Structural Engineering and Mechanics, *Vol. 96, No. 6 (2025) 523-533*
https://doi.org/10.12989/sem.2025.96.6.523

Multi-agent LLM framework integrating CAD, FEA, and optimization

Hojun Lee[1], Hyo-Jin Kim[2] and Jaeho Jung*[1]

[1]*Department of Mechanical Engineering, Chungbuk National University, 1 Chungdae-ro, Seowon-Gu, Chungbuk 28644, Republic of Korea*
[2]*KAIST InnoCORE PRISM-AI Center, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, 34141, Republic of Korea*

(*Received October 22, 2025, Revised November 17, 2025, Accepted November 18, 2025*)

Abstract. Mechanical design workflow typically involves iterative cycles of computer-aided design (CAD) modeling, finite element analysis, and optimization, each requiring significant expertise and manual effort. Recent advances in large language models (LLMs) have enabled the automation of individual stages, including CAD modeling and finite element analysis. Moreover, several studies have utilized LLMs for mechanical design. However, fully integrated end-to-end automation remains limited. Therefore, this study proposes a novel end-to-end framework based on LLM agents that achieves full automation of the entire design–analysis–optimization workflow. Driven entirely by natural language inputs, the framework integrates automatic parameter extraction, CAD modeling, meshing, structural analysis, and optimization. The proposed framework performs structural analyses and achieves optimization goals while preserving design constraints. This is demonstrated through four case studies: a cantilever beam, two-section bar, desk, and flanged pipe. Notably, the proposed approach preserves critical design parameters that are often implicit, thereby mimicking the decision-making of experienced engineers. These results demonstrate the feasibility of democratizing mechanical design by enabling non-experts to perform sophisticated tasks. Although the current implementation is confined to linear analysis and exhibits reduced robustness in highly complex scenarios, this work provides a promising foundation for AI-driven automation in engineering design.

Keywords: automated mechanical design; computer-aided design; design optimization; finite element analysis; large language model

1. Introduction

Mechanical design tasks are typically performed through iterative processes involving computer-aided design (CAD) for modeling, computer-aided engineering (CAE) for analyzing, and optimization for improving the solution. This traditional workflow is inherently fragmented because it spans different domains of expertise, including CAD modeling, CAE analyses, such as structural, thermal, and fluid analyses, and multidisciplinary optimization. Developing expertise in these areas is a highly time-intensive process. Moreover, because substantial parts of the workflow are performed manually, significant human and time resources are required, increasing development costs and undermining design efficiency.

To address these challenges, various efforts have been made to integrate or automate the design-analysis-optimization workflow. For instance, Wang *et al.* (2017) proposed a CAD/CAE-integrated framework for the structural design of machine tools. They combined a topology architecture model and a CAD/CAE coupling method with automated parameter extraction, finite element analysis (FEA) script generation, and response surface method optimization into a two-stage structural optimization procedure. This combined approach improved efficiency significantly. Lee (2005) proposed a CAD/CAE integration approach using feature-based nonmanifold topology modeling, which simultaneously generates and manages both design and analysis models. The incorporation of multi-resolution and multi-abstraction techniques to CAD/CAE methods enabled the immediate extraction of geometries with different levels of detail and abstraction from a single master model, eliminating redundant model conversions during iterative design–analysis cycles while maintaining consistency. In a similar direction, Uhm *et al.* (2008) developed a T-spline-based finite element framework that achieves direct CAD/CAE integration by employing a unified representation for both geometry and analysis models. These studies provide concrete examples of automation and integration processes that link CAD/CAE, analysis, and optimization, representing solid attempts to improve the efficiency of design cycles. In the industry, commercial software packages such as Siemens NX and ANSYS Workbench support collaborative efficiency by integrating design, analysis, and optimization functionalities. However, these methods have the limitation that they require personnel with specialized expertise.

This dependence on human expertise highlights the need for a new paradigm that can comprehend complex instructions and automate knowledge-based tasks. The rapid advancement of artificial intelligence (AI) has opened new possibilities across diverse fields (Jordan and Mitchell, 2015, LeCun *et al.* 2015), with large language models (LLMs) emerging as a transformative technology to address

*Corresponding author, Assistant Professor
E-mail: jhj@cbnu.ac.kr

ISSN: 1225-4568 (Print), 1598-6217 (Online)

this challenge. Recently, the LLM domain has expanded significantly in both performance and application scope (Zhao *et al.* 2023, Kumar *et al.* 2025). A series of increasingly powerful models, such as OpenAI's GPT family (Brown *et al.* 2020) and Meta's LLaMA (Touvron *et al.* 2023), have improved model scale, training data, and inference capabilities. By leveraging their sophisticated natural language understanding and generation capabilities, these models can perform various knowledge-intensive tasks, including code generation, data analysis, complex report generation, and technical documentation (Zhao *et al.* 2023).

LLMs can potentially automate complex mechanical design processes. For instance, Badagabettu *et al.* (2024) proposed Query2CAD, which uses LLMs to generate complete CAD models from natural language queries. The output is iteratively refined to obtain highly accurate CAD models by employing a self-refinement loop that integrates BLIP2-based caption generation with human feedback. Yuan *et al.* (2025) introduced CAD-Editor, a framework for modifying existing CAD models in response to natural language instructions. Through an automated data generation pipeline and a locate-then-infill editing structure, their approach enhanced the text-to-CAD alignment and enabled the effective execution of complex editing tasks. Hou *et al.* (2025) developed a framework that utilized LLMs to automatically generate physically valid simulation codes from natural language inputs. Utilizing a graph convolutional network–transformer-based step-by-step planning, they improved both the accuracy and executability of the generated code. Liang *et al.* (2025) proposed an LLM-based framework for the analysis of two-dimensional structures. Relying solely on natural-language problem descriptions, their system generated code using open-source FEA tools and produced complete structural analysis reports, including visualizations. Similarly, Park *et al.* (2024) used GPT-4o to validate FEA of a truss structure and perform design modifications and parameter sensitivity studies, thereby obtaining a simple parameter-driven model selection. The study showed that LLMs can support an end-to-end design-to-analysis workflow. Tian and Zhang (2025) extended the use of LLMs beyond single-model pipelines by configuring LLMs as multi-agent systems that collaborate to generate FEA code automatically. The authors evaluated the problem-solving effectiveness of the approach. Separately, Li *et al.* (2024) presented LLM4CAD. They showed that GPT-4/4V can produce 3D CAD models from diverse inputs, including text, sketches, and images. Subsequently, Sun *et al.* (2025) demonstrated that task-specific fine-tuning with a customized dataset can improve the reliability of CAD generation and its alignment with the user's intent.

