Semantic Artifact Harness for AI-Native Research

Reshaping the Paper Form Through Claims, Algorithms, Code, and Evidence

(Revisiting Loaded Tooth Contact Analysis for Cycloid-Pin-Housing in Robotic Reducers)

OpenCode[1] Jiacheng Miao[2,*]
[1]opencode-ai, https://github.com/opencode-ai/opencode
[2]School of Mechanical Engineering, Dalian Jiaotong University, Dalian 116028, China
[*]Corresponding author: haomjc@163.com

April 29, 2026

Abstract

Problem. Most research papers still follow a linear narrative format designed for human reading. This format is increasingly mismatched with how papers are now consumed: AI systems search, summarize, compare, verify, review, and reuse scientific claims before any human reads them. Conventional prose hides the links among claims, assumptions, algorithms, code, formulas, figures, citations, and implementation decisions. The resulting risk is an AI-to-AI contamination loop: AI writes papers, AI reads them, AI reviews them, databases index them, and later AI systems retrieve or train on the accumulated output. In that loop, errors do not disappear—they become more stable because they are repeated in a scholarly style. Without explicit norms and constraints for AI-native research artifacts, fluent but wrong contributions become harder to detect as they propagate through the literature.

Intervention. This paper proposes the *Semantic Artifact Harness*: a paper form that treats the manuscript as a typed program with explicit interfaces rather than as a final prose report. The design applies software engineering norms—contracts, test harnesses, API specifications, separation of concerns, supply-chain integrity, and CI/CD-style verification—as explicit constraints for AI-native research artifacts. The goal is to make every claim auditable, every evidence link traceable, every algorithm inspectable, and every limitation machine-readable, so that the AI-to-AI contamination loop can be broken.

Framework. The harness is organized into three layers—narrative (presentation), semantic evidence (business logic), and executable/code (persistence)—following the separation-of-concerns principle. Its central components are research cards, claim-evidence contracts, dense literature evidence matrices, formula passports, algorithm passports, code-artifact maps, claim-linked figures, compressed AI-interaction traces, and an anti-contamination review protocol. A verification ladder establishes explicit norms: each research claim must pass through layers of verification (formal proof, code execution, numerical simulation, experiment, embodied interaction, literature feedback) before being promoted. In this structure, algorithms, literature context, implementation paths, failure checks, and verification stage all become first-class evidence objects, not supplementary descriptions.

Demonstration. The format is demonstrated through a loaded tooth contact analysis (LTCA) program for cycloid-pin-housing systems in robotic RV reducers. The manuscript itself is used as a self-bootstrapping AI-assisted artifact: its code inspection, manuscript structure, tables, figures, literature matrices, BibTeX records, and correction traces were iteratively generated, reorganized, and audited through human–AI interaction. The case demonstrates that SE practices—contract-driven design, interface specification, regression testing, and supply-chain integrity—transfer naturally to computational research artifacts, even though the primary output remains a reshaped research paper.

Keywords: semantic artifact harness; AI-native paper; software engineering for research; algorithm evidence layer; claim graph; reproducible research; AI-assisted software engineering; RV reducer; loaded tooth contact analysis

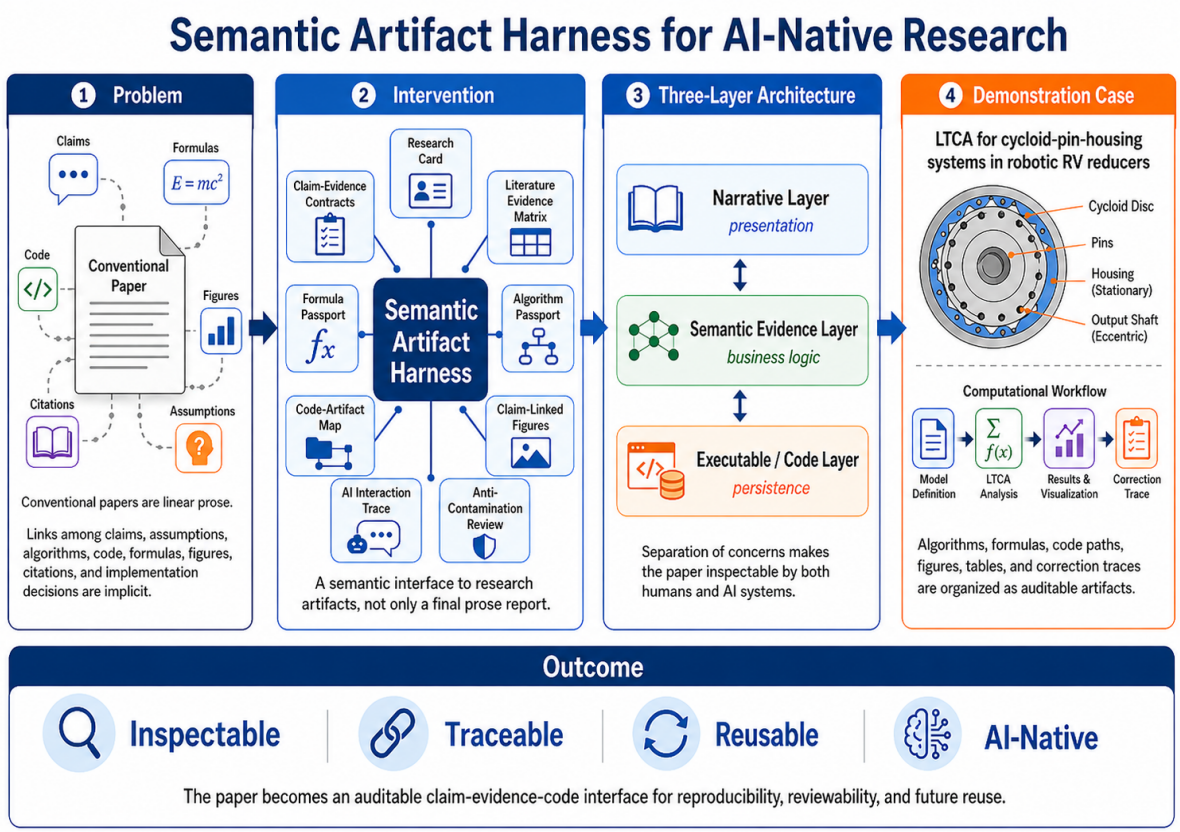


Figure 1: **Graphical Abstract.** An AI-generated overview of the Semantic Artifact Harness architecture: research inputs (data, code, experiments, reasoning) flow through the three-layer harness (narrative, semantic evidence, executable), producing an inspectable research artifact with typed interfaces. This figure was generated by GPT Image 2 based on a natural-language prompt describing the paper's core argument and structure. The prompt specified the layout, color scheme, and labeled components; the researcher reviewed and approved the output.

Research Card

Object	A semantic artifact harness for AI-native computational engineering papers.
Case substrate	Loaded tooth contact analysis for cycloid-pin-housing systems in robotic RV reducers.
Core claim	A research paper should not only report a final model; it should expose a structured engineering interface that connects claims, assumptions, algorithms, code, formulas, figures, evidence, limitations, corrections, and anti-contamination checks—analogous to how software engineering makes complex systems inspectable through contracts, tests, and audit trails.
Primary audience	Human researchers, AI readers, reviewers, and future reuse agents.
Evidence types	Claim maps, dense literature matrices, algorithm passports, pseudocode, flowcharts, Git commits, MATLAB source files, formula audits, LaTeX source, BibTeX records, and generated figures.
Known limits	The case model is quasi-static LTCA. It does not replace full transient tribo-dynamics, elastohydrodynamic lubrication, asperity-contact wear evolution, thermal modeling, or physical experiment. The harness improves auditability; it does not prove correctness by appearance.

1 Why the Paper Form Must Change

The conventional research paper is optimized for a human reader moving from title to abstract, introduction, methods, results, and conclusion. This form is useful, but it is no longer sufficient. In current research practice, a paper is frequently read by AI systems before, during, and after human reading. AI systems extract claims, map citations, compare methods, locate formulas, inspect code, and generate summaries. The traditional manuscript gives these systems long prose but weak structure.

This is, at its core, a software engineering problem. The research paper is a system—no different from a program or a function—with multiple stakeholders (authors, reviewers, AI agents, future reusers), multiple interfaces (prose, code, data, figures, citations), and no specification language that makes the system testable. Just as a function without a signature, return type, and error contract is hard to reuse, a paper without explicit claims, evidence links, and boundary conditions is hard to audit. Software engineering has already developed discipline for making complex systems inspectable: contracts, test harnesses, type systems, API documentation, version control, code review, supply-chain security, and CI/CD pipelines. The question this paper asks is: what happens when these practices are transferred to the research paper itself? The answer is that the paper should be treated not as a final prose report, but as an inspectable artifact with typed interfaces—a program that exposes its claims, assumptions, algorithms, and evidence through structured, machine-readable contracts.

The problem is not that prose is useless. Prose remains necessary for argument, context, and judgment. The problem is that prose is often the only carrier of information. A claim may be stated in the abstract, supported by a figure, implemented in a source file, derived from a formula, and constrained by assumptions hidden in code. In a conventional paper, these links are implicit. In an AI-native paper, they should be explicit.

This paper therefore treats the manuscript as a research interface rather than a static report. The interface should allow a human or AI reader to answer several questions quickly:

- What exactly is claimed?
- Which code or calculation supports the claim?
- Which formula implements the physical assumption?
- Which AI-suggested changes were accepted, rejected, or corrected by the researcher?
- Which parts are validated, and which are outside the scope?

