ベンチマークデータセット ControlBench を用いた制御工学用カスタム GPT の性能評価

九州工業大学　○尾郷樹　古賀雅伸

Performance evaluation of a custom GPT for control engineering using the benchmark dataset ControlBench

Abstract : In this paper, we evaluate GPT-4o using ControlBench, a benchmark dataset for control engineering. Additionally, we proposed a method to improve response accuracy by developing a customized GPT that leverages prompt engineering methods, and we also evaluate this customized GPT.

1　はじめに

近年、生成 AI（Generative AI）技術は画像生成や自然言語処理の分野で目覚ましい発展を遂げている。そのため、LLM(Large Language Model) が制御理論においてどのような役割を果たせるのかは非常に興味深い。LLM を制御系設計プロセスに組み込むことで、設計の各段階を大幅に効率化できる可能性がある。

筆者らの先行研究において、大学学部レベルの古典制御理論の演習問題に対して、GPT-4 の能力評価を行った [1]。基本的な演習問題 97 問に対して、GPT-4 は約 66% の正答率を示した。また、[2] では、大学学部レベルの現代制御理論の演習問題に対して、GPT-4o の能力評価および回答精度を向上させる手法を提案した。さらに、[3] では、GPT-4、Claude 3 Opus、Gemini 1.0　Ultra などの主要な LLM が制御工学に対してどれほどの能力を示すのか、制御工学に関する 147 問の問題セットである ControlBench を用いたベンチマーク評価を行っている。ベンチマーク評価の結果、GPT-4 は 45.6%の正答率を示している。しかしながら、GPT モデルの評価は GPT-4 のみに留まっており、より新しい GPT モデルに対する評価は行われていない。さらに、GPT モデルの回答精度を向上させる手法は提案されていない。

そこで、本論文では、GPT-4 に比べ数学や推論能力に優れた GPT-4o に対するベンチマーク評価を行う。また、[1]、[2] で提案した回答精度向上手法を ControlBench に対して適用し、評価・考察を行う。

2　提案手法

2.1　GPTs

GPT モデルをはじめとした LLM は、一般に、ユーザーが入力 (プロンプト) を提供し、モデルは入力に基づいて出力を生成する。そのため、モデルから望ましい出力を得るためにプロンプトを最適化する手法である、プロンプトエンジニアリングに関する研究が注目されるようになった [4]。

プロンプトエンジニアリングは、大規模データセットを必要とせずに、特定のタスクへの適応を可能にする手法である。追加のデータやモデルの再学習なしに、プロンプトの工夫だけで望ましい応答を引き出すことが可能となる。これにより、データやコストが限られた状況でも効果的に LLM の性能を向上させることが期待される。

OpenAI の提供する GPTBuilder では、GPTs と呼ばれるカスタマイズされた ChatGPT を作成できる。主に「Instructions」、「Knowledge」、「Capacities」、「Actions」の 4 つのパラメータを編集することで、特定のタスクに適した GPT を構築することが可能である。GPTBuilder で編集可能なパラメータの概要を表 1 に示す。

表 1: Parameters of GPTs

Parameter	Function
Instructions	Prompts given in advance to GPT
Knowledge	Utilization of data
Capabilities	Whether to use Code Interpreter
Actions	Use of third party APIs

本研究では、Instructions および Capabilities(Code Interpreter) を用いる。Instructions に与えた指示は以下の通りである。

Instructions

- You are given a control engineering problem.
- Be sure to answer the given question according to the following constraints.

Constraints

- Always use ”Code Interpreter” for numerical and symbolic calculations.
- Always use Code Interpreter to draw graphs.

