was not explicitly converted into interval notation, the mathematical result was correct and clearly presented.

Gemini Flash exhibited the most unusual behavior. Instead of solving the task directly, the model entered a meta-level discussion about Polish interval notation and attempted to reason how the answer should be written. As a result, no complete solution was produced. This suggests that the model became distracted by formatting considerations rather than focusing on the mathematical task itself.

Overall, this experiment demonstrates that even simple algebraic problems may reveal weaknesses in output formatting and instruction following. Most models successfully solved the inequality, but Gemini Flash failed because of output-generation issues rather than mathematical reasoning errors, while GPT-4o produced a genuinely incorrect mathematical result.

Additional observation—Gemini Flash reproducibility

An additional execution of Gemini Flash was performed using exactly the same input as in the original experiment. Unlike the first attempt, which produced a malformed response consisting largely of internal reasoning fragments and formatting-related text, the repeated query returned the correct solution set

$$x \in (-\infty, 1].$$

This behavior is noteworthy because the difference between the two outputs is not merely a variation in explanation style or formatting quality. The first response effectively failed to solve the task, while the second produced the fully correct mathematical result. The discrepancy therefore illustrates a significant lack of output determinism. For a relatively simple linear inequality, the model alternated between complete failure and complete success despite identical input data. Such variability may be problematic in educational applications, where reproducibility and reliability are often as important as raw problem-solving capability.

4.9. Experiment 8: Determining the Tangent of an Acute Angle

In this experiment, the models were asked to determine the value of $\tan \alpha$ from the equation

$$\frac{2 \sin \alpha + 3 \cos \alpha}{\cos \alpha} = 4,$$

under the additional condition that α is an acute angle.

The correct solution is obtained by dividing each term in the numerator by $\cos \alpha$:

$$2 \tan \alpha + 3 = 4,$$

which yields

$$2 \tan \alpha = 1,$$

and therefore

$$\tan \alpha = \frac{1}{2}.$$

GPT-5:

GPT-5 immediately returned the correct value $\tan \alpha = \frac{1}{2}$ without showing intermediate calculations. The result was mathematically correct and concise.

Claude Opus 4:

Claude Opus provided a complete step-by-step transformation of the equation, correctly converting the expression into a linear equation in $\tan \alpha$. The final answer was correct and well justified.

Mistral Large:

Mistral Large solved the problem correctly and presented a concise derivation. The reasoning was mathematically sound and led to the correct result.

Grok-4:

Grok-4 returned the correct value $\tan \alpha = \frac{1}{2}$ directly, without intermediate steps. The answer was fully correct.

GPT-4o:

GPT-4o produced an incorrect result, returning $\frac{2}{3}$ instead of $\frac{1}{2}$. No reasoning was provided, making the source of the error impossible to identify from the output.

GPT-4o-mini:

GPT-4o-mini correctly transformed the equation into

$$2 \tan \alpha + 3 = 4,$$

and obtained the expression

$$\tan \alpha = \frac{4 - 3}{2}.$$

Although the final simplification was not performed explicitly, the resulting expression is equivalent to the correct answer. This response was therefore assessed as essentially correct but incomplete.

Gemini Flash:

Gemini Flash returned the correct value $\tan \alpha = \frac{1}{2}$ immediately. The solution was concise and mathematically correct.

The experiment demonstrated that the task was comparatively easy for most models. Five models produced the exact correct answer, GPT-4o-mini stopped one simplification step before the final result, and only GPT-4o failed by providing an incorrect numerical value. Unlike several earlier algebraic experiments, this trigonometric transformation caused little difficulty and showed strong consistency across the tested systems.

4.10. Experiment 9: Simplification of an Algebraic Expression

In this experiment, the models were asked to simplify the expression

$$(3x + y)^2 - (3x - y)^2,$$

using the prompt “oblicz wartość wyrażenia”.