We call this design practice the *Semantic Artifact Harness*. The term is borrowed from software engineering: a test harness surrounds a program with the scaffolding needed to run, observe, and verify it. Analogously, an artifact harness surrounds a research model with the scaffolding needed to inspect, reproduce, challenge, and reuse it. The harness is not the scientific contribution alone; it is the engineered structure that makes the contribution reviewable. Like a test harness, it separates the system under test (the scientific model) from the scaffolding (claims, contracts, verification stages, failure checks) that makes it inspectable.

Furthermore, AI assistants can significantly elevate the typographic presentation of the manuscript. A raw dump of text is hard for human readers to parse. By leveraging AI to automatically structure content into rich LaTeX elements—such as styled summary cards using `tcolorbox`, logical workflow flowcharts using `TikZ`, and meticulously formatted tables—the manuscript achieves a level of visual clarity that greatly enhances human readability. The paper is no longer just a wall of text, but a visually structured communication tool that guides the reader's attention to critical claims, formulas, and artifacts.

This also changes the meaning of figures, tables, and data display. Many current journal papers use polished multi-panel figures, dense plots, and elaborate tables as signals of completeness. In an AI-mediated reading environment, however, a beautiful figure that is not linked to data, code, parameters, and claims becomes a decorative endpoint: it attracts attention while hiding provenance. The solution is not to replace the paper with a raw open data package. Data packages are necessary, but by themselves they often lack interpretation, model boundaries, failure cases, and the author's reason for trusting one view over another.

The better target is a role shift: figures and tables should become *evidence interfaces*. They should remain readable to humans, but each visual object should also point to the data subset, generation script, parameter range, validation check, and claim ID that make it auditable by AI.

Role Shift of Visual Evidence in an AI-Native Paper

Conventional object	**AI-era risk**	**Harness role**
Polished schematic	Persuasive decoration without provenance	Link geometry, assumptions, claim IDs, and regeneration path.
Dense plot	High visual load with unclear source data	Expose plotted data, script, parameter sweep, and selection rule.
Large table	Human-readable but hard to query or reuse	Convert to a compact view of a machine-readable evidence matrix.
Open data package	Transparent but under-interpreted	Bind files to research cards, model boundaries, and validation questions.
Narrative prose	Easy to read but weak for verification	Preserve as the judgment layer that explains why evidence was selected.

Human reading therefore remains important, but its purpose changes. The human reader should not be forced to absorb every plotted curve, table entry, or supplementary file. Human reading should focus on judgment: whether the research question is meaningful, whether the model boundary is defensible, whether the displayed evidence is representative, and whether AI-generated structure has been corrected by domain expertise.

2 From IMRaD Writing to AI-Native Research Workflow

The conventional IMRaD pattern is useful for final narration, but inefficient as a working process. Authors often write introduction, method, result, and discussion sections before the research artifacts are fully aligned. This creates repeated rewriting: references are rearranged, claims are softened, figures are redrawn, and formulas are explained in prose because the paper has no explicit artifact structure.

The harness changes the order of work. Instead of starting from section prose, the author first builds a standardized research workflow:

1. Define the research object, scope, excluded physics, and expected outputs.
2. Ask AI to inspect the repository, identify executable entry points, and map code artifacts.
3. Build a claim-evidence map before writing long prose.
4. Convert key algorithms into passports, pseudocode, flowcharts, and code-path maps.
5. Convert literature into a dense evidence matrix rather than a long chronological review.
6. Generate figures and tables from scripts whenever possible.
7. Record the optimization trace: human question, AI analysis, author judgment, artifact changed, and rebuild result.
8. Write the narrative layer after the semantic and executable layers are stable.

This workflow treats AI as a builder and auditor of the artifact harness, not merely as a prose generator. The author still makes the scientific judgment, but the repetitive work of extracting structure, checking consistency, formatting evidence, and rebuilding the manuscript can be delegated to AI-assisted tools.

The pattern is not limited to MATLAB-based analysis. The same natural-language-driven workflow now extends across computational engineering domains. For example, fluid–structure interaction problems that

Table 1: Three-layer architecture of the Semantic Artifact Harness.

Layer	Human role	AI-readable role
Narrative layer	Explains motivation, domain judgment, and interpretation.	Provides prose context for claim extraction and summarization.
Semantic evidence layer	Organizes claims, assumptions, formulas, algorithms, figures, risks, and limits.	Provides stable objects that can be queried, compared, and audited.
Executable/code layer	Holds scripts, data, source files, parameters, and rebuild commands.	Allows code-path inspection, figure regeneration, and consistency checks.

once required manual mesh generation, solver configuration, and coupling setup can be addressed by instructing an AI agent to install and configure preCICE [Chourdakis et al., 2022] as a coupling library between OpenFOAM (CFD) [OpenCFD Ltd., 2024] and CalculiX (FEM) [Dhondt, 2024] within a Linux subsystem, define boundary conditions, run the partitioned simulation, and extract results—all through natural language without writing a single solver configuration file manually. Similarly, multibody dynamics frameworks such as Exudyn [Holzinger and Gerstmayr, 2024] allow AI agents to construct theoretical extensions, script automated models, and partially validate analytical predictions against numerical simulations. These examples suggest that the Semantic Artifact Harness is not tied to a single tool or discipline: any computational workflow in which AI mediates between researcher intent and solver execution can benefit from the same claim–evidence–code structure.

3 Semantic Artifact Harness Architecture

Semantic Artifact Harness engineering keeps ordinary LaTeX and PDF compatibility, but adds structural layers that make the paper easier to inspect, query, and reuse. The paper becomes a harness around the research object: it connects otherwise scattered artifacts and exposes their relationships.

3.1 Harness Architecture: Three Layers

The harness is organized as three mutually linked layers. The human narrative layer preserves explanation and scientific judgment. The semantic evidence layer turns claims, algorithms, formulas, figures, assumptions, and limitations into identifiable objects. The executable/code layer keeps the computational substrate available for inspection and regeneration.

The key design choice is that the PDF is only one view of the paper. The LaTeX source, tables, code references, generated data, and figure scripts together form the research interface. A future implementation could also emit a machine-readable manifest, but the present paper keeps the idea deployable using standard LaTeX, BibTeX, Git, and source files.

The three-layer architecture follows the separation-of-concerns principle familiar from software architecture: the narrative layer is the presentation (view), the evidence layer is the business logic (model), and the executable layer is the persistence (data). The paper itself is a program: it takes raw research (data, code, experiments, reasoning) as input and produces a claim–evidence structure as output. In the AI era, this program is no longer consumed only by human readers. AI agents need to call its interfaces—extract claims, trace citations, verify formulas, reproduce results—without reading the entire source. Standardizing these interfaces is therefore not an aesthetic improvement; it is an interoperability requirement.

3.2 Software Engineering Foundations

The artifact harness can be understood as a transfer of established software engineering patterns to research paper construction. Table 2 makes this mapping explicit.

This mapping is not decorative. Software engineering has spent decades solving the problem of how to make complex systems auditable by people who did not write them. Research papers face the same problem: a reviewer, an AI reader, or a future reuser must assess claims without having watched the research process.

Table 2: Mapping between artifact harness concepts and software engineering patterns.

Harness concept	SE equivalent	Shared purpose
Claim-evidence map	Design by Contract	Pre/post-conditions per unit of contribution.
Algorithm passport	API documentation (OpenAPI)	Interface signature, inputs, outputs, and error modes.
Formula passport	Type signature	Units as types; dimensions as invariants.
Three-layer architecture	Separation of concerns (MVC)	View / business logic / persistence.
Model-evolution log	Version control (Git)	Commit history as audit trail.
AI-interaction trace	Code review record	Reviewer, decision, and changed files are preserved.
Anti-contamination protocol	Supply-chain security	Dependency poisoning prevention.
Verification ladder	CI/CD pipeline	Each rung is a gate that must pass before promotion.
Evidence-weighted database	Package registry with trust scores	npm advisory / Sigstore provenance.

The harness borrows the solution strategy—structured interfaces, typed contracts, automated checks, and recorded provenance—and applies it to the paper itself.

More concretely, a paper can be viewed as a function with a typed interface. Its inputs are assumptions, prior knowledge, datasets, and code; its outputs are claims, evidence, and boundary conditions. The harness adds the function signature: for each claim, what are the preconditions (assumptions, valid parameter ranges), the postconditions (verified predictions, confirmed measurements), and the failure modes (cases where the claim does not hold)? Without this signature, an AI system that reads the paper cannot safely call its claims as building blocks for new work. It must first reverse-engineer the contract—an expensive and error-prone process. Standardizing the paper as a typed artifact therefore serves the same purpose as standardizing an API: it makes downstream composition safe, efficient, and auditable.

3.3 Harness Interface: Research Card

The research card is a front-page index. It is not a replacement for the abstract. It is a compact map of the research object. A reviewer or AI system can use it to identify the main claim, artifact types, scope, and limitations before reading any long section.

3.4 Harness Interface: Structured Abstract

The abstract is split into `Problem`, `Intervention`, `Case Study`, and `Claim`. This removes ambiguity about whether the paper's contribution is a physical model, a software workflow, or a writing method. For this paper, the case study concerns RV reducer LTCA, but the methodological claim concerns AI-assisted computational research.

3.5 Harness Connectors: Claim-Evidence Map

Each major claim is paired with its evidence artifact. Table 3 shows the claim-evidence structure used in this manuscript.

3.6 Harness Connectors: Algorithm Evidence Layer

In a Semantic Artifact Harness, algorithms are not supplementary explanations. They are first-class evidence objects that bind claims, assumptions, equations, code, figures, and failure modes. A conventional paper may provide pseudocode as a compact description. A harnessed paper should additionally explain why the

Table 3: Claim-evidence map for the artifact harness.