2.2 Code Interpreter

Code Interpreter は、ChatGPT に統合されたコード実行のサンドボックス環境である。ユーザーの入力に基づいてプログラムコードを生成・実行し、その結果をリアルタイムで提供することができる。Python を使用したデータ処理や分析、可視化のほか、行列演算などの数学的な数値計算やシミュレーションも可能である。数値計算ライブラリである NumPy や数式処理ライブリの SymPy、数学・科学・工学分野のための数値解析ライブラリの SciPy など、多くの Python ライブラリに対応している。

また、LLM は一般的なテキストを生成する性能と比較して、コードを効果的に生成する際に顕著な能力を示すことが知られている。特に計算エラーに関連する問題は、Program of Thoughts（PoT）を用いることで軽減可能である [5]。例えば化学の問題において、LLM の回答出力を Python コードに変換することで、計算エラーを大幅に減少させることが示されている [6]。また、予め具体的な Python コードを与え、必要に応じて実行させることで、GPT モデルの欠点とも言える回答のランダム性を軽減させることができる [1]。Code Interpreter の実行時の流れを図 1 に示す。

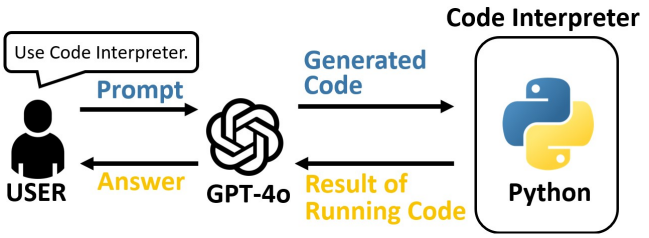

図 1: Flow to use Code Interpreter

3 GPT モデルの性能評価

3.1 ControlBench

制御工学用ベンチマークデータセットである ControlBench は、ミシガン大学（EECS 460）とイリノイ大学アーバナ・シャンペーン校（ECE 486）の制御工学に関する講義から集めた問題から構成されている [3]。また、マルチモーダル化が進む LLM の性能を多面的に評価するために、テキスト形式と画像形式の要素が融合されている。各トピックの総問題数および視覚的要素を含む問題数を表 2 に示す。

3.2 実験手順

GPT モデルへの入力は、ControlBench の TeX ファイルからコピーしたものである。そうして得られた GPT モデルの回答と模範解答を照合し、正誤判定を行う。なお、正誤判定についての採点基準は、以下の通りである。

表 2: Total number of problems for each topic and problems with visual component

Topic	Problems (Visuals)
Background	28 (0)
Stability	19 (0)
Time response	21 (3)
Block diagrams	5 (5)
Control System Design	24 (0)
Bode Analysis	15 (13)
Root-Locus Design	7 (1)
Nyquist Design	5 (4)
Gain/Phase Margins	9 (0)
System Sensitivity Measures	3 (0)
Loop-shaping	4 (0)
Advanced Topics	7 (0)
Total	**147 (26)**

- 答えが完全に正しい場合にのみ得点を与え、それ以外の部分的な正答には点を与えない。
- 小問内に複数の問題が定義されている場合も、全ての問題に正答している場合のみ正解と判定する。
- ChatGPT に問題を与えたのち、追加の指示は与えず、1 度の出力で得られた回答のみで採点する。

また、本論文での評価指標である Accuracy(ACC) は、全問題に対する正答数の割合である。

3.3 実験結果

実験結果を表 3 に示す。作成した問題セットをトピックごとに分類し、それぞれのトピックに対するモデルの正答率 (ACC) をパーセンテージ (%) で表している。ACC の横の括弧内で正答数/総問題数を示している。表 3 より、GPT-4o は学部レベルの制御工学の問題に対して 42.9% の正答率を示し、Code Interpreter を使うよう指示した GPTs では 47.6%の正答率を示し、4.7 ポイントの正答率の向上が見られた。なお、Transformer をモデルアーキテクチャとした多くの LLM は、確率に基づいた予測による回答をする点に注意が必要である [7]。そのため、この結果は制御工学に対する GPT モデルの能力の、ひとつの目安として捉える必要がある。

4 考察

[2] において、GPTs を用いて現代制御理論用の GPTs を作成し、大学院入試問題 45 問での評価の結果、正答率が 26.6%向上し、64.4%の正答率が得られることが確認さ