However, these approaches are limited to specific stages of the workflow. Yu *et al.* (2024) proposed an intelligent, interactive system that links LLMs with CAD via secondary API development and an expert knowledge base built from FEA analysis data. The system was developed to guide the design and provide optimization suggestions. However, the modeling and analysis steps of the approach still require user intervention. Consequently, complete automation of the design process is an open problem. To the best of our knowledge, no prior studies have fully integrated the design-analysis-optimization workflow into a single, end-to-end pipeline that automates three-dimensional structural analysis. To address this gap, this study proposes a novel framework integrating CAD modeling, FEA, and optimization into an end-to-end pipeline driven entirely by natural languages using LLMs. The primary advantage of this framework is the democratization of the mechanical design process, which enables individuals with basic knowledge of engineering software to execute sophisticated design tasks.

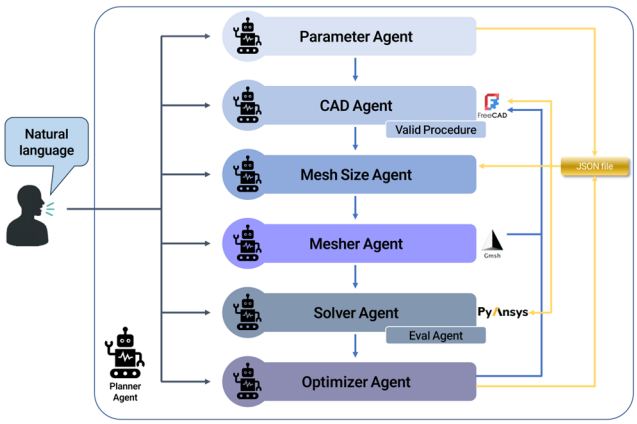

Fig. 1 Architecture and workflow of the proposed multi-agent LLM framework

The remainder of this paper is organized as follows: Section 2 describes the proposed methodology, Section 3 presents comparative results for several examples, Section 4 provides discussions, and Section 5 presents the concluding remarks.

2. Method

The performance of LLMs is highly sensitive to the quality of prompts provided by users (Giray, 2023, Sahoo *et al.* 2024). Therefore, in this study, prompts were systematically designed and refined through an iterative trial-and-error process to optimize performance empirically. The overall framework is illustrated in Fig. 1.

The Parameter Agent extracts numerical parameters from natural language inputs, and the extracted parameters are output in JavaScript Object Notation (JSON) format. The CAD Agent generates CAD models based on natural language instructions and validates the process to detect and correct errors. After creating the CAD model, the Mesh Size Agent infers a uniform grid size, and the Mesher Agent generates the mesh. The CAD model is automatically regenerated if an error occurs during meshing to prevent workflow failure. The Solver Agent produces analysis code using the Python interface of the ANSYS Mechanical APDL (PyMAPDL), delivers the requested results, and the integrated Evaluation Agent verifies whether the results satisfy the conditions specified by the user, thereby improving the accuracy. The Optimizer Agent modifies the parameters in the JSON file to optimize the geometry based on the target objectives defined by the user.

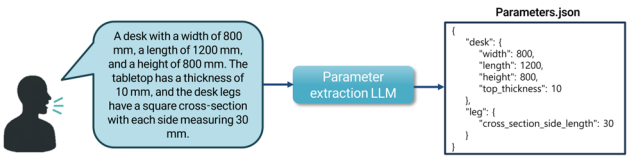

Fig. 2 Workflow of the Parameter Agent. Numerical parameters are extracted from natural language, and default values are generated for missing values. The parameters are exported as a JSON file

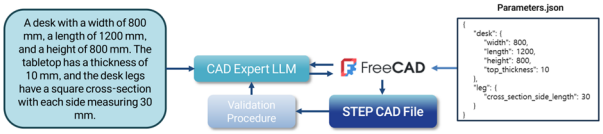

Fig. 3 Workflow of the CAD Agent. 3D CAD models (STEP files) are generated by interpreting natural language descriptions and JSON parameters. It includes an iterative validation process to ensure model robustness

The Planner Agent develops the workflow by sequentially executing the parameter, CAD, Mesh Size Agent, Mesher Agent, Solver Agent, and Optimizer Agents. When the Optimizer Agent adjusts the parameters, the CAD Agent reinitiates the workflow automatically, and the process is repeated until the specified objectives are achieved, thereby realizing a fully automated design process. This framework applies to three-dimensional problems. Moreover, because the entire process can be conducted using only natural language input, non-experts can easily perform mechanical design tasks.

2.1 Parameter agent

The process of the Parameter Agent is illustrated in Fig. 2. The LLM employed for this agent was GPT-5. The Parameter Agent extracts numerical parameters expressed in natural language and converts them into JSON files. If a numerical parameter is not explicitly defined in the natural language input, the agent automatically generates a reasonable default for the missing parameter. The parameter information is exported as a JSON file, enabling model modifications during the optimization stage without requiring further natural-language inputs. Instead, it relies on the numerical parameters stored in the JSON file.

2.2 CAD agent

The CAD Agent process is illustrated in Fig. 3. This agent also employs GPT-5 as the underlying LLM. The CAD Agent generates CAD models from the natural language inputs. Although the agent interprets the geometric shape from the textual description, it retrieves the numerical parameters from the JSON file produced by the Parameter Agent. The CAD Agent utilizes FreeCAD (Falck *et al.* 2012), an open-source software, and generates Python scripts that drive the modeling process. In this study, FreeCAD (version 1.0.0) was executed via the command-line interface in batch mode, without launching the graphical user interface, and the CAD geometry was

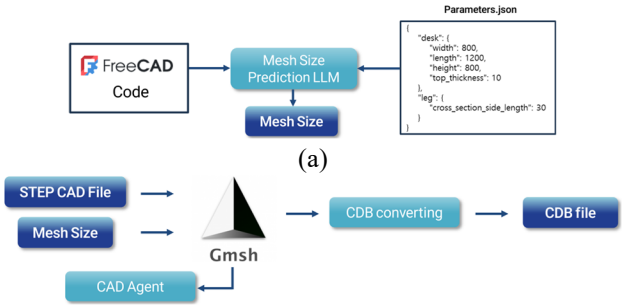

Fig. 4 Workflow of the Mesh Size and Mesher Agents: (a) Mesh Size Agent determines the uniform mesh element size (mm), and (b) Mesher Agent generates the mesh with Gmsh 4.14.0 and handles mesh file conversion and error recovery

constructed through the FreeCAD Python API. The resulting CAD model is subsequently converted into Standard for the Exchange of Product Data (STEP) files to enable further analysis. A validation procedure was integrated to ensure that the CAD file was error-free. This validation included verifying file existence, confirming that its size was not excessively small, ensuring proper file loading, and verifying that the CAD model contained no topological inconsistencies. If errors were detected, the corresponding error messages were fed back to the CAD Agent, which regenerated the model. This iterative validation and correction process helps mitigate potential errors, thereby improving the robustness of the agent.

