Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.083643

ARTICLE

Improving Differential Equation Solving in Compact Language Models via Activation Steering and Reinforcement Learning

Anton Surkov, Vera Ignatenko* and Sergei Koltcov

Laboratory for Social and Cognitive Informatics, National Research University Higher School of Economics, Saint Petersburg, Russia

*Corresponding Author: Vera Ignatenko. Email: vignatenko@hse.ru

Received: 07 April 2026; Accepted: 10 June 2026; Published: 23 July 2026

ABSTRACT: Large language models have recently demonstrated promising capabilities in mathematical reasoning; however, their performance on tasks requiring strict symbolic manipulation, such as solving differential equations, remains limited, especially for compact models. In this work, we investigate whether activation steering combined with reinforcement learning can improve the quality of solutions generated by pretrained language models without modifying their weights. In particular, we focus on relatively small-scale models, which exhibit limited baseline performance on symbolic mathematical tasks, and study whether their capabilities can be enhanced through activation-level interventions. The proposed approach introduces trainable steering vectors that are injected into the internal activations of the model during generation. These vectors are learned separately for different classes of differential equations, forming a set of specialized steering experts. Training is performed using a reinforcement learning framework, where multiple candidate solutions are generated for each problem and evaluated using reward functions based on symbolic correctness, structural coincidence, or BLEU similarity. The optimization process updates only the steering parameters. Experiments are conducted on a general-purpose language model, a mathematically specialized model, and an instruction-tuned mathematical model using a synthetically generated dataset of ordinary differential equations. The results demonstrate that activation steering significantly improves the symbolic correctness and overall quality of generated solutions across most equation types. Steering applied across all transformer layers consistently outperforms steering restricted to the final layers. The largest improvements are observed for homogeneous and polynomial equations, while higher-order inhomogeneous equations remain the most challenging. Among the evaluated models, the Math-Instruct variant combined with activation steering achieves the strongest overall performance across most equation types, particularly for homogeneous, polynomial, and first-order equations. Notably, although the mathematically specialized model demonstrates stronger baseline performance than the general-purpose model, the proposed method substantially improves the latter, which in some cases achieves comparable or even higher correctness scores under steering. Overall, the results show that activation steering can significantly enhance the performance of compact language models, allowing them to partially compensate for their limited baseline capabilities on differential equation solving tasks. These findings indicate that activation steering combined with reinforcement learning serves as an effective and lightweight adaptation mechanism for improving the quality of solutions to differential equations generated by compact language models. The results highlight the potential of activation-level control as an alternative to full model fine-tuning, particularly in resource-constrained settings.

KEYWORDS: Large language models; steering; reinforcement learning; mathematical reasoning; differential equations

1 Introduction

Large language models (LLMs) have demonstrated strong performance across a wide range of natural language processing tasks and have recently shown promising capabilities in mathematical reasoning. In particular, modern LLMs are able to generate step-by-step solutions to mathematical problems, perform symbolic manipulations, and approximate analytical reasoning processes. These capabilities make LLMs a potentially useful tool for supporting mathematical problem solving and scientific computing tasks.

However, despite these advances, solving problems that require strict symbolic correctness remains a significant challenge for LLMs. Tasks such as solving differential equations involve not only multi-step reasoning but also precise algebraic transformations and exact equivalence between analytical expressions. In such settings, even small deviations in symbolic structure may lead to mathematically incorrect solutions. This limitation is particularly pronounced for compact language models, which are often preferred in practice due to computational constraints but typically exhibit weaker reasoning performance compared to larger models.

Existing approaches to improving mathematical reasoning in LLMs primarily rely on fine-tuning or reinforcement learning at the level of model parameters [1]. While effective, these methods require updating the full set of model weights, which can be computationally expensive and difficult to scale. In addition, recent research has shown that important behavioral properties of LLMs are encoded in their internal activation space, which opens the possibility of controlling model behavior through targeted activation-level interventions rather than weight updates [2,3].

Activation steering methods aim to modify model outputs by introducing structured changes to internal representations during inference. These methods have been successfully applied to control high-level model behavior, such as style, safety, and reasoning patterns. However, their application to tasks requiring strict symbolic correctness, particularly in the domain of differential equations, remains largely unexplored.

In this work, we investigate whether activation steering combined with reinforcement learning can improve the quality of solutions to differential equations generated by compact language models without modifying the original model weights. We focus on learning steering vectors that are applied to the internal activations of the model during generation and are optimized using task-specific reward signals.

The objective of this study is to evaluate the effectiveness of such activation-level control for improving both the structural and symbolic correctness of generated solutions. In particular, we aim to analyze how different reward formulations and steering configurations affect model performance across different classes of differential equations.

2 Related Work

2.1 Steering Methods in LLMs

Recent research shows that behavioral properties of LLMs are encoded as structured directions in the activation space of the model [2,3]. This observation forms the basis of representation engineering. The goal of representation engineering is not only to interpret but also to control the internal representations of neural networks. Within this paradigm, model behavior is viewed as a function of latent representations, and interventions in these representations are used to modify model outputs without fine-tuning.

A number of surveys [4,5] emphasize that representation engineering enables the modification of high-level model properties (such as honesty, safety, or reasoning style) through manipulations of activation vectors corresponding to these concepts. Within this framework, steering methods implement such modifications by introducing targeted interventions in the activation space. A common mechanism in these

approaches is the steering vector, which represents a direction in the activation space associated with a particular behavioral property of the model.

The key theoretical assumption underlying steering methods is the linear representation hypothesis, according to which high-level concepts are encoded as approximately linear directions in the activation space. Under this interpretation, modifying activations along a direction associated with a certain behavior (for example, reasoning depth) can systematically change the model output. Classical steering methods are therefore based on linear operations such as directional ablation or the addition of an activation vector, where the direction corresponding to the desired behavior is injected into or removed from the residual stream.

