PAPER

ProtAgents: protein discovery *via* large language model multi-agent collaborations combining physics and machine learning†

Alireza Ghafarollahi[a] and Markus J. Buehler [ID] *[ab]

Designing *de novo* proteins beyond those found in nature holds significant promise for advancements in both scientific and engineering applications. Current methodologies for protein design often rely on AI-based models, such as surrogate models that address end-to-end problems by linking protein structure to material properties or *vice versa*. However, these models frequently focus on specific material objectives or structural properties, limiting their flexibility when incorporating out-of-domain knowledge into the design process or comprehensive data analysis is required. In this study, we introduce ProtAgents, a platform for *de novo* protein design based on Large Language Models (LLMs), where multiple AI agents with distinct capabilities collaboratively address complex tasks within a dynamic environment. The versatility in agent development allows for expertise in diverse domains, including knowledge retrieval, protein structure analysis, physics-based simulations, and results analysis. The dynamic collaboration between agents, empowered by LLMs, provides a versatile approach to tackling protein design and analysis problems, as demonstrated through diverse examples in this study. The problems of interest encompass designing new proteins, analyzing protein structures and obtaining new first-principles data – natural vibrational frequencies – *via* physics simulations. The concerted effort of the system allows for powerful automated and synergistic design of *de novo* proteins with targeted mechanical properties. The flexibility in designing the agents, on one hand, and their capacity in autonomous collaboration through the dynamic LLM-based multi-agent environment on the other hand, unleashes great potentials of LLMs in addressing multi-objective materials problems and opens up new avenues for autonomous materials discovery and design.

1 Introduction

Proteins, the building blocks of life, serve as the fundamental elements of many biological materials emerging from natural evolution over the span of 300 million years. Protein-base biomaterials like silk, collagen and tissue assemblies such as skin exhibit diverse structural features and showcase unique combinations of material properties. The underlying sequences of amino acids (AAs) in a protein determines its unique there-dimensional structure, which, in turn, dictates its specific biological activity and associated outstanding properties. This inherent relationship has inspired scientists

[a]Laboratory for Atomistic and Molecular Mechanics (LAMM), Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139, USA. E-mail: mbuehler@MIT.EDU

[b]Center for Computational Science and Engineering, Schwarzman College of Computing, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139, USA

† Electronic supplementary information (ESI) available: The full records of different conversation experiments along with additional materials are provided as supplementary materials. See DOI: https://doi.org/10.1039/d4dd00013g

in the field of materials design and optimization to draw valuable insights from nature for creating novel protein-based materials. The diversity in protein design is immense, with over 20^{100} possible AA sequences for just a relatively small 100-residue protein. However, the natural evolutionary process has sampled only a fraction of this vast sequence space. This leaves a substantial portion uncharted, presenting a significant opportunity for the *de novo* design of proteins with potentially remarkable properties.[1] Despite this potential, the extensive design space, coupled with the costs associated with experimental testing, poses formidable challenges in *de novo* protein design. Navigating this intricate landscape necessitates the development of a diverse set of effective tools enabling the targeted design of *de novo* proteins with specific structural features or properties.

Over the past years, in the field of *de novo* protein design, data-driven and machine learning methods have emerged as powerful tools, offering valuable insights and accelerating the discovery of novel proteins with desired properties.[2–15] These methods have opened great avenues for predicting structure, properties, and functions of proteins solely based on their

underlying AA sequence. For instance, the development of deep learning (DL)-based AlphaFold 2 marked a significant breakthrough in the field of 3D folding protein prediction with a level of accuracy that in some cases rivaled expensive and time-consuming experimental techniques.[16] Moreover, deep learning-based models have been developed to explore structure–property relationships in the analysis and design of proteins. These models encompass a broad spectrum of structural and mechanical properties, serving either as constraints or target values. For example, various DL-models developed predict the secondary structure of proteins from their primary sequences. Prediction of mechanical properties of spider silk protein sequences have been enabled by DL models.[17–22] Moreover, DL-based models such as graph neural networks[23] and transformer-based language models[24] show enhanced accuracy in predicting the protein natural frequencies compared to physics-based all-atom molecular simulations. The development of such DL models significantly reduces the cost of screening the vast sequence space to target proteins with improved or optimized mechanical performance.

A frontier, however, that still exists is how we can create intelligent tools that can solve complex tasks and draw upon a diverse set of knowledge, tools and abilities. Another critical issue is that the combination of purely data-driven tools with physics-based modeling is important for accurate predictions. Moreover, such tools should ideally also be able to retrieve knowledge from, for instance, the literature or the internet. All these aspects must be combined in a nonlinear manner where multiple dependent steps in the iteration towards and answer are necessary to ultimately provide the solution to a task. As we will discuss in this study, such an integration of tools, methods, logic, reasoning and iterative solution can be implemented through the deployment of a multi-agent system driven by sophisticated Large Language Models (LLMs).

LLMs[25,26] have represented a paradigm shift in modeling problems across a spectrum of scientific and engineering domains.[8,27–41] Such models, built upon attention mechanism and transformer architectures,[42] have emerged as powerful tools recently in the field of materials science and related areas, contributing to various aspects ranging from knowledge retrieval to modeling, design, and analysis. For example, models such as ChatGPT and the underlying GPT-4 architecture,[43] part of the Generative Pretrained Transformer (GPT) class, demonstrate exceptional proficiency in mastering human language, coding,[44] logic and reasoning.[45] Recent studies highlight their ability to proficiently program numerical algorithms and troubleshoot code errors across several programming languages like Python, MATLAB, Julia, C, and C++.[46] The GPT class of LLMs has also represented a new paradigm in simulating and predicting the materials behavior under different conditions,[28] a field of materials science often reserved for conventional deep learning frameworks[47] such as Convolutional Neural Networks,[48,49] Generative Adversarial Networks,[50–52] Recurrent Neural Networks22,54,55 (ref. 20, 53 and 54), and Graph Neural Networks.[23,55–58] Moreover, due to

their proficiency in processing and comprehending vast amount of different types of multimodal data, LLMs show promising capabilities in materials analysis and prediction application including key knowledge retrieval,[35] general language tasks, hypothesis generation,[29] and structure-to-property mapping.[28,59]

At the same time, LLMs are typically not best equipped to solve specific physics-based forward and inverse design tasks, and are often focused on leveraging their conversational capabilities. Here, LLMs have been instrumental in powering conversable AI agents, facilitating the transition from AI–human conversations to AI–AI or AI–tools interactions for increased autonomy.[31,35,60–62] This capability represents a significant advancement, enabling intelligent mediation, fostering interdisciplinary collaboration, and driving innovation across disparate domains, including materials analysis, design, and manufacturing. The overall process could be deemed as adapting a problem-solving strategy dictated and directed by the AI system comprised of different agents. Thereby, the entire process can be AI automated with reduced or little human intervention. Depending on the complexity of the problem, using the idea of labor division, the agents have the capability to break the overall task into subtasks for which different agents or tools are used consecutively to iteratively solve the problem until all subtasks have accomplished and the solution has achieved. There is no intrinsic limitation in defining the type of tools, making the multi-agent model a versatile approach in addressing problems across scales and disciplines. The tools could range from a simple linear mathematical function to sophisticated deep neural network architectures. The use of llm-based multi-agent strategies for discovering new materials and automating scientific research has been examined in previous studies, with applications spanning mechanics,[31] chemistry,[63,64] and materials science.[35] The comprehensive analysis by[65] specifically addresses the implementation of multi-agent strategy in enhancing biomedical research and scientific discovery, underscoring their potential to transform the field.

In this paper, we propose a multi-agent strategy to the protein design problems by introducing ProtAgents, a multi-agent modeling framework to solve protein-related analysis and design problems by leveraging customized functions across domains and disciplines. The core underpinning concept of the multi-agent systems is the state-of-the-art LLMs combined with a series of other tools. The LLM backbone demonstrates exceptional abilities in analysis, rational thinking, and strategic planning, essential for complex problem-solving. Leveraged by these capabilities, the proposed model aims to reduce the need for human intervention and intelligence at different stages of protein design. The agent model consists a suite of AI and physics-based components such as:

● Physics simulators: obtain new physical data from simulations, specifically normal modes and vibrational properties by solving partial differential equations (PDEs).

● Generative AI model: conditional/unconditional *de novo* protein design, based on a denoising diffusion model.

● Fine-tuned transformer model: predict mechanical properties of proteins from their sequence.

● Retrieval agent: retrieve new data from a knowledge database of scientific literature.

The main contribution of our work is summarized as follows.

● We propose ProtAgents, a pioneering multi-agent modeling framework that combines state-of-the-art LLMs with diverse tools to tackle protein design and analysis problems.

● Our model harnesses the collective capabilities of agents with specialized expertise that interact autonomously and nonlinearly to solve the protein-related task.

● Equipped with various tools and functions, the model demonstrates an advanced ability to integrate new physical data from different disciplines, surpassing conventional deep learning models in versatility and problem-solving capacity in protein science.

● Our model significantly minimizes the need for human interference throughout different stages of the problem-solving process.

● ProtAgents operates on textual input, thereby enabling non-expert researchers to effectively address and analyze challenges within the realm of protein design.

The versatility of the approach in solving complex tasks is exhibited by providing a series of experiments in the context of proteins design, modeling, and data analysis.

The plan of this paper is as follows. In Section 2, we present an overview of the multi-agent framework developed to tackle multi-objective complex tasks. Subsequently, we delve into a series of experiments where each task is initially introduced, followed by a detailed examination of various aspects throughout the problem-solving process by the multi-agent teamwork. A comprehensive discussion regarding the multi-agent framework and future prospects is provided in Section 3.

2 Results and discussion

We present a series of computational experiments aimed at evaluating the effectiveness and potential of a multi-agent modeling framework for various challenges within the domain of protein modeling, design, and analysis. The multi-agent framework consists of a team of agents, each powered by a state-of-the-art general purpose large language model, GPT-4,[43] accessed *via* the OpenAI API[66] and characterized by a unique profile that details its role, and communication protocols, such as sharing information and engaging with humans *via* language as shown in Fig. 1a. Furthermore, agents are given access to a set of tools with various functionalities across domains. As shown in Fig. 1b each function is characterized by a descriptive profile and input parameters. The outline of the proposed multi-agent framework is shown in Fig. 1c, illustrating the collaborative efforts of a team of agents with the following entities.

● "User": human that poses the question.

● "Planner": develops a plan to solve the task. Also suggests the functions to be executed.

● "Assistant": who has access to all the customized functions, methods, and APIs and executes them to find or compute the relevant data necessary to solve the task.

● "Critic": responsible for providing feedback about the plan developed by "planner" as well as analyzing the results and

handling the possible mistakes and providing the output to the user.

The agents are organized into a team structure, overseen by a manager who coordinates overall communication among the agents. Table 1 lists the full profile for the agents recruited in our multi-agent framework. Moreover, a generic structure showing the dynamic collaboration between the team of agents proposed in the current study is depicted in Fig. 2. Further details can be found in the Materials and methods section 4.

It is noteworthy that critical issues in the realm of protein design surpass the capabilities of mere Python code writing and execution. Instead, addressing these challenges necessitates the utilization of external tools specifically tailored for protein design and analysis, and the writing, adaptation, correction and execution of code depends nonlinearly on the progression of the solution strategy that is developed by the system.

