Taylor & Francis
Taylor & Francis Group

Science and Technology of Advanced Materials: Methods

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

Simple implementation examples of agent AI on free energy calculation and phase-field simulation

Toshiyuki Koyama, Yusuke Matsuoka & Akimitsu Ishii

To cite this article: Toshiyuki Koyama, Yusuke Matsuoka & Akimitsu Ishii (11 Dec 2025): Simple implementation examples of agent AI on free energy calculation and phase-field simulation, Science and Technology of Advanced Materials: Methods, DOI: 10.1080/27660400.2025.2601436

To link to this article: https://doi.org/10.1080/27660400.2025.2601436

Publisher: Taylor & Francis & The Author(s). Published by National Institute for Materials Science in partnership with Taylor & Francis Group

Journal: *Science and Technology of Advanced Materials: Methods*

DOI: 10.1080/27660400.2025.2601436

Simple implementation examples of agent AI on free energy calculation and phase-field simulation

Toshiyuki Koyama[1*)], Yusuke Matsuoka[1)], Akimitsu Ishii[1)]

1) Research Center for Structural Materials, National Institute for Materials Science

* CONTACT: Toshiyuki Koyama, E-mail: KOYAMA.Toshiyuki@nims.go.jp
Research Center for Structural Materials (RCSM), National Institute for Materials Science (NIMS)
1-2-1, Sengen, Tsukuba, Ibaraki 305-0047, JAPAN

Abstract

With the development of an environment that equips large language models with tools, chat-type artificial intelligence (AI) is currently shifting to agent-type AI. Although building an agent AI with complex behavior is an important issue, a method for easily implementing simple agent AI is also useful in materials research. The objectives of this study are to demonstrate the usefulness of simple agent AI codes, and to distribute them as supplemental materials. The key points are summarized as follows. (1) Using Gibbs energy calculations as an example, we demonstrated a simple method for constructing agent AI, including explanation and distribution of the python code. (2) We applied this method to a scratch code development of phase-field simulation, and the template python code was also distributed. (3) Using the simple agent AI technique, we were able to easily verify that the Jarzynski equality is applicable for the diffusion behavior in diffusion couple, which provides a new approach for directly evaluating free energy change from diffusion flux information.

Keywords: generative AI, LLM, LangGraph, Gibbs energy, Phase-field method, Jarzynski equality

1. Introduction

Currently, research and development related to agent artificial intelligence (AI) is progressing at a tremendous pace after the release of large language models (LLMs) such as ChatGPT (GPT-3.5) and image-generative AI such as Stable Diffusion in 2022 [1–4]. The capabilities of generative AI have expanded year by year, and the shift from chat-type AI to agent AI has accelerated, as indicated in the roadmap of Open AI, with the goal of attaining an artificial general intelligence [5]. The development focus has moved to enhance the inference abilities of an LLM. Agent AI is often utilized for limited tasks such as document retrieval, with a retrieval-augmented generation (RAG) system as an example [6,7]. However, an agent AI framework now has much broader applicability to research in science and engineering [1–3,8]. We previously proposed a fundamental concept for effectively utilizing agent AI in materials research [8]. In the current study, we demonstrate simple agent AI implementation examples on free energy calculation and phase-field simulation and discuss the effective environment for the future development of agent AI in this field.

Historically, the evolution of AI can be explained in terms of the following three stages, analogous to the evolution of living things. Phase I is the "birth of an eye," meaning the stage when AI became capable of distinguishing between objects such as a dog and cat. In biological evolution, the birth of an eye at the Cambrian period led to an explosive increase in species and dramatic natural selection [9]. Phase II is the "invention of language and the acquisition of communication ability," which corresponds to the recent emergence of LLMs. This is the period when AI itself began to acquire language and communication skills. Phase III is the "utilization of external resources and tools," which corresponds to the transformation of chat-type LLMs to agent AI, which is the main topic of this paper. Originally, LLMs were known to have the following four inherent limitations [10]. (1) LLMs could not access the latest information (e.g., today's weather or stock prices). (2) They were unable to access specific resources of information. (3) Their logic was insufficient, and sometimes even simple calculations were incorrect. (4) Catastrophic forgetting (i.e., fine-tuning for specific tasks could reduce the versatility of an LLM). To overcome these issues, the concept of agent AI was proposed. Although the word “agent” generally refers to autonomous actions that do not involve human contributions, agent AI can be defined in this paper as “a system that uses an LLM to decide the control flow of an application” according to the definition by Chasethe [11]. “The control flow of an application" refers to the method for using external resources and tools through an LLM, the details of which are explained in the next section. (Conceptually, the utilization of external resources by an LLM may correspond to Gutenberg's invention of the printing press and the emergence of the Internet, which were evolutional events for humankind. Without exaggeration, the utilization of tools by an LLM will be equivalent to the Industrial Revolution.) Furthermore, an approach that combines multiple LLMs, known as “orchestration,” has also recently emerged [12]. Therefore, it is no overstatement to say that agent AI is currently undergoing a significant evolutionary period.

We previously proposed a fundamental concept for effectively using agent AI in materials research

[8], which is explained again in the next section. The agent AI that can perform highly complex tasks such as MatPilot [13] is undoubtedly useful. On the other hand, it is also beneficial in materials research that simple and iterative tasks can be easily implemented as agent AI using in-house code. This study focused on specific cases that used free energy calculations and phase-field simulations as typical examples to demonstrate how this concept was implemented as a useful research method, where the python codes are distributed as supplemental materials. Furthermore, as an application example of this simple agent AI technique, we tried to confirm Jarzynski equality is applicable for the diffusion behavior in diffusion couple, which provides a new calculation method for directly evaluating free energy changes from diffusion flux information.

2. Model framework

Figure 1 shows a typical example to understand the basic behavior of agent AI, which answers the question: “What is the appropriate attire for going out in Tokyo today?” [3,8]. We put the agent AI at the center of Fig. 1, with the LLM and Google search tool at the top and bottom, respectively. Note that the LLM is placed outside the agent AI. In many cases, an agent AI is regarded as an extension of an LLM, and the LLM is placed inside the agent AI. However, there is now flexibility in choosing which LLM to use from several those are available. This study adopted a definition of agent AI that utilized an LLM. Therefore, we placed the LLM outside the agent AI, as shown in Fig. 1.

In Fig.1, today's temperature in Tokyo is required as information. However, because the LLM has only learned past information, it does not know the current temperature in Tokyo. Although the LLM does not know the current temperature in Tokyo, it knows that we can use Google search. Thus, the LLM sends back “Use tool: Google search” along with “input keyword: current temperature in Tokyo” to the agent AI. The agent AI can execute the Google search tool, which returns, for example, the current temperature of 35 °C. (Note that LLM itself does not have the ability to use every external environment.) The agent AI returns the current temperature value to the LLM, which allows it to give a reasonable answer: "Tokyo is 35 °C, so cool biz is good!” In short, the LLM gives us the appropriate tool name to use in the next step and the input parameters required for the tool. A list of tool names and information on how to use them are provided to the LLM in advance via the agent AI. In other words, the agent AI acts as a bridge between tools and the LLM.

The above explanation is a familiar example, whereas Fig. 2 shows a case involving free energy calculations [8]. Here, given the alloy composition and temperature, we ask the AI agent to calculate the Gibbs energy of an Fe alloy. (An example of Python code is explained in the next section.) The program for calculating the Gibbs energy is placed at the bottom of Fig. 2, and the tool name is “Gibbs energy calc.” The agent AI is asked “What is the Gibbs energy of Fe-20at%Cr alloy at 1000 K?” Needless to say, the LLM cannot calculate the Gibbs energy, but because it has been taught by the agent AI to use the tool “Gibbs energy calc,” the LLM returns the following instruction to the agent AI: “Use the tool ‘Gibbs energy calc’; set the input parameters: alloy system: Fe-Cr, composition: 0.2,

temperature: 1000K.” When the agent AI executes the tool using these input parameters [alloy system: Fe-Cr, composition: 0.2, temperature: 1000 K], it returns -2920.3 (J/mol). After the agent AI provides this Gibbs energy value to the LLM, it returns the correct answer: “The Gibbs energy of Fe-20at%Cr at 1000 K is -2920.3 (J/mol).” On the other hand, it is interesting to note that when it is asked, “What is the highest mountain in Japan?”, the agent AI responds, “Mount Fuji.” Therefore, this agent AI is a specific enhancement of the existing LLM.