The correct solution follows directly from the difference-of-squares identity

$$a^2 - b^2 = (a - b)(a + b),$$

where

$$a = 3x + y, \quad b = 3x - y.$$

Therefore,

$$(3x + y)^2 - (3x - y)^2 = [(3x + y) - (3x - y)][(3x + y) + (3x - y)] = (2y)(6x) = 12xy.$$

GPT-5:

GPT-5 immediately returned the correct result $12xy$ without intermediate calculations. The response was concise and fully correct.

Claude Opus 4:

Claude Opus expanded both binomials using standard algebraic identities and correctly simplified the expression to $12xy$. The reasoning was complete and mathematically sound.

Mistral Large:

Mistral Large applied the difference-of-squares identity directly, obtaining the factors $(2y)$ and $(6x)$ before arriving at the correct result $12xy$. The solution was concise and correct.

Grok-4:

Grok-4 returned the correct result $12xy$ immediately, without showing intermediate steps.

GPT-4o:

GPT-4o produced an incorrect result of $16xy$. No derivation was provided, making the source of the error difficult to identify. The answer indicates a computational mistake in an otherwise straightforward symbolic transformation.

GPT-4o-mini:

GPT-4o-mini incorrectly concluded that the expression was equal to zero. This suggests that the model may have incorrectly interpreted the two squared terms as identical expressions and failed to account for the sign change inside the second binomial.

Gemini Flash:

Gemini Flash returned the correct result $12xy$ without additional explanation. The answer was concise and mathematically correct.

This experiment proved relatively easy for most of the tested models. Five of the seven systems obtained the correct result, while GPT-4o and GPT-4o-mini produced incorrect answers. Interestingly, the task required only elementary symbolic manipulation, yet the two OpenAI models based on the GPT-4o architecture failed despite their strong performance on several more complex problems in earlier experiments. This result highlights that occasional errors may occur even in simple algebraic transformations and that overall model capability does not always guarantee reliability on routine symbolic tasks.

Effect of prompt wording

The same algebraic expression was tested again using the alternative prompt “*uproszcz wyrażenie*” (“simplify the expression”) instead of “*oblicz wartość wyrażenia*” (“compute the value of the expression”). Most models produced results consistent with the original experiment. However, two notable changes were observed. GPT-4o, which previously returned the incorrect result $16xy$, correctly simplified the expression to $12xy$ in the repeated test. Conversely, Mistral Large, which had previously solved the problem correctly, produced the incorrect factorization $(2x + 2y)(2x - 2y)$. These results demonstrate that even minor

changes in natural language instructions may significantly affect model behavior, sometimes improving and sometimes degrading mathematical performance. The experiment therefore provides another example of prompt sensitivity in symbolic manipulation tasks.

Experiment 10—logarithmic expression

The tenth experiment evaluated the ability of the tested models to process a simple logarithmic expression:

$$\log_2 32 - \log_2 8.$$

The task required recognition of powers of two and direct application of the definition of the logarithm. Most models solved the problem correctly, typically by converting the logarithms into integer exponents:

$$\log_2 32 = 5, \quad \log_2 8 = 3,$$

thereby obtaining the correct result equal to 2.

The experiment revealed an interesting failure of GPT-4o-mini. Although the model correctly identified the first logarithm, it returned the intermediate expression 5/1 instead of completing the subtraction step. This suggests that the error did not originate from mathematical reasoning itself, but rather from premature termination of the solution process. All other evaluated models produced the correct answer. The task therefore confirms that elementary logarithmic transformations are reliably handled by modern frontier models, while smaller models may still exhibit output truncation or incomplete reasoning even for very simple problems.

5. Discussion

The results summarized in Table 1 present the details that allow a comparative evaluation of the novel aspects of AI-enhanced human–computer interaction implemented within equation editor software. LLM models taken into consideration were compared along two key factors: mathematical correctness and response time. Both aspects are critical in the context of interactive problem solving within the Equation Wizard and derived educational applications.

