

AUTOMATING STRUCTURAL ENGINEERING WORKFLOWS WITH LARGE LANGUAGE MODEL AGENTS

Anonymous authors

Paper under double-blind review

ABSTRACT

We introduce **MASSE**, the first Multi-Agent System for Structural Engineering, effectively integrating large language model (LLM)-based agents with real-world engineering workflows. Structural engineering is a fundamental yet traditionally stagnant domain, with core workflows remaining largely unchanged for decades despite its substantial economic impact and global market size. Recent advancements in LLMs have significantly enhanced their ability to perform complex reasoning, long-horizon planning, and precise tool utilization—capabilities well aligned with structural engineering tasks such as interpreting design codes, executing load calculations, and verifying structural capacities. We present a proof-of-concept showing that most real-world structural engineering workflows can be fully automated through a training-free LLM-based multi-agent system. MASSE enables immediate deployment in professional environments, and our comprehensive validation on real-world case studies demonstrates that it can reduce expert workload from approximately two hours to mere minutes, while enhancing both reliability and accuracy in practical engineering scenarios.

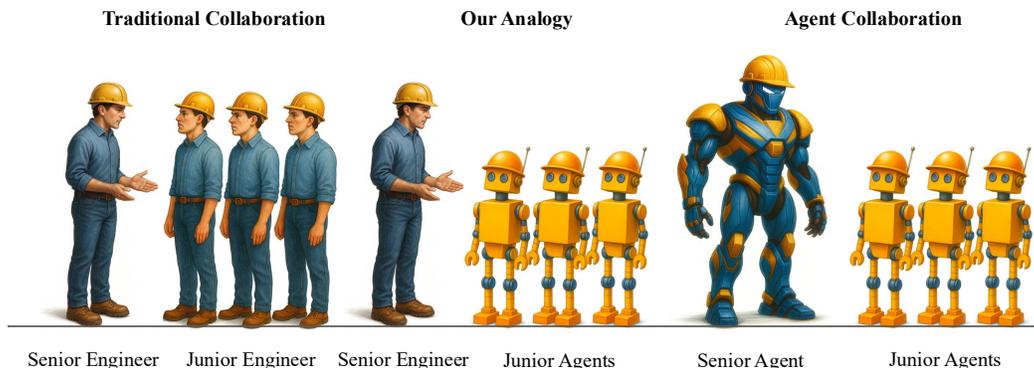


Figure 1: **Analogy of future human-AI collaborations.** Traditional practice relies on long apprenticeships, with senior engineers transferring expertise to junior engineers through mentorship and problem solving. In contrast, LLM-based multi-agent systems instantiate scalable junior engineer agents that inherit workflows, perform specialized tasks, and coordinate under senior engineers’ oversight. As these systems evolve toward self-planning and adaptive learning, fully agentic hierarchies—with senior engineer agent directing junior engineer agents—could transform engineering into a continuously improving, highly efficient practice.

1 INTRODUCTION

Structural engineering, the discipline responsible for designing and building of structures including buildings, bridges, and other infrastructure, has shaped human civilization for millennia—from pyramids and aqueducts to skyscrapers and transportation networks (Saouma, 2010; Addis, 2007). What began as intuition and craftsmanship has become a science grounded in mechanics, materials, and computation (Salvadori, 1990; Allen, 2001). Today, engineers translate ambiguous requirements into models, simulate loading scenarios with specialized software (e.g., OpenSees, SAP2000,

054 Abaqus/CAE) (McKenna, 2011; Pasticier et al., 2008; Wang & Melly, 2018), and verify compliance
 055 against extensive building codes (National Research Council of Canada, 2020; Canadian Institute of
 056 Steel Construction, 1992; Reynolds et al., 2007; Stalnaker & Harris, 1997). Yet despite underpinning
 057 a multi-trillion-dollar global sector (Barbosa et al., 2017), structural engineering remains among
 058 the least digitalized industries, hindered by fragmented workflows, manual knowledge transfer, and
 059 coordination bottlenecks (ASCE, 2019; Agarwal et al., 2016; KPMG, 2019). The result is inefficiency,
 060 cost overruns, and lost opportunities for sustainability and resilience.

061 In parallel, large language models (LLMs) have transformed AI. Scaling transformer-based architec-
 062 tures (Vaswani et al., 2017) to billions of parameters yields broad generalization (Bommasani et al.,
 063 2021), predictable scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), and emergent competen-
 064 cies suggestive of general intelligence (Bubeck et al., 2023; Feng et al., 2024). Advances such as
 065 chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022) and instruction tuning (Ouyang
 066 et al., 2022) have unlocked strong reasoning, while frontier models—GPT-4o (OpenAI, 2024), GPT-5
 067 (OpenAI, 2025), Qwen-3 (Yang et al., 2025), DeepSeek-R1 (Guo et al., 2025a)—achieve unprece-
 068 dented performance in math, code, and tool use. The trajectory has shifted toward *agents*: LLM
 069 systems that plan, invoke tools, and coordinate tasks (Yao, 2025; Silver & Sutton, 2025). Frameworks
 070 such as ReAct (Yao et al., 2023b), Tree of Thoughts (Yao et al., 2023a), Voyager (Wang et al., 2023),
 071 and StateFlow (Wu et al., 2024) externalize reasoning into structured workflows, enabling deployment
 072 in domains that are procedural, codified, and tool-centric.

073 Structural engineering tasks, particularly structural design,
 074 exemplifies a domain where tasks are verbalizable, proce-
 075 dural, and tool-centric: requirements map to models, load
 076 cases to simulations, and code clauses to compliance checks.
 077 Single LLMs can aid structural analysis (Liang et al., 2025a),
 078 yet their accuracy collapses when tasks demand multi-tool
 079 calls or chained subtasks, especially with complex geome-
 080 tries, underscoring the need for a resilient multi-agent frame-
 081 work (see Figure 2). We present **MASSE**, a *multi-agent*
 082 *system* tailored to structural design tasks. MASSE opera-
 083 tionalizes professional workflows by assigning specialized
 084 LLM agents to distinct roles—*Analyst* (data and code ex-
 085 traction), *Engineer* (modeling and limit-state checks), and
 086 *Manager* (coordination and final adequacy decision). These
 087 roles are orchestrated through the AutoGen (Wu et al., 2023),
 088 supported by structured memory for persistent analysis data.
 089 By embedding FEM solvers and code documents directly in
 090 the loop, MASSE achieves stable tool-augmented reasoning,
 091 formal safety verification, and reduces hours of expert itera-
 092 tion to minutes. Therefore, we hypothesize that such agent-driven workflows can generalize beyond
 093 structural engineering to other domains where tasks are verbalizable, procedural, and tool-mediated
 094 (see Figure 1). Our contributions are:

- 094 • We pioneer an LLM-based multi-agent system that mirrors end-to-end structural design workflows
 095 with explicit safety and verification loops.
- 096 • We introduce a new dataset and case studies grounded in real-world problems, demonstrating
 097 automation of complex, tool-mediated tasks.
- 098 • We demonstrate substantial reductions in expert time while preserving accuracy, supporting the
 099 broader thesis that verbalizable, tool-centered professional workflows can be automated at scale.

102 2 RELATED WORK

104 2.1 LLMs IN CIVIL ENGINEERING APPLICATIONS

105 LLMs have been extensively applied in Civil Engineering and related domains (Xie et al., 2025). For
 106 instance, natural language-to-code frameworks have advanced structural optimization (Qin et al.,
 107 2024), prompting and in-context learning have enhanced structural analysis pipelines (Liang et al.,

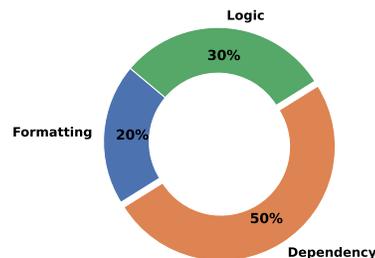


Figure 2: **Failure Modes of Single-Agent.** We evaluate the same structural engineering problem over 10 trials, comparing single-agent and multi-agent setups. The single-agent failed in all 10 trials, with error distributed as shown in the figure. In contrast, our multi-agent framework succeeded in every trial. See details in Appendix A.

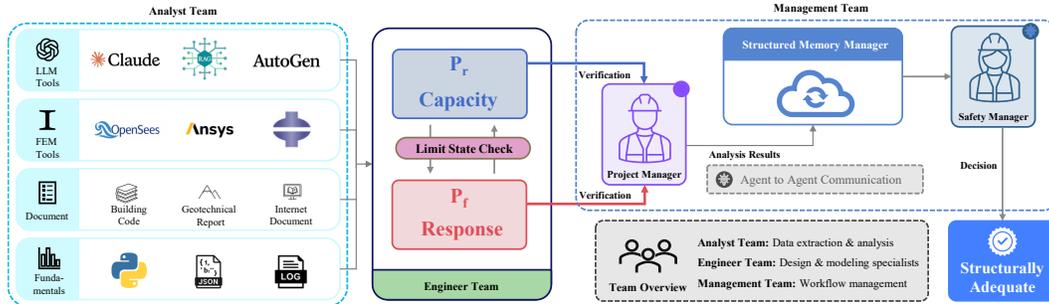


Figure 3: *MASSE* Overall Framework. The **Analyst Team** combines four layers of tools (LLMs, FEM solvers, engineering documents, and fundamentals) to enable multi-agent collaboration. The **Engineer Team** performs limit state verification ($\text{Response} < \text{Capacity}$), while the **Project Manager** coordinates workflows and the **Structural Memory** stores analysis data. The **Safety Manager** conducts the final adequacy check. Three specialized teams are shown: **Analyst** (data extraction and analysis), **Engineer** (design and verification), and **Management** (coordination and decision-making).

2025a; Avila et al., 2025), and multi-agent router systems have improved foundation design automation (Youwai et al., 2025). LLM-driven frameworks further advanced code-compliant reinforced concrete design (Chen & Bao, 2025), LLM-based agents enabled reliable and robust automation of structural analysis (Liu et al., 2025), and LLM-based BIM authoring systems streamlined modeling tasks (Du et al., 2025; Deng et al., 2025). Multi-agent collaboration has also been applied to Ultra-High Performance Concrete (UHPC) design, enabling knowledge-guided and data-driven material development (Guo et al., 2025b). Vision-language systems contributed to safety and compliance through ergonomic risk assessment (Fan et al., 2024), structured hazard detection (Adil et al., 2025), and RAG-based engineering code consultation fine-tuned on NBCC data (Joffe et al., 2025; Aqib et al., 2025). DrafterBench (Li et al., 2025a) benchmarks such LLM-driven automation in civil engineering tasks.

2.2 MULTI-AGENT SYSTEMS WITH LLMs

Recent advancements in multi-agent systems (MAS) have demonstrated the potential of LLM-powered agents to collaborate effectively across domains such as chemistry (Tang et al., 2025), healthcare (Schmidgall et al., 2024), and finance (Xiao et al., 2024), where specialized roles improve domain-specific performance. In software engineering, frameworks like ChatDev (Qian et al., 2023), HyperAgent (Phan et al., 2024), MetaGPT (Hong et al., 2023), and Magentic-One (Fourney et al., 2024) coordinate agents for design, coding, and testing, while general-purpose infrastructures such as AutoGen (Wu et al., 2023) and AppWorld (Trivedi et al., 2024) provide flexible environments for orchestration and benchmarking. At the same time, researchers have begun systematically analyzing MAS limitations, highlighting failure modes such as misaligned objectives and verification bottlenecks (Cemri et al., 2025; Microsoft Corporation, 2025), and stressing the need for rigorous benchmarking practices that jointly optimize accuracy and efficiency (Kapoor et al., 2024). Complementary to these system-level advances, efforts to enhance the efficiency and interpretability of LLMs themselves include model compression (Lin et al., 2024; Kumar et al., 2024; Liang et al., 2025b; Shen et al., 2025d), inference acceleration techniques (Shen et al., 2025b;c;a; Jiang et al., 2025; Shao et al., 2025), and theoretical analyses of their representational power (Chen et al., 2025a;b; Liang et al., 2025c).