The tools are incorporated into the model *via* the assistant agent who oversees executing the tools. To assess the performance of the multi-agent framework in handling complex interdisciplinary tasks, we have defined a rich library of functions each with special powers in solving the protein problems. Each function has a distinct profile that describes its role and requires one or more entities as inputs, each of which is also profiled to specify its identity and type such as string or integer. This helps the agents not only understand which function to choose but also how to provide the input parameters in the correct format. The functions provide the ability to, for instance, retrieve knowledge, perform protein folding, analyze the secondary structure, and predict some parameters through a pre-trained autoregressive language model. Additionally, a function can carry out simulations to compute the protein natural frequencies, thus allowing the model to integrate the new physics-based data. A full list of functions implemented in the current study is provided in Table S1 in the ESI.† It is worth mentioning that all the tools implemented in our multi-agent system are fixed, predefined functions, and the agents have not been given the ability to modify them.

Given the complexities residing in the protein design problems, the primary contribution of our multi-agent framework lies in assessing whether the team of agents can discern the requisite tools for a given query and evaluating the framework's capability to initiate the execution of these tools, along with providing the necessary inputs. The designated tasks are intentionally designed to be sufficiently complex, involving multiple subtasks where, for some cases, the execution of each depends on the successful completion of the preceding ones. This design showcases the model's capacity for the automated handling of intricate tasks, eliminating or substantially reducing the need for human intervention. Although the multi-agent framework allows for the human intervention at different stages, we skip that to further examine the team's capability in handling different possible situations, for instance in case of a failure.

2.1 Experiment I: knowledge retrieval, computations, and analysis

As the first example, we pose a task which involves a multi-step knowledge retrieval, analysis, and computations for a set of
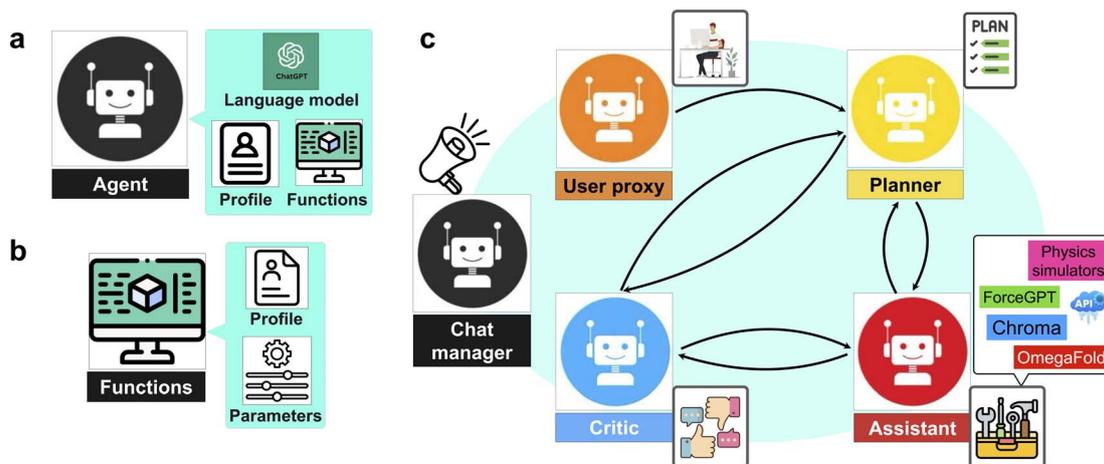
Fig. 1 Multi-agent AI framework for automating protein discovery and analysis. (a) A genetic agent structure in a multi-agent modeling environment that can communicate *via* language, has a focus defined by a profile, and has access to custom functions. (b) A function is customized by a profile and a set of parameters. (c) The structure of a team of agents, each with special expertise, that communicate to each other and allow for mutual correction and a division of labor. Given different profiles for each agent, agents are designed that are expert on describing the problem (*user_proxy*), plan making (*planner*), function executing (*assistant*), and result evaluation (*critic*). The whole process is automated *via* a dynamic group chat under the leading chat manager, offering a versatile approach in solving challenging tasks in the context of protein design and analysis without human intervention.

protein PDB identifiers. We start the first round of conversation by posing the following question:

> **user_proxy**
>
> Can you provide examples of protein names whose mechanical properties have been studied through experiments?

The planer then correctly suggests the function "retrieve_content" to be executed with the argument "examples of protein names whose mechanical properties have been studied through experiments". Upon execution of the function, the assistant provides us with a list of protein names. Upon inspection, we find that the agent has successfully identified experimentally studied proteins, despite an abundance of information on proteins studied theoretically, for instance, through coarse-grain simulations. Since we are interested in the PDB ids, we continue the chat by a follow-up question "Can you provide me with the PDB ids for these proteins?" when "user_proxy" is being asked to provide feedback to chat manager. Again, the planner suggests "retrieve_content" function with the following message.

> **Planner**
>
> ***** Suggested function Call: retrieve_content *****
>
> Arguments:
> {
> "message": "PDB ids for I27 domain from titin, fibronectin type III domains, E2lip3, T4 lysozyme, ubiquitin, titin immunoglobulin domains, cell adhesion molecule, spectrin repeats, and tenascin"
> }

The "*Assistant*" agent then calls the function and gives the following output:

> **Assistant**
>
> Here are the PDB IDs for the proteins you mentioned:
>
> 1. I27 domain from titin: 1wit
> 2. Ubiquitin: 1ubq
> 3. T4 lysozyme: 1nct
> 4. Titin immunoglobulin domains: 1tit, 1qjo
> 5. E2lip3: 1hz6/2ptl
> 6. Tenascin: 1ten
> 7. Spectrin repeats: 1emb
> 8. FnIII: 1g1c, 1fnh, 1vsc, 1cfc, 1fnf/1ttf/1ttg
> 9. Cell adhesion molecule: 1bni/1bnr, 1b6i, 1rsy/1dqv, 1aj3, 1ksr/1whl, 1u4q, 1hci, 1n11

Upon careful examination of the results, we observe that, despite all the PDB ids exist in the source database, the PDB ids do not quite match with the protein names except for a few cases (1ubq, 1ten). Nevertheless, note that the error is caused by the poor performance of the "*retrieve_content*" function, which implements Llama index, and the team of agents cannot mitigate that as they have no access to the knowledge database. In fact, the entire retrieval augmented generation process is solely performed by Llama index and the agents merely contribute to this process by providing the query, calling the "*retrieve_content*" function, and returning the results. As such, we continue to test the agent-team capability in more challenging queries centered around computational tasks and physics-based simulations by assigning the following task in the next round of conversation.

Table 1 The profiles of the agents implemented in the current study to solve multi-objective tasks in the context of protein design and analysis

Agent#	Agent role	Agent profile
1	user_proxy	user_proxy. Plan execution needs to be approved by user_proxy
2	Planner	Planner. You develop a plan. Begin by explaining the plan. Revise the plan based on feedback from the critic and user_proxy, until user_proxy approval. The plan may involve calling custom function for retrieving knowledge, designing proteins, and computing and analyzing protein properties. You include the function names in the plan and the necessary parameters. If the plan involves retrieving knowledge, retain all the key points of the query asked by the user for the input message
3	Assistant	Assistant. You have access to all the custom functions. You focus on executing the functions suggested by the planner or the critic. You also have the ability to prepare the required input parameters for the functions
4	Critic	user_proxy. You double-check the plan, especially the functions and function parameters. Check whether the plan included all the necessary parameters for the suggested function. You provide feedback
5	Group chat manager	You repeat the following steps: dynamically selecting a speaker, collecting responses, and broadcasting the message to the group
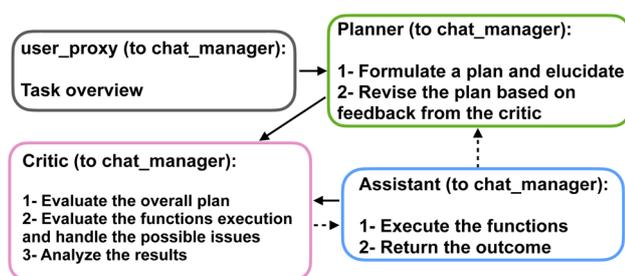
Fig. 2 A generic flowchart showing the dynamic interaction between the multi-agent team members organized by the group chat manager to solve protein design and analysis problems. The manager selects the working agents to collaborate in the team work based on the current context of the chat, thus forming close interactions and enabling mutual corrections.

user_proxy

For the first 10 protein PDB ids in the list, if the AA sequence length of the protein is less than 128, do the following: (a) analyze the secondary structure of the protein, (b) compute its first 13 natural frequencies, (c) determine its structure classification.

The above is a complex multi-step analysis and computation task encompasses aspects such as secondary structures analysis, natural frequency calculations, and structure classification. Additionally, the task is subject to an initial condition that must be satisfied before proceeding through the next sequence of steps, adding an extra layer of complexity. In response, the planner comes up with a detailed plan which consists of all the actions that need to be taken to complete the task. Moreover, the plan mentions all the corresponding functions that need to be executed to accomplish the task. More importantly, the *"planner"* perfectly realizes to fetch the protein structures first before starting to analyze the secondary structure, although it was not explicitly mentioned in the task query.

The teamwork proceeds by a follow-up feedback provided by the *"critic"* agent about all the plan steps and functions which is concluded by the following statement.

critic

Overall, the plan rightly identifies all the necessary functions and respective parameters for each step. It illustrates a good understanding of what needs to be done to fulfill the user request. Good job!

Therefore, the positive feedback from the *"critic"* further supports the good performance of the planner in addressing all the critical steps required to accomplish the tasks.

The *"assistant"* agent then follows the plan by calling and executing the corresponding functions, starting with AA length calculation, until all the steps have been undertaken. The results show that all the inputs to the functions are properly identified and provided and the functions are executed without any error. The conditional statement included in the tasks is also correctly satisfied for each protein, that is the computations are conducted only if the sequence length is less than 128 and are emitted otherwise. For instance, for the protein with PDB id "1hz6" the AA length is returned as 216 by the "assistant" which is then followed by the following message from the *"critic"*

critic

The length of the amino-acid sequence for the PDB id '1hz6' is 216, which is greater than 128. Therefore, we will not perform the steps (a) analyze the secondary structure of the protein, (b) compute its first 13 natural frequencies, and (c) determine its structure classification for the '1hz6'.

Moving to the next PDB id in the list...

After completion of all the tasks, the assistant returns a summary of all the results for each protein as representatively shown below for PDB id "1wit":

Assistant

For PDB id '1wit':
- It has an amino acid sequence length of 93.
- The analysis of its secondary structure yielded the following composition: H: 0%, B: 3.23%, E: 51.61%, G: 3.23%, I: 0%, T: 13.98%, S: 5.38%, P: 0%, -: 22.58%.
- Its first 13 natural frequencies are [4.3755, 5.0866, 5.5052, 6.7967, 7.908, 8.1947, 9.0166, 9.8528, 11.0632, 11.3968, 11.7355, 12.1279, 12.3498].
- Its CATH classification is 2.60.40.10.

The results suggest that the framework effectively retains all outputs, demonstrating its strong memory even in the face of diverse and extended results. As the last round of conversation, we ask to save all the results which allows us to load them at later time for other purposes:

user_proxy

Save all the results into a csv file where each row corresponds to a protein and each column represents a protein property, utilizing proper identifiers.

In response, the planner suggests to call the python function "*save_to_csv_file*". The main task here is to generate the dictionary of results in JSON and with appropriate structure as instructed by the user. However, we see that upon generating the JSON data and inputting it into the function by the "*assistant*" agent, the following error occurs:

Assistant

***** Response from calling function "save_to_csv_file" *****

Error: Expecting ',' delimiter: line 1 column 141 (char 140)

Without any human intervention, the agent team is able to resolve the issue by mutual correction. In particular, the "*critic*" identifies the cause of error by writing.

