Science and Technology of Advanced Materials: Methods

ISSN: 2766-0400 (Online) Journal homepage: www.tandfonline.com/journals/tstm20

Exploring utilization of generative AI for research and education in data-driven materials science

Takahiro Misawa, Ai Koizumi, Ryo Tamura & Kazuyoshi Yoshimi

To cite this article: Takahiro Misawa, Ai Koizumi, Ryo Tamura & Kazuyoshi Yoshimi (2025) Exploring utilization of generative AI for research and education in data-driven materials science, Science and Technology of Advanced Materials: Methods, 5:1, 2535956, DOI: 10.1080/27660400.2025.2535956

To link to this article: https://doi.org/10.1080/27660400.2025.2535956

SCIENCE AND TECHNOLOGY OF ADVANCED MATERIALS: METHODS
2025, VOL. 5, NO. 1, 2535956
https://doi.org/10.1080/27660400.2025.2535956

Taylor & Francis
Taylor & Francis Group

FOCUS ISSUE ARTICLE

OPEN ACCESS Check for updates

Exploring utilization of generative AI for research and education in data-driven materials science

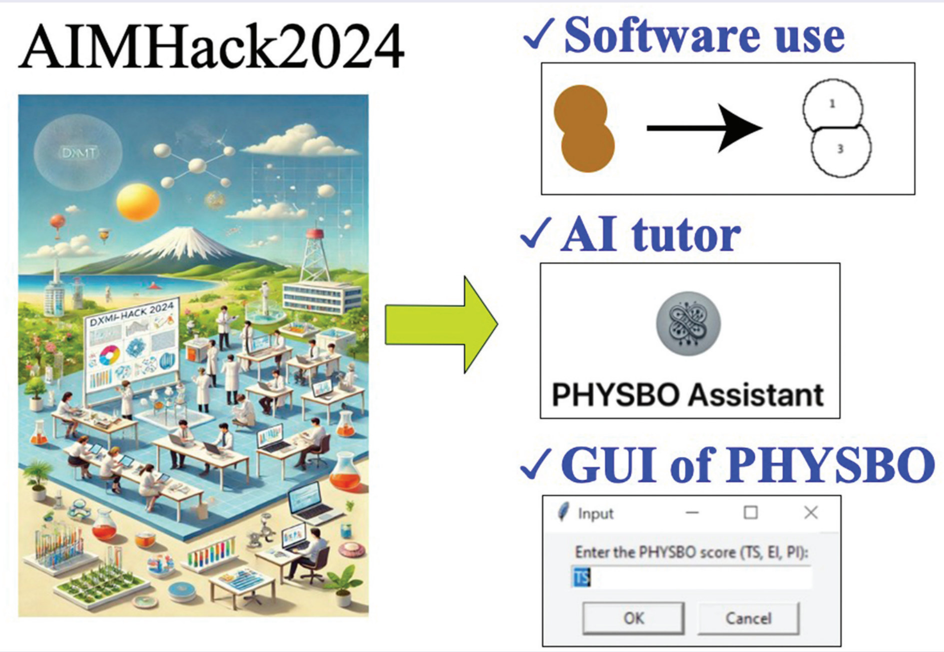
Takahiro Misawa [a]* Ai Koizumi [b], Ryo Tamura [b,c] and Kazuyoshi Yoshimi [a]

[a]Institute for Solid State Physics, The University of Tokyo, Kashiwa, Chiba, Japan; [b]Center for Basic Research on Materials, National Institute for Materials Science, Tsukuba, Japan; [c]Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Japan

ABSTRACT
Generative AI has recently had a profound impact on various fields, including daily life, research, and education. To explore its efficient utilization in data-driven materials science, we organized a hackathon – AIMHack2024—in July 2024. In this hackathon, researchers from fields such as materials science, information science, bioinformatics, and condensed matter physics worked together to explore how generative AI can facilitate research and education. Based on the results of the hackathon, this paper presents topics related to (1) conducting AI-assisted software trials, (2) building AI tutors for software, and (3) developing GUI applications for software. While generative AI continues to evolve rapidly, this paper provides an early record of its application in data-driven materials science and highlights strategies for integrating AI into research and education.

ARTICLE HISTORY
Received 3 April 2025
Revised 25 June 2025
Accepted 14 July 2025

KEYWORDS
Generative AI; data-driven materials science; visualization; GUI; development of software packages



IMPACT STATEMENT
Based on the outcomes of the hackathon (AIMHack2024), this paper highlights novel strategies for AI-assisted software use, building AI tutors, and GUI development. These results demonstrate the transformative potential of generative AI for advancing research and education in data-driven materials science.

