

TOWARDS AN AI DRIVEN SIMULATION FRAMEWORK

S. Rifat*, N. Ebrahimi Pour*, F. Dressel*, S. Roller*†

* German Aerospace Center, Institute of Software Methods for Product Virtualization, Nöthnitzer Straße 46b, 01187 Dresden, Germany

† TU Dresden, Chair of Software Methods for Product Virtualization, Nöthnitzer Straße 46a, 01187 Dresden, Germany

Abstract

The enhancement of computational power and the ability to facilitate sophisticated real-world applications has given rise to the demand for the efficient execution of such simulations. They are inherently complex and involve multiple physical phenomena, that have to be taken into account. In recent years, there have been advances in numerical methods to allow for more efficient computation. Nevertheless, the available software packages are highly complex, necessitating special training for their usage. The configuration of such a complex simulation is not straightforward. Furthermore, scientists are required to invest time and have in-depth knowledge of the code to understand its functionalities. Consequently, this approach requires a significant investment of time and resources.

Currently, Large Language Models (LLMs) are being appreciated for its recent advancement in software development such as code generation, function-level performance evaluation etc. Despite such remarkable achievements, LLMs still struggle to analyze and understand the infrastructure of an operational software stack. This step is crucial for Artificial Intelligence (AI) driven automated development of a private or customized industrial research workflow. In this work we introduce a LLM based multi-agent framework which is able to collaborate between individual modules of a complex software architecture and compile the whole simulation from a single user prompt in real time. Code repository level agents are specialized in navigating the code base and retrieving the file to be compiled for specific simulation scenario by comparing the user instruction against the code base which is stored in the agent's memory. Furthermore, the integration of coupling level agents provide a baseline for combining multiple libraries or modules which is an essential process in task-agnostic Software Engineering (SE) workflows.

We investigate the potential of LLMs in assisting users with information they need to get familiar when working with unexplored software packages. For this purpose the source code of the solver, a highly parallel computational fluid dynamic solver, is provided to the LLM for demonstration. In first studies the utilised LLM algorithm provided accurate responses to questions asked by the user to understand the capabilities of the solver. Extensive experiments on the role of multi stage agents reveals the compatibility of LLMs in efficient aerodynamic simulation tasks.

Keywords

Large Language Models, Source Code Generation, Source Code Analysis, Agentic Framework, Numerical Simulation, Virtual Product, Aerospace Engineering

1. INTRODUCTION

As aerodynamic systems continue to evolve and become more complex by integrating high-fidelity simulations, multidisciplinary models, and real-time data streams, the underlying software architecture must also be made easily accessible and effortless for users to meet these demands. The increasing software complexity arises from the need to coordinate heterogeneous tools, data formats, computational platforms, and domain-specific solvers within a unified framework. A loosely coupled architectural approach is therefore essential, promoting modularity and maintainability while enabling parallel development across disciplines such as fluid dynamics, structural mechanics, control systems, and embedded software. The proposed approach addresses these challenges by supporting component-based integration, streamlining large-scale aerodynamic workflows, promising a flexible foundation for deployment within complex industrial applications.

LLMs have demonstrated substantial capabilities across a broad spectrum of natural language processing tasks [1–6]. Recent studies further suggest that these models may be beginning to exhibit characteristics aligned with aspects of artificial general intelligence [7, 8]. Among the most impactful applications of LLMs is automated code gener-

ation, wherein natural language inputs are translated into executable source code. Models such as ChatGPT have significantly advanced this domain, enabling automation in tasks such as code completion, language translation, and code repair. Despite these advances, current LLMs rely heavily on large-scale training datasets and exhibit optimal performance primarily in zero-shot or few-shot contexts. A persistent challenge lies in their propensity to produce hallucinated outputs—responses that, while syntactically and semantically coherent, may deviate from factual accuracy, intended meaning, or contextual relevance [9]. This limitation poses a significant barrier to their reliable deployment in domains that require deterministic and reproducible outcomes, particularly in engineering and safety-critical applications. Addressing this challenge remains an essential step toward broader, dependable adoption of LLMs in such contexts. Recent progress in long-context language models (LCLMs) shows strong potential to reshape how we approach Retrieval-Augmented Generation (RAG) systems [10]. Thanks to their extended context windows, LCLMs can handle much larger amounts of information at once, which reduces the need for complex retrieval pipelines that were previously necessary due to context length constraints. This makes it easier to update

