大規模言語モデルにおける多段階推論による数値計算の評価と改善

大政 裕哉　後藤 功雄　二宮 崇
愛媛大学
{omasa@ai.cs., goto.isao.fn@, ninomiya.takashi.mk@}ehime-u.ac.jp

概要

大規模言語モデル（LLM）の多段階推論における基礎的能力を評価するために，数値計算能力を検証する．四則演算タスクを設計して桁数を 1 桁から 6 桁まで拡張したデータセットを構築し，7〜9B 規模の 6 種類の LLM を評価した．その結果，4 桁以上の乗算・除算で多くのモデルの性能が大きく低下することを確認した．Few-shot，Fine-Tuning，JSON 形式を適用することにより一部のモデルでの有効性を確認した．一方，その効果はモデルや演算に依存することが明らかになった．

1　はじめに

大規模言語モデルは，数学オリンピックの問題や高度な数理推論タスク，文章問題において高い性能を示している．一方で，大規模言語モデル（LLM）は，単純な加算や乗算であっても誤った結果を出力する場合があることが報告されている [1, 2]．このような傾向は，高難度の数学タスクと比較して，むしろ基本的な数値計算において顕著であることが示されている [1]．基礎的な数値計算の性能は，単なる計算精度の問題にとどまらず，LLM がアルゴリズム的な推論をどの程度内在的に獲得しているかを測る重要な指標である．LLM の数値計算能力や数値推論能力を体系的に評価する研究が進められており，数値表現の扱い方や計算過程の明示が性能に影響を与えることが報告されている [3, 4]．

本研究では，公開されている LLM を対象に，四則演算タスクを用いた体系的な評価を行い，多段階推論における基礎的な能力を検証する．問題の難度を制御して誤りの傾向を分析するとともに，事例の提示や追加学習の有無が計算精度に与える影響を検証する．その結果，桁数の増加に伴い計算精度が大きく低下し，特に乗算および除算で顕著な性能劣化がみられた．事例の提示や計算過程を含む追加学習，指示形式の工夫は，一部の設定において計算精度の向上に有効であることを確認した．

2　関連研究

LLM の数学能力に関する研究は，文章題や多段階推論を含む数学タスクを中心に発展してきた．代表的なベンチマークである GSM8K[5] は，文章で記述された計算問題に対する能力を評価することができ，LLM は一定の性能を達成できることが示されている [6]．この文章題ベースのデータセットは文脈理解や推論課程の生成を強く要求するため，演算子や桁数といった，基礎的な数値計算能力を厳密に分析することは難しい．

LLM は高度な数学ベンチマークで高性能を達成するものがある一方で，基本的な計算に失敗する場合がある [7]．数値計算能力は自然言語処理能力の延長として部分的に獲得される一方で，厳密な数値演算は難しい [3]．数値処理能力そのものに着目した研究では，LLM の加算と乗算が記号のパターンマッチに依存していることが報告されている [8]．加算と減算を対象にした性能評価も実施されている [9]．桁数や精度といった数値表現の制約が算術推論性能に影響を与えることが報告されている [4]．

本研究は，四則演算の加算，減算，乗算，除算のすべてに対して数値の桁数を制御して難易度ごとに多段階推論に対する LLM の能力を比較可能な形で検証する点に特徴がある．

3　タスク設計

本研究では，先行研究の知見を踏まえ，文章理解を伴わない純粋な数値計算タスクに焦点を当てる．特に，同一の数値ペアに対して演算子のみを変更するというタスク設計を採用することで，加算・減算・乗算・除算の違いを直接比較可能な形で評価す

る．このような設定により，既存の文章題ベースの評価では捉えにくかった，LLM の基礎的な数値計算能力とその誤り傾向を詳細に分析できるようにすることを目指す．

3.1 データセットの構築

LLM の基礎的な数値計算能力を評価するために，新たに数値計算データセットを構築した．本データセットは，二つの整数からなる数値ペアに対して加算，減算，乗算，除算のうち 1 つの演算を行う単純な算術タスクから構成されており，文章理解や問題解釈を必要としない点に特徴がある．

データセットにおける数値は，同じ桁数同士の計算を対象として 1 桁，2 桁，4 桁，6 桁の 4 種類を設定した．例えば，2 桁の場合には「2 桁 + 2 桁」，「2 桁 − 2 桁」といった形式の計算のみを含めている．これにより，桁数の違いが計算精度に与える影響を明確に分析できるようにしている．