1. Introduction

The emergence of generative AI, such as ChatGPT [1], has had a significant impact on various fields, including materials science, where it has been used for tasks such as data analysis, automated literature review, and computational modeling. In July 2024, to explore efficient ways to integrate generative AI into research and

CONTACT Kazuyoshi Yoshimi k-yoshimi@issp.u-tokyo.ac.jp; Takahiro Misawa tmisawa@issp.u-tokyo.ac.jp Institute for Solid State Physics, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Chiba, 277-8581, Japan

*Both authors are contributed equally

Supplemental data for this article can be accessed online at https://doi.org/10.1080/27660400.2025.2535956 education in materials science, specifically in automated data analysis, literature review, and software development, we organized a hackathon, AIMHack2024 [2]. Approximately 30 participants, ranging from students to industry researchers, took part in the hackathon. The participants came from diverse backgrounds, including materials science, computational science, bioinformatics, and condensed matter physics, which fostered interdisciplinary collaboration.

The workshop was designed to accommodate both beginners and advanced users. For beginners, the guided demonstrations introduced fundamental steps for using selected software from a curated list on a portal site such as MatDaCs (MaterialsDataCommons) [3]. Through MatDaCs, participants could access and use various computational materials science applications, including machine learning frameworks. Participants were encouraged to perform basic calculations with the selected software, gaining practical experience under the supervision of generative AI. Advanced users, on the other hand, were given the flexibility to select and independently explore software of their choice, following a structured workflow similar to that used by beginners. This approach allowed them to investigate more advanced features and customize their usage according to their research interests.

The primary emphasis of the hackathon was to enhance participants' ability to critically evaluate AI-generated content. Participants were encouraged to systematically verify AI-generated responses by cross-referencing them with established literature and their domain knowledge. They also tested the AI-generated code and tutorials by running the recommended software, assessing both functionality and accuracy. This exercise was designed not only to enhance technical proficiency but also to foster critical thinking about the reliability and applicability of AI-generated content.

The collaborative efforts in the hackathon aimed to refine participants' strategies for using AI tools in their research activities, facilitating AI integration into research workflows and practical scientific applications. The key results of the hackathon included the implementation of AI-assisted software trials using ImageJ [4] for automated image analysis, the development of an AI tutor for HYSBO [5], a Bayesian optimization tool, using MyGPTs, and the creation of a graphical user interface (GUI) to enhance the usability of PHYSBO. These results highlight the increasing impact of generative AI in advancing materials science research and education.

We note that these three topics are closely related to the typical workflow in materials science research. First, image recognition of experimental results is a crucial first step for data accumulation, and mastering the use of ImageJ directly contributes to efficient data acquisition. Second, Bayesian optimization is a widely used method for determining the next experimental conditions based on accumulated data, and gaining a deeper understanding of PHYSBO through an AI tutor is important for improving research quality. Third, a GUI is useful for executing Bayesian optimization in a user-friendly manner, and our demonstration of a simple method for GUI development provides practical insights for a broad range of developers. Thus, the three topics addressed in this paper correspond to essential components of the research process in materials science.

This paper is organized as follows. Section 2 presents AI-assisted software trials using ImageJ. Section 3 discusses the development of an AI tutor using MyGPTs for PHYSBO. Section 4 describes the creation of a GUI for PHYSBO with AI support. Finally, Section 5 provides a summary of the findings and discusses future directions for AI in materials science.

2. AI-assisted trial use of software

In this section, we present an example of how ChatGPT can be used to assist in the trial use of software for materials science. During the hackathon, we chose the image analysis software ImageJ [4] from the list of software available on MatDaCs [3] as an example. Since ImageJ can count objects in an image, we decided to explore this feature with it.

As an example, we prepared an image with several circles, as shown in Figure 1(a). We then uploaded this image to ChatGPT and entered the following prompt:

> How can I count the number of circles in this image using ImageJ?

ChatGPT generated the following step-by-step instructions[1]:

(1) Open the Image
- Launch ImageJ.
- Select `File` → `Open...` from the menu and open the target image.

(2) Convert to Grayscale
- Select `Image` → `Type` → `8-bit` from the menu to convert the image to grayscale.

(3) Adjust the Threshold
- Select `Image` → `Adjust` → `Threshold`
- Use the sliders to select only the circles (they should appear in red).- Click "Apply."