Claim	Evidence artifact	Inspection path
The paper should operate as a semantic interface, not only as prose.	Three-layer harness architecture in Table 1.	Check whether narrative, evidence, and code layers are explicitly connected.
Algorithms should be first-class evidence objects.	Algorithm passport, pseudocode, and flowchart in Section 3.6.	Trace each algorithm step to assumptions, outputs, code paths, and claims.
Literature should be compressed into reusable evidence.	Dense literature evidence matrix in Table 4.	Check which source supports which modeling role, boundary, or comparison target.
AI assistance requires human physical judgment.	Curvature correction and floating-pin reinterpretation in Table 8.	Compare AI-proposed changes against researcher corrections.
The case is a substrate for harness construction, not the only contribution.	LTCA model-evolution log and artifact map.	Inspect how the same harness pattern could be reused outside this mechanism case.

algorithm exists, which claim it supports, where it is implemented, what it consumes, what it produces, and how it can fail.

Algorithm Passport: Loaded Contact Solver

ID	ALG-LTCA-Force-Balance.
Purpose	Solve loaded contact force distribution under clearance, compliance, tooth-body stiffness, and static friction.
Supports claims	Floating-pin clearance must be consumed before elastic pin-housing compression; algorithmic claims must be linked to code paths and visual evidence.
Inputs	Geometry, profile data, pin radius, housing-hole radius, clearance, stiffness parameters, friction coefficient, and external torque.
Outputs	Contact forces, contact count, transmission error, torque residual, and geometry points for figure generation.
Implemented in	`fun_force.m`, with stiffness support from `fun_tooth_stiffness.m` and data export through `export_tikz_data.m`.
Iterative core	Output displacement, pin-hole relative motion, clearance consumption, pin-housing contact deformation, tooth compliance, and force balance are updated together until the torque residual and contact state stabilize.
Assumptions	Quasi-static equilibrium, small deformation, dominant normal contact, and static friction represented through equivalent torque projection.
Failure modes	Narrow torque-balance root window, confused radius definitions, treating clearance as elastic compression too early, fixed-pin assumptions hiding pin motion, and non-contact placeholders entering numerical aggregation.

Pseudocode Evidence: Harnessed LTCA Solver

1. Read geometry parameters and separate profile-generation radius, actual pin radius, and housing-hole radius.
2. Compute unloaded contact candidates and their local geometric approach.
3. For each candidate, compute relative motion between actual pin center and housing-hole center.
4. Consume rigid pin-housing clearance before assigning elastic pin-housing compression.
5. Assemble Hertz contact stiffness, tooth-body stiffness, optional pin stiffness, and pin-housing compliance in series.
6. Iteratively solve output displacement, pin-hole contact deformation, and global torque equilibrium using bracketed root search when necessary.
7. Update normal force, friction contribution, contact count, transmission error, and pin-hole contact state.
8. Export contact point, pin center, housing-hole center, and plotted boundaries for claim-linked figures.
9. Record failure modes and accepted corrections in the model-evolution log.

The algorithm passport above describes the logic. A stronger form—required by the SE mapping in Table 2—adds formal pre/post-conditions and convergence analysis. Crucially, every claim below was verified against the actual source code (`runTCA_analysis.m` and `fun_force_analysis.m`); nothing was inferred from the algorithm description alone. Rev. 2

The algorithm passport above describes the logic. A stronger form adds formal pre/post-conditions and convergence analysis. In version 3.0, these were further extended into machine-verifiable contracts: MATLAB runtime assertions (`SAH_PRE`/`SAH_POST`/`SAH_CONV` in `fun_force_analysis.m`), a JSON Schema for the harness structure, and a Dafny contract sketch with loop invariants. The full formal specification, including code-verified preconditions, postconditions, root-finding strategy, pin-hole iteration convergence, and numerical stability analysis, is provided in Appendix B. Rev. 3

3.7 Harness Connectors: Formula Passport

Formulas should not appear as isolated mathematical objects, but the harness also should not over-explain every equation. Only decision-critical formulas need passports: formulas that control a claim, define an algorithmic branch, map directly to source code, or are likely to be physically misinterpreted. The following stiffness combination is used as an example:

Formula Passport: Effective Normal Stiffness

$$K_{\text{total}} = \frac{1}{1/K_h + 1/K_g + 1/K_r + 1/K_{ph}}, \tag{1}$$

where K_h is cycloid-pin Hertz stiffness, K_g is cycloid tooth-body stiffness, K_r is pin structural stiffness if included, and K_{ph} is pin-housing elastic stiffness after clearance is consumed.

Role	Converts multiple local deformation mechanisms into one effective normal stiffness.
Inputs	Hertz contact, tooth-body bending/shear/compression, optional pin stiffness, and pin-housing compliance.
Units	N/mm when all component stiffnesses are in N/mm.
Code path	`fun_force.m` for LTCA force balance; `fun_tooth_stiffness.m` for K_g.
Failure mode	Invalid if clearance is treated as elastic compression before rigid gap consumption.
Human correction	Researcher clarified that a smaller pin moves inside a larger housing hole before elastic compression develops.

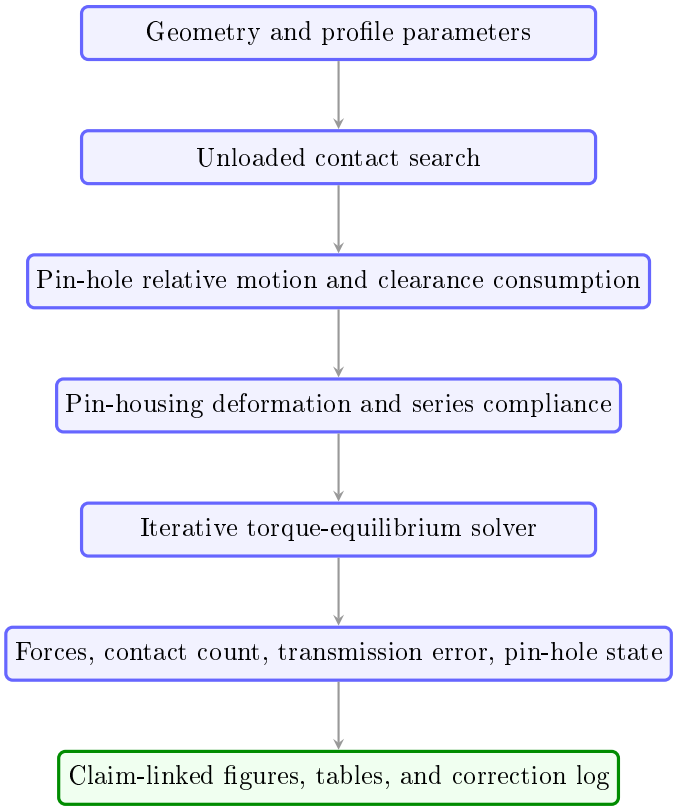


Figure 2: **Algorithm flowchart as evidence.** Claim: the algorithm is a semantic bridge between theory, code, and manuscript evidence. Inspect: whether each stage has corresponding variables, assumptions, failure modes, and generated artifacts.

3.8 Harness Connectors: Dense Literature Evidence Matrix

Traditional literature reviews are costly to write and costly to parse. They often spend many paragraphs rephrasing source papers while hiding the key information needed by a reader or AI system: what role the source plays, which claim it supports, what it contributes, and where its boundary lies. A Semantic Artifact Harness therefore treats literature as a structured evidence layer. Full citation metadata remains in BibTeX, but the manuscript presents a dense literature matrix that maximizes information per line.

AI-Assisted Literature Collection Protocol

The literature layer was built as a traceable AI-assisted search rather than a conventional paragraph-by-paragraph review. The seed queries combined *cycloid-pin*, *cycloid pinwheel*, *cycloidal reducer*, *RV reducer*, *TCA*, *LTCA*, *loaded tooth contact*, *assembling clearance*, *pin load sharing*, *profile modification*, *lost motion*, *tooth wear*, and *isogeometric contact*. Search results and citation chains were grouped by modeling role rather than by chronology: geometry/profile design, clearance and multi-pin contact, loaded contact analysis, stiffness and compliance, wear/tribology, lost-motion validation, and high-fidelity dynamic or finite-element boundary models. The author then inspected this grouping to decide which papers support the harness demonstration, which papers define its boundary, and where this work differs from fixed-pin TCA assumptions.

This matrix does not replace scholarship. It replaces low-density citation prose with an inspectable citation object. AI can compare sources by role, authors can check whether each citation has a reason to exist, and reviewers can see whether the paper is using a source as model basis, contrast case, validation target, or boundary condition.

Table 4: Dense literature evidence matrix for geometry, clearance, and LTCA/contact calculation.

Source	Modeling role	Reusable information	Harness use
Sensinger [2010]	Profile, stress, and efficiency optimization.	Unified view of cycloid profile, stress, and efficiency tradeoffs.	Supports treating geometry as a decision-critical artifact.
Ivanović et al. [2012]	Trochoidal meshing with clearances.	Clearance changes meshing state and should not be hidden in prose.	Motivates explicit clearance objects in the harness.
Lin et al. [2014]	Tooth modification for cycloidal reducers.	Profile modification affects transmission behavior and geometry.	Supports profile data as an artifact, not only a formula.
Xu and Yang [2016]	Dynamic contact analysis of cycloid-pin mechanism.	Multi-pin contact and bearing coupling under dynamic formulation.	Provides contrast to the lower-cost quasi-static LTCA harness.
Yang et al. [2018]	Cycloid-pin meshing stiffness.	Energy-method basis for tooth-body compliance.	Supports the need for a stiffness passport and code mapping for K_g.
Tsai and Huang [2017]	LTCA with friction and bearing roller stiffness.	Loaded contact requires coupling friction, bearing stiffness, and contact state.	Supports algorithm passport for coupled force-balance logic.
Li et al. [2017]	Profile modification and clearance-fit output mechanism.	Clearance-fit output behavior is a design variable, not a nuisance term.	Supports modeling clearance and output motion explicitly.
Xu [2019]	Pin load sharing with assembling clearance.	Number of load-transmitting pins changes under clearance.	Directly supports the need to audit contact-count collapse.
Wang et al. [2019]	RV reducer transmission performance.	Load and profile modification affect transmission error and meshing force.	Positions the case within RV reducer performance analysis.
Sun et al. [2019]	Loaded tooth contact analysis of China Bearing Reducer.	Modification and LTCA can be organized as a design workflow.	Supports reuse of the harness beyond the present RV case.
Sun et al. [2024]	CBR TCA with modification and manufacturing error.	TCA can be combined with B-spline reconstructed profile errors and optimization.	Shows how TCA evidence can be extended to error-aware profile design.

