Evaluating the Accuracy of Large Language Models for Geometry Generation in Physics-Based Simulations

Ossama Shafiq, Alessio Alexiadis, Amin Rahmat, Bahman Ghiassi

Evaluating the Accuracy of Large Language Models for Geometry Generation in Physics-Based Simulations

Ossama Shafiq*, Alessio Alexiadis*, Amin Rahmat* and Bahman Ghiassi*

***Correspondence**: Ossama Shafiq (oxs391@student.bham.ac.uk), Alessio Alexiadis (a.alexiadis@bham.ac.uk), Amin Rahmat (a.rahmat@bham.ac.uk) and Bahman Ghiassi (b.ghiassi@bham.ac.uk)

Abstract

This study evaluates the performance of nine Large Language Models (LLMs) in generating geometry files and simulation input files for physics-based simulations. We assessed the models' capabilities in creating both simple (square bar) and complex (wheel and axle) geometries, as well as their corresponding simulation input files. A comprehensive scoring system was developed to evaluate geometry fidelity, correctness of simulation parameters, completeness of files, and simulation output accuracy. Results show that advanced models like GPT-4o, GPT-4, and LLAMA-3-70B consistently demonstrated high proficiency across all tasks, while smaller or earlier model versions struggled. Performance varied significantly between simpler and more complex tasks, highlighting the challenges posed by increasing geometric and simulation complexity. The study reveals the potential of advanced LLMs in automating complex engineering tasks while also emphasising the need for careful model selection and potential human oversight. This research provides insights into the current capabilities and limitations of LLMs in engineering simulations and Computed Aided Design (CAD) operations, paving the way for their integration into future engineering workflows.

Keywords: *Large Language Models, Physics-based Simulations, Geometry, GMSH, GPT-4, LLAMA, AI in Engineering Design*

Highlights

- Evaluated nine LLMs for geometry and simulation file generation tasks.
- Advanced models outperformed smaller ones but still had notable limitations.
- Significant performance gaps emerged with complex geometry and simulations.
- Created a scoring system to compare LLM performance in engineering workflows.
- Results highlight the need for human oversight and further model improvements.

1. Introduction

Physics-based simulations are fundamental in scientific and engineering research, enabling the study of complex physical systems through computational models. These simulations rely on numerical methods such as the Finite Element Method (FEM), Boundary Element Method (BEM), or Finite Volume Method (FVM) (Gao et al., 2023), often implemented in both open-source and commercial software packages. Commercial tools (e.g., COMSOL (COMSOL AB, 2024), ANSYS (ANSYS, 2024)) add an additional layer of abstraction to these solvers, potentially reducing user control over the underlying algorithms but simplifying the workflow. (Dongarra et al., 2011)

(Gao et al., 2023)(COMSOL AB, 2024)

However, each additional "layer" (commercial interfaces or LLM-based automation) may introduce sources of error, emphasise black-box approaches, or require deeper expertise to verify correctness. Despite these concerns, physics-based simulations have transformed automotive design, aerospace engineering and materials science(Huzni et al., 2022). Yet, preparing accurate geometries and input files often demands specialised knowledge. Here, Large Language Models (LLMs) can automate significant portions of geometry and mesh preparation, theoretically lowering barriers for newcomers. Recently, the advent of LLMs such as OpenAI's GPT-3 and GPT-4 has marked a new milestone in human technological advancement. LLMs are a class of artificial intelligence models designed to understand and generate human-like text based on vast amounts of data. These models are built using a type of neural network architecture known as transformers, which were introduced by Vaswani et al. (2017). Transformers are highly effective at handling sequential data and have become the foundation for most state-of-the-art LLMs due to their ability to process and generate contextually relevant text over long passages.

LLMs are characterised by their large number of parameters—often in the billions—and their ability to generate coherent and contextually appropriate text across a wide range of topics. They are trained on extensive datasets that include text from books, articles, websites, and other textual resources, allowing them to capture the patterns and structures of human language. While transformers are the backbone of these models, they can also be employed in other machine learning applications beyond LLMs, such as in image processing and reinforcement learning. However, in the context of LLMs, transformers specifically enable these models to perform natural language processing tasks with unprecedented accuracy and fluency (Brown et al., 2020; Radford et al., 2019)

Despite their impressive capabilities in language generation and processing, LLMs inherently lack an understanding of the physical laws and principles that govern the real world. They can generate text that appears knowledgeable based on the patterns in their training data, but they do not possess an intrinsic conceptual understanding of physics or the ability to solve equations derived from physical laws. However, integrating LLMs with physics-based simulation tools can be valuable in automating certain tasks, such as pre-processing setup, assisting in creating simulation inputs, generating reports, or helping non-experts interact more easily with complex simulation software. By easing the user experience and providing contextual support, LLMs can help reduce the barriers to entry in simulation workflows, although they cannot replace the deep understanding of physical principles required for accurate simulations.

Integrating LLMs with physics-based simulation tools presents a groundbreaking opportunity to revolutionise the simulation workflow by addressing key challenges beyond workflow efficiency. By leveraging the natural language processing capabilities of LLMs, users can streamline the entire simulation setup process. For instance, a user can describe the requirements of their simulation in plain text, specifying physical phenomena, boundary conditions, material properties, and desired outcomes. The LLM can then interpret this as natural language input, automatically generating the necessary input files for multiphysics simulation software. This integration not only simplifies complex tasks but also tackles specific issues such as reducing human error in configuration, improving accuracy in defining simulation parameters, and accelerating overall setup time. By ensuring that input data is correctly interpreted and applied, this approach enhances the reliability of simulations. This integration would also make advanced computational simulations more accessible to users who may lack extensive technical expertise in simulation software.

Moreover, such an approach can significantly reduce the time and effort required to set up complex simulations, allowing researchers and engineers to focus more on analysis and innovation rather than on the intricacies of simulation software. In practice, this can democratise access to high-fidelity simulations, enabling a broader range of industries and research fields to harness the power of multiphysics simulations. The potential for error reduction is also significant, as LLMs can help standardise the input generation process, reducing the likelihood of user-induced errors in simulation setup. The integration of LLMs with simulation tools can also pave the way for more adaptive and intelligent simulation systems. For example, the LLM can be programmed to learn from past simulations, continuously improving its understanding of how to best set up new simulations based on previous outcomes and user feedback. This can lead to more accurate and efficient simulations over time, further enhancing the capabilities of simulation-based research and development.

This concept aligns with recent advancements in AI-driven simulation tools, where machine learning models are increasingly being used to enhance simulation accuracy and efficiency (Daw et al., 2017; Raissi et al., 2019). The use of LLMs for input file generation represents an evolution of this trend, where the goal is to simplify the interface between the user and the simulation software, effectively creating a more intuitive and user-friendly simulation environment.

Alexiadis and Ghiassi, (2024) assessed the capabilities and challenges of using machine learning models, specifically LLMs, in generating input files for multiphysics simulations. The findings revealed that LLMs, when properly prompted, can efficiently produce these input files, and that the meshing process can be partially automated. However, the study identified the creation of the initial geometry file as the most critical and challenging step in the process. The previous integrated approach was highlighted in the results of this study, indicating a potential trend toward enhancing physics-based simulation tools with specialised LLM-powered chatbots. These advanced chatbots would facilitate the setup process for users, making it more user-friendly and straightforward.