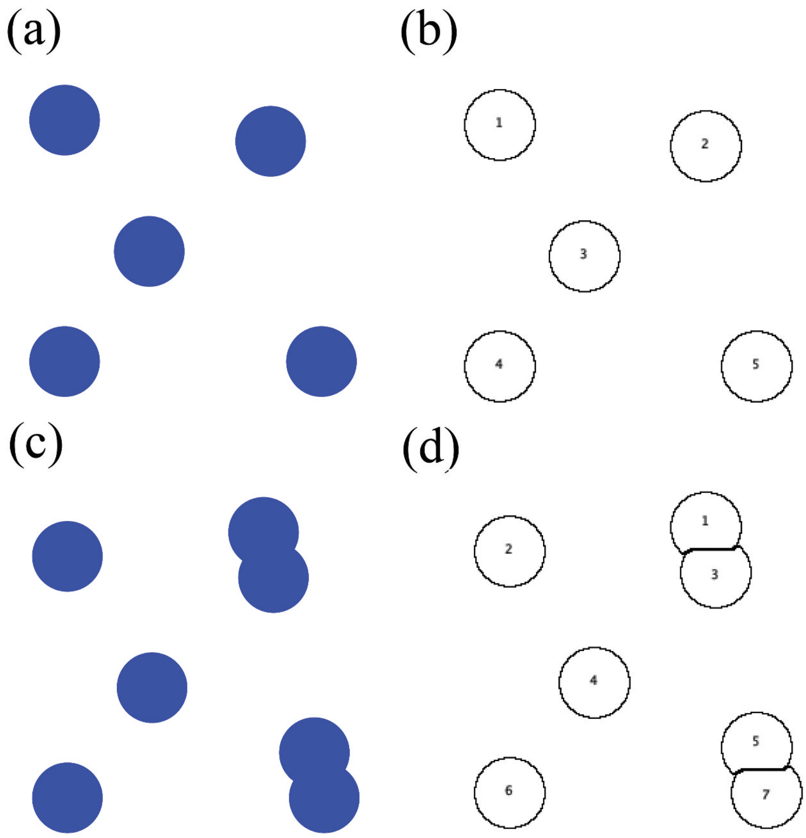


Figure 1. Sample images: (a) circles without overlap and (c) circles with overlap. ImageJ analysis results: (b) counting result for non-overlapping circles and (d) counting result for overlapping circles.

(4) Convert to Binary Image
- Select `Process` → `Binary` → `Make Binary` to create a black-and-white image.

(5) Perform Particle Analysis
- Select `Analyze` → `Analyze Particles...`.
- Set an appropriate "Size" range (e.g., `100-Infinity` to filter out noise).
- Check "`Display results`", "`Show outlines`", and "`Summarize`".
- Click "OK" to perform the analysis.

(6) View the Results
- The results window will display the count of detected circles.

By following these instructions and operating ImageJ, we were able to count the circles, as shown in Figure 1(b). Typically, when using such software, one begins by reading manuals and learning how to use it, which can take a considerable amount of time. However, using generative AI in this way, we can significantly reduce the effort required for learning the software, which presents a major advantage. In fact, when we demonstrated this result, many participants were greatly amazed by how convenient generative AI is.

Furthermore, as a more advanced test, we examined whether ImageJ could distinguish overlapping objects. For this test, we prepared an image with several overlapping circles, as shown in Figure 1(c). We then uploaded this image to ChatGPT and entered the following prompt:

> How can I count individual particles when they overlap, as shown in the attached image?

ChatGPT generated the following step-by-step instructions:

(1) Open the Image
- Launch ImageJ.
- Select `File` → `Open...` from the menu and open the target image.

(2) Convert to Grayscale
- Select `Image` → `Type` → `8-bit` from the menu to convert the image to grayscale.

(3) Adjust the Threshold
- Select `Image` → `Adjust` → `Threshold...`.
- Use the sliders to highlight only the circles (they should appear in red).
- Click "Apply."