2.3 Mesh size agent & Mesher agent

The processes of the Mesh Size and Mesher Agents are illustrated in Fig. 4. The Mesh Size Agent, implemented using GPT-5, determines an appropriate uniform element size in millimeters for the FEA and records this information in a structured format. Based on this specification, the Mesher Agent generates a mesh using Gmsh (Geuzaine *et al.* 2009). The Mesher Agent imported the STEP geometry using the built-in STEP reader in Gmsh and subsequently synchronized the boundary representation using the Open Cascade kernel before meshing. The version of Gmsh used in this study was 4.14.0. The mesh was constructed using linear four-node tetrahedral elements (SOLID285). If errors occur during mesh generation, the workflow returns to the CAD Agent, which reconstructs the CAD model to prevent failure. The finalized mesh is first saved as an .inp file, a text-based input file widely used by finite element solvers such as Abaqus, and then converted into the CDB format—a coded database archive native to the ANSYS environment—to ensure direct integration with the solver via PyMAPDL.

2.4 Solver agent

The Solver Agent process is illustrated in Fig. 5. This agent also employs GPT-5 as the underlying LLM. The Solver Agent receives natural language inputs describing the model geometry, boundary conditions, and material

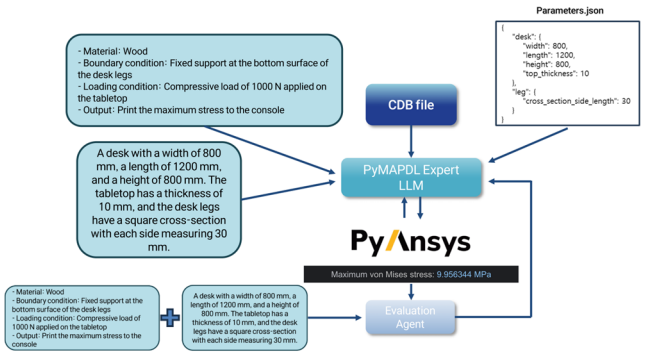

Fig. 5 Solver Agent workflow. Analysis code is generated from natural language inputs and is coordinated with an internal Evaluation Agent to revise the analysis accuracy

properties. Based on this information, it generates the corresponding analysis code in ANSYS PyMAPDL to perform structural simulations. Because the other agents focus on the geometry and mesh data, the Solver Agent focuses solely on analyzing the model and producing the results requested by the user. An internal Evaluation Agent, also implemented in GPT-5, ensures the validity of the generated analysis. The Evaluation Agent verifies whether the code of the Solver Agent correctly reflects the specified geometry, boundary conditions, and output requirements. If discrepancies or errors are detected, the corresponding feedback is returned to the Solver Agent, which revises the analysis to reduce potential errors and improve robustness. The mesh generated by the Mesher Agent, stored in CDB format, is loaded into PyMAPDL, where the cdread command is used to reconstruct the finite element database directly within the PyMAPDL model space. The version of PyMAPDL used in this study was 0.70.2, which corresponds to ANSYS Mechanical APDL 2024 R2.

2.5 Optimizer agent

The Optimizer Agent process is illustrated in Fig. 6. This agent evaluates deviations from the target values specified by the user and optimizes to achieve the desired design objectives. The Optimizer Agent was implemented using GPT-5. Instead of relying on a specific numerical optimization algorithm, LLM autonomously performed the optimization process. To enhance traceability, the optimization history is recorded in a text log, which is used to refine prompts and guide the process toward achieving target values. The parameters updated during optimization were extracted in the JSON format and fed back into the CAD Agent to regenerate the design. In cases where errors occur, the corresponding error messages are incorporated into subsequent prompts, allowing the agent to resolve the issues iteratively and continue the optimization process.

3. Results

Four design examples—Cantilever Beam, Two-Section Bar, Desk, and Flanged Pipe—were developed using the proposed framework to validate its performance and

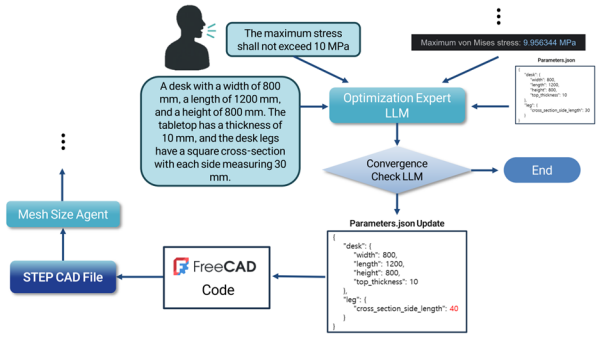

Fig. 6 Workflow of the Optimizer Agent. Design parameters are iteratively updated based on target deviations and error feedback to refine the design

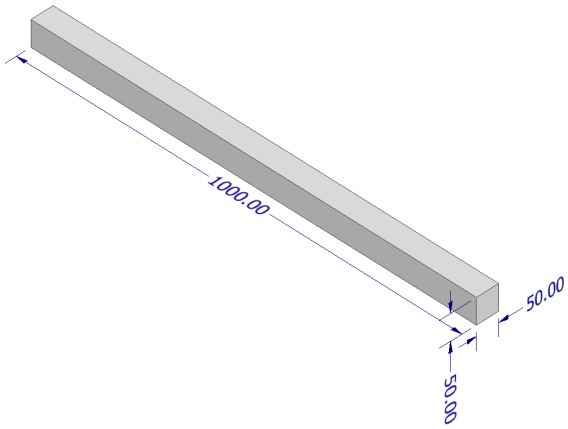

Fig. 7 Initial configuration of the 1,000 mm cantilever beam (50×50 mm cross-section) generated by the CAD Agent from the user's natural language prompt

effectiveness. For the Flanged Pipe example, two different boundary conditions and optimization objectives were applied to examine whether the proposed methodology could maintain stable performance under more complex optimization scenarios.

3.1 Cantilever beam

A cantilever beam was first considered. The beam had a square cross-section of 50 mm×50 mm and a length of 1,000 mm. The user provided the prompt: "length 1 m, width and height 50 mm, cantilever beam, aligned with the x-axis". Based on this input, the Parameter Agent extracted the corresponding beam parameters into a structured JSON file. The CAD Agent interpreted the geometry requested in the prompt. Subsequently, it constructed a CAD model utilizing the parameter values stored in the JSON file. The initial geometry of the cantilever beam generated by the CAD Agent is illustrated in Fig. 7.