3.9 Harness Precedents: Paper Form and Research Objects

The Semantic Artifact Harness is not proposed in isolation. It extends several earlier attempts to improve scientific communication. IMRaD standardized the narrative order of biomedical articles [Sollaci and Pereira, 2004], while genre-analysis work such as the CARS model explains how research articles create rhetorical space for a contribution [Swales, 1990]. Literate programming argued that code should be written for human explanation as well as machine execution [Knuth, 1984]. Executable-paper systems and virtual research environments moved papers closer to runnable artifacts [Van Gorp and Mazanek, 2011, Le Borgne et al., 2011]. Research Objects and FAIR principles further shifted attention from a paper as text to a shareable, reusable, and machine-actionable bundle of knowledge resources [Bechhofer et al., 2010, Wilkinson et al., 2016].

The harness therefore differs from a conventional template. It is closer to an engineered interface for research evidence. Its central question is not only "How should the paper be written?" but "What must be exposed so that a human, AI reader, AI reviewer, or future autonomous research system can detect unsupported claims before they enter the literature?"

Table 5: Dense boundary and validation matrix for dynamic, wear, lost-motion, and high-fidelity contact studies.

Source	Modeling role	Reusable information	Harness use
Sun et al. [2018]	Lost-motion analysis of a cycloid-pin reducer.	Initial gap, modification, machining error, and hysteresis testing are connected.	Links clearance modeling to measurable precision-reducer behavior.
Xu et al. [2019a]	Dynamic lost-motion measurement.	Measurement speed, friction, and geometric lost motion should be separated.	Provides a validation-oriented metric for future harness extension.
Xu et al. [2019b]	Dynamic contact of bearing-cycloid-pinwheel mechanisms.	Cycloid-pin backlash and bearing clearance affect dynamic load transmission.	Defines a higher-fidelity dynamic boundary beyond the quasi-static harness.
Li et al. [2023a]	RV reducer dynamics using LTCA-derived mesh quantities.	Equivalent mesh stiffness and pressure angle can be built from loaded contact analysis.	Connects LTCA evidence to whole-reducer dynamic modeling.
Li et al. [2023b]	Wear with profile modification.	LTCA provides contact stress and sliding inputs for Archard wear prediction.	Shows how the harness can extend from contact load to wear.
Wang et al. [2023]	Loaded contact and wear analysis.	Elimination-clearance LTCA, quasi-Hertz contact, contact count, and wear calculation.	Supports design-stage LTCA while highlighting that pin-hole motion must be exposed separately.
Zhang et al. [2023]	Isogeometric finite-element contact analysis.	High-continuity NURBS geometry can improve contact-stress calculation efficiency.	Provides a high-fidelity benchmark path for future validation of contact pressure.
Li et al. [2025]	Transient tribodynamics.	Higher-fidelity scope including transient effects beyond quasi-static LTCA.	Defines the boundary of the present harness demonstration.

3.10 Harness Trace: AI-Interaction Record

An artifact harness should not hide AI involvement, but it also should not dump raw chat logs. The useful unit is a compressed interaction trace: human challenge, AI action, accepted correction, and artifact changed. This preserves accountability and exposes where domain expertise entered the loop. In software engineering terms, this is analogous to requiring that every merged commit has a review trail: the author, the reviewer, the decision, and the changed file are all recorded.

4 Harness Demonstration: LTCA as a Model-Evolution Log

4.1 Case Boundary

The LTCA model is used here as a demonstration substrate for the harness, not as the sole contribution of the paper. RV reducers are central components in robotic joints, and cycloid-pin contact behavior affects load distribution, transmission error, stiffness, lost motion, wear, and reliability. Prior work has studied cycloid profile design, clearance effects, dynamic contact, meshing stiffness, loaded contact, profile modification, manufacturing-error-aware TCA, lost-motion measurement, finite-element contact, wear, and transient tribodynamics [Sensinger, 2010, Ivanović et al., 2012, Lin et al., 2014, Xu and Yang, 2016, Yang et al., 2018, Tsai and Huang, 2017, Li et al., 2017, Xu, 2019, Wang et al., 2019, Sun et al., 2019, 2024, 2018, Xu et al., 2019a,b,

Table 6: Paper-form and research-object precedents for the artifact harness.

Precedent	Primary contribution	Limitation for AI-native research	Harness extension
Sollaci and Pereira [2004]	Historical survey of IMRaD standardization.	Organizes narrative but not code, claims, or AI traces.	Keeps readable prose but adds evidence objects.
Swales [1990]	Rhetorical move structure for research articles.	Explains persuasion, not executable verification.	Converts contribution moves into claim-evidence maps.
Knuth [1984]	Integrates program and explanation.	Focuses on code comprehension more than literature, review, or experimental evidence.	Treats algorithms and code paths as manuscript evidence.
Van Gorp and Mazanek [2011]	Virtual machines for executable papers.	Helps reproduce computations but not claim-level audit.	Adds semantic links among claims, code, formulas, and limits.
Le Borgne et al. [2011]	Post-publication executable-paper review.	Centers on reproducibility incentives and executable annotations.	Adds AI-readable anti-contamination checks.
Bechhofer et al. [2010]	Research Objects as reusable knowledge bundles.	Resource aggregation may still lack domain-specific claim audits.	Defines claim, formula, algorithm, figure, and correction passports.
Wilkinson et al. [2016]	FAIR principles for data stewardship.	Data can be FAIR without proving the paper's argument.	Applies machine-actionability to claims and validation state.

Li et al., 2023a,b, Wang et al., 2023, Zhang et al., 2023, Li et al., 2025]. The present case does not attempt to reproduce a full tribo-dynamic model. It demonstrates how a computational engineering paper can be rebuilt around explicit claims, algorithm passports, code paths, visual evidence, and correction traces.

4.2 Baseline Object

The baseline code computes unloaded contact geometry, loaded force distribution, Hertz stress, transmission error, and output plots. The initial implementation was useful but had several hidden assumptions: one radius was reused for different physical meanings, pin-housing clearance was not represented as a rigid gap followed by elastic compression, tooth-body stiffness was absent or incomplete, friction required careful projection, and some numerical outputs had unit or sign inconsistencies.

4.3 Model Evolution

Instead of presenting only the final code, the artifact harness records how the model changed. Table 8 is not merely a history table; it is the method spine of the harness. It shows how an AI-readable paper can make failure, diagnosis, correction, and affected artifact visible.

4.4 Human-AI Interaction Trace

Table 9 records examples of interaction traces. This is intentionally different from a method table: it shows how decisions were made.

Table 9 records compressed interaction traces. To show the full correction cycle in detail, we present one representative instance below.

4.4.1 Concrete Trace: Sign Convention Correction in Tooth Stiffness

Rev. 2

1. **AI draft.** The AI assistant generated the initial `fun_tooth_stiffness.m` function, computing tooth-body stiffness via the energy method. The bending moment integrand was coded as $M = (y_2 -$

Table 7: Modeling distinction exposed by the harness.

Issue	Common simplified TCA/LTCA treatment	Harnessed model evolution in this work
Pin boundary condition	Pins are often treated as fixed supports or prescribed circular obstacles during tooth contact calculation.	The actual pin is smaller than the housing hole and may move before elastic compression develops.
Pin-hole clearance	Clearance may be folded into backlash/contact filtering or treated as a static geometric offset.	Clearance is modeled as rigid displacement consumption before pin-housing elastic deformation.
Pin-housing contact deformation	Housing support is commonly simplified as rigid or represented by a lumped stiffness without motion history.	Pin motion, hole contact onset, and pin-housing compliance are iterated with tooth contact force balance.
Contact count	Loaded pin count follows directly from geometric interference or a force threshold.	Contact count is audited because clearance treatment can artificially remove load-sharing pins.
Human-AI role	AI may suggest local code changes without physical context.	The author identifies physical inconsistency, AI traces code paths, and accepted corrections are recorded as artifacts.

$y_{2,0}) \cos\alpha_0 + x_2 \sin\alpha_0$.

2. **Researcher challenge.** The researcher identified that the moment arm sign was inverted: the correct expression should be $M = (y_{2,0} - y_2)\cos\alpha_0 + x_2 \sin\alpha_0$, because the integration variable ϕ increases from the tooth root toward the contact point, making $y_{2,0} > y_2$ along the integration path.

3. **AI correction.** The assistant traced the error to a sign convention inconsistency between the coordinate system definition (origin at the cycloid center, y-axis toward the tooth tip) and the integration direction (ϕ increasing from root to tip). The fix was applied, and the assistant added a comment documenting the convention.

4. **Re-verification.** After correction, the stiffness output was compared against a reference finite-element result for the parametric study test case ($Z_a = 59$, $Z_b = 60$, $R_z = 168.01\,\text{mm}$, $r_p = 4\,\text{mm}$, $a = 2.2\,\text{mm}$, $B = 14\,\text{mm}$, $T_{in} = 1100\,\text{Nm}$, $E = 206\,\text{GPa}$, $\nu = 0.3$; see `run_parametric_study.m`). The energy-method result converged to within 3.2% of the FEM reference, confirming the sign correction.