表 3: Comparison of GPTs, GPT-4o and GPT-4

	GPTs	GPT-4o	GPT-4(出典:[3])
Topics	**ACC**	**ACC**	**ACC**
Background	60.7% (17/28)	53.5% (15/28)	60.7% (17/28)
Stability	73.7% (14/19)	63.2% (12/19)	57.9% (11/19)
Time response	76.2% (16/21)	42.9% (9/21)	57.1% (12/21)
Block diagrams	40.0% (2/5)	20.0% (1/5)	40.0% (2/5)
Control System Design	41.7% (10/24)	33.3% (8/24)	29.2% (7/24)
Bode Analysis	0.0% (0/15)	6.66% (1/15)	6.66% (1/15)
Root-Locus Design	14.3% (1/7)	28.6% (2/7)	28.6% (2/7)
Nyquist Design	0.0% (0/5)	0.0% (0/5)	0.0% (0/5)
Gain/Phase Margins	55.6% (5/9)	55.6% (5/9)	66.7% (6/9)
System Sensitivity Measures	100.0% (3/3)	100.0% (3/3)	100.0% (3/3)
Loop-shaping	0.0% (0/4)	50.0% (2/4)	25.0% (1/4)
Advanced Topics	28.6% (2/7)	71.4% (5/7)	71.4% (5/7)
Total	**47.6% (70/147)**	**42.9% (63/147)**	**45.6% (67/147)**

れている。結果を表 4 に示す。そこで本研究では、数値計算や数式処理における回答精度向上が最も重要であると考え、Instructions に Code Interpreter を使う指示のみを与えた。しかしながら、表 3 より大幅な精度向上は見られなかった。そのため、Code Interpreter を用いる手法だけでなく、Zero-shot CoT Prompting[8]、Few-Shot Prompting[9, 10] など、様々なプロンプトエンジニアリング手法を複合的に組み合わせる必要があると考えられる。

また、[3] では採点基準が明記されていないため、本研究での結果との単純比較はできない点に注意されたい。

5 おわりに

本研究では、制御工学用ベンチマークデータセットである ControlBench を用いた、GPT-4o および制御工学用カスタム GPT の性能評価を行った。今後の課題として、LLM の回答のランダム性に対する検証が求められる。実験結果に対して統計的検定を行い、結果に有意な差が見られるかどうかを判断する必要がある。

参考文献

[1] Itsuki Ogo and Masanobu Koga. Can chatgpt pass classical control theory exam? In *2024 63rd Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 1073–1078, August 2024.

[2] Itsuki Ogo and Masanobu Koga. Can ChatGPT pass modern control theory exam? In *ICCAS 2024, The 24th International Conference on Control, Automation and Systems*, Jeju, Korea, 2024.

[3] Darioush Kevian, Usman Syed, Xingang Guo, Aaron Havens, Geir Dullerud, Peter Seiler, Lianhui Qin, and Bin Hu. Capabilities of large language models in control engineering: A benchmark study on gpt-4, claude 3 opus, and gemini 1.0 ultra. *arXiv preprint arXiv:2404.03647*, 2024.

[4] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9), jan 2023.

[5] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2023.

[6] Siru Ouyang, Zhuosheng Zhang, Bing Yan, Xuan Liu, Yejin Choi, Jiawei Han, and Lianhui Qin. Structured chemistry reasoning with large language models. *arXiv preprint arXiv:2311.09656*, 2024.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

表 4: Comparison of GPT-4o and ModernControlGPT on various topics (出典:[2])

	GPT-4o	**MCGPT**
Topics	**ACC**	**ACC**
State Space Representation	20.0% (1/5)	40.0% (2/5)
LaplaceTransform	16.7% (0.5/3)	33.3% (1/3)
Stability	25.0% (2/8)	50.0% (4/8)
Controllability	50.0% (3/6)	66.7% (4/6)
Observability	66.7% (2/3)	66.7% (2/3)
Coordinate Transformation	16.7% (0.5/3)	33.3% (1/3)
Transfer Function	87.5% (3.5/4)	100.0% (4/4)
Realization	-	-
State Feedback	35.0% (3.5/10)	80.0% (8/10)
Pole placement method	-	-
Optimal regulator	-	-
Servo system	-	-
Observer	33.3% (1/3)	100.0% (3/3)
Total	**37.8% (17/45)**	**64.4% (29/45)**

[8] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.