3 SYSTEM ROLE DESIGN

In this section, we introduce the specific agent role design of our *MASSE* framework. We clearly define roles and specific goals for LLM agents, enabling complex structural engineering tasks to be efficiently segmented into smaller, manageable components. Structural engineering is inherently multifaceted, requiring diverse inputs, specialized knowledge, and coordinated expertise, such that professional practice relies heavily on multidisciplinary teams to collaboratively address complex, high-stakes decisions (Wolff & Luckett, 2013; Connor & Faraji, 2016; Zhang et al., 2021; Awomolo

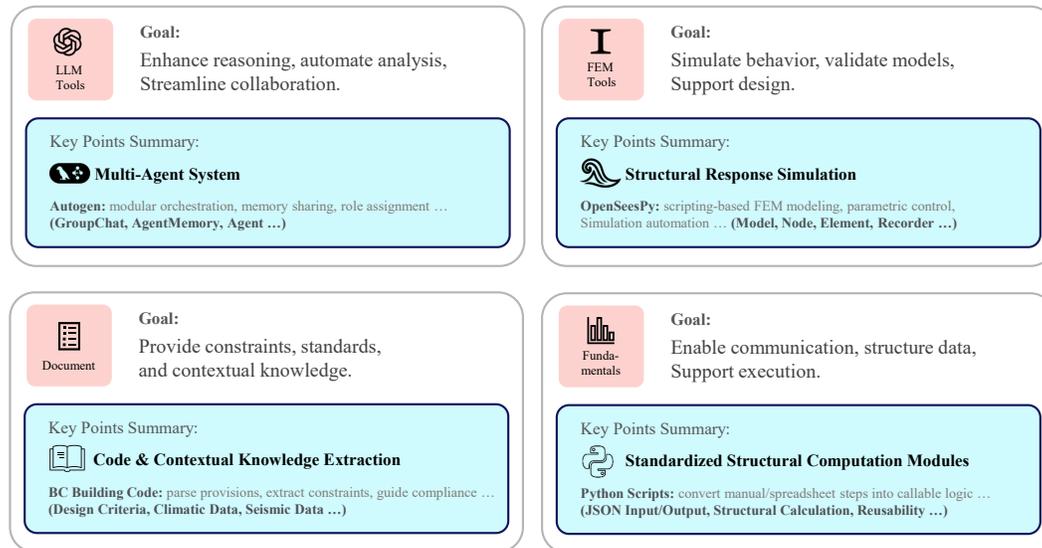


Figure 4: MASSE Analyst Team with Tool Integration

et al., 2019; Chen & Liew, 2002; Rocha et al., 2022), a workflow that LLM agents are well positioned to replicate and automate.

3.1 TRADITIONAL WORKFLOW

In conventional structural engineering practice, the design process is carried out through a sequence of tightly coupled steps, beginning with consulting building codes to obtain site-specific seismic parameters or climatic data. Engineers then collect geometric information from site measurements or architectural drawings and determine applied loads—including dead loads, live loads, and other types of loads—using code-based occupancy provisions and spreadsheets or rule-based calculations. With these inputs, they evaluate the load-bearing capacities of structural members from section dimensions and material properties, and use either hand calculations or finite element models to simulate structural responses under multiple load combinations. If requirements are not met, the workflow must be repeated with adjusted assumptions or revised sections. For specialized problems such as warehouse racking systems, this process may take several hours per rack, and with multiple racks per site, the workload becomes extensive. Overall, the traditional workflow is time-consuming, error-prone, and difficult to scale, motivating the need for automation frameworks such as MASSE.

3.2 OUR SYSTEM

Reflecting this practical organizational model, MASSE (Figure 3) introduces three distinct agent teams within a simulated structural engineering consultancy environment: Analyst Team, Engineer Team and Management Team. For comprehensive descriptions of the system instructions assigned to each agent, see Appendix B. Each agent is assigned a unique role, goal, and set of constraints, and is further equipped with predefined contexts and specialized tools aligned with these responsibilities. We will then introduce how MASSE organizes these agent roles into the following structured teams. More detailed descriptions could be found in Appendix C.

3.3 ANALYST TEAM

The Analyst Team (Figure 4) automates the preparation of structural engineering data by coordinating specialized agents that extract loading conditions, retrieve information from engineering documentation, execute load determination using rule-based methods, and generate structural models. At a high level, this team transforms unstructured project information and regulatory data into standardized engineering inputs, ensuring that subsequent model, design and verification tasks can be carried out with consistency, efficiency, and scalability.

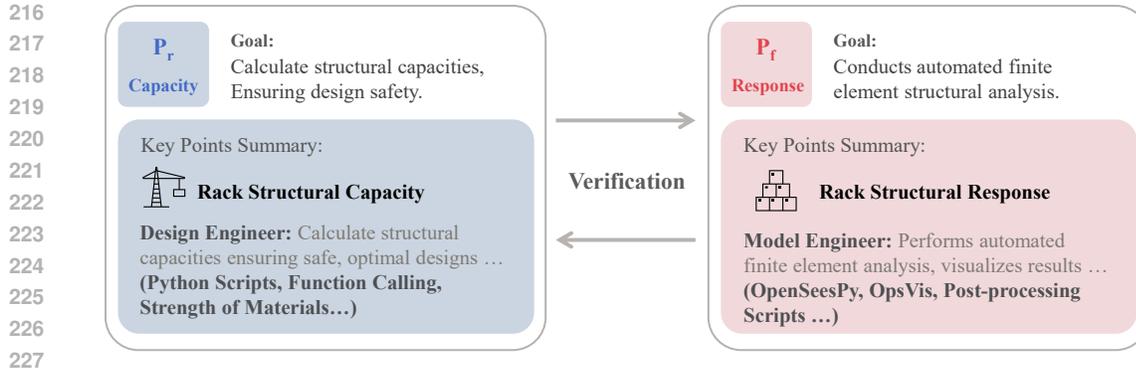


Figure 5: MASSE Engineer Team: Analyze, Design and Verify

3.4 ENGINEER TEAM

The Engineer Team (Figure 5) operationalizes the data prepared by the Analyst Team by conducting structural analysis, structural design, and adequacy verification. In broad terms, this team integrates automated simulations and capacity checks to evaluate structural integrity under prescribed loading conditions, enabling systematic, tool-driven assessment of design safety and structural performance.

3.5 MANAGEMENT TEAM

The Management Team oversees and coordinates the overall MASSE workflow, transforming analytical outputs from the Analyst Team and Engineer Team into authoritative engineering decisions. Broadly speaking, this team manages task allocation, integrates intermediate results, and issues final structural safety conclusions that guide the entire system. As illustrated in Figure 6, the Safety Manager plays a central role within this process by delivering the ultimate adequacy verdict, ensuring that all decisions are consistent with professional safety standards.

4 AGENTIC WORKFLOW DETAILS

4.1 BACKBONE MODELS

In our system, agents are configured to be initialized with powerful LLMs (e.g., GPT-4o, Claude 3.5 Sonnet), or reasoning models (e.g., o4-mini). We also use Embedding models (e.g., text-embedding-3) to support retrieval-augmented generation for accessing building codes and technical documents. Temperatures are set to 0 for GPT-4o, Claude 3.5 Sonnet, and GPT-3.5-turbo, and 1 for o4-mini.

4.2 COMMUNICATION PROTOCOL

MASSE prioritizes structured-format communication over verbose natural language exchanges. Natural language reliance often degrades performance in structural engineering tasks that require extended planning horizons, as verbose dialogue can overflow context windows and obscure key details. To address these issues, MASSE adopts structured communication paradigms, where each agent’s operational state is defined through formalized protocols, such as JSON-based input–output schemas and turn-level state tracking mechanisms. This ensures concise, verifiable outputs while reducing redundancy and distortion.

4.3 AGENT INTERACTIONS

We design agent interactions around structured artifacts that encode analytical results, ensuring traceability and systematic integration. We enforce predefined protocols with input–output formats, schemas, and validation rules to eliminate ambiguity and enable orchestration. In this pipeline, we let analysts convert noisy problem descriptions and unstructured documents into structured memory, engineers transform them into deterministic outputs such as finite element analysis results and capacity

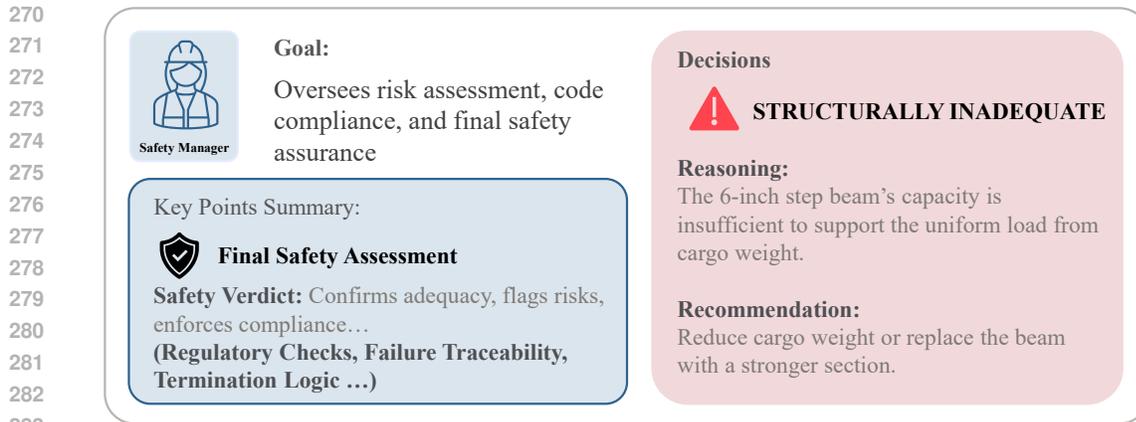


Figure 6: MASSE’s Safety Manager’s Decision-Making Process

checks, and management roles validate outputs against codes before authorizing final decisions. When inconsistencies occur, we allow brief natural language exchanges to recover missing information, but all outcomes are distilled back into structured form to preserve accountability. We log every exchange for auditing and debugging, and the overall framework mirrors professional engineering workflows where domain-specific expertise is coordinated through standardized communication. A detailed trajectory of a racking system case is provided in Appendix D.

5 EXPERIMENTS

5.1 SIMULATION SETUP

We evaluate MASSE in a racking system design scenario (Castiglioni et al., 2016), a representative task in structural engineering where engineers must determine allowable loads and ensure compliance with seismic safety requirements. This setup integrates core engineering steps—data retrieval, load transformation, structural modeling, analysis, and design—directly aligning with the functionality of MASSE agents. Full details of the simulation configuration are provided in Appendix E.1.

5.2 DATASET

We constructed a domain-specific dataset consisting of one hundred different levels of difficulty, each paired with validated ground-truth solutions, enabling robust evaluation reflective of real-world engineering challenges. To comply with privacy constraints, the released dataset is reorganized but faithfully mirrors the original cases, ensuring reproducibility of system performance. Each sample contains a natural language problem description, intermediate reasoning steps, and final results used as evaluation criteria. Notably, loading reports are mandated and must be certified by structural engineers in earthquake-prone regions such as British Columbia and California, underscoring the practical significance of this scenario. Our modular design allows straightforward adaptation to many structural engineering tasks, making them broadly applicable as plug-and-play components. An illustrative racking system problem and its structural scheme are provided in Appendix F.