A significant development in this direction is activation engineering, which enables control of model behavior by modifying activations during the inference stage. In these approaches, steering vectors are computed from contrastive prompt pairs and injected into the model during the forward pass [2]. One of the most well-known methods in this category is Activation Addition (ActAdd), where the steering vector is defined as the difference between activations obtained from two contrastive prompts. Adding this vector shifts model generation toward the desired behavior while largely preserving the model's overall performance [2].

Building on this idea, Contrastive Activation Addition (CAA) constructs steering vectors by averaging activation differences across a set of positive and negative behavioral examples and applies these vectors to the residual stream during inference. This approach has been shown to effectively control high-level properties of model behavior and is compatible with both fine-tuning and reinforcement learning methods [6].

In a number of recent studies, steering vectors and representation engineering techniques are also investigated as tools for analyzing and improving reasoning processes in LLMs. For example, the authors of work [7] show that targeted modifications of internal representations can influence activations associated with reasoning processes and lead to improved performance on tasks involving logical inference and multi-step reasoning. Similarly, the authors of work [8] use steering vectors as a tool for interpreting reasoning mechanisms, enabling the identification of internal representations associated with reasoning processes and analysis of how their modification affects the quality of generated answers.

Another emerging direction in steering research is the use of sparse representations to improve the accuracy and stability of model control. In [9], the authors propose constructing steering vectors using features extracted by sparse autoencoders. Unlike traditional approaches that operate in dense activation spaces, this method allows for better disentanglement of features and facilitates the identification of components corresponding to specific behavioral properties of the model. As a result, steering interventions become more interpretable and stable. In [10], it is shown that steering vectors can improve generation quality and increase model efficiency at inference time without additional fine-tuning, indicating that appropriate modifications of internal activations can activate latent capabilities of the model.

Several recent works also explore new mechanisms for constructing and applying steering vectors. In [11], the authors propose a method of compositional steering, where control over model behavior is achieved using special steering tokens that allow combining multiple behavioral directions and enable more flexible regulation of model generation. Another line of work focuses on improving the reliability of LLM outputs. In [12], the authors introduce a steering approach aimed at improving the faithfulness of generated answers to the provided context, demonstrating that activation modifications can reduce hallucinations and improve contextual consistency.

Beyond practical applications, several works investigate the theoretical properties of steering vectors. In [13], the authors analyze the problem of identifying steering directions in activation space and show that

such vectors may be non-unique and dependent on the chosen representation space, which should be taken into account when interpreting steering interventions.

Another related research direction explores the combination of steering techniques with reinforcement learning [14,15]. While most steering approaches construct steering vectors directly from activation statistics or contrastive examples, reinforcement learning can also be used to optimize activation-level interventions using task-specific rewards [16]. In such settings, the parameters controlling activation modifications are updated through policy optimization based on the quality of generated outputs [17]. This formulation allows steering vectors to be learned directly from task performance signals rather than derived from pre-collected contrastive data. However, the application of reinforcement learning to optimize steering parameters remains relatively underexplored compared to traditional fine-tuning approaches.

Overall, existing research demonstrates that steering vectors can be used for controlling the behavior of LLMs, improving reasoning performance, enhancing the faithfulness of generated outputs, and for analyzing internal model mechanisms. At the same time, the application of steering methods to specific scientific domains remains limited.

2.2 Steering in Mathematical Tasks

In recent years, steering methods have been increasingly applied not only for controlling the behavior of large language models but also for improving their effectiveness in tasks requiring mathematical and formal reasoning. In particular, several studies investigate targeted modifications of internal representations to enhance reasoning quality and evaluate these approaches on mathematical benchmarks.

In [14], a method based on modifying the bias parameters of the model is proposed to influence reasoning processes. The results show that even small changes in internal parameters can significantly affect the reasoning trajectory and improve performance on tasks involving mathematical inference.

In [18], a method of adaptive steering at inference time is introduced. The proposed approach employs external verifiers to evaluate intermediate reasoning steps and dynamically adjust the model's internal representations. Experimental results demonstrate that this mechanism can significantly improve solution accuracy in mathematical and logical reasoning tasks.

Another line of research focuses on applying steering methods to formal theorem proving. In [19], it is shown that controlling internal representations can improve the efficiency of automated proof search. The proposed approach guides the generation of formal proofs and improves model performance on formal verification benchmarks.

Despite the rapid growth of research on steering methods for large language models, existing work has primarily focused on tasks involving general logical or mathematical reasoning, such as solving Olympiad-style problems, performing arithmetic computations, or formal theorem proving. In these studies, steering is mainly used to activate latent reasoning capabilities or to adjust the reasoning trajectory during inference. However, the application of steering methods to mathematics problems requiring strict algebraic transformations and the derivation of analytical solutions remains largely unexplored. In particular, to the best of our knowledge, there are currently no studies investigating the use of steering vectors for improving the quality of differential equation solutions generated by LLMs.

Therefore, in this work we aim to partially fill this gap by investigating the potential of steering vectors for improving the quality of differential equation solutions produced by compact language models.

3 Methods

In this section, we describe the proposed activation steering framework for improving the quality of ordinary differential equation solutions generated by pretrained language models. Let y denote the target analytical solution for a given differential equation E. Given a base model, our goal is to learn steering parameters that modify the model's internal activations during generation to increase the quality of the generated solutions $\hat{y}$.

In our approach, steering vectors are learned separately for each class of differential equations (e.g., second-order homogeneous, second-order inhomogeneous, polynomial, etc.). This design allows the model to adapt to class-specific patterns in generated solutions for different equation types.

From an architectural perspective, the proposed setup can be viewed as a modular collection of specialized steering experts. During training and evaluation, the corresponding steering expert is selected based on the equation-type labels available in the dataset. Thus, each steering expert is associated with a particular class of differential equations and is optimized independently for that category.

