AI-based Intelligent System for Applied problems: Model Generation Web Application Architecture*

Oleksandr M. Khimich[1,†], Oleksandr V. Popov[1,†], Tamara V. Chistyakova[1,†], Elena A. Nikolaevskaya[1,*], Pavlo S. Yershov[1,†]

[1] V.M Glushkov Institute of Cybernetics of the NAS of Ukraine, Academician Glushkov Avenue, 40, Kyiv, 03187, Ukraine

Abstract

Mathematical modeling and computer experiments are fundamental approaches for investigating complex processes in science, engineering, and other domains. In our earlier work, we introduced the concept and architecture of the AIMM (Artificial Intelligence for Mathematical Modeling) system, designed to integrate a web-based modeling interface, large language models (LLMs), and high-performance hybrid computing resources on the SKIT supercomputer. That study established the theoretical foundation and modular microservice architecture of AIMM. The paper continues this research by detailing the design and implementation of the Model Generation Web Application (MGWA), a key subsystem within AIMM. The MGWA provides a prompt-based pipeline workflow for stepwise construction of computational models, guiding users from informal problem statements to solver-ready representations. Furthermore, the developed prototype of MGWA has been validated through preliminary testing, demonstrating its applicability for real-world problem settings.

Keywords

mathematical modeling; Artificial Intelligence; parallel algorithms; ISCM; AIMM; MGWA, MIMD architecture; GPU; solution reliability; convolutional neural networks; modeling automation; machine learning

1. Introduction

Mathematical modeling and the related computer experiment are among the primary means of studying objects, processes, and phenomena of various natures—in science, engineering, economics, society, and beyond. Mathematical modeling, in particular, involves substituting a real-world system or phenomenon with its abstract mathematical representation, thereby enabling computational experiments to be carried out on a computer. Such an approach allows the exploration of intricate processes that would be otherwise impractical or prohibitively expensive to study in real-world settings, leveraging modern computational technologies and numerical methods [1].

The vast majority of such studies require solving systems of linear algebraic equations (SLAEs) with matrices of arbitrary structure and extremely large orders. The properties of a computational problem may differ from those of the original mathematical problem, since the input data in a computer are represented approximately [2-9]. This increases the need to ensure the reliability of solutions, especially when modeling complex physical and mechanical processes. Here, a key factor becomes the use of Artificial Intelligence (AI), capable of automating critically important stages—from problem formulation to mathematical model construction and the selection of optimal solution methods and algorithms. Powerful parallel computers and the rapid development of AI make it possible to automate all stages of model building and analysis [1].

* ProfIT AI'25: 5th International Workshop of IT-professionals on Artificial Intelligence, October 15–17, 2025, Liverpool, UK

* Corresponding author.

† These authors contributed equally.

khimich505@gmail.com (O.M. Khimich); alex50popov@gmail.com (O.V. Popov); tamara.chistjakova@gmail.com (T.V. Chistyakova); elena_nea@ukr.net (E.A. Nikolaevskaya); yershov.pavel.wsk@gmail.com (P.S. Yershov)

0000-0002-8103-4223 (O.M. Khimich); 0000-0002-1217-2534 (O.V. Popov); 0000-0002-5145-0189 (E.A. Nikolaevskaya); 0000-0002-9072-7996 (P.S. Yershov)

In practice, the advancement of high-performance computing is closely tied to the development of user-oriented software—intelligent systems that facilitate interaction in the language of a specific domain and automate the entire workflow of obtaining computational solutions. A powerful framework for this lies in knowledge-based technologies that integrate hybrid artificial intelligence techniques, including machine learning. Addressing these challenges in the core areas of computational mathematics (systems of linear algebraic equations, algebraic eigenvalue problems, systems of ordinary differential equations, and nonlinear equations) can be framed within a paradigm that unites three pillars: high-performance computing, computer mathematics, and artificial intelligence.

Publications [10]–[14] represent landmark studies that have shaped the development of artificial intelligence and GPT models. At present, the field of artificial intelligence is undergoing rapid and multifaceted development. Among the most widely known advancements are the products of OpenAI [15], including the GPT family of models. GPT represents a natural language processing technology built on a transformer architecture, trained on extensive collections of textual data. This enables the generation of coherent and high-quality texts. Through training on massive datasets, such models acquire the ability to capture both syntactic and semantic structures of language, which makes them particularly effective tools for constructing semantic networks and domain-specific models. The ChatGPT system [16] is based on the GPT-3 [17], GPT-3.5 [18], GPT-4 [19] and GPT-5 [20] large language models developed by OpenAI. Its fine-tuning has been achieved using a combination of supervised learning techniques and reinforcement learning approaches.

The development and implementation of artificial intelligence (AI) methods and technologies open up new opportunities, while simultaneously giving rise to a range of critically important challenges. Among the most vulnerable aspects are the risks associated with the unreliability and unsafe operation of AI systems (AIS). Such threats are caused by the unpredictable behavior of systems in situations that go beyond the data used for training or testing, limited input sequences, as well as possible failures due to imperfect hardware and software implementations, physical defects, or cyberattacks [21]. In addition, an important task is the formulation and enforcement of clear requirements for key AI characteristics, such as ethics, explainability, trustworthiness, and others, which have been defined and systematized in [22, 23]. In [24], the concept of guarantee-capable AI systems is proposed, based on the development of the von Neumann paradigm (VNP), presented through a set-theoretic description that takes into account various components—AI and AIS quality characteristics.