Critic

It seems like there is an error in the JSON format being used to feed data into the csv conversion function. The JSON string appears to be improperly formed leading the function to fail. The values for "Secondary Structure" and "First 13 Frequencies" are stringified JSON objects themselves, which maybe causing

Guided by the feedback from the *critic*, the "*assistant*" then reconstructs the JSON file from the output results and is able to successfully execute the function and thus save the results in a csv file as shown in Table 2. The complete group chat records can be found in Table S2 of the ESI.†

The main outcomes of this experiment are as follows.

● This experiment exemplifies how multi-agent strategy enables us to go beyond the knowledge of pre-trained LLMs by retrieving new knowledge from external sources.

● This experiment shows an instructive collaboration between the AI–AI agents in problem-solving as well as error handling.

● Multi-agent systems enable us to provide feedback at various stages and pose follow-up questions throughout the process.

● The experiment highlights the failure of the model in extracting correct knowledge from external sources. As discussed above, the last point directly stems from the failure of the function responsible for knowledge retrieval. To circumvent this, two main strategies can be adopted: (a) we can guide the agents to provide feedback on the reliability of the generated content, although this may be limited by the constraints of the pre-trained language models' knowledge; (b) we can implement more advanced Retrieval-Augmented Generation (RAG) models. Since knowledge retrieval is not the main focus of this paper, we will leave this aspect for future work.

2.2 Experiment II: *de novo* protein design using chroma

An important characteristic of the multi-agent model is its capability in handling very complex tasks in the context of protein design and analysis. This partly stems from the possibility of incorporating customized functions in the model for various purposes from knowledge retrieval to performing physics-based simulations. In this example, we explore the model's performance in using the state-of-the art models in *de novo* protein design, Chroma,[15] and protein folding, Omega-Fold.[4] Chroma offers a unified approach for protein design that can directly sample novel protein structures and sequences with the possibility to direct the generative design towards the desired functions and properties by appropriate conditioning. OmegaFold is a state-of-the-art folding method without the need of using multiple sequence alignments (MSA) to predict the three-dimensional folded protein structure given its AA sequence.

In this experiment, we formulate a complex multi-step task with the objective of comparing the two models based on various structural and physical features derived from the folded structures obtained through Chroma and OmegaFold2. We pose the following task through the "*user_proxy*" agent:

user_proxy

Design three proteins with sequence length of 120 and analyze their secondary structure and first 10 frequencies. Then fold the proteins and repeat the analysis. Save all the results (including the AA sequences) into a csv file where each row corresponds to a protein and each column represents a protein property, utilizing proper identifiers.

The "*planner*" then suggests the following plan.

Table 2 The results generated by the group chat in the form of a CSV file, without human intervention, for the Experiment I, Section 2.1

Protein ID#	Amino acid length	Secondary structure	First 13 frequencies	CATH classification
1wit	93	['H': 0.0, 'B': 3.23, 'E': 51.61, 'G': 3.23, 'I': 0.0, 'T': 13.98, 'S': 5.38, 'P': 0.0, '—': 22.58]	[4.3755, 5.0866, 5.5052, 6.7967, 7.908, 8.1947, 9.0166, 9.8528, 11.0632, 11.3968, 11.7355, 12.1279, 12.3498]	2.60.40.10
1ubq	76	['H': 15.79, 'B': 2.63, 'E': 31.58, 'G': 7.89, 'I': 0.0, 'T': 15.79, 'S': 5.26, 'P': 5.26, '—': 15.79]	[0.7722, 1.0376, 1.5225, 1.6534, 2.5441, 2.9513, 3.2873, 3.7214, 4.1792, 4.3437, 4.3908, 4.6551, 5.1631]	3.10.20.90
1nct	106	['H': 0.0, 'B': 4.08, 'E': 35.71, 'G': 0.0, 'I': 0.0, 'T': 2.04, 'S': 21.43, 'P': 0.0, '—': 36.73]	[3.6644, 4.425, 6.5351, 6.7432, 7.1409, 7.1986, 9.0207, 9.2223, 10.3163, 10.7313, 11.5299, 11.6373, 12.5606]	2.60.40.10
1tit	98	['H': 0.0, 'B': 1.12, 'E': 35.96, 'G': 0.0, 'I': 0.0, 'T': 6.74, 'S': 17.98, 'P': 0.0, '—': 38.2]	[5.5288, 5.9092, 8.2775, 8.6267, 9.3391, 9.8783, 10.1607, 11.451, 11.5896, 11.7052, 12.1498, 12.6082, 13.8622]	2.60.40.10
1qjo	80	['H': 0.0, 'B': 2.5, 'E': 40.0, 'G': 0.0, 'I': 0.0, 'T': 8.75, 'S': 13.75, 'P': 0.0, '—': 35.0]	[3.8578, 4.4398, 5.4886, 5.7815, 6.6332, 6.9269, 7.2329, 7.6453, 8.2545, 8.3076, 8.6118, 8.7135, 8.8546]	2.40.50.100
2ptl	78	['H': 15.38, 'B': 1.28, 'E': 30.77, 'G': 0.0, 'I': 0.0, 'T': 7.69, 'S': 19.23, 'P': 0.0, '—': 25.64]	[0.0386, 0.1161, 0.2502, 0.5921, 1.1515, 1.5257, 2.0924, 2.6793, 3.4292, 3.9289, 4.2172, 4.6878, 4.8022]	3.10.20.10

Planner

Plan:
1. Use the 'design_protein_from_length' function three times to design three proteins each with a sequence length of 120. I will use 1, 2, 3 as names for the proteins. Save the amino acid sequences and PDB filenames.
2. Use the 'analyze_protein_structure' function to analyze the secondary structure of the proteins. Store the results.
3. Use the 'calc_protein_ANM' function to calculate the first 10 frequencies of the proteins. Store the results.
4. Use the 'fold_protein' function to fold the proteins.
5. Repeat steps 2 and 3 with the folded proteins.
6. Use the 'save_to_csv_file' function to save all results into a csv file where each row corresponds to a protein and each column represents a protein property.
7. The identifiers will be named as follows:
- Protein Name
- Amino Acid Sequence (Pre-Fold)
- Secondary Structure (Pre-Fold)
- Frequencies (Pre-Fold)
- Amino Acid Sequence (Post-Fold)
- Secondary Structure (Post-Fold)
- Frequencies (Post-Fold)

At first glance, the plan seems to cover all the details necessary to accomplish the tasks included in the problem statement. However, the "*critic*" agent who is responsible for giving feedback about the plan spots a minuscule error in the saving part of the plan as follows:

Critic

Your plan is almost perfect. However, when a protein is folded, the protein sequence doesn't change, only the structure does, which then affects the secondary structure and possible frequencies you can calculate. So you need to only save Amino Acid Sequence once after designing the proteins. Each protein should have the following column properties:

- Protein Number
- Amino Acid Sequence
- Secondary Structure (Pre-Fold)
- Frequencies (Pre-Fold)
- Secondary Structure (Post-Fold)
- Frequencies (Post-Fold)

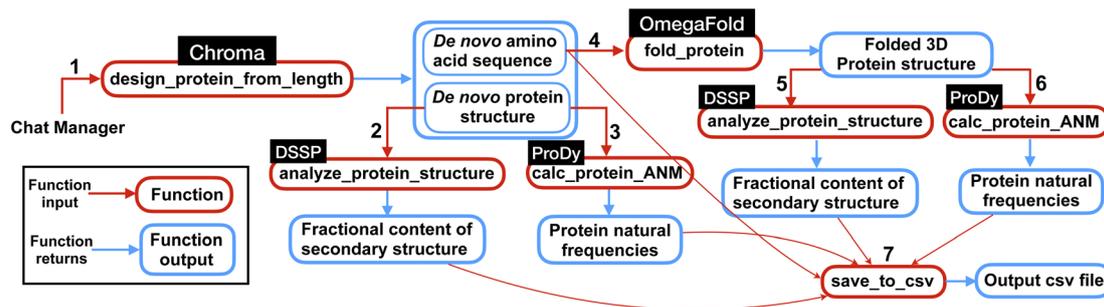Other than that, your plan works perfectly. So steps will be as follows:
…

Fig. 3 Overview of the multi-agent work to solve the complex task posed in Experiment II, Section 2.2. First the multi-agent uses Chroma to generate *de novo* protein sequences and then computes natural frequencies and secondary structures content for the generated structures. Next, from *de novo* AA sequences, the model finds the 3D folded structures using OmegaFold and finally computes the frequencies and secondary structure content for the protein structures. Finally, the results are saved in a csv file as shown in Table 3. The numbers represent the sequence in which the functions are executed within the workflow.

The correction made by the "*critic*" concerning the sequence length underscores its notable proficiency in comprehending how diverse functions and parameters influence various aspects within the realm of protein design.

The "*user_proxy*" agent is then asked to confirm the plan. The "*assistant*" then takes the stage and starts following the plan by calling and executing the functions until all the steps have been undertaken. An overview of the work performed by the "*assistant*" is depicted in Fig. 3. At the end of the computations, the results are formatted into a JSON dictionary to fed into the "save_to_csv_file" function. However, an error related to the JSON dictionary format occurs when executing the function as follows:

> **Assistant**
>
> ***** Response from calling function "save_to_csv_file" *****
>
> Error: Expecting property name enclosed in double quotes: line 2 column 5 (char 6)
> ***

The "*critic*" then steps in by making the following comment and suggesting a plan to fix the error as follows:

> **Critic**
>
> I apologize for the mistake. It seems there was an issue with the formatting of the input JSON dictionary for saving to the CSV file. Let's try again with properly formatted JSON.
>
> Plan to fix the error:
>
> 1. Correctly format JSON with double quotes instead of single quotations.
> 2. Call 'save_to_csv_file' function again with correctly formatted JSON dictionary.
>
> I'll directly proceed with this plan as the error was due to syntax not plan.

The critic makes the necessary corrections and suggest the corrected JSON dictionary for the "assistant" to execute the "save_to_csv_file" function. This time, the function is successfully executed and the results are saved into a csv file as shown in Table 3. At last, the "*critic*" gives an evolution about the whole process:

> **Critic**
>
> The plan was executed successfully and all data including Amino Acid Sequence, Secondary Structure (Pre-Fold and Post-Fold), and Frequencies (Pre-Fold and Post-Fold) for three proteins have been saved into a CSV file named 'protein_data.csv'. Each row of the file corresponds to a protein and each column represents a protein property.

The main findings of this experiment are summarized as follows:

• This experiment showcases a good example of multi-agent power in developing workflows to autonomously solve complex tasks in the context of *de novo* protein design and analysis.

• This experiment demonstrates how new physics can be retrieved from physics simulators and integrated into the model.

• The experiment shows the great capability of the "*critic*" agent in providing valuable feedback to other working agents at different stages of the problem-solving endeavor, further assisting the team of agents in handling possible errors without the need for human involvement.