Based on the STEP file generated by the CAD Agent, the Mesh Size Agent recommended an initial element size of 10 mm. The Mesher Agent used this specification to produce an initial mesh comprising 12,791 elements, as shown in Fig. 8. The mesh geometry was visualized using the ANSYS Mechanical software to confirm the quality of the generated model.

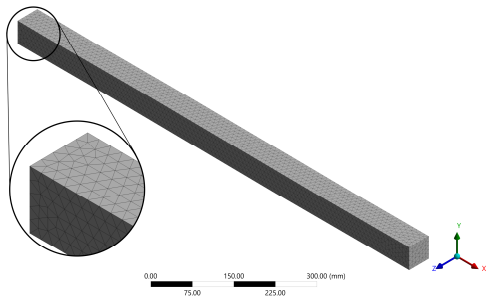

Fig. 8 Initial mesh of the cantilever beam generated by the Mesher Agent, which consisted of 12,791 elements with a specified element size of 10 mm

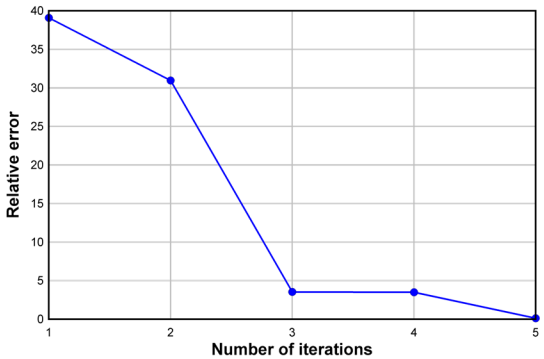

Fig. 9 Relative error of the cantilever beam during optimization. The figure shows the design's evolution as the Optimizer Agent modifies the parameters to achieve the target displacement (2 mm)

Table 1 Parameter updates during optimization iterations for the Cantilever Beam

		Number of iterations				
		1	2	3	4	5
Parameter (mm)	Length	1,000	1,000	1,000	1,000	1,000
	Width	50	60	55	54	54.5
	Height	50	60	55	54	54.5

As described in Section 2, the Solver Agent required inputs for the boundary conditions and material properties. The user provided the prompt for the cantilever beam analysis: "material: steel, boundary condition: fixed at one end, load: 1,000 N applied on the opposite end face in the y-direction, output: maximum absolute displacement in the load direction". The Solver Agent generated the corresponding analysis and yielded the result: "Maximum absolute displacement in the y-direction: 2.781732 mm". Subsequently, the Optimizer Agent was assigned a target value of 2 mm for the maximum displacement in the y-direction, with a tolerance of 1%. The first optimization iteration yielded an updated geometry, with each cross-sectional dimension increased by 10 mm. The modified parameters were stored in a JSON file to regenerate the CAD model. Optimization was performed iteratively by repeating this process multiple times. The optimization procedure for the cantilever beam is illustrated in Fig. 9, and the evolution of the parameters across the iterations is summarized in Table 1.

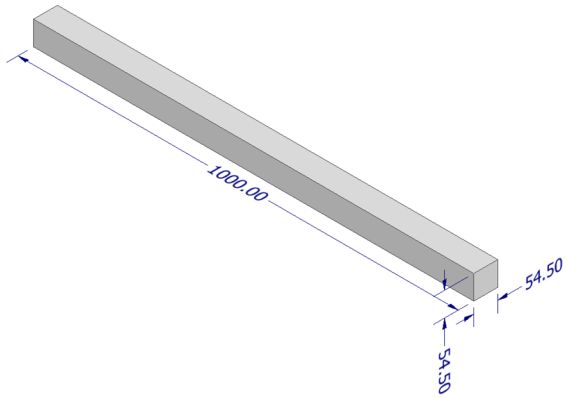

Fig. 10 Final optimized geometry of the cantilever beam, which shows the modified cross-sectional dimensions while the beam length remains unchanged. Units in mm.

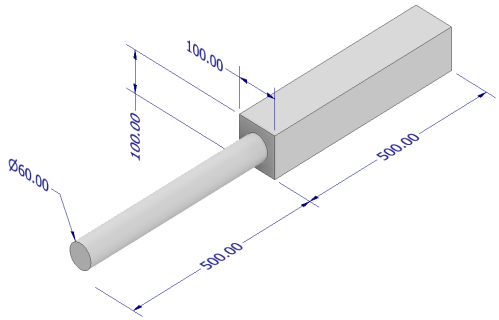

Fig. 11 Initial configuration of the 1,000-mm two-section bar (100×100 mm square section from 0–500 mm and 60-mm-diameter circular section from 500–1000 mm) generated by the CAD Agent from the user's natural language prompt

The final optimized geometry of the cantilever beam is shown in Fig. 10. The results indicate that the beam length remained constant, whereas only the cross-sectional dimensions were modified. This outcome is consistent with standard practices in structural design, where the beam length is typically fixed by connection or boundary constraints, and optimization focuses on altering the cross-sectional properties to satisfy specific performance targets. Although these constraints were not explicitly defined in the prompts provided to the LLM agent, the optimization process inherently preserved the beam length while varying the cross-section to achieve the desired objectives.

3.2 Two-section bar

We considered a two-section bar comprising a 100×100 mm square member and a 60 mm-diameter circular member, each 500 mm in length. The user constructed this example by providing the following prompt: "length 1000 mm, vertical column aligned with the z-axis (model axis aligned with z), from 0 to 500 mm a square cross-section of 100 mm×100 mm, and from 500 to 1000 mm a circular cross-section with a diameter of 60 mm. The centroids of both the sections were concentric and shared the same longitudinal axis". Based on this prompt, the Parameter Agent successfully extracted the numerical parameters and

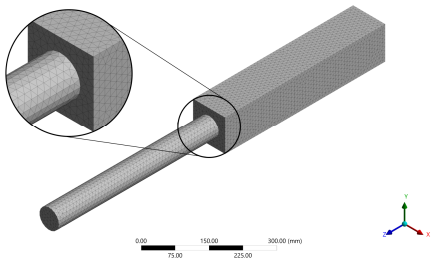

Fig. 12 Initial mesh of the two-section bar generated by the Mesher Agent. The model comprised 31,037 elements with a specified element size of 10 mm

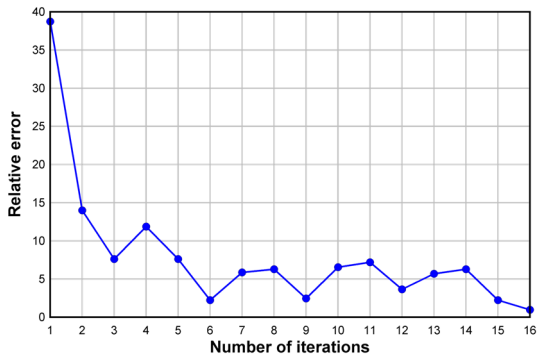

Fig. 13 Relative error of the two-section bar during optimization. The design's evolution varies as the Optimizer Agent modifies geometric parameters to achieve the target displacement (0.2 mm)

stored them in structured JSON format without errors. Subsequently, the CAD Agent used these parameters to generate the corresponding CAD model. The initial geometry of the two-section bar constructed by the CAD Agent is illustrated in Fig. 11.