In our earlier work [25], we introduced a new approach to solving applied problems that involves using AI at all stages—from problem formulation to obtaining a reliable solution. AIMM (Artificial Intelligence for Mathematical Modelling) is an Intelligent System for research and solving applied problems, designed for the automatic investigation and solution of mathematical modeling tasks on multi-core computers with MIMD architecture and graphics processing units (GPUs). The system is developed on the basis of the Intelligent System of Computer Mathematics (ISCM) [26]. Special attention is paid to applying AI in the process of mathematical model construction, automatic selection of numerical methods, and implementation on hybrid computing architectures (CPU+GPU). Such an approach significantly enhances the efficiency and reliability of modeling, enabling a broader range of users to work with high-performance computing systems without deep expertise in applied mathematics or programming. Ultimately, this opens up new opportunities for research in fields where accuracy, adaptability, and speed of obtaining results are crucial—from engineering and medicine to economics and environmental science.

The system is aimed at the complete automation and optimization of all stages of solving complex applied problems—from formulating the problem in the language of the subject area, to building a mathematical model, adaptively selecting numerical methods, and obtaining reliable solutions. The proposed system is designed to support automated modeling and numerical solution of computational problems by integrating a modeling web interface, a large language model (LLM), and a high-performance computing backend deployed on the SKIT supercomputer [27]. The system

architecture builds on previous research on implementing computational solutions through the interaction of modules for formalization, processing, generation, and solving of mathematical problems [28].

Building upon the theoretical foundations and modular microservice architecture of AIMM, this study presents the design and implementation of the Model Generation Web Application (MGWA), a pivotal subsystem within the broader framework. MGWA implements a prompt-based pipeline modelling workflow, which systematically guides users from informal problem statements to solver-compatible representations through a series of semantically linked prompt stages. The current prototype has undergone initial validation, demonstrating its feasibility for practical modelling tasks. While the system has shown promising results, testing and iterative refinement remain in progress to ensure robustness and adaptability across a range of problem domains.

2. Prompt-Based Pipeline Architecture

In our earlier work [25], authors introduced the concept and architecture of the AIMM system, designed to automate mathematical modeling and the numerical solution of computational problems by integrating a web-based modeling interface, a large language model (LLM), and a high-performance computing backend deployed on the SKIT supercomputer. The architecture follows a modular microservice paradigm, combining three main subsystems: the web application for problem formalization, the LLM-assisted modeling module, and the SKIT-based computational backend. The concept of generating problem statements using large language models has already been successfully implemented in the GrantsForScience project [29]. Each stage of the modeling process is implemented as a distinct interaction with the LLM, guided by a specifically designed prompt constructed in line with prompt engineering principles [30]. This design enables flexible model construction, automatic selection of numerical methods, and efficient execution on hybrid CPU/GPU architectures.

This paper focuses on the architectural and implementation aspects of the Model Generation Web Application (MGWA), which serves as the entry point to the computational workflow within the AIMM system (Fig. 1).

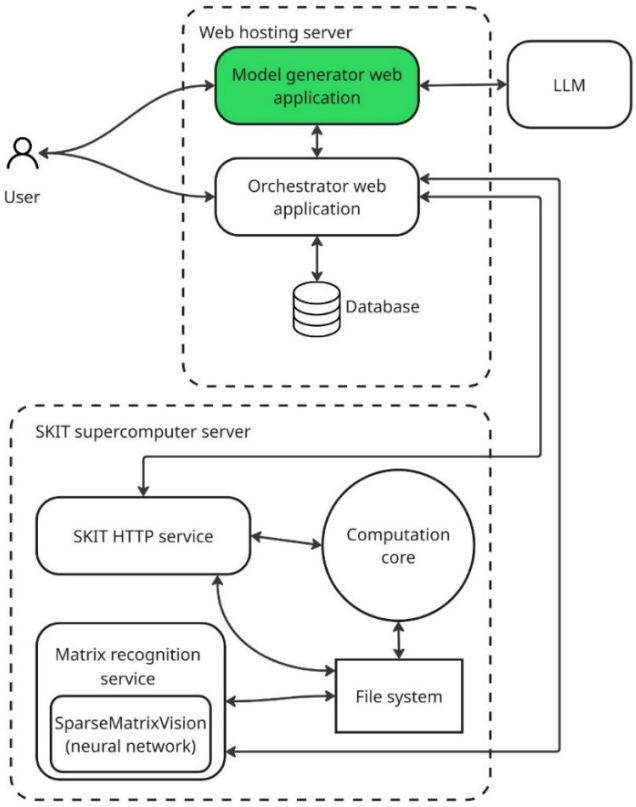

Figure 1: AIMM Architecture.

The MGWA provides a structured environment for the incremental construction of computational models from a formalized problem description. The modeling process is organized into sequential stages, where the user can examine, edit, and validate intermediate outcomes. These stages cover the formulation of the physical model (identifying processes and assumptions), the development of the mathematical description (differential equations and boundary conditions), discretization into a numerical scheme, synthesis of the computational model (choosing algorithms and methods), and finally, configuration of execution parameters such as solvers and data formats. At the final stage, the system generates a draft computational solution schema that encapsulates key properties of the intended numerical implementation—such as data structures, precision, and solver characteristics. This schema serves as a foundational interface for injecting problem-specific input data into AIMM and informs the subsequent selection of a suitable algorithmic implementation from the SKIT backend.