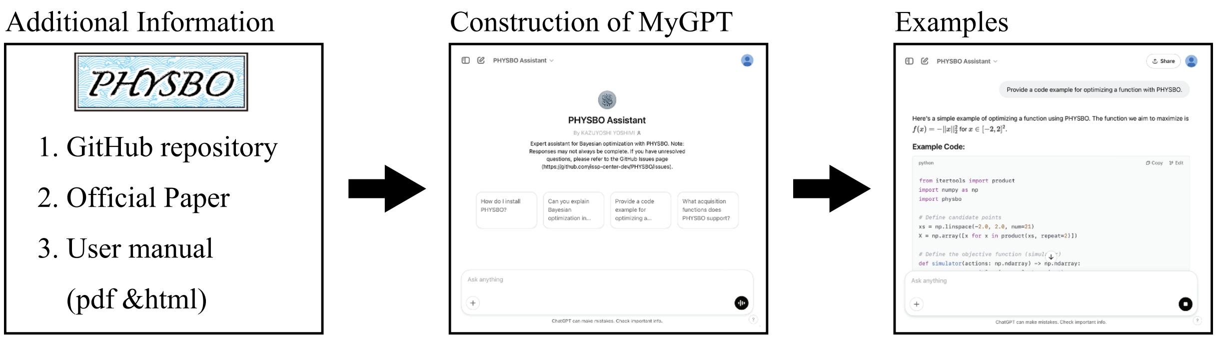


Figure 2. Overview of the process for generating tutorials for PHYSBO using MyGPT. MyGPT was initially tested only with URLs to PHYSBO's official documentation, followed by an improved setup where full documentation PDFs were provided.

(4) Convert to Binary Image
- Select `Process` → `Binary` → `Make Binary` to create a black-and-white image.

(5) Apply Watershed Segmentation
- Select `Process` → `Binary` → `Watershed`.
- This will separate touching circles by adding dividing lines.

(6) Perform Particle Analysis
- Select `Analyze` → `Analyze Particles...`.
- Set an appropriate "Size" range (e.g., `100-Infinity` to filter out noise).
- Check "`Display results`", "`Show outlines`", and "`Summarize`".
- Click "OK" to perform the analysis.

(7) View the Results
- The results window will display the number of detected circles.
- A new image will appear with outlined particles.

From these results, we learned that the *watershed* function can be used to detect overlapping areas. By following these instructions and operating ImageJ, we confirmed that the watershed function successfully separated overlapping regions, allowing for accurate object counting, as shown in Figure 2(d). In a standard approach before the emergence of generative AI, finding such a feature required reading the manual and searching for the relevant section. However, we found that this effort could be significantly reduced by asking generative AI.

This experiment demonstrated that generative AI can be an effective tool for learning how to use unfamiliar software. Traditionally, learning how to use software typically required conducting web searches and gathering information from relevant websites. However, generative AI is emerging as an alternative to this process. Based on this tutorial, several participants successfully used software they had not used before.

3. Building an AI tutor for PHYSBO with MyGPT

Recent advances in generative AI have accelerated automated code maintenance in software development [6,7]. In particular, the emergence of Large Language Model (LLM)-based tools for code completion and debugging assistance has contributed to developer productivity [8,9]: tools such as GitHub Copilot [10] and Amazon CodeWhisperer [11] can understand the intent of a program, suggest appropriate code, and complete it accordingly [12,13]. This technological advance is moving beyond code completion to a deeper understanding of the semantics of code. In the future, AI may be able to integrate software specifications, manuals, and source code to automatically generate tutorials and documentation for users.

In this section, we explore the above possibility of using MyGPT to automatically generate tutorials for a specific piece of software. MyGPT is a tool that allows users to build customized conversational AI based on pre-provided information. The goal of this section is to evaluate the usefulness of MyGPT in generating tutorials when relevant information is provided in advance. Specifically, we assess MyGPT's ability to generate tutorials in terms of accuracy. Based on the results, the potential for future AI-based document generation is discussed.

As our target software, we selected PHYSBO [5], a Bayesian optimization package. Bayesian optimization is a method that uses Gaussian processes as a regression model to estimate the objective function from explanatory variables and suggest those predicted to yield the best objective function value. One of the key features of PHYSBO is its ability to propose multiple candidates and appropriately select an acquisition function, including Thompson sampling (TS), expected improvement (EI), and probability of improvement (PI). Furthermore, the computational cost, which typically requires $O(N^3)$ for the number of training data N, has been reduced to $O(N)$.

To evaluate the effectiveness of MyGPT in generating accurate tutorials and explanations for PHYSBO, we provided it with various sources of reference information. Initially, we configured MyGPT with a system prompt specifying its role as an AI assistant specializing in PHYSBO. The following official sources were used as reference materials:

- GitHub repository: https://github.com/issp-center-dev/PHYSBO
- Official paper: https://www.sciencedirect.com/science/article/pii/S0010465522001242
- User manual: https://issp-center-dev.github.io/PHYSBO/manual/master/en/index.html

Using this setup, we tested the accuracy of MyGPT by asking it the following question:

> What acquisition functions does PHYSBO support?

However, MyGPT initially provided an incorrect response, claiming that PHYSBO implements the Upper Confidence Bound (UCB) acquisition function, which is not implemented. Additionally, MyGPT failed to mention PHYSBO's support for multi-objective optimization.