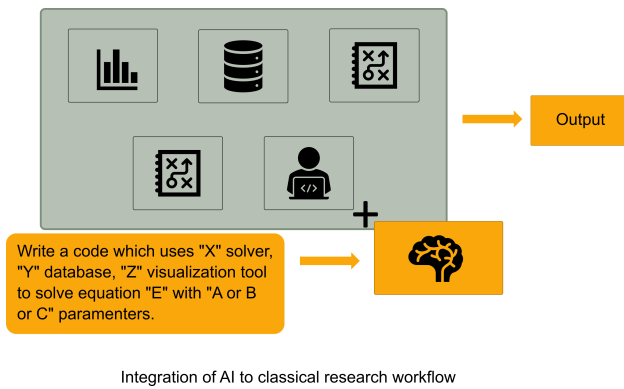


FIG 1. Augmentation of existing software architecture with LLMs

or modify the knowledge being used, since relevant context can simply be inserted directly. In fact, many LCLMs are now capable of holding entire knowledge bases within a single input window, effectively acting as a temporary working memory for handling new queries. Additionally, by integrating LCLMs into RAG-based frameworks, it becomes possible to stabilize the model's responses—reducing variability and lowering the chances of generating hallucinated or misleading content. Despite the strong performance of RAG, its practical adoption is often constrained by the complexity of its multi-stage processing pipelines. Building on this foundation, recent developments have introduced more flexible and intelligent frameworks—such as multi-agent LLM systems—which leverage advanced capabilities including context-sensitive memory [11], multi-step planning [12], and seamless integration with external tools [13]. Drawing parallels with the development of human intelligence, ongoing research has highlighted the potential of enhancing LLMs through tool augmentation, significantly boosting their effectiveness in complex problem-solving scenarios [13–16]. A typical agentic setup [17, 18] involves decomposing a task into a series of manageable subtasks, each handled through collaboration between an instructor agent and one or more assistant agents, each equipped with its own specialized tools. The instructor issues task-specific commands, and the assistant agents respond with partial solutions. Through iterative, multi-turn communication, these agents work together to generate comprehensive outputs—ranging from software artifacts and mathematical results to scientific conclusions. This collaborative, language-based coordination among agents enables the system to address a wide spectrum of challenging tasks, including but not limited to mathematical reasoning [12, 19], software development [17, 20], game-playing [21–24], social simulations [25–28], and scientific research [29, 30]. The incorporation of agentic systems into aerodynamic workflow streamlines the resolution of software-related challenges, reducing the complexity typically encountered before running meaningful simulations. It is also worth noting that standard benchmarks often do not effectively capture these specialized capabilities. To overcome this limitation, we utilize a novel, customized benchmark designed to evaluate LCLMs under more practical and realistic conditions. Using this benchmark, we investigate improvements in LCLMs in-context retrieval and reasoning skills, observing a steady enhancement in access to detailed information alongside the advancement of chatbot technologies.

In our work we show a multi-agent based simulation

approach which has the capability to interact with custom workflows and code repositories having predefined containers which ensures consistent environment setting desired by the software package. Furthermore, we present a RAG based communication system through which user can interact with our aerodynamic simulation framework (*Ateles*) [31] to attain useful information. Additionally, a chatbot with prolonged memory which can be utilized to retrieve granular level information about specific files or libraries will be presented. Lastly, we evaluate the mentioned approaches and compare the capabilities of LLM (llama, qwen and mistral) with varying ranges of parameters (1 billion to 13 billion) and conclude our work with a short summary and outlook.

2. METHODOLOGY

2.1. Problem overview

Computer simulation is the process of mathematical modelling, performed on a computer, which is designed to predict the behaviour of or the outcome of a real-world or physical system. From the perspective of traditional aerodynamic simulation scheme, the region surrounding an object—such as an aircraft or a vehicle—is broken down into a mesh which is composed of numerous small cells. Each cell contains essential flow variables like velocity, pressure, and temperature, which evolve over time and space according to multi-disciplinary numerical methods such as Navier-Stokes equations, Euler equations, Maxwell equations etc. Generally, the simulation begins with specified initial and boundary conditions, typically derived from experimental data or real-world measurements, and advances by iteratively solving the governing equations to update the flow field. The accuracy of the simulation is dependent on mesh resolution, the quality of input data, and the numerical algorithms employed. Higher-fidelity predictions of aerodynamic performance—such as lift, drag, and flow separation—are achieved through finer meshes, more accurate physical models and well-defined boundary conditions across various operating scenarios.

From a software engineering standpoint, the integration of these components necessitates a thorough understanding of the underlying mathematical principles, as well as a detailed analysis of the supporting software infrastructure. Addressing the inherent complexity of multi-scale and multi-physics problems requires the deployment of advanced, specialized, and scalable computational tools. Ensuring seamless interoperability between heterogeneous modules depends on the adoption of standardized interfaces and clearly defined communication protocols—both of which are essential for facilitating effective cross-disciplinary collaboration. In an era of rapid technological advancement, the ability to efficiently incorporate novel algorithms, data sources, and simulation environments is increasingly critical. In light of these demands, this study explores the applicability of LLMs, alongside the emerging paradigm of agentic AI systems, as a means to establish a flexible, extensible, and robust framework for aerodynamic simulation test cases.

2.2. Retrieval Augmented Generation (RAG)

LLMs have made impressive strides in fields like natural language processing [1, 32], text generation [7], and even programming tasks [33, 34]. However, despite their capabili-

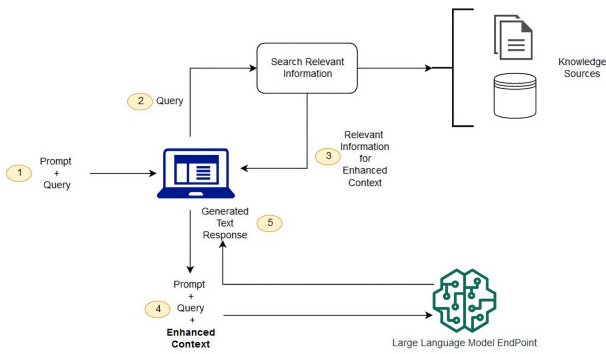


FIG 2. Retrieval Augmented Generation [36]

ties, LLMs still struggle with certain limitations; most notably, their tendency to produce content that appears convincing but is actually incorrect or fabricated [33, 35]. This issue becomes especially problematic when applying LLMs to more complex, task-driven scenarios that go beyond simple conversations [7].