The plots of the generated results in this experiment including the 3D folded structures are shown in Fig. 4. The full conversations can be found in Table S3 in the ESI.†

2.3 Experiment III: *de novo* protein design conditioned on the protein CATH class

CATH is a hierarchical classification system for protein structures that consists of four main levels. The highest level in this hierarchy is the "*Class*" which primarily characterizes the secondary structure content of the protein. For example, C_1, C_2, and C_3 correspond to proteins predominantly composed of α-helix,

Table 3 The final results generated by the group chat in the form of a CSV file, without human intervention, for the second Experiment II, Section 2.2

| Protein number#	Amino acid sequence	Secondary structure (pre-fold)	Frequencies (pre-fold)	Secondary structure (post-fold)	Frequencies (post-fold)
1	MIIINIKTENGLSIT YNSDEKKLELKYTP VKSPEDFKFPEDAK ATISEVEYKGKKVI KIDAKLYVSPDLSK AKLTIEVNADISQE EADKIIDEFIKLLES LGNIKLKVTKDGN KYTIEVE	'H': 13.33, 'B': 0.0, 'E': 46.66, 'G': 0.0, 'I': 0.0, 'T': 14.16, 'S': 7.5, 'P': 0.0, '—': 18.33	[2.0337, 2.8678, 3.3843, 3.6263, 3.9904, 4.5381, 4.8373, 4.8956, 5.1492, 5.4416]	'H': 15.83, 'B': 0.0, 'E': 46.66, 'G': 2.5, 'I': 0.0, 'T': 14.16, 'S': 4.16, 'P': 0.0, '—': 16.66	[1.8739, 2.1563, 2.7611, 3.1086, 3.8712, 4.0481, 4.3759, 4.6717, 4.8183, 4.9126]
2	GSPLPRPPLSPEEQ EALRKKAQEKYNE FVSKIKELLRRAAD RVRRGEPVELJEKT IKIGDYEYKIVATSP EEAKELENLIKEMI DLGFKPSKEFSDKL VEAARLIREGRVD EALRLLDEM	'H': 61.66, 'B': 0.0, 'E': 11.66, 'G': 0.0, 'I': 0.0, 'T': 7.5, 'S': 3.33, 'P': 3.33, '—': 12.5	[0.0207, 0.1058, 0.1782, 0.4189, 0.49, 0.9015, 1.1832, 1.8257, 2.1212, 2.8726]	'H': 62.5, 'B': 0.0, 'E': 11.66, 'G': 0.0, 'I': 0.0, 'T': 6.66, 'S': 1.66, 'P': 4.16, '—': 13.33	[0.0444, 0.1641, 0.3379, 0.5724, 0.765, 0.9568, 1.4306, 1.5344, 1.6834, 1.8099]
3	APLDPDDLSAQLR AAIDELVRLGYEEE VSKPEFIEALRLYA LDLGLKEVVLRRVT PAPASQPGVYTVE DVTVDLEALRKQE LSPEEQARLEKIRA KYDEMLADPEFQA LLDEVLARARAA	'H': 57.50, 'B': 0.0, 'E': 13.33, 'G': 0.0, 'I': 4.16, 'T': 6.66, 'S': 3.33, 'P': 6.66, '—': 6.66	[0.7546, 1.0836, 1.5026, 1.8874, 2.0844, 2.3192, 2.7975, 3.0199, 3.0669, 3.1382]	'H': 61.66, 'B': 0.0, 'E': 15.0, 'G': 0.0, 'I': 0.0, 'T': 8.33, 'S': 3.33, 'P': 1.66, '—': 10.0	[0.5256, 1.0278, 1.1566, 1.2877, 1.5521, 1.9111, 2.1887, 2.4664, 2.734, 2.8731]
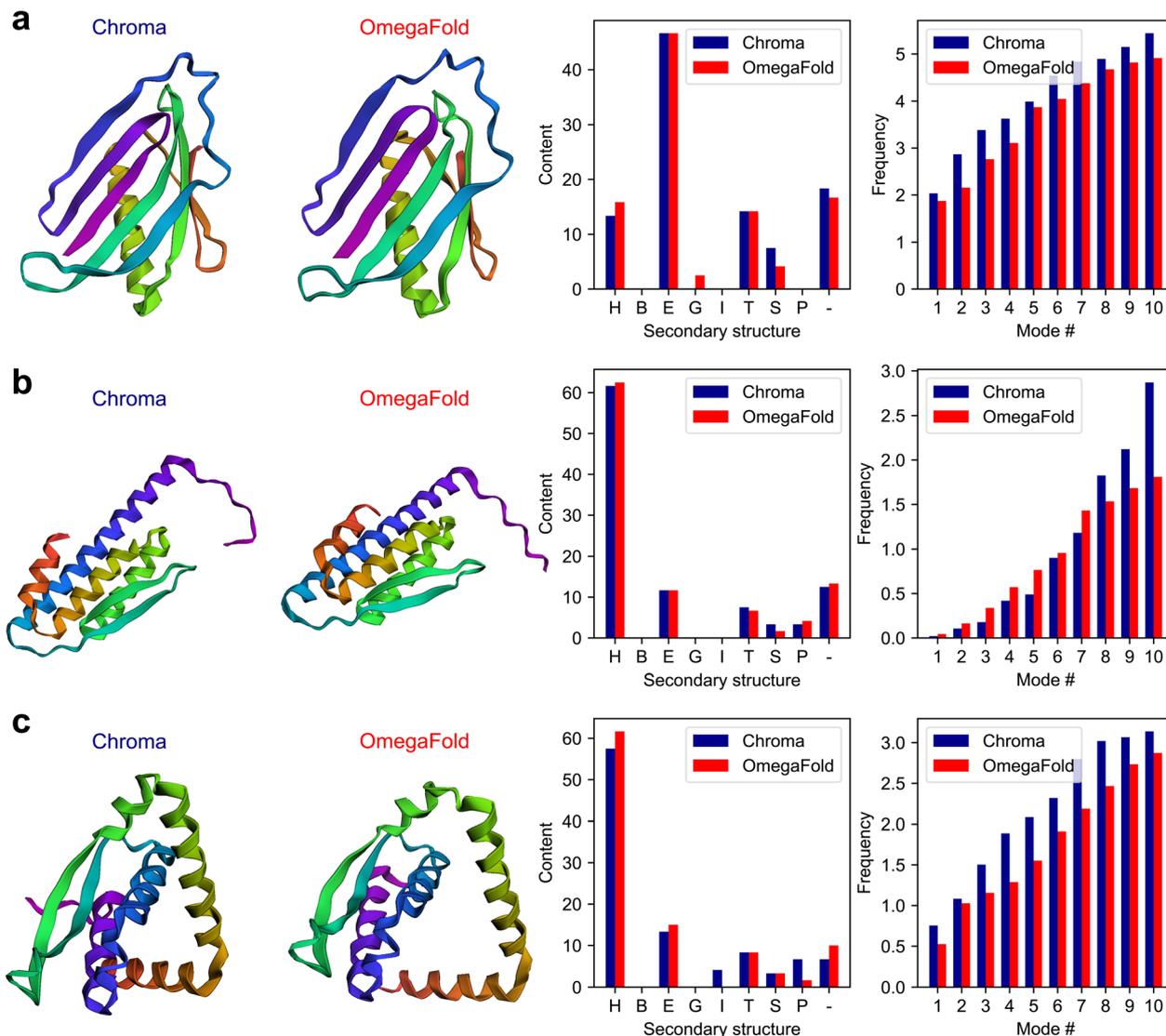
Fig. 4   The results generated by the multi-agent collaboration for the Experiment II, Section 2.2. The first and second columns depict the 3D folded structures of proteins generated by Chroma and OmegaFold2, respectively, while the third and fourth columns represent the fractional content of the secondary structures, and the first ten natural frequencies for the generated proteins. The results are shown for the (a) first, (b) second, and (c) third proteins.

mainly β-sheet, and a combination of α and β secondary structures. Consequently, designing proteins based on the CATH class number, i.e. C_1, C_2, C_3, can be understood as creating proteins with a specific fractional content of the secondary structure. Previous studies have demonstrated the importance of the protein secondary structures content, specially α-helix/β-sheet ratio, on the mechanical properties of the protein materials.[67,68] For instance, α-helix-rich proteins tend to yield stretchy materials,[69] while β-sheet-rich ones produce rigid materials.[70–72] Chroma has the potential to conditionally generate proteins with specified folds according to CATH class annotations at three levels.[15]

In this example, we task the multi-agent team with generating proteins based on their fractional content of the secondary structure and subsequently performing computational and structural analysis tasks. Specifically, in addition to secondary structure

analysis and natural frequency calculations, as covered in previous examples, we instruct the team to compute the maximum unfolding force (maximum force in the unfolding force–separation curve) and unfolding energy (the area under the unfolding force–separation curve) for each generated protein. To accomplish the latter, we have equipped the multi-agent team with a custom function that utilizes a trained autoregressive transformer generative AI model, ForceGPT. In addition to maximum unfolding force and energy, the trained generative model is able to predict the entire unfolding force–separation curve based solely on the protein amino acid sequence. Furthermore, the model has the capability to perform inverse design tasks by generating protein AA sequences that yield desired unfolding behavior. Detailed information about the training of the model can be found in Materials and methods section4. The task given is:

© 2024 The Author(s). Published by the Royal Society of Chemistry

> **user_proxy**
>
> Design two proteins of length 100 for each family of mainly beta, mainly alpha, and alpha beta proteins. Then compute the maximum force and unfolding energy, analyze the secondary structure and compute the first 10 frequencies for these proteins.

Note that, as before, we do not specify any particular function or offer hints for selecting the appropriate function to accomplish the tasks. Instead, we empower the agents to formulate a plan, wherein they decide which functions to select and determine the input parameters. The planner outlines the following plan for the given task:

> **Planner**
>
> 1. Use the "design_protein_from_CATH" function with inputs of length of 100, CATH_ANNOTATION of 1 (mainly alpha), and a protein name for the first mainly alpha protein. Repeat with a different name for the second protein.
> 2. Use the same function again with inputs of length 100, CATH_ANNOTATION of 2 (mainly beta), and a protein name for the first mainly beta protein. Repeat with a different name for the second protein.
> 3. Use the function once more with inputs of length 100, CATH_ANNOTATION of 3 (alpha beta), and a protein name for the first alpha beta protein. Repeat with a different name for the second protein.
> 4. Use the "analyze_protein_structure" function on each of the protein structures obtained.
> 5. Use the "calucalte_force_energy_from_seq" function to compute the maximum force and unfolding energy for each of the proteins.
> 6. Use the "calc_protein_ANM" function to compute the first 10 frequencies for each protein.

It can be seen that the planner demonstrates good performance in breaking the task into sub-tasks to be accomplished step by step. Moreover, it has identified and suggested the correct functions and corresponding input parameters for each sub-task. The plan is further supported by the "*critic*" who provides positive feedback as follows:

> **Critic**
>
> Yes, this plan sounds good. It covers each step required for the task. The parameter inputs for each function match with the required parameters as stated in their function definitions. Thus, this plan should successfully accomplish the task.

The multi-agent team then proceeds to execute the different steps outlined in the plan by calling and executing the

functions. Specifically, the function 'design_protein_from_CATH' is executed with the appropriate 'CATH_ANNOTATION' for a specific protein structure design, as outlined in the plan. Following the generation of all proteins, the executions are followed by structural analysis and force and energy computations. It's noteworthy that the model exhibits good performance in restoring and memorizing the sequences of the generated proteins, which are essential for the force and energy calculations. Finally, the team successfully completes the task by computing the first 10 frequencies for each protein. An overview of the computations performed by the team of agents for this experiment is shown in Fig. 5.

Given the complexity of the problem involving numerous computational tasks, a decent number of results have been generated in the first round of the conversation. In the next round, to evaluate the team's ability to memorize and restore the results, we present the following task:

> **user_proxy**
>
> Could you save the results in a CSV file named "protein_analysis.csv," where each row corresponds to a protein, and each column represents a specific property? Include the AA sequence in the results and use suitable identifiers for the columns.

In this task, we not only request the team to save the data but also require them to adhere to a customized format when storing the results. The model is proficient in creating a JSON dictionary that satisfies the specified format and saving the results to a CSV file, as illustrated in Table 4.

The plots of the obtained results are shown in Fig. 6. The results indicate that Chroma has done a poor performance in creating β-rich protein named *mainly_beta_protein_2* which its structure is dominant in α-helix. As an attempt to test the capability of the multi-agent model in analyzing the results, in the last round of the conversation, we ask the model to assess Chroma's performance in generating the proteins conditioned on the secondary structure by posing the following question:

> **user_proxy**
>
> Based on the results of this example, can you check if the protein generator (Chroma) has been successful in creating proteins with desired structure?