Based on the STEP file generated by the CAD Agent, the Mesh Size Agent recommended an initial element size of 10 mm. This specification enabled the Mesher Agent to generate an initial mesh comprising 31,037 elements, as shown in Fig. 12. The mesh geometry was visualized using ANSYS Mechanical software to confirm the quality and validity of the generated model.

The boundary conditions and material properties were defined using the following prompt: "material: wood, boundary condition: fixed at the base along the longitudinal axis, load: total of 100,000 N applied to the top surface in the direction of the z-axis, output: maximum absolute displacement in the load direction". The Solver Agent generated the corresponding analysis, yielding "Maximum absolute displacement in the z-axis direction: 0.122504 mm". Subsequently, the Optimizer Agent was assigned a target displacement of 0.2 mm in the z-axis direction, with a tolerance of 1%. In the first optimization iteration, the square cross-section was reduced by 20 mm on each side, whereas the diameter of the circular cross-section was reduced by 10 mm. Consequently, the maximum displacement increased from 0.122504 to 0.171986 mm, approaching the target. The optimization procedure is illustrated in Fig. 13. A total of 16 optimization iterations were required. In the final model, the square cross-section

Table 2 Parameter updates at selected iterations for the two-section bar

		Number of iterations			
		4	8	12	16
Parameter (mm)	Square Length	500	500	500	500
	Square size	68	72	74	71
	Circle Length	500	500	500	500
	Circle Diameter	44	45	45	46

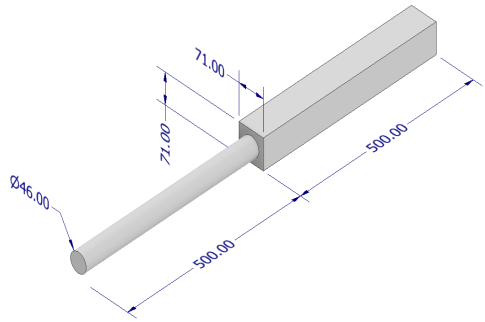

Fig. 14 Final optimized geometry of the two-section bar. The figure shows the modified cross-sectional dimensions while the overall length remains unchanged. Units in mm

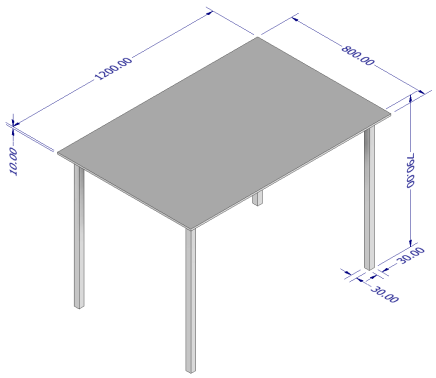

Fig. 15 Initial configuration of the desk model (800×1200×10-mm tabletop supported by four 790-mm-high square legs with a 30×30-mm cross-section) generated by the CAD Agent from the user's natural language prompt

was reduced by 29 mm on each side, and the diameter of the circular cross-section was reduced by 16 mm. The representative parameter changes at the 4th, 8th, 12th, and 16th iterations are summarized in Table 2. Similar to the cantilever beam case described in Section 3.1, the overall length of the structure remained unchanged throughout the optimization process. The final optimized geometry of the two-section bar is shown in Fig. 14.

3.3 Desk

A validation test was performed using a desk model representative of everyday applications. The test aimed to examine the practical applicability of the proposed framework. The desk consisted of a tabletop measuring 800 mm×1200 mm×10 mm (width×length×thickness), supported by four square legs each 790 mm in height with a

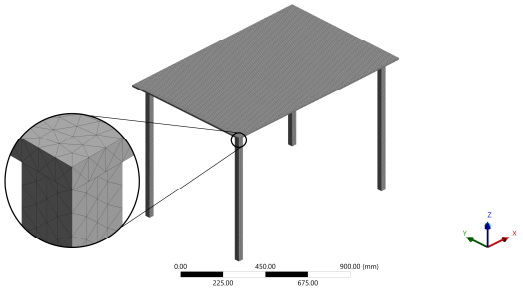

Fig. 16 Initial mesh of the desk model generated by the Mesher Agent. The model comprised 87,383 elements with a specified element size of 10 mm

30 mm×30 mm cross-section. The user provided the following prompt to verify the LLM agents: "desk with width 800 mm, length 1200 mm, height 800 mm, tabletop thickness 10 mm, z-axis as the vertical direction, and four legs with square cross-sections of 30 mm per side". Based on this input, the Parameter Agent successfully extracted the numerical parameters and stored them in a structured JSON format without errors. The CAD Agent then generated the desk model using these parameters. The initial geometry of the desk constructed using the CAD Agent is shown in Fig. 15.

The Mesh Size Agent recommended an initial element size of 10 mm, based on the STEP file generated by the CAD Agent after one retry. This specification enabled the agent to create an initial mesh comprising 87,383 elements. The mesh geometry was visualized using ANSYS Mechanical software to review the quality of the generated model (Fig. 16). In this example, two cases were examined for validation. Case 1 utilized the original multi-agent configuration without modifying the initial prompt. Case 2 included additional material information to the initial prompt. Thus, the JSON file contained both geometric numerical parameters and material type.

For Case 1, the following prompt was provided to define the boundary conditions and material properties for this example: "material: wood, boundary condition: legs fixed at the base, load: 1000 N compressive force applied on the tabletop, output: maximum stress". The Solver Agent produced the result "Maximum von Mises stress: 9.956344 MPa". The Optimizer Agent was then assigned to reduce the maximum stress to less than 5 MPa. In the first optimization step, the tabletop thickness was increased by 10 mm, and the side length of each leg cross-section was increased by 10 mm. Consequently, the maximum stress decreased to 3.915861 MPa, successfully achieving the target. The optimization process and final desk model are presented in Figs. 17 and 18, respectively.

