# Agentic AI for Mathematical Equation Solving

Sergeev Nikita
*Innopolis university*

n.sergeev@innopolis.university

Semenove Diana
*Innopolis university*

di.semenova@innopolis.university

***Abstract*—This project proposes the development of an AI agent capable of solving a range of mathematical equations, from basic arithmetic to complex computational tasks. The agent will use large language model capabilities to interpret, analyze, and solve mathematical problems presented in natural language or symbolic form. The system will be evaluated on standard mathematical datasets with metrics for accuracy and computational efficiency.**

## I. Project Idea

The development of AI systems capable of solving algorithmically-generated mathematical problems presents unique challenges in symbolic reasoning and pattern recognition. Our project will create a specialized AI system that:

- Processes the structured mathematical problems from the DeepMind dataset [1], handling both textual and symbolic representations
- Implements modular solvers for each of the 56 mathematical categories (arithmetic, algebra, calculus, etc.)
- Generates exact solutions matching the dataset's answer format requirements
- Validates solutions through computational verification methods
- Adapts to different difficulty levels (easy/medium/hard) within the dataset's framework

## II. Formal Problem Definition

Let $x \in \mathcal{X}$ be a mathematical question posed in natural or symbolic language, and let $y \in \mathcal{Y}$ be the desired exact numerical answer. The agent approximates a function $f : \mathcal{X} \to \mathcal{Y}$ such that:

$$f(x) = \texttt{SymPy\_eval}(\texttt{CodeGen}(\texttt{LLM}(x)))$$

where `LLM` provides reasoning and planning, `CodeGen` generates executable code, and `SymPy_eval` computes the result symbolically.

## III. Methodology

Our AI agent follows a rigorous solve-check-learn pipeline for mathematical problem solving:

1) **Problem Understanding**:
   - Dual input processing: LLM for semantics + symbolic parser for equations
   - Automatic classification of problem type
2) **Solution Generation**:
   - Hybrid solver combining:
     - Rule-based methods for basic operations
     - Neural networks for complex problems
     - SymPy for symbolic mathematics
   - Outputs numeric answers with confidence scores
3) **Validation & Learning**:
   - Multi-step verification: noitemsep
     - Algebraic recomputation
     - Numerical approximation
     - Alternative method cross-checking
   - Continuous improvement through:
     - Error analysis Solution pattern memorization
     - Strategy optimization

## IV. Dataset Description

We utilize the Google DeepMind Mathematics Dataset [1], which contains:

- 2.5 million algorithmically generated math problems
- 56 modules covering arithmetic to advanced topics
- Multiple difficulty levels (easy/medium/hard)
- 90%-10% train-test split
- Problems in algebra, calculus, probability, and number theory

The dataset provides questions with exact answers but no step-by-step solutions.

## V. Agent Architecture Overview

Our system is designed as an agentic pipeline inspired by recent work on tool-augmented language models and planning-based reasoning agents [2], [3], [4]. The architecture integrates large language models for natural language understanding and reasoning, Python-based symbolic solvers for precise computation, and a refinement loop for self-correction.

The agent operates in the following stages:

1) **Problem Parsing and Classification:** The input is parsed into structured components. A lightweight classifier (prompt-based or rule-based) identifies the mathematical domain and required operations.
2) **Planning and Decomposition:** Inspired by ReAct [2] and MRKL-style agents [5], the agent formulates a high-level plan. For example, multi-step problems are decomposed into sequential symbolic subgoals.
3) **Code Generation and Tool Invocation:** Using chain-of-thought prompting, the agent generates annotated

Python code. SymPy is used for algebraic manipulation, integration, simplification, etc. This design mitigates common LLM limitations in symbolic math which are common for classical deep learning approaches for symbolic math [6].

4) **Self-Evaluation and Correction:** Following the MMAU protocol [4], the agent analyzes execution results. If outputs are ill-formed (e.g., non-integer, exception, zero-division), the agent re-plans or edits the code. Execution feedback is looped into the next reasoning step.

This modular and self-correcting architecture ensures interpretability, robustness, and alignment with tool-augmented problem-solving benchmarks.

## VI. Timeline and Contributions

TABLE I
Project Timeline

| **Task** | **Deadline** | **Responsible Member** |
|---|---|---|
| Proposal and Literature Review | June 27 | Both |
| Dataset Collection and Preprocessing | June 29 | Student 1 |
| Base System Architecture | June 31 | Student 2 |
| Natural Language Understanding Module | July 2 | Student 1 |
| Symbolic Computation Integration | July 5 | Student 2 |
| Solution Strategy Implementation | July 7 | Both |
| Verification System Development | July 11 | Student 1 |
| Evaluation Framework | July 15 | Student 2 |
| Final Integration and Testing | July 17 | Both |

## References


[1] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," *arXiv preprint arXiv:1904.01557*, 2019.

[2] S. Yao, J. Zhao, D. Yu, H. W. Chung, K. Narasimhan, and J. Liu, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[3] T. Schick, A. N. Dwivedi-Yu, L. Chilton *et al.*, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[4] D. Zhang *et al.*, "Mmau: A benchmark for evaluating math agents with tools," *arXiv preprint arXiv:2310.01810*, 2023.

[5] E. Wu *et al.*, "Autogpt: Building autonomous llm agents," 2022, online: https://github.com/Torantulino/Auto-GPT.

[6] G. Lample and F. Charton, "Deep learning for symbolic mathematics," in *International Conference on Learning Representations (ICLR)*, 2019.