Consider the Prompt-Based Pipeline Architecture. Each task proceeds through a predefined sequence of transformation steps. These steps are selectively enabled for each pipeline, depending on the nature of the problem area. The default logical structure, based on the concept proposed in [31], follows four key stages:

1. *Formalized Description.* The user provides a natural language description of the problem. The system uses a structured prompt to extract entities such as inputs, outputs, goals, knowns, and constraints. The result is a logically organized representation suitable for further abstraction.
2. *Mathematical Model.* The physical model is reformulated into a mathematical abstraction, typically consisting of systems of equations, variational formulations, or operator-based representations. This step bridges physical reasoning with computational methods, ensuring that governing equations, boundary conditions, and assumptions are explicitly captured in formal mathematical notation. It plays a critical role in enabling the transition toward discretization, especially in continuum mechanics, field theories, or control systems. This stage may involve defining PDEs, ODEs, algebraic systems, or integral formulations.
3. *Discrete Model.* This step discretizes the conceptual model into algebraic forms—often sparse matrices, vectors, and systems of equations. It specifies the problem space in terms suitable for numerical manipulation. Sparse matrix formulations are especially prevalent in engineering and physical modeling. The system supports the description of matrix topology, dimensions, symbolic constraints, and storage formats.
4. *Computer Solution Draft.* The final stage generates a preliminary computational skeleton that will serve as input to downstream modules of ISCM [26]. This includes: expected data structures (e.g., sparse matrices, load vectors), orientation of data (e.g., row-wise vs column-wise formulation), precision constraints, accepted numerical methods or solver classes.

This representation is not a finalized codebase, but rather a structured blueprint for automated solution orchestration on HPC infrastructure.

2.1. MGWA Prototype

The MGWA prototype (AISolver) was implemented using Python Flask [32] and Jinja2 [33], with a lightweight HTML/CSS frontend and minimal UI dependencies (Fig. 2). The architecture is modular and extensible, comprising several key components:

- *Prompt Engine* – Responsible for dynamically executing configured prompt instructions using OpenAI-family large language models (LLMs). Each prompt defines its own model selection (e.g., GPT-4.1), temperature, token limits, and other parameters. Prompts are

designed to transform input models from the previous stage into structured representations for the next.

- *Workflow Engine* – Orchestrates the step-by-step generation of models in accordance with the structure of a selected pipeline. Each transition between steps is governed by the pipeline logic and proceeds automatically unless explicitly modified by the user.
- *Admin Interface* – Enables system developers to define and configure modeling pipelines, including prompt content, LLM parameters, and activation settings for each step. Prompt development follows modern prompt engineering practices and is debugged in a dedicated testing environment before deployment.
- *User Interface* – Supports the complete modeling workflow from task creation to results review. It guides users through step-by-step generation of models based on the pipeline configuration. Each model is displayed and edited on a dedicated page.

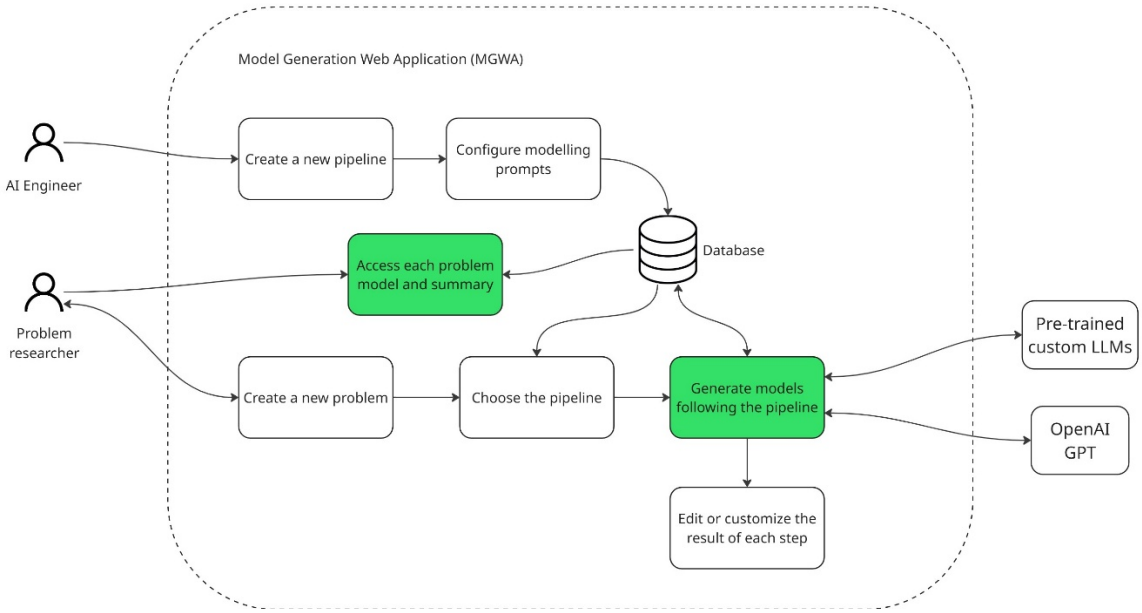

Figure 2: Model Generation Web Application (MGWA) Architecture.

The system's core data entities include:

- *Task (Problem)* – Represents a user-defined modeling problem. A task is associated with a specific pipeline that defines the modeling stages required for that problem type.
- *Pipeline* – A sequence of prompts representing distinct model abstraction levels (e.g., formalized, physical, mathematical, discrete, computational). The structure is configurable and may include optional steps depending on the domain.
- *Prompt* – A configurable wrapper around a call to an LLM. In system terminology, a prompt is not merely a textual instruction but a complete specification including LLM selection, prompt body, execution parameters, and its position in the pipeline.
- *Model* – The output of a single prompt execution. Models are stored as text-based content, which may include plain text, HTML, or JSON structures. The interface integrates an interactive CKEditor [34] for user interaction with models, supporting equations, images, and rich formatting.

System administrators (who are also developers) design and configure pipelines by defining prompt sequences. Each prompt step is bound to a specific transformation stage in the modeling process. Prompts can be enabled or disabled per pipeline to suit different types of tasks. The developer uses prompt engineering technics [35]. Once developed and tested on a testing environment, pipelines are deployed to the main application.