Table 1. Computation times and qualitative correctness ratings for experiments across all tested LLMs. Correctness ratings: ☆☆☆—fully correct and clearly presented result; ☆☆—correct result with minor presentation issues; ☆—partially correct, incomplete, or syntactically flawed result; ☹—mathematically incorrect result. Responsiveness categories: ⏱—up to 5 s; ⏱⏱—6–15 s; ⏱⏱⏱—16–30 s; ⏱⏱⏱⏱—more than 30 s; [!]
—no response received or processing failed. The reported ratings summarize the qualitative behavior consistently observed across repeated executions rather than statistical proportions of individual outcomes.

Task	GPT-5	Claude-Opus-4	Mistral-Large	Grok-4	Gpt-4o	Gpt-4o-Mini	Gemini-Flash
Solve the equation: $x^2 + 2x - 3 = 0$ Prompt: "rozwiąż równanie"	⏱⏱⏱⏱ ☆☆	⏱ ☆☆☆	⏱ ☆	⏱⏱⏱⏱ ☆☆	⏱ ☹	⏱ ☆	⏱⏱⏱⏱ ☆☆☆
Compute the limit: $\lim_{n \rightarrow +\infty} \frac{15n+24}{2\sqrt{3n^2+4n-2}}$ Prompt: "oblicz granicę"	⏱⏱⏱ ☆☆☆	⏱ ☹	⏱ ☆	⏱⏱⏱⏱ ☆☆☆	⏱ ☹	⏱ ☆	⏱⏱⏱⏱ ☆☆

Table 1. Cont.

Task	GPT-5	Claude-Opus-4	Mistral-Large	Grok-4	Gpt-4o	Gpt-4o-Mini	Gemini-Flash
Compute the limit: $\lim_{n \rightarrow \infty} \left(\frac{n+2}{n+5}\right)^{3n}$ Prompt: "oblicz granicę"	⌘⌘⌘⌘ ☆☆☆	⌘⌘ ⊖	⌘ ⊖	⌘⌘⌘⌘ ☆☆☆	⌘ ⊖	⌘ ⊖	⌘⌘ ☆☆☆
Solve the equation: $4x^4 - 13x^2 + 3 = 0$ Prompt: "rozwiąż równanie"	⌘⌘⌘⌘ ☆☆☆	⌘⌘ ⊖	⌘ ⊖	⌘⌘⌘⌘ ☆☆☆	⌘ ⊖	⌘⌘ ⊖	⌘⌘ ☆☆
Compute the 5th term of the sequence (5A): $a_n = (-1)^n \cdot \frac{2-n}{n^2}$ Prompt: "oblicz a_5"	⌘⌘⌘ ⊖	⌘⌘ ☆	⌘ ☆☆☆	⌘⌘⌘ ⊖	⌘ ⊖	⌘ ⊖	⌘⌘⌘⌘ ☆☆☆
Compute the 5th term of the sequence (5B): $a_n = (-1)^n \cdot \frac{2-n}{n^2}$ Prompt: "oblicz a5"	⌘⌘⌘ ☆☆☆	⌘⌘ ⊖	⌘⌘ ⊖	⌘⌘⌘⌘ ⊖	⌘ ⊖	⌘ ⊖	⌘⌘⌘⌘ ☆☆☆
Solve the equation: $2 + a = \frac{9a}{2a+1}$ Prompt: "dla jakiej wartości a równość jest prawdziwa"	⌘⌘⌘ ☆☆☆	⌘⌘⌘ ☆☆☆	⌘ ⊖	⌘⌘ ☆☆☆	⌘ ⊖	⌘ ⊖	⌘⌘⌘ ☆☆☆
Solve the inequality: $3 - x \geq \frac{5x-1}{2}$ Prompt: "podać zbiór wszystkich rozwiązań nierówności"	⌘⌘⌘⌘ ☆☆☆	⌘⌘ ☆☆☆	⌘ ☆☆	⌘⌘ ☆☆	⌘ ⊖	⌘⌘⌘ ☆☆	⌘⌘⌘ ☆☆☆
Determine $\tan \alpha$: $\frac{2 \sin \alpha + 3 \cos \alpha}{\cos \alpha} = 4$ Prompt: "oblicz tangens alfa (alfa jest kątem ostrym)"	⌘⌘ ☆☆☆	⌘⌘ ☆☆☆	⌘ ☆☆☆	⌘⌘ ☆☆☆	⌘ ⊖	⌘⌘ ☆☆	⌘⌘ ☆☆☆
Simplify the expression: $(3x + y)^2 - (3x - y)^2$ Prompt: "oblicz wartość wyrażenia"	⌘⌘ ☆☆☆	⌘⌘ ☆☆☆	⌘ ☆☆☆	⌘⌘ ☆☆☆	⌘ ⊖	⌘ ⊖	⌘⌘ ☆☆☆
Simplify the expression: $(3x + y)^2 - (3x - y)^2$ Prompt: "uproszcz wyrażenie"	⌘⌘ ☆☆☆	⌘ ☆☆☆	⌘ ☆	⌘ ☆☆☆	⌘ ☆☆☆	⌘ ⊖	⌘⌘ ☆☆☆
Compute the value: $\log_2 32 - \log_2 8$ Prompt: "oblicz wartość wyrażenia"	⌘⌘ ☆☆☆	⌘ ☆☆☆	⌘ ☆☆☆	⌘ ☆☆☆	⌘ ☆☆☆	⌘ ☆	⌘⌘⌘⌘ ☆☆☆