3. Python code and response examples

To understand the behavior of agent AI, it is useful to look at the simple program code. Here, we explain a specific implementation. Figure 3 shows the Python code written using LangGraph [14] for the case shown in Fig. 2, where Ollama [15] was used as the LLM system, and qwen 3 [15] was employed as the language model. Ollama was selected as the LLM system out of consideration for both convenience and security in scientific and technological research. Recently, there have been significant developments in the field of local LLMs [16], and Ollama is a representative open-source environment for easily running local LLMs. All of the calculations mentioned in this paper can be executed in a closed computing system such as a local PC.

In Fig. 3, (A) shows the import section of Python library. (B) and (C) are subroutines that calculate the Gibbs energy values of the bcc phase in Fe-Cr and Fe-C alloys, respectively. (D) shows the declaration of the tools in (B) and (C), the definition of the LLM, and the construction of the agent AI by connecting the tools and LLM. Finally, (E) shows question example. Because the algorithm in Fig. 3 is a common form of a reasoning and acting (ReAct) agent [7,17], the code is quite simple, as shown in (D). The standard LLM and ReAct agent settings are now provided by libraries, as indicated by the red text in (A), making it very easy to create agent AI code in the programming environment of LangChain and LangGraph.

In Fig. 3, (B) and (C) show the scratch code for calculating the Gibbs energy values of bcc phase in Fe-Cr and Fe-C alloys using Gibbs energy functions obtained from the CALPHAD database [18–20]. By replacing these parts with various calculation programs used in materials research, it becomes an LLM that can carry out a wide range of calculations in this field. The important point is that we can ask questions in natural language when executing the tools. Thus, this system will provide the benefit of allowing anyone, from students to professional researchers, to easily utilize various software tools in materials research.

Next, we explain how the LLM infers and answers questions with specific examples. In the following, Q is the question content, which is shown in blue in Fig. 3(E), and A is the response of the LLM to the question.

== Example 1 ==========

Q: Calculate the Gibbs energy of an Fe-C alloy with a composition 0.03 C at 1000 K.

A: Okay, the user asked for the Gibbs energy of an Fe-C alloy with 0.03 C at 1000 K. I called the Fe_C_gibbs_energy function with comp1=0.03 and temp1=1000. The response from the tool was -40676.555090907925 J/mol. Now I need to present this result in a clear way.

First, I should confirm the parameters used: temperature is 1000 K, and the carbon composition is 0.03 atomic fraction. The function returned a value of approximately -40,676.56 J/mol. Since Gibbs energy is often expressed with appropriate units, I'll make sure to include the unit. Also, considering significant figures, the input values (0.03 and 1000 K) might imply two decimal places for the composition, but the temperature is given as a whole number. The result has many decimal places, so rounding to a reasonable number, maybe two decimal places, would make it more readable. However, the exact value from the tool is precise, so perhaps present it as is but format it nicely. I should also mention that the value is negative, which indicates the system is stable under these conditions. Finally, offer further assistance in case they need more calculations.

The Gibbs energy of the Fe-C alloy with 0.03 atomic fraction of carbon at 1000 K is **-40,676.56 J/mol**.

This negative value indicates the alloy is thermodynamically stable under these conditions. Let me know if you need calculations for other compositions or temperatures!

==========

This is a single-point calculation of the Gibbs energy of bcc phase in Fe-C alloy with a fixed alloy composition and temperature. The correct value is calculated based on the tool program of Fig. 3(C). The LLM explains how it arrived at the calculation result, considering the significant digits.

==== Example 2 ==========

Q: For an Fe-Cr alloy, calculate the Gibbs energy at 1000 K with Cr compositions of 0.2 and 0.3.

A: Okay, let me process this. The user asked for the Gibbs energy of an Fe-Cr alloy at 1000 K with Cr compositions of 0.2 and 0.3. I called the Fe_Cr_gibbs_energy function twice with those compositions. The responses came back as -2920.27 and -3145.90 J/mol.

Now, I need to present these results clearly. The user might be studying phase diagrams or thermodynamics of alloys. They probably want to understand how Gibbs energy changes with composition. Maybe they're working on a project related to materials science, like optimizing alloy properties.

I should state the temperatures and compositions again for clarity. Then list the Gibbs energies with their respective compositions. It's also good to mention that the values are negative, which indicates the system is stable under these conditions. Maybe add a note about the significance of the values in terms of alloy stability or phase behavior. But keep it concise since the user might need this for further calculations or analysis.

The Gibbs energy for the Fe-Cr alloy at 1000 K is calculated as follows:

- **Cr composition = 0.2**: Gibbs energy = **-2920.27 J/mol**
- **Cr composition = 0.3**: Gibbs energy = **-3145.90 J/mol**

These values indicate the stability of the alloy at the given temperature and compositions. Negative Gibbs energy suggests the system is thermodynamically favorable under these conditions. Let me know if you need further analysis!

==========

This is a case where two calculations were performed using the Fe-Cr alloy tool, fixing the temperature and changing the alloy composition. Correct values were calculated using the tool program shown in Fig. 3(B). The LLM also explains how it arrived at the calculation results.

===== Example 3 ==========

Q: Calculate the Gibbs energy values of Fe-Cr alloys. The temperature is 1000 K, and the Cr compositions range from 0.1 to 0.9 in 0.1 increments.

A: Okay, let me process this. The user asked for the Gibbs energy values of Fe-Cr alloys at 1000 K with Cr compositions from 0.1 to 0.9 in 0.1 increments. I called the Fe_Cr_gibbs_energy function for each composition. The tool responses provided the Gibbs energy values for each comp1. Now I need to present these results clearly.

First, I should list the compositions and their corresponding Gibbs energies. Let me check the order of the responses to make sure they match the comp1 values. The first response was for 0.1, then 0.2, up to 0.9. The energies are given in the order of the tool calls. I'll format them into a table for clarity. Also, I should mention the units (J/mol) and maybe note that the values are calculated at 1000 K. Let me verify if all nine compositions are covered. Yes, 0.1 to 0.9 in steps of 0.1 is nine values. The responses have nine entries, so they match. I'll present the data in a clear, tabular format and summarize if needed. The user

might want to see how the Gibbs energy changes with Cr content, so highlighting the trend could be helpful. But since they just asked for the values, sticking to the data with proper formatting is key.
Here are the calculated Gibbs energy values for Fe-Cr alloys at 1000 K, with Cr compositions ranging from 0.1 to 0.9 (atomic fraction):

Cr Composition (comp1)	Gibbs Energy (J/mol)
0.1	-2379.30
0.2	-2920.27
0.3	-3145.90
0.4	-3201.99
0.5	-3155.68
0.6	-3031.73
0.7	-2812.61
0.8	-2429.62
0.9	-1729.10

The values show a non-linear trend, reaching a minimum at ~0.4 Cr content. Let me know if you need further analysis!
==========

This is an example of a loop calculation being requested in words. It is interesting that the LLM automatically responded in table form. Even if there is a simple subroutine for calculating a single point, agent AI has the advantage of providing the desired results using the loop calculation with different conditions, which can be carried out by simply giving the instruction in words. It should be emphasized that there is no need to create program code with a loop calculation.

==== Example 4 ==========

Q: Calculate the Gibbs energy values of Fe-Cr alloys. The temperature is 1000 K, and the alloy compositions range from 0.1 to 0.3 in 0.01 increments. Then, find the alloy composition that has a Gibbs energy close to -3000 (J/mol).