3.1 Models and Datasets

The experiments were conducted using three models: the Qwen2.5-Math-1.5B model [20], its instruction-tuned variant Qwen2.5-Math-1.5B-Instruct [21], and the base Qwen2.5-1.5B model [22,23].

These models represent different levels of domain specialization. The base Qwen2.5-1.5B is a general-purpose language model without explicit training for mathematical reasoning. The Qwen2.5-Math-1.5B model is further pre-trained on mathematical data, which enhances its ability to solve mathematical problems. Finally, the instruction-tuned variant Qwen2.5-Math-1.5B-Instruct is additionally aligned using instruction-following data and reinforcement learning, enabling it to better adhere to problem-solving formats and produce structured solutions.

These models were selected due to their moderate size and public availability, which enables efficient experimentation while maintaining sufficient reasoning capability for symbolic mathematical tasks.

The dataset was generated automatically using a symbolic pipeline based on SymPy. Differential equations and their analytical solutions were first constructed in symbolic form and only then converted into LaTeX using SymPy's built-in latex function. This approach was chosen to ensure consistency between symbolic and textual representations and to maximize LaTeX parseability.

The dataset includes six types of ordinary differential equations: second- and third-order homogeneous linear equations, second- and third-order inhomogeneous linear equations, polynomial equations, and first-order linear equations. Homogeneous equations were generated from sampled characteristic roots, while inhomogeneous and first-order equations were solved symbolically using SymPy's dsolve function. Polynomial equations were generated in the form $y' = p(x)$, where $p(x)$ is a randomly sampled polynomial.

Each generated equation–solution pair was then automatically validated using sympy. parsing.latex.parse_latex. Only pairs for which both the equation and the solution were successfully parsed were included in the final dataset, while invalid examples were logged separately.

The final dataset was stored in tabular form and included the equation, its analytical solution, and the corresponding equation type. The dataset was subsequently divided into training and test subsets. The training set contains 200 equations for each equation type, while the test set contains 100 equations per type. Importantly, the training and test sets were strictly separated and contained different differential equations in order to prevent memorization of symbolic solutions.

Training was performed separately for each equation type.

3.2 Multi-Steering Experts

To control the model's behavior, we employed the MoSE (Multi-Steering Experts) framework, in which each equation type is associated with its own steering expert. A steering expert consists of trainable vectors injected into selected transformer layers. Let $h_l \in \mathbb{R}^d$ denote the hidden representation at layer l. The steered hidden state is defined as

$$\tilde{h}_l = h_l + v_l^{(t)}, \tag{1}$$

where $v_l^{(t)} \in \mathbb{R}^d$ is the steering vector corresponding to equation type t at layer l. Thus, steering is implemented as an additive intervention in the activation space. Depending on the experimental setting, steering is applied either to all transformer layers or only to the last five layers.

Although the proposed Multi-Steering Experts (MoSE) framework conceptually supports multiple steering experts with inference-time expert selection depending on the equation type, the current study adopts a modular training setup in which each steering expert is trained independently in a separate run, with one expert corresponding to a single equation type. This design choice was primarily motivated by compatibility with the standard Group Relative Policy Optimization (GRPO) implementation provided by the Transformer Reinforcement Learning (TRL) framework, allowing the use of stable and widely adopted reinforcement learning infrastructure without introducing a custom multi-expert GRPO pipeline.

During evaluation, the equation type was obtained from the dataset labels and used to select the corresponding steering expert. The resulting steering vectors can therefore be jointly stored and selected conditionally depending on the target equation category, which remains consistent with the intended MoSE framework. In future work, this setup could be naturally extended with an automatic routing procedure, for example by incorporating an equation-type classifier or another expert-selection mechanism.

3.3 Training Procedure

Training is performed using a reinforcement learning framework implemented through the TRL library with the GRPO algorithm. GRPO is a policy optimization method that updates model behavior based on relative reward signals within groups of generated samples.

The parameters of the pretrained language model remain frozen throughout the entire training process; only the steering vectors are optimized. For each input problem, the model generates multiple candidate solutions, which are evaluated by a reward function measuring the quality of the generated outputs. The resulting reward signals are then used to update the steering vectors via policy optimization. In our study, we consider several reward functions reflecting different aspects of solution quality, including symbolic correctness, structural similarity, and token-level overlap. A detailed description of these reward functions is provided in the next subsection.

The above training procedure can be formalized as follows. For clarity of exposition, we adopt a simplified sequence-level formulation of the GRPO objective, omitting token-level normalization and importance sampling terms used in the full implementation [24,25].

Given an input E_i, the model generates K candidate outputs $\hat{y}_{i,1}, \hat{y}_{i,2}, \ldots, \hat{y}_{i,K}$. Each candidate is assigned a scalar reward

$$r_{i,k} = R(\hat{y}_{i,k}, y_i), \tag{2}$$

where $R(\cdot, \cdot)$ denotes the reward function.

The GRPO objective maximizes the expected reward of generated samples under the policy π_θ, where θ denotes the parameters of the steering vectors:

$$\mathcal{L}(\theta) = \mathbb{E}_{\hat{y}\sim\pi_\theta(\cdot|E_i)}\left[R(\hat{y}, y_i)\right]. \tag{3}$$

In practice, the policy is updated using group-based advantage estimation over the set of generated candidates:

$$A_{i,k} = r_{i,k} - \frac{1}{K}\sum_{j=1}^{K} r_{i,j}, \tag{4}$$

and the gradient update is approximated as

$$\nabla_\theta \mathcal{L}(\theta) \approx \sum_{k=1}^{K} A_{i,k}\, \nabla_\theta \log \pi_\theta(\hat{y}_{i,k} \mid E_i). \tag{5}$$

Thus, the GRPO optimizer updates the steering vectors to increase the expected reward over the generated samples.