The rationale for the qualitative ratings presented in Table 1 is discussed in the corresponding experiment descriptions, particularly for responses exhibiting formatting deficiencies, syntactic inconsistencies, or other characteristics influencing the assigned category.

5.1. Overall Accuracy

The highest level of mathematical reliability was observed for gpt-5, grok-4, and gemini-flash. These models consistently produced correct results across most tasks, including multi-step algebraic transformations and limit computations.

However, an important distinction emerges when considering prompt sensitivity. While gemini-flash maintained high correctness across both variants of Experiment 5, gpt-5 required a simplified prompt (without subscript notation) to produce the correct result. Grok-4, despite strong performance in earlier experiments, failed on sequence evaluation tasks.

claude-opus-4 demonstrated mixed performance. It handled simpler algebraic transformations well but produced systematic computational errors in more complex expressions.

mistral-large showed inconsistent behavior. While it occasionally produced correct results, it also generated outputs that were syntactically invalid or logically incorrect, particularly in structured tasks.

The weakest performance was observed for gpt-4o and gpt-4o-mini, both of which frequently returned incorrect results or failed to complete the computation.

5.2. Response Time

In terms of latency, smaller and optimized models such as gpt-4o, gpt-4o-mini, and mistral-large consistently produced responses within 1–5 s. However, this speed advantage did not translate into reliable correctness.

Among the more accurate models, gemini-flash achieved a favorable balance, with moderate response times (typically 6–63 s) and consistently correct outputs.

gpt-5 and grok-4 required significantly longer processing times (often exceeding 30 s), reflecting the computational cost of deeper reasoning. claude-opus-4 remained relatively fast (5–12 s), but its accuracy limitations reduce the practical benefit of this latency.

Even though the absolute values of the measured response times may change with the development of the infrastructure, their relationships provide valuable information in terms of practical usefulness and the possibility of implementing production solutions using them.

5.3. Combined Ranking

The combined assessment prioritizes mathematical reliability over responsiveness. For educational applications, a mathematically incorrect solution is considered substantially more harmful than increased response time. Consequently, models producing incorrect results in even a small subset of representative task categories receive a strong penalty in the final qualitative assessment.

The ranking therefore reflects three factors considered jointly:

1. Mathematical correctness;
2. Syntactic compliance with EQED requirements;
3. Responsiveness of the complete AI-assisted workflow.

Per design, the first factor dominates the evaluation. EQED compliance serves as a secondary criterion because it directly affects the ability of the editor to visualize and process generated content. Response time is treated as a tertiary factor used primarily to distinguish between models exhibiting similar correctness levels.