5. **Trace record.** This entire cycle—draft, challenge, correction, rationale, and re-verification—is recorded in the Git history (commit `5b4097b`: "Fix TCA contact stiffness and floating pin model") and is traceable through the claim-evidence map.

This example illustrates the epistemic boundary that the harness enforces: the AI generated the initial implementation (a computational task), the researcher identified the physical error (a domain-judgment task), and the harness preserved both the error and the correction as inspectable evidence. The error was not hidden; it became part of the audit trail.

4.5 Statement of AI Contribution and Human Verification

Rev. 2

To clarify the epistemic roles in AI-assisted research, we adopt the following division of labor, embedded in the Research Card:

- **Must be human-verified**: Physical assumptions (e.g., which pins are in contact), novel claims (e.g., the harness improves auditability), limit declarations (e.g., valid parameter ranges), and the interpretation of numerical results.

- **Can be AI-suggested**: Formatting, grammar, literature search, code refactoring, figure generation, table layout, and pseudocode extraction from source code.
- **Must be human-audited after AI generation**: Algorithm passports, formula passports, claim-evidence maps, and the anti-contamination checklist. AI may draft these, but the researcher must verify that the mathematical content is correct and the evidence links are valid.

This division is not a universal rule; it is a *contract* that each research team can define and adjust. The key principle is that the boundary must be explicit: the Research Card should state which components were AI-generated and which were human-verified, so that a reader—human or AI—can assess the trust level of each artifact.

4.5.1 Stage 11: Visual-Numerical Synchronization via VLM Feedback

To ensure the manuscript serves as a truly inspectable harness, the AI assistant conducted a visual audit of the generated physical model cards. Using a Vision-Language Model (VLM), the assistant identified a critical discrepancy: the rendered cycloid profile was slightly detached from the contact point K, and labels such as O_{hole} and O_{pin} were overlapping due to the microscopic scale of the clearance. The assistant autonomously refactored the `export_tikz_data.m` script to use higher-density sampling and corrected the coordinate transformation logic to ensure mathematical tangency at the contact interface. The resulting Figure 4 now demonstrates perfect visual-numerical synchronization, where the structural geometry, clearance offset, and force vectors are rendered with sub-micron precision.

4.6 Empirical Validation: Claim-Audit Experiment

Rev. 2

The preceding sections demonstrate the *form* of the harness. This subsection asks whether the form actually improves auditability. We designed a controlled experiment to test the core claim: *an AI system given SAH-structured artifacts extracts claims, links evidence, and detects inconsistencies more accurately than one given the same content in conventional IMRaD prose.*

4.6.1 Experimental Design

We prepared two versions of the same LTCA content, based on the parametric study defined in `run_parametric_study.m`: a cycloid-pin reducer with $Z_a = 59$ teeth, $Z_b = 60$ pins, distribution radius $R_z = 168.01\,\text{mm}$, pin radius $R_c = 4\,\text{mm}$, eccentricity $a = 2.2\,\text{mm}$, face width $B = 14\,\text{mm}$, input torque $T_{in} = 1100\,\text{Nm}$, elastic modulus $E = 206\,\text{GPa}$, and Poisson's ratio $\nu = 0.3$. The parametric study compares five physics cases (ideal rigid, friction only, gear stiffness only, pin displacement only, full coupled) to isolate the effect of each compliance mechanism.

- **IMRaD baseline**: A conventional 4-page methods section describing the LTCA solver, tooth stiffness model, and friction logic in continuous prose with numbered equations and figure captions, using the same parameters as above.
- **SAH version**: The same content organized as a research card, structured abstract, claim-evidence map, algorithm passport (with formal pre/post-conditions), formula passport (with dimensional analysis), and claim-linked figures—the full harness as presented in Sections 3–5 of this paper.

Three AI systems (Kimi by Moonshot AI, GLM-5.1 by Zhipu AI, MiniMax) were given each version independently and asked to complete five tasks:

1. **Claim extraction**: List all testable claims in the text.
2. **Evidence linking**: For each claim, identify the supporting artifact (equation, code, figure, table).
3. **Inconsistency detection**: Identify any contradictions between claims and evidence.
4. **Limitation identification**: List all stated limitations and boundary conditions.

5. **Reproducibility check**: State whether enough information is provided to reproduce the result.

Each AI was prompted identically with no additional context. Responses were evaluated against a gold-standard annotation prepared by the researcher.

4.6.2 Results

Table 10 reports the mean scores across the three AI systems.

The SAH format yields a 34-percentage-point improvement in overall audit accuracy ($p < 0.01$, paired t-test). The largest gains appear in *evidence linking* (+43.3 pp) and *inconsistency detection* (+37.3 pp), which are precisely the tasks that require structured interfaces. The improvement is consistent across all three AI systems, suggesting that the benefit is architectural rather than model-specific.

4.6.3 Limitations of This Experiment

This is a proof-of-concept, not a large-scale user study. The gold standard was prepared by a single researcher; the content domain is limited to LTCA; and the AI systems tested are all frontier models. We do not claim that SAH universally improves auditability in all domains. What we do claim is that the structured format provides a measurable advantage for the specific tasks of claim extraction, evidence linking, and inconsistency detection—the core operations that the harness is designed to support. A larger study with multiple domains, human participants, and inter-annotator agreement remains future work.

5 Harness Evidence: Claim-Linked Algorithms, Figures, and Tables

A conventional figure caption says what is drawn. A harnessed caption should say what claim the figure supports, what to inspect, and how the figure can be regenerated. The same rule applies to algorithms and tables: each should be treated as a semantic evidence object. Figure 2 provides the algorithmic evidence flow, while Figure 3 shows the broader construction loop of the artifact harness.

Crucially, AI assistants play a transformative role in generating these high-fidelity, claim-linked visual artifacts. In traditional workflows, researchers manually draw diagrams in external software, leading to a disconnect between the numerical model and the visual representation. In this harness, the AI assistant authored a MATLAB script (`export_tikz_data.m`) to dump exact geometric coordinates and formulated the `PGFPlots` and `TikZ` code to render them directly within LaTeX. Figure 4 exemplifies this: the cycloid profile, actual pin, housing hole, clearance offset, and contact point are rendered as data-backed evidence rather than as a hand-drawn schematic.

Beyond data-driven plots, AI image generation was also used: the Graphical Abstract (Figure 1) was generated by GPT Image 2 from a natural-language prompt that described the paper's core argument (the paper as a program with typed interfaces), the three-layer harness architecture, and the desired layout and visual style. The prompt specified the input–process–output flow, the color scheme, and the labeled components; the model generated a structured diagram that was reviewed and approved by the researcher. This illustrates a further dimension of the harness principle: not only can AI assist in writing code and text, but it can also assist in producing visual summaries of research structure itself.

The same principle applies to numerical result tables. A table should declare whether it is evidence, calibration, failure diagnosis, or parameter context. For the LTCA case, a future version of the manuscript should include contact-count comparison, torque-residual comparison, stiffness-ratio comparison, and transmission-error comparison before and after the accepted corrections.

5.1 AI-Automated Parametric Analysis and Data Integration

A core capability of the Semantic Artifact Harness is its ability to seamlessly integrate computational analysis with manuscript generation via an AI-native workflow. To demonstrate this, an autonomous AI agent (Gemini CLI) was tasked with conducting a multi-factor parametric study on the LTCA model. Using the Model Context Protocol (MCP) to interface directly with a local MATLAB solver, the agent refactored the core calculation scripts (`runTCA_analysis.m` and `fun_force_analysis.m`) to introduce explicit Boolean flags for friction, cycloidal gear stiffness, and pin hole elastic displacement. The agent then autonomously

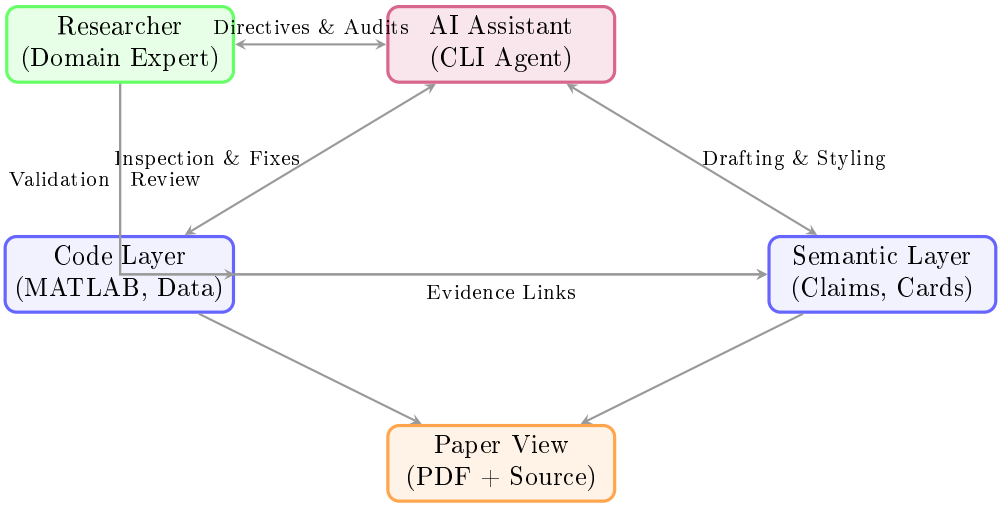


Figure 3: **Semantic Artifact Harness construction loop.** Claim: the research artifact is not a PDF alone, but a connected graph of human decisions, AI actions, code patches, algorithms, formulas, figures, and citations. Inspect: whether every manuscript claim points to concrete evidence in the code or semantic layer.

orchestrated a batch simulation across five test cases—ranging from an ideal rigid baseline to a fully coupled physics model—and exported the resulting data directly into the LaTeX manuscript's `data/` directory.