To improve the accuracy of MyGPT's responses, we uploaded the official PHYSBO documentation and the research paper as PDF files. We then repeated the same query, resulting in a significantly improved and accurate response. MyGPT correctly listed PHYSBO's available acquisition functions:

For Single-Objective

- **Thompson Sampling (TS)**-Selects the input that maximizes a sampled regression function from the posterior distribution.
- **Expected Improvement (EI)**-Selects the input where the expected value of improvement over the current best value is maximized.
- **Probability of Improvement (PI)**-Selects the input with the highest probability of exceeding the current maximum objective function value.

For Multi-Objective Optimization

- **Thompson Sampling (TS)**-A multi-objective extension of Thompson Sampling, which samples from multiple Gaussian processes.
- **Hypervolume-Based Probability of Improvement (HVPI)**-Measures the probability of improving the non-dominated region in multi-dimensional objective space.
- **Expected Hypervolume Improvement (EHVI)**-Estimates the expected improvement in hypervolume for Pareto-front optimization.

Furthermore, MyGPT accurately noted that these acquisition functions balance exploration and exploitation in Bayesian optimization. It also correctly stated that PHYSBO allows users to specify the acquisition function via the `score` argument in its optimization functions.

The results indicate that providing URLs alone was insufficient to ensure accurate responses, since MyGPT generated plausible but incorrect information. This suggests that MyGPT may rely on general prior knowledge rather than accurately retrieving information from the linked sources. In contrast, uploading structured documents (PDFs) significantly improved response accuracy, probably because the model could directly extract relevant content. These findings highlight the importance of explicitly providing MyGPT with domain-specific documentation when using it for software tutorials and documentation generation. Future work will explore whether structured summaries or extracted key sections from documentation further improve accuracy and reduce the need to upload full PDFs.

To further evaluate the capabilities of MyGPT, we tested its ability to generate a hyperparameter optimization tutorial using PHYSBO for neural networks configured with Keras [14]. We provided the following prompt:

> Create a tutorial on optimizing hyperparameters in PHYSBO for neural networks configured using Keras.

MyGPT generated a step-by-step tutorial covering key aspects of hyperparameter tuning with PHYSBO, including:

- Installation of required packages.
- Definition of the hyperparameter search space, including learning rate, batch size, and number of hidden units.
- Implementation of a simple feedforward neural network in Keras, using the MNIST [15] dataset as an example.
- Evaluation of hyperparameter configurations using PHYSBO.
- Execution of Bayesian optimization to determine the best hyperparameter set.
- Extraction and reporting of the optimal hyperparameter values.

The generated tutorial included relevant Python code snippets that demonstrate the integration of PHYSBO with Keras. The core of the tutorial focused on employing TS and Bayesian optimization strategies to maximize model performance. The Python code generated by MyGPT for setting up and optimizing hyperparameters is uploaded to the ISSP data repository [16].

To validate the accuracy and effectiveness of the tutorial, we executed the generated code. The optimization process successfully converged to an optimal configuration, achieving an accuracy of 0.9742. The final hyperparameters were:

- Learning Rate: 0.001
- Batch Size: 16
- Number of Hidden Units: 128

Additionally, intermediate optimization steps exhibited steady improvements, demonstrating the effectiveness of Bayesian optimization for hyperparameter tuning. The results confirm that MyGPT can generate functionally correct and executable tutorials when provided with structured reference materials such as official documentation and research papers. However, continuous validation is still required to ensure accuracy, particularly when integrating AI-generated outputs into software documentation workflows.

This demonstration has shown that MyGPT can effectively generate executable tutorials for PHYSBO-based hyperparameter optimization when structured reference materials are provided. While initial responses contained inaccuracies, uploading domain-specific documentation significantly improved accuracy. The results suggest that AI-driven tutorial generation can enhance software usability and accessibility, though continuous validation remains necessary. Future work should explore integrating structured prompt engineering techniques to further refine the quality of generated outputs.

4. Development of a GUI application from Python scripts with the help of generative AI

Many Python scripts are available as open-source software (OSS) packages on GitHub, providing significant value for data analysis and machine learning tasks. However, if a proper Python environment is not set up, users cannot utilize these scripts on a local PC. To address this issue, converting Python scripts into Windows executable files is an effective way that allows them to be executed without any Python environment setup on Windows PCs. This approach offers considerable advantages, such as enabling practical exercises in university and high school lectures, as well as corporate seminars, simply by sharing the executable files. Additionally, in product development, this method eliminates the need to set up Python environments on individual PCs. Another option for running analyses without setting up a Python environment is the use of cloud services. However, cloud-based solutions pose security risks when dealing with confidential data, as their use may not be feasible for such scenarios. By contrast, Windows executable files allow offline and local execution of analyses. One of their key benefits is their ability to ensure a high level of security against data breaches.