It is important to make clear the difference between the objective of this study and that of Physics-Informed Machine Learning (PIML). In fact, while PIML explicitly integrates physical laws into the learning process to improve model accuracy and reliability (Alexiadis, 2023, 2024), our study focuses on assessing the capabilities of LLMs, which do not inherently understand or apply these principles, in automating the setup of physics-based simulations. While Alexiadis and Ghiassi (2024) used a single LLM in their study, here we compare multiple LLMs to evaluate their ability to produce files for mechanical simulations. This is not a black-and-white analysis, where the answer is either entirely

correct or completely wrong. LLMs often fail to generate the correct files but can still perform most of the heavy lifting for the user.

Nonetheless, there is a paradox: LLMs may empower less-experienced users but also risk fostering uncritical acceptance of simulation outputs. An inexperienced user might assume a solver is correct without understanding potential pitfalls. Therefore, our study aims to investigate how reliably LLMs can generate these input files while also acknowledging the risks if domain expertise is lacking. Specifically, we address the following research questions: How accurately can LLMs generate geometry (.geo) and simulation (.sif) files for physics-based simulations? What limitations arise when dealing with more complex geometries and assemblies? How can we mitigate the risk that LLM-generated simulations might lead to erroneous results in the hands of non-experts?

2. Previous Work

The exploration of LLMs in the realm of physics-based simulations is a relatively nascent field, yet several studies have laid the groundwork for understanding their potential and limitations. Initial research demonstrated that LLMs, such as GPT-3, can be employed for generating code and documentation for various scientific tasks, including simple physics simulations (Chen et al., 2021). However, these early models often faced significant challenges in handling complex physical systems due to their inherent limitations in understanding physical laws and ensuring numerical accuracy (Marcus, 2020).

Current studies highlight the potential of LLMs to enhance Model-Based Systems Engineering (MBSE) methodologies, particularly through the integration of simulation frameworks that can support complex system analyses and physical modelling (Xie et al., 2022). However, the existing literature indicates that the development of comprehensive modelling and simulation infrastructures remains insufficient, limiting the full application of MBSE in practical scenarios.

In the context of computational fluid dynamics (CFD) and other data-driven simulations, LLMs are being explored for their ability to streamline the computational processes traditionally reliant on solving partial differential equations (PDEs) (Iserte et al., 2023). The shift towards data-driven methods, powered by artificial intelligence (AI), aims to alleviate the computational burden associated with these complex simulations, thereby enhancing efficiency and accessibility (Vadyala et al., 2022). Nonetheless, the literature reveals a gap in the systematic application of LLMs to automate and optimise the parameterisation of these simulations, which can further enhance their applicability in real-world scenarios.

Digital Twins (DTs) are another area where LLMs are gaining traction, particularly in their capacity to create virtual replicas of physical systems that can run simulations and support decision-making (Drissi Elbouzidi et al., 2023). The literature indicates that while DTs have been widely adopted across various industries, the integration of LLMs into DT frameworks remains underexplored. This presents a significant opportunity for future research to investigate how LLMs can enhance the predictive capabilities of DTs, particularly in dynamic environments where real-time data integration is crucial.

Moreover, the application of AI techniques, including LLMs, in magnetohydrodynamics and other multi-physics simulations has shown promise in accelerating scientific computations (Rosofsky and

Huerta, 2023). Neural operators, a class of deep learning models, have been employed to model complex simulations with reduced computational costs.

MyCrunchGPT introduces a transformative framework that integrates ChatGPT with scientific machine learning (SciML) processes, enhancing usability and accessibility in computational science and engineering (Kumar et al., 2023). This framework automates critical stages such as data preprocessing, simulation parameter configuration, and result interpretation through intuitive natural language interactions. While the potential of MyCrunchGPT to simplify complex simulation tasks is successfully demonstrated in Kumar et al., 2023, it subtly reveals significant gaps. There is an unclear extent to which this model generalises across diverse scientific fields, potentially limiting its universal applicability without significant adaptations. Additionally, the study does not deeply explore the scalability of the system, leaving unanswered questions about its performance under the computational demands of larger-scale simulations. Furthermore, while the integration of ChatGPT enhances user accessibility, the depth of this integration and its efficiency in real-world, complex scenario applications remain inadequately addressed. These gaps suggest areas for further research, particularly in testing the framework's robustness across varied scientific domains and scaling operations.

FLUID-GPT presents a novel hybrid machine learning model that integrates a Generative Pre-trained Transformer (GPT-2) with a convolutional neural network (CNN) to efficiently and accurately predict particle trajectories and surface erosion in industrial systems (Yang et al., 2023). This model is designed to overcome the limitations of traditional computational fluid dynamics (CFD) by reducing computational resources and training times while maintaining high prediction accuracy. However, the research exposes several gaps in its application. The paper focuses primarily on the model's efficiency and accuracy but less on the robustness of the predictions across varying operational conditions or different types of industrial applications. It also does not address the long-term sustainability of using such advanced AI models in terms of continuous learning and adaptation to new data or environmental changes, which are critical for practical deployment in dynamic industrial settings. The adaptability of FLUID-GPT to other physics-based simulations beyond particle trajectories and erosion is not explored, indicating a need for further validation of the model's utility across a broader range of physical phenomena.

The integration of GPT-4 with the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) to facilitate the creation and understanding of input files for molecular dynamics simulations was shown in (Verduzco et al., 2023). The research demonstrates GPT-4's capability to generate usable LAMMPS input files from verbal descriptions and to produce detailed descriptions suitable for scientific publications from these files. This functionality is shown to significantly decrease the amount of routine work researchers have to perform, expedite the training process for new users, and improve the reproducibility of computational experiments. Despite these advancements, there are significant gaps where GPT-4's application can be enhanced (Verduzco et al., 2023). First, while GPT-4 successfully handles simpler tasks and generates functional input files, its performance on complex multi-step simulations is less reliable, often requiring substantial modifications by the user. This limitation points to a need for improved understanding of complex simulation setups and better handling of intricate computational instructions by the model. Furthermore, while GPT-4 aids in the reproducibility of experiments by providing detailed methodological descriptions, the accuracy and completeness of these descriptions vary, suggesting room for enhancement in how the model

interprets and translates computational tasks into human-readable formats. These gaps underscore the potential areas for future research and development to fully harness AI capabilities in automating and improving scientific computing workflows.

3. Methodology

In this study, we explored the limitations and capabilities of LLMs specifically for generating critical input files necessary for conducting physics-based simulations. The focus on file generation is crucial because the accuracy and completeness of these files directly impact the success of the simulations. The files we generated include geometry files (.geo) and simulation input files (.sif), which are integral to the setup and execution of simulations using Gmsh and ELMER, respectively. We chose these well-known open-source tools, but we expect the results to be comparable with other simulation software.

We focus on discrete geometry representations via .geo files that Gmsh can interpret, rather than parametric/NURBS-based CAD models. This choice reflects the standard practice in many computational workflows, where a meshable geometry file is the immediate requirement for finite element or finite volume solvers. We selected two specific geometries for evaluation: a simple square bar and a more complex wheel and axle system. These geometries were chosen to represent a spectrum of complexity, from straightforward shapes to multi-component assemblies, allowing us to assess the LLMs' performance across different levels of difficulty.