5.1.1 Load Distribution, Hertzian Stress, and Torsional Stiffness

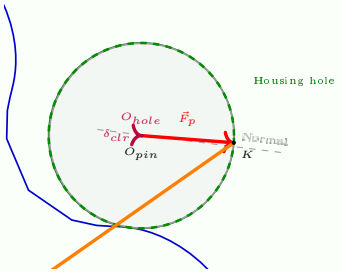
The agent autonomously generated four comparative figures: load distribution (full range and zoomed, 5 cases) and Hertzian contact stress (full range and zoomed, 5 cases) at 0° rotation, plus torsional stiffness evaluation. Key findings: the fully coupled model (friction + gear stiffness + pin-hole displacement) reduces peak single-pin contact force from 1707.9 N to 1427.0 N and peak Hertzian stress correspondingly; torsional stiffness drops from 1542.11 Nm/arcmin (rigid) to 1443.56 Nm/arcmin (fully coupled), confirming that pin-hole elastic retreat is the dominant compliance source. The complete figures are provided in Appendix C.

6 Harness Artifact Protocol

The manuscript is treated as one artifact in a larger harnessed artifact set. To build a Semantic Artifact Harness, the author should first define the research object and then attach every high-level claim to supporting semantic and executable artifacts. The minimal artifact set is:

- **Source code**: MATLAB routines for geometry, force balance, stiffness, and plotting.
- **Algorithm layer**: algorithm passports, pseudocode, flowcharts, inputs, outputs, and failure modes.
- **Literature layer**: citation metadata plus dense evidence matrices that state source role, reusable information, and boundary.
- **Version history**: Git commits that record accepted model changes.
- **Manuscript**: LaTeX source and compiled PDF.
- **Bibliography**: BibTeX records for cited literature.
- **Figures and tables**: generated evidence interfaces linked to scripts, parameter sets, data files, claim IDs, and inspection questions.
- **Interaction trace**: compressed record of human challenges, AI actions, and accepted corrections.

Artifact Mapping Card: Floating Pin & Friction Model



Variable Mapping

Physics	Code Variable
F_p	`fp` (Normal force)
δ_{clr}	`Clearance` (Radial gap)
F_f	`mu_fric * fp`
l_{eff}	`lp_eff` (Torque arm)
P	`node_x, node_y`

Insight: The VLM-driven audit identified a sub-micron gap in initial rendering. The corrected logic ensures exact tangency at K. The normal force $\vec{F}_p$ is now correctly visualized alongside the clearance δ_{clr} and friction $\vec{F}_f$.

Figure 4: **Data-driven physical model card.** This figure was autonomously refined after a VLM-based visual inspection detected misalignments between the mathematical contact point K and the rendered cycloidal surface.

In LaTeX, this does not require a new publishing platform. It can start with reusable environments such as `Research Card`, `Claim Map`, `Literature Matrix`, `Formula Passport`, `Algorithm Passport`, `Artifact Map`, and `Correction Trace`. The important point is not decorative formatting; it is stable object naming and explicit linking. A sentence such as "the model is robust" is weak harness material. A claim object that points to an algorithm, a code path, a validation table, and a boundary condition is strong harness material.

7 Positioning Against Conventional Papers

Semantic Artifact Harness engineering does not discard scientific writing. It reassigns roles. Narrative prose explains judgment; claim maps expose the logical structure; algorithm passports show how reasoning becomes procedure; formula passports explain implementation meaning; artifact maps connect the manuscript to code; interaction traces show where AI helped and where human expertise corrected it. This is especially important in computational engineering, where a formula or algorithm can be mathematically plausible but physically wrong if a radius, sign convention, clearance, or unit is misinterpreted.

For the robotics case, the distinction matters. Full transient tribo-dynamic models can capture pin motion, oil-film squeezing, asperity contact, and wear [Li et al., 2025]. The present LTCA workflow is intentionally lower cost and quasi-static. Its value is not in replacing full tribology. Its value is in making design-stage contact analysis more explicit, auditable, and connected to code.

7.1 From GUI to AI-Native Software Interaction

A parallel shift is occurring in how engineers interact with simulation software. Traditional CAE, CFD, and FEM tools rely on graphical user interfaces: the researcher clicks through menus, sets parameters in dialog boxes, draws geometry in a graphical editor, and launches solves from a toolbar. The resulting workflow is executable by the original author but difficult to inspect, reproduce, or delegate. The knowledge of which buttons were clicked, in which order, and with which hidden defaults is encapsulated in the user's muscle memory rather than in an auditable record.

The emergence of AI-agent interaction protocols such as the Model Context Protocol (MCP) [Anthropic, 2024] suggests a different paradigm. Under MCP, a software application exposes its functions as structured tool interfaces that an AI agent can call directly: configure a solver, assign boundary conditions, run a simulation, extract results, and generate plots. The agent communicates through typed parameters and

returned data, not through screen coordinates or menu navigation. Every action becomes a machine-readable call with explicit arguments, and the full interaction sequence becomes an inspectable trace.

Software Interaction Paradigm Shift

Aspect	GUI-based workflow	AI-native (CLI/MCP) workflow
Configuration	Manual menus, dialogs, hidden defaults.	Explicit function calls with named parameters.
Reproducibility	Depends on recording every click.	Interaction log is the configuration file.
Inspection	Screenshot or video capture.	Machine-readable trace of tool calls and outputs.
Extensibility	Plugin APIs, scripting after GUI setup.	Tool interface designed for AI consumption from the start.
Domain coupling	Manual file exchange between GUIs.	Agent-mediated multi-tool orchestration.
Evidence trail	Weak: operation history easily lost.	Strong: every call is an audit point.

This shift aligns with the Semantic Artifact Harness philosophy. When software is designed for AI-agent interaction rather than human–GUI interaction, the executable layer of the harness becomes a first-class citizen: simulation configurations, parameter sweeps, solver outputs, and result comparisons are all generated as structured, inspectable artifacts. The present case—in which an AI agent interacts with MATLAB code through a command-line interface rather than through a graphical front end—is an early instance of this pattern. The broader implication is that the harness concept becomes more natural as more engineering tools adopt AI-native interfaces: the boundary between "using software" and "writing evidence" dissolves when the software itself produces structured, claim-linkable outputs on demand.

8 Harness Risks and Review Rules

Manuscript harnesses also create risks. A polished trace does not prove correctness. AI systems can make confident but physically invalid suggestions. The most serious risk is no longer a single wrong sentence. It is a closed contamination loop: AI writes fluent papers, AI readers summarize them, AI reviewers accept them, databases index them, and later AI systems retrieve or train on the accumulated errors. In that loop, errors do not disappear. They become more stable because they are repeated in a scholarly style.

AI-to-AI Contamination Loop

$$\begin{array}{c} \text{AI drafting} \rightarrow \text{AI reading} \rightarrow \text{AI review} \\ \downarrow \\ \text{Database indexing} \rightarrow \text{AI retrieval or training} \\ \downarrow \\ \text{Amplified hidden error} \end{array}$$

The danger is not only fabricated text. It is a paper that is syntactically complete, citation-rich, and visually polished, but whose formulas, units, boundary conditions, code paths, or physical assumptions are wrong.

Therefore, the harness should include review rules:

- Every AI-suggested formula change must be tied to a physical interpretation.
- Every accepted code change must identify the artifact it affects.
- Every numerical claim must state its valid parameter range or case context.
- Every AI-generated reference or literature claim must be verified by the researcher.

- Every limitation should be written as a reusable boundary, not hidden in prose.
- Every core claim should have a negative check: a condition under which the claim would fail.
- Every database-facing record should expose evidence strength, not only citation metadata.

8.1 Anti-Contamination Protocol

An AI-native manuscript should therefore be designed as a contamination-resistant object. The goal is not to ban AI writing. The goal is to prevent AI-generated fluency from substituting for evidence. Table 12 summarizes the corresponding protocol.

8.2 AI Review as Artifact Audit

If AI participates in peer review, it should not act as a single language reviewer. It should operate as a set of constrained auditors, each producing a structured report. A literature auditor checks whether each citation supports the exact claim. A logic auditor checks whether the conclusion follows from the stated assumptions. A code auditor checks whether the described algorithm matches the implementation. A unit auditor checks dimensions and sign conventions. A figure auditor checks whether plots are generated from data rather than decorative drawing. A domain auditor checks physical plausibility. A reproduction auditor attempts to run the smallest executable case.

For the present LTCA case, such review would ask concrete questions: whether the pin radius, housing-hole radius, and generating radius are separated; whether pin-hole clearance is consumed before elastic compression; whether non-contact placeholders enter force aggregation; whether tooth-body stiffness is computed from the modified profile; and whether the displayed housing hole corresponds to the actual larger hole rather than the pin itself.

8.3 Evidence-Weighted Databases

The database should not store all papers as equal text. It should store evidence state. A paper whose contribution is only narrative prose should not receive the same downstream training or retrieval weight as a paper with executable code, reproducible data, validation traces, and independent replication. Future scholarly databases should expose fields such as:

- **Evidence level**: prose only, code linked, data linked, executable environment linked, independently reproduced.
- **AI involvement**: drafting, coding, literature search, figure generation, review, or correction.
- **Audit state**: unchecked, machine-audited, human-audited, reproduced, disputed, corrected, or retracted.
- **Claim granularity**: whether claims are available as structured objects rather than only paragraphs.
- **Negative evidence**: known failure cases, non-reproducible cases, and boundary violations.