3.4 Reward Function

In this study, the reward function is designed to reflect symbolic properties of the generated solutions. In particular, we consider rewards based on symbolic correctness, structural coincidence, and sequence-level similarity. A format-based reward was initially explored but was not used in the final experiments, as compact models often failed to consistently follow strict output formatting constraints, which led to a reduced number of valid generated solutions.

In our experiments, the reward is defined as either $R = R_{\text{correct}}$, $R = R_{\text{coin}}$, or $R = R_{\text{BLEU}}$, depending on the configuration. Here, R_{correct} evaluates mathematical correctness, R_{coin} measures symbolic term overlap between the target analytical solution and the generated one, and R_{BLEU} estimates the Bilingual Evaluation Understudy (BLEU) similarity score between the generated and reference solutions. These reward functions are described in more detail below:

- R_{coin}: the coincidence reward measures the proportion of symbolic terms in the predicted expression that match the target expression, with $R_{\text{coin}} \in [0, 1]$. This reward is used to assess whether steering improves partial structural agreement between expressions.
- R_{correct}: correctness is evaluated using symbolic verification. If the generated expression is symbolically equivalent (as determined using SymPy) to the reference solution, the reward is $R_{\text{correct}} = 1$. If exact equivalence is not established but both expressions are successfully parsed, a partial reward is assigned based on symbolic overlap: $R_{\text{correct}} = 0.5 \cdot R_{\text{coin}}$. Otherwise, $R_{\text{correct}} = 0$.
- R_{BLEU}: the BLEU [26] similarity score measures the overlap between n-grams in a candidate sequence and a reference sequence. Although originally proposed for machine translation, this metric is also commonly used for comparing mathematical expressions. In our setting, both expressions are first normalized and tokenized into sequences of mathematical tokens before computing BLEU. While BLEU does not guarantee mathematical equivalence, it provides a useful baseline reward for guiding the optimization of steering vectors when symbolic verification is not applied.

3.5 Evaluation Metrics

In addition to the reward functions used during training, we evaluate the quality of generated solutions using the same criteria, along with an additional strict symbolic equivalence metric. The evaluation metrics are described in more detail below.

- Coincidence score: measures the proportion of symbolic terms in the predicted expression that match the reference expression. This metric reflects partial structural agreement between expressions.
- Correctness score: evaluates symbolic correctness using the same formulation as in the reward function R_{correct}. It assigns a value of 1 if the generated expression is symbolically equivalent to the reference solution, a partial score $0.5 \cdot R_{\text{coin}}$ if both expressions are parseable but not equivalent, and 0 otherwise.
- BLEU score: computes the similarity between tokenized mathematical expressions based on n-gram overlap. Although BLEU does not guarantee mathematical equivalence, it provides a useful measure of structural similarity and serves as a baseline metric.
- Symbolic equivalence (SymPy equivalence): a strict binary metric that checks whether the generated expression is symbolically equivalent to the reference solution using SymPy. This metric takes values in $\{0, 1\}$ and reflects exact mathematical correctness.

In our experiments, symbolic equivalence is treated as the primary evaluation metric because it reflects exact mathematical correctness of the generated solution. At the same time, the hybrid correctness score is also reported in the main text because it captures partial structural agreement between generated and reference expressions in cases where exact symbolic equivalence is not achieved. Together, these metrics provide complementary information about both strict correctness and intermediate improvements in symbolic solution quality. In addition, BLEU-based evaluation results are reported for two of the considered models in order to analyze how token-level similarity between mathematical expressions correlates with symbolic correctness.

3.6 Experimental Settings

We consider six experimental configurations for steering: BLEU-based reward, symbolic-correctness reward, and coincidence-based reward. For each reward type, two variants are evaluated: (1) steering is applied to all transformer layers, and (2) steering is restricted to the last five transformer layers.

Training is performed for 5 epochs with a batch size of 8 and a maximum generation length of 1600 tokens. The generation parameters are set to temperature 0.7 and top_p = 0.95. The learning rate is set to $2 \cdot 10^{-4}$, and the number of candidate solutions generated for each input problem during training is set to 4. The following prompt was used: “Solve the following differential equation step by step, then give the final answer in LaTeX format. Equation: {equation} Solution:“” ”

In the present study, the reinforcement learning setup, reward configuration, and generation hyperparameters were intentionally kept fixed across all experiments in order to isolate the effect of activation steering under consistent conditions.

Notably, all experiments were conducted using a single NVIDIA A100 GPU. Since only the steering vectors are optimized, the proposed approach is computationally lighter than full model fine-tuning and can be trained using relatively modest hardware resources. Depending on the equation type, the training time varied from approximately 7–8 h for polynomial differential equations, which represent the simplest category in our setup, to approximately 20 h for third-order inhomogeneous equations, which were the most computationally demanding. These results indicate that the proposed framework can be trained efficiently while still providing substantial improvements in performance.

To ensure consistent reward computation and evaluation across all experimental configurations, the generated outputs undergo an additional preprocessing stage. For each generated response, the final mathematical answer is automatically extracted from the model output. The extracted answer is then normalized and converted into a symbolic expression whenever possible. In particular, mathematical expressions are expanded and simplified, the independent variable is normalized to x, and arbitrary integration constants are converted to a unified notation C. In addition, permutations of integration constants are taken into account during symbolic comparison.

Both reward computation and evaluation are based on these normalized symbolic representations, allowing the evaluation to focus on mathematical equivalence independently of non-essential differences in symbolic representation, such as variable naming, constant ordering, or term ordering.