To produce accurate, domain-specific outputs, LLMs must be adapted to the knowledge context in which they operate. RAG achieves this adaptation by augmenting the model's parametric knowledge with external, structured information sources. Rather than relying exclusively on internally stored representations, a RAG system retrieves relevant documents from a designated knowledge base and conditions the model's generative process on both the user query and the retrieved evidence [17]. A canonical RAG pipeline consists of three core stages: indexing, retrieval, and generation.

- **Indexing:** Data is preprocessed, transformed into vector embeddings, and stored in a searchable index to enable efficient access.
- **Retrieval:** This stage is often subdivided into (i) pre-retrieval, where search parameters and filtering criteria are defined; (ii) retrieval, in which the most semantically relevant items are identified; and (iii) post-retrieval, where results are refined, ranked, and filtered for contextual alignment.
- **Generation:** The retrieved context is incorporated with the query to produce factually grounded and contextually coherent responses.

Figure 2 shows such system where documents are being retrieved from vector databases to generate LLM responses. Document retrieval is powered by embedding models for example, BERT or OpenAI's text-embedding-ada-002 which encode text into high-dimensional vector representations suitable for similarity search. The underlying knowledge base may comprise corporate guidelines, technical specifications, or domain-specific problem-solution records. For vector storage, popular backends include FAISS, Chroma, and LanceDB. In this work, LanceDB is employed as the primary embedding store.

2.3. Instruction Tuning of LLMs and Tool Calling

LLMs can be prompted to perform a diverse range of natural language processing (NLP) tasks by providing task-specific examples within the input. However, a common limitation is hallucination, which arises because the standard training objective for most modern LLMs—predicting the next token from large-scale internet text—differs from the intended goal of “following user instructions in a helpful and safe manner” [1, 37–40]. Instruction-tuned LLMs address this

gap by incorporating a Reward Model (RM) trained on human-labeled datasets, enabling them to align predictions with user intent. InstructGPT represents a pioneering example of this paradigm, demonstrating the ability to follow instructions for tasks such as code summarization, answering code-related queries, and, in some cases, adhering to instructions in multiple languages—even when such instructions are underrepresented in its fine-tuning data distribution. Empirical evidence shows that instruction fine-tuning scales effectively with both the number of training tasks and model size [41]. Leveraging these capabilities, we design our pipeline to facilitate precise and context-aware interaction with our custom codebase. LLMs have shown impressive zero-shot and few-shot capabilities across a wide range of natural language processing tasks [1, 2]. However, these models still face fundamental limitations that scaling alone cannot fully resolve. Notable challenges include their inability to access up-to-date information [9], a tendency to generate fabricated facts [42], reduced effectiveness with low-resource languages [43], insufficient accuracy in complex mathematical computations [44], and a lack of awareness regarding temporal context [45].

One effective approach to overcome these issues is to augment LLMs with external tool integration, such as search engines, calculators, or scheduling systems. Our research harnesses the tool-calling capabilities of LLMs to facilitate the smooth discovery and integration of such APIs. The tool-calling process comprises three fundamental phases: (i) Sampling API Calls – For each API, specialized prompts are developed to instruct the language model to annotate sample inputs with the appropriate API calls. (ii) Executing API Calls – These annotated API calls are then executed to obtain results, with the execution method tailored to the specific type of API, such as running external scripts, invoking auxiliary neural networks, or querying retrieval systems. (iii) Filtering API Calls – The utility of each API call is assessed by comparing the model's performance under three scenarios: excluding the call, including the call without its output, and including the call with its output. An API call is deemed advantageous if providing both the input and output enhances the model's next-token prediction accuracy compared to the other scenarios.

2.4. Agentic orchestration

In addressing complex reasoning tasks, such as predicting outcomes in aerodynamic experiments, scientists typically decompose problems into a sequence of manageable steps. For instance: “A paper airplane glides 2 meters when thrown lightly; when thrown with greater force, it glides 5 meters; therefore, increasing the throwing force increases the glide distance.” Our workflow mirrors this approach by generating structured chains of reasoning or thought which are ordered intermediate conclusions that systematically lead to the final answer. We show that sufficiently large language models can produce such chains of thought when few-shot prompting provides examples of step-by-step reasoning. This methodology enables the systematic and interpretable resolution of specific tasks, enhancing both clarity and reliability in model-driven analyses.

When fine-tuned with zero-shot instruction, equipped with domain-specific tools, and applied to specialized tasks, LLMs evolve into agents capable of complex problem-solving. Leveraging linguistic interaction and chain-of-thought (CoT) reasoning [12], these agents can tackle diverse challenges. Agentic orchestration can be cate-