A: Okay, let's see. The user asked to calculate the Gibbs energy for Fe-Cr alloys at 1000 K with compositions from 0.1 to 0.3 in 0.01 increments. Then find the composition closest to -3000 J/mol.
First, I called the Fe_Cr_gibbs_energy function for each composition step. The responses came back as a list of values. Now I need to check which of these values is closest to -3000.
Looking at the provided Gibbs energy values:
Starting from 0.1 (comp1=0.1) the value is -2379.3, then it increases in magnitude negatively (more negative) as comp1 increases. Wait, actually, the numbers are getting more negative: -2379, -2455, -2525, etc., down to -3145.8997 at comp1=0.3.
Wait, the target is -3000. Let me list the values with their comp1:
comp1=0.1: -2379.3, comp1=0.11: -2455.6, comp1=0.12: -2525.8, comp1=0.13: -2590.3, comp1=0.14: -2649.7
comp1=0.15: -2704.5, comp1=0.16: -2755.0, comp1=0.17: -2801.6, comp1=0.18: -2844.5, comp1=0.19: -2883.99
comp1=0.2: -2920.27, comp1=0.21: -2953.58, comp1=0.22: -2984.10, comp1=0.23: -3012.01, comp1=0.24: -3037.46
comp1=0.25: -3060.59, comp1=0.26: -3081.54, comp1=0.27: -3100.44, comp1=0.28: -3117.39, comp1=0.29: -3132.51
comp1=0.3: -3145.90

Looking for the value closest to -3000. Let's check each: At comp1=0.23, the value is -3012.01, which is 12.01 below -3000. comp1=0.22: -2984.10, which is 15.9 above -3000. So between comp1=0.22 and 0.23, the closest is comp1=0.23 with -3012.01 (difference 12.01) versus comp1=0.22 at -2984.10 (difference 15.9). Therefore, comp1=0.23 is closer. Wait, but wait: the target is -3000. Let me calculate the absolute differences:
For comp1=0.22: |-2984.10 - (-3000)| = |15.9| = 15.9
For comp1=0.23: |-3012.01 - (-3000)| = |-12.01| = 12.01

So comp1=0.23 is closer. However, the user might want the closest value. Alternatively, maybe there's a value even closer? Let me check the next ones.
comp1=0.24: -3037.46, difference is 37.46 above -3000.
comp1=0.21: -2953.58, difference is 46.42 above.
So the closest is comp1=0.23 with -3012.01, which is 12.01 below -3000. Alternatively, maybe there's a value between 0.22 and 0.23? But the increments are 0.01, so the data points are at 0.22 and 0.23. Therefore, the answer is comp1=0.23 with Gibbs energy -3012.01 J/mol, which is the closest to -3000.
The Gibbs energy values for Fe-Cr alloys at 1000 K are calculated for compositions ranging from 0.1 to 0.3 (in 0.01 increments). The composition closest to a Gibbs energy of **-3000 J/mol** is:
Alloy composition (Cr atomic fraction): 0.23
Calculated Gibbs energy: -3012.01 J/mol
This value is **12.01 J/mol** below -3000, making it the closest in the provided dataset.
==========

This is a case where a series of loop calculations are performed. Then, a specific condition is given, and finally a solution that satisfies the condition is selected from the calculation results. Therefore, this is a simple example of assigning two sequential tasks to the agent AI. It can be seen that the LLM thought logically and arrived at the correct solution. In addition, it was necessary to use the appropriate prompt when asking the LLM in order to obtain the correct answer. As shown in the Q part, describing questions sequentially is effective.

4. Application to phase-field simulation code development

Phase-field (PF) simulation [21–24] is one of the computational methods for calculating material microstructure developments. In this section, we explain the effective use of agent AI for the scratch code development for the PF method. Note that, since this is an application of agent AI to code development, there is nothing new in the PF method itself.

Because source code is often created from scratch for a PF method, numerous trial-and-error calculations with various conditions are needed during the code development stage (this is referred to as Case 1 henceforth). Recently, the use of PF simulation for the mass production of microstructure images prepared by changing materials and process conditions has been rapidly increasing in order to obtain training data for machine learning or fine-tuning an image generative AI [25] (Case 2). In these cases, the elemental calculations are simple, but the amount of work is enormous. Let us recall that the main advantage of an agent AI is its ability to patiently perform simple and iterative tasks. Using agent AI, we can execute simulations by specifying the material and process conditions and their ranges in natural language. Therefore, by creating only a single basic simulation tool for an agent AI, we can assign the tasks of Cases 1 and 2 to the agent AI. Considering Cases 1 and 2, we designed the agent AI system shown in Fig. 4 as a simple example for various tasks related to PF simulation scratch code construction, focusing on a spinodal decomposition from a supersaturated solid solution during isothermal aging in a disordered phase of an A-B binary alloy [21,26]. (Note that this PF simulation is equivalent to a calculation based on the Cahn–Hilliard non-linear diffusion equation.) There are four tools: (1) Simulate the spinodal decomposition and display the simulated microstructures currently being calculated (input parameters: alloy composition, aging temperature, and time step); (2) Change

the initial composition field (distinguished by the index number), and save the microstructure data calculated during the simulation (input parameters: index number, alloy composition, aging temperature, and time step); (3) Display the microstructures that have already been simulated (input parameter: file name); and (4) Display the microstructure and save an image of each microstructure to disk in an image file format (input parameter: file name). Tool (1) is mainly used in Case 1 and is useful for the development stage of simulation code. Tool (2) is useful in Case 2, because it enables the easy mass production of microstructure training data for machine learning. Tools (3) and (4) are used for confirming the calculation results of tool (2) and converting them into image files, respectively. It should be repeated again that the advantage of using agent AI is the ability to provide all of the input instructions in natural language. Not only is it possible to simply assign calculation tasks, but the input data range can also be specified, thereby enabling flexible changes in the calculation conditions and making it possible to obtain results even when using simple tools.

Figure 5 shows examples of the calculation results on (2) and (4) in Fig. 4. Many simulation output files with different initial composition fields calculated all at once are listed in Fig. 5(a), and Figure 5(b) shows the microstructural changes obtained from one of these fields, where a typical microstructure development by spinodal decomposition is calculated. The sample code of the agent AI investigated in this study, including the calculation results and responses from the LLM, is available in the supplementary material of this paper or on GitHub [27]. For further details, refer to the python code sample.

5. Application of Jarzynski equality to diffusion couple

Agent AI is also beneficial to increase the efficiency of verification tasks for new theory. In this section, we explain the results of verifying whether the Jarzynski equation [28] is applicable to a diffusion work in diffusion couple based on the diffusion simulation that is a simplest phase-field method, assuming a hypothetical A-B binary alloy system. The Jarzynski equality is a fundamental relation in non-equilibrium statistical mechanics that connects the work done W during a non-equilibrium process to the free energy change ΔF, which is derived from the Crooks fluctuation theorem [29] that describes the statistical relationship between forward and backward processes in non-equilibrium thermodynamics. To the authors' knowledge, there are no studies applying this theory to the diffusion couple. The Jarzynski equation [28] is given by

$$\exp\left(-\frac{\Delta F}{RT}\right)=\left\langle \exp\left(-\frac{W}{RT}\right)\right\rangle,$$

where R is gas constant and T is absolute temperature. The purpose of this analysis is to verify whether ΔF can be directly determined from information about the diffusion flux J. The calculation target is the diffusion couple of the α phase in a hypothetical A-B binary alloy, where the Gibbs energy is expressed using the following regular solid solution approximation [20, 21]:

$$G_{\mathrm{m}}^{(\alpha)}(c,T) = G_{\mathrm{A}}^{(\alpha)}(T)(1-c) + G_{\mathrm{B}}^{(\alpha)}(T)c + L_0 c(1-c) + RT\{c\ln c + (1-c)\ln(1-c)\}.$$

$G_{\mathrm{X}}^{(\alpha)}(T)$ is the Gibbs energy of pure component X, and the energy reference was set to $G_{\mathrm{A}}^{(\alpha)}(T) = G_{\mathrm{B}}^{(\alpha)}(T) = 0$ in this analysis. c is the molar fraction of component B (the molar fraction of component A is $1-c$), and L_0 is the interatomic interaction parameter which is assumed to be constant for simplicity.