Thus, the proposed methodology learns equation-type-specific steering vectors that act directly in the activation space of a frozen pretrained language model. Unlike most existing steering approaches that derive steering vectors from activation differences, the proposed framework learns steering parameters through reinforcement learning guided by symbolic correctness signals. This design makes it possible to study how activation-level control affects the exact correctness of generated solutions and structural agreement with reference analytical expressions.

4 Results

The full experimental results for all evaluation metrics, reward configurations, and equation types are available in the project repository on GitHub (https://github.com/hse-scila/MathSteering), together with the source code and datasets. In this section, we present the main results and focus on the most representative findings.

4.1 Results for Qwen2.5-Math-1.5B

Table 1 presents the results of symbolic equivalence evaluation (the most strict evaluation metric) for the Qwen2.5-Math-1.5B model for all equation types and all reward types for the setting when steering is applied to all layers. The results show that activation steering substantially improves exact equivalence across most equation types compared to the baseline model. The largest improvements in symbolic equivalence scores relative to the baseline model are observed for homogeneous ($\Delta = 0.32$) and first-order equations ($\Delta = 0.34$), where the best performance is achieved when optimizing the symbolic correctness reward. For polynomial equations, the coincidence-based reward leads to the highest equivalence scores (0.93). In contrast, higher-order inhomogeneous equations show only limited improvements ($0 - 0.12$) across all configurations.

Table 1: Effect of activation steering with different reward functions on the symbolic equivalence between generated and reference differential equation solutions for the Qwen2.5-Math-1.5B model when steering is applied across all transformer layers.

Equation Type	Baseline	R_{BLEU}	R_{coin}	R_{correct}
hom. 2	0.090	0.400	0.350	**0.410**
hom. 3	0.020	0.110	**0.140**	**0.140**
inhom. 2	0.010	0.050	**0.120**	0.070
inhom. 3	0.000	**0.069**	0.000	0.000
polynomial	0.810	0.820	**0.930**	0.890
first-order	0.380	0.640	0.710	**0.720**

Note: Boldface indicates the highest value(s) for each equation type.

Table 2 demonstrates the results for Qwen-math-1.5B in terms of correctness quality metric. Overall, steering significantly improves the correctness of generated solutions compared to the baseline model across all equation types. The best results are typically achieved when training is performed with the symbolic correctness reward R_{correct}, particularly for homogeneous equations and first-order equations. For inhomogeneous equations and polynomial cases, the coincidence-based reward R_{coin} yields slightly better performance. In particular, the polynomial category achieves the highest correctness scores overall, reaching 0.944 under the coincidence reward. Interestingly, optimizing BLEU reward leads not only to improvements in lexical similarity but also to substantial gains in symbolic correctness. This suggests that for the considered classes of differential equations, similarity between expressions correlates with their mathematical validity.

Table 2: Effect of activation steering on the symbolic correctness between generated and reference differential equation solutions for the Qwen2.5-Math-1.5B model when steering is applied across all transformer layers.

Equation Type	Baseline	R_{BLEU}	R_{coin}	R_{correct}
hom. 2	0.132	0.478	0.445	**0.5**
hom. 3	0.053	0.14	0.159	**0.168**
inhom. 2	0.063	0.173	**0.224**	0.212
inhom. 3	0.033	**0.129**	0.085	0.081
polynomial	0.815	0.83	**0.944**	0.896
first-order	0.466	0.677	0.751	**0.756**

Note: Boldface indicates the highest value for each equation type.

Table 3 demonstrates the results for the same model in terms of BLEU similarity score for all equation types for R_{BLEU} reward and R_{correct} reward for two configuration types: when steering is applied to all layers and the last 5 layers. These results show that steering applied across all transformer layers consistently outperforms steering restricted to the last five layers across all equation categories and reward configurations.

Table 3: Comparison of BLEU scores for differential equation solutions generated by the Qwen2.5-Math-1.5B model under different activation steering settings and reward functions across multiple equation types.

Equation Type	Baseline	R_{BLEU} (all)	R_{BLEU} (last 5)	R_{correct} (all)	R_{correct} (last 5)
homo2	0.204	**0.396**	0.212	0.332	0.209
homo3	0.176	**0.340**	0.199	0.324	0.172
inhomo2	0.158	**0.305**	0.166	0.301	0.171
inhomo3	0.046	**0.112**	0.072	0.098	0.060
poly	0.498	**0.616**	0.483	0.475	0.447
first_order	0.166	**0.209**	0.186	0.176	0.162

Note: Boldface indicates the highest value for each equation type.

4.2 Results for Qwen2.5-Math-Instruct-1.5B

Table 4 reports the results in terms of exact symbolic equivalence for Qwen2.5-Math-Instruct-1.5B model. The results demonstrate that activation steering significantly improves exact equivalence for most equation types. In particular, for homogeneous equations, equivalence increases from 0.11 to 0.56 for second-order equations, and from 0.02 to 0.24 for third-order equations under R_{coin} reward. For polynomial and first-order equations, the model achieves the highest performance overall. In particular, polynomial and first-order equations reach near-perfect equivalence scores, up to 0.99 under R_{correct} reward. These results indicate

that for simpler equation types, activation steering can almost fully recover correct analytical solutions. The BLEU-based reward demonstrates mixed behavior across different equation classes. For homogeneous equations, optimizing R_{BLEU} leads to very low symbolic equivalence scores, even below the baseline model in the second-order case. At the same time, for polynomial equations, the BLEU-based reward achieves strong performance, reaching equivalence scores of 0.98. For higher-order inhomogeneous equations, only limited improvements in equivalence scores were observed across all reward configurations.

Table 4: Effect of activation steering with different reward functions on the symbolic equivalence between generated and reference differential equation solutions for the Qwen2.5-Math-Instruct-1.5B model when steering is applied across all transformer layers.