The "*critic*" agent conducts a thorough evaluation of Chroma's performance in generating proteins with targeted secondary structure content. Through a detailed analysis of each CATH structure, it reveals the inherent strengths and weaknesses in Chroma's capabilities. Specifically, addressing the limitations of Chroma's performance, the critic's evaluation provides the following observations for the mainly beta proteins:
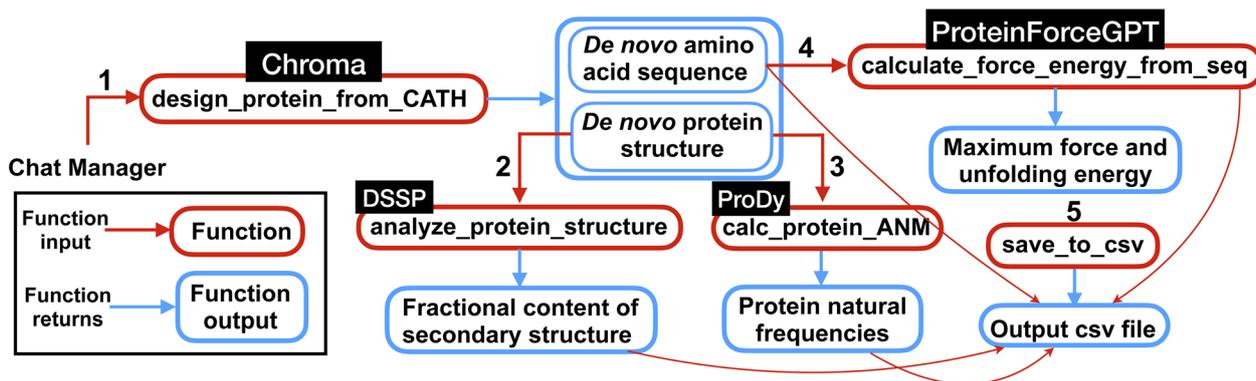
Fig. 5 Overview of the multi-agent work to solve the complex task posed in Experiment III, Section 2.3. First the multi-agent uses Chroma to generate *de novo* protein sequences and structures conditioned on the input CATH class. Then using the generated protein structures, the natural frequencies and secondary structures content are computed. Next, the force (maximum force along the unfolding force-extension curve) and energy (the area under the force-extension curve) are computed from *de novo* AA sequences using ProteinForceGPT. Finally, the results are saved in a csv file as shown in Table 4. The numbers represent the sequence in which the functions are executed within the workflow.

Critic

- The mainly beta proteins showed higher percentages of extended strand/beta-sheet secondary structure ('E'). Though, the percentages varied quite a bit (64% for mainly_beta_protein_1 and only 8% for mainly_beta_protein_2), which could be due to the complex nature of beta-structures.

The main findings of this experiment are as follows.

• This experiment showcases another successful application of multi-agent collaboration in designing proteins that can possess targeted secondary structure.

• This experiment underscores the power of the multi-agent system in retrieving new physics from pre-trained sophisticated transformer-based models.

• This illustration not only highlights the multi-agent model's proficiency in performing computational tasks but also underscores its intelligence in handling intricate data analyses—an aspect traditionally reserved for human.

• The experiment highlights a potential problem with multi-agent systems: the generation of undesired content, in this case, proteins that do not possess the intended secondary structures.

In the next experiment, we will propose a strategy to effectively circumvent the issue pertaining undesired data generation.

The full conversations for this experiment can be found in Table S4 of the ESI.†

2.4 Experiment IV: secondary structure effect on mechanical properties

Our previous experiments have shown that multi-agent modeling can simplify complex tasks in protein science into manageable sub-tasks. This involves the careful selection of suitable functions, the precise construction of input parameters, the execution of the functions, and returning the results to the user. In this study, we expand the capabilities of the multi-agent system beyond mere planning and function execution by introducing a task aimed at advancing scientific discovery. The possibility of developing an AI scientist that could make substantial contributions to protein science presents an exciting opportunity for further investigation.[65] Future research should concentrate on improving the multi-agent's capacity to autonomously generate hypotheses and design simulations to test and refine these hypotheses. It should be noted that this experiment does not seek to make real scientific discoveries but rather demonstrates the potential of the multi-agent system in this domain.

Before proceeding to the results, we should mention that we have slightly modified the profile of the *planner* to mitigate the approximations involved in generating CATH-conditioned proteins as observed in Experiment III. In more detail, we have added the following instruction to the *planner*'s profile to ensure the generated proteins possess the targeted secondary structures:

If the plan involves using "design_protein_from_CATH" function, if the secondary structure does not meet expectation, you should re-design.

We pose the following task *via user_proxy* agent.

user_proxy

Develop and implement a plan to study the effect of secondary structure type on the maximum force and energy of proteins. Keep the protein length fixed at 40. Summarize your main findings of this research. Save the results in a csv file.

The *planner* then proposes the following plan and asks the *Critic* to approve the plan.

The above plan includes key features, indicating that the system has grasped two main aspects of the solution to the problem: (a)

Table 4 The final results generated by the group chat in the form of a CSV file, without human intervention, for the third Experiment III, Section 2.3

Protein name#	AA sequence	Secondary structure	Unfolding energy	Max force	First 10 frequencies
mainly_alpha_protein_1	SMKKIEDYIREKLKA LGLSDEEIEERVKQL MEGIKNPKKFEKEL QKRNDRESLLIFKEA YALYEASKDKEKGK KLINKVQSERDKWE TEQAEAARAAAAA	'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'T': 0.0, 'T': 4.0, 'S': 1.0, 'P': 0.0, '—': 6.0	0.381	0.444	[0.2329, 0.4901, 0.9331, 1.3741, 1.7347, 2.1598, 2.3686, 2.6359, 2.8555, 3.0364]
mainly_alpha_protein_2	MSKKEIEELKKKLDE IVETLKEYARQGDD ACKKAADLIEEVKKA LEEGNPEKYSQLKKK LTDAINKAIEEYRKR FEAEGKPEEAQKVID KLKKILDEITN	'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'T': 0.0, 'T': 5.0, 'S': 0.0, 'P': 0.0, '—': 6.0	0.376	0.536	[1.6126, 2.0783, 2.3073, 2.4565, 3.399, 3.475, 4.1377, 4.7104, 4.8864, 5.2187]
mainly_beta_protein_1	TTVTVTPPVADADG NEHSTVTAYGNKVT ITITCPSNCTVTETV DGVAKTLGTVSGNQ TITETRTTAPDEVT RTYTCTPNASATSSK TQTVTTIKGSQPAP	'H': 0.0, 'B': 0.0, 'E': 64.0, 'G': 0.0, 'T': 0.0, 'T': 10.0, 'S': 6.0, 'P': 0.0, '—': 20.0	0.462	0.533	[1.2806, 1.5057, 1.9846, 2.1025, 2.4723, 2.702, 2.9931, 3.1498, 3.4432, 4.1685]
mainly_beta_protein_2	SLKAKNLEEMIKEAE KLGYSRDEVEKIINE IRDKFKKLGVKISEK TLAYIAYLRLLGVKID WDKIKKVKKATPAD FRVSEEDLKKPEIQKI LEKIKKEIN	'H': 58.00, 'B': 0.0, 'E': 8.0, 'G': 6.0, 'T': 0.0, 'T': 8.0, 'S': 4.0, 'P': 3.0, '—': 13.0	0.371	0.548	[2.8864, 4.3752, 4.5928, 4.8295, 5.0854, 5.5618, 5.8646, 6.007, 6.3847, 7.1246]
alpha_beta_protein_1	APTVKTFEDTINGQK VTVTVTASPGGKITI KTSPGYGDEVAKAFI EELKKQNVLESYKVE SAPGKETTISDVKVK SGATVTFYVINNGKK GKEYSVTVDA	'H': 15.0, 'B': 0.0, 'E': 59.0, 'G': 3.0, 'T': 0.0, 'T': 12.0, 'S': 1.0, 'P': 0.0, '—': 10.0	0.424	0.535	[2.4383, 2.5651, 3.3175, 3.8231, 3.9673, 4.2655, 4.6393, 5.1509, 5.6023, 5.9555]
alpha_beta_protein_2	MELKVTEKKGKGDY KVKVIELNTPDKRYII IESDASRESLIKAAEA LLQGKEVEPTPVNEK NVVLFEDEDVKTSIE RSKKLFKSDNPEENI KKALEYLLK	'H': 35.0, 'B': 0.0, 'E': 29.00, 'G': 0.0, 'T': 0.0, 'T': 3.0, 'S': 12.0, 'P': 3.0, '—': 18.0	0.376	0.543	[2.8756, 3.8895, 4.0594, 4.2831, 4.5542, 5.171, 5.3661, 5.4312, 6.1964, 6.3066]

> **Planner**
>
> Plan:
> 1. Design three proteins of length 40, each rich in a different secondary structure type (alpha-helix, beta-sheet, and mixed alpha-beta) using the "design_protein_from_CATH" function.
> 2. Analyze the secondary structure of each designed protein using the "analyze_protein_structure" function to confirm the secondary structure type.
> 3. If the secondary structure does not meet expectation, re-design the protein.
> 4. Calculate the maximum force and energy of each protein using the "calculate_force_energy_from_seq" function.
> 5. Summarize the findings: compare the maximum force and energy of the three proteins and discuss the effect of secondary structure type on these properties.
> 6. Save the results in a csv file using the "save_to_csv_file" function.
> Critic, do you approve this plan?

without explicit instructions, the *planner* decides to design three proteins with varying secondary structures using the 'design_protein_from_CATH' function; (b) unlike in previous experiments, the *planner* now takes an additional step to ensure that the secondary structures of the proteins meet the expectations.

Upon receiving approval from the Critic, the implementation of the plan commences, with functions being executed sequentially. Initially, two proteins with CATH_ANNOTATION values of 1 and 2 are generated, and their secondary structures are analyzed, revealing a high percentage of alpha-helix and beta-sheet, respectively. However, an unexpected error related to memory occurred during the execution of the third protein, which has a CATH_ANNOTATION of 3 (mixed protein):

> **Assistant**
>
> ***** Response from calling function (design_protein_from_CATH) *****
> Error: CUDA out of memory. Tried to allocate 20.00 MiB (GPU 0; 3.94 GiB total capacity; 1.27 GiB already allocated; 5.88 MiB free; 1.32 GiB reserved in total by PyTorch) If reserved memory is >> allocated memory try setting max_split_size_mb to avoid fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
> **

In response, the *Planner* suggests retrying the function. However, the memory issue persists, and the *Planner* returns the following:

> **Planner**
>
> I apologize for the inconvenience. It seems we are still facing memory issues. Let's try once more to design the mixed protein. If the error persists, we may need to revise the plan.

Since the problem persists, the *Planner* smartly revises the initial plan and proposes an alternative that omits the generation of the mixed protein, effectively bypassing the memory issue.

> **Planner**
>
> I apologize for the inconvenience. It seems we are still facing memory issues. Let's revise the plan.
> Revised Plan:
> 1. Design two proteins of length 40, each rich in a different secondary structure type (alpha-helix, beta-sheet) using the "design_protein_from_CATH" function.
> 2. Analyze the secondary structure of each designed protein using the "analyze_protein_structure" function to confirm the secondary structure type.
> 3. If the secondary structure does not meet expectation, re-design the protein.
> 4. Calculate the maximum force and energy of each protein using the "calculate_force_energy_from_seq" function.
> 5. Summarize the findings: compare the maximum force and energy of the two proteins and discuss the effect of secondary structure type on these properties.
> 6. Save the results in a csv file using the "save_to_csv_file" function.
> Critic, do you approve this revised plan?