Domain experts, acting as end-users, create tasks by selecting an available pipeline and providing an initial formal description of the problem. The system then automatically generates models step-by-step using the configured prompts, following the chosen pipeline workflow, where the output of one step serves as the input to the next. Each result is presented on a dedicated page, with options for regeneration or manual refinement.

The modeling flow follows a fixed sequence: Formalized Description, Mathematical Model, Discrete Model, Computer Solution Draft.

All outputs can be edited and revisited. Users may also return to previous steps, adjust the input, and re-trigger generation from any point in the pipeline.

Upon completion, the user obtains a complete chain of model representations, each reflecting a distinct abstraction level [Fig. 3]. A dedicated summary view presents all generated content across the pipeline steps. Previously created tasks remain editable and can be iteratively improved.

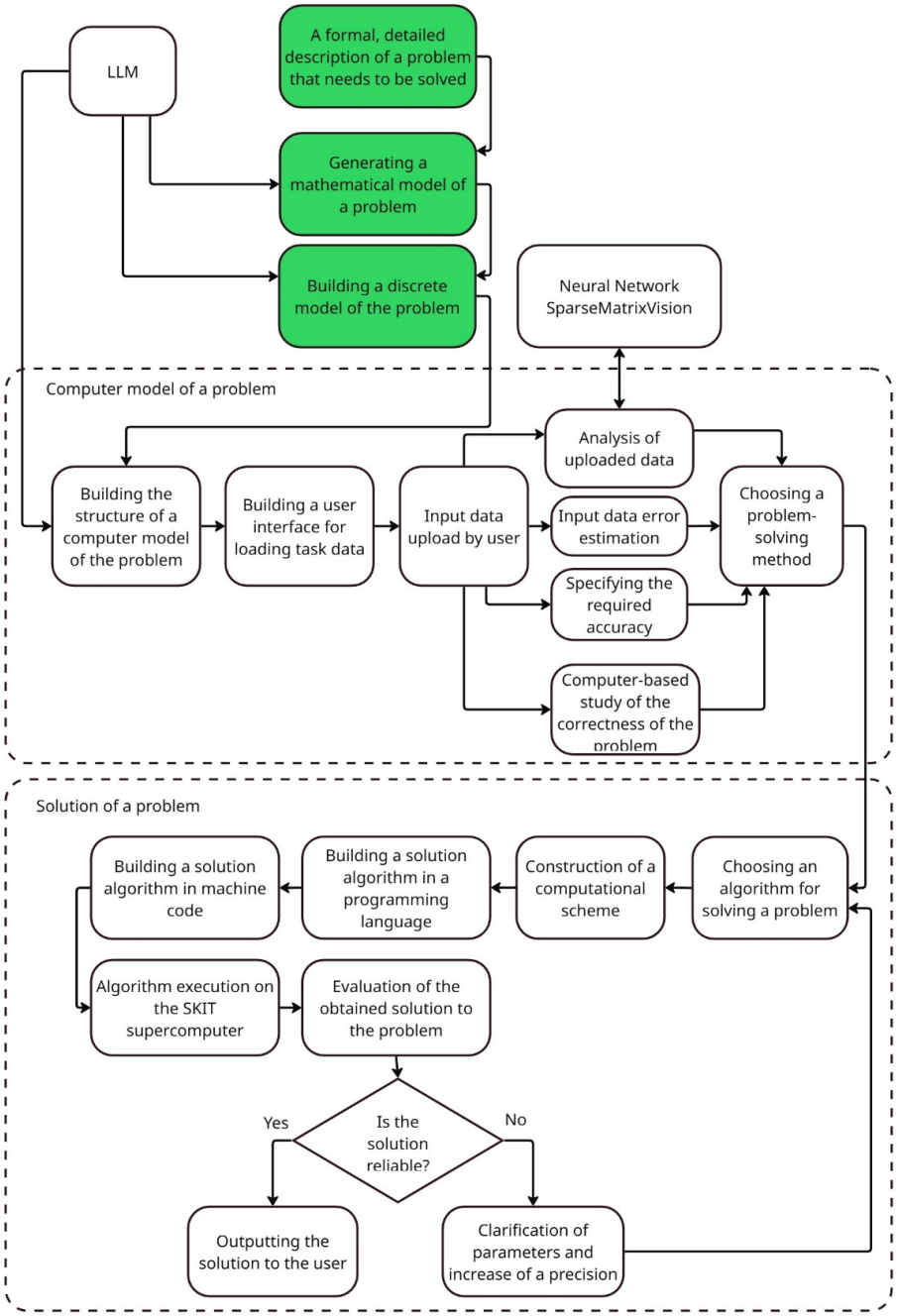

Figure 3: AIMM Prototype workflow.

By exposing prompt engineering and template control to the administrator, the system supports the creation of highly customized pipelines tailored to the specific requirements of different scientific and engineering domains.

Future extensions of the system are planned to enhance usability and collaboration. These include PDF export of results, tools for collaborative editing, integrated evaluation metrics, and activity logging. Following extended testing and debugging of the MGWA prototype, the system is

scheduled for integration with other AIMM components to support end-to-end problem-solving workflows. This will include automated task execution, input management, and solution orchestration within the AIMM infrastructure. Additionally, domain-specific pipelines will be developed to support modeling scenarios in fields such as computational mechanics, process engineering, and scientific computing.