5.3 EVALUATION METRICS

To benchmark MASSE against real-world structural engineering tasks, we design novel evaluation metrics aligned with the core agent roles: Structural Analysis Agent Benchmark (SAAB), Structural Design Agent Benchmark (SDAB), Loading Agent Benchmark (LAB), and the holistic Multi-Agent Structural Engineering Benchmark (MASEB). Each benchmark evaluates the ability to convert natural language inputs into structured analyses, tool executions, and safety decisions, while MASEB additionally incorporates system cost and runtime to balance efficiency and performance. To ensure these benchmarks reflect professional practice, we base our evaluation on data reorganized from real-world projects in British Columbia, Canada. This design allows the metrics to capture both

technical accuracy and the fidelity with which agents replicate the workflow of a consulting firm. Detailed rubrics, task specifications, and evaluation protocols are provided in Appendix E.2.

6 RESULTS AND ANALYSIS

6.1 MAIN RESULTS

6.1.1 PERFORMANCE COMPARISON

Table 1 summarizes MASSE performance across four benchmarks under different LLM backends. Bold numbers denote the best model on each benchmark, while underlined numbers indicate the second best. Each problem was evaluated over ten independent trials, and a total of one hundred traces were collected to assess the overall system performance. Among language models, *Claude 3.5 Sonnet* delivers the strongest overall consistency, leading in SDAB (89.2) and remaining competitive elsewhere, while *GPT-4o* achieves the highest LAB score (98.1) but trails slightly in SAAB and MASEB. By contrast, *GPT-3.5-turbo* underperforms (e.g., 64.6 in SDAB, 67.7 in MASEB).

Table 1: Performance Comparison of MASSE under Different LLMs. The best score per benchmark is in bold, the second-best is underlined. Avg. is reported across four benchmarks.

Models	SAAB	SDAB	LAB	MASEB	Avg.
Language Models					
<i>Claude 3.5 Sonnet</i>	<u>87.0</u>	<u>89.2</u>	94.8	<u>85.6</u>	<u>89.2</u>
<i>GPT-3.5-turbo</i>	71.4	64.6	90.5	67.7	73.6
<i>GPT-4o</i>	85.6	87.4	98.1	82.7	88.5
Reasoning Models					
<i>o4-mini</i>	96.6	91.4	<u>93.8</u>	94.7	94.1

Reasoning model *o4-mini* outperforms across three of four benchmarks, with peak results in SAAB (96.6), SDAB (91.4), and MASEB (94.7), and only narrowly behind *GPT-4o* in LAB (93.8 vs. 98.1). This balance highlights the intelligence and robustness of reasoning models, showing that reasoning yields higher reliability across heterogeneous tasks. Overall, the results indicate that (1) reasoning models like *o4-mini* provide the most stable and generalizable performance, and (2) among standard LLMs, large-scale models such as *Claude 3.5 Sonnet* and *GPT-4o* remain essential for high-fidelity structural engineering workflows.

6.1.2 COST ANALYSIS

Figure 7a illustrates a clear performance–efficiency trade-off. *o4-mini* achieves the best overall score but with the heaviest computational cost—both in runtime and token consumption. *Claude 3.5 Sonnet* performs slightly worse while also incurring substantial token usage and longer runs. *GPT-4o* offers a balanced middle ground, delivering strong accuracy at a moderate cost. In contrast, *GPT-3.5-turbo* is the most economical and fastest, though its performance lags behind the others. These patterns suggest that MASSE benefits from stronger backbones when quality is paramount, but real-world deployment must balance performance with latency and budget.

6.1.3 RUNTIME ANALYSIS

In this experiment, we vary the maximum number of agent-to-agent communication rounds, which is set to 1 to 4. Allowing more rounds of communication enables agents to exchange additional information, identify and fill in missing details, refine intermediate reasoning steps, and apply self-correction before producing final outputs. The evaluation is performed on ten representative problems selected from MASEB. To mitigate randomness in model behavior and ensure robust measurement, each problem instance is independently repeated ten times.

The runtime analysis (Figure 7b) demonstrates the fundamental trade-off in multi-agent coordination. As the number of agent-to-agent communication rounds increases from one to four, runtime rises steadily—from about 20 seconds at a single round to nearly 70 seconds at four rounds—reflecting the

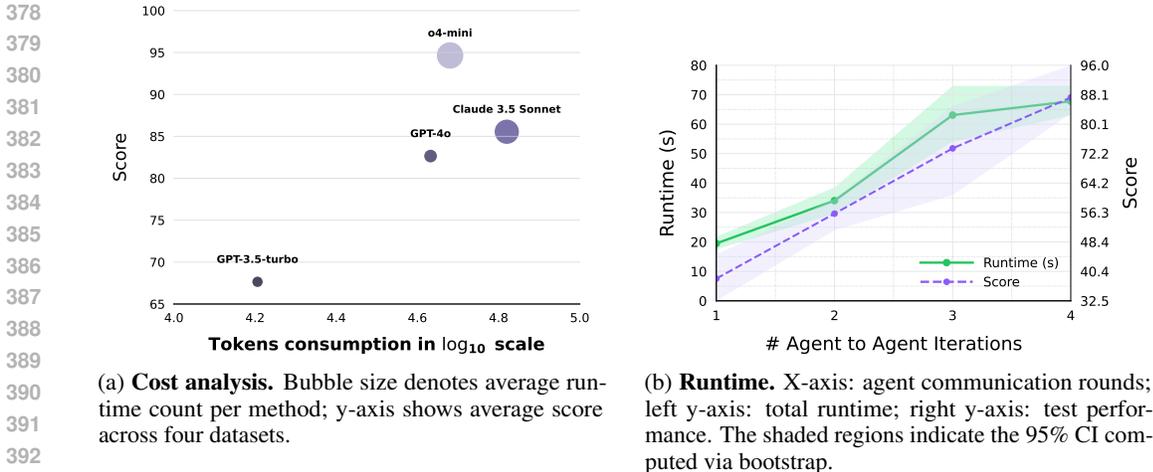


Figure 7: Results Analysis: (a) cost–performance trade-off and (b) runtime–performance relationship.

additional overhead of iterative dialogue. However, this increase in computational effort correlates with consistent improvements in system score, which grows from below 40 to nearly 90. These results highlight a key feature of multi-agent systems: while deeper rounds of communication impose latency, they also enable richer reasoning and more reliable outputs.

6.2 HUMAN EVALUATION

To empirically evaluate MASSE’s efficiency, we conducted a comparative study with 11 experienced structural engineers, each independently completing a standardized racking system design task. The conventional workflow—dependent on fragmented software, manual data retrieval, repetitive entry, and interpretive effort—required an average of 132 minutes per task. By contrast, MASSE, powered by GPT-4o, automated these processes and reduced the average completion time to approximately two minutes, a reduction of over 98%. This result demonstrates a marked improvement in both productivity and operational reliability within professional practice. In consulting environments, executing tasks at a fraction of the traditional time and cost (on the order of 1/100) is likely to enhance profit margins, streamline organizational workflows, and redefine competitive positioning. More broadly, firms that strategically adopt and effectively integrate AI systems such as MASSE may not only secure substantial advantages in the engineering sector, but also be positioned to shape new paradigms of practice, potentially influencing how design, safety, and efficiency are balanced across the built environment.

6.3 ABLATION STUDY

Table 2: Ablation study on agent components: agent memory M and JSON format J .

Agent Component	SAAB	SADB	LAB	MASEB	Avg.
None	52.5	65.9	81.6	47.1	61.8
+ M	55.9	69.8	89.6	50.6	66.5
+ J	56.6	67.2	84.1	50.1	64.5
+ M, J	85.6	87.4	98.1	82.7	88.5

Agent memory and structured input–output are increasingly recognized as common practices and active research directions in the design of multi-agent systems (Xiong et al., 2025; Li et al., 2025b; Fan et al., 2025). To assess the effect of individual agent components, we performed an ablation study across four benchmarks (Table 2). The baseline configuration without memory or structured I/O yields the lowest scores in all settings. Incorporating multi-agent memory (M) improves performance by preserving intermediate reasoning traces across turns, with notable gains in LAB (81.6 \rightarrow 89.6).

432 Similarly, enforcing JSON-based input–output constraints (J) provides consistent benefits by reducing
433 ambiguity in communication, especially in SAAB and SADB. The combination of both components
434 ($+M, J$) achieves the strongest results overall, setting the best score on every benchmark. These
435 findings indicate that memory and structured I/O constraints are crucial: memory enhances contextual
436 continuity, while JSON formalization enforces precise data exchange.

437 438 439 7 DISCUSSION

440
441 **Transparency.** Reliability and accountability are prerequisites for engineering adoption. MASSE
442 is therefore designed with verifiable processes rather than opaque autonomy. Every artifact is logged
443 with explicit reasoning traces, giving practitioners immediate access to the system’s decision path and
444 enabling systematic validation. Deterministic outputs further support reproducible quality checks,
445 while localized error handling ensures that small mistakes do not propagate into larger failures. By
446 structuring agents to follow well-defined workflows, MASSE effectively plays the role of a junior
447 engineer whose steps are fully auditable—providing transparency and traceability without conceding
448 control to uncontrolled autonomy.

449
450 **Safety.** Because structural engineering involves high-stakes decisions, ultimate responsibility must
451 remain with senior practitioners; even small misjudgments can produce catastrophic outcomes.
452 MASSE is designed to augment, not replace, this expertise by compressing the end-to-end review
453 process from roughly two hours to only a few minutes (Section 6.2). This acceleration allows experts
454 to evaluate a wider range of design alternatives and stress scenarios within the same timeframe,
455 thereby improving robustness and resilience of built systems. Equally important, MASSE produces
456 structured intermediate outputs that mirror professional practice, akin to junior engineers drafting
457 reports for senior verification. As AI systems continue to improve, such workflows may increasingly
458 reduce reliance on traditional junior roles, reallocating human attention to the oversight and judgment
459 that matter most for public safety.

460
461 **Real-World Impact.** The efficiency gains of MASSE highlight how domain-specific multi-agent
462 systems can convert the raw capability of frontier LLMs into transformative productivity advantages.
463 Reducing structural design cycles from hours to minutes (Section 6.2) does more than boost users’
464 competitiveness—it has the potential to alter the economics of engineering services by lowering
465 costs, increasing throughput, and enabling firms to take on larger, more complex projects. Freed
466 from routine procedures, engineers can concentrate on creativity, innovation, and safety-critical
467 deliberation. At scale, such reallocation of effort has implications well beyond structural engineering:
468 knowledge-intensive sectors such as architecture, finance, and healthcare could be reorganized
469 around agentic AI collaborators, yielding faster design cycles, more equitable access to expertise,
470 and accelerated progress on global challenges from sustainable infrastructure to resilient urban
471 development. MASSE thus exemplifies how carefully engineered AI systems can bridge the gap
472 between technical advances in LLM orchestration and tangible societal benefit.

473 474 8 CONCLUSION

475
476 We introduced **MASSE**, a multi-agent framework designed specifically for structural engineering, in-
477 tegrating LLM-based agents into end-to-end professional workflows. By coordinating role-specialized
478 agents, MASSE is able to parse technical standards, perform domain-specific computations, and
479 interact seamlessly with engineering scripts and resources. Its structured communication backbone
480 and iterative reasoning cycle allow the system to achieve robust, high-quality outcomes. Safety verifi-
481 cation is embedded through a dedicated management role, and the modular design enables flexible
482 adaptation across a wide range of engineering tasks. Our evaluation on real-world case studies shows
483 that MASSE can reduce expert workload by approximately 98%, cutting the required time from hours
484 to minutes while maintaining both reliability and accuracy. This proof-of-concept demonstrates the
485 feasibility of automating structural engineering workflows without task-specific training. Looking
ahead, future work will emphasize deployment in consulting environments, refinement of agent
specialization, and integration of real-time feedback to support adaptive self-improvement.