In this section, we introduce the development of a GUI application for the Bayesian optimization package PHYSBO (see Figure 3), created with the help of ChatGPT. The original code used to convert to a Windows executable file was based on the PHYSBO code implemented in NIMO [17,18] (`ai_tool_physbo.py`). In NIMO, the process begins by preparing candidates.csv, which organizes all experimental condition candidates into a table. The rightmost column of `candidates.csv` is designated to store the objective function values: if an experiment has already been conducted, the corresponding objective function value is entered, while for experiments not yet performed, the field remains blank. An example is shown in Figure 3. The experimental conditions with completed objective function values are used as training data, and the next experimental conditions to be tested are proposed from the blank fields. These proposed experimental conditions are output as `proposals.csv`. It is important to

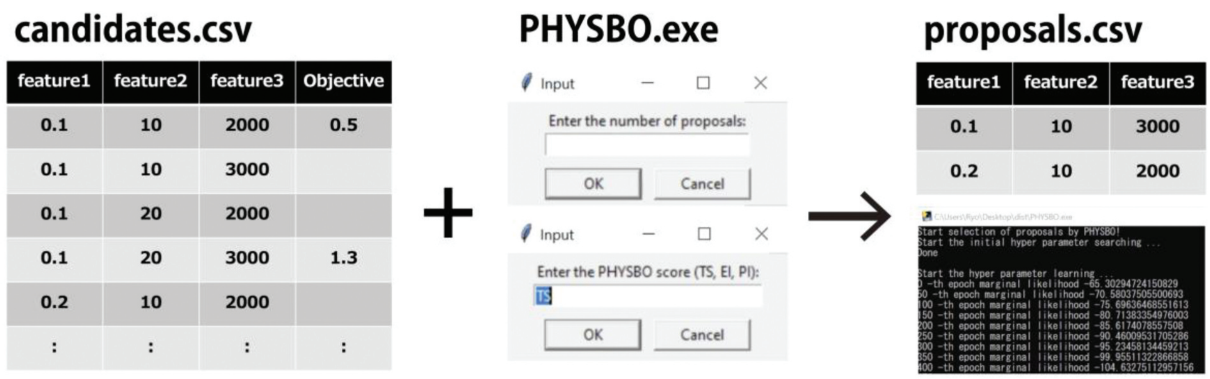


Figure 3. Windows executable file for PHYSBO was developed with the help of ChatGPT. Based on the candidate.csv file, which includes experimental conditions and some objective function values, promising candidates expected to yield higher objective function values are proposed.

note that the proposed explanatory variables are those predicted to maximize the objective function value. If the goal is to find experimental conditions that minimize the objective function value, this can be achieved by attaching a negative sign to the objective function values when storing them in `candidates.csv`.

The Windows executable file was created through interactions with ChatGPT by following the steps below.

(i) The prepared PHYSBO code was loaded into ChatGPT, and the following command was given:

> Convert the following script into an exe file so that it will work in a Windows environment.

In response, the Python code was generated, along with the following instructions. The PHYSBO code that was loaded can be found at https://github.com/NIMS-DA/nimo/tree/main/nimo/ai_tools.

To convert the given Python script into a Windows executable file, you can use PyInstaller. Below are the steps:

1. **Install PyInstaller**: Ensure you have pyinstaller installed. You can install it via pip:
   `pip install pyinstaller`
2. **Prepare the Script**: Make sure your script works correctly by itself when executed. It appears the provided script uses external dependencies such as numpy, physbo, etc. Ensure these are installed in your Python environment:
   `pip install numpy physbo`
3. **Generate the Executable**: Run the following command in the terminal where the script is located:
   `pyinstaller --onefile PHYSBO.py`
   - --onefile ensures the executable is packaged into a single file.
   - This will create a standalone .exe file in the dist folder.
4. **Locate the Executable**: After the command completes, the PHYSBO.exe file will be located in the dist directory.
5. **Test the Executable**: Move the .exe to a folder with the required input files (e.g. candidates.csv) and test it in a Windows environment.

By following these instructions, the Windows executable file was successfully created, executed, and verified.

(ii) We instructed ChatGPT as follows to modify the Python code so that the number of proposed candidates (`num_proposals`) and the acquisition function (`physbo_score`) could be specified as arguments:

> Please allow users to specify two arguments: `num_proposals` and `physbo_score`.

In response, the modified code was generated, and the following message was provided.