Equation Type	Baseline	R_{BLEU}	R_{coin}	R_{correct}
hom. 2	0.110	0.010	**0.560**	0.510
hom. 3	0.020	0.010	**0.240**	0.220
inhom. 2	0.020	0.100	**0.140**	0.120
inhom. 3	0.000	**0.020**	0.010	0.000
polynomial	0.640	0.980	0.970	**0.990**
first-order	0.507	0.660	0.940	**0.990**

Note: Boldface indicates the highest value for each equation type.

Table 5 presents the results for the Qwen2.5-Math-Instruct-1.5B model in terms of symbolic correctness. Overall, activation steering leads to substantial improvements in the correctness of generated solutions across most equation types compared to the baseline model. The most significant gains are observed when using R_{coin} and R_{correct} reward functions. In particular, for homogeneous equations, correctness increases from 0.158 to 0.667 and 0.623 for second-order equations, and from 0.021 to 0.360 and 0.344 for third-order equations under coincidence- and correctness-based rewards, respectively. Similar improvements are observed for inhomogeneous equations, although the gains are more moderate. For polynomial and first-order equations, the model achieves the highest performance overall. In contrast, optimizing the BLEU-based reward leads to weaker improvements in symbolic correctness and, in some cases, even degrades performance compared to the baseline. This suggests that token-level similarity alone is insufficient for reliably improving the symbolic correctness of generated differential equation solutions.

Table 5: Effect of activation steering with different reward functions on the symbolic correctness between generated and reference differential equation solutions for the Qwen2.5-Math-Instruct-1.5B model when steering is applied across all transformer layers.

Equation Type	Baseline	R_{BLEU}	R_{coin}	R_{correct}
hom. 2	0.158	0.015	**0.668**	0.623
hom. 3	0.021	0.038	**0.360**	0.344
inhom. 2	0.108	0.280	**0.314**	0.278
inhom. 3	0.062	0.081	0.089	**0.090**
polynomial	0.740	0.985	0.978	**0.993**
first-order	0.601	0.705	0.950	**0.990**

Note: Boldface indicates the highest value for each equation type.

Overall, the results for the Qwen2.5-Math-Instruct-1.5B model further confirm the effectiveness of activation steering.

4.3 Results for General Qwen2.5-1.5B

Table 6 presents the results of symbolic equivalence evaluation for the Qwen2.5-1.5B general-purpose model. The results show that activation steering leads to substantial improvements in exact symbolic correctness compared to the baseline across most equation types. The largest gains are observed for homogeneous second-order ($\Delta = 0.44$), polynomial ($\Delta = 0.33$), and first-order equations ($\Delta = 0.46$), where the model achieves high equivalence scores under coincidence-based and correctness-based rewards. For higher-order inhomogeneous equations, performance remains limited, with near-zero equivalence across all configurations.

Table 6: Effect of activation steering with different reward functions on the symbolic equivalence between generated and reference differential equation solutions for the Qwen2.5-1.5B model when steering is applied across all transformer layers.

Equation Type	Baseline	R_{BLEU}	R_{coin}	R_{correct}
hom. 2	0.070	0.020	0.500	**0.510**
hom. 3	0.020	0.000	**0.180**	**0.180**
inhom. 2	0.010	0.020	**0.040**	0.030
inhom. 3	0.000	0.000	0.000	0.000
polynomial	0.650	0.850	**0.980**	0.950
first-order	0.380	0.600	0.770	**0.840**

Note: Boldface indicates the highest value(s) for each equation type.

Among the considered reward functions, the coincidence-based reward R_{coin} and the symbolic correctness reward R_{correct} generally achieve the best results, indicating that rewards aligned with structural or symbolic properties of the solution are particularly effective for improving equation solving performance.

Table 7 demonstrates the results for the general Qwen2.5-1.5B in terms of correctness quality metric. Overall, steering significantly improves the symbolic correctness of generated solutions compared to the baseline model across most equation types. The improvements are particularly pronounced for homogeneous second-order equations and polynomial differential equations. In these cases, the correctness score increases from 0.0875 to 0.6125 and from 0.6738 to 0.9850, respectively, when appropriate reward functions are used.

Table 7: Effect of activation steering with different reward functions on the symbolic correctness between generated and reference differential equation solutions for the Qwen2.5-1.5B model when steering is applied across all transformer layers.

Equation Type	Baseline	R_{BLEU}	R_{coin}	R_{correct}
hom. 2	0.088	0.022	0.607	**0.612**
hom. 3	0.033	0.0	**0.303**	0.299
inhom. 2	0.052	0.125	**0.170**	0.117
inhom. 3	**0.026**	0.0	0.0	0.001
polynomial	0.674	0.850	**0.985**	0.964
first-order	0.423	0.651	0.821	**0.876**

Note: Boldface indicates the highest value for each equation type.

Overall, these results indicate that steering can significantly enhance the quality and symbolic correctness of solutions generated by general-purpose models.

Table 8 presents BLEU scores for generated solutions obtained with the general-purpose Qwen2.5-1.5B model under different steering configurations and reward functions. The results show that steering improves the quality of generated solutions compared to the baseline model for most equation types. The BLEU-based reward R_{BLEU} leads to the largest improvements in terms of BLEU score. For general model we also obtain that steering applied across all transformer layers achieves higher BLEU scores than steering restricted to the last five layers.

Table 8: Comparison of BLEU scores for differential equation solutions generated by the Qwen2.5-1.5B model under different activation steering settings and reward functions across multiple equation types.

Equation Type	Baseline	R_{BLEU} (all)	R_{BLEU} (last 5)	R_{correct} (all)	R_{correct} (last 5)
homo2	0.188	0.534	0.198	**0.677**	0.407
homo3	0.181	**0.527**	0.203	0.453	0.199
inhomo2	0.216	**0.312**	0.165	0.244	0.158
inhomo3	0.059	0.0	0.0	0.004	**0.064**
poly	0.25	**0.587**	0.286	0.277	0.282
first_order	0.169	**0.239**	0.141	0.200	0.147

Note: Boldface indicates the highest value for each equation type.