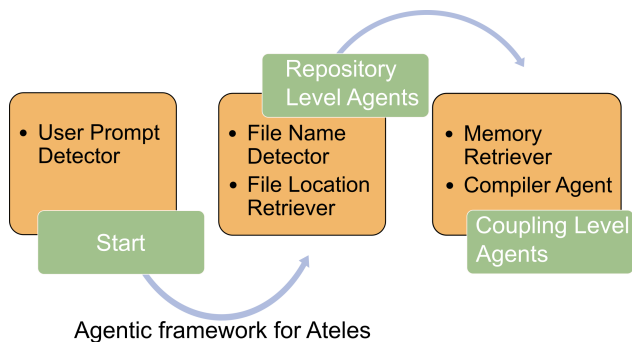


FIG 3. Multi Agent Orchestration

gorized into three variants. Standalone Large Language Models (LLMs) are powerful systems that can perform tasks and generate text on their own, without needing other agents or external tools. They are highly versatile, capable of handling everything from answering questions to summarizing content or creating new text. Yet, despite their independence, they have limitations, such as lacking access to real-time information, struggling with complex reasoning, and having limited memory for long-term context. Single Agent Systems (SAS) take this idea further by focusing on a single autonomous agent optimized for a particular task. These systems are simpler to design and maintain and can perform efficiently, but they often lack flexibility and struggle in dynamic, unpredictable environments [13]. In contrast, Multi Agent Systems (MAS) bring multiple autonomous agents together, allowing them to collaborate, share knowledge, and coordinate actions. This collaboration makes them more adaptable, scalable, and capable of tackling complex, large-scale challenges that a single agent could not handle alone [11].

3. RESULTS AND VALIDATION

In this section we perform a detailed investigation of our proposition. To prepare a simulation framework we have to confirm that with optimized architecture, the agents are able to understand the methodology of accomplishing a simulation process. Furthermore, for user intent wise use cases, the system should be able to navigate through the codebases and understand the documentation of respective code. The adopted methods for testing the current frameworks are discussed below.

3.1. Validation Case - CFD Solver

To validate our algorithm we consider *Ateles* [31], a fluid dynamic solver, that is based on a high-order modal discontinuous Galerkin method. Yielding high convergence rates for smooth solutions and thus very precise simulation results (ref). Several fluid dynamic equations are implemented in *Ateles* [31] allowing for the simulation of different numerical problems such as one directional fluid-structure acoustics interactions or aeroacoustics. It provides different time-stepping schemes such as Runge-Kutta 4th step or the Implicit-mixed-explicit Runge-Kutta time stepping scheme (IMEX). Geometries are modeled as immersed boundaries, which can move over time using the Brinkman penalization method. Different boundary conditions are implemented, fulfilling the requirements of users for different problems. The solver is written in modern Fortran including 1660 subroutines and 340 functions. In total the solver consists of over 200.000 lines of code. *Ateles* is a component of

the APES simulation framework [31], which includes pre- and post-processing tools. It is also designed for hybrid parallelization and is therefore well suited for large scale simulations on supercomputing systems. The solver provides a user manual with various small examples to test the different implementations of the solver.

3.2. Experiment setup

We first evaluate the impact of prompt engineering on our simulation test cases. Separate investigations were conducted for standalone LLMs, single-agent systems, and multi-agent systems, each requiring tailored datasets to suit their respective methodologies.

3.2.1. Framework functionality testing

As an initial step, we examined the framework's basic functionality by tracking LLM responses under different configurations. Few-shot prompt injections produced varied outputs, highlighting the importance of prompt design. When interacting with structured data, LLMs must interpret the underlying data grammar. To test this capability, we represented custom code repositories in structured formats such as tree hierarchies or JSON. We then constructed a series of prompts incorporating structured examples from the *Ateles* [31] codebase to evaluate whether standalone LLMs could effectively process and utilize such data. These experiments yielded valuable insights that informed the iterative refinement of our framework. Subsequent integration of tool usage and multi-agent systems produced a functional AI-driven simulation workflow. Figure 3 shows the sequence of agentic architecture. Two specialized agent were deployed into the framework. The repository-level agent interacts directly with custom code repositories. In its initial stage, it parses user input to determine intent, extracting key terms and identifying the corresponding configuration files required to execute the desired simulation. As *Ateles* [31] supports CFD simulations, selecting the correct equations is essential for scientific accuracy; the repository-level agent facilitates this navigation process. Given that the workflow is designed to autonomously execute simulations, there is an inherent risk of uncontrolled interaction with the operating system, which could compromise the working environment or introduce security vulnerabilities. To mitigate this, we employ a sandbox agent, capable of connecting securely to remote workstations and high-performance computing clusters. In our implementation, this connection is facilitated via SSH. On the remote workstation, a dedicated podman container—pre-configured with all necessary libraries and dependencies—remains active. This setup creates a clear separation between the local environment and the virtual simulation environment, thereby enhancing security and reducing dependency management complexity for researchers. With this architecture, a single user command can initiate a complete CFD simulation in the remote containerized environment, eliminating the need for CFD specialists to manage software installation or architecture intricacies, and streamlining the overall research workflow.

3.2.2. Custom Dataset

The comprehension capability of LLMs was evaluated using a manually curated dataset, enabling us to assess their applicability to our specific use case. In version 0.1 of the dataset, 7 simple question answer pair were formulated. Furthermore, we prepared 30 distinct test question-answer pairs as dataset version 0.2. Of these, 18 required concise

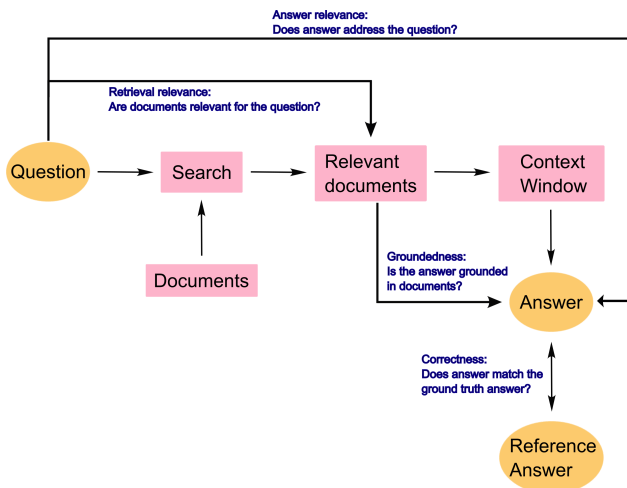


FIG 4. Evaluation with LLM as a judge

responses, 10 focused on in-depth methodological topics, and 2 posed significant challenges as they involved custom repository-level coding skills.