To test feasibility, we performed simple linear-elastic static analyses. For the square bar, the boundary conditions included a fixed constraint at one end and an applied point load at the opposite face. Similarly, for the wheel-and-axle geometry, the axle ends were constrained while a small torque was applied to the wheels. These conditions provide a basic check that each simulation runs correctly and produces physically plausible stress distributions.

The choice to focus on file generation stems from the observation that preparing these input files often requires a significant amount of manual effort, technical expertise, and precision. Errors in file generation can lead to incorrect simulations, making this an ideal area to evaluate the utility of LLMs. By automating this process, LLMs can potentially reduce the time and expertise required to set up complex simulations, thereby making advanced physics-based simulations more accessible.

The methodology comprised several key steps:

Selection of LLMs: We selected a diverse set of LLMs to evaluate their performance in generating the necessary files for physics-based simulations. The models included MIXTRAL 8X7B, MIXTRAL 8X22B, LLAMA-2-70B, LLAMA-3-8B, LLAMA-3-70B, GPT-3.5 Turbo, GPT-4, GPT-4o, and PHI-3-Mini. This selection provided a broad spectrum of models with varying capabilities, ensuring a comprehensive evaluation of their strengths and weaknesses.

Creation of Geometry Files: The first step in the file generation process involved creating geometry files (.geo) that describe the geometrical properties of the models. These files were generated by

providing structured prompts to the selected LLMs. We focused on evaluating how well the LLMs can translate the geometric descriptions into accurate .geo files.

Mesh Generation: Once the .geo files were created, we used Gmsh, an open-source 3D finite element mesh generator (Geuzaine and Remacle, 2009), to create the corresponding 3D meshes. The meshing process was conducted using the default settings in Gmsh to maintain consistency across all geometries. The resulting mesh files (.msh) were then converted into ELMER-compatible mesh files using the ElmerGrid tool.

Generation of Simulation Files: The next step involved generating simulation input files (.sif) for ELMER, an open-source multiphysics simulation software. These files contained the necessary simulation parameters, such as material properties, boundary conditions, and solver settings. The LLMs were prompted to produce these files based on the geometries and meshes previously generated.

Simulation Execution and Assessment: Finally, we conducted simulations using ELMER based on the .sif files generated by the LLMs. The performance of each LLM was assessed by comparing the accuracy and completeness of the generated files, as well as the fidelity of the simulation results. The assessment focused on key aspects such as geometry fidelity, correctness of simulation parameters, completeness of files, and the accuracy of the simulation outputs.

By following this structured methodology, we aimed to rigorously evaluate the capabilities and limitations of different LLMs in generating files for physics-based simulations. The use of open-source tools like Gmsh and ELMER ensured that our findings would be broadly applicable and accessible to the scientific community.

Workflow

This section outlines the systematic workflow followed to generate and evaluate geometry and simulation files using LLMs for physics-based simulations. The workflow, as illustrated in Figure 1, consists of several key stages, each supported by specific code implementations designed to automate and streamline the process.

We purposely propose a relatively simple code in this workflow. If the goal of this study is to leverage LLMs to enable non-expert users to set up multiphysics simulations, the same principle of accessibility must apply to the code used in this publication. Therefore, we have taken care to explain all aspects of the code in detail and, whenever possible, minimise its complexity to ensure that users with varying levels of expertise can easily understand and implement it.

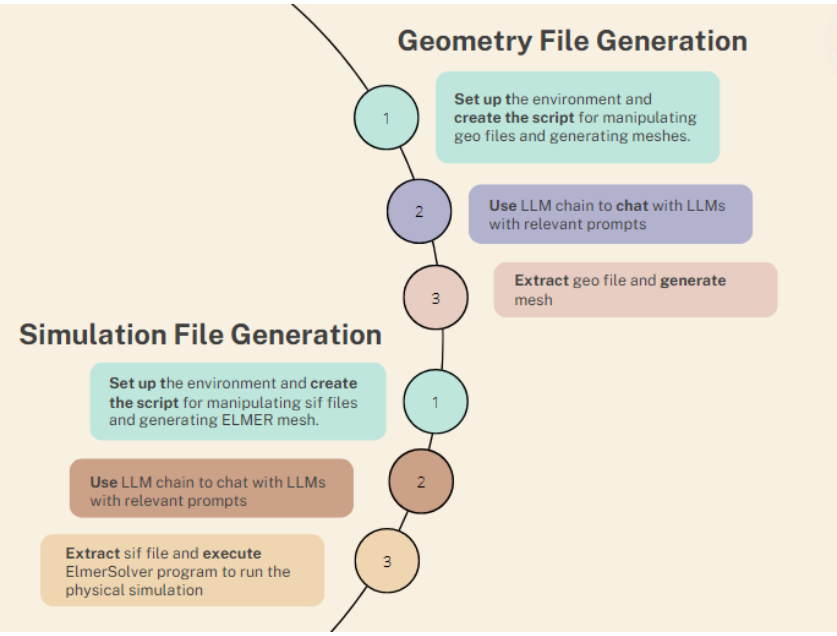

Figure 1. Protocol
Geometry and simulation file generation overall protocol.

The process begins with setting up the environment and creating scripts that allow for the manipulation of .geo and .sif files, which are necessary for generating geometries and running simulations in Gmsh and ELMER, respectively. These scripts were written in Python, using libraries such as LangChain (Reynolds, 2024) for interacting with LLMs and Gmsh's Python API for meshing tasks.

In the LangChain framework, the LLMChain object is configured to connect a LLM with a prompt, incorporating a memory mechanism and specific output controls. The Conversation Buffer Window Memory is set to retain the last eight interactions, allowing the model to maintain context across multiple turns in a conversation. The language model is further configured with parameters such as maximum number of tokens was set to 4096, which limits the response length, and temperature was set to 0.00, ensuring that the output is deterministic. This setup enables the generation of contextually aware, controlled responses while efficiently managing conversational history *(the code is available on GitHub).*

In the **Geometry File Generation** phase, we first set up the environment and created scripts to manipulate .geo files and generate meshes. The LLMs were then prompted using a series of pre-designed prompt templates that guided the models to produce the required geometry files. The generated .geo files were subsequently extracted, and a mesh was generated using Gmsh. The code for this part of the workflow involved initialising the LLM with appropriate API calls, passing the prompts, and processing the text output to save the .geo file, followed by a call to the Gmsh API to generate the mesh *(the code is available on GitHub).*

The **Simulation File Generation** phase followed a similar process. The environment was set up, and scripts were created for manipulating .sif files and generating the necessary ELMER mesh. The LLMs were again engaged through a series of prompts tailored to extract the simulation parameters, which

were then formatted into a .sif file. The generated .sif file was extracted and fed into the ElmerSolver program to run the physical simulation. The code used in this phase involved similar steps as the geometry file generation but focused on extracting and structuring simulation parameters *(the code is available on GitHub).*

Finally, the **extraction and execution** steps were performed. The scripts ensured that the outputs from the LLMs were correctly formatted, and that the simulation can be executed within ELMER. We used specific functions to extract the contents of the .geo and .sif files from the LLM output, convert them into usable formats, and execute the simulations. This entire process was automated using Python scripts, which handled everything from the interaction with the LLMs to the execution of the simulations.