Let us note that, the results of steering for the general-purpose model remain lower than those obtained with the specialized mathematical model for complex types of equations such as second- and third-order inhomogeneous equations in terms of all evaluation metrics.

Before summarizing the results, we briefly note a limitation of the evaluation procedure related to symbolic verification. In some cases, values reported as 0.0 may in fact correspond to small non-zero scores. This is due to limitations of symbolic verification in SymPy: for certain generated expressions, particularly long solutions containing repeated symbolic patterns, the simplification and equivalence checking procedures may fail to terminate within a reasonable time. In such cases, the metric is recorded as zero. However, empirical inspection shows that these instances correspond to solutions with very low structural or symbolic similarity to the reference, and therefore the true metric values are expected to be close to zero.

Overall, the experimental results demonstrate that activation steering combined with reinforcement learning consistently improves the quality and symbolic correctness of solutions to differential equations across all considered models. As expected, the specialized mathematical models exhibit stronger baseline performance in terms of symbolic equivalence and correctness across most equation types with respect to the general model. However, activation steering substantially improves the performance of all three models. In particular, for structurally simpler equation classes such as homogeneous and polynomial equations, the steered general model achieves the results that are comparable to, and in some cases exceed, those of the specialized model math model. At the same time, for more complex equation types, such as second- and third-order inhomogeneous equations, all models still exhibit relatively low performance, although activation steering nevertheless provides measurable improvements over the corresponding baseline models in several configurations.

5 Conclusions

In this work, we investigated the use of activation steering methods to improve the ability of compact language models to solve differential equations. The proposed approach combines activation-level interventions with reinforcement learning, allowing steering parameters to be optimized using task-specific reward

functions without modifying the original model weights. This framework enables targeted control of internal model representations during the generation process.

To evaluate the robustness of the proposed method, experiments were conducted on three types of models: a general-purpose language model, a mathematically specialized model, and an instruction-tuned mathematical model. Evaluating the method across these architectures allows us to demonstrate the generalizability of the proposed approach under different levels of domain specialization.

The experimental results reveal several important observations. First, steering applied across all transformer layers consistently outperforms steering restricted to the final layers.

Second, the magnitude of improvement varies across equation types. The largest gains are observed for homogeneous and polynomial differential equations, where steering substantially improves both the quality and symbolic correctness of generated solutions. Third, higher-order inhomogeneous equations remain the most challenging category for all considered models. Although measurable improvements are still observed for several settings, the achieved performance remains substantially lower than for structurally simpler equation categories.

Forth, among the evaluated models, the Math-Instruct variant combined with activation steering achieves the strongest overall performance across most equation types, particularly for homogeneous, polynomial, and first-order equations.

Fifth, after applying activation steering, the general-purpose model can in some cases achieve correctness scores comparable to or even exceeding those of the specialized mathematical model, despite having lower baseline performance. One possible interpretation is that the internal representations of the general-purpose model may be more amenable to activation-level interventions than those of the specialized mathematical model. In contrast, the representations learned by the mathematically specialized model may be more strongly adapted to its original training objective, potentially making them less sensitive to relatively small steering perturbations. However, a more detailed analysis of how activation steering affects internal model representations is left for future work.

For the instruction-tuned model, an additional effect may arise from alignment training, which encourages adherence to solution structure and output conventions. In this setting, activation steering may further improve adherence to solution structure and output conventions, leading to particularly strong performance on structurally simpler equation types.

Another important observation is that reward functions based on symbolic structure consistently outperform the BLEU-based reward. In particular, both R_{coin} and R_{correct} typically lead to better results than R_{BLEU} across models and equation types. This suggests that surface-level sequence similarity is less informative for differential equation solving than rewards that explicitly capture symbolic relationships between the generated and reference expressions.

At the same time, the comparison between R_{coin} and R_{correct} is more nuanced. Since R_{correct} is defined not only through exact symbolic equivalence but also through a partial coincidence-based component, it can be viewed as a hybrid symbolic reward rather than a purely sparse binary signal. As a result, R_{coin} and R_{correct} often achieve comparable performance, with neither reward uniformly dominating across all equation types. This indicates that an effective reward for steering optimization should balance strict mathematical correctness with sufficiently dense intermediate feedback.

It is important to note that the experiments in this study were conducted using relatively small-scale language models. The goal of this design was to investigate whether activation steering combined with reinforcement learning can improve the performance of compact models on differential equation solving tasks. The results suggest that the proposed approach can significantly improve the quality and symbolic

correctness of solutions generated by smaller models, allowing them to approach the performance of more specialized architectures on several classes of differential equations.

At the same time, the current implementation relies on predefined equation-type labels for steering expert selection and does not include a dynamic routing mechanism. This simplified setup was intentionally adopted in order to isolate and evaluate the effect of equation-type-specific activation steering under controlled experimental conditions. In addition, the current study is limited to a specific set of synthetically generated differential equations and does not evaluate the proposed framework on broader classes of equations, noisy symbolic expressions, or real-world equations. Extending the framework with automatic expert selection or dynamic routing strategies, as well as evaluating the approach on more diverse mathematical problems, therefore represents important directions for future work and would improve the practicality and generalizability of the proposed approach.

In addition, future work may explore more sophisticated reward formulations, adaptive steering strategies, and systematic sensitivity analysis of reinforcement learning and generation hyperparameters. Another important direction is the application of the proposed framework to broader classes of symbolic mathematical problems.

Acknowledgement: The research made use of computational resources provided by the university's HPC facilities.