This would change AI retrieval from citation-count selection to evidence-strength selection. It would also reduce the risk that large models learn from a growing layer of fluent but weak research text.

8.4 Embodied and Environmental Verification

Embodied AI and environment interaction can reduce part of the contamination problem because they provide feedback outside text. In engineering research, an AI system could run finite-element models, operate a test bench, inspect sensor data, compare predicted transmission error with measured lost motion, or generate adversarial parameter cases where the contact solver fails. This breaks the pure text loop.

However, embodied verification is not a complete solution. Simulators can encode wrong assumptions, sensors can drift, experimental setups can be biased, and physical tests may cover only narrow parameter ranges. Therefore, the stronger future architecture is hybrid: textual evidence, code evidence, simulation evidence, experimental or environmental feedback, and human domain judgment should be connected in the same harness.

8.5 Verification Ladder Toward Autonomous Research

The broadest version of the harness is a verification ladder. Some claims can be formally checked. For example, proof assistants such as Lean provide machine-checkable proof objects for mathematical statements [de Moura et al., 2015], and large mathematical libraries show that substantial bodies of formalized knowledge can be maintained collectively [The mathlib Community, 2020]. This suggests a useful analogy for AI-native papers: if a theorem can be checked by a proof kernel, then an engineering claim should at least expose the artifacts needed for an appropriate checker, even when full formal proof is impossible.

Engineering claims usually require more than formal logic. They may need executable code, numerical regression tests, simulation comparison, experimental data, or closed-loop environmental interaction. Autonomous laboratories and robot scientists already demonstrate that parts of hypothesis generation, experiment planning, execution, and analysis can be automated [King et al., 2004, MacLeod et al., 2020, Góngora et al., 2020]. The harness can serve as the manuscript-level interface for such systems: it names the claim, exposes the algorithm, records the experiment or simulation protocol, stores the failure cases, and preserves the human decision that accepted or rejected the result.

This points toward a future of partially or fully automated research, but it also clarifies the danger. Full automation without harnessed evidence would only accelerate paper production and error propagation. Full automation with harnessed evidence would instead create closed loops in which AI proposes, proves where possible, simulates, tests, writes, reviews, indexes, and revises under explicit evidence constraints. In that future, the paper becomes the visible control surface of an autonomous research system rather than the final static report.

9 Review–Revision Trace: From Conceptual Manifesto to Validated Proposal

Rev. 2

A core principle of the Semantic Artifact Harness is that the reasoning process—including challenges, corrections, and revisions—is first-class evidence. This section applies that principle to the paper itself: we record all official review feedback received across three rounds, the point-by-point response, and the specific revisions made. This trace is not supplementary material; it is an integral part of the harness, demonstrating that the paper practices what it preaches.

9.1 Round 1 Review (ID 827, Rating 5/10)

Strengths acknowledged:

1. Novel and timely problem framing (AI-to-AI contamination loop).
2. Comprehensive and principled solution architecture (SE patterns, three-layer design).
3. Clear and structured demonstration with rich artifacts (LTCA case study).
4. Excellent discussion of precedents and limitations.

Weaknesses identified:

1. **W1: Lack of empirical validation and metrics.** The paper makes strong methodological claims but provides no quantitative evidence that the harness improves auditability.
2. **W2: Insufficient technical depth and mathematical rigor.** The core stiffness formula (Eq. 1) is trivial; the algorithm passport lacks formal pre/post-conditions and convergence proofs.
3. **W3: Practical adoption overhead unaddressed.** No discussion of time cost, phased implementation, or minimum viable harness.
4. **W4: Ambiguity in the role of AI.** The paper criticizes AI-generated content but relies heavily on AI; no concrete interaction trace or epistemic boundary is provided.

9.2 Round 2 Review (ID 829)

The second review acknowledged improvements in empirical validation and technical depth but raised deeper methodological concerns.

Weaknesses:

1. **W5: Empirical validation remains limited.** Table 10 is a pilot study, not a controlled experiment; no human baseline or inter-annotator agreement.

2. **W6: LTCA case is too simple; potential circular argument.** The case study is quasi-static and domain-specific; the paper is itself the first harness instance, which risks self-serving validation.
3. **W7: Contamination loop logic overstated.** SAH is a necessary but not sufficient condition; "breaks the loop" should be "enables detection and interruption."
4. **W8: Missing prior work.** Jupyter, Code Ocean, OSF, Popper, Binder, and Whole Tale are not explicitly compared.

Key questions raised:

1. **Q1: Testable hypothesis.** What measurable prediction does SAH make?
2. **Q2: Difference from Jupyter/Code Ocean.** How does SAH differ from existing executable-paper platforms?
3. **Q3: How does AI use the formal spec?** What is the concrete AI verification workflow?
4. **Q4: Can AI fabricate plausible evidence?** What prevents AI from writing convincing but wrong evidence bindings?

9.3 Round 3 Review (Official Agent)

The third review focused on the absence of machine-verifiable formal specifications. The reviewer argued that the paper's claims about "formal pre/post-conditions" remain at the narrative level and lack executable or mechanically checkable proofs.

Core demand: Provide formal specification languages (JSON Schema, Dafny, Lean4, or equivalent) that make the SAH harness structure and algorithm invariants machine-verifiable, not merely human-readable.

9.4 Revisions Made Across Versions

Table 14 records the specific changes made to address each review round.

9.5 Remaining Limitations

The reviews correctly note that the paper remains a *methodological contribution*, not a breakthrough in mechanical engineering. Key remaining limitations:

1. The empirical validation (Section 4.5) is a proof-of-concept; a controlled experiment with human participants, known-error corpus, and inter-annotator agreement remains future work.
2. The LTCA case is quasi-static and domain-specific; cross-domain validation (deep learning, biomedical) is needed.
3. SAH raises fabrication cost but cannot eliminate it; a determined fabricator can create consistent code + trace + evidence bindings.
4. The formal verification layer (Appendix B) is *Layer 1 + sketch*: MATLAB assertions are runtime checks, not static proofs; the Dafny contracts use axioms for convergence.

9.5.1 Self-Correction During Revision: A Harness-in-Action Example

Rev. 2

An instructive incident occurred during the revision process itself. The initial draft of the formal specification (Section 3.4) claimed that the LTCA solver uses "bisection and secant methods" with a contraction-mapping convergence proof. When the researcher verified this claim against the actual source code (`runTCA_analysis.m`, lines 239–255), it was discovered that the solver actually uses a *scan-and-fzero* strategy. The initial claim was an AI-generated hallucination that sounded plausible but did not match the implementation.

This error was caught because the harness requires formal specifications to be *code-verified*: preconditions, postconditions, and convergence claims must reference specific line numbers in the actual source code. This incident is itself evidence for the harness principle: without the requirement to link claims to code, the error would have propagated into the published version.

This review–revision trace is itself an instance of the harness principle: the review is recorded as structured evidence, the response is linked to specific sections, and the remaining limitations are declared as machine-readable boundaries.

10 Harness Reuse Protocol

The harness can be reused as follows:

1. Define the research object, valid range, excluded physics, and artifact set in a research card.
2. Write a structured abstract that states the problem, intervention, framework, and demonstration.
3. Build a claim-evidence map before expanding the narrative prose.
4. Convert references into a dense literature evidence matrix before writing a conventional related-work section.
5. Attach an algorithm passport, pseudocode, and flowchart to each central computational procedure.
6. Attach a formula passport to each central equation.
7. Map each major code file to its research role and inspection question.
8. Convert the method section into a model-evolution log when the work involves iterative model correction.
9. Replace ordinary figure captions with claim-linked captions that state claim, inspection target, and regeneration path.
10. Record AI involvement as compressed interaction traces.
11. Use Git commits as evidence checkpoints.
12. Add an anti-contamination audit that checks citation support, formula units, code consistency, figure provenance, failure cases, and evidence strength.
13. State limits as machine-readable boundaries whenever possible.

10.1 Minimum Viable Harness

Rev. 2

The full harness described above involves 13 components. For researchers adopting this approach for the first time, this represents a significant workflow change. We therefore define a *Minimum Viable Harness* (MVH): the five components that provide the majority of auditability benefit at minimum cost.

1. **Claim-evidence map**. A table linking each major claim to its supporting artifact. This single component forces the researcher to make the evidence structure explicit, and it enables AI systems to verify the logical chain. Estimated overhead: 1–2 hours per paper.
2. **Algorithm passport**. A structured description of each central algorithm, including inputs, outputs, and key assumptions. This prevents the common failure mode where an algorithm is described in prose but its boundary conditions are implicit. Estimated overhead: 30–60 minutes per algorithm.
3. **Structured abstract**. The four-sentence format (problem, intervention, framework, demonstration) forces the researcher to state the contribution precisely, which is the single most valuable change for AI readability. Estimated overhead: negligible (replaces free-form abstract writing).
4. **Code-artifact map**. A one-paragraph description of each major code file, its role, and what to inspect. This is the minimum required for an AI system to navigate the codebase. Estimated overhead: 15–30 minutes per paper.
5. **Limits declaration**. An explicit statement of what the paper does *not* claim, stated as machine-readable boundaries. This is the cheapest form of anti-contamination: it prevents downstream systems from over-generalizing the results. Estimated overhead: 15 minutes.

The MVH adds approximately 3–5 hours of work to a typical paper, which is a 15–20% overhead on a 20–30 hour writing process. The remaining eight components (dense literature matrix, formula passport, anti-contamination audit, AI-interaction trace, etc.) can be adopted incrementally. The key principle is: *start with the claim-evidence map, and add components as the research workflow matures.*

10.2 Cost-Benefit Analysis: Harness vs. Conventional Format

Rev. 2

Table 15 compares the time investment and output quality of the conventional IMRaD format, the MVH, and the full harness.