The initial conditions used in Case 1 were the same for Case 2, except that the output requirement was set to "maximum absolute displacement along the load direction". The Solver Agent generated the result "Max absolute displacement along load direction: 3.26585 mm". The Optimizer Agent was then assigned the target to reduce the maximum displacement to below 1 mm. In the first optimization step, no geometric dimensions were modified, instead, only the material property was changed, where the

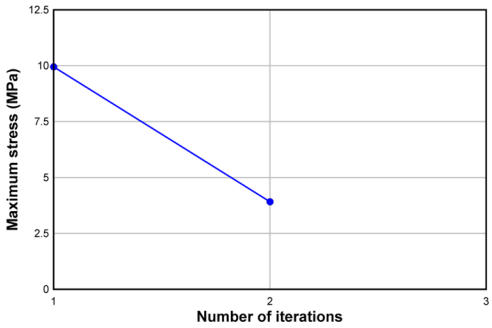

Fig. 17 Maximum stress of the desk model (Case 1) during optimization. The figure shows the evolution of the design as the Optimizer Agent modifies geometric parameters to achieve the target stress level (5 MPa)

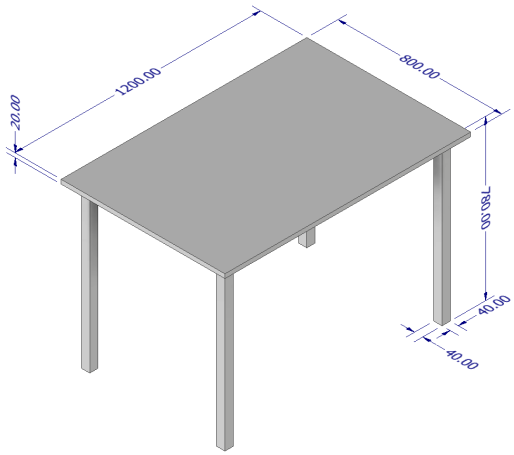

Fig. 18 Final optimized geometry of the desk model (Case 1). The figure shows the increased tabletop thickness and enlarged leg cross-sections while the overall dimensions remain unchanged. Units in mm

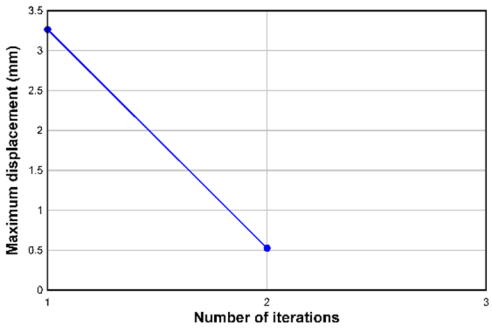

Fig. 19 Maximum displacement of the desk model (Case 2) during optimization. The figure shows the evolution of the design as the Optimizer Agent modifies material parameters to achieve the target displacement (1 mm)

tabletop and legs were replaced with aluminum. Consequently, the maximum displacement decreased to 0.527530 mm, successfully meeting the target requirement. The optimization process is presented in Fig. 19.

3.4 Flanged pipe

A validation test was performed using a flanged pipe to

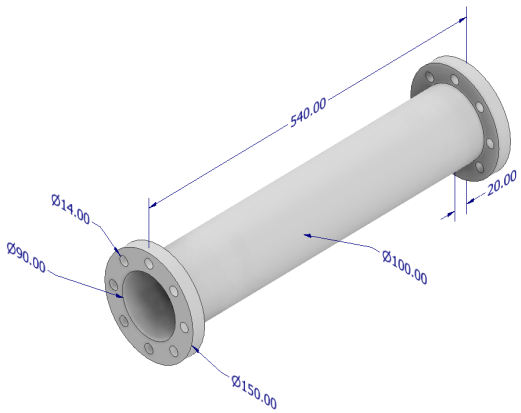

Fig. 20 Initial configuration of the flanged pipe (100 mm diameter and 5 mm thickness, with flanges, at both ends, consisting of eight bolt holes) generated by the CAD Agent from the user's natural language prompt. All dimensions are in millimeters (mm)

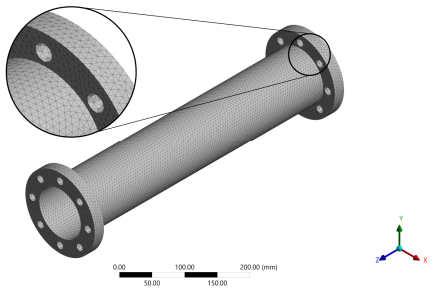

Fig. 21 Initial mesh of the flanged pipe model generated by the Mesher Agent. The model comprised 60,570 elements with a specified element size of 5 mm

further assess the practical applicability of the proposed methodology. The flanged pipe represents a frequently used component in industrial settings. In this example, the performance of the Parameter Agent was evaluated by specifying certain flanged pipe dimensions. The user provided the following prompt: "a pipe with flanges at both ends, each flange containing eight bolt holes. The pipe has a diameter of 100 mm and a thickness of 5 mm. The flange base was placed on the xy-plane". The Parameter Agent automatically assigns appropriate values for parameters not explicitly defined in the prompt. These parameters enable the CAD Agent to generate the initial geometry of the flanged pipe, as shown in Fig. 20.

The Mesh Size Agent recommended an initial element size of 5 mm, based on which the Mesher Agent generated a mesh comprising 60,570 elements. The mesh was visualized using ANSYS Mechanical software to verify its quality, as shown in Fig. 21.

In this example, two cases were examined for validation. For Case 1, the user provided the prompt: "material: steel, boundary condition: flange base fixed, load: internal pressure of 5 MPa applied to the inner surface of the pipe, output: maximum stress". The Solver Agent produced "Maximum von Mises stress: 50.525271 MPa". The Optimizer Agent was then assigned the target of reducing the maximum stress to less than 30 MPa while

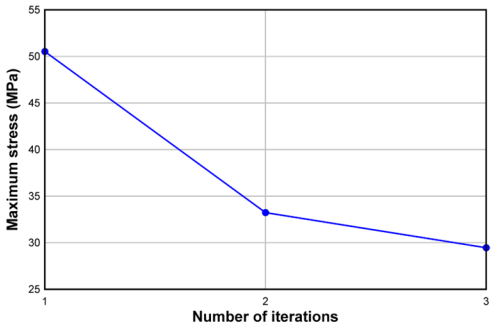

Fig. 22 Maximum stress of the flanged pipe model (Case 1) during optimization. The figure shows the evolution of the design as the Optimizer Agent modifies geometric parameters to achieve the target stress level (30 MPa)