Funding Statement: The study was implemented in the framework of the Basic Research Program at HSE University (HSE-BR-2025-68).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Sergei Koltcov and Anton Surkov; methodology, Anton Surkov and Sergei Koltcov; software, Anton Surkov; validation, Vera Ignatenko and Anton Surkov; formal analysis, Vera Ignatenko and Anton Surkov; investigation, Anton Surkov and Vera Ignatenko; resources, Sergei Koltcov; data curation, Anton Surkov; writing—original draft preparation, Vera Ignatenko; writing—review and editing, Anton Surkov and Sergei Koltcov; visualization, Vera Ignatenko; supervision, Sergei Koltcov; project administration, Vera Ignatenko; funding acquisition, Sergei Koltcov. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data and source codes that support the findings of this study are openly available in GitHub at https://github.com/hse-scila/MathSteering.

Ethics Approval: Not applicable.

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

References

1. Wang PY, Liu TS, Wang C, Li Z, Wang Y, Yan S, et al. A survey on large language models for mathematical reasoning. ACM Comput Surv. 2026;58(8):209. doi:10.1145/3786333.
2. Turner AM, Thiergart L, Leech G, Udell D, Vazquez JJ, Mini U, et al. Steering language models with activation engineering. arXiv:2308.10248. 2024.
3. Zou A, Phan L, Chen S, Campbell J, Guo P, Ren R, et al. Representation engineering: a top-down approach to AI transparency. arXiv:2310.01405. 2025.
4. Bartoszcze L, Munshi S, Sukidi B, Yen J, Yang Z, Williams-King D, et al. Representation engineering for large-language models: survey and research challenges. arXiv:2502.17601. 2025.
5. Wehner J, Abdelnabi S, Tan D, Krueger D, Fritz M. Taxonomy, opportunities, and challenges of representation engineering for large language models. arXiv:2502.19649. 2025.

6. Rimsky N, Gabrieli N, Schulz J, Tong M, Hubinger E, Turner A. Steering Llama 2 via contrastive activation addition. In: Ku LW, Martins A, Srikumar V, editors. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA, USA: ACL; 2024. p. 15504–22.
7. Højer B, Jarvis OS, Heinrich S. Improving reasoning performance in large language models via representation engineering. arXiv:2504.19483. 2025.
8. Venhoff C, Arcuschin I, Torr P, Conmy A, Nanda N. Understanding reasoning in thinking language models via steering vectors. arXiv:2506.18167. 2025.
9. He Z, Jin M, Shen B, Payani A, Zhang Y, Du M. SAE-SSV: supervised steering in sparse representation spaces for reliable control of language models. In: Christodoulopoulos C, Chakraborty T, Rose C, Peng V, editors. Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA, USA: ACL; 2025. p. 2207–36.
10. Kang X, Shi D, Chen L. Model whisper: steering vectors unlock large language models' potential in test-time. arXiv:212.04748. 2025.
11. Radevski G, Gashteovski K, Hong G, Lawrence C, Glavaš G. Compositional steering of large language models with steering tokens. arXiv:2601.05062. 2026.
12. Anand N, Somasundaram S, Phukan A, Saxena A, Mukherjee K. ContextFocus: activation steering for contextual faithfulness in large language models. arXiv:601.04131. 2026.
13. Venkatesh S, Kurapath AM. On the non-identifiability of steering vectors in large language models. arXiv:2602.06801. 2026.
14. Sinii V, Gorbatovski A, Cherepanov A, Shaposhnikov B, Balagansky N, Gavrilov D. Steering LLM reasoning through bias-only adaptation. In: Christodoulopoulos C, Chakraborty T, Rose C, Peng V, editors. Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA, USA: ACL; 2025. p. 9202–11.
15. Ferrao J, van der Lende M, Lichkovski I, Neo C. The anatomy of alignment: decomposing preference optimization by steering sparse features. arXiv:2509.12934. 2025.
16. Chai T, Mitra C, Huang B, Gare GR, Lin Z, Arbelle A, et al. Activation reward models for few-shot model alignment. arXiv:2507.01368. 2025.
17. Sinii V, Balagansky N, Aksenov Y, Kurochkin V, Laptev D, Gorbatovski A, et al. Small vectors, big effects: a mechanistic study of RL-induced reasoning via steering vectors. arXiv:2509.06608. 2025.
18. Nguyen T, Le T. ATLAS: adaptive test-time latent steering with external verifiers for enhancing LLMs reasoning. arXiv:2601.03093. 2026.
19. Kirtania S, Iyer A. Steering LLMs for formal theorem proving. arXiv:2502.15507. 2025.
20. Yang A, Zhang B, Hui B, Gao B, Yu B, Li C, et al. Qwen2.5-math technical report: toward mathematical expert model via self-improvement. arXiv:2409.12122. 2024.
21. Qwen2.5-Math-1.5B-Instruct. [cited 2026 Apr 3]. Available from: https://huggingface.co/Qwen/Qwen2.5-Math-1.5B-Instruct.
22. Team Q. Qwen2.5: a party of foundation models. 2024 [cited 2026 Apr 3]. Available from: https://qwenlm.github.io/blog/qwen2.5/.
23. Yang A, Yang B, Hui B, Zheng B, Yu B, Zhou C, et al. Qwen2 technical report. arXiv:2407.10671. 2024.
24. Shao Z, Wang P, Zhu Q, Xu R, Song J, Bi X, et al. DeepSeekMath: pushing the limits of mathematical reasoning in open language models. arXiv:2402.03300. 2024.
25. GRPO trainer. [cited 2026 Apr 3]. Available from: https://huggingface.co/docs/trl/en/grpo_trainer.
26. Papineni K, Roukos S, Ward T, Zhu WJ. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. ACL '02. Stroudsburg, PA, USA: ACL; 2002. p. 311–8. doi:10.3115/1073083.1073135.