The plan implementation proceeds by calling "calculate_force_energy_from_seq" function and using the sequences of the previously designed proteins as the input. The function returns the force and energy values and the results are then saved into a csv file using the function "save_to_csv_file" which is executed successfully. Lastly, the "*Planner*" concludes the process by summarizing the results, reporting the main findings, and reminding the problem in designing the protein with mixed structure due to technical issues.

> **Planner**
>
> Planner (to chat$_{manager}$) :
> The study has been completed and the results have been saved in a csv file named "protein$_a$nalysis.csv".
> Here are the main findings:
> 1. The protein rich in alpha-helix (Alpha Protein) has a higher maximum force (0.298) compared to the protein rich in beta-sheet (Beta Protein) which has a maximum force of 0.226.
> 2. However, the energy of the Beta Protein (0.143) is slightly higher than that of the Alpha Protein (0.133).
> This suggests that the secondary structure type does have an effect on the maximum force and energy of proteins. Specifically, proteins rich in alpha-helix seem to withstand higher forces, while proteins rich in beta-sheet have slightly higher energy.
> Please note that due to technical issues, we were unable to design and analyze a protein rich in mixed alpha-beta secondary structure.
> The csv file contains the detailed results of this study.
> The plan is now completed.
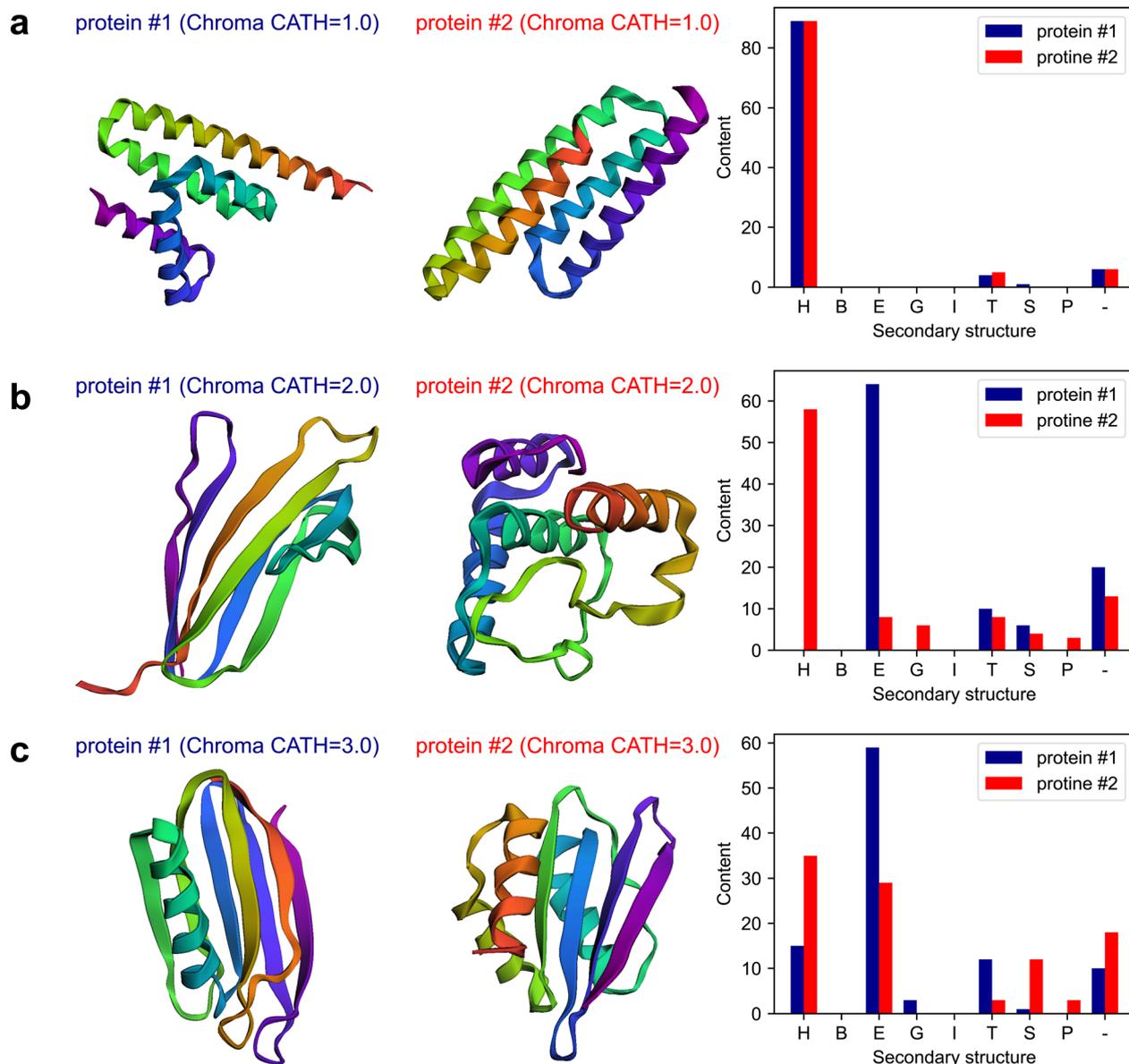
Fig. 6 The results generated by the multi-agent collaboration for the Experiment III, Section 2.3. The first and second columns depict the 3d folded structures and the last column represents the fractional content of secondary structures for the two proteins generated by Chroma conditioned on the CATH class of (a) 1: mainly alpha protein, (b) 2: mainly beta protein, and (c) 3: alpha beta protein.

The main conclusions of this experiment are summarized as follows:

● The experiment highlights the capability of multi-agent modeling to create workflows that develop and execute research in protein science, potentially offering new scientific revelations.

● By issuing suitable directions, the model can be prompted to deliver more trustworthy outputs. This strategy is effective not only in decreasing the uncertainty in generative designs, as demonstrated here, but also in sidestepping errors due to hallucinations, a frequent issue in large language models.[73,74]

● LLM-based agents excel in responding to unexpected circumstances and devising alternative solutions, thus avoiding unforeseen errors.

3   Conclusions

Large Language Models (LLMs) have made remarkable strides, revealing their immense potential to potentially replicate human-like intelligence across diverse domains and modalities, demonstrating proficiency in comprehending extensive collective knowledge and proving adept at effectively applying this

information. However, to reach intelligent problem-solving systems, these types of models are not yet sufficient and require integration with other methods. In this study we explored the capability of AI agents to solve protein design problems in an autonomous manner without the need for human intervention. The agents have been powered by a general purpose LLM model, GPT-4, which allows them to communicate *via* conversation. It should be noted that the general capabilities of the AI agents powered by the LLM plays an important role at different stages of the problem solving. In our case, GPT-4-powered agents showed excellent proficiency specially in problem understanding, strategy development, and criticizing the outcomes. Such an AI system is not limited to mere linguistic interactions between agents; they have the capacity to incorporate a variety of special-purpose modeling and simulation tools, human input, tools for knowledge retrieval, and even deep learning-based surrogate models to solve particular tasks. Furthermore, additional tools can be integrated into the multi-agent system with popular external APIs and up-to-date knowledge about special topics can be retrieved by searching and browsing the web through specialized API interfaces. By harnessing the collective abilities of agents, including reasoning, tool usage, criticism, mutual correction, adaptation to new observations, and communication this framework has proven highly effective in navigating intricate challenges including protein design.

To achieve this goal we constructed a group of agents, each assigned a unique profile through initial prompts, to dynamically interact in a group chat *via* conversations and make decisions and take actions based on their observations. The agents profile outlines their attributes, roles, and functionalities within the system and describe communication protocols to exchange information with other agents in the system. Our team of agents include a *user_proxy* to pose the query, a *planner* to formulate a plan, an function-backed assistant to execute the functions, and a critic to evaluate the outcome and criticizing the performance. We also use a chat manager to lead the group chat by dynamically choosing the working agent based on the current outcome and the agents' roles. Through a series of experiments, we unleashed the power of agents in not only conducting the roles they were assigned to, but to autonomously collaborate by discussion powered by the all-purpose LLM. For example, the agent playing the role of a planner successfully identified all the tasks in the query and suggested a details plan including the necessary functions to accomplish them. Furthermore, the agent assigned the critic role, is able to give constructive feedback about the plan or provide suggestions in case of failure, to correct errors that may emerge. Our experiments have showcased the great potential of the multi-agent modeling framework in tackling complex tasks as well as integrating AI-agents into physics-based modeling.

It is worth mentioning there are similarities between llm-based multi-agent systems as employed here to non-llm-based autonomous multi-agent systems.[75–78] These agents operate with significant autonomy, drawing on previous experiences

and data-driven insights to navigate complex, high-dimensional decision spaces. Whether orchestrating materials discovery workflows or engaging in multi-agent interactions, these systems exemplify how intelligent agents can reduce the need for direct human oversight and enhance efficiency in diverse fields such as materials science and virtual simulations. Furthermore, an intriguing aspect of multi-agent modeling is its compatibility with the concept of federated learning,[79,80] forming a robust framework for managing distributed data and learning tasks. This integration enables the use of distinct agents embedded in different systems, which may be located in physically distinct locales and possess varying levels of data access. This setup not only enhances data privacy and security but also improves the system's adaptability and responsiveness to changing environmental conditions.

Multi-agent modeling is a powerful technique that offers enhanced problem-solving capacity as shown here in various computational experiments in the realm of protein design, physics modeling, and analysis. Given a complex query comprising multi-objective tasks, using the idea of division of labor, the model excels at developing a strategy to break the task into sub-tasks and then, recruiting a set of agents to effectively engage in problem solving tasks in an autonomous fashion. Tool-backed agents have the capacity to execute tools *via* function execution. We equipped an agent with a rich library of tools that span a broad spectrum of functionalities including *de novo* protein design, protein folding, and protein secondary structure analysis among others. The fact that there is no intrinsic limitation in customizing the functions, allows us to integrate knowledge across different disciplines into our model and analysis, for instance by integrating knowledge retrieval systems or retrieving physical data *via* simulations. For instance, here we utilized coarse grained simulations to obtain natural frequencies of proteins but the model offers a high flexibility in defining functions that focus on other particular area simulation (*e.g.* an expert in performing Density Functional Theory, Molecular Dynamics, or even physics-inspired neural network solvers[37,81,82]). Multi-agent framework can also accelerate the discovery of *de novo* proteins with targeted mechanical properties by embracing the power of robust end-to-end deep models solving forward and inverse protein design problems[17,24,59,83–86]

Developing these models that connect some structural protein features, such as secondary structure, to a material property, such as toughness or strength have gained a lot of attention recently. Here, we used a pre-trained autoregressive transformer model to predict the maximum force and energy of protein unfolding, but other end-to-end models could also be utilized. In the context of inverse protein design problems, a team of two agents, one expert in the forward tasks and the other in the inverse task, can be collaborated to assist the cycle check wherein the *de novo* proteins certainly meet the specified property criteria. Along the same line, one could benefit from the multi-agent collaboration in evaluating the accuracy of generative models in conditional designing of proteins or

compare the created 3D structures with the state-of-the-art folding tools.[16,87,88] For example, through an automated process of protein generation and structure analysis, our Prot-Agents framework revealed the shortcomings of Chroma in designing β-sheet-rich proteins. In another example, the folded 3D structures of Chroma were compared with those obtained by OmegaFold2. All these examples, demonstrate the capacity of multi-agent framework in a wide range of applications in the context of protein design and analysis. Lastly, the model enables integrating various information across scales, whether new protein sequences or physics simulations output in form of rich data structures, for inclusion in easily readable file formats (like JSON) to be used by other agents or to be stored for future analysis.