Future extensions of the system are planned to enhance usability and collaboration. These include PDF export of results, tools for collaborative editing, integrated evaluation metrics, and activity logging. Following extended testing and debugging of the MGWA prototype, the system is scheduled for integration with other AIMM components to support end-to-end problem-solving workflows. This will include automated task execution, input management, and solution orchestration within the AIMM infrastructure. Additionally, domain-specific pipelines will be developed to support modeling scenarios in fields such as computational mechanics, process engineering, and scientific computing.

2.2. Prototype approbation

The AISolver prototype was tested on a real-world engineering problem “Cantilever beam subjected to a concentrated load at its free end”. A developer-level user can manage modeling pipelines (Fig. 4). Prompts are configured within pipelines according to the modeling domain and the specifics of the application area.

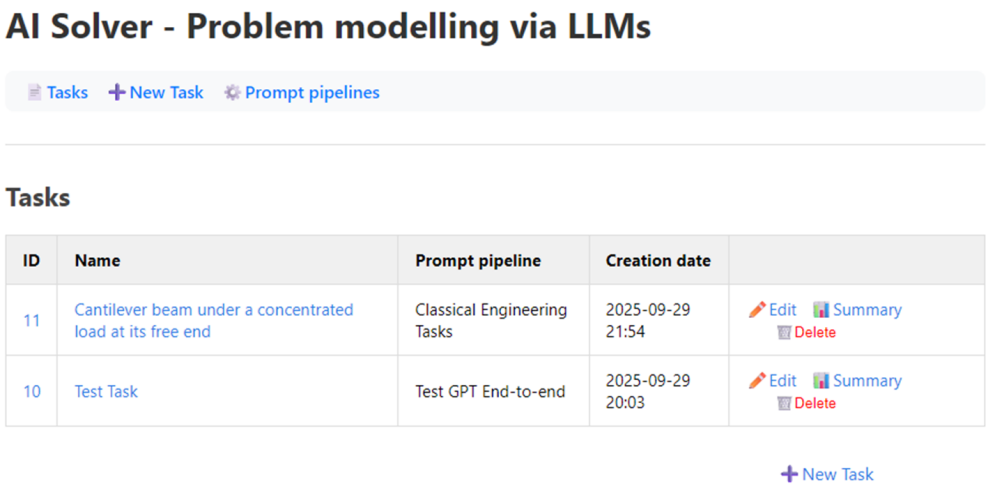

Figure 4: Prompt pipelines.

Figure 5 illustrates the editing of pipeline settings, including the prompt body, LLM type, and parameters used to configure the generation of a specific model within the pipeline workflow. The parameter configuration in AI Solver was designed to ensure a stable and reproducible modeling process. For all stages the GPT-4o model was used, providing an optimal balance between accuracy and flexibility for scientific tasks. The Temperature parameter was set to 0.6, maintaining a balance between diversity and logical consistency of results. For the physical model, the output size was limited to Max tokens = 1024, while for the mathematical model it was increased to 2048 to accommodate more complex systems of equations. The Top-P parameter remained at 1.0, ensuring full coverage of generation variants. This configuration ensures consistency between stages, clear structuring of generated outputs, and suitability for further automation of the model-building process within the AIMM system.

Figures 6–9 present the consecutive modeling stages: formalized problem description, Mathematical Model, Discrete Model, and Computer Solution Draft. Each stage is supported by LLM-powered transformations and allows interactive refinement of intermediate results.

A distinctive feature of the MGWA is its high configurability: expert users can define domain-specific prompt pipelines, while end-users are guided through a transparent and reproducible modeling process.

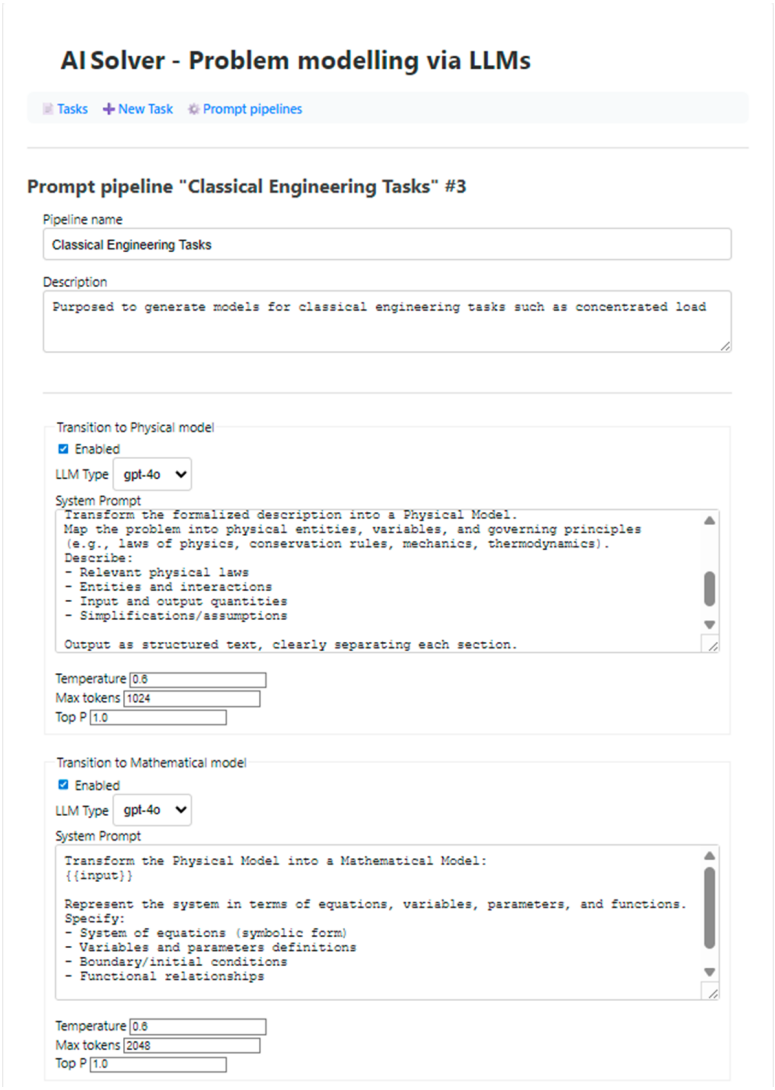

Figure 5: AISolver configuration.