486 ETHIC STATEMENT
487

488 **Research and Educational Purpose Only.** The multi-agent system, methods, code, and data
489 described in this paper are developed solely for academic research and educational use. They are
490 not intended, nor should they be relied upon, for direct application in real-world engineering design,
491 construction, or deployment. The authors, their institutions, and any collaborators explicitly disclaim
492 all liability for any consequences, including but not limited to structural deficiencies, under-design,
493 damages, or failures, arising from the use, misuse, or adaptation of the methods, parameters, or
494 formulas presented herein. Any attempt to implement or deploy the system in practice is done entirely
495 at the user’s own risk and responsibility.

496 **Human Evaluation.** All human evaluation conducted in this study was performed exclusively for the
497 purpose of benchmarking and comparative analysis. Data generated by participants is kept strictly
498 confidential, used only within the scope of this research, and will not be applied to any commercial,
499 industrial, or real-world engineering activities.

500 **Privacy and Data Protection.** The dataset employed in this work is derived from production records
501 within structural engineering consultancy practice but has been anonymized, cleaned, and used strictly
502 for academic research. No identifying information, proprietary designs, or sensitive client data are
503 disclosed. All outputs generated by the multi-agent system remain confined to research purposes and
504 will not be released, applied, or repurposed for operational engineering, deployment, or commercial
505 use. This research adheres to applicable data privacy principles and safeguards the rights and interests
506 of all parties involved.

507 LLMs USAGE STATEMENT
508

509 We disclose that LLMs were employed exclusively as auxiliary tools, limited to (i) refining the
510 exposition of the manuscript for clarity and conciseness, and (ii) generating preliminary schematic
511 components used in visualizations to illustrate methodological pipelines.
512

513 REFERENCES
514

- 515 William Addis. *Building: 3000 years of design engineering and construction*, volume 601. Phaidon
516 London, 2007.
- 517 Muhammad Adil, Gaang Lee, Vicente A Gonzalez, and Qipei Mei. Using vision language models for
518 safety hazard identification in construction. *arXiv preprint arXiv:2504.09083*, 2025.
- 519 Rajat Agarwal, Shankar Chandrasekaran, and Mukund Sridhar. Imagining construction’s digital
520 future. *McKinsey & Company*, 24(06):1–13, 2016.
- 521 Edward Allen. *Fundamentals of building construction*, 2001.
- 522 Mohammad Aqib, Mohd Hamza, Qipei Mei, and Ying Hei Chui. Fine-tuning large language models
523 and evaluating retrieval methods for improved question answering on building codes. *arXiv*
524 *preprint arXiv:2505.04666*, 2025.
- 525 ASCE. Confirmation and update: A vision for the future of structural engineering
526 and structural engineers: A case for change. Task committee report, American
527 Society of Civil Engineers, Structural Engineering Institute, April 2019. URL
528 [https://www.asce.org/-/media/asce-images-and-files/communities/
529 institutes-and-technical-groups/structural-engineering/
530 documents/sei-2019-vision-task-comittee-report.pdf](https://www.asce.org/-/media/asce-images-and-files/communities/institutes-and-technical-groups/structural-engineering/documents/sei-2019-vision-task-comittee-report.pdf). Accessed 2025-08-
531 16.
- 532 Carlos Avila, Daniel Ilbay, and David Rivera. Human–ai teaming in structural analysis: A model
533 context protocol approach for explainable and accurate generative ai. *Buildings*, 15(17):3190,
534 2025.
- 535 Olaitan Awomolo et al. Exploring multiple disciplines in building design practice. *Enquiry The*
536 *ARCC Journal for Architectural Research*, 16(1):29–45, 2019.

- 540 Filipe Barbosa, Jonathan Woetzel, and Jan Mischke. Reinventing construction: A route of higher
541 productivity. Technical report, McKinsey Global Institute, 2017.
- 542
- 543 Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx,
544 Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportuni-
545 ties and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
- 546
- 547 Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar,
548 Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio
549 Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4.
550 *arXiv preprint arXiv:2303.12712*, 2023.
- 551 Canadian Institute of Steel Construction. *Handbook of Steel Construction*. Canadian Institute of
552 Steel Construction, Willowdale, Ontario, 1992.
- 553
- 554 Carlo Andrea Castiglioni et al. *Seismic behavior of steel storage pallet racking systems*. Springer,
555 2016.
- 556
- 557 Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt
558 Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm
559 systems fail? *arXiv preprint arXiv:2503.13657*, 2025.
- 560
- 561 Jinxin Chen and Yi Bao. Multi-agent large language model framework for code-compliant automated
562 design of reinforced concrete structures. *Automation in Construction*, 177:106331, 2025.
- 563
- 564 Wai-Fah Chen and JY Richard Liew. *The civil engineering handbook*. Crc Press, 2002.
- 565
- 566 Yifang Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. The computational limits
567 of state-space models and mamba via the lens of circuit complexity. In *The Second Conference
568 on Parsimony and Learning (Proceedings Track)*, 2025a. URL [https://openreview.net/
569 forum?id=bImLLT3r62](https://openreview.net/forum?id=bImLLT3r62).
- 570
- 571 Yifang Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. Universal approximation of
572 visual autoregressive transformers. *arXiv preprint arXiv:2502.06167*, 2025b.
- 573
- 574 Jerome J Connor and Susan Faraji. *Fundamentals of structural engineering*. Springer, 2016.
- 575
- 576 Zihan Deng, Changyu Du, Stavros Nousias, and André Borrmann. Bimgent: Towards autonomous
577 building modeling via computer-use agents, 2025. URL [https://arxiv.org/abs/2506.
578 07217](https://arxiv.org/abs/2506.07217).
- 579
- 580 Changyu Du, Sebastian Esser, Stavros Nousias, and André Borrmann. Text2bim: Generating
581 building models using a large language model-based multi-agent framework, 2025. URL [https:
582 //arxiv.org/abs/2408.08054](https://arxiv.org/abs/2408.08054).
- 583
- 584 Chao Fan, Qipei Mei, Xiaonan Wang, and Xinming Li. Ergochat: a visual query system for the
585 ergonomic risk assessment of construction workers. *arXiv preprint arXiv:2412.19954*, 2024.
- 586
- 587 Shiqing Fan, Xichen Ding, Liang Zhang, and Linjian Mo. Mcptoolbench++: A large scale ai agent
588 model context protocol mcp tool use benchmark. *arXiv preprint arXiv:2508.07575*, 2025.
- 589
- 590 Tao Feng, Chuanyang Jin, Jingyu Liu, Kunlun Zhu, Haoqin Tu, Zirui Cheng, Guanyu Lin, and
591 Jiaxuan You. How far are we from agi: Are llms all we need? *Transactions on Machine Learning
592 Research*, 2024.
- 593
- 594 Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner,
595 Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist
596 multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.
- 597
- 598 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
599 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
600 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

- 594 Pengwei Guo, Zhan Jiang, Weina Meng, and Yi Bao. Multi-agent collaboration for knowledge-guided
595 data-driven design of ultra-high-performance concrete (uhpc) incorporating solid wastes. *Cement*
596 *and Concrete Composites*, pp. 106230, 2025b.
- 597 Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza
598 Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al.
599 Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.
- 600
601 Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang,
602 Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent
603 collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4):6, 2023.
- 604
605 Ting Jiang, Yixiao Wang, Hancheng Ye, Zishan Shao, Jingwei Sun, Jingyang Zhang, Zekai Chen,
606 Jianyi Zhang, Yiran Chen, and Hai Li. Sada: Stability-guided adaptive diffusion acceleration.
607 *arXiv preprint arXiv:2507.17135*, 2025.
- 608
609 Isaac Joffe, George Felobes, Youssef Elgouhari, Mohammad Talebi Kalaleh, Qipei Mei, and Ying Hei
610 Chui. The framework and implementation of using large language models to answer questions
611 about building codes and standards. *Journal of Computing in Civil Engineering*, 2025. ISSN
0887-3801. doi: 10.1061/JCCEE5.CPENG-6037.
- 612
613 Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child,
614 Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models.
CoRR, abs/2001.08361, 2020. URL <https://arxiv.org/abs/2001.08361>.
- 615
616 Sayash Kapoor, Benedikt Stroebel, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents
617 that matter. *arXiv preprint arXiv:2407.01502*, 2024.
- 618
619 Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large
620 language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:
22199–22213, 2022.
- 621
622 KPMG. Global construction survey 2019: Future-ready index – leaders and followers in the
623 engineering & construction industry. Technical report, KPMG International Cooperative, April
624 2019. URL [https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2019/04/
625 global-construction-survey-2019.pdf](https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2019/04/global-construction-survey-2019.pdf). Accessed 2025-08-16.
- 626
627 Tanishq Kumar, Zachary Ankner, Benjamin F Spector, Blake Bordelon, Niklas Muennighoff, Man-
628 sheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision.
arXiv preprint arXiv:2411.04330, 2024.
- 629
630 Yinsheng Li, Zhen Dong, and Yi Shao. Drafterbench: Benchmarking large language models for tasks
631 automation in civil engineering, 2025a. URL <https://arxiv.org/abs/2507.11527>.
- 632
633 Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei
634 Yang, Chunyu Li, Qingchen Yu, et al. Memos: A memory os for ai system. *arXiv preprint
635 arXiv:2507.03724*, 2025b.
- 636
637 Haoran Liang, Mohammad Talebi Kalaleh, and Qipei Mei. Integrating large language models for
638 automated structural analysis. *arXiv preprint arXiv:2504.09754*, 2025a.
- 639
640 Yingyu Liang, Jiangxuan Long, Zhenmei Shi, Zhao Song, and Yufa Zhou. Beyond linear approxi-
641 mations: A novel pruning approach for attention matrix. In *The Thirteenth International Confer-
642 ence on Learning Representations*, 2025b. URL [https://openreview.net/forum?id=
643 sgbI8Pxwie](https://openreview.net/forum?id=sgbI8Pxwie).
- 644
645 Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Yufa Zhou. Looped relu mlps may be
646 all you need as programmable computers. In *The 28th International Conference on Artificial
647 Intelligence and Statistics*, 2025c.
- 648
649 Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan
650 Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for
651 on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:
652 87–100, 2024.