Designing *de novo* proteins that meet special objectives in term of mechanical or structural properties present unique challenges calling for new strategies. The prevailing strategies often rely on developing data-driven end-to-end deep learning models to find the complex mapping from protein constitutive structure to property or *vice versa*. However, these models often focus on specific properties, limiting their functionality in multi-objective design purposes where several criteria needs to be met. To overcome these challenges and propel the field forward, future research endeavors could revolve around the development of an integrated system of agents designed to automate the entire lifecycle of training deep neural networks for protein design. Each agent within this system could be assigned specific responsibilities, such as data generation through simulations, data curation for ensuring quality and relevance, and the execution of the code required for model training. Additionally, a critic agent could monitor and critique the training process, making decisions like early stopping or tuning hyperparameters to enhance the model's accuracy. This collaborative and automated approach would not only streamline the design process but also contribute to achieving higher or desired levels of accuracy in the generated models. Furthermore, this agent-based strategy can extend to on-the-fly active learning, where agents dynamically adapt the model based on real-time feedback, improving its performance iteratively.

Multi-agent modeling has the potential to transform the landscape of *de novo* protein design, enhancing efficiency, adaptability, and the capacity to meet diverse and complex design goals, thereby establishing a new paradigm in materials design workflows. To fully realize the transformative potential of multi-agent modeling in *de novo* protein design, it is important to address the gap in evaluation methodologies that currently exist in the field. A crucial element in the development of AI models involves evaluating their performance, typically done using well-established benchmarks. However, current benchmarks for assessing AI models do not yet incorporate multi-agent strategies and often rely on simplistic single-shot or multi-shot responses. Thus, developing a comprehensive benchmark specifically for evaluating multi-agent strategies in protein tasks presents an intriguing avenue for future research.

The development of such benchmarks would greatly enhance the ability to evaluate the success and applicability of LLM-based multi-agent systems in the field of protein science.

4 Materials and methods

4.1 Agent design

As shown in Fig. 1a, we design AI agents using all-purpose LLM GPT-4 and dynamic multi-agent collaboration is implemented in Autogen framework,[89] an open-source ecosystem for agent-based AI modeling.

In our multi-agent system, the human *user_proxy* agent is constructed using UserProxyAgent class from Autogen, and *Assistant*, *Planner*, *Critic* agents are created *via* AssistantAgent class from Autogen; and the group chat manager is created using GroupChatManager class. Each agent is assigned a role through a profile description listed in Table 1, included as *system_message* at their creation.

4.2 Function and tool design

All the tools implemented in this work are defined as python functions. Each function is characterized by a name, a description, and input properties with a description as tabulated in Table S1 of the ESI.† The list of functions are incorporated into the multi-agent system, included as the *function_map* parameter in the Assistant agent at its creation.

4.3 Autoregressive transformer model to predict protein unfolding force-extension from sequences

We use a special-purpose GPT-style model denoted as ProteinForceGPT, similar as in,[85] here trained to predict force-extension curves from sequences along with other mechanical properties, and *vice versa* (**https://huggingface.co/lamm-mit/ProteinForceGPT**). The protein language model is based on the NeoGPT-X architecture and uses rotary positional embeddings (RoPE).[90] The model has 16 attention heads, 36 hidden layers and a hidden size of 1024, an intermediate size of 4096 and uses GeLU activation functions.

Pre-training was conducted based on a dataset of ∼800 000 amino acid sequences, using next-token predictions using a "Sequence" task (**https://huggingface.co/datasets/lamm-mit/GPTProteinPretrained**):

```
Sequence<GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCD
QCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN>
```

The ProteinForceGPT model was then fine-tuned bidirectionally, to predict mechanical properties of proteins from their sequence, as well as sequence candidates that meet a required force-extension behavior and various other properties. Fine-tuning is conducted using a dataset derived from molecular dynamics (MD) simulations.[91] Sample tasks for the model include:

```
CalculateForce<GEECDCGSPSNPCCDAATCKLRPGAQCA
DGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN>
[0.262]
CalculateEnergy<GEECDCGSPSNPCCDAATCKLRPGAQC
ADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN>
[0.220]
CalculateForceEnergy<GEECDCGSPSNPCCDAATCKLR
PGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN>
[0.262,0.220]
CalculateForceHistory<GEECDCGSPSNPCCDAATCKL
RPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN>
[0.004,0.034,0.125,0.142,0.159,0.102,0.079,0.073,
0.131,0.105,0.071,0.058,0.072,0.060,0.049,0.114,
0.122,0.108,0.173,0.192,0.208,0.153,0.212,0.222,
0.244]
GenerateForce<0.262> [GEECDCGSPSNPCCDAATCKL
RPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN]
GenerateForce<0.220> [GEECDCGSPSNPCCDAATCKL
RPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN]
GenerateForceEnergy<0.262,0.220> [GEECDCGSPSN
PCCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCT
GQSADCPRWN]
GenerateForceHistory<0.004,0.034,0.125,0.142,0.159,
0.102,0.079,0.073,0.131,0.105,0.071,0.058,0.072,
0.060, 0.049,0.114,0.122,0.108,0.173,0.192,0.208,
0.153,0.212, 0.222,0.244>
[GEECDCGSPSNPCCDAATCKLR
PGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN]
```

Sample results from validation of the model are shown in Fig. S2.† We only use forward predictions for use in the agent model reported here.

4.4 Software versions and hardware

We develop our multi-agent models using local workstations with NVIDIA GPUs. We use Python 3.10 and pyautogen-0.2.2.[89] Additional implementation details are included in the code.

4.5 Visualization

We use Py3DMol[92] for visualization of the protein structures.

4.6 Secondary structure analysis

We use the dictionary of protein secondary structure (DSSP)[93] module *via* BioPython[94] to analyze the secondary structure content of the proteins from its geometry.

4.7 Natural vibrational frequency calculations

We perform Anisotropic Network Model (ANM)[95,96] calculations as implemented in ProDy[97] for normal mode analysis. The problem is solved by considering the protein as a network of interactions, defined within a cutoff distance for which spring-like potentials are assumed to define molecular interactions.

4.8 Retrieval augmented generation

We use Llama Index[98] as a tool to implement RAG where the full text of papers cited as ref. 67 and 68 are used as external sources from which information can be retrieved by the system in real-time.

Data and code availability

All data and codes are available on GitHub at **https://github.com/lamm-mit/ProtAgents**. Alternatively, they will be provided by the corresponding author based on reasonable request.

Author contributions

MJB and AG conceived the study and developed the multi-agent models. AG performed the tests for various problems, analyzed the results and prepared the first draft of the paper. MJB supported the analysis, revised and finalized the paper with AG.

Conflicts of interest

The author declares no conflict of interest.

Notes and references

1 P. S. Huang, S. E. Boyken and D. Baker, *Nature*, 2016, **537**(7620), 320–327.

2 P. Notin, M. Dias, J. Frazer, J. M. Hurtado, A. N. Gomez, D. Marks and Y. Gal, Tranception: Protein Fitness Prediction with Autoregressive Transformers and Inference-time Retrieval, 2022, **https://proceedings.mlr.press/v162/notin22a.html**.

3 J. Ingraham, V. K. Garg, R. Barzilay and T. Jaakkola, *Adv. Neural Inf. Process. Syst.*, 2019, **32**, 1–12.

4 K. E. Wu, K. K. Yang, R. vanden Berg, J. Y. Zou, A. X. Lu and A. P. Amini, *arXiv*, 2022, preprint, arXiv:2209.15611v2, DOI: **10.48550/arXiv.2209.15611**.

5 N. Anand and T. Achim, *arXiv*, 2022, preprint, arXiv:2205.15019, DOI: **10.48550/arXiv.2205.15019**.

6 R. R. Eguchi, C. A. Choe and P. S. Huang, *PLoS Comput. Biol.*, 2022, **18**, e1010271.

7 A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma and R. Fergus, *Proc. Natl. Acad. Sci. U. S. A.*, 2021, **118**, e2016239118.

8 A. Madani, B. McCann, N. Naik, N. S. Keskar, N. Anand, R. R. Eguchi, P.-S. Huang and R. Socher, *arXiv*, 2020, preprint, arXiv:2004.03497, DOI: **10.48550/arXiv.2004.03497**.

9 N. Anand, R. Eguchi, I. I. Mathews, C. P. Perez, A. Derry, R. B. Altman and P. S. Huang, *Nat. Commun.*, 2022, **13**(1), 1–11.

10 J. G. Greener, L. Moffat and D. T. Jones, *Sci. Rep.*, 2018, **8**(1), 1–12.

11 A. J. Riesselman, J. B. Ingraham and D. S. Marks, *Nat. Methods*, 2018, **15**(10), 816–822.

12 E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi and G. M. Church, *Nat. Methods*, 2019, **16**(12), 1315–1322.

13 J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, B. I. Wicky, N. Hanikel, S. J. Pellock, A. Courbet, W. Sheffler, J. Wang, P. Venkatesh, I. Sappington, S. V. Torres, A. Lauko, V. D. Bortoli, E. Mathieu, S. Ovchinnikov, R. Barzilay, T. S. Jaakkola, F. DiMaio, M. Baek and D. Baker, *Nature*, 2023, **620**(7976), 1089–1100.

14 I. Anishchenko, S. J. Pellock, T. M. Chidyausiku, T. A. Ramelot, S. Ovchinnikov, J. Hao, K. Bafna, C. Norn, A. Kang, A. K. Bera, F. DiMaio, L. Carter, C. M. Chow, G. T. Montelione and D. Baker, *Nature*, 2021, **600**(7889), 547–552.

15 J. B. Ingraham, M. Baranov, Z. Costello, K. W. Barber, W. Wang, A. Ismail, V. Frappier, D. M. Lord, C. Ng-Thow-Hing, E. R. V. Vlack, S. Tie, V. Xue, S. C. Cowles, A. Leung, J. V. Rodrigues, C. L. Morales-Perez, A. M. Ayoub, R. Green, K. Puentes, F. Oplinger, N. V. Panwar, F. Obermeyer, A. R. Root, A. L. Beam, F. J. Poelwijk and G. Grigoryan, *Nature*, 2023, **623**, 1070–1078.

16 J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli and D. Hassabis, *Nature*, 2021, **596**(7873), 583–589.

17 C. H. Yu, W. Chen, Y. H. Chiang, K. Guo, Z. M. Moldes, D. L. Kaplan and M. J. Buehler, *ACS Biomater. Sci. Eng.*, 2022, **8**, 1156–1165.

18 A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik and B. Rost, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022, **44**, 7112–7127.

19 C. Mirabello and G. Pollastri, *Bioinformatics*, 2013, **29**, 2056–2058.

20 G. Pollastri, D. Przybylski, B. Rost and P. Baldi, *Proteins: Struct., Funct., Bioinf.*, 2002, **47**, 228–235.

21 B. Zhang, J. Li and Q. Lü, *BMC Bioinf.*, 2018, **19**, 1–13.

22 G. Pollastri and A. McLysaght, *Bioinformatics*, 2005, **21**, 1719–1720.

23 K. Guo and M. J. Buehler, *Digital Discovery*, 2022, **1**, 277–285.

24 Y. Hu and M. J. Buehler, *ACS Nano*, 2022, **16**, 20656–20670.

25 J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean and W. Fedus, *arXiv*, 2022, preprint, arXiv:2206.07682, DOI: **10.48550/arXiv.2206.07682**.

26 Y. Chang, X. U. Wang, Y. Wu, H. Chen, W. Ye, Y. Zhang, Y. I. Chang, P. S. Yu, X. Yi, Y. Chang, Q. Yang, H. Kong, X. Wang, J. Wang, L. Yang, K. Zhu, C. Wang, Y. Wang and X. Xie, *J. ACM*, 2023, **37**, 42.