Based on the observed results, the models can be ranked as follows:

1. gemini-flash—best overall balance of correctness and speed; robust across tasks and prompt variations.
2. gpt-5—very high accuracy, but sensitive to prompt formulation and relatively slow.

3. grok-4—strong performance in algebra and limits, but less reliable in sequence-based tasks.
4. claude-opus-4—fast and structured, but affected by recurring computational errors.
5. mistral-large—fast but inconsistent; occasional correctness offset by structural failures.
6. gpt-4o—very fast but unreliable for mathematical problem solving.
7. gpt-4o-mini—lowest overall performance; limited reasoning capability despite low latency.

5.4. Key Observations

Several important patterns emerge from the experiments:

- Correctness and latency are negatively correlated. Faster models tend to produce less reliable results.
- Prompt formulation has a measurable impact on model performance, especially in symbolic contexts.
- Some models produce mathematically correct results but fail to adhere to the required formal representation, which limits their usability in automated systems.
- Models optimized for general reasoning (e.g., gpt-5, gemini-flash) outperform lightweight variants in multi-step mathematical tasks.

This distinction between mathematical correctness and syntactic compliance emerged as one of the most important observations of the study. Several models successfully solved mathematical tasks while simultaneously producing outputs that could not be processed reliably by the Equation Wizard environment.

From the perspective of practical HCI applications, these dimensions should be evaluated separately. A mathematically correct answer that cannot be visualized, parsed, or further manipulated by software components may still reduce the overall usefulness of the interaction. Consequently, successful integration of LLMs into mathematical software requires both semantic correctness and structural compatibility.

5.5. Model Suitability for the Equation Wizard Environment

The following assessment should be interpreted as an exploratory qualitative comparison within the scope of the experiments performed in this study rather than as a universal ranking of contemporary LLM services. It reflects the observed behavior of the evaluated models under the adopted experimental conditions and with the selected benchmark tasks, and should therefore be interpreted as a practical indication of their suitability for integration with the Equation Wizard environment rather than as a general measure of their mathematical capabilities.

Within the evaluated task set and under the experimental conditions adopted in this study, gemini-flash exhibited the most favorable balance between mathematical correctness, syntactic compliance, and responsiveness. This observation should be understood in the context of the benchmark used in this work and the specific requirements of the Equation Wizard environment. The obtained results suggest that gemini-flash appears particularly suitable for integration into the investigated AI-enhanced mathematical editing workflow.

A broader benchmark covering additional mathematical domains, larger task sets, and user-centered evaluation studies would be required before stronger conclusions regarding general model superiority could be justified.

Nevertheless, gpt-5 remains a strong alternative in scenarios where maximum reasoning accuracy is required and longer response times are acceptable. The obtained results provide preliminary evidence that selected frontier LLM services can be effectively integrated into multimodal mathematical editing environments. However, further studies involving larger task sets, broader groups of users, and dedicated usability evaluations are

required before educational or accessibility-oriented deployment recommendations can be formulated with greater confidence.

5.6. Limitations

Several limitations should be acknowledged.

First, although the number of experiments was expanded and additional categories such as trigonometric, logarithmic, inequality, and parameter-based tasks were included, the benchmark remains limited in comparison with large-scale mathematical evaluation suites.

Second, the study intentionally focuses on symbolic mathematical expressions suitable for direct manipulation within a mathematical editor. Consequently, text-formulated problems and broader educational scenarios were outside the scope of the experiments.

Third, the evaluated systems were accessed through provider-hosted REST services. Therefore, the observed responsiveness reflects the behavior of the complete service architecture rather than isolated model inference speed.

Finally, the investigated models evolve continuously and may change over time. The reported results should therefore be interpreted as observations obtained under specific experimental conditions rather than permanent characteristics of individual models.