また，比較可能性を確保するため，各データでは数値Ａと数値Ｂをランダムに選択したうえで，その数値ペアを固定し，演算子のみを変更する設計を採用した．具体的には，同一の数値ペアに対して，加算，減算，乗算，除算の 4 種類の演算をそれぞれ適用する．この設計により，演算子の違いによる性能差を，数値そのものの影響から切り離して評価することが可能となる．

以下に，数値ペアが 89 と 91 の場合の例を示す．

- $89 + 91 = 180$
- $89 - 91 = -2$
- $89 \times 91 = 8099$
- $89 \div 91 = 0.978021978021978$

このように，本データセット（NumPair）では数値ペアを共有した 4 種類の演算結果を含めることで，LLM が演算子ごとにどのような誤り傾向を示すかを詳細に分析できる構成とする．テストデータを 1000 件，Fine-Tuning のデータを学習用 9,000 件，検証用 1,000 件構築した．

3.2 評価指標

評価指標として，モデルが出力した最終的な数値が正解と一致しているかどうかに基づく正答率を用いる．計算過程の内容や記述の妥当性は評価指標とせず，最終的な数値結果のみを評価する．

除算結果の評価では，生成結果および正解値を浮動小数点数に変換し，明示的な丸め処理を行わずに比較する．さらに，生成結果と正解値で小数点以下の桁数が異なる場合には，生成結果の小数点以下桁数に合わせて正解値を切り捨てた数値と比較する．この評価手順は，除算において各桁が余りに基づいて逐次的に決定されるという筆算の性質を踏まえたものである．各小数桁は独立した離散的判断の結果であるため，最終桁のみを四捨五入で補正する評価は筆算と整合しない．この評価手順により，小数点以下の出力精度の違いによる過度な不一致を防ぎつつ，数値的な整合性を重視した判定を行う．

4 ベースラインと改善手法

本節では，ベースラインおよび改善手法について説明する．

本研究では，このデータセットを用いて，複数の LLM に対する数値計算性能の評価を行う．さらに，プロンプトの違いや追加学習の有無が計算精度に与える影響についても検証する．

4.1 ベースライン

本研究では，ベースラインとして Zero-shot 推論に基づく Chain-of-Thought（CoT）形式を用いる．プロンプトを付録 A 図 1 に示す．また，フォーマットとして自由形式に近いマークダウン形式を用いる．

4.2 改善手法

Few shot ベースラインと異なり，計算手順を明示的かつ詳細に記述した例をプロンプト中に与えることで，モデルが各桁ごとの演算や中間結果を段階的に出力するように誘導する．

Fine-Tuning Fine-Tuning に用いた学習データは，数値計算の問題に加え，人間が行う筆算手順を自然言語で逐次的に記述した計算過程を含む形式で構築する．各データでは，被演算数と演算子を明示し，部分計算，桁移動，繰り上がり（繰り下がり），および余りの更新を段階的に記述する．

乗算では，乗数を一の位から順に処理し，各桁の部分積と繰り上がりを記述した後，桁位置に応じてシフトした部分積を加算する手順を示す．除算では，長除法に基づき，桁を順に下ろしながら商を求め，小数 5 桁まで計算を継続する過程を記述する．

Fine-Tuning では計算過程を明示的に含むデータを用いることで，モデルに計算の過程を学習させる．Fine-Tuning にはパラメータ効率の高い手法である

表 1　使用モデル

モデル名	サイズ
MetaMath-7Llemma-7B (MetaMath)[1)][12]	7B
Qwen2.5-Math-7B-Instruct (Qwen2.5-Math)[2)][13]	7B
Qwen2.5-7B-Instruct (Qwen2.5)[3)][14]	7B
DeepSeek-R1-Distill-Qwen-7B (DeepSeek)[4)][15]	7B
Llama-3.1-8B-Instruct (Llama-3.1)[5)][16]	8B
GLM-Z1-9B-0414 (GLM-Z1)[6)][17]	9B

QLoRA[10] 用いる．

JSON フォーマット　同一内容であっても，プロンプトの書式差が LLM の性能に大きく影響することが示されている [11]．モデルに計算過程と最終結果の生成を指示する際に，出力の構造化を促す JSON 形式の指示を用いる．

5　評価実験

5.1　実験設定

本研究では，モデルおよびソフトウェアが公開されている 6 種類の LLM を対象とし，基礎的な数値計算能力の差異を比較した．使用したモデルを表 1 に示す．モデル間を比較可能にするため，すべてのモデルに対して同一のデータセットおよび評価手順を適用する．