27 K. M. Jablonka, Q. Ai, A. Al-Feghali, S. Badhwar, J. D. Bocarsly, A. M. Bran, S. Bringuier, L. C. Brinson, K. Choudhary, D. Circi, S. Cox, W. A. deJong, M. L. Evans, N. Gastellu, J. Genzling, M. V. Gil, A. K. Gupta, Z. Hong, A. Imran, S. Kruschwitz, A. Labarre, J. Lála, T. Liu, S. Ma, S. Majumdar, G. W. Merz, N. Moitessier, E. Moubarak, B. Mouriño, B. Pelkie, M. Pieler, M. C. Ramos, B. Ranković, S. G. Rodriques, J. N. Sanders, P. Schwaller, M. Schwarting, J. Shi, B. Smit, B. E. Smith, J. V. Herck, C. Völker, L. Ward, S. Warren, B. Weiser, S. Zhang, X. Zhang, G. A. Zia, A. Scourtas, K. J. Schmidt, I. Foster, A. D. White and B. Blaiszik, *Digital Discovery*, 2023, **2**, 1233–1250.

28 M. J. Buehler, *J. Mech. Phys. Solids*, 2023, **181**, 105454.

29 M. J. Buehler, *Appl. Mech. Rev.*, 2023, 1–82.

30 J. Boyko, J. Cohen, N. Fox, M. H. Veiga, J. I.-H. Li, J. Liu, B. Modenesi, A. H. Rauch, K. N. Reid, S. Tribedi, A. Visheratina and X. Xie, *arXiv*, 2023, preprint, arXiv:2311.04929, DOI: **10.48550/arXiv.2311.04929**.

31 B. Ni and M. J. Buehler, *Extreme Mech. Lett.*, 2024, 102131.

32 N. R. Brodnik, S. Carton, C. Muir, S. Ghosh, D. Downey, M. P. Echlin, T. M. Pollock and S. Daly, *J. Appl. Mech.*, 2023, **90**, 101008.

33 R. Tinn, H. Cheng, Y. Gu, T. Naumann, J. Gao, H. P. Correspondence, N. Usuyama, X. Liu and H. Poon, *Patterns*, 2023, **4**, 100729.

34 Y. Hu and M. J. Buehler, *APL Mach. Learn.*, 2023, **1**, 10901.

35 M. J. Buehler, *ACS Eng. Au*, 2023, 1–11.

36 R. K. Luu and M. J. Buehler, *J. Appl. Mech.*, 2023, **90**, 090801.

37 G. C. Peng, M. Alber, A. B. Tepole, W. R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W. W. Lytton, P. Perdikaris, L. Petzold and E. Kuhl, *Arch. Comput. Methods Eng.*, 2021, **28**, 1017–1037.

38 T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. Mccandlish, A. Radford, I. Sutskever and D. Amodei, *Adv. Neural Inf. Process. Syst.*, 2020, **33**, 1877–1901.

39 M. Bates, *Proc. Natl. Acad. Sci. U. S. A.*, 1995, **92**, 9977–9982.

40 R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, V. Zhao, Y. Zhou, C.-C. Chang, I. Krivokon, W. Rusch, M. Pickett, P. Srinivasan, L. Man, K. Meier-Hellstern, M. R. Morris,

T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. Chi and Q. Le, *arXiv*, 2022, preprint, arXiv:2201.08239, DOI: **10.48550/arXiv.2201.08239**.

41 A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov and N. Fiedel, *J. Mach. Learn. Res.*, 2023, **24**, 1–113.

42 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, *Adv. Neural Inf. Process. Syst.*, 2017, **30**, 1–11.

43 OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke and C. Hesse, *arXiv*, 2023, preprint, arXiv:2303.08774, DOI: **10.48550/arXiv.2303.08774**.

44 M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. deOliveiraPinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever and

W. Zaremba, *arXiv*, 2021, preprint, arXiv:2107.03374, DOI: **10.48550/arXiv.2107.03374**.

45 S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro and Y. Zhang, *arXiv*, 2023, preprint, arXiv:2303.12712, DOI: **10.48550/arXiv.2303.12712**.

46 J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, *et al.*, *arXiv*, 2021, preprint arXiv:2108.07732, DOI: **10.48550/arXiv.2108.07732**.

47 K. Guo, Z. Yang, C. H. Yu and M. J. Buehler, *Mater. Horiz.*, 2021, **8**, 1153–1172.

48 Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, *IEEE Trans. Neural Netw. Learn. Syst.*, 2022, **33**, 6999–7019.

49 K. O'Shea and R. Nash, *Int. J. Res. Appl. Sci. Eng. Technol.*, 2015, **10**, 943–947.

50 E. L. Buehler and M. J. Buehler, *Biomed. Eng. Adv.*, 2022, **4**, 100038.

51 Z. Yang, C. H. Yu and M. J. Buehler, *Sci. Adv.*, 2021, **7**, 1–10.

52 D. Repecka, V. Jauniskis, L. Karpus, E. Rembeza, I. Rokaitis, J. Zrimec, S. Poviloniene, A. Laurynenas, S. Viknander, W. Abuajwa, O. Savolainen, R. Meskys, M. K. Engqvist and A. Zelezniak, *Nat. Mach. Intell.*, 2021, **3**, 324–333.

53 R. Cao, C. Freitas, L. Chan, M. Sun, H. Jiang and Z. Chen, *Molecules*, 2017, **22**, 1732.

54 Y.-C. Hsu, C.-H. Yu and M. J. Buehler, *Matter*, 2020, **3**, 197–211.

55 W. Lu, Z. Yang and M. J. Buehler, *J. Appl. Phys.*, 2022, **132**, 74703.

56 A. Strokach, D. Becerra, C. Corbi-Verge, A. Perez-Riba and P. M. Kim, *Cell Syst.*, 2020, **11**, 402–411.e4.

57 R. You, S. Yao, H. Mamitsuka and S. Zhu, *Bioinformatics*, 2021, **37**, i262–i271.

58 Z. Yang and M. J. Buehler, *npj Comput. Mater.*, 2022, **8**(1), 1–13.

59 W. Lu, D. L. Kaplan and M. J. Buehler, *Adv. Funct. Mater.*, 2023, 2311324.

60 H. Zhang, W. Du, J. Shan, Q. Zhou, Y. Du, J. B. Tenenbaum, T. Shu and C. Gan, *arXiv*, 2023, preprint, arXiv:2307.02485, DOI: **10.48550/arXiv.2307.02485**.

61 L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei and J.-R. Wen, *Front. Comput. Sci.*, 2024, **18**, 186345.

62 Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang and T. Gui, *arXiv*, 2023, preprint, arXiv:2309.07864, DOI: **10.48550/arXiv.2309.07864**.

63 D. A. Boiko, R. MacKnight, B. Kline and G. Gomes, *Nature*, 2023, **624**, 570–578.

64 A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White and P. Schwaller, *Nat. Mach. Intell.*, 2024, **6**, 525–535.

65 S. Gao, A. Fang, Y. Huang, V. Giunchiglia, A. Noori, J. R. Schwarz, Y. Ektefaie, J. Kondic and M. Zitnik, *Empowering Biomedical Discovery with AI Agents*, 2024.

66 OpenAI API, https://openai.com/blog/openai-api.

67 M. Sikora, J. I. Sułkowska and M. Cieplak, *PLoS Comput. Biol.*, 2009, **5**, e1000547.

68 J. I. Sułkowska and M. Cieplak, *J. Phys.: Condens. Matter*, 2007, **19**, 283201.

69 T. Ackbarow, X. Chen, S. Keten and M. J. Buehler, *Proc. Natl. Acad. Sci. U. S. A.*, 2007, **104**, 16410–16415.

70 T. P. Knowles and M. J. Buehler, *Nat. Nanotechnol.*, 2011, **6**(8), 469–479.

71 Z. Xu and M. J. Buehler, *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.*, 2010, **81**, 061910.

72 Z. Qin and M. J. Buehler, *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.*, 2010, **82**, 061906.

73 T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest and X. Zhang, Large Language Model based Multi-Agents: A Survey of Progress and Challenges, *arXiv*, 2024, preprint, arXiv:2402.01680, DOI: 10.48550/arXiv.2402.01680.

74 L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin and T. Liu, A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions, *arXiv*, 2023, preprint, arXiv:2311.05232, DOI: 10.48550/arXiv.2311.05232.

75 A. Dorri, S. S. Kanhere and R. Jurdak, *IEEE Access*, 2018, **6**, 28573–28593.

76 C. P. Gomes, J. Bai, Y. Xue, J. Björck, B. Rappazzo, S. Ament, R. Bernstein, S. Kong, S. K. Suram and R. B. van Dover etal, *MRS Commun.*, 2019, **9**, 600–608.

77 J. H. Montoya, K. T. Winther, R. A. Flores, T. Bligaard, J. S. Hummelshøj and M. Aykol, *Chem. Sci.*, 2020, **11**, 8517–8532.

78 A. G. Kusne and A. McDannald, *Matter*, 2023, **6**, 1880–1893.

79 Q. Yang, Y. Liu, T. Chen and Y. Tong, *ACM Trans. Intell. Syst. Technol.*, 2019, **10**, 1–19.

80 C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li and Y. Gao, *Knowl.-Based Syst.*, 2021, **216**, 106775.

81 I. E. Lagaris, A. Likas and D. I. Fotiadis, *IEEE Trans. Neural Netw.*, 1998, **9**, 987–1000.

82 M. Raissi, P. Perdikaris and G. E. Karniadakis, *J. Comput. Phys.*, 2019, **378**, 686–707.

83 F. Y. Liu, B. Ni and M. J. Buehler, *Extreme Mech. Lett.*, 2022, **55**, 101803.

84 C. H. Yu and M. J. Buehler, *APL Bioeng.*, 2020, **4**, 16108.

85 M. J. Buehler, *J. Appl. Phys.*, 2023, **134**, 084902.

86 B. Ni, D. L. Kaplan and M. J. Buehler, *Chem*, 2023, **9**, 1828–1849.

87 M. AlQuraishi, *Curr. Opin. Chem. Biol.*, 2021, **65**, 1–8.

88 W. Gao, S. P. Mahajan, J. Sulam and J. J. Gray, *Patterns*, 2020, **1**, 100142.

89 Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. Awadallah, R. W. White, D. Burger and C. Wang, *arXiv*, 2023, preprint, arXiv:2308.08155, DOI: 10.48550/arXiv.2308.08155.

90 J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo and Y. Liu, *Neurocomputing*, 2024, **568**, 127063.

91 B. Ni, D. L. Kaplan and M. J. Buehler, *Sci. Adv.*, 2023, 1–15.

92 N. Rego and D. Koes, *Bioinformatics*, 2015, **31**, 1322–1324.

93 W. Kabsch and C. Sander, *Biopolymers*, 1983, **22**, 2577–2637.

94 P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski and M. J. D. Hoon, *Bioinformatics*, 2009, **25**, 1422.

95 A. R. Atilgan, S. R. Durell, R. L. Jernigan, M. C. Demirel, O. Keskin and I. Bahar, *Biophys. J.*, 2001, 505–515.

96 P. Doruker, A. R. Atilgan and I. Bahar, *Proteins: Struct., Funct., Bioinf.*, 2000, 512–524.

97 A. Bakan, L. M. Meireles and I. Bahar, *Bioinformatics*, 2011, **27**, 1575–1577.

98 LlamaIndex (formerly GPT Index), a data framework for LLM applications, https://github.com/run-llama/llama_index.