6. Summary

This study investigated the quality of AI-enhanced human–computer interaction within a multimodal equation editor, particularly the ability of selected large language models to solve mathematical problems at the level of a Polish high school final exam. The experiments were conducted using the Equation Wizard enhanced with an AI assistant module and mobile assistant providing voice interface. AI assistant for Equation Wizard enables structured interaction with LLMs and supports MathML, $\text{\LaTeX}$ and internal EQED representations. Each model was evaluated on a set of representative tasks, including solving equations, computing limits, and evaluating sequence terms. The assessment considered not only mathematical correctness, but also syntactic validity of the results in the assumed math encoding format, which significantly affects the quality of visualization as well as the response time—the key factor influencing the total user experience.

The study confirms that contemporary LLM services can be integrated effectively into multimodal mathematical editing environments, although substantial differences exist between model families with respect to correctness, syntactic compliance, and responsiveness. The results further demonstrate that mathematical correctness and formal compatibility constitute partially independent dimensions of performance and should therefore be evaluated separately when AI services are integrated with mathematical software.

The results obtained within the adopted experimental configuration show noticeable differences between the evaluated models with respect to accuracy and reliability. Within the evaluated benchmark, GPT-5, Grok-4, and Gemini consistently demonstrated strong performance in multi-step reasoning tasks, albeit not without limitations. In particular, some models exhibited sensitivity to prompt formulation, as observed in Experiment 5. At the same time, faster models such as GPT-4o and GPT-4o-mini consistently underperformed in terms of correctness, despite very low latency. The experiments further revealed an inconsistency between the models' semantic understanding of custom-encoded input and their syntactic fidelity in generating mathematically equivalent output in the same encoding. This limitation indicates that even when the numerical result is correct, deviations from the prescribed output syntax or formatting can significantly reduce the practical usability of the solutions generated by LLMs.

The obtained results allow the research questions to be answered as follows.

RQ1 was answered positively. Contemporary LLM services can be effectively integrated into multimodal mathematical editing environments and successfully support solving representative secondary school mathematical problems.

RQ2 revealed substantial differences between models with respect to syntactic compliance. Several mathematically correct solutions could not be directly processed by the Equation Wizard because of formatting inconsistencies, confirming that correctness and formal compatibility should be evaluated separately.

RQ3 showed that, within the investigated Equation Wizard workflow, the most practically useful models were those combining high mathematical correctness with acceptable responsiveness and stable EQED generation. The results indicate that overall usefulness depends on the balance of these factors rather than on response speed alone.

The experimental results support Hypotheses 1 and 2. Hypothesis 3 was partially confirmed: models with the shortest response times did not consistently provide the best overall user experience, whereas correctness and reliability proved more important for practical use within AI-assisted mathematical problem-solving environments.

The main conclusion is that no single model is optimal in all aspects, but a clear trade-off exists between accuracy and response time. Within the evaluated task set and experimental conditions, gemini-flash demonstrated the most favorable balance between correctness, syntactic compliance, and responsiveness. Within the investigated experimental configuration, GPT-5 appeared to be a strong alternative for high-precision tasks, while other models may still be suitable in scenarios where response speed is prioritized over reliability. These findings highlight the importance of evaluating LLMs not only in terms of raw accuracy, but also in the context of structured workflows and user interaction constraints, as they operate in a particular environment and build dedicated human–computer interaction process.

Perhaps the most important observation is that mathematical correctness does not necessarily imply syntactic compatibility with downstream software tools. Several models successfully solved mathematical tasks while simultaneously generating outputs that could not be reliably processed by the editor. This distinction appears particularly relevant for future AI-assisted mathematical software.

Future work should include larger groups of participants, extended mathematical benchmarks, and dedicated usability studies in educational and accessibility-oriented scenarios.