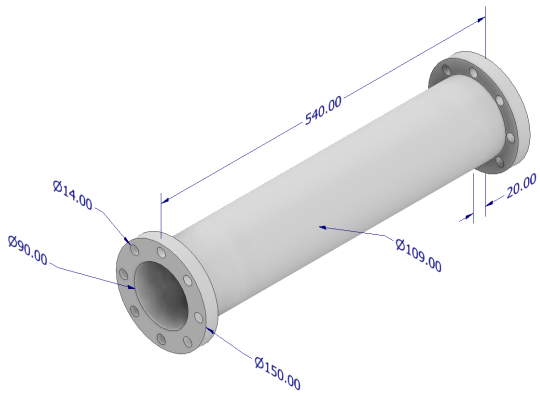

Fig. 23 The final optimized geometry of the flanged pipe model (Case 1). It has an increased pipe wall thickness while the overall length remains unchanged. All dimensions are in mm

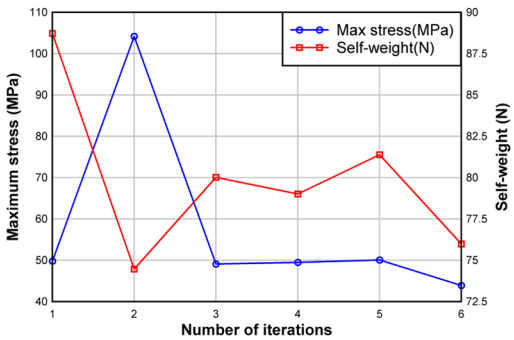

Fig. 24 Maximum stress and self-weight of the flanged pipe model (Case 2) during optimization. The figure shows the evolution of the design as the Optimizer Agent modifies the geometric parameters to achieve two distinct objectives with their respective target values: stress (45 MPa) and self-weight (80 N)

maintaining the inner diameter of the pipe. After three optimization iterations, the target was achieved. The optimization process resulted in an increase of 9.5 mm in pipe wall thickness. The parameter changes across iterations are summarized in Table 3. The optimization procedure and final optimized model are illustrated in Figs. 22 and 23, respectively.

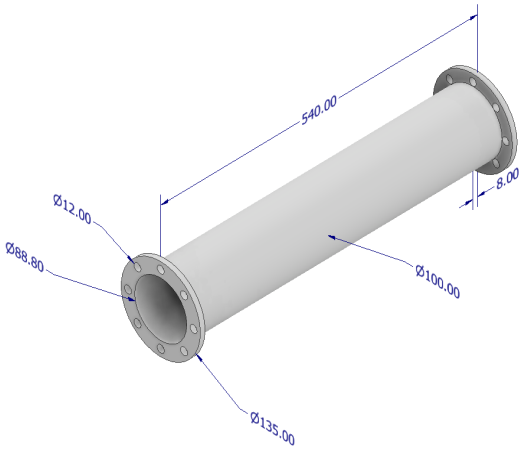

Fig. 25 Final optimized geometry of the flanged pipe model (Case 2). The figure shows the modified cross-sectional dimensions while the overall length remained unchanged. Units in mm

Table 3 Updates of design parameters during optimization iterations for the flanged pipe (Case 1)

		Number of iterations		
		1	2	3
Parameter (mm)	Length	540	540	540
	Pipe Outer Diameter	100	107	109
	Pipe Inner Diameter	90	90	90
	Pipe wall Thickness	5	8.5	9.5
	Flange Outer Diameter	150	150	150
	Flange Inner Diameter	90	90	90
	Flange Thickness	20	20	20
	Bolt Hole Diameter	14	14	14
	Bolt Circle Diameter	125	125	125

In Case 2, the same initial conditions as in Case 1 were applied, except that the output requirements in the Solver Agent were modified. Specifically, the maximum stress and self-weight were set as outputs. The analysis results were "Maximum von Mises stress: 49.7993 MPa, Self-weight: 88.7191 N". The Optimizer Agent was tasked with reducing the maximum stress to below 45 MPa and the self- weight to below 80 N. After six optimization iterations, both conditions were satisfied. In the final design, the pipe wall thickness increased by 0.6 mm, whereas the outer diameter of the flange decreased by 15 mm, the inner diameter decreased by 2 mm, and the flange thickness decreased by 12 mm. Additionally, the bolt circle diameter was reduced by 8 mm, and the bolt hole diameter was reduced by 2 mm. The parameter changes across iterations are summarized in Table 4. The optimization process is illustrated in Fig. 24, and the final optimized model is shown in Fig. 25.

4. Discussion

The four cases studied demonstrated that the proposed multi-agent LLM framework can autonomously perform the entire design–analysis–optimization loop without manual intervention. The system successfully translated natural language descriptions into parametric models, executed structural simulations, and iteratively adjusted design variables. The results showed that LLMs can serve as code generators and as task-level reasoning entities. The cantilever beam, two-section bar, desk (Case 1), and flanged pipe (Case 1) examples validated single-objective optimization through geometric parameter tuning. Moreover, the flanged pipe (Case 2) case confirmed that the same workflow can be extended to multi-objective optimization. Furthermore, the desk (Case 2) experiment demonstrated that material properties can be treated as design variables, indicating that the JSON-based parameterization scheme supports parameter types beyond geometry. This implies that the framework can optimize heterogeneous sets of parameters and that geometry, material properties, and potentially other system-level variables can be jointly optimized within a unified workflow. Note that these results were achieved without any task-specific fine-tuning of the LLM. Therefore, additional performance improvements are expected if knowledge-grounding techniques or domain adaptation approaches are incorporated. Particularly, integrating external engineering knowledge through methods, such as retrieval-augmented generation—a mechanism where relevant domain knowledge is retrieved from a structured database or document repository and dynamically provided to the LLM during reasoning and code generation, could significantly enhance accuracy and reliability. Nevertheless, these implications should be interpreted within the scope of the present study. Our validation focused primarily on relatively simple geometries and linear structural analyses. When more complex assemblies or tightly coupled constraints were introduced, the autonomous success rate decreased. This suggests that scaling the framework to more complex engineering problems requires more capable LLMs, structured constraint handling, and improved reasoning for interdependent design variables.

Table 4 Parameter updates during optimization iterations for the Flanged Pipe (Case 2)

		Number of iterations					
		1	2	3	4	5	6
Parameter (mm)	Length	540	540	540	540	540	540
	Pipe Outer Diameter	100	100	100	100	100	100
	Pipe Inner Diameter	90	90	90	90	90	88.8
	Pipe wall Thickness	5	5	5	5	5	5.6
	Flange Outer Diameter	150	125	150	146	160	135
	Flange Inner Diameter	90	90	90	90	90	88.8
	Flange Thickness	20	28	13	15	11	8
	Bolt Hole Diameter	14	16	12	16	12	12
	Bolt Circle Diameter	120	108	120	120	142	112

5. Conclusions

This study introduced a novel framework using LLM-based agents to automate an end-to-end mechanical design workflow, encompassing CAD modeling, FEA, and

optimization, all driven by natural language inputs. This approach significantly lowers the entry barrier for engineering design, enabling non-experts to execute complex design tasks that typically require specialized software expertise.