3.2.3. Evaluation metric

Two approaches have been tested to check the applicability of agentic framework. For testing the workflow, off course a human evaluator is better for observation. We check manually the success of the pipeline.

Furthermore, for evaluating RAG approach along with its generation capability, we utilize LLMs as a judge. The term LLM-as-a-judge [46] describes the use of large language models to evaluate, rank, or score outputs produced by humans, other models, or multiple candidates from the same system. Unlike conventional evaluation metrics such as BLEU [47] or ROUGE [48], which depend on fixed statistical formulas, LLM judges leverage advanced semantic understanding and reasoning to deliver context-aware, fine-grained assessments. This approach has been applied in domains including natural language generation, code review, and creative content analysis. Empirical studies [49, 50] indicate that LLM judges can match or even surpass traditional metrics in aligning with human judgments.

Figure 4 shows how the responses are being measured by LLM judges. Correctness denotes the degree of alignment between the generated output and the established reference answer. Relevance evaluates the extent to which the system's response directly addresses the user's query. Retrieval relevance assesses whether the retrieved documents accurately correspond to the intended embedded segments stored within the vector database. Groundedness measures the extent to which the generated content is substantiated by the retrieved documents.

3.3. Results and discussion

The chronological evaluation overview of our work is presented in this section. We had to test the feasibility of working with AI agents along with the capability of tool calling decision making process of the system.

3.3.1. Simulation agents

Table 1 presents the manually measured success rate of locating and executing simulation files across structured and unstructured repository formats. The results highlight

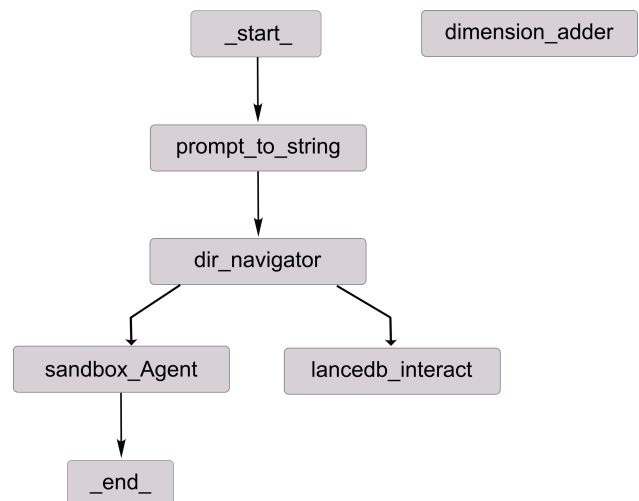


FIG 5. Integration of simulation agents as a component of multi-agent system

a clear limitation of smaller and medium-sized LLMs (e.g., LLaMA-3B, Alfred-40B): they consistently failed to navigate tree or JSON-based repositories, particularly when dealing with complex nested structures and parent-child relationships. This failure was further amplified by vector database chunking, which splits JSON structures into fragments and causes retrieval systems to recombine them incorrectly, often resulting in contextual mismatches.

Instruction fine-tuning with repository path examples slightly improved navigation, but performance deteriorated as repository size and complexity increased. In contrast, when a dedicated file-search tool was introduced into the agentic system, the success rate dramatically improved. By allowing the system to explicitly call the tool for file discovery, simulations could reliably identify and execute the correct files, achieving a 100% success rate once the file location was found. This confirms that tool integration is essential for reliable repository-level interaction.

Providing diverse, well-crafted examples within the prompt markedly improved results. These examples, containing key domain-specific terms, enhanced chunk retrieval from the vector store and preserved structural integrity. Larger models, such as Alfred-40B, could then accurately reconstruct the full structure and identify the exact location of the required configuration file.

Figure 5 shows the integration of directory navigator agent and sandbox agent to the workflow. To support autonomous operation, a simple finder tool was integrated into the pipeline, enabling the LLM to select and invoke tools according to user instructions. When a file search is required, the model extracts key terms from the query, calls the finder tool, and retrieves the file's location. In figure 5, the *prompt_to_string* method extracts meaningful keywords from the user intended simulation use case. Then, the *dir_navigator* approach sends these keywords to our vector database which is *lancedb_interact*. From the vector store, the location of the file along with contents are retrieved. Eventually the file along with its location and content is passed to the *sandbox_agent*, which operates in a remote workstation with all necessary dependencies pre-installed in a containerized environment. Moreover, to separate 1D, 2D or 3D use cases, another tool is integrated into the system. Based on user requirements, it has to be ensured that most certain variables are present in the requested chunks. The method *dimension_adder* assists in resolving such edge

Repository format	Prompt injection (Llama 3B)	Prompt injection (Llama 3B instruct)	Prompt injection (Alfred 40B instruct)	Mentioned LLMs with tool call
Tree structure	0%	0%	0%	100%
JSON	5%	5%	7%	100%
With example in prompt	20%	20%	40%	100%

TAB 1. Investigation on structured data understanding by LLMs and integration of tools to the workflow

cases. Coordinated tool calls between the sandbox agent and the repository-level agent then execute the simulation seamlessly.