To modify the script to accept command-line arguments for `num_proposals` and `physbo_score`, we can use the argparse module. Here's the updated script:

Changes made:

1. Added the argparse module to parse command-line arguments.
2. Included `--num_proposals` and `--physbo_score` as required arguments.
3. Updated the `main()` function to read these arguments and pass them to the PHYSBO class.

Usage:

After converting this script to an .exe file using pyinstaller, you can specify arguments like this:

`PHYSBO.exe --num_proposals 5 --physbo_score TS`

This ensures flexibility for users to define the number of proposals and the scoring method.

(iii) The Windows executable file created through the above steps requires `num_proposals` and `physbo_score` to be specified as arguments, and they need to be entered via the command prompt on a Windows computer. To enable GUI-based specification of these two arguments, we asked ChatGPT the following:

> Please allow the user to specify two arguments, `num_proposals` and `physbo_score`, in the GUI.

In response, the following message was provided.

Changes made:

1. Replaced command-line arguments with a GUI using tkinter.
2. Added input dialogs for `num_proposals` and `physbo_score` using simpledialog.
3. Added error handling and user feedback using messagebox.

How it works:

1. When the script runs, users are prompted with GUI dialogs to input:
   (a) Number of Proposals.
   (b) PHYSBO Score (TS, EI, or PI).
2. If valid inputs are provided, the selection process proceeds, and a success message is displayed.
3. Errors during input or execution show an error message. This ensures the script is user-friendly in a Windows environment.

Although we did not explicitly mention to ChatGPT that there are three types of PHYSBO scores, ChatGPT automatically suggested a feature to allow users to select among three acquisition functions.

(iv) To make the application more user-friendly, we aimed to introduce a feature that allows the user to select an input file equivalent to `candidates.csv`. Therefore, we instructed ChatGPT as follows:

> Please allow the user to specify an input file equivalent to '`candidates.csv`' in the GUI.

In response, the following message was generated.

> Updates:
>
> 1. Added a file selection dialog using `filedialog.askopenfilename` to allow the user to specify the `input file`.
> 2. Updated the `main()` function to use the selected file as the `input_file`.
>
> How it works:
>
> - When the script runs:
>
> 1. A file dialog prompts the user to select an input file (`candidates.csv` equivalent).
> 2. The user specifies the number of proposals and PHYSBO score as before.
> 3. The selected input file is used in the optimization process.

This enhancement makes the script more flexible for different input files.

(v) In the application created above, only a file selection screen appeared, and no message was displayed. Therefore, we instructed ChatGPT as follows:

> Output a message to ask the user for the name of the input file in the GUI.

In response, the following message was generated.

> Updates:
>
> 1. Added a message box using `messagebox.showinfo` to notify the user to select an input file before the file dialog appears.
> 2. This makes the process more user-friendly by explicitly prompting the user.
>
> How it works:
>
> 1. A message box pops up asking the user to select the input file.
> 2. The file dialog then opens for the user to browse and select the required input file.

This improvement enhances clarity and guides the user through the process.

(vi) A feature to output predicted values for the test data through the GUI was implemented. In NIMO, setting the `output_res` argument to True allows the output of predicted values, variance values, and acquisition function values for the test data. To enable this option in the GUI, we instructed ChatGPT as follows:

> Please allow users to specify whether `output_res` should be False or True in the GUI.

In response, the following message was generated.

> Updates:
>
> 1. Added a `messagebox.askyesno` dialog to ask the user whether they want to enable `output_res`.
> 2. The user's choice (True for Yes, False for No) is passed to the PHYSBO class.
>
> How it works:
>
> 1. After selecting the input file and entering other parameters, the user is prompted with: Do you want to export prediction results?
> 2. Clicking '`Yes`' sets `output_res` to True.
> 3. Clicking '`No`' sets `output_res` to False.

This allows users to choose whether to export the prediction results, directly via the GUI. With the steps above, PHYSBO.exe has been completed and is publicly available on https://isspns-gitlab.issp.u-tokyo.ac.jp/physbo-dev/physbo-gallery/-/tree/master/tools/physbo_exe.

(vii) Finally, to create a manual for the developed executable application, we instructed ChatGPT as follows:

> Please create a manual for the GUI application created.

In response, the manual was generated along with the following message:

> The user manual for the PHYSBO GUI Application has been created. It includes instructions for setup, operation, and troubleshooting. If you need modifications or additional details, feel free to ask!

A video demonstrating the execution of PHYSBO.exe can be found as Supplementary Movie 1. The manual created by ChatGPT could be used with almost no modifications. In addition, the manual for preparing the candidates.csv file was generated by loading the NIMO manual into ChatGPT. The two manuals were then merged and slightly modified by ChatGPT, and the result is available at https://isspns-gitlab.issp.u-tokyo.ac.jp/physbo-dev/physbo-gallery/-/tree/master/tools/physbo_exe.