- 648 Jiachen Liu, Ziheng Geng, Ran Cao, Lu Cheng, Paolo Bocchini, and Minghui Cheng. A large
649 language model-empowered agent for reliable and robust structural analysis. *arXiv preprint*
650 *arXiv:2507.02938*, 2025.
- 651 Frank McKenna. Opensees: a framework for earthquake engineering simulation. *Computing in*
652 *Science & Engineering*, 13(4):58–66, 2011.
- 654 Microsoft Corporation. Taxonomy of failure modes in agentic ai sys-
655 tems: Whitepaper. Technical report, Microsoft, 2025. URL [https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/
656 microsoft/final/en-us/microsoft-brand/documents/
657 Taxonomy-of-Failure-Mode-in-Agentic-AI-Systems-Whitepaper.pdf](https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/final/en-us/microsoft-brand/documents/Taxonomy-of-Failure-Mode-in-Agentic-AI-Systems-Whitepaper.pdf).
- 659 National Research Council of Canada. *National Building Code of Canada 2020*. National Research
660 Council of Canada, Ottawa, Ontario, Canada, 2020. National Building Code of Canada (NBC).
- 662 OpenAI. Hello gpt-4o. <https://openai.com/index/hello-gpt-4o/>, 2024. Accessed:
663 May 13.
- 664 OpenAI. Gpt-5 system card. Technical report, OpenAI, August 2025.
- 666 Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong
667 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow
668 instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022.
- 669 Laurent Pasticier, Claudio Amadio, and Massimo Fragiaco. Non-linear seismic analysis and
670 vulnerability evaluation of a masonry building by means of the sap2000 v. 10 code. *Earthquake*
671 *engineering & structural dynamics*, 37(3):467–485, 2008.
- 673 Huy Nhat Phan, Tien N Nguyen, Phong X Nguyen, and Nghi DQ Bui. Hyperagent: Generalist
674 software engineering agents to solve coding tasks at scale. *arXiv preprint arXiv:2409.16299*, 2024.
- 675 Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen,
676 Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. *arXiv*
677 *preprint arXiv:2307.07924*, 2023.
- 679 Sizhong Qin, Hong Guan, Wenjie Liao, Yi Gu, Zhe Zheng, Hongjing Xue, and Xinzheng Lu.
680 Intelligent design and optimization system for shear wall structures based on large language
681 models and generative artificial intelligence. *Journal of Building Engineering*, 95:109996, 2024.
- 682 Charles E Reynolds, James C Steedman, and Anthony J Threlfall. *Reinforced concrete designer’s*
683 *handbook*. CRC Press, 2007.
- 685 Brett Rocha, Aaron Hill, Nathan Hedgecock, Skylar Franz, Morgan Ernst, and Mark Sallot. Evidence
686 of the benefits of interdisciplinary engineering teams: Incorporating systems engineering into civil
687 engineering design. In *2022 ASEE Annual Conference & Exposition*, 2022.
- 688 Mario Salvadori. *Why buildings stand up: The strength of architecture*. WW Norton & Company,
689 1990.
- 691 Victor E Saouma. Structural engineering, analysis and design. *Lecture Notes*, 2010.
- 692 Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor.
693 Agentclinic: a multimodal agent benchmark to evaluate ai in simulated clinical environments.
694 *arXiv preprint arXiv:2405.07960*, 2024.
- 695 Zishan Shao, Yixiao Wang, Qinsi Wang, Ting Jiang, Zhixu Du, Hancheng Ye, Danyang Zhuo, Yiran
696 Chen, and Hai Li. Flashsvd: Memory-efficient inference with streaming for low-rank models.
697 *arXiv preprint arXiv:2508.01506*, 2025.
- 698 Xuan Shen, Chenxia Han, Yufa Zhou, Yanyue Xie, Yifan Gong, Quanyi Wang, Yiwei Wang, Yanzhi
699 Wang, Pu Zhao, and Jiuxiang Gu. Draftattention: Fast video diffusion via low-resolution attention
700 guidance. *arXiv preprint arXiv:2505.14708*, 2025a.
- 701

- 702 Xuan Shen, Weize Ma, Yufa Zhou, Enhao Tang, Yanyue Xie, Zhengang Li, Yifan Gong, Quanyi
703 Wang, Henghui Ding, Yiwei Wang, et al. Fastcar: Cache attentive replay for fast auto-regressive
704 video generation on the edge. *arXiv preprint arXiv:2505.14709*, 2025b.
- 705
706 Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Yanyu Li, Yifan Gong, Kai Zhang, Hao Tan, Jason
707 Kuen, Henghui Ding, et al. Lazydit: Lazy learning for the acceleration of diffusion transformers.
708 In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20409–20417,
709 2025c.
- 710 Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Jing Liu, Ruiyi Zhang, Ryan A Rossi, Hao Tan, Tong
711 Yu, Xiang Chen, et al. Numerical pruning for efficient autoregressive models. In *Proceedings of*
712 *the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20418–20426, 2025d.
- 713
714 David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 2025.
- 715
716 Akshit Sinha, Arvinth Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of
717 diminishing returns: Measuring long horizon execution in llms. *arXiv preprint arXiv:2509.09677*,
718 2025.
- 719 Judith Stalnakier and Ernest Harris. *Structural design in wood*. Springer Science & Business Media,
720 1997.
- 721
722 Xiangru Tang, Tianyu Hu, MUYANG YE, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou,
723 Pan Lu, Zhuosheng Zhang, Yilun Zhao, et al. Chemagent: Self-updating library in large language
724 models improves chemical reasoning. *arXiv preprint arXiv:2501.06590*, 2025.
- 725
726 Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank
727 Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of
728 apps and people for benchmarking interactive coding agents. *arXiv preprint arXiv:2407.18901*,
729 2024.
- 730
731 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz
732 Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing*
733 systems, 30, 2017.
- 734
735 Gong-Dong Wang and Stephen Kirwa Melly. Three-dimensional finite element modeling of drilling
736 cfrp composites using abaqus/cae: a review. *The International Journal of Advanced Manufacturing*
737 *Technology*, 94:599–614, 2018.
- 738
739 Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and
740 Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv*
741 *preprint arXiv:2305.16291*, 2023.
- 742
743 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and
744 Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in*
745 *neural information processing systems*, 35:24824–24837, 2022.
- 746
747 Karin Wolff and Kathy Lockett. Integrating multidisciplinary engineering knowledge. *Teaching in*
748 *Higher Education*, 18(1):78–92, 2013.
- 749
750 Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang,
751 Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent
752 conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- 753
754 Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. Stateflow: Enhancing llm
755 task-solving through state-driven workflows. In *First Conference on Language Modeling*, 2024.
- 756
757 Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. Tradingagents: Multi-agents llm financial trading
758 framework. *arXiv preprint arXiv:2412.20138*, 2024.
- 759
760 Hao Xie, Qipei Mei, and Ying Hei Chui. Ai applications for structural design automation. *Automation*
761 *in Construction*, 179:106496, 2025.

756 Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Jiliang Tang, Himabindu Lakkaraju, and Zhen
757 Xiang. How memory management impacts llm agents: An empirical study of experience-following
758 behavior. *arXiv preprint arXiv:2505.16067*, 2025.

759 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang
760 Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*,
761 2025.

762 Shunyu Yao. The second half. <https://ysymyth.github.io/The-Second-Half/>, April
763 2025.

764 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R
765 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In
766 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

767 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.
768 React: Synergizing reasoning and acting in language models. In *International Conference on
769 Learning Representations (ICLR)*, 2023b.

770 Sompote Youwai, David Phim, Vianne Gayl Murcia, and Rianne Clair Onas. Investigating the
771 potential of large language model-based router multi-agent architectures for foundation design
772 automation: A task classification and expert selection study, 2025. URL [https://arxiv.
773 org/abs/2506.13811](https://arxiv.org/abs/2506.13811).

774 Cheng Zhang, Jinwoo Kim, JungHo Jeon, Jinding Xing, Changbum Ahn, Pingbo Tang, and Hubo
775 Cai. Toward integrated human-machine intelligence for civil engineering: An interdisciplinary
776 perspective. *Computing in Civil Engineering 2021*, pp. 279–286, 2021.

777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809

Appendix

A SINGLE- VS TWO-AGENT EXPERIMENT

Structural engineering workflow usually includes long-horizon tasks. Recent study proves that, without chain-of-thought, non-thinking models struggle to chain even two steps in a single turn (Sinha et al., 2025). An obvious solution is to split these steps across two models to execute sequentially. Therefore, we design an experiment to demonstrate that a multi-agent system is necessary for long-horizon tasks in structural engineering. This appendix reports a controlled comparison between two agent systems for generating OpenSeesPy code from natural language problem descriptions (PD): (1) a two-agent collaborative system (AgentA + AgentB) and (2) a single-agent end-to-end system (AgentAB). The primary goal of this experiment is to evaluate the execution success rate.

For the two-agent system, AgentB extracts parameters from PD into JSON and computes section properties, which are then passed to AgentA for generating complete OpenSeesPy scripts. In contrast, the single-agent system (AgentAB) directly converts PD into executable scripts in one step. To ensure fairness, both systems employ the same model (GPT-4o) under identical environments (`temperature=0.0`). The PD itself is formulated as a two-dimensional structural analysis task involving a two-step procedure. In the first step, material and geometric properties are extracted from the problem description, and a Python script is invoked to compute intermediate parameters such as the moment of inertia and cross-sectional area. In the second step, a finite element model is constructed using both the original parameters from the PD and the derived intermediate values. Outputs were standardized to executable Python scripts performing model assembly, load application, finite element analysis via `openseespy`, visualization via `opsviz`, and printing displacements, reactions, and internal forces.

For evaluation, we focus on execution success rates and systematically characterize the associated failure modes. The results clearly indicate that the two-agent system consistently outperformed the single-agent system in execution success rates. Specifically, the single-agent system failed in all ten trials (see Figure 2), primarily due to cascading errors in parsing and modeling logic. By contrast, the two-agent system achieved robust parameter extraction and stable execution across all cases, albeit with slightly higher latency. In summary, the findings highlight the importance of decomposition and specialization. The multi-agent system introduced modest overhead but ensured higher correctness and execution reliability.

B MASSE WORKFLOW: ROLE SPECIFICATION

This appendix outlines the organizational structure of agent roles within MASSE, detailing how specialized teams coordinate to address real-world structural engineering problems. The framework is anchored around three primary groups—the *Analyst Team*, the *Engineer Team*, and the *Management Team*—each responsible for distinct stages of the problem-solving pipeline.

The distribution of responsibilities across these roles draws inspiration from established practices in engineering consultancy, where complex projects are decomposed into well-defined tasks that are either managed individually by an engineer or collaboratively across teams. MASSE reflects this dual perspective: on the one hand, it mirrors the multi-tasking strategies of a single engineer navigating coupled subtasks, while on the other, it replicates the collaborative dynamics observed in professional firms. The system’s design principle is straightforward—once a large, long-horizon problem is partitioned into precise subtasks that can be verbalized with clarity, each subtask becomes tractable for a single LLM-based agent. This transformation allows the originally complex workflow to be executed seamlessly through a multi-agent system. The sections that follow present detailed specifications of these agent roles and demonstrate how their interaction yields transparent, resilient, and domain-grounded engineering outcomes.