3.3.2. Retrieval methodology testing

We conducted a further investigation into content generation capabilities. Initial responses from Gemma 1B and LLaMA 3B were systematically recorded and analyzed. In agent based system, multiple LLMs communicate between each other to resolve user requirements. It has been noted that, small language models can be utilized to reduce time and space complexity of the whole agentic system [51, 52]. Therefore, for our baseline framework we have chosen the small language models which are locally runnable with simple CPU architecture. Moreover as Gemma 1B was not trained on instruction-based datasets, the LLaMA 3B instruction-tuned model was employed as the evaluator to ensure reliable assessment. For shorter queries, the token consumption was found to be relatively balanced, indicating efficient response generation for concise prompts. The whole system was tested in 4 different phases.

Phase 1: Feasibility Study with Version-0.1 Dataset

The feasibility of the Retrieval-Augmented Generation (RAG) approach was initially assessed using a version-0.1 dataset consisting of seven simple question–answer pairs. These questions were derived directly from documentation, such as whether Ateles is parallelizable with MPI or whether mesh generation is supported. Results presented in Table 2 and Table 3 indicate that the RAG approach is effective in navigating documentation and retrieving factually grounded answers. Furthermore, the LLM which are not instruction fine-tuned can not be used as LLM judge as they are not pre-trained for such tasks.

Phase 2: Expanded Evaluation with Version-0.2 Dataset

A more comprehensive evaluation was conducted with a version-0.2 dataset containing 30 question–answer pairs. Retrieval and generation quality were assessed using large language models (LLMs) as evaluators. The following LLMs were benchmarked: a) InternLM2 (7B), b) Phi-4 (14B), c) Alfred (40B), d) Mistral (7B), e) LLaMA-3 (3B), f) Gemma (1B). The key findings of this phase are explained below.

Top-k retrieval: Selecting the top 5 most relevant chunks yielded optimal performance. Retrieving more documents increased hallucination risk.

Model reliability: Models with more than 7B parameters performed consistently better in interpreting queries and grounding responses in retrieved evidence.

Small model behavior: Smaller models handled direct and straightforward questions but struggled with reasoning-intensive or technical queries, such as simulation methodologies or code dependency analysis.

Exceptions: Qwen (2.5B) occasionally performed beyond expectation, suggesting that some smaller models may still excel at targeted tasks.

Phase 3: Limitations of Small and Large Models

Small models: Adequate for simple factual queries but

weak in handling logical reasoning, multi-step dependencies, and technical detail.

Large models: More powerful, yet still prone to errors on advanced simulation-related queries, such as identifying code vulnerabilities or intricate dependencies.

Observation: Performance declines as query complexity increases, highlighting the importance of contextual guidance during the retrieval–generation process.

Phase 4: Memory Integration into the Agentic System

To improve reliability, memory augmentation was incorporated into the agentic pipeline. This enabled multi-turn conversations where users could iteratively refine queries and guide the system toward precise answers. For some sample question such as: "Can you provide the scheme, tracking and mesh information for euler 1D equation?", the one shot answer from all the LLM mentioned above generates results having a precision of 0.163, recall of 0.167 and fmeasure of 0.164. Whereas, sequence of questions such as: a) "What is the procedure for running one of the testcases?", b) "Okay, let's say, I am in the folder ateles/tutorial/testcase/euler. Now what do i do? ", c) "In the lua files for 1D Euler equation, what parameters should i be configuring?", d) "Provide me with a template for writing equation definitions?", e) "Provide a template for writing the equation definitions for the Euler equation system in Lua.", f) "Why are you only providing global variables? Can you provide the scheme, tracking and mesh information for euler 1D equation?" lead towards the required detailed response from the codebase. This is really exciting because, leveraging such high level code navigation, the simulation can be run from simple prompts. The multi-faceted advantages brought by such conversational systems are:

Guided exploration: Multi-step queries enabled the system to progressively narrow down answers, such as identifying exact simulation variables.

Configuration file handling: The agent could locate and modify configuration files by inserting, deleting, or adjusting parameters according to user-defined conditions (e.g., environment variables or multi-physics setups).

Improved contextual understanding: Continuous conversation allowed the agent to build awareness of simulation pipelines, enhancing its ability to provide grounded and actionable responses.