The diffusion work W, physically being the friction work associated with atom movement [21], is formulated as follows [30].

$$W(t) = \int_0^t \frac{dw}{dt}dt, \quad \frac{dw}{dt} \equiv \int_x \frac{J^2(x,t)}{M}dx, \quad J(x,t) \equiv -M\frac{d\mu^{(\alpha)}}{dx}, \quad \mu^{(\alpha)} \equiv \frac{dG_{\mathrm{m}}^{(\alpha)}}{dc}$$

$\mu^{(\alpha)}$ is the diffusion potential [21], and the local diffusion flux J was assumed to follow the generalized Fick's first law (x is the one-dimensional spatial coordinate) [21, 22]. M is the mobility of atom diffusion, which is expressed as $M \equiv c_0(1-c_0)D(T)/(RT)$ based on the interdiffusion theory [21]. c_0 is the alloy composition, $D(T)$ is the diffusion coefficient given by $D(T) = D_0\exp\{-Q_0/(RT)\}$, where D_0 and Q_0 represent the frequency factor and activation energy of diffusion, respectively. The quantity J^2/M is the dissipation function of the first kind [30], which is defined as the product of the chemical force for diffusion J/M and the diffusion flux J, hence it is physically corresponding to the friction work of diffusion. As for the right-hand side of the Jarzynski equation, an ensemble calculation is required. Therefore, the ensemble calculation was performed for the diffusion work at each individual position and $\Delta F(t)$ was calculated by following equations.

$$\begin{aligned} &\left\langle \frac{dw}{dt} \right\rangle \equiv -RT\ln\int_x \exp\left(-\frac{1}{RT}\frac{J^2(x,\tau)}{M}\right)dx = -RT\ln\int_x \exp\left(-\frac{J^2(x,\tau)}{c_0(1-c_0)D(T)}\right)dx, \\ &\Delta F(t) = W(t) = \int_0^t \left\langle \frac{dw}{dt} \right\rangle dt \end{aligned} \tag{1}$$

Figures 6(a)-(c) show examples of the temporal evolution of the composition profiles in diffusion couple, calculated based on the following diffusion equation [21], where the conditions $T = 1600(\mathrm{K})$ and $L_0 = 25000(\mathrm{J/mol})$ are employed.

$$\frac{dc}{dt} = -\frac{dJ}{dx} = M\frac{d^2\mu^{(\alpha)}}{dx^2}$$

Figure 6(a) is the initial composition profile of diffusion couple. The initial compositions on the left and right sides of the diffusion couple are taken as $c_{\mathrm{L}} = 0.93$ and $c_{\mathrm{R}} = 0.07$, respectively. In this calculation, the diffusion parameters $D_0 = 8.9\times10^{-5}(\mathrm{m}^2/\mathrm{s})$ and $Q_0 = 291(\mathrm{kJ/mol})$ of γ-Fe are used [31], and the time t is expressed in real time (sec). The free energy change associated with the temporal evolution of the composition profile can be also calculated directly from the composition field based on eq. (2), where the spatial integral is over unit length.

$$\Delta F(t) = \Delta G(t) = \int_x G_{\mathrm{m}}^{(\alpha)}\{c(x,t)\}dx - \int_x G_{\mathrm{m}}^{(\alpha)}\{c(x,0)\}dx \qquad (2)$$

Since this is a solid-state diffusion, the volume change is considered negligible, we assumed Helmholtz energy = Gibbs energy. Figure 6(d) shows the temporal change of the free energy $\Delta F(t)$ during the evolution of composition profiles. The results are plotted in red and black curves for eq. (1) and eq. (2), respectively. It can be confirmed that the two curves are almost overlapping. This result indicates the possibility of directly estimating $\Delta F(t)$ using the diffusion flux J. Strictly speaking, since a diffusion mobility of real alloy system is also a function of free energy, accurate evaluation by eq. (1) is not so easy. However, when the diffusion coefficient can be approximated as a function of temperature only, it is possible to quantitatively determine $\Delta F(t)$ using J from eq. (1).

In this analysis, an agent AI was employed to verify the effects of material parameter values (interaction parameter $L_0(\mathrm{J/mol})$: -25000 ～ 25000), process conditions (temperature $T(K)$: 1600 ～ 1800), and numerical calculation errors (spatial mesh number N in finite difference calculation: 100 ～ 600). Figure 7 shows the examples of these calculations, where each figure is in the same representation as Fig. 6(d). Although there are no results inconsistent with the conclusion in Fig. 6(d), it was confirmed that numerical calculation errors appears under the conditions of spatial mesh number $N = 100$ and the interatomic interaction parameter $L_0 = -25000(\mathrm{J/mol})$. The series of calculations showed that the spatial mesh number $N \geq 300$ is necessary to suppress the numerical calculation error (Figure 6(d) uses $N = 400$). By using agent AI to verify calculation results, we can perform analysis quite efficiently and simply by specifying the range of variables in natural language. The python program codes in this section are also available in the supplementary material or on GitHub.

6. Discussion

This study focuses on the agent AI that performs simple tasks using local LLMs from the viewpoint of research security and convenience. Then we considered the qwen series [15] to be suitable models

for the LLM, and used qwen3:8b in our analysis. Since LLM performance varies with LLM size (Number of parameters in a neural network), Table 1 shows the results of comparing performance among qwen series according to the LLM size. Examples 1 to 4 correspond to the four example questions in section 3, and the meanings of each symbol are as follows: ○ Correct answer; △ Partially correct, and includes incorrect answer; × Incorrect answer or the LLM is going around in circles. As anticipated, the larger the LLM size, the more correct answers are obtained, and it was concluded that qwen3:8b or higher is required for the simple tasks in this study.

Another important point is that tools and agents can be combined arbitrarily, i.e., there is no need to use only one agent. As a trial, we added a PF simulation program for the spinodal decomposition of bcc phase in Fe-Cr alloy [20] to the tool part of Fig. 3 and constructed a system with three agents and a supervisor agent. We then gave the instruction, "Execute phase-field simulation for Fe-50at%Cr at 673 K isothermal aging." As expected, we obtained the correct simulation results for the spinodal decomposition. This code is also available in the supplementary material of this paper or on GitHub [28]. Furthermore, in recent years, Anthropic released the model context protocol (MCP) [32,33], which is a protocol for constructing external servers equipped with functions for resources, tools, and prompts, with an agent AI linked to an LLM. In the case of Fig. 3, tools (B) and (C) are placed on the MCP server side. The client-side software specifies the LLM, and connects the resources and tools on the MCP server side to the LLM.

The authors are now planning to integrate all the simulation program codes of PF method, which had been developed previously in our group [21-24], into the format of Fig. 4. Therefore, it is expected that this approach will eventually extend to cover the wide range of materials science and engineering.

7. Conclusions

Currently, the framework for agent AI can be constructed very easily using an environment such as LangGraph. Therefore, an agent AI will be utilized on a daily basis at various levels of materials research. This paper focused on the emergence of this new research method and explained the simple examples in detail. The important points are summarized as follows.

1) Using Gibbs energy calculations as an example, we demonstrated a simple method for constructing an agent AI with the framework of LangGraph. The python codes are available in the supplementary material of this paper or on GitHub.

2) We applied the agent AI framework to the scratch code development of phase-field simulation, and the usefulness was confirmed. The template python code was also distributed.

3) Agent AI is also beneficial to increase the efficiency of verification tasks for new theory. Using the simple agent AI technique, we were able to easily verify that the Jarzynski equality is applicable for the diffusion behavior in diffusion couple, which provides a new approach for directly evaluating free energy change from diffusion flux information.