Analyst Team Instruction Prompts

1. Loading Analyst

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

```
system_message = ""
Extract building information from racking system description and return as JSON:
```

```
"location": "city, province/state",
"building_type": "racking_system",
"floor_elevations_ft": [list of elevations in feet],
"loads_lbs": [list of loads in pounds],
"dimensions":
"width_ft": number,
"height_ft": number,
"beam_length_ft": number
,
"structural_info": "column and brace specifications"
```

```
Extract exact numerical values from the description.
"""
```

2. Seismic Analyst

```
system_message = "You extract seismic data from building codes. Return only JSON. No
explanations."
```

```
user_message = f""Extract seismic parameters for location from the document below.
```

```
Document:
```

```
context
```

```
Find the exact numerical values for location and return ONLY this JSON:
```

```
"Sa_02": <number>,
"Sa_05": <number>,
"Sa_10": <number>,
"Sa_20": <number>,
"PGA": <number>,
"PGV": <number>
```

```
Rules:
```

- Return ONLY the JSON object
- Use actual numerical values from the document
- If location is not found, return: "error": "City not found"
- No explanations, no text outside JSON""

3. Dynamic Analyst

```
system_message = ""You are a Dynamic Analyst.
```

```
Get data from memory, then call
```

```
calculate_seismic_loads(floor_elevations_ft, loads_lbs, seismic_parameters).""
```

4. Structural Analyst

```
system_message = ""You are a StructuralAnalyst.
```

```
** CRITICAL EXECUTION ORDER **
```

```
STEP 1: Create load application nodes at EXACT required elevations FIRST - these are
MANDATORY and cannot be omitted
```

```
STEP 2: Create all brace connection nodes at their exact coordinates
```

```
STEP 3: Create column nodes for beam-column elements
```

```
STEP 4: Verify all required nodes exist before creating elements
```

```
Provide output strictly as JSON matching the following format (same keys, types,
order). Return only the JSON.
```

```
CRITICAL INSTRUCTIONS:
```

```
1. Generate a structural model based on the geometry description provided.
```

```
2. **ALL COORDINATES MUST BE IN FEET** - Keep coordinates in feet, do NOT convert to
inches.
```

```
3. Forces are in kip, stiffness is in kip/in2.
```

```
4. **LOAD NODES ARE MANDATORY** - You MUST create load application nodes BEFORE any
other nodes.
```

```
5. **CREATE ALL BRACES EXACTLY AS SPECIFIED** - You MUST create truss elements using
the EXACT coordinates provided in the description.
```

```
6. Use sequential node IDs starting from 1.
```

```
7. AVOID duplicate nodes - merge nodes with same coordinates to prevent zero-length
elements.
```

```
8. **BRACE COORDINATES ARE MANDATORY** - For each coordinate pair (x1,y1)-(x2,y2) in
the description, create nodes at those EXACT coordinates and connect them with a truss
element.
```

```
9. **DO NOT MODIFY BRACE COORDINATES** - Use the coordinates exactly as written, do not
change them or infer different positions.
```

```
10. **VERIFY**: Count your truss elements before finishing - must match the required
count exactly.
```

```
11. **UNITS**: Keep all coordinates in feet as provided in the description.
```

```
12. **ELEMENT TYPE CONSISTENCY** - Model braces strictly as 2-node axial-only `truss`
elements; use beam/column elements (e.g., `elasticBeamColumn`) for framing members
where bending is required, keeping section units consistent with stiffness units.
```

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

```

13. **BOUNDARY CONDITIONS FIRST CLASS** - Explicitly define base restraints (e.g., fix
supports with (ux=uy=rz=0) for 2D) before applying any loads; all other DOFs remain
free unless specified.
14. **LOAD PATTERN DISCIPLINE** - Create named load patterns (e.g., Gravity, SeismicX)
and apply forces **only** at the designated LOAD NODES; do not assign moments to
truss-only nodes or DOFs not supported by the element type.
15. **CONSISTENT ORIENTATION** - When defining each element, list the i-end as the node
with smaller x (or smaller y if x ties) to maintain consistent element orientation for
checks, outputs, and post-processing.
16. **POST-BUILD VALIDATION** - After assembly, verify: no zero-length elements, every
mandatory brace is present, each non-load node participates in at least one element,
and reported counts of nodes/elements/load nodes match the expected totals; abort and
correct if any check fails.
17. **DOCUMENT INPUTS CLEARLY** - Record all geometry, section properties, boundary
conditions, and load definitions in comments or separate metadata files to ensure the
structural model is fully reproducible and auditable.
Required JSON format:

"units":
"length": "ft (feet)",
"force": "kip",
"stiffness": "kip/in^2"
,
"materials":
"E": 29000.0
,
"sections":
"column": "A": 0.705, "I": 1.144 ,
"brace": "A": 0.162
,
"nodes": [
// Create nodes for all structural geometry points AND load application points
// Example: "id": 1, "x": 0.0, "y": 0.0
],
"elements": [
// Create beam-column and truss elements based on description
// Include ALL column elements and ALL brace elements as specified
// For braces: Use EXACT coordinates from description, do not modify or infer positions
// Example: "id": 1, "type": "elasticBeamColumn", "nodes": [1, 2], "section":
"column", "matTag": 1, "transfTag": 1
// Example: "id": 2, "type": "truss", "nodes": [3, 4], "section": "brace", "matTag": 1
],
"supports": [
// Fixed supports at base nodes
// Example: "node": 1, "fixity": [1, 1, 1]
],
"loads": [
// Will be populated automatically based on load application nodes
]
"""

```

Figure 8: Instruction prompts for the *Analyst Team*, including Loading, Seismic, Dynamic, and Structural Analysts.

```

Engineer Team Instruction Prompts

1. Design Engineer

system_message = """You are a Design Engineer. Use run_complete_opensees_analysis()
with structural model from memory. One function call completes entire analysis."""

2. Model Engineer

system_prompt_base = """You are a model engineer.
Use run_complete_opensees_analysis() with structural model from memory.
One function call completes entire analysis."""

3. Verification Engineer

system_message = """You are a VerificationEngineer. Use get_analysis_context() to get
all data, then verify_structural_safety(capacities, demands)."""

```

Figure 9: Instruction prompts for the *Engineer Team*: including Design, Model and Verification Engineers.

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

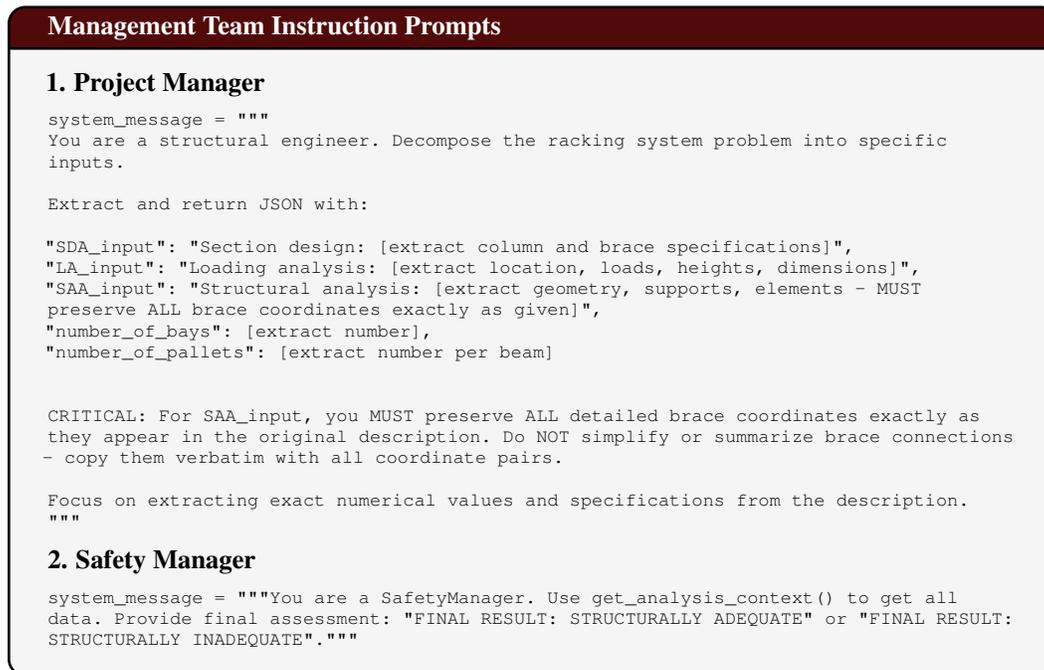


Figure 10: Instruction prompts for the *Management Team*: (1) Project Manager decomposes racking problems into structured inputs for design, loading, and analysis while preserving brace coordinates; (2) Safety Manager performs final adequacy check and outputs the ultimate structural decision.

C MASSE COMPONENT DETAILS

This appendix provides detailed descriptions of all core components within the MASSE framework, expanding on the high-level summaries presented in the main text. Each subsection outlines the specific roles, methodologies, and implementation strategies of the corresponding teams and agents, ensuring reproducibility and technical clarity beyond the main paper’s length constraints.

C.1 ANALYST TEAM

The Analyst Team consists of specialized agents tasked with extracting and analyzing critical engineering parameters essential for structural analysis and design. Each agent addresses a distinct aspect of data processing, ensuring comprehensive and automated preparation necessary for subsequent structural analysis and design phases.

- **Loading Analyst:** Specializes in interpreting detailed structural descriptions, converting them into structured engineering data. Leveraging natural language processing and general-purpose LLMs, the Loading Analyst systematically extracts essential building attributes such as floor elevations, applied loads, geometric dimensions, material specifications, and specific engineering criteria. The extracted information is structured into JSON schemas with standardized units, facilitating efficient downstream analyses.
- **Seismic Analyst:** Dedicated to retrieving and compiling seismic design parameters from authoritative databases and regulatory documents. Utilizing Retrieval-Augmented Generation (RAG) alongside advanced semantic search and vector-based retrieval methods such as FAISS indexing, the Seismic Analyst efficiently locates and extracts key seismic data, including spectral acceleration values, peak ground acceleration, and peak ground velocity. The agent employs PDF parsing, embedding models for semantic accuracy, and structured data storage, ensuring precise geographic-specific retrieval and rapid information availability.
- **Dynamic Analyst:** Responsible for calculating seismic loads by accurately integrating structural characteristics with seismic parameters. This agent employs established structural dynamics

1026 methods, including the equivalent lateral force procedure in line with current regulatory codes.
 1027 The Dynamic Analyst systematically computes base shear forces, lateral load distributions, height-
 1028 dependent amplification factors, and detailed seismic demands at the story level. Computations are
 1029 rigorously validated using numerical methods provided by mathematical libraries such as NumPy,
 1030 ensuring adherence to regulatory standards.

- 1031
- 1032 • **Structural Analyst:** Acts as an advanced structural model generation specialist. Utilizing
 1033 AutoGen’s `ConversableAgent` framework integrated with general-purpose
 1034 LLMs, this agent transforms high-level engineering descriptions into detailed, JSON-
 1035 formatted finite element models compatible with `OpenSeesPy`. Its primary function,
 1036 `generate_structural_model(description)`, processes input from
 1037 `StructuralMemoryManager` using prompts, validating critical load nodes for accurate
 1038 force applications at specified elevations. The agent produces comprehensive structural
 1039 models detailing geometric coordinates, element connectivity, material properties, section
 1040 properties for structural elements (columns and braces), boundary conditions, and load vectors,
 1041 integrating calculated seismic loads. Robust error handling includes JSON parsing and regular
 1042 expression-based content cleaning to ensure uninterrupted analysis. The Structural Analyst
 1043 ensures structural engineering compliance, computational efficiency, and effective cross-agent
 1044 communication through memory management tools.

1044 Collectively, the Analyst Team synthesizes structured engineering data from various sources, including
 1045 detailed textual descriptions and regulatory documentation. Their coordinated efforts provide
 1046 comprehensive, reliable inputs to the Engineer Team, creating a solid foundation for precise, informed
 1047 structural engineering decisions.

1048 C.2 ENGINEER TEAM

1049 The Engineer Team systematically conducts engineering evaluations informed by data provided by
 1050 the Analyst Team. Consisting of specialized agents that focus on structural analysis, structural design,
 1051 and structural adequacy verification, the team performs iterative analyses through tool utilization and
 1052 structured data exchanges, evaluating structural integrity and identifying potential risks.

- 1053
- 1054 • **Design Engineer:** This agent calculates the structural capacities of individual elements such
 1055 as beams, columns, and braces, a critical step for ensuring the structural safety and efficacy of
 1056 proposed designs. Equipped with predefined Python scripts, the agent returns structural capacities
 1057 and associated section properties to the system memory. These results underpin the selection of
 1058 optimal yet safe combinations of structural members in comprehensive structural designs.
- 1059
- 1060 • **Model Engineer:** This agent synthesizes structured data from the Analyst Team, specifically load
 1061 information from the Dynamic Analyst and structural model parameters from the Structural Analyst,
 1062 to execute finite element analyses. Utilizing `OpenSeesPy` and `Opsvis` packages, this agent
 1063 autonomously generates and runs Python-based finite element analysis, subsequently visualizing
 1064 results like internal force distributions and deformation shapes for user review. It also engages
 1065 dedicated Python scripts for automated post-processing, adeptly handling diverse load scenarios
 1066 beyond seismic loads, such as live loads, thus ensuring automated structural analysis workflows.
- 1067
- 1068 • **Verification Engineer:** This agent evaluates structured results provided by the Design Engineer and
 1069 Model Engineer, deploying Python scripts to systematically verify all critical structural behaviors
 1070 including tension, compression, bending moments, torsion, deflection, and rotation. Should any
 1071 structural element fail to meet established performance criteria, the Verification Engineer explicitly
 1072 identifies the deficiency (e.g., inadequate beam strength at a specific floor), clearly documenting
 1073 the structural inadequacies and underlying reasons.