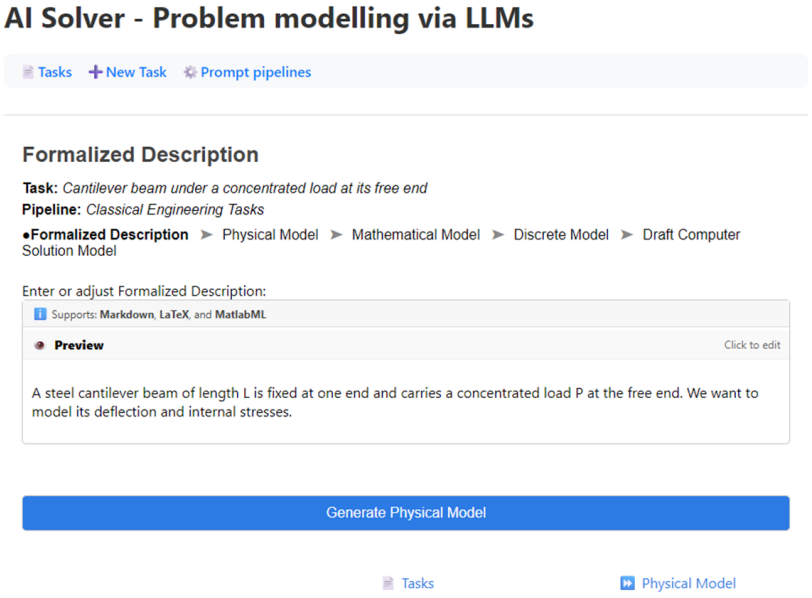

Figure 6: Formalized description entered by user.