Acknowledgments: This work was supported in part by the JST SIP ("Materials Integration" for Revolutionary Design System of Structural Materials), MEXT Data Creation and Utilization Materials Research and Development Project (Data Creation and Utilization Magnetic Materials Research Center JPMXP1122715503), and JST-CREST (JPMJC22C3).

References

[1] Oliveira Jr. ON, Christino L, Oliveira MCF, Paulovich FV. Artificial Intelligence Agents for Materials Sciences. J. Chem. Inf. Model. 2023; 63(24), 7605–7609. doi: 10.1021/acs.jcim.3c01778

[2] Van M-H, Verma P, Zhao C, Wu X. A Survey of AI for Materials Science: Foundation Models, LLM Agents, Datasets, and Tools. arXiv preprint arXiv:2506.20743, 2025. doi: 10.48550/arXiv.2506.20743.

[3] AI Google Search Agent. https://github.com/Jana2207/AI_Google_Search_Agent, 2025.

[4] Foster D. Generative Deep Learning (2nd Edition). Australia & New Zealand: O'Reilly Media; 2023. ISBN: 9781098134181

[5] Roadmap for General-purpose Artificial Intelligence. https://chatgpt4online.org/openai-5-level-development-strategy-to-agi/, 2025.

[6] Lewis P, Perez E, Piktus A, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv preprint arXiv.2005.11401, 2021. doi: 10.48550/arXiv.2005.11401

[7] Nishimi K, Yoshida S, Oshima Y. Introduction to RAG and AI Agents [Practical] with LangChain and LangGraph. Japanese: Gijutsu hyoronsha. 2024. ISBN: 9784297145309

[8] Koyama T. Contribution of Phase-Field Method to Iron and Steel Science and Engineering. Ferrum. 2025; 30(7), 474–479. Japanese.

[9] Parker A. In the Blink of an Eye: How Vision Kick-Started the Big Bang of Evolution, London: Natural History Museum. 2017. ISBN: 9780565094003

[10] Karpas E, Abend O, Belinkov Y, et al. MRKL Systems: A Modular, Neuro-symbolic Architecture that Combines Large Language Models, External Knowledge Sources and Discrete Reasoning. arXiv preprint arXiv.2205.00445, 2022. doi:10.48550/arXiv.2205.00445

[11] Homepage of LangChain, "What is an AI agent?", https://blog.langchain.dev/what-is-an-agent/

[12] Compare Top 11 LLM Orchestration Frameworks in 2025. https://research.aimultiple.com/llm-orchestration/, 2025.

[13] Ni Z, Li Y, Hu K, et al. MatPilot: an LLM-enabled AI Materials Scientist under the Framework of Human-Machine Collaboration. arXiv preprint arXiv:2411.08063. doi: 10.48550/arXiv.2411.08063

[14] Nastase D. LangGraph: Managing AI Agents Systems using LangGraph.js. Independently pub. 2024. ISBN: 9798300423896

[15] Ollama. Ollama tool. https://ollama.com/, 2025.

[16] Lim G. Ollama Crash Course: Build Local LLM Powered Apps. Independently pub. 2025. ISBN: 9798311074919

[17] Yao S, Zhao J, Yu D, et al. ReAct: Synergizing Reasoning and Acting in Language Models. Conference paper at ICLR 2023, arXiv preprint arXiv.2210.03629, 2023. doi: 10.48550/arXiv.2210.03629

[18] Lukas HL, Fries SG, Sundman B. Computational Thermodynamics: The Calphad Method. Cambridge: Cambridge University Press. 2007. ISBN: 9780521868112

[19] NIMS Materials Database, MatNavi. https://mits.nims.go.jp/, 2025.

[20] Abe T. Computational Materials Design Engineering -Computational thermodynamics - (Supplemented new ed.). Japanese: Uchida Rokakuho. 2019. ISBN: 9784753659395

[21] Koyama T. Computational Materials Design Engineering -Microstructure Simulations - (Supplemented new ed.). Japanese: Uchida Rokakuho. 2019. ISBN: 9784753659401

[22] Koyama T, Takaki T. Introduction to Phase Field Method. Japanese: Maruzen, 2013. ISBN: 9784621086582

[23] Koyama T. Chapter 21 Phase Field Approach. In: Czichos H, Saito T, Smith LE eds. Springer Handbook of Metrology and Testing (2nd ed). Springer-Verlag. 2011. ISBN: 9783642166402

[24] Koyama T. Phase-field method, JSAP Rev. 2025; 250202. doi: 10.11470/jsaprev.250202

[25] Bishop CM, Bishop H. Deep Learning: Foundations and Concepts. Switzerland: Springer. 2023. ISBN: 9783031454677

[26] Miyazaki T, Takeuchi A, Koyama T, et al. Computer Simulation of Phase Decomposition in the Regular Solid Solution based upon the Cahn-Hilliard's Non-Linear Diffusion Equation. Mater. Trans. JIM. 1991; 32, 915–920. doi: 10.2320/matertrans1989.32.915

[27] GitHub homepage. https://github.com/, 2025.

[28] Jarzynski C. Nonequilibrium equality for free energy differences. Phys. Rev. Lett. 1997; 78(14), 2690–2693. doi: 10.1103/PhysRevLett.78.2690

[29] Crooks GE. Entropy production fluctuation theorem and the nonequilibrium work relation for free energy differences. Phys. Rev. E. 1999; 60(3), 2721–2726. doi:10.1103/PhysRevE.60.2721.

[30] Ichiyanagi M. Variational principles of irreversible processes. Physics Reports. 1994; 243(3), 125-182. doi: 10.1016/0370-1573(94)90052-3

[31] Metals data book (3rd edition). Japan Institute of Metals (Ed.), Maruzen Pub. 1993. ISBN: 4621038257C3057

[32] Anthropic homepage: Introducing the Model Context Protocol.
https://www.anthropic.com/news/model-context-protocol, 2025.
[33] Onda M, Otsubo Y. Introduction to MCP. Japanese: Shuwa system. 2025. ISBN: 9784798075730

Captions

Fig. 1 Explanation of the basic behavior of agent AI.

Fig. 2 Agent AI calculating Gibbs energy of Fe alloy.

Fig. 3 Example of Python code for an agent AI that calculates the Gibbs energy of the bcc phase in Fe alloys.

Fig. 3 (Continued)

Fig. 4 Tools used by the agent AI to handle various tasks in phase-field simulation. The python code is available in the supplementary material.

Fig. 5 (A) The simulation output files with different initial composition fields calculated all at once. (B) 2D simulation image of the microstructural changes obtained from one of these fields, i.e., the spinodal decomposition of A-50at%B alloy during isothermal aging at 1000 K: (a) $t' = 0$, (b) $t' = 4.0 \times 10^2$, (c) $t' = 6.0 \times 10^2$, (d) $t' = 8.0 \times 10^2$, (e) $t' = 1.2 \times 10^3$, and (f) $t' = 2.0 \times 10^3$.

Fig. 6 Temporal evolution of the coupling composition profiles (a)-(c), and the temporal change of free energy (d). Black and red curves are calculated from Gibbs energy equation and Jarzynski equation, respectively, where the two curves are almost overlapping.

Fig. 7 Examples of calculation to verify the validity of the free energy change of Fig. 6(d).

Table 1 Performance comparison of the qwen series based on the LLM size.

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

Supplementary material: The original python codes explained in the article are available online at https://doi.org/[DOI of this paper]. The same python codes can also be downloaded from GitHub homepage of https://github.com/ts-koyama/CMD-project. Since the LangChain library is updated frequently, the latest version of python codes in this article are available from the GitHub homepage.

Funding: This research was partially supported by the JST SIP (“Materials Integration” for Revolutionary Design System of Structural Materials), MEXT Data Creation and Utilization Materials Research and Development Project (Data Creation and Utilization Magnetic Materials Research Center JPMXP1122715503), and JST-CREST (JPMJC22C3).