Overall, this workflow not only illustrates the logical sequence of tasks involved but also shows how each step is supported by code implementations designed to automate the interaction with LLMs, handle file generation, and execute simulations. The detailed code snippets and functions used for each step are crucial for understanding the practical execution of this workflow *(the code is available on GitHub).*

Development of Prompt Templates

To facilitate effective interaction with the LLMs, we developed prompt templates designed to guide the models in generating the required outputs. These templates were structured to ensure clarity and consistency across different models. Specifically, the templates provided the necessary structure and instructions for the LLMs to generate .geo files, which describe the geometrical properties of the models, and .sif files, which are used as input for ELMER physics-based simulations. This structured approach ensured that the generated files met the requirements for accurate geometry representation and simulation *(the code is available on GitHub).*

Implementation of Functions

To process the outputs from the LLMs and facilitate the generation and validation of geometry and simulation files, we implemented several key functions in Python, leveraging specific libraries tailored to interact with the LLMs and simulation tools. These functions are crucial in automating the workflow and ensuring that the outputs are correctly formatted and ready for simulation. The functions described below are also integrated into the workflow presented in Figure 1, which illustrates how these components fit into the overall process.

1. **extract_and_save_geo_file**: This function extracts the content of a .geo file from the response text generated by the LLM and saves it to a file. The LLM, accessed via the LangChain library, generates text that describes the geometry. This text is then parsed by the function to identify relevant sections and save them in a structured .geo file format compatible with Gmsh (Figure 3a).
2. **generate_mesh**: Utilising the Gmsh Python API, this function creates a 3D mesh from the .geo file. The function initialises Gmsh, sets mesh size options (using default settings for consistency), generates the mesh, and writes it to an output .msh file. This mesh file is an

intermediate step between geometry creation and simulation setup, essential for discretizing the model into elements that ELMER can process (Figure 3b).

3. **generate_ELMER_mesh**: This function uses the ElmerGrid tool, a command-line utility that comes with the ELMER software, to convert the .msh file generated by Gmsh into a format that ELMER can use. This conversion is necessary because ELMER requires a specific mesh format to perform simulations, and this function automates that conversion process (Figure 4a).
4. **extract_and_save_sif_file**: Similar to the extract_and_save_geo_file function, this function extracts the content of a .sif file from the LLM's response and saves it to a file. The .sif file contains simulation parameters such as material properties, boundary conditions, and solver settings. This function ensures that the text generated by the LLM is structured correctly for ELMER to execute the simulation (Figure 4b).

These functions were developed in Python, utilising libraries such as LangChain for LLM interaction, Gmsh's Python API for mesh generation, and system calls to ElmerGrid for mesh conversion. Interaction with the LLM was managed through API calls, where prompt templates were used to generate the desired output text for the .geo and .sif files.

Generation of Geometry Files

The process of generating geometry files involved several structured steps, following a method similar to that described by Alexiadis and Ghiassi (2024) in their work on integrating LLMs with simulation tools. First, we imported the necessary libraries and dependencies, including LangChain, and initialised the LLM using the appropriate API keys and model specifications *(the code is available on GitHub).*

To interact with the LLMs effectively, we utilised a combination of system and user prompts, much like the approach employed by Alexiadis and Ghiassi (2024). The system prompts provided the LLM with context and instructions on the format and structure of the output, while the user prompts specified the details of the geometry to be generated. This combination ensured that the LLMs received clear and consistent instructions across different sessions, facilitating the accurate generation of .geo files *(the code is available on GitHub).*

After the prompts were processed by the LLMs, the extract_and_save_geo_file function was employed to parse the LLM's output and save it as a geometry file. This function ensured that the text generated by the LLM adhered to the .geo file format required by Gmsh, and was ready for subsequent simulation steps.

Generation of Simulation Files

The creation of simulation files is a critical step in setting up and executing physics-based simulations. In this study, the term "simulation files" specifically refers to .sif (Solver Input File) files used by the ELMER software. These files contain all the necessary information to define the physical model, including material properties, boundary conditions, solver settings, and other parameters required to run the simulation.

To generate these simulation files, we began by providing carefully crafted prompts to the selected LLMs. These prompts were designed to elicit the correct structure and content needed for a .sif file. The LLMs processed the prompts and generated text that outlined the simulation parameters based on the geometry previously defined *(the code is available on GitHub).*

Once the LLM generated the appropriate response, the text was extracted and formatted into a .sif file. This step was crucial because it ensured that the LLM's output was correctly structured according to the specific requirements of ELMER. The .sif file included key details such as the type of physical simulation to be conducted (e.g., structural analysis, thermal analysis), the material properties to be applied, the boundary conditions, and the numerical methods to be used.

In addition to generating the .sif file, it was necessary to create a mesh compatible with ELMER from the geometry defined earlier. This was achieved by converting the .msh file, which was generated from the .geo file during the geometry phase, into an ELMER-compatible mesh using the ElmerGrid tool. The mesh, along with the .sif file, forms the complete set of inputs required to run a simulation in ELMER.

Simple Geometry: Square Bar

The simple geometry was a square bar, characterised by a length of 10 cm and a square cross-section with side lengths of 1 cm. This geometry was chosen for its simplicity, which allows for straightforward verification of the generated files and serves as a fundamental model to establish the baseline performance of the LLMs in producing accurate geometric representations and simulation files.

Complex Geometry: Wheel and Axle System

The complex geometry was a wheel and axle system. This system comprised two wheels, each with a radius of 5 cm and a width of 2 cm, connected by an axle with a radius of 1 cm and a length of 20 cm. This model showcases the complexity involved in creating assemblies with multiple interacting parts.

Assessment of Performance: Criteria and Scoring System

To ensure a comprehensive evaluation of the LLMs' performance, we implemented a detailed assessment methodology focusing on the following criteria and scoring system (Figure 2).

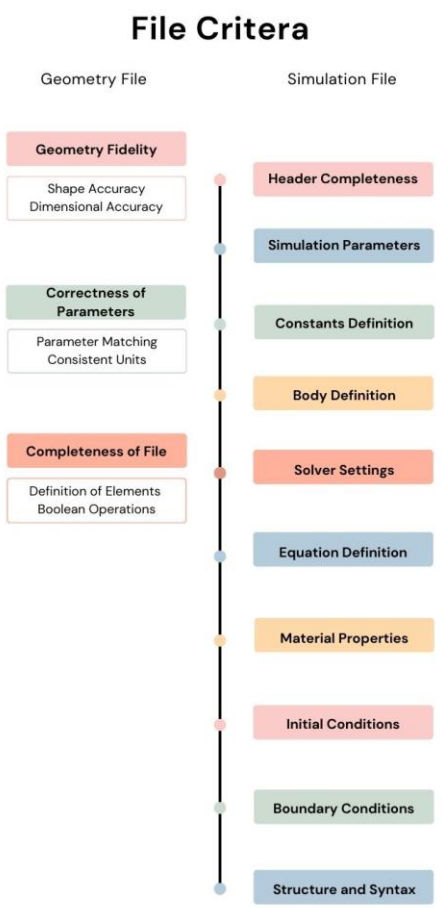

Figure 2. Criteria
Criteria to score geometry and simulation file generation, and benchmark across LLMs.

It is important to note that while this system provides a structured approach to evaluation, it is inherently somewhat arbitrary due to the subjective nature of assessing variations and deviations. We have taken care to define our criteria as clearly as possible but acknowledge that different evaluators might apply slightly different judgments.