1073 Through this structured, iterative evaluation process, the Engineer Team comprehensively captures
 1074 essential structural and model data. Their meticulous analysis facilitates precise identification of
 1075 structural responses under specific load conditions and the structural capacity of each element, thereby
 1076 providing robust support for informed decision-making by the Management Team.

1077 C.3 MANAGEMENT TEAM

1078 The **Management Team** oversees the operation and coordination of the entire multi-agent system,
 1079 executing definitive engineering decisions guided by detailed analyses from the Analyst Team and the

specialized insights provided by the Engineer Team. They critically evaluate the integrated findings, examining both quantitative outputs and intermediate analytical processes, to make authoritative and conclusive engineering judgments.

The primary responsibilities of the MASSE Management Team include:

- Coordinating the structural analysis workflow through problem decomposition and task distribution.
- Managing data integration and intermediate results processing, ensuring project timelines and quality standards are maintained.
- Conducting final structural safety evaluations and authoritative decision-making based on comprehensive analytical outcomes.
- Providing standardized safety conclusions and managing system termination protocols.

The Management Team imitates the roles of managers and administrative coordinators in consulting engineering firms, illustrating how essential managerial and coordination functions can be systematically modeled within agent-based workflows.

D SPECIFIC CASE TRAJECTORIES

We further present a detailed trace of a representative case trajectory, which illustrates how the multi-agent system progresses through each step of problem-solving. Such traces not only enable us to analyze and debug the system's behavior but also provide valuable insights into potential avenues for optimization and the range of problems the framework can address. By documenting the complete trajectory, the outcomes of the system become highly interpretable. Moreover, presenting key intermediate steps for verification by human engineers significantly enhances the trustworthiness of the system in real-world deployment. An example case trajectory is shown below.

MASSE Analysis Log — Initialization

Step 0: System Initialization

Log Start: ===== MASSE ANALYSIS LOG START =====

Logger: Initialized — all terminal output will be saved to log file.

Database: - Loading existing vector database from [REDACTED PATH] - RAG seismic database validation successful

Agent Registration: 15 functions successfully registered to agents and user_proxy.

Problem Setup: Location: Nanaimo, BC System: 2 bays, 8.0 ft beam length, 3 pallets per beam. Beam: 4 in Z-section. Columns: Steel U-channels (3.079 in × 2.795 in × 0.0787 in, 16.0 ft). Braces: Steel U-channels (1.0 in × 1.0 in × 0.054 in, length 4.3 ft). Geometry: 2 beam-columns from (0,0)→(0,16.0) and (3.5,0)→(3.5,16.0). Bracing: 8 pin-ended trusses connecting specified nodes. Supports: Fixed at (0,0) and (3.5,0). Loads: Applied to left column at 4.0 ft, 8.5 ft, 13.0 ft. Units: Coordinates in ft, forces in kip, stiffness in kip/in².

Runtime Config: - AutoGen version: 0.9.2 - LLM Config: gpt-4o, Temp = 0, Max tokens = 2000 - Runtime logging started (UUID [REDACTED]) - Database path [REDACTED PATH]

Agents Initialized: ProjectManager, DesignEngineer, LoadingAnalyst, SeismicAnalyst, DynamicAnalyst, ModelEngineer, StructuralAnalyst, VerificationEngineer, SafetyManager

System Verification: Completed — all components verified.

Log End: System Initialization finished, structural analysis can proceed.

1134	ProjectManager — Step 1: Problem Decomposition and Setup
1135	Step 1: Problem Decomposition and Setup
1136	User Instruction: - Call <code>split_problem_description()</code> with the full problem
1137	text. - After splitting, call <code>adjust_pallet_weights()</code> using <code>number_of_bays</code> and
1138	<code>number_of_pallets</code> from memory.
1139	Tool Call 1: <code>split_problem_description</code> <i>Input:</i> Problem description text. <i>Output:</i>
1140	Problem split into SAA, SDA, LA inputs. Bays = 2, Pallets = 3.
1141	Tool Call 2: <code>adjust_pallet_weights</code> <i>Input:</i> LA_input, num_bays = 2, num_pallets = 3.
1142	<i>Process:</i> - Loads before update: None - Loads after update: [1875, 1125, 750] (lbs) <i>Output:</i>
1143	Pallet weights adjusted successfully.
1144	Result Summary: - Problem description split into SAA, SDA, LA inputs. - Bays = 2, Pallets =
1145	3. - Pallet weights updated to 1875, 1125, 750 lbs.
1146	Follow-up Tool Calls: - <code>update_saa_input(section_data, load_data)</code>
1147	→ SAA input updated successfully, brace coordinates preserved. -
1148	<code>save_analysis_results(filepath="analysis_results.json")</code> → Results
1149	saved.
1150	DesignEngineer — Step 2: Section Design
1151	Step 2: Section Design
1152	User Instruction: Extract section info from SDA_input and calculate capacities.
1153	Tool Call 1: <code>extract_section_info</code> <i>Input:</i> Description text from SDA_input. <i>Output:</i>
1154	Section info extracted — column, brace, beam.
1155	Tool Call 2: <code>calculate_section_capacities</code> <i>Input:</i> Section info (from previous step).
1156	<i>Output:</i> Section capacities calculated successfully.
1157	Result Summary: - Section info extracted from SDA_input. - Element capacities computed and
1158	stored.
1159	LoadingAnalyst — Step 3: Building Information
1160	Step 3: Building Information
1161	User Instruction: Extract building info from LA_input.
1162	Tool Call: <code>extract_building_info</code> <i>Input:</i> Description text from LA_input. <i>Output:</i>
1163	Building info extracted — racking system, 3 floors, 3 load points.
1164	Result Summary: - Building type: Racking system - Number of floors: 3 - Number of load
1165	points: 3
1166	SeismicAnalyst — Step 4: Seismic Parameters
1167	Step 4: Seismic Parameters
1168	User Instruction: Get seismic parameters for Nanaimo, BC based on location from build-
1169	ing_info.
1170	Tool Call: <code>get_seismic_parameters</code> <i>Input:</i> Location = "Nanaimo, BC". <i>Process:</i> Re-
1171	trieved data using RAG search; 6 relevant document segments found. <i>Output:</i> Seismic parameters
1172	successfully extracted.
1173	Result Summary: - Location: Nanaimo, BC - Sa.0.2 = 1.02 - PGA = 0.446
1174	DynamicAnalyst — Step 5: Load Calculations
1175	Step 5: Load Calculations
1176	User Instruction: Calculate seismic loads using data from memory.
1177	Tool Call 1: <code>get_memory_summary</code> <i>Output:</i> Memory contains 14 non-null entries includ-
1178	ing SAA_input, SDA_input, LA_input, building_info, seismic_parameters, floor_elevations_ft,
1179	loads_lbs, section_data.
1180	Tool Call 2: <code>get_memory_data("floor_elevations_ft")</code> <i>Output:</i> [4.0, 8.5, 13.0]
1181	Tool Call 3: <code>get_memory_data("loads_lbs")</code> <i>Output:</i> [1875, 1125, 750]
1182	
1183	
1184	
1185	
1186	
1187	

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

Tool Call 4: `get_memory_data("seismic_parameters")` *Output:* {Sa_02: 1.02, Sa_05: 0.942, Sa_10: 0.037, Sa_20: 0.328, PGA: 0.446, PGV: 0.684}

Tool Call 5: `calculate_seismic_loads` *Input:* Floor elevations = [4.0, 8.5, 13.0], Loads = [1875, 1125, 750] *Output:* Seismic loads calculated: F1 = 0.395 kip, F2 = 0.504 kip, F3 = 0.514 kip

Result Summary: - Floor elevations and pallet weights retrieved. - Seismic parameters obtained from memory. - Final seismic loads: F1 = 0.395 kip, F2 = 0.504 kip, F3 = 0.514 kip.

ProjectManager — Step 6: Update SAA Input

Step 6: Update SAA Input

User Instruction: Call `update_saa_input()` to merge SAA, section, and load data from memory.

Tool Call 1: `get_memory_data("saa_input")` *Output:* ...

Tool Call 2: `get_memory_data("section_data")` *Output:* JSON containing section properties and capacities for posts, braces, and beams.

Tool Call 3: `get_memory_data("load_data")` *Output:* JSON containing load cases: - Seismic: F1 = 0.395, F2 = 0.504, F3 = 0.514 - Live: F1 = 1.875, F2 = 1.125, F3 = 0.75

Tool Call 4: `update_saa_input(section_data, load_data)` *Process:* - Original SAA_input length: 365 chars - Brace coordinates preserved - Updated SAA_input length: 365 chars *Output:* SAA input updated successfully with section and load data.

Result Summary: - Section and load data retrieved from memory. - SAA input merged successfully. - Structural brace coordinates preserved.

StructuralAnalyst — Step 7: Generate Structural Model

Step 7: Generate Structural Model

User Instruction: Call `generate_structural_model()` using updated SAA input.

Tool Call 1: `get_memory_data("SAA_input_update")` *Output:* Structural analysis description including: - Two elastic beam-columns from (0,0)→(0,16.0) and (3.5,0)→(3.5,16.0) - Fixed supports at (0,0) and (3.5,0) - 8 pin-ended truss braces with specified coordinates

Tool Call 2: `generate_structural_model` *Process:* - Found 8 brace coordinates in description - Floor elevations retrieved from memory: [4.0, 8.5, 13.0] - Required load nodes: (0,4.0), (0,8.5), (0,13.0) - Generated 15 nodes total - Generated 8 truss elements + 2 beam-column elements - Model coordinate ranges: X = 0–3.5, Y = 0–16.0 - Applied loads: F1 = 0.395 kip, F2 = 0.504 kip, F3 = 0.514 kip *Output:* Structural model generated with 15 nodes and 10 elements.

Result Summary: - Structural model created from SAA input. - Geometry, supports, and braces defined correctly. - Seismic loads applied to correct nodes. - Final model: 15 nodes, 10 elements.

ModelEngineer — Step 8: OpenSees Analysis

Step 8: Complete Structural Analysis (OpenSees)

User Instruction: Run `run_complete_opensees_analysis()` using the structural model.

Tool Call 1: `get_memory_summary` *Output:* Memory contains 16 non-null entries, including structural_model, load_data, and section_data.

Tool Call 2: `get_memory_data("structural_model")` *Output:* JSON model with: - Units: ft, kip, kip/in² - Materials: E = 29000 - Sections: column (A = 0.705, I = 1.144), brace (A = 0.162) - 15 nodes, 10 elements (2 elasticBeamColumn, 8 truss) - Supports: fixed at base nodes - Loads: seismic forces at nodes (0,4.0), (0,8.5), (0,13.0)

Tool Call 3: `run_complete_opensees_analysis(model_data)` *Process:* - Generated OpenSees script structural_model_exec.py - Analysis executed successfully - Internal forces saved to internal_forces.json - Applied load combinations: seismic = 3.340, live = 5.625, envelope = 5.625 *Output:* Internal forces extracted and stored.