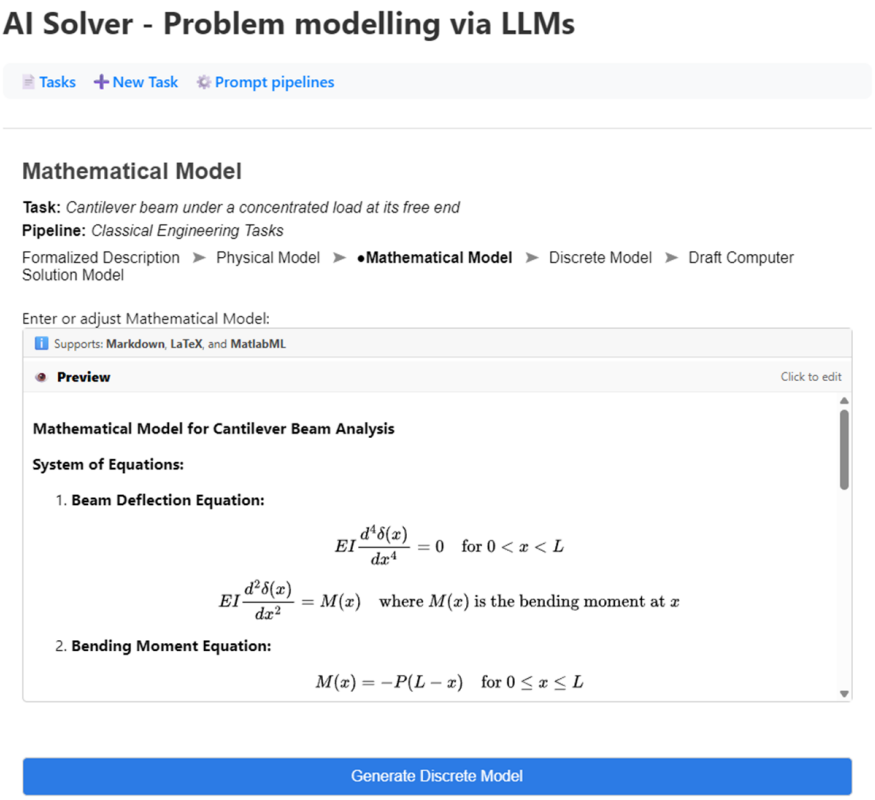

Figure 7: Mathematical Model

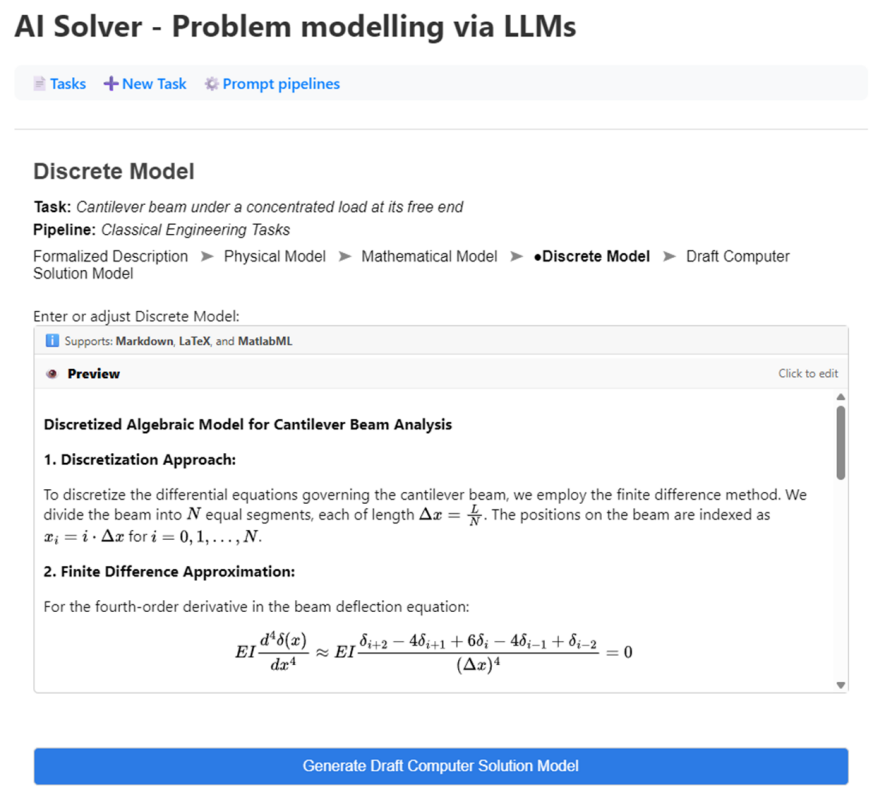

Figure 8: Discrete Model

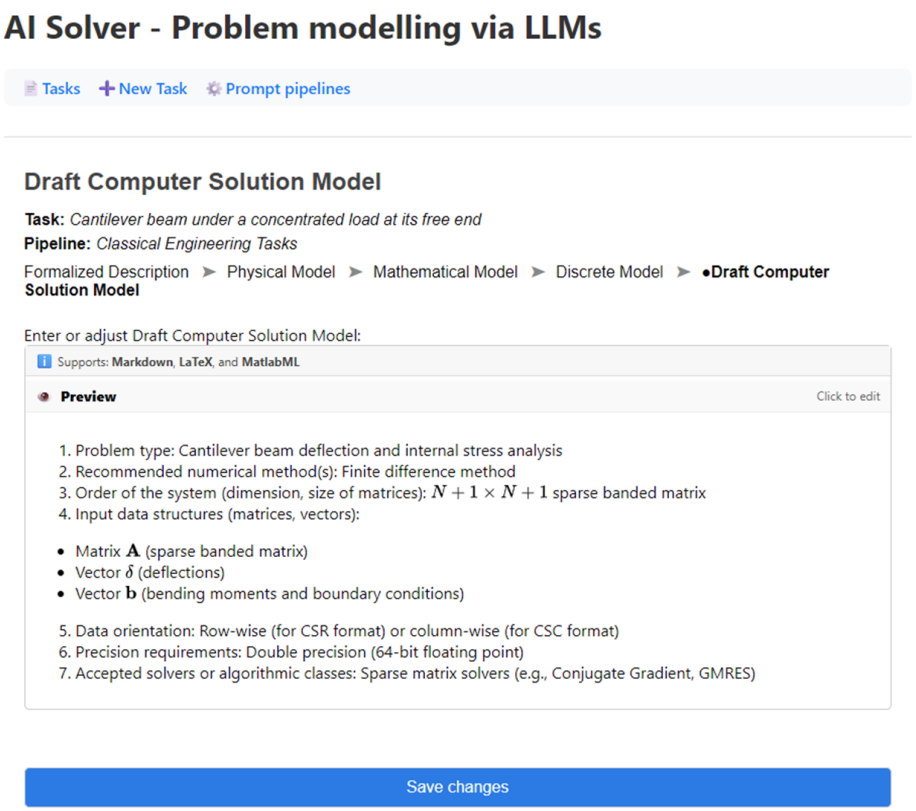

Figure 9: Generated Computer Solution Draft.

Each model is persistently stored in the database as structured text data (e.g., HTML, JSON), allowing end users to revisit, review, and refine them at any stage of the workflow. In particular, the Computer Solution Draft serves a dual role. On one hand, it provides a machine-readable schema that enables automated generation of the user interface for task-specific input configuration and integration with the AIMM solution workflow. On the other hand, it acts as a user-facing guide, informing the domain expert about the expected structure, semantics, and parameters of the input data required to proceed with problem resolution. The LLM demonstrates rapid execution (usually within seconds), while its effective performance is directly contingent upon user actions, such as input formulation and interaction timing

3. Conclusions

This paper introduces a prompt-driven web application that streamlines the early stages of scientific problem modeling. It supports modular configuration, domain-specific adaptation, and LLM-guided transformation of problem formulations. As part of the larger AIMM ecosystem, it plays a foundational role in transforming verbal task descriptions into computationally tractable formats.

A key innovation of MGWA is the use of prompt chaining, which structures the modeling workflow into a configurable pipeline and enables modular application of language-model-driven transformations at different levels of abstraction. The design of the system emphasizes flexibility: expert users or administrators can tailor domain-specific pipelines by adjusting prompts and parameters of the language model. At the same time, the tool is reusable, since defined pipelines can be applied to a variety of tasks and fine-tuned to match the specifics of particular problem categories. For end-users, the application offers a guided process, leading them step by step through model creation while preserving the possibility to regenerate or refine results at any stage.

Finally, the system is domain-independent, ensuring applicability across a wide spectrum of scientific and engineering disciplines.

This work extends the AIMM concept toward practical applicability, bridging the gap between natural-language task formulation and solver-ready computational structures. By enabling transparent, adaptive, and domain-agnostic modeling workflows, the MGWA strengthens the AIMM ecosystem as a foundation for intelligent numerical software. Future research will focus on expanding the configurability of prompt pipelines, enhancing collaborative modeling features, and applying the system to real-world domains.