4. CONCLUSION

In this work, we analyzed the applicability of LLM based agentic orchestration system to augment legacy codes of Aerodynamic simulation system with currently emerging natural language technologies. These systems are promised to be an important factor for future digital twin concepts. Our inquiry into such systems shows unfolds the potential of agent based simulation frameworks. Furthermore, the addition of memory significantly improved the agent's ability to interact with both documentation and codebases. This enhancement is especially important for tasks requiring detailed reasoning, such as preparing simulation pipelines from natural language prompts. When integrated

Inputs	Reference	Outputs	Correctness	Groundness	Relevance	Retrieval relevance	Tokens
Central comp.	TreElm	TreElm	true	true	true	true	915
Script format	Lua	Lua	true	true	true	true	1134
Suite	APES	University of Siegen	false	true	false	false	461
Fortran env variable	FC	FC	true	true	true	false	1210
APES relevant components	Aotus, TreELM	Euler, Gaussian, Planar	false	true	true	true	731
MPI parallelization	Yes	Yes	true	false	true	true	997
Mesh generation	Yes	Yes	true	true	true	false	1001

TAB 2. RAG evaluation: Answer generated through 1B gemma model evaluated through 3B llama model

Inputs	Reference	Outputs	Correctness	Groundness	Relevance	Retrieval relevance	Tokens
Central comp.	TreElm	TreElm	true	true	true	true	897
Script format	Lua	Lua	true	true	true	true	940
Suite	APES	Do not know	false	false	false	false	396
Fortran env variable	FC	FC, export FC = mpif90	true	true	true	true	1153
APES relevant components	Aotus, TreELM	Lua, source term	false	false	true	true	807
MPI parallelization	Yes	Yes	true	false	true	true	881
Mesh generation	Yes	No built-in, TreELM	false	true	true	true	906

TAB 3. RAG evaluation: Answer generated through 3B llama model evaluated through 3B llama model

Models	Retrieval relevance	Relevance	Groundness	Correctness
Gemma 1b	66.6%	70%	60%	40%
Qwen 2.5b	83.3%	60%	53.3%	56.7%
Llama 3b	73.3%	33.3%	50%	26.7%
Mistral 7b	73.3%	73.3%	70%	36.7%
Internlm 7b	76.7%	73.3%	60%	53.3%
Phi 14b	80%	73.3%	66.7%	43.3%
Alfred 40b	73.3%	76.6%	60%	53.3%

TAB 4. RAG evaluation: Perfomance of RAG on different models on Dataset version 02

into existing systems, memory-augmented conversational agents are expected to: **(i)** increase the accuracy of simulation setup, **(ii)** reduce reliance on parametric memory, **(iii)** enhance user trust by grounding answers in retrieved documents, **(iv)** enable dynamic code modifications tailored to simulation needs.

The integration of additional tools has the potential to create a more advanced research environment. We are actively exploring the feasibility of such systems, and our future research directions include, but are not limited to:

- Testing more advanced LLMs with more parameters on the framework.
- Extension of test dataset.
- Extension of the framework to other use cases beside Ateles [31].
- Integrate High Performance Computing (HPC) systems into the environment agents.
- Transform the existing engineering workflows to more simple and faster frameworks.

We plan to collaborate with advanced HPC labs and industry experts to develop the framework into an AI based multi-physics simulation framework.

Contact address:

safayet.rifat@dlr.de

References

- [1] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [2] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- [3] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.
- [4] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.
- [5] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [6] Google DeepMind. Gemini: Google's multimodal large language model. <https://deepmind.google/technologies/gemini/>, 2023. Accessed via official blog post or press release.
- [7] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- [8] Michal Kosinski. Theory of mind may have spontaneously emerged in large language models. *arXiv preprint arXiv:2302.02083*, 2023.

- [9] Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8460–8478, Dublin, Ireland, 2022. Association for Computational Linguistics. DOI: [10.18653/v1/2022.acl-long.579](https://doi.org/10.18653/v1/2022.acl-long.579).
- [10] Mingjian Jiang, Yangjun Ruan, Luis Lastras, Pavan Kapanipathi, and Tatsunori Hashimoto. Putting it all into context: Simplifying agents with LCLMs. *arXiv preprint arXiv:2505.08120*, 2025. Preprint.
- [11] Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, Zhaozhuo Xu, and Chaoyang He. Llm multi-agent systems: Challenges and open problems. *arXiv preprint*, (arXiv:2402.03578), 2024. DOI: [10.48550/arXiv.2402.03578](https://doi.org/10.48550/arXiv.2402.03578).
- [12] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.
- [13] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.
- [14] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023.
- [15] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.
- [16] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.
- [17] Xinying Qian, Ying Zhang, Yu Zhao, Baohang Zhou, Xuhui Sui, Li Zhang, and Kehui Song. Timer: Time-aware retrieval-augmented large language models for temporal knowledge graph question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6942–6952, Miami, Florida, USA, 2024. Association for Computational Linguistics. DOI: [10.18653/v1/2024.emnlp-main.394](https://doi.org/10.18653/v1/2024.emnlp-main.394).
- [18] Ruixin Hong, Hongming Zhang, Hong Zhao, Dong Yu, and Changshui Zhang. Faithful question answering with monte-carlo planning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3944–3965, Toronto, Canada, 2023. Association for Computational Linguistics. DOI: [10.18653/v1/2023.acl-long.218](https://doi.org/10.18653/v1/2023.acl-long.218).
- [19] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland, 2022. Association for Computational Linguistics. DOI: [10.18653/v1/2022.acl-long.395](https://doi.org/10.18653/v1/2022.acl-long.395).
- [20] Zuzanna Osika, Jazmin Zatarain Salazar, Diederik M. Roijers, Frans A. Oliehoek, and Pradeep K. Murukannaiah. What lies beyond the pareto front? a survey on decision-support methods for multi-objective optimization. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6741–6749. International Joint Conferences on Artificial Intelligence, 2023. DOI: [10.24963/ijcai.2023/755](https://doi.org/10.24963/ijcai.2023/755).
- [21] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 13484–13508, Toronto, Canada, 2023. Association for Computational Linguistics. DOI: [10.18653/v1/2023.acl-long.754](https://doi.org/10.18653/v1/2023.acl-long.754).
- [22] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.
- [23] Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*, 2023.
- [24] Chen Gong, Peilin Wu, and Jinho D. Choi. Aligning speakers: Evaluating and visualizing text-based speaker diarization using efficient multiple sequence alignment. In *Proceedings of the 35th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 778–783, 2023. DOI: [10.1109/ICTAI59109.2023.00119](https://doi.org/10.1109/ICTAI59109.2023.00119).
- [25] Daniel S. Park, Gustavo M. Lyra, Aaron M. Ellison, Rafael K. B. Maruyama, Daniel dos Reis Torquato, Rafael C. Asprino, Benjamin I. Cook, and Charles C. Davis. Herbarium records provide reliable phenology estimates in the understudied tropics. *Journal of Ecology*, 111(2):327–337, 2023. DOI: [10.1111/1365-2745.14047](https://doi.org/10.1111/1365-2745.14047).
- [26] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023. DOI: [10.48550/arXiv.2306.03341](https://doi.org/10.48550/arXiv.2306.03341).
- [27] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023. DOI: [10.48550/arXiv.2303.03846](https://doi.org/10.48550/arXiv.2303.03846).
- [28] Wei Dai, Jionghao Lin, Hua Jin, Tongguang Li, Yi Shan Tsai, Dragan Gasevic, and Guanliang Chen. Can large language models provide feedback to students? a case study on chatgpt. In *Proceedings of the 2023 IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 323–325. IEEE, 2023. DOI: [10.1109/ICALT58122.2023.00100](https://doi.org/10.1109/ICALT58122.2023.00100).
- [29] Wenyu Huang, Mirella Lapata, Pavlos Vougiouklis, Nikos Papasantopoulos, and Jeff Pan. Retrieval augmented generation with rich answer encoding. In *Proceedings of the 13th International Joint Conference on Natural Language Processing*