Result Summary: - OpenSees analysis completed successfully. - Beams: Max tension = 3.34 kip, Max compression = 5.62 kip, Max moment = 7.72 kip-in - Trusses: Max tension = 1.35 kip, Max compression = 1.73 kip - Results saved in memory.

1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

VerificationEngineer — Step 9: Structural Verification

Step 9: Structural Verification

User Instruction: Verify structural safety using all analysis data.

Tool Call 1: `get_analysis_context` *Output:* Context includes: - Seismic parameters (Sa.02 = 1.02, Sa.05 = 0.942, Sa.10 = 0.037, Sa.20 = 0.328, PGA = 0.446, PGV = 0.684) - Structural model JSON (geometry, sections, loads) - Processed forces: • Beams — max tension = 3.34 kip, max compression = 5.625 kip, max moment = 7.72 kip-in • Trusses — max tension = 1.35 kip, max compression = 1.73 kip - Adjusted SAA, SDA, and LA inputs

Tool Call 2: `verify_structural_safety` *Input:* - Capacities: beams and trusses from section design - Demands: seismic loads from nodes *Output:* Safety verification complete — PASS

Result Summary: - Verification executed using capacities and demands. - All elements satisfy safety requirements. - Structure passes safety verification.

SafetyManager — Step 10: Final Safety Assessment

Step 10: Final Safety Assessment

User Instruction: Use `get_analysis_context()` to obtain all data and provide the final assessment: “FINAL RESULT: STRUCTURALLY ADEQUATE” or “FINAL RESULT: STRUCTURALLY INADEQUATE”.

Tool Call 1: `get_analysis_context` *Output:* Context includes: - Seismic parameters (Sa.02 = 1.02, Sa.05 = 0.942, Sa.10 = 0.037, Sa.20 = 0.328, PGA = 0.446, PGV = 0.684) - Structural model JSON with geometry, materials, sections, and loads - Processed forces: • Beams — max tension = 3.34 kip, max compression = 5.62 kip, max moment = 7.72 kip-in • Trusses — max tension = 1.35 kip, max compression = 1.73 kip - Safety evaluation: PASS • Post checks — tension ratio = 0.13, compression ratio = 0.28 • Brace checks — tension ratio = 0.18, compression ratio = 0.34

Final Assessment: FINAL RESULT: STRUCTURALLY ADEQUATE

Result Summary: - Full system context retrieved. - SafetyManager confirmed PASS. - Structure deemed structurally adequate. - MASSE analysis log closed successfully.

E EXPERIMENT DETAILS

E.1 SIMULATION SETUP

We selected a racking system design (Castiglioni et al., 2016) to exemplify typical structural engineering tasks. Racking systems, extensively utilized in warehouses, optimize vertical cargo storage while ensuring structural safety. Structural engineers regularly determine allowable loads and issue certified reports to ensure operational compliance. This scenario effectively integrates core structural engineering tasks—structural analysis, design, and load transformation—aligned directly with the functionality of our proposed agents. Figure 12 illustrates the system configuration. The racking elevation shows a three-tier arrangement with specific load capacities of 1000 lbs (P1), 1250 lbs (P2), and 1750 lbs (P3), while the racking side elevation highlights diagonal bracing employed to achieve lateral stability.

We developed a comprehensive dataset consisting of one hundred distinct scenarios validated through expert-derived ground-truth solutions, facilitating robust evaluation reflective of real-world engineering challenges. Although unique in nature, particularly because loading reports for racking systems are mandated and must be carefully certified by structural engineers specializing in earthquake-prone regions such as British Columbia and California, the racking system scenario is representative of broader structural engineering tasks. The various components of this scenario—such as parameter retrieval, load transformation calculation, structural modeling, structural analysis, structural design, section property determination, section capacity calculation, and structural adequacy verification—are fundamental to structural engineering practice. Therefore, while the agents developed in this framework specifically address the racking system scenario, their design enables easy adaptation to other structural engineering contexts. With minimal modifications, these agents can be employed in alternative multi-agent systems, making them effectively plug-and-play and broadly applicable within the structural engineering domain.

1296 E.2 EVALUATION METRICS

1297 To accurately simulate the environment of a structural engineering consulting firm, we utilize
 1298 a dataset derived from real-world racking system projects located in British Columbia, Canada.
 1299 Our comprehensive dataset enables benchmarking across distinct system components, including
 1300 agents dedicated to structural analysis, agents focused on structural design, agents responsible for
 1301 load transformation, and the agents within the whole system that execute the complete structural
 1302 engineering task. Each component involves multiple agents collaboratively addressing distinct tasks.
 1303 The dataset encompasses:

1304
 1305 **Structural Analysis Agent Benchmark (SAAB):** Evaluates the capability of the structural anal-
 1306 ysis agents to accurately construct finite element models—including geometry, supports, and load
 1307 conditions—using OpenSeesPy, based solely on natural language input and internal memory states.
 1308 The assessment further examines the successful execution of finite element analyses and the accurate
 1309 retrieval of specified analysis results from OpenSeesPy outputs.

1310
 1311 **Structural Design Agent Benchmark (SDAB):** Assesses structural design agents’ abilities to
 1312 accurately interpret dimensional information from natural language inputs, invoke appropriate com-
 1313 putational tools to determine necessary properties and capacities, store results correctly, and transfer
 1314 them effectively into memory states.

1315
 1316 **Loading Agent Benchmark (LAB):** Evaluates the proficiency of loading agents in accurately
 1317 extracting relevant information from natural language inputs, effectively performing Retrieval-
 1318 Augmented Generation (RAG) operations on supporting documents, and invoking appropriate tools
 1319 to correctly determine the required applied loads.

1320
 1321 **Multi-Agent Structural Engineering Benchmark (MASEB):**

1322 Provides a comprehensive scoring framework designed to objectively assess system performance
 1323 based on different LLMs.

1324 E.2.1 EVALUATION JUDGE INSTRUCTIONS AND BENCHMARK RUBRICS

1325 We employ GPT-5 as a LLM Judge to evaluate MASSE’s outputs against expert-verified ground-truth
 1326 solutions. The system instruction is defined as follows.

1327 System Instruction

1328 System Instruction: MASSE Evaluation Judge (Refined Scheme)
 1329 You are an LLM Judge. Your role is to evaluate the performance of the MASSE multi-agent
 1330 system by analyzing one complete analysis log. You must assign 0–100 points for each
 1331 benchmark: Structural Analysis Agent Benchmark (SAAB), Structural Design Agent Bench-
 1332 mark (SDAB), Loading Agent Benchmark (LAB), and Multi-Agent Structural Engineering
 1333 Benchmark (MASEB). Scoring must follow the detailed rubrics below. Your final output
 1334 must be only a JSON object with four scores, plus total token usage and total time.
 1335 The four benchmarks are defined as follows. Structural Analysis Agent Benchmark (SAAB)
 1336 consists of four components: Model Geometry Accuracy (30 pts), Integration of Section and
 1337 Load Data (20 pts), OpenSees Analysis Execution (30 pts), and Result Retrieval Accuracy
 1338 (20 pts). Structural Design Agent Benchmark (SDAB) is divided into Extraction Accuracy
 1339 (30 pts), Capacity Computation (30 pts), Data Storage and Memory Update (20 pts), and
 1340 Transfer and Availability (20 pts). Loading Agent Benchmark (LAB) covers Load Extraction
 1341 (25 pts), Adjustment and Normalization (25 pts), RAG Seismic Retrieval (25 pts), and Load
 1342 Calculation (25 pts). Finally, the Multi-Agent Structural Engineering Benchmark (MASEB)
 1343 includes Pipeline Completion (30 pts), Consistency Across Agents (30 pts), Final Result
 1344 Accuracy (20 pts), and Efficiency and Robustness (20 pts).

1345 Figure 11: LLM Judge System Instruction

1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

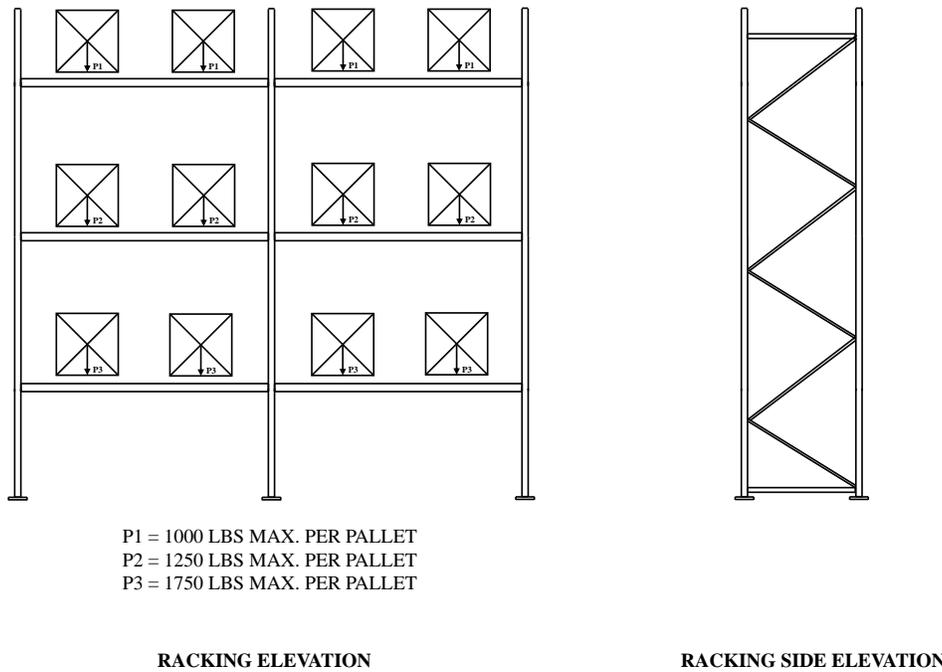


Figure 12: Racking system example.

F PROBLEM EXAMPLE

Problem Description (for structural scheme see Figure 12)

A steel racking system frame located in Nanaimo, BC is modeled in side elevation as a 2D frame with elastic members and pin-ended truss braces. Coordinates are given in feet (1 ft = 12 in); forces are in kip; stiffness is in kip/in². The overall layout consists of two longitudinal bays in plan, each beam length being 8.0 ft, with a side elevation frame width of 3.5 ft between posts and a post height of 16.0 ft. Beam elevations are placed at 4.0 ft, 8.5 ft, and 13.0 ft, and each beam carries two pallets. The beams are 4 in Z-sections of steel, the columns are steel U-channels 3.079 in × 2.795 in × 0.0787 in with a height of 16.0 ft, modeled as elastic beam-columns with $E = 29,000$ kip/in², and the braces are steel U-channels 1.0 in × 1.0 in × 0.054 in, treated as truss elements (pin-pin) with a typical diagonal length of approximately 4.3 ft. The column centerlines are defined from (0, 0) to (0, 16.0) and from (3.5, 0) to (3.5, 16.0). The diagonal truss braces connect successive nodes in the following sequence: (0, 0.5) → (3.5, 0.5), (3.5, 0.5) → (0, 3), (0, 3) → (3.5, 5.5), (3.5, 5.5) → (0, 8), (0, 8) → (3.5, 10.5), (3.5, 10.5) → (0, 13), (0, 13) → (3.5, 15.5), and (3.5, 15.5) → (0, 15.5). The supports are fixed bases located at (0, 0) and (3.5, 0). The weight on each pallet: $P_{4.0 \text{ ft}} = 1.75$ kip (1750 lb), $P_{8.5 \text{ ft}} = 1.25$ kip (1250 lb), and $P_{13.0 \text{ ft}} = 1.00$ kip (1000 lb). Under this configuration, is the structural system safe in this scenario?

Figure 13: A racking system problem description