推論時には温度パラメータを 0 に設定し，サンプリングを行わない決定論的生成を用いる．すべての推論はモデル単体で実行し，外部の計算ツールは使用しない．計算過程を含む出力が途中で打ち切られないよう，最大生成トークン数は 2,058 とし，複数問題を効率的に処理するためバッチ推論を用いる．

Few-shot 推論および Fine-Tuning は，Zero-shot 条件において最も高い性能を示したモデルのみを対象として実施した．これは，基礎的な計算能力を有するモデルに改善手法を適用することで，その効果をより明確に評価するためである．Fine-Tuning の学習率 1e-4，エポック数 3，バッチサイズ 1，勾配蓄積ステップ数 8（実効バッチサイズ 8）とした．

5.2　結果

本節では，加算 (add)，減算 (sub)，乗算 (mul)，除算 (div) の各演算について，桁数，演算種別，モデル間の性能差，および Few-shot，Fine-Tuning，JSON 形式の効果を検証する．表 2 〜 6 に，各設定における正答率を示す．

表 2　1 桁，Zero-shot の演算結果（%）

モデル名	add	sub	mul	div
MetaMath	100.00	100.00	100.00	63.33
Qwen2.5-Math	100.00	100.00	100.00	84.44
Qwen2.5	100.00	100.00	100.00	84.44
DeepSeek	100.00	98.00	90.00	60.00
Llama-3.1	100.00	100.00	100.00	88.89
GLM-Z1	100.00	100.00	100.00	61.78

表 3　2 桁，Zero-shot の演算結果（%）

モデル名	add	sub	mul	div
MetaMath	95.50	62.40	50.60	23.30
Qwen2.5-Math	91.70	84.80	87.50	61.70
Qwen2.5	99.80	99.90	93.40	79.00
DeepSeek	94.40	92.80	96.20	45.90
Llama-3.1	97.50	85.20	98.70	84.30
GLM-Z1	98.10	98.10	97.70	54.50

表 4　4 桁，Zero-shot の演算結果（%）

モデル名	add	sub	mul	div
MetaMath	27.60	51.80	0.10	9.40
Qwen2.5-Math	96.40	96.00	20.70	1.80
Qwen2.5	98.30	96.30	4.50	13.00
DeepSeek	56.00	85.50	12.50	53.70
Llama-3.1	36.50	53.40	14.30	56.60
GLM-Z1	94.90	88.30	68.10	18.50

表 5　6 桁，Zero-shot の演算結果（%）

モデル名	add	sub	mul	div
MetaMath	5.10	5.90	0.00	7.60
Qwen2.5-Math	29.90	90.50	0.20	0.50
Qwen2.5	97.00	96.00	0.00	13.50
DeepSeek	64.40	88.20	12.70	24.00
Llama-3.1	29.90	24.00	0.00	42.30
GLM-Z1	86.80	87.60	49.40	21.00

表 6　4 桁，Few-shot，Fine-Tuning の演算結果（%）

モデル名	mul	div
GLM-Z1 (ベースライン)	68.10	-
Few-shot	89.90	-
Few-shot+JSON	90.20	-
Fine-Tuning	57.50	-
Llama-3.1 (ベースライン)	-	56.60
Few-shot	-	0.00
Few-shot+JSON	-	0.10
Fine-Tuning	-	91.50

桁数の影響　Zero-shot 設定では，すべてのモデルにおいて桁数の増加に伴う正答率の低下が確認された．1 桁および 2 桁では多くのモデルが 90%以上と高い正答率を示したが，4 桁以降で性能低下が顕著となり，6 桁では大きな劣化が見られた．これは，桁数の増加により CoT で生成される部分計算が増え，各段階の誤りが最終結果に累積しやすくなるためと考えられる．

演算の種類の影響　演算種別では，加算および減算が最も安定した性能を示し，桁数が増加しても比較的高い正答率を維持した．一方，乗算及び除算では桁数増加に伴う性能低下が顕著であり，特に乗算は 4 桁以降で大きく低下した．除算は 1 桁からほかの演算より難易度が高い傾向確認された．