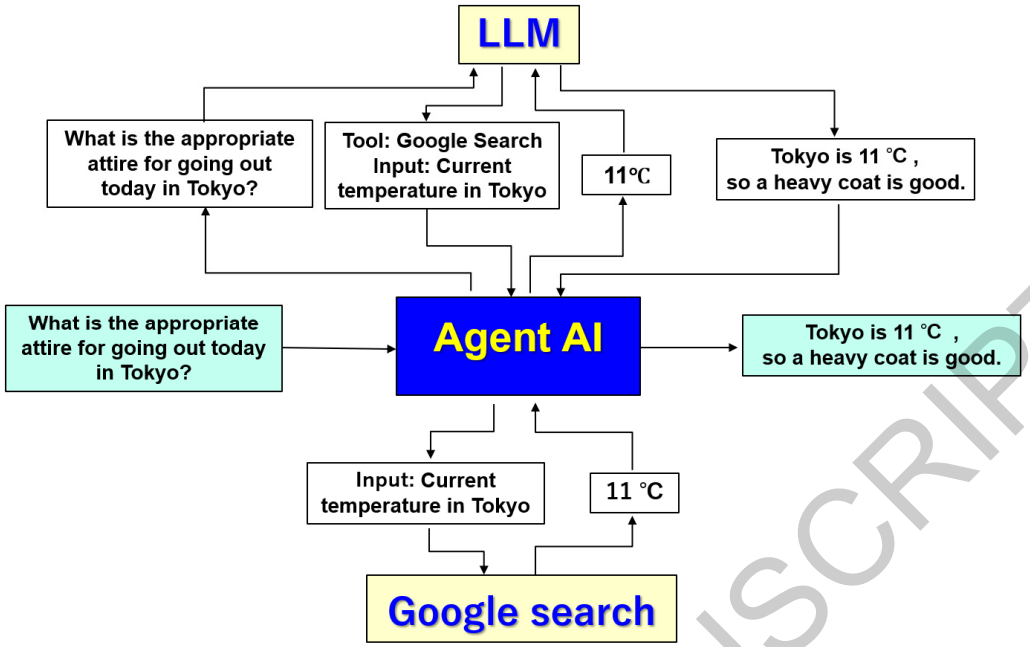


Fig. 1 Explanation of the basic behavior of agent AI.

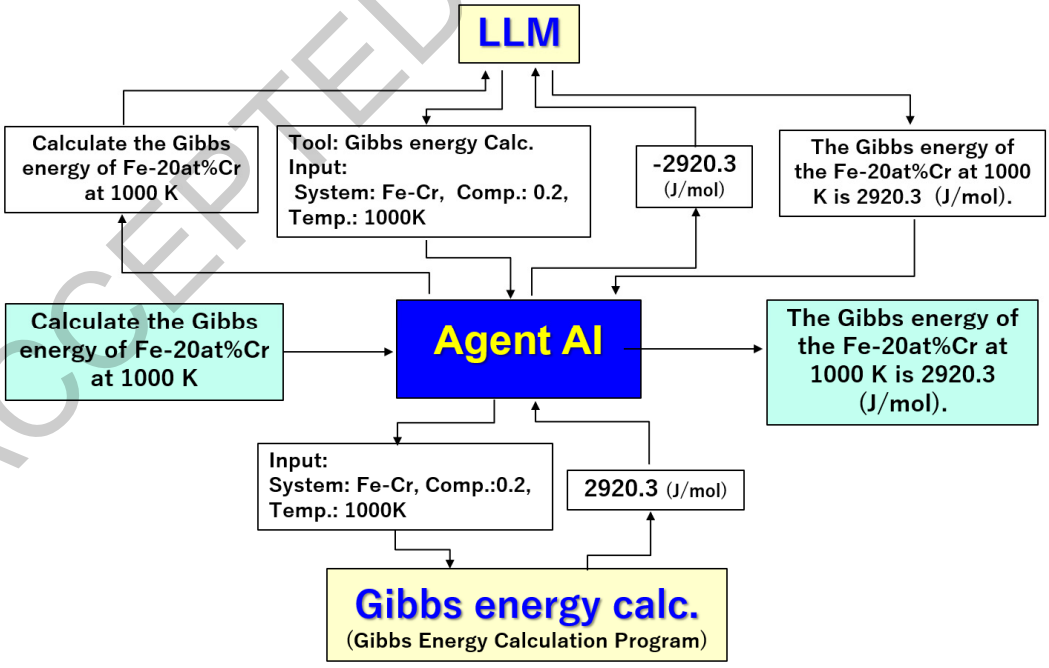


Fig. 2 Agent AI calculating Gibbs energy of Fe alloy.

(A) Definition of libraries

```
import os
import requests
import numpy as np
from langchain_ollama import ChatOllama                    # LangChain and LangGraph libraries
from langchain_core.tools import tool
from langgraph.prebuilt import create_react_agent
from langchain.schema import AIMessage
```

(B) Gibbs energy of Fe-Cr alloy calculated by an extended regular solution model [18-20]

```
# Definition of tools
@tool
def Fe_Cr_gibbs_energy(temp1: float, comp1: float) -> float:
    """
    As for Fe-Cr binary alloy, receive float type temperature and float type alloy composition
    as input parameters, and return the calculated float type Gibbs energy in J/mol.

    Args:
        temp1 (float): temperature, a float type input parameter, unit is K.
        comp1 (float): alloy composition, a float type input parameter, atomic fraction of Cr.
    Returns:
        Gc (float):Gibbs energy calculation result, a float type, unit is J/mol
    """
    T=temp1; c=comp1

    RR = 8.3145 # Gas constant [J／（mol K）]
    p_mag=0.4; D_mag=518./1125.+11692./15975.*(1./p_mag-1.)

    c1=1.-c; c2=c
    L0_FeCr=20500.0-9.68*T
    gc1=L0_FeCr*c1*c2+RR*T*(c1*np.log(c1)+c2*np.log(c2))
    # We set Gc_Fe_bcc=Gc_Cr_bcc=0.

    Tc=1043.*c1-311.5*c2+c1*c2*(1650.-550.*(c1-c2))
    if Tc<0.: Tc=-Tc
    Bc=2.22*c1-0.008*c2-0.85*c1*c2
    if Bc<0.: Bc=-Bc

    tau=T/Tc
    if tau<=1.:
        ftau=1.-1./D_mag*(79./140./p_mag/tau+474./497.*(1./p_mag-1.)*( np.power(tau,3.)/6.
                                                    +np.power(tau,9.)/135.+np.power(tau,15.)/600.) )
    else:
        ftau=-1./D_mag*(np.power(tau,-5.)/10.+np.power(tau,-15.)/315.+np.power(tau,-25.)/1500.)

    gmag=RR*T*np.log(1.+Bc)*ftau
    Gc=gc1+gmag
    return Gc
```

Fig. 3 Example of Python code for an agent AI that calculates the Gibbs energy of the bcc phase in Fe alloys.