1. **Geometry Fidelity**: The accuracy of the geometric definitions was compared against a baseline model. This involved verifying that the points, lines, surfaces, and volumes defined by the LLMs matched the expected dimensions and structures of the simple and complex geometries. Scoring of 0-30 was based on shape accuracy and dimensional accuracy.
    - **Shape Accuracy**: A perfect match with the baseline shape received 15 points. Minor deviations, defined as slight alterations that do not affect the overall recognisability of the shape, received 10 points. For instance, a corner slightly out of place or a minor surface irregularity would be considered minor. Noticeable deviations that slightly altered the shape but still remained recognisable received 5 points, while major deviations or incorrect shapes received 0 points. The distinction between minor and noticeable deviations was made based on the extent to which the geometry's function or purpose would be impacted in a real-world application.

- **Dimensional Accuracy**: Perfect matching of all dimensions received 15 points. Minor dimensional errors, which were defined as errors within 1% of the expected value, received 10 points. Significant dimensional errors, defined as errors within 5% of the expected value, received 5 points. Major dimensional inaccuracies, defined as errors greater than 5%, received 0 points.

2. **Correctness of Simulation Parameters**: The parameters used in the geometry and simulation files, such as material properties, boundary conditions, and solver settings, were checked for correctness. This included ensuring that values such as Young's modulus, Poisson's ratio, and density were accurately defined.
   - **Parameter Matching**: Perfect matching of parameters with the baseline received 15 points. Minor discrepancies, such as a slight difference in the value of Young's modulus that would still allow the simulation to run correctly but with slightly altered results (e.g., a difference within 1% of the expected value), received 10 points. Significant errors, such as a difference in Young's modulus that would noticeably affect the simulation results (e.g., a difference within 5% of the expected value), received 5 points. Incorrect or missing parameters that would result in failed or invalid simulations received 0 points.
   - **Consistent Units**: Perfect consistency in the use of units received 15 points. Minor inconsistencies, where units were mostly correct but had small errors that can be easily corrected (e.g., using cm instead of mm in one instance), received 10 points. Significant inconsistencies, where the errors would require more careful adjustments (e.g., multiple unit discrepancies across different parameters), received 5 points. Major inconsistencies or the use of completely incorrect units received 0 points.

3. **Completeness of Files**: The completeness of the generated files was assessed to determine whether they included all necessary elements required for successful simulation. This involved verifying the presence of all essential sections in the .geo and .sif files, such as headers, constants, body properties, solvers, equations, materials, initial conditions, and boundary conditions.
   - **Definition of Elements**: Complete and correct definitions received 15 points. Minor missing elements, which were defined as non-critical elements that can be easily added without affecting the simulation's core functionality, received 10 points. Significant missing elements, which can affect the simulation outcome but would not prevent the simulation from running, received 5 points. Incorrect or missing major elements, which would prevent the simulation from running correctly, received 0 points.
   - **Boolean Operations**: Gmsh utilises Boolean operations to manipulate geometric entities such as surfaces, volumes, and solids, with these operations being defined in the corresponding .geo file. Correct Boolean operations received 15 points. Minor errors, which can be corrected with minimal impact on the overall geometry (e.g., small misalignments), received 10 points. Significant errors, where the operations were incorrect but still allowed the geometry to be somewhat usable, received 5 points. Incorrect or missing Boolean operations that resulted in unusable geometry received 0 points.

4. **Simulation Output**: The visual and numerical accuracy of the simulation results produced by the LLM-generated files were compared to the baseline. This included evaluating the displacement, stress distribution, and deformation patterns in the simulated models. Visual comparisons were made to ensure that the results matched the expected behaviour of the simple and complex geometries under specified conditions (Figure 3, 4).
    - Scoring for this criterion was based on how closely the simulated results aligned with expected outcomes. Perfect alignment with expected displacement, stress distribution, and deformation patterns received 100 points, while deviations were scored based on their impact on the simulation's validity and usefulness.
    -

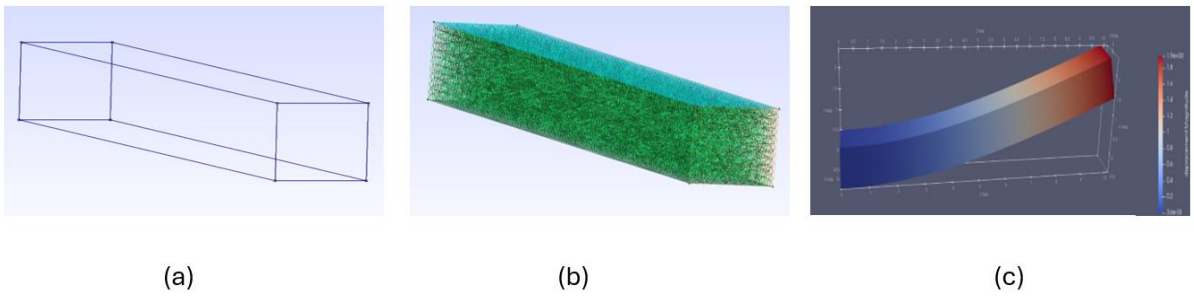

Figure 3. Square Bar Visualisation
The expected and correct visual representation of the square bar (a) geometry and (b) mesh prior to running any simulation, (c) and the square bar after the simulation run.

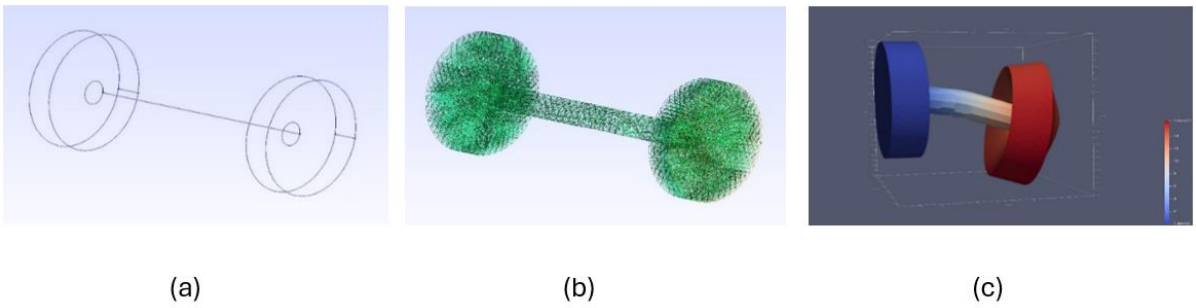

Figure 4. Wheel and Axle Visualisation
The expected and correct visual representation of the wheel and axle (a) geometry and (b) mesh prior to running any simulation, (c) and the wheel and axle after the simulation run.

4.Results and Discussion

4.1 Simple Geometry

In the generation of a square bar geometry, several LLMs demonstrated exceptional performance, closely matching the baseline model with high precision. GPT-4o, GPT-4, LLAMA-3-70B, MIXTRAL 8X7B,

and MIXTRAL 8X22B all achieved perfect or near-perfect scores (96-100/100), (Figure 5, 6), showcasing their ability to accurately define the geometry, apply correct parameters, and create complete files.

GPT-4o stood out by demonstrating advanced capabilities, including the use of Boolean operations, which wasn't seen in other models' outputs. This highlights its sophisticated approach to geometric modelling tasks.