モデル間比較　同一条件下でもモデル間の性能差は大きく，4 桁および 6 桁の加算・減算では一部モデルが 90%以上と高い正答率を維持した．演算子ごとの比較では，4 桁乗算では GLM-Z1，4 桁除算では Llama-3.1 が相対的に高い性能を示し，モデル事に得手・不得手な演算が異なることが明らかになった．

Few-shot の効果　Zero-shot で高い性能を示したモデルに Few-shot を適用した結果，4 桁乗算では GLM-Z1 の正答率が 68.10%から 89.90%へと大きく向上した．一方，4 桁除算における Llama-3.1 では正答率が 0.00%となり，Few-shot の効果はモデルに強く依存することが示された．

Fine-Tuning の効果　Fine-Tuning の効果は演算によって異なった．4 桁乗算では GLM-Z1 の正答率が 57.50%と Zero-shot の 68.10%を下回った．一方，4 桁除算では Llama-3.1 の正答率が 56.60%から 91.50%へと大幅に向上し，計算手順が明示的な演算では学習による計算過程の獲得が有効である可能性が示された．

JSON 形式の効果　Few-shot に JSON 形式を導入した場合，乗算では正答率がわずかに向上したが，除算では改善は見られなかった．この結果から，JSON 形式の有効性も演算およびモデルに依存することが示唆される．

5.3　誤り分析

モデルの違いにより，誤りの出方には性質の異なる傾向が確認された．主な誤りは，途中計算の破綻，部分的に正しい中間結果を含む誤答，桁の取り違えなどであり，その傾向はモデル事に異なっていた．以下では，4 桁乗算タスクと 4 桁除算タスクに着目し，具体的な誤り事例を通じてモデルごとの誤り傾向を分析する．

4 桁乗算タスク　GLM-Z1 の 4 桁乗算では，計算アルゴリズム自体は正しいものの，部分積の桁移動や中間結果の保持における局所的な誤りが，最終結果に大きな誤差を生じさせる例が確認された．付録 A 表 7 に例を示す．このような誤りは，大きな数値を構成する途中段階で生じた小さな数値のずれが後続の加算過程によってそのまま伝播・固定される点が特徴的である．また，一部では自己修正が機能する例（付録 A 表 8）が見られたが，自己修正は常に有効に機能するわけではなく，誤った数値をそのまま正当化して最終出力としてしまう例（付録 A 表 9）も確認された．このことから，モデルの検証能力は十分とはいえず，誤りを確実に検証・修正する機能には限界があることが示唆される．

4 桁除算タスク　Fine-Tuning を施した Llama-3.1 の 4 桁除算では，誤りは一様ではなく，いくつかの典型的なパターンに分類できることが確認された．

逐次的な手順は概ね妥当だが，途中の乗算や減算におけるミスが後続桁へ累積するケースが見られた（付録 A 表 10）．特に，商の各桁と除数の積における繰り上がり処理の誤りが，最終的な解を正解から乖離させる原因となっている．

これらから，計算手順の欠如ではなく，逐次的な数値更新を長く安定して維持する能力に課題があることが示唆される．

6　おわりに

本研究では，四則演算と桁数を制御した評価タスクを設計し，6 種類の LLM を対象に基礎的な数値計算能力を評価した．その結果，7〜9B 規模の LLM は 4 桁以上の乗算・除算で大きく性能が低下することが明らかになり，6 桁加算でも半数のモデルが 50%未満であるなど，多い桁数の計算に対する脆弱性を確認した．一方で，Few-shot，Fine-Tuning，JSON 形式を適用することで 1 つ以上のモデルで性能が向上することを確認した．

謝辞

本研究は国立研究開発法人情報通信研究機構の委託研究（課題番号：225）および JSPS 科研費 JP24K15071 および大学発新産業創出基金事業スタートアップ・エコシステム共創プログラム JPMJSF2316 の助成を受けたものです.

参考文献

[1] Andrew Gambardella, Yusuke Iwasawa, and Yutaka Matsuo. Language models do hard arithmetic tasks easily and hardly do easy arithmetic tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**, pp. 85–91, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[2] Zheng Zhang. Comprehension without competence: Architectural limits of LLMs in symbolic computation and reasoning. **Transactions on Machine Learning Research**, 2025.

[3] Forrest McKee David Noever. Numeracy from literacy: Data science as an emergent skill from large language models. **arXiv:2301.13382**, 2023.