(C) Gibbs energy of Fe-C alloy calculated by a sub-lattice model for interstitial site [18-20]

```
# Definition of tools
@tool
def Fe_C_gibbs_energy(temp1: float, comp1: float) -> float:
    """
    As for Fe-C binary alloy, receive float type temperature and float type alloy composition
    as input parameters, and return the calculated float type Gibbs energy in J/mol.

    Args:
        temp1 (float): temperature, a float type input parameter, unit is K.
        comp1 (float): alloy composition, a float type input parameter, atomic fraction of C.
    Returns:
        Gc (float):Gibbs energy calculation result, a float type, unit is J/mol
    """
    T=temp1; c=comp1

    RR = 8.3145 # Gas constant [J／（mol K）]
    p_mag=0.4; D_mag=518./1125.+11692./15975.*(1./p_mag-1.)

    GHserCC=-17368.441+170.73*T-24.3*T*np.log(T)-4.723e-04*T*T+2562600./T-2.643e+08/T/T+1.2e+10/T/T/T
    GHserFe=1225.7+124.134*T-23.5143*T*np.log(T)-.00439752*T*T-5.8927e-08*T*T*T+77359./T
    GBccFe_Va=GHserFe
    GBccFe_C=322050.+75.667*T+GHserFe+3.*GHserCC
    LFe_CVa_0=-190.*T

    BMBccFe_C=2.22;     BMBccFe_Va=2.22
    TCBccFe_C=1043.;    TCBccFe_Va=1043.

    yC=1./3.*c/(1.-c);    yVa=1.-yC

    Tc=yVa*TCBccFe_Va+yC*TCBccFe_C
    Bc=yVa*BMBccFe_Va+yC*BMBccFe_C

    tau=T/Tc
    if tau<=1.:
        ftau=1.-1./D_mag*(79./140./p_mag/tau+474./497.*(1./p_mag-1.)*( np.power(tau,3.)/6.
                                                    +np.power(tau,9.)/135.+np.power(tau,15.)/600.))
    else:
        ftau=-1./D_mag*(np.power(tau,-5.)/10.+np.power(tau,-15.)/315.+np.power(tau,-25.)/1500.)

    gmag=RR*T*np.log(Bc+1.)*ftau

    Gc_bcc=yVa*GBccFe_Va+yC*GBccFe_C+yVa*yC*LFe_CVa_0+3.*RR*T*(yVa*np.log(yVa)+yC*np.log(yC))+gmag
    Gc=Gc_bcc/(1.+3.*yC)
    return Gc
```

(D) Definition of the agent implemented by tools and LLM

```
tools = [Fe_Cr_gibbs_energy, Fe_C_gibbs_energy]   # Declaration of tools
LLM = "qwen3:8b"                                  # LLM name
model = ChatOllama(                               # LLM model definition
    model=LLM,
    temperature=0
)
agent = create_react_agent(                       # Definition of the agent implemented by tools and LLM
    model=model,
    tools=tools
)
```

(E) Query and answer

```
# Run the agent
messages = {
    "messages": [{"role": "user", "content": """Calculate the Gibbs energy of an Fe-C alloy
                                                with a composition 0.03 C at 1000 K."""}]    # Query
}
answer = agent.invoke(messages)                                                              # Answer

# ---- Parsing the response message ----------------------------------------------------------------------------------------------------
messages_list = answer["messages"] # Get the messages list
filtered_messages = [msg.content for msg in messages_list if isinstance(msg, AIMessage)] # Extract assistant messages only.
formatted_output = "¥n- " + "¥n- ".join(filtered_messages) # Formatted and output
print(formatted_output)
```

Fig. 3 (Continued)

Four tools useful for handling phase-field simulation

(1) Check_phase_field_simulation(aging_temp1: float, alloy_comp1: float, time_step1: float)

- ◆ Simulate the spinodal decomposition and display the simulated microstructures currently being calculated.
- ● Input parameters: [alloy composition, aging temperature, time step]

(2) Save_phase_field_simulation(index: int, aging_temp1: float, alloy_comp1: float, time_step1: float)

- ◆ Set initial composition field distinguished by index number, then save the microstructure data calculated during the simulation.
- ● Input parameters: [index number, alloy composition, aging temperature, time step]

(3) Load_data_display_images(Fname: str)

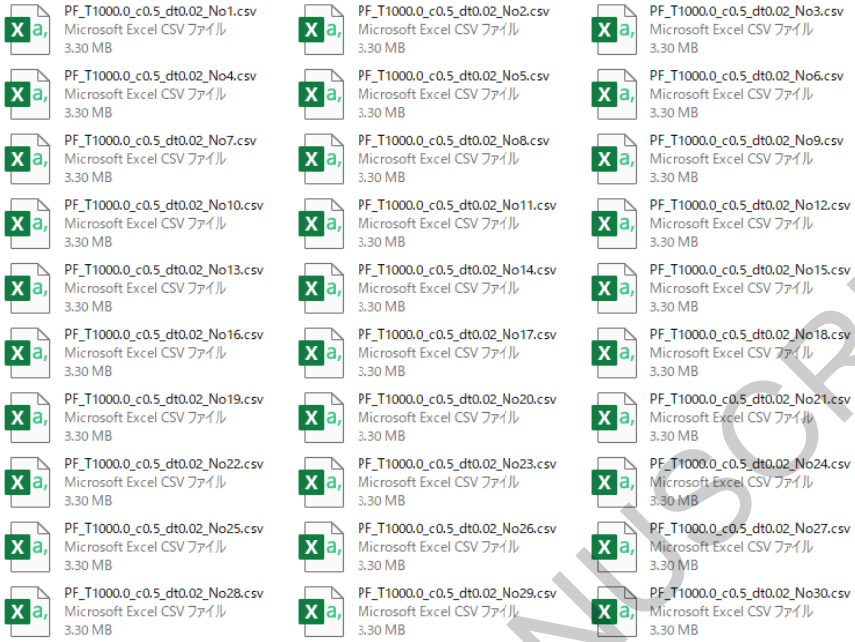
- ◆ Display the microstructures which have already been simulated.
- ● Input parameter: [file name]

(4) Load_data_save_images(Fname: str)

- ◆ Display the microstructures and save images of each microstructure to disk in image file format.
- ● Input parameter: [file name]

Fig. 4 Tools used by agent AI to handle various tasks in phase-field simulation. The python code is available in the supplementary material.

(A)



(B)

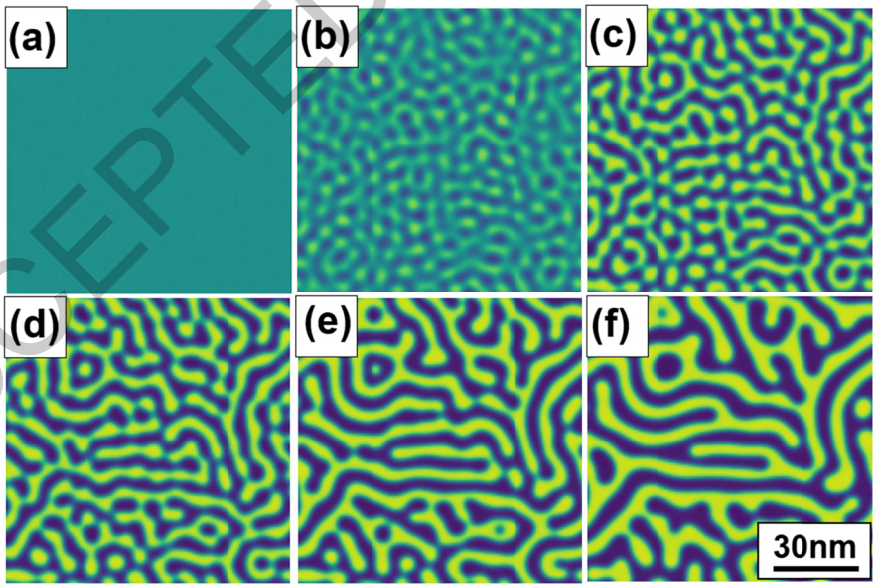


Fig. 5 (A) The simulation output files with different initial composition fields calculated all at once. (B) 2D simulation image of the microstructural changes obtained from one of these fields, i.e., the spinodal decomposition of A-50at%B alloy during isothermal aging at 1000 K: (a) $t' = 0$, (b) $t' = 4.0 \times 10^2$, (c) $t' = 6.0 \times 10^2$, (d) $t' = 8.0 \times 10^2$, (e) $t' = 1.2 \times 10^3$, and (f) $t' = 2.0 \times 10^3$.