Author Contributions: Conceptualization, A.B. and Z.S.; methodology, A.B. and Z.S.; software, A.B. and Z.S.; validation, A.B. and Z.S.; formal analysis, A.B. and Z.S.; writing—original draft preparation, A.B. and Z.S.; writing—review and editing, A.B. and Z.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Thinn, M.M.; Thu, Y.K.; Nwe, H.M.; Yee, N.N.; Myint, T.; Thant, H.A.; Supnithi, T. Machine translation of LaTeX based mathematical equations to spoken mathematics. In *Proceedings of the 2020 24th International Computer Science and Engineering Conference (ICSEC)*; IEEE: New York, NY, USA, 2020; pp. 1–6.
2. Cuartero-Olivera, J.; Hunter, G.; Pérez-Navarro, A. Reading and writing mathematical notation in e-learning environments. In *ELearn Center Research Paper Series*; Universitat Oberta de Catalunya (UOC): Barcelona, Spain, 2012; pp. 11–20.
3. Stefanova, T.; Georgiev, S. Possibilities for using AI in mathematics education. *Math. Educ. Math.* **2024**, *53*, 117–125.
4. Bin Mohamed, M.Z.; Hidayat, R.; Binti Suhaizi, N.N.; Sabri, N.B.M.; Bin Mahmud, M.K.H.; Binti Baharuddin, S.N. Artificial intelligence in mathematics education: A systematic literature review. *Int. Electron. J. Math. Educ.* **2022**, *17*, em0694.