Four case studies were implemented for validation. The results demonstrate the versatility of the framework across diverse geometries under various conditions. The results also indicate that the proposed approach is not limited to a specific type of structure. Instead, the framework can be applied to various geometries and conditions, facilitating automated design for relatively simple engineering problems. Furthermore, the framework exhibited practical potential for integration into industrial automation pipelines. A particularly notable finding was that certain parameters typically fixed in conventional design practices, such as the lengths of beams and bars or the overall height of the desk, were also preserved during the optimization process. These design decisions are typically made by experienced engineers, underscoring the practical relevance of the proposed methodology for real-world applications.

Nevertheless, this study did not address more complex scenarios in actual industrial environments. The current framework is limited to linear structural analysis and does not address scenarios involving nonlinearities, fluid–structure interaction, and multiphysics simulations. The results suggested that the success rate of the automated process decreases as the complexity of the geometry and constraints in the examples increases. Therefore, future studies should focus on enhancing the robustness of the framework and expanding its capabilities to include a broader range of complex engineering problems. Representative examples include geometrically nonlinear fiber–beam formulations, isogeometric plate analyses, and large-scale fluid–structure interaction simulations for liquid containment systems. These problems provide challenging testbeds in which future multi-agent LLM frameworks could assist in automating advanced finite element analysis and element development (Hu and Wu 2014, Shojaee and Valizadeh 2012, Park and Cho 2012).

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-24533642).

References

Badagabettu, A., Yarlagadda, S.S. and Farimani, A.B. (2024), "Query2CAD: Generating CAD models using natural language queries", arXiv preprint arXiv:2406.00144.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., ... & Amodei, D. (2020), "Language models are few-shot learners", *Adv. Neur. Inform. Proc. Syst.*, **33**, 1877-1901.

Falck, B., Falck, D. and Collette, B. (2012), *FreeCAD How-To*, Packt Publishing Ltd, Birmingham, UK.

Geuzaine, C. and Remacle, J.F. (2009), "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities", *Int. J. Numer. Meth. Eng.*, **79**(11), 1309-1331. https://doi.org/10.1002/nme.2579.

Giray, L. (2023), "Prompt engineering with ChatGPT: A guide for academic writers", *Ann. Biomed. Eng.*, **51**(12), 2629-2633. https://doi.org/10.1007/s10439-023-03352-y.

Hou, S., Johnson, R., Makhija, R., Chen, L. and Ye, Y. (2025), "AutoFEA: Enhancing AI copilot by integrating finite element analysis using large language models with graph neural networks", *Proceedings of the AAAI Conference on Artificial Intelligence*, **39**(22), 24078-24085.

Hu, Z. and Wu, M. (2014), "Large displacement geometrically nonlinear finite element analysis of 3D Timoshenko fiber beam element", *Struct. Eng. Mech.*, **51**(4), 601-625. https://doi.org/10.12989/sem.2014.51.4.601.

Jordan, M.I. and Mitchell, T.M. (2015), "Machine learning: Trends, perspectives, and prospects", *Sci.*, **349**(6245), 255-260. https://doi.org/10.1126/science.aaa8415.

Kumar, P. (2024), "Large language models (LLMs): Survey, technical frameworks, and future challenges", *Artif. Intell. Rev.*, **57**(10), 260. https://doi.org/10.1007/s10462-024-10888-y.

LeCun, Y., Bengio, Y. and Hinton, G. (2015), "Deep learning", *Nature*, **521**(7553), 436-444. https://doi.org/10.1038/nature14539.

Lee, S.H. (2005), "A CAD–CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques", *Comput.-Aid. Des.*, **37**(9), 941-955. https://doi.org/10.1016/j.cad.2004.09.021.

Li, X., Sun, Y. and Sha, Z. (2024), "LLM4CAD: Multi-modal large language models for 3D computer-aided design generation", *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, **88407**, V006T06A015.

Liang, H., Kalaleh, M.T. and Mei, Q. (2025), "Integrating large language models for automated structural analysis", arXiv preprint arXiv:2504.09754.

Park, M., Bong, G., Kim, J. and Kim, G. (2024), "Structural analysis and design using generative AI", *Struct. Eng. Mech.*, **91**(4), 393-401. https://doi.org/10.12989/sem.2024.91.4.393.

Park, S.W. and Cho, J.R. (2012), "Adaptive fluid-structure interaction simulation of large-scale complex liquid containment with two-phase flow", *Struct. Eng. Mech.*, **41**(4), 559-573. https://doi.org/10.12989/sem.2012.41.4.559.

Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S. and Chadha, A. (2024), "A systematic survey of prompt engineering in large language models: Techniques and applications", arXiv preprint arXiv:2402.07927.

Shojaee, S. and Valizadeh, N. (2012), "NURBS-based isogeometric analysis for thin plate problems", *Struct. Eng. Mech.*, **41**(5), 617-632. https://doi.org/10.12989/sem.2012.41.5.617.

Sun, Y., Li, X. and Sha, Z. (2025), "Large language models for computer-aided design fine-tuned: Dataset and experiments", *J. Mech. Des.*, **147**(4), 041710. https://doi.org/10.1115/1.4067713.

Tian, C. and Zhang, Y. (2024), "Optimizing collaboration of LLM-based agents for finite element analysis", arXiv preprint arXiv:2408.13406.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023), "Llama: Open and efficient foundation language models", arXiv preprint arXiv:2302.13971.

Uhm, T.K., Kim, K.S., Seo, Y.D. and Youn, S.K. (2008), "A locally refinable T-spline finite element method for CAD/CAE integration", *Struct. Eng. Mech.*, **30**(2), 225-245. https://doi.org/10.12989/sem.2008.30.2.225.

Wang, J., Niu, W., Ma, Y., Xue, L., Cun, H., Nie, Y. and Zhang, D. (2017), "A CAD/CAE-integrated structural design framework for machine tools", *Int. J. Adv. Manuf. Technol.*, **91**(1), 545-568. https://doi.org/10.1007/s00170-016-9721-y.

Yu, M., Sheng, Y., Wan, Z., Yang, S., Sun, M. and Cai, H. (2024),

"An intelligent interactive system based on LLM to combine CAD API development with FEA", *2024 8th International Conference on Electrical, Mechanical and Computer Engineering (ICEMCE)*, 1749-1755.

Yuan, Y., Sun, S., Liu, Q. and Bian, J. (2025), "Cad-editor: A locate-then-infill framework with automated training data synthesis for text-based cad editing", arXiv preprint arXiv:2502.03997.

Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J.R. (2023), "A survey of large language models", arXiv preprint arXiv:2303.18223, **1**(2).

PL