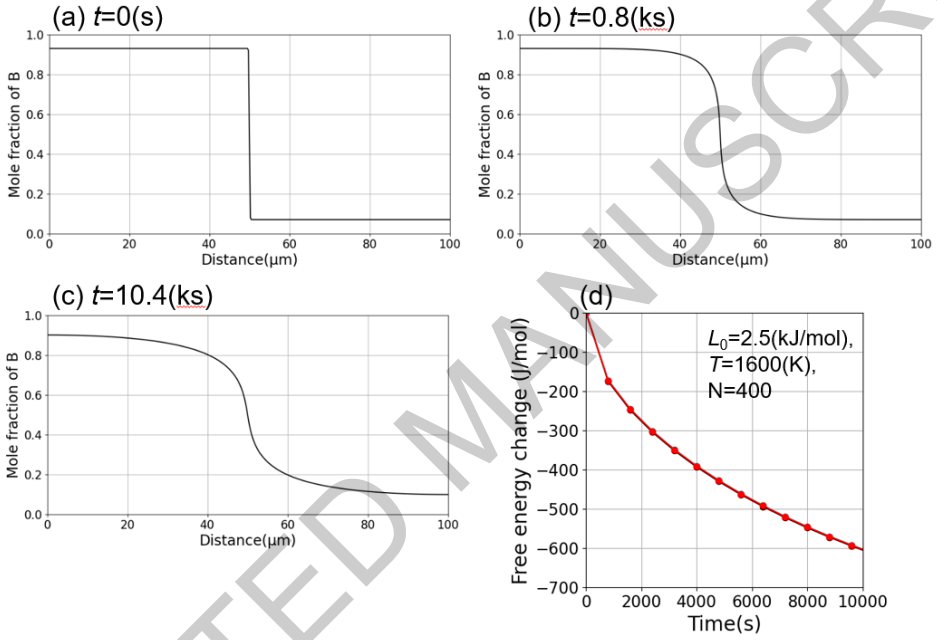


Fig. 6 Temporal evolution of the coupling composition profiles (a)-(c), and the temporal change of free energy (d). Black and red curves are calculated from Gibbs energy equation and Jarzynski equation, respectively, where the two curves are almost overlapping.

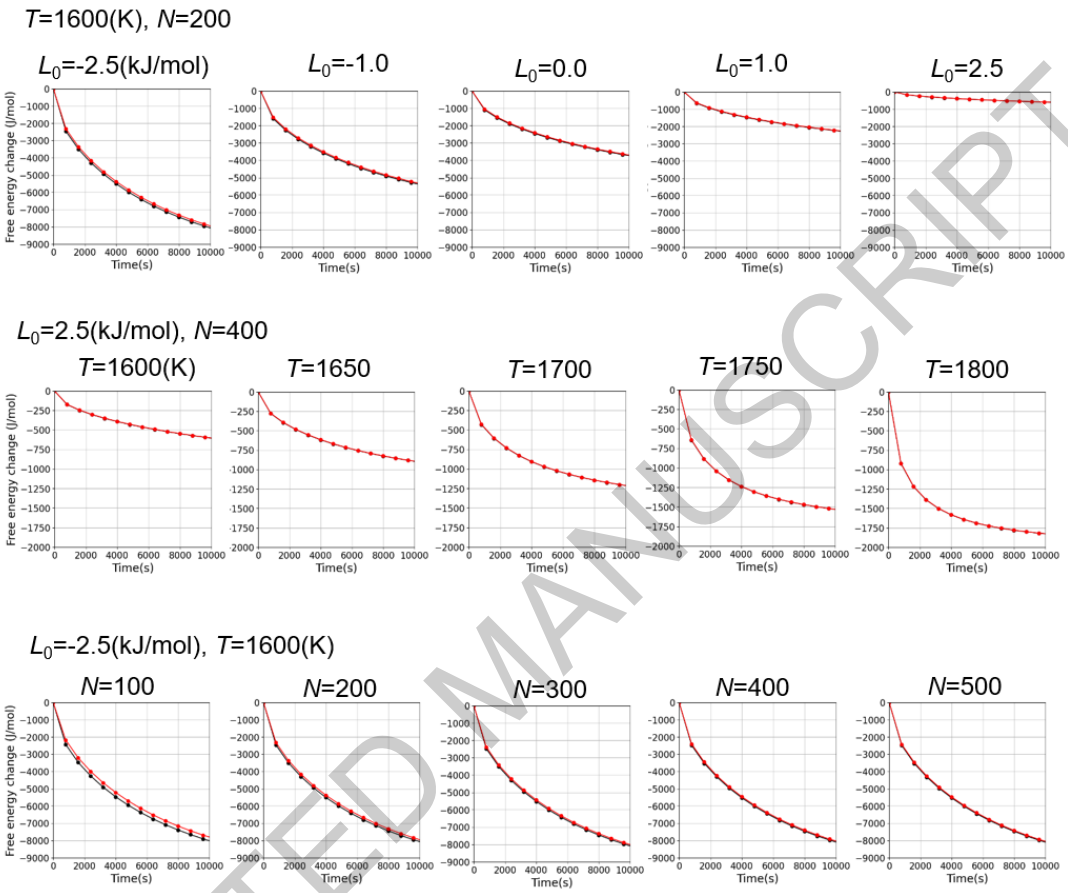
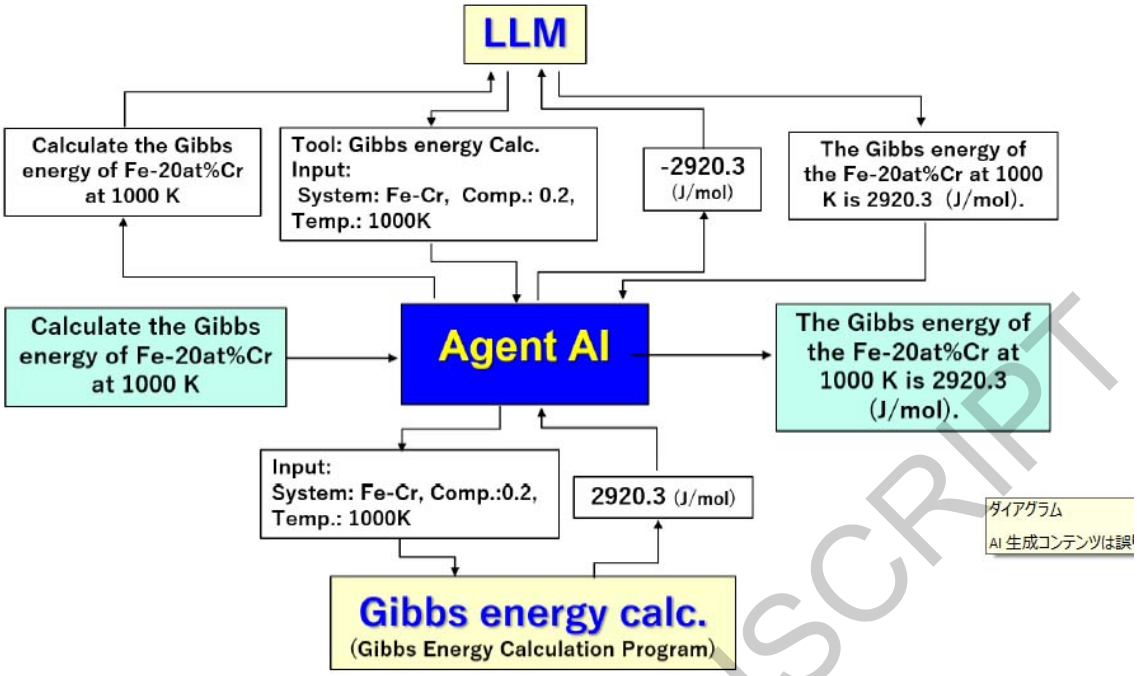


Fig. 7 Examples of calculations to verify the validity of the free energy change of Fig. 6(d).

Table 1 Performance comparison of the qwen series based on the LLM size.

	qwen3:0.6b	qwen3:1.7b	qwen3:4b	qwen3:8b	qwen3:14b	qwen3:30b
Example 1	○	○	○	○	○	○
Example 2	○	○	○	○	○	○
Example 3	△	○	○	○	○	○
Example 4	×	△	△	○	○	○



Using Gibbs energy calculations and diffusion simulations as examples, we demonstrated the implementation method and usefulness of simple agent AI, where sample python codes are available as supplemental materials.

Graphical Abstract