The results confirm the feasibility of extending AIMM from conceptual architecture toward a working intelligent modeling environment. Future research will focus on expanding pipeline configurability, collaborative modeling features, and applying the system in domains such as navigation and motion control, where accuracy, adaptability, and real-time performance are essential.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-4o for grammar and spelling checks. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Khimich, V. Sydoruk, P. Yershov, Intellectualization Of Computation Based On Neural Networks For Mathematical Modeling, in: IEEE International Conference on Advanced Trends in Information Theory, ATIT, pp. 445-448. Decembre. 2019, doi: 10.1109/ATIT49449.2019.9030444. Available: https://ieeexplore.ieee.org/document/9030444

[2] J.H. Wilkinson, Rounding Errors in Algebraic Processes, London: H.M. Stat. Off, 1963.

[3] J.H. Wilkinson, C. Reinsch, Handbook for Automatic Computation. Springer Berlin, Heidelberg, 1971.

[4] V.V. Voevodin, Roundoff errors and stability in direct methods of linear algebra. M: CC MSU, 1969.

[5] I.N. Molchanov, Machine methods for solving applied problems. Algebra, approximation of functions, Naukova dumka, Kyiv, 1987.

[6] A.N. Khimich, Perturbation bounds for the least squares problem, Cybern. Syst. Anal. 32(3), 434–436 (1996). https://doi.org/10.1007/BF02366509

[7] J. Forsyth, M. Malcolm, K. Mouler, Machine Methods of Mathematical Computing, M.: Mir, 1980.

[8] J. Golub, C. Van Loan, Matrix Computations. M.: Mir, 1999.

[9] C. Lawson, R. Henson, Numerical solution of problems by the method of least squares. M.: Nauka, 1980.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is All You Need, NeurIPS (2017). doi:10.48550/arXiv.1706.03762.

[11] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL-HLT (2019). doi:10.48550/arXiv.1810.04805.

[12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, NeurIPS (2020). doi:10.48550/arXiv.2005.14165.

[13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language Models are Unsupervised Multitask Learners, OpenAI Blog (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[14] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving Language Understanding by Generative Pre-Training, OpenAI Blog (2018). URL: https://openai.com/research/language-understanding-generative-pre-training.
[15] OpenAI, URL: https://openai.com/
[16] GPTChat, URL: https://openai.com/chatgpt/
[17] OpenAI GPT-3, URL: https://openai.com/index/gpt-3-apps/
[18] OpenAI GPT-3.5, URL: https://openai.com/index/gpt-3-5-turbo-fine-tuning-and-api-updates/
[19] OpenAI GPT-4, URL: https://openai.com/index/gpt-4/
[20] OpenAI GPT-5, URL: https://openai.com/index/introducing-gpt-5/
[21] Ding, W., Abdel-Basset, M., Hawash, H. & Ali, A. M., Explainability of AI methods, applications and challenges: A comprehensive survey, Information Science 615 (2022) 238-292 doi: 10.1016/j.ins.2022.10.013
[22] V.Kharchenko, H.Fesenko, O. Illiashenko, Basic model of non-functional characteristics for assessment of artificial intelligence quality. Radioelectron. Comput Syst 2, (2022) 1-14. doi: 10.32620/reks.2022.2.11
[23] V.Kharchenko, H.Fesenko, O. Illiashenko, Quality Models for Artificial Intelligence Systems: Characteristic-Based Approach, Development and Application. Sensors 22(13) (2022) 1-32. doi: 10.3390/s22134865
[24] Kharchenko, V.S. Conceptual foundations of warrantable artificial intelligence systems, Reports of the National Academy of Sciences of Ukraine 2 (2025) 11-23. URL: http://jnas.nbuv.gov.ua/article/UJRN-0001571650
[25] A.N. Khimich, O.V. Popov, T.V. Chistyakova, E.A. Nikolaevskaya, P.S. Yershov, AI-based Intelligent System for Applied problems: conseption and architecture, in: 2025 IEEE 8th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC) October 21-24, 2025, Kyiv, Ukraine
[26] A.N. Khimich, T.V. Chistyakova, V.A. Sydoruck, P.S. Yershov, Intellectual computer mathematics system Inparsolver, Artificial Intelligence 25(4) (2020) 60–71, December doi: 10.15407/jai2020.04.060
[27] SKIT SKIT-5, URL: http://icybcluster.org.ua/index.php?lang_id=2&menu_id=5
[28] V.A. Sydoruk, P.S. Yershov, D.O. Bohurskyi, O.R. Marochkanych, Intellectualization of calculations for mathematical modeling tasks of complex processes and objects, Comp. Math. 1 (2019) 143-150
[29] O.M. Khimich, S.V. Yershov, E.A. Nikolaevskaya, P.S. Yershov, Development of an Intelligent Search Engine using GPT model for GrantsForScience platform, in: CEUR Workshop Proceedings, volume 3777, September 2024 [Online], pp 111-119, Available: https://ceur-ws.org/Vol-3777/short3.pdf
[30] L. Reynolds, K. McDonell, Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm, 2021, arXiv:2102.07350 [Online]. URL: https://arxiv.org/abs/2102.07350
[31] A.N. Khimich, I.N. Molchanov, A.V. Popov, T.V. Chisttyakova, M.F. Yakovlev, Parallel algorithms for solving problems of computational mathematics. Kyiv: Naukova dumka, 2008.
[32] Flask - Python Web Framework, URL: https://flask.palletsprojects.com/en/2.0.x/
[33] Jinja2, URL: https://jinja.palletsprojects.com/en/stable/
[34] CKEditor, URL: https://ckeditor.com/ckeditor-5/
[35] Prompt engineering technics, URL: https://www.promptingguide.ai/