5. Bier, A.; Sroczynski, Z. Rule based intelligent system verbalizing mathematical notation. *Multimed. Tools Appl.* **2019**, *78*, 28089–28110.
6. Bier, A.; Sroczynski, Z. AI Assistant for Editing Mathematical Expressions in Polish. In *Proceedings of the 44th International Business Information Management Association Computer Science Conference (IBIMA)*; Soliman, K.S., Ed.; IBIMA: King of Prussia, PA, USA, 2024.
7. Sancho-Vinuesa, T.; Córcoles, C.; Huertas, M.; Pérez-Navarro, A.; Marquès, D.; Eixarch, R.; Villalonga, J. Automatic Verbalization of mathematical formulae for web-based learning resources in an on-line environment. In *Proceedings of the INTED2009 Proceedings*; IATED: Valencia, Spain, 2009; pp. 4312–4321.
8. Fateman, R. *How Can We Speak Math?*; Technical Report; Computer Science Division, EECS Department, University of California at Berkeley: Berkeley, CA, USA, 2013.
9. Attanayake, D.; Denholm-Price, J.; Hunter, G.; Pfluegel, E.; Wigmore, A. Speech Interfaces for Mathematics: Opportunities and Limitations for Visually Impaired Learners. In *Proceedings of the IMA International Conference on Barriers and Enablers to Learning Maths: Enhancing Learning and Teaching for All Learners*; Institute of Mathematics and Its Applications (IMA): Essex, UK, 2015.
10. Riga, P.; Kouroupetroglou, G.; Ioannidou, P.P. An Evaluation Methodology of Math-to-Speech in Non-English DAISY Digital Talking Books. In *Proceedings of the International Conference on Computers Helping People with Special Needs*; Springer: Cham, Switzerland, 2016; pp. 27–34.
11. Maćkowski, M.; Brzoza, P.; Żabka, M.; Spinczyk, D. Multimedia platform for mathematics' interactive learning accessible to blind people. *Multimed. Tools Appl.* **2017**, *77*, 6191–6208.
12. Mazzei, A.; Monticone, M.; Bernareggi, C. Using NLG for speech synthesis of mathematical sentences. In *Proceedings of the 12th International Conference on Natural Language Generation*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 463–472.
13. Souza, A.; Freitas, D. Towards a prosodic model for synthesized speech of mathematical expressions in mathml. In *Proceedings of the 9th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 105–110.
14. Lupták, D.; Novotný, V.; Štefánik, M.; Sojka, P. WebMiaS on Docker: Deploying Math-Aware Search in a Single Line of Code. In *Proceedings of the International Conference on Intelligent Computer Mathematics*; Springer: Cham, Switzerland, 2021; pp. 159–164.
15. Shirai, S.; Nakahara, T.; Fukui, T. MathTOUCH Editor: Rich-text Editor for Math E-learning Using an Intelligent Math Input Interface. *J. Inf. Process.* **2023**, *31*, 775–785.
16. Shirai, S.; Pollanen, M.; Kang, S.; Fukui, T. Feature Expansion of the Equation Editor in Mathematics Classroom Collaborator (MC 2) for Smartphones. In *Proceedings of the 2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*; IEEE: New York, NY, USA, 2023; pp. 1–4.
17. Jain, N.; Yadav, R.; Karpathy, A. Bibby AI—AI Latex Editor writing assistant for researchers vs Overleaf Alternative vs OpenAI Prism. (Bibby AI Latex Editor). *arXiv* **2026**, arXiv:2602.16432.
18. Saravanan, D.; Muthuraj, M.; Rangasamy, P.; Me, S.R.; Tamizharasan, D. AI-Powered Real-Time Math Equation Editor with Multi-Modal Input for Enhanced Interaction. In *Proceedings of the 2025 International Conference on Electronics and Renewable Systems (ICEARS)*; IEEE: New York, NY, USA, 2025; pp. 1130–1137.
19. Grace Lau, C.; Tang, H.; Chien, T. Exploring foundation students' AI math tool preferences, duration of use, and confidence in mathematics problem-solving. *J. Math. Sci. Teach.* **2025**, *5*, em089. <https://doi.org/10.29333/mathsciteacher/16869>.
20. Wang, P.Y.; Liu, T.S.; Wang, C.; Li, Z.; Wang, Y.; Yan, S.; Jia, C.; Liu, X.H.; Chen, X.; Xu, J.; et al. A survey on large language models for mathematical reasoning. *ACM Comput. Surv.* **2026**, *58*, 209.
21. Fang, M.; Wan, X.; Lu, F.; Xing, F.; Zou, K. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *Sci. Data* **2025**, *12*, 1392.
22. Zhang, F.; Li, C.; Henkel, O.; Xing, W.; Baral, S.; Heffernan, N.; Li, H. Math-LLMs: AI cyberinfrastructure with pre-trained transformers for math education. *Int. J. Artif. Intell. Educ.* **2025**, *35*, 509–532.
23. Chudziak, J.A.; Kostka, A. Ai-powered math tutoring: Platform for personalized and adaptive education. In *Proceedings of the International Conference on Artificial Intelligence in Education*; Springer: Cham, Switzerland, 2025; pp. 462–469.
24. Oh, S. Integration of MATH41 and Generative AI in Pre-Service Mathematics Teacher Education: An Empirical Study on Lesson Design Competency. *IEEE Access* **2025**, *13*, 128959–128973.
25. Kaya, D.; Yavuz, S. Can generative ai and ChatGPT break human supremacy in mathematics and reshape competence in cognitive-demanding problem-solving tasks? *J. Intell.* **2025**, *13*, 43.
26. Schmitt, J.; Bérczi, G.; Dekoninck, J.; Feusi, J.; Gehringer, T.; Appenzeller, R.; Bryan, J.; Canova, N.; de Wolff, T.; Gaia, F.; et al. Improofbench: Benchmarking ai on research-level mathematical proof generation. *arXiv* **2025**, arXiv:2509.26076.
27. Weber, S. FormulAI—An AI-Powered Equation Generator That Creates Editable Microsoft Word Formulas from Plain-Text Descriptions in Seconds—No MathType, No LaTeX Knowledge Required. Available online: <https://formulai.io/> (accessed on 17 June 2026).

28. Bier, A.; Sroczyński, Z. Rule-based intelligent system for dictating mathematical notation in Polish. In *Proceedings of the ACHI: The Thirteenth International Conference on Advances in Computer-Human Interactions*; IARIA: Wilmington, DE, USA, 2020; pp. 352–356.
29. Bier, A.; Sroczynski, Z. Expanding the Capabilities of Voice User Interface for Math Formula Editor Through Interactive Commands. In *Artificial Intelligence and Machine Learning*; Soliman, K.S., Ed.; Springer: Cham, Switzerland, 2024; pp. 62–70.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.