[4] Guhao Feng, Kai Yang, Yuntian Gu, Xinyue Ai, Shengjie Luo, Jiacheng Sun, Di He, Zhenguo Li, and Liwei Wang. How numerical precision affects arithmetical reasoning capabilities of LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, **Findings of the Association for Computational Linguistics: ACL 2025**, pp. 46–85, Vienna, Austria, July 2025. Association for Computational Linguistics.

[5] Mohammad Bavarian Mark Chen Heewoo Jun Lukasz Kaiser Matthias Plappert Jerry Tworek Jacob Hilton Reiichiro Nakano Christopher Hesse John Schulman Karl Cobbe, Vineet Kosaraju. Training verifiers to solve math word problems. **arXiv:2110.14168**, 2021.

[6] Qihuang Zhong, Kang Wang, Ziyang Xu, Juhua Liu, Liang Ding, and Bo Du. Achieving >97% on GSM8K: Deeply understanding the problems makes LLMs better solvers for math word problems, 2025.

[7] Yang Yan, Yu Lu, Renjun Xu, and Zhenzhong Lan. Do large language models truly grasp addition? A rule-focused diagnostic using two-integer arithmetic. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, **Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing**, pp. 13467–13483, Suzhou, China, November 2025. Association for Computational Linguistics.

[8] Roy Xie Ruidi Chang Hanjie Chen Chunyuan Deng, Zhiqi Li. Language models are symbolic learners in arithmetic. **arXiv:2410.15580**, 2024.

[9] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of the transformers with simple arithmetic tasks. **CoRR**, Vol. abs/2102.13019, , 2021.

[10] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, **Advances in Neural Information Processing Systems**, Vol. 36, pp. 10088–10115. Curran Associates, Inc., 2023.

[11] David Koleczek Arshdeep Sekhon Franklin X Wang Sadid Hasan Jia He, Mukund Rungta. Does prompt formatting have any impact on LLM performance? **arXiv:2411.10541**, 2024.

[12] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models, 2024.

[13] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024.

[14] An Yang et al. Qwen2.5 technical report, 2025.

[15] Daya et al. Guo. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. **Nature**, Vol. 645, No. 8081, 2024.

[16] Aaron Grattafiori et al. The llama 3 herd of models, 2024.

[17] Team GLM. ChatGLM: A family of large language models from GLM-130B to GLM-4 all tools, 2024.

A　付録

```
”Solve the following arithmetic problem.\n\n”
”Follow these rules:\n”
”- Show only the minimal necessary reasoning.\n”
”- Each step must be a simple numerical operation.\n”
”- Do not invent steps that are not needed.\n”
”- Keep reasoning short and purely arithmetic.\n”
”- After the steps,  output the final answer on the last line as:\n”
”  Final Answer: <number>\n\n”
”Problem: {question}\n\n”
”Solution:\n”
```

図 1　ベースライン CoT プロンプト

表 7　4 桁乗算（4924×3）における部分積計算の誤り例

桁	計算内容	正しい計算	モデルの出力	備考
1 の位	4×3	12	12	正しい
10 の位	$2 \times 3 + 1$	7	7	正しい
100 の位	9×3	27	27	正しい
1000 の位	$4 \times 3 + 2$	14	14	正しい
全体	–	14772	**14767**	**桁配置の誤り**

表 8　4 桁乗算（4312×2784）における自己修正成功の例

段階	推論内容	出力数値	判定
1	部分積を計算し加算を実行	11004608	誤り
2	別の加算順序で再計算を試行	12004608	不一致を検出
3	加算箇所の誤りに気づく（桁落ち）	–	誤り特定
4	正しい加算を再実行	**12004608**	**自己修正成功**
5	標準的な筆算形式で再検証	12004608	正答を確信

表 9　4 桁乗算の部分積（4924×3）における自己修正失敗の例

段階	推論内容	出力数値	判定
1	初回計算を実行	14767	誤り
2	「Wait,　let me check again」と再計算を宣言	–	自己修正開始
3	同一の計算手順を再実行	**14767**	**同一誤答**
4	「So yes」と結論付け	14767	誤りを確信

表 10　算術操作ミスにおける誤りの累積（$8769 \div 8845$）

段階	推論内容	出力数値	判定
1	小数第一位の商を 9 と推定	9	妥当
2	除算の積（9×8845）を算出	**79505**	**誤り**（正：79605）
3	誤った積を用いて余りを算出（87690 – 79505）	8185	派生エラー
4	誤った余りに基づき次の商を 9 と推定	9	妥当（手順上）
5	最終回答を構成	0.99265	**誤答**（正：0.99140...）