Since ChatGPT can easily perform language translation for the manuals, it becomes clear that the creation of both the Windows executable file and the manuals can largely be automated by generative AI. To prevent the Windows executable file from becoming too large, we used miniconda to prepare a minimal Python environment before creating the executable file. For simple GUI applications like this, generative AI can be effectively utilized to easily create them, and we expect it to be useful when creating educational content.

5. Summary

In conclusion, this paper introduces three applications of generative AI: Testing software, creating an AI tutor for software, and developing a GUI. These tasks usually require significant effort, but generative AI greatly reduces the required workload, as demonstrated in this paper. While generative AI does not always produce correct answers, human verification, such as testing, is necessary. However, our study confirms that generative AI often provides accurate results. Therefore, the use of generative AI in education and research in materials science is expected to become essential in the near future.

The rapid progress of generative AI will likely make the content of this paper outdated within a few years. However, we anticipate that this paper will serve as a valuable record of the early applications of generative AI in materials science.

Note

1. We used ChatGPT 4o and ImageJ 1.54 g in this tutorial.

Acknowledgements

A part of this study is supported by Data creation and utilization-type MaTerial R&D project (DxMT).

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Takahiro Misawa http://orcid.org/0000-0001-6799-6672
Ryo Tamura http://orcid.org/0000-0002-0349-358X

References

[1] OpenAI. ChatGPT; 2025 [cited 2025 Mar 7]. Available from: https://chatgpt.com
[2] AIMHack2024. AIMHack2024; 2024 [cited 2025 Feb 18]. Available from: https://dxmt.mext.go.jp/news/1105
[3] MatDaCs. MatDaCs; 2025 [cited 2025 Feb 18]. Available from: https://mat-dacs.dxmt.mext.go.jp
[4] ImageJ. ImageJ; 2025 [cited 2025 Feb 18]. Available from: https://imagej.net/software/imagej/
[5] Motoyama Y, Tamura R, Yoshimi K, et al. Bayesian optimization package: PHYSBO. Comput Phys Commun. 2022;278:108405. Available from: https://www.sciencedirect.com/science/article/pii/S0010465522001242
[6] Bubeck S, Chandrasekaran V, Eldan R, et al. Sparks of artificial general intelligence: early experiments with gpt-4. arXiv preprint arXiv:2303.12712. 2023; Available from: https://arxiv.org/abs/2303.12712
[7] Chen M, Tworek J, Jun H, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374. 2021; Available from: https://arxiv.org/abs/2107.03374
[8] Imai S. Is github copilot a substitute for human pair-programming? An empirical study. In: 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion); Pittsburgh, PA, USA. IEEE; 2022. p. 319–321. doi: 10.1145/3510454.3522684
[9] Zhang B, Liang P, Zhou X, et al. Practices and challenges of using github copilot: an empirical study. arXiv:2303.08733. 2023; Available from: https://arxiv.org/abs/2303.08733
[10] GitHub, Inc. Github copilot; 2025 [cited 2025 Feb 18]. Available from: https://github.com/features/copilot
[11] Amazon Web Services. Amazon codewhisperer; 2025 [cited 2025 Feb 18]. Available from: https://aws.amazon.com/codewhisperer/
[12] Yetiştiren B, Özsoy I, Ayerdem M, et al. Evaluating the code quality of ai-assisted code generation tools: an empirical study on github copilot, amazon codewhisperer, and chatgpt. arXiv preprint arXiv:2304.10778. 2023; Available from: https://arxiv.org/abs/2304.10778
[13] Pearce H, Ahmad B, Tan B, et al. Asleep at the keyboard? assessing the security of github copilot's code contributions. arXiv:2108.09293. 2021; Available from: https://arxiv.org/abs/2108.09293
[14] Chollet F. Deep learning with python. New York (NY): Manning Publications Co; 2017.
[15] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–2324. doi: 10.1109/5.726791
[16] PHYSBO Gallery. 2022 [cited 2025 Mar 9]. Available from: https://isspns-gitlab.issp.u-tokyo.ac.jp/physbo-dev/physbo-gallery
[17] Tamura R, Tsuda K, Matsuda S. NIMS-OS: an automation software to implement a closed loop between artificial intelligence and robotic experiments in materials science. Sci Technol Adv Mater. 2023;3(1):2232297. doi: 10.1080/27660400.2023.2232297
[18] NIMO. 2024 [cited 2025 Mar 9]. Available from: https://github.com/NIMS-DA/nimo