Table 1. Scoring of the square bar geometry for each LLM. The criteria reflect a scoring of 0-30 for Geometry Fidelity and Correctness of Parameters, and 0-40 for Completeness of Files. The scoring indicates whether all, most, some or none of the elements of the output were correct.

Geometry Criteria (Square Bar)	***MIXTRAL 8X7B***	***MIXTRAL 8X22B***	***LLAMA 2 70B***	***LLAMA 3 8B***	***LLAMA 3 70B***	***GPT-3.5 Turbo***	***GPT-4***	***GPT-4o***	***PHI-3 Mini***
Geometry Fidelity	20	30	15	20	30	20	30	30	10
Shape Accuracy	10	15	5	10	15	10	15	15	5
Dimensional Accuracy	10	15	10	10	15	10	15	15	5
Correctness of Parameters	20	25	10	20	30	25	30	30	10
Parameter Matching	10	13	5	10	15	15	15	15	5
Consistent Units	10	12	5	10	15	10	15	15	5
Completeness of Files	20	35	15	25	40	15	40	40	10
Definition of Elements	10	18	10	15	20	10	20	20	5
Boolean Operations	10	17	5	10	20	5	20	20	5
Total Score /100	**60**	**90**	**40**	**65**	**100**	**60**	**100**	**100**	**30**

MIXTRAL 8X7B

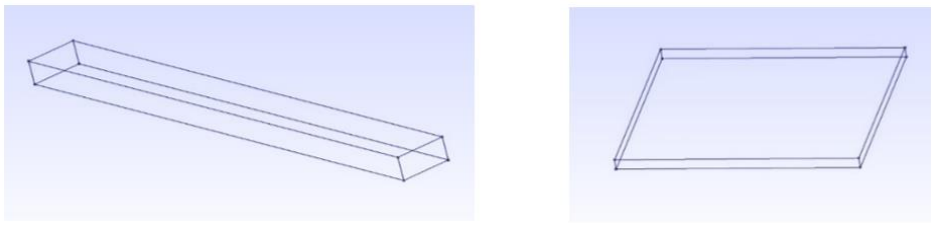

Figure 5. MIXTRAL 8X7B
The (a) original and (b) updated output for the square bar geometry from MIXTRAL 8X7B.

MIXTRAL 8X22B

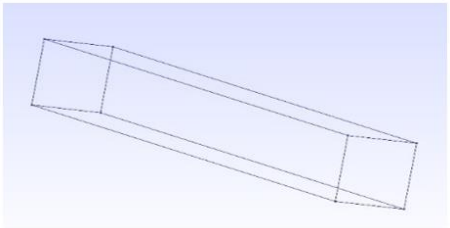

Figure 6. MIXTRAL 8X22B
The output for the square bar geometry from MIXTRAL 8X22B was consistent with one another, presenting an identical expected output in comparison to the original.

LLAMA-3-8B (45/100) and LLAMA-2-70B (40/100) showed significant limitations in their ability to generate accurate square bar geometries. These models struggled with proper alignment, geometrical definitions, and overall structure, resulting in outputs that would require substantial modifications for practical use.

PHI-3 Mini (35/100) demonstrated the most significant difficulties, producing output that fundamentally misunderstood the task, creating a geometry file that was largely incorrect and unusable for the intended purpose.

It's noteworthy that the larger, more advanced models generally outperformed their smaller counterparts, suggesting a correlation between model size/complexity and the ability to handle specialised engineering tasks like CAD geometry creation.

These results highlight the rapid advancements in LLM capabilities, with newer models showing remarkable proficiency in complex geometric modelling tasks. However, they also underscore the continued limitations of some models, emphasising the need for careful selection and potential human oversight when applying LLMs to engineering applications.

4.2 Complex Geometry

In the generation of the more complex wheel and axle geometry, we observed a wider range of performance across the evaluated LLMs, reflecting the increased difficulty of the task.

GPT-4o demonstrated exceptional performance, achieving the highest score (96/100). It stood out by not only accurately defining the geometry and dimensions but also by incorporating advanced features such as Boolean operations to unite the wheels and axle. This showcased GPT-4o's sophisticated approach to complex geometric modelling tasks.

GPT-4 and LLAMA-3-70B also performed excellently, both scoring 91/100. These models produced accurate geometries with precise alignment and dimensions that closely followed the design specifications. Their outputs would require minimal to no modifications for practical use in engineering applications.

Table 2. Scoring of the wheel and axle geometry for each LLM. The criteria reflect a scoring of 0-30 for Geometry Fidelity and Correctness of Parameters, and 0-40 for Completeness of Files. The scoring indicates whether all, most, some or none of the elements of the output were correct.

Geometry Criteria (Wheel & Axle)	***MIXTRAL 8X7B***	***MIXTRAL 8X22B***	***LLAMA 2 70B***	***LLAMA 3 8B***	***LLAMA 3 70B***	***GPT-3.5 Turbo***	***GPT-4***	***GPT-4o***	***PHI-3 Mini***
Geometry Fidelity	20	25	10	15	29	20	28	28	10
Shape Accuracy	10	13	5	10	15	12	15	14	5
Dimensional Accuracy	10	12	5	5	14	8	13	14	5
Correctness of Parameters	25	28	15	20	28	15	28	30	10
Parameter Matching	15	13	10	15	13	5	13	15	5
Consistent Units	10	15	5	5	15	10	15	15	5
Completeness of Files	20	25	15	10	35	25	35	38	15

Definition of Elements	10	15	10	5	20	15	20	19	10
Boolean Operations	10	10	5	5	15	10	15	19	5
Total Score /100	**65**	**78**	**40**	**45**	**92**	**60**	**91**	**96**	**35**

LLAMA-3-70B

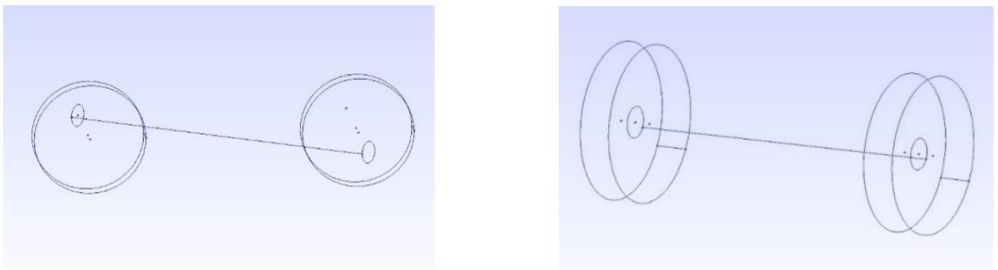

Figure 7. LLAMA-3-70B
The (a) original and (b) updated output for the wheel and axle geometry from LLAMA-3-70B

MIXTRAL 8X22B and MIXTRAL 8X7B showed good performance (both scoring 78/100), (Figure 8), accurately defining the basic geometry but with some minor issues in implementation. These models demonstrated a solid understanding of the task requirements but fell short in some aspects of the more complex geometry.