- and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1012–1025, Nusa Dua, Bali, 2023. Association for Computational Linguistics. DOI: 10.18653/v1/2023.ijcnlp-main.65.
- [30] Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. Gpt detectors are biased against non-native english writers. *arXiv preprint arXiv:2304.02819*, 2023. DOI: 10.48550/arXiv.2304.02819.
- [31] University of Siegen. Ateles documentation. <https://apes-suite.org/ateles/page/>.
- [32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. DOI: 10.48550/arXiv.1706.03762.
- [33] Paul Richards and Jim Barker. *Automotive Fuels Reference Book, Fourth Edition*. SAE International, 2023. ISBN: 9781468605792. DOI: 10.4271/9781468605792.
- [34] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Lu Chen, Jinshu Lin, and Dongfang Lou. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*, 2023. DOI: 10.48550/arXiv.2307.07306.
- [35] Mansi Agnihotri and Anuradha Chug. A systematic literature survey of software metrics, code smells and refactoring techniques. *Journal of Information Processing Systems*, 16(4):915–934, 2020. DOI: 10.3745/JIPS.04.0184.
- [36] Volodymyr Zhukov. What is retrieval-augmented generation (rag) and how can it reduce hallucinations in genai applications?
- [37] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.
- [38] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.
- [39] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, et al. Scaling language models: Methods, analysis insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.
- [40] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.
- [41] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- [42] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919. Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2020.acl-main.173.
- [43] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.
- [44] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094. Association for Computational Linguistics, 2021. DOI: 10.18653/v1/2021.naacl-main.168.
- [45] Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisen-schlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273, 2022. DOI: 10.1162/tacl.0.0459.
- [46] Hongli Zhou Yingqi Qu Jing Liu Muyun Yang Bing Xu Tiejun Zhao Hui Huang, Xingyuan Bu. An empirical study of llm-as-a-judge for llm evaluation: Fine-tuned judge model is not a general substitute for gpt-4. 2024.
- [47] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics. DOI: 10.3115/1073083.1073135.
- [48] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.
- [49] Huaixiu Steven Zheng, Xinyun Chen, Yewen Wang, Aashi Jain, Yutong He, Yujia Xie, Yujia Qin, Yujie Lu, Denny Zhou, and Charles Sutton. Take a step back: Evoking reasoning via abstraction in large language models. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- [50] Hung-Ju Chiang, Chia-Ying Chien, Chia-Yu Chang, Yi-Chun Huang, Chia-Yen Hsu, Chia-Yen Lin, Chia-Ying Tsai, Chia-Ling Hsieh, Chia-Yu Lin, Chia-Hsiu Chen, Chia-Yu Lee, Chia-Hui Wu, Chia-Hsiang Chen, Chia-Yu Wang, Chia-Hui Lin, Chia-Yu Huang, Chia-Hsiu Wang, Chia-Yu Hsieh, Chia-Hsiu Lin, and Chia-Yu Chen. Male germ cell-associated kinase (mak) is required for axoneme formation during ciliogenesis in zebrafish photoreceptors. *Journal of Cell Science*, 137(5):jcs261234, 2024. DOI: 10.1242/jcs.261234.
- [51] Peter Belcak, Greg Heinrich, Pavlo Molchanov, et al. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025.

- [52] NVIDIA. How small language models are key to scalable agentic ai. NVIDIA Technical Blog, 2025. <https://developer.nvidia.com/blog/how-small-language-models-are-key-to-scalable-agentic-ai/>.