MIXTRAL 8X7B

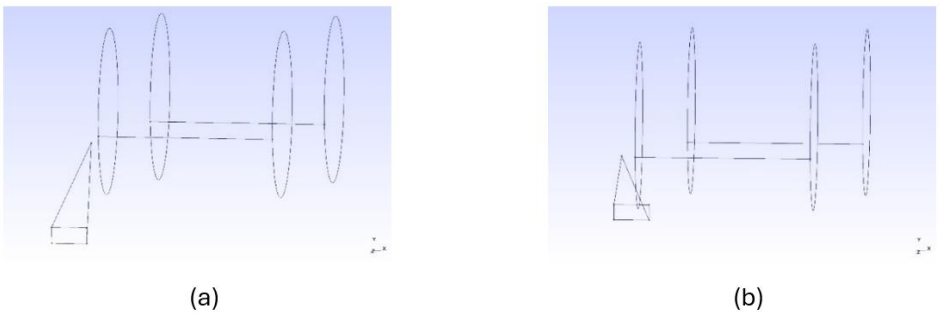

Figure 8. MIXTRAL 8X7B
The (a) original and (b) updated output for the wheel and axle geometry from MIXTRAL 8X7B.

GPT-3.5 Turbo (65/100) and LLAMA-3-8B (45/100) encountered more significant challenges. While they showed a basic understanding of the geometry, their outputs included errors in alignment, dimensional accuracy, or completeness that would require substantial corrections for practical use.

LLAMA-2-70B (40/100) and PHI-3 Mini (35/100) struggled the most with this complex geometry task. Their outputs showed fundamental misunderstandings or significant errors that resulted in unusable or highly inaccurate geometries.

These results highlight the increasing sophistication of newer LLM iterations in handling complex CAD-like tasks. The superior performance of models like GPT-4o in incorporating advanced geometric operations points to the potential for LLMs to automate increasingly complex design tasks.

However, the varied performance across models also underscores the challenges that remain in applying LLMs to specialised engineering tasks. It emphasises the importance of selecting appropriate models for specific tasks and the potential need for human oversight and verification, especially when dealing with complex geometries crucial for engineering applications, providing a clear illustration of each model's capabilities and deficiencies.

4.3 Simulation

The simulation tasks, involving the creation of Simulation Input Files (SIF) for both the square bar and the more complex wheel and axle system, revealed significant variations in the capabilities of the evaluated LLMs.

For the square bar simulation, GPT-4, GPT-4o, LLAMA-3-70B, MIXTRAL 8X7B, and MIXTRAL 8X22B all achieved perfect scores (20/20), demonstrating exceptional proficiency in creating accurate and complete SIF files. These models correctly defined all necessary components, including header information, simulation parameters, material properties, solver settings, and boundary conditions. GPT-3.5 Turbo performed well (18/20) but had a notable issue with reversed boundary condition targets, which can significantly affect simulation results. LLAMA-3-8B (17/20) showed good understanding but had some minor omissions and errors. LLAMA-2-70B (4/20) and PHI-3 Mini (6/20) struggled significantly, producing incomplete or largely incorrect outputs that would not be usable for simulations without extensive modifications.

Table 3. Scoring of the square bar simulation for each LLM. The criteria reflect a scoring of 0-2, where 0 suggests missing or incorrect, 1 partially correct, and 2 suggests complete and correct outputs.

Geometry Criteria (Square Bar)	***MIXTRAL 8X7B***	***MIXTRAL 8X22B***	***LLAMA 2 70B***	***LLAMA 3 8B***	***LLAMA 3 70B***	***GPT-3.5 Turbo***	***GPT-4***	***GPT-4o***	***PHI-3 Mini***
Header Completeness	2	2	1	2	2	2	2	2	0
Simulation Parameters	2	2	1	1	2	2	2	2	1
Constants Definition	2	2	1	1	2	2	2	2	1
Body Definition	2	2	0	2	2	2	2	2	1
Solver Settings	2	2	0	2	2	2	2	2	0
Equation Definition	2	2	0	2	2	2	2	2	0
Material Properties	2	2	0	1	2	1	2	2	1
Initial Conditions	2	2	0	2	2	2	2	2	1
Boundary Conditions	2	2	0	2	2	1	2	2	1
Total Score /20	**20**	**20**	**4**	**17**	**20**	**18**	**20**	**20**	**6**

For the more complex wheel and axle simulation, GPT-4o, GPT-4, and LLAMA-3-70B maintained their perfect scores (20/20), showcasing their ability to handle increased complexity with high accuracy. MIXTRAL 8X22B and MIXTRAL 8X7B performed very well (19/20), with only minor, non-critical additions to their outputs. GPT-3.5 Turbo (18/20) and LLAMA-3-8B (12/20) showed decreased performance with the increased complexity, highlighting the challenges posed by more intricate simulation setups. LLAMA-2-70B (3/20) and PHI-3 Mini (1/20) again struggled significantly with this task.

Table 4. Scoring of the wheel and axle simulation for each LLM. The criteria reflect a scoring of 0-2, where 0 suggests missing or incorrect, 1 partially correct, and 2 suggests complete and correct outputs.

Geometry Criteria (Wheel & Axle)	*MIXTRAL 8X7B*	*MIXTRAL 8X22B*	*LLAMA 2 70B*	*LLAMA 3 8B*	*LLAMA 3 70B*	*GPT-3.5 Turbo*	*GPT-4*	*GPT-4o*	*PHI-3 Mini*
Header Completeness	2	2	1	2	2	2	2	2	0
Simulation Parameters	2	2	1	1	2	2	2	2	0
Constants Definition	2	2	0	1	2	2	2	2	0
Body Definition	1	2	0	2	2	1	2	2	1
Solver Settings	2	2	0	1	2	2	2	2	0
Equation Definition	2	2	0	1	2	2	2	2	0
Material Properties	2	1	0	2	2	1	2	2	0
Initial Conditions	2	2	0	0	2	2	2	2	0
Boundary Conditions	2	2	0	1	2	1	2	2	0
Total Score /20	**20**	**20**	**3**	**12**	**20**	**20**	**20**	**20**	**1**

Overall, this assessment highlights the remarkable capabilities of advanced LLMs like GPT-4o, GPT-4, and LLAMA-3-70B in handling both simple and complex simulation tasks with high fidelity. These models demonstrated consistent performance across both geometries, suggesting they are approaching practical utility in engineering simulations. The MIXTRAL models showed strong performance, particularly in the square bar task, indicating their potential for handling engineering simulation setups. However, the performance drop in some models (like GPT-3.5 Turbo and LLAMA-3-8B) between the simple and complex tasks underscores the challenges posed by increasing complexity in simulation requirements. The significant struggles of LLAMA-2-70B and PHI-3 Mini across both tasks emphasize the rapid advancements in the field, with newer model iterations showing markedly improved capabilities.

4.4 Limitations and Oversight

LLMs show promise for automating geometry and file generation; however, these outputs must be validated by an engineer with domain knowledge. Even the best-performing models occasionally introduced small errors, e.g., reversed boundary conditions or omitted material parameters. In practice, these mistakes might go unnoticed if the user lacks expertise, compounding the risk of systematically erroneous simulations. Therefore, LLM-driven automation should serve as an assistant rather than a replacement for an experienced analyst.

5.Conclusions

In the study, we evaluated of nine LLMs in generating geometry files and simulation input files for static physics-based simulations. revealed significant variations in performance across different models and tasks. In the square bar geometry task, GPT-4o, GPT-4, LLAMA-3-70B, MIXTRAL 8X7B, and MIXTRAL 8X22B demonstrated exceptional capabilities, achieving near-perfect or perfect scores (96-100/100). GPT-4o stood out by incorporating advanced features like Boolean operations. However, LLAMA-3-8B, LLAMA-2-70B, and PHI-3 Mini struggled significantly, producing outputs that would require substantial modifications for practical use.

The more complex wheel and axle geometry task further differentiated the models' capabilities. GPT-4o again excelled (96/100), showcasing advanced geometric modelling skills. GPT-4 and LLAMA-3-70B also performed excellently (91/100), while other models showed varying degrees of accuracy and completeness in their outputs.

In the square bar simulation input file generation task, GPT-4, GPT-4o, LLAMA-3-70B, MIXTRAL 8X7B, and MIXTRAL 8X22B achieved perfect scores (20/20), demonstrating proficiency in creating accurate and complete SIF files. GPT-3.5 Turbo and LLAMA-3-8B showed good understanding but had minor issues, while LLAMA-2-70B and PHI-3 Mini struggled significantly. The wheel and axle simulation task saw GPT-4o, GPT-4, and LLAMA-3-70B maintaining perfect scores, showcasing their ability to handle increased complexity. MIXTRAL models performed very well, with only minor, non-critical additions. Other models showed decreased performance with the increased complexity.

This study highlights several key findings. Advanced LLMs (GPT-4o, GPT-4, LLAMA-3-70B) consistently demonstrated high proficiency across all tasks, suggesting their potential for practical application in engineering simulations and CAD operations. Newer model iterations generally outperformed older ones, indicating rapid advancements in the field. Performance varied significantly between simpler and more complex tasks, emphasizing the challenges posed by increasing geometric and simulation complexity. Some models (like MIXTRAL 8X7B and 8X22B) showed strong performance in simulation tasks despite initial geometry generation issues, highlighting their potential with proper input. Smaller or earlier model versions (LLAMA-2-70B, LLAMA-3-8B, PHI-3 Mini) struggled consistently, underlining the importance of model size and training in handling specialised tasks.

These findings underscore the potential of advanced LLMs in automating complex engineering tasks while also highlighting the need for careful model selection, potential human oversight, and continued development. Future research should focus on enhancing LLMs' capabilities in handling complex geometries, improving consistency across tasks, and integrating advanced CAD operations into their outputs. The advanced models (GPT-4o, GPT-4, LLAMA-3-70B) excelled, while smaller/earlier versions struggled. Crucially, performance gaps became more pronounced for complex geometries, highlighting the reliance on accurate definitions and robust meshing.

We also acknowledge that enabling less-experienced users to conduct advanced simulations via LLMs comes with the risk of uncritical acceptance of faulty outputs. Hence, we recommend that LLMs be positioned as assistive tools rather than end-to-end solutions. Future work may focus on deeper integration of domain-specific constraints and robust error-checking within LLMs, ensuring a safer transition to widespread adoption.

This study introduces a scoring system and a basic code framework for evaluating and comparing the capabilities of LLMs, which we applied to a selected number of models. As new LLMs are developed at a rapid pace, both our scoring system and code can be readily adapted to assess and extend this analysis to future models.

6.Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to refine the language and enhance clarity. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

7.References

1. Alexiadis, A. (2023) A minimalistic approach to physics-informed machine learning using neighbour lists as physics-optimized convolutions for inverse problems involving particle systems. *Journal of Computational Physics*, 473: 111750. doi:10.1016/j.jcp.2022.111750.
2. Alexiadis, A. (2024) Designing a set of criteria for evaluating artificial neural networks trained with physics-based data to replicate molecular dynamics and other particle method trajectories. *Frontiers in Nanotechnology*, 6. doi:10.3389/fnano.2024.1373316.
3. Alexiadis, A. and Ghiassi, B. (2024) From text to tech: Shaping the future of physics-based simulations with AI-driven generative models. *Results in Engineering*, 21. doi:10.1016/j.rineng.2023.101721.
4. ANSYS, Inc. (2024) *ANSYS®* .
5. Brown, T.B., Mann, B., Ryder, N., et al. (2020) *Language Models are Few-Shot Learners*.
6. Chen, M., Tworek, J., Jun, H., et al. (2021) *Evaluating Large Language Models Trained on Code*.
7. COMSOL AB (2024) *COMSOL Multiphysics®* .
8. Daw, A., Karpatne, A., Watkins, W., et al. (2017) *Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling*.
9. Drissi Elbouzidi, A., Ait El Cadi, A., Pellerin, R., et al. (2023) The Role of AI in Warehouse Digital Twins: Literature Review. *Applied Sciences*, 13 (11): 6746. doi:10.3390/app13116746.
10. Gao, X.-W., Zhu, Y.-M. and Pan, T. (2023) Finite line method for solving high-order partial differential equations in science and engineering. *Partial Differential Equations in Applied Mathematics*, 7: 100477. doi:10.1016/j.padiff.2022.100477.
11. Geuzaine, C. and Remacle, J. (2009) Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79 (11): 1309–1331. doi:10.1002/nme.2579.
12. Huzni, S., B.M. Ibrahim, I., Fonna, S., et al. (2022) Physics-based surrogate model for reinforced concrete corrosion simulation. *Results in Engineering*, 16: 100659. doi:10.1016/j.rineng.2022.100659.
13. Iserte, S., González-Barberá, A., Barreda, P., et al. (2023) A study on the performance of distributed training of data-driven CFD simulations. *The International Journal of High Performance Computing Applications*, 37 (5): 503–515. doi:10.1177/10943420231160557.
14. Kumar, V., Gleyzer, L., Kahana, A., et al. (2023) *MyCrunchGPT: A chatGPT assisted framework for scientific machine learning*. Available at: http://arxiv.org/abs/2306.15551.
15. Marcus, G. (2020) *The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence*.
16. Radford, A., Wu, J., Child, R., et al. (2019) *Language Models are Unsupervised Multitask Learners*. Available at: https://github.com/codelucas/newspaper.
17. Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378: 686–707. doi:10.1016/j.jcp.2018.10.045.
18. Reynolds, L. (2024) *LangChain: Open-source library for building LLM applications.* .
19. Rosofsky, S.G. and Huerta, E.A. (2023) Magnetohydrodynamics with physics informed neural operators. *Machine Learning: Science and Technology*, 4 (3): 035002. doi:10.1088/2632-2153/ace30a.

20. Vadyala, S.R., Betgeri, S.N., Matthews, J.C., et al. (2022) A review of physics-based machine learning in civil engineering. *Results in Engineering*, 13: 100316. doi:10.1016/j.rineng.2021.100316.
21. Vaswani, A., Brain, G., Shazeer, N., et al. (2017) *Attention Is All You Need*.
22. Verduzco, J.C., Holbrook, E. and Strachan, A. (2023) *GPT-4 as an interface between researchers and computational software: improving usability and reproducibility*. Available at: http://arxiv.org/abs/2310.11458.
23. Xie, K., Zhang, L., Li, X., et al. (2022) SES-X: A MBSE Methodology Based on SES/MB and X Language. *Information*, 14 (1): 23. doi:10.3390/info14010023.
24. Yang, S.D., Ali, Z.A. and Wong, B.M. (2023) FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer): Efficient Predictions of Particle Trajectories and Erosion. *Industrial and Engineering Chemistry Research*, 62 (37): 15278–15289. doi:10.1021/acs.iecr.3c01639.