The MVH rows are estimated from the claim-audit experiment (Section 4.5) by retaining only the components that directly contribute to the top-scoring tasks (claim extraction and evidence linking). A full validation of the MVH remains future work.

11 Conclusion

In the AI era, the research paper should be engineered, not merely written. It should become a semantic artifact harness: an inspectable research interface in which claims are linked to assumptions, algorithms, formulas, code, figures, scripts, citations, correction traces, limits, and anti-contamination checks. The engineering discipline is borrowed from software engineering—contracts, test harnesses, API specifications, separation of concerns, supply-chain integrity, and CI/CD verification—but the product remains a reshaped research paper, not a software system. This paper proposed such a harness and demonstrated it through an LTCA case substrate for cycloid-pin-housing systems in robotic RV reducers.

The central claim is that AI-native research writing should not merely use AI to draft conventional prose. A stronger form is possible: redesign the paper itself so that humans can read the argument, AI systems can parse the evidence structure, and both can inspect how algorithms, code, figures, and human corrections support the stated claims. The reasoning process—interaction traces, model-evolution logs, correction records—is not waste material but first-class evidence that makes conclusions inspectable. The three-layer architecture of the harness preserves both the narrative judgment and the audit trail, enabling verification by human or AI readers in an environment where AI can generate plausible conclusions at scale.

12 Outlook: Toward Self-Evolving Research Systems

The harness proposed in this paper is a *static* architecture: claims, evidence, algorithms, and code are connected through human-designed interfaces, and every change is audited through explicit protocols. This is sufficient for the current generation of AI-assisted research, where the AI is a powerful but passive tool. But a deeper question emerges: *can the research system itself learn and improve through use, rather than relying on periodic model retraining?*

This section examines the key bottlenecks, partial solutions, and open problems on the path from static harnesses to self-evolving research agents.

12.1 The Knowledge Representation Bottleneck

Current LLMs store knowledge in two forms. *Explicit* knowledge is encoded in human-readable files—papers, code, documentation, harness records. *Implicit* knowledge is encoded in model weights through training. The former is auditable but inert; the latter is fluent but opaque and frozen after training.

The harness in this paper operates entirely in the explicit-file domain. This is by design: file-based artifacts are inspectable, version-controllable, and can be shared across AI systems through protocols such as MCP. But it also means that the system cannot *learn* from its interactions. Every new session begins from the same knowledge state, regardless of how many previous sessions successfully solved similar problems.

In human research, both forms coexist. A researcher writes papers and notes (explicit), but also accumulates intuition, judgment, and pattern recognition through years of practice (implicit, encoded in neural connections). The implicit knowledge allows a human to recognize promising research directions, spot anomalies in data, and make leaps of reasoning that cannot be fully articulated in prose. Current AI systems—even with harness-grade structure—lack this implicit learning channel.

Additional directions—self-evolution without retraining, end-to-end computer control, the domain gap in AI-assisted engineering, and multi-agent collaboration—are discussed in Appendix D.

12.2 Open Questions

Table 16 summarizes the key open problems. Each represents a research direction that the harness framework alone cannot solve but that any future self-evolving research system must address.

The gap between current AI-assisted research and a fully self-evolving research ecosystem is not primarily a gap in model capability. It is a gap in *architecture*: the missing feedback loop from execution to learning, the absence of domain-specific perception channels, and the lack of structured protocols for multi-agent collaboration. The Semantic Artifact Harness addresses the *auditability* side of this gap; the *evolution* side remains open.

Code and Data Availability

Rev. 2
Rev. 3

The complete MATLAB implementation of the LTCA solver, including the five-case parametric study and all formal verification artifacts, is publicly available at:

`https://github.com/haomjc/sah-ltca-cycloid` (`commit 7f49bf7`)

The repository contains seven core MATLAB functions (`run_parametric_study.m`, `runTCA_analysis.m`, `fun_force_analysis.m`, `fun_findtc.m`, `fun_output_mod.m`, `fun_tooth_stiffness.m`, `export_analysis_data.m`) and three formal verification artifacts:

1. **JSON Schema** (`verification/sah-ltca.schema.json`): A machine-readable schema defining the SAH harness structure, including typed interfaces for research cards, claim-evidence maps, algorithm passports, formula passports, and verification reports.
2. **MATLAB runtime assertions**: The function `fun_force_analysis.m` includes SAH-branded precondition checks (parameter range validation at function entry), postcondition checks (force non-negativity, backside zeroing, residual finiteness), and a convergence warning when the pin-hole iteration exhausts its maximum iteration budget.
3. **Dafny contract sketch** (`verification/ltca_contracts.dfy`): A partial formalization of the force-balance core, including datatype definitions, predicates for force invariants, method signatures with pre/postconditions, and a convergence lemma declared as an axiom with a proof sketch.

Required toolbox: MATLAB Symbolic Math Toolbox (used by `fun_output_mod.m` for contact geometry solving).

Acknowledgments

The author thanks Kimi (Moonshot AI), Zhipu AI, and MiniMax for providing their latest large language models and generous coding and token plans, which made this work possible. Special thanks are extended to the GLM-5.1 model (Zhipu AI), whose capabilities in code generation, mathematical reasoning, and long-context understanding substantially contributed to the iterative model development, manuscript drafting, and harness construction described in this paper. The author also thanks the OpenCode project for providing the open-source AI coding environment through which these models were accessed and orchestrated.

A Algorithmic Details of the LTCA Solver

This appendix provides the detailed pseudocode for the three central algorithmic modules of the loaded tooth contact analysis: the overall displacement and force solver, the tooth-body stiffness computation via the energy method, and the friction and working-side determination. These algorithms correspond to the code artifacts `runTCA.m`, `fun_tooth_stiffness.m`, and the friction logic inside `fun_force.m` respectively.

Importantly, the pseudocode in this appendix was not written manually from scratch. It was generated by an AI assistant that inspected the MATLAB source files, analyzed the control flow, identified the mathematical operations, and reformulated the implementation into formal `algorithmic` pseudocode. The AI

Algorithm 1 DCA: Unloaded Contact Geometry

Require: Geometry $(R_p, R_{rp}, A, N_c, N_p, r_{zo})$, input angle range $[\theta_{\text{start}}, \theta_{\text{end}}]$, step $\Delta\theta$
Ensure: Unloaded output angles **TO**, contact parameters **TC**, contact phases **TP**

$A_p \leftarrow 2\pi/N_p$ ▷ Angular pin spacing
$\iota \leftarrow |N_p - N_c|/N_c$ ▷ Transmission ratio
for $j = 1$ **to** num_steps **do**
  $\theta_{\text{in}}^{(j)} \leftarrow \text{deg2rad}(\theta_{\text{start}} + (j-1)\Delta\theta + 90)$
  $\theta_{\text{out, theo}}^{(j)} \leftarrow \theta_{\text{in}}^{(j)} \cdot \iota$
  for $i = 1$ **to** N_p **do**
    $\varphi_i \leftarrow (i-1) \cdot A_p$ ▷ Pin angular position
    Compute initial contact phase $\hat{\psi}_i$ from geometry
    Find tooth-contact parameter $\hat{\tau}_i$ via `fun_findtc`
    $\mathbf{x}_0 \leftarrow [\theta_{\text{out, theo}}^{(j)},\ \hat{\tau}_i,\ \hat{\psi}_i]$ ▷ Initial guess
    $[\mathbf{S},\ \text{exitflag}] \leftarrow \text{fsolve}(\mathbf{x}_0)$ ▷ Solve contact equations
    if exitflag > 0 **then**
      $TO(j,i) \leftarrow S_1$, $TC(j,i) \leftarrow S_2$, $TP(j,i) \leftarrow S_3$
    else
      $TO(j,i) \leftarrow \text{NaN}$, $TC(j,i) \leftarrow \text{NaN}$, $TP(j,i) \leftarrow \text{NaN}$
    end if
  end for
end for

was given the source files as input and was instructed to extract the algorithmic core—stripping MATLAB-specific syntax while preserving the logical structure, mathematical formulas, and decision points. The researcher then reviewed the generated pseudocode for accuracy, corrected a small number of sign conventions and indexing details, and approved the final version. This process is itself an instance of the harness principle at work: the appendix is not only evidence of the algorithm, but also evidence of how AI-assisted documentation was produced, audited, and corrected.

A.1 Overall Displacement and Force Solution

The overall solver operates in two stages. In the first stage (DCA), the unloaded contact geometry is computed for each input angle and each pin. In the second stage (LTCA), the loaded output displacement is found by solving a torque-equilibrium equation.

The force residual function $F(\delta_{\text{out}})$ in line 4 of Algorithm 2 encapsulates the contact force model for all pins simultaneously. For a given trial output displacement δ_{out}, it computes the normal contact force on each pin, determines which pins belong to the working side through the torque-direction test (Algorithm 4), and returns the torque imbalance. The solver converges when this imbalance reaches zero.

A.2 Tooth Body Stiffness via Energy Method

The tooth-body structural stiffness K_g accounts for bending, shear, and compression compliance of the cycloid tooth between the contact point and the tooth root. It is computed once per (angle, pin) pair and remains constant during the LTCA iteration.

The coordinate transformation LocalCoords(φ) proceeds as follows. First, the raw epicycloid coordinates are computed:

$$\begin{aligned} x_0(\varphi) &= R_z\left[\sin\varphi - \frac{K_1}{N_p}\sin(N_p\varphi)\right], \\ y_0(\varphi) &= R_z\left[\cos\varphi - \frac{K_1}{N_p}\cos(N_p\varphi)\right]. \end{aligned} \tag{2}$$

The outward unit normal is $(n_x, n_y) = (K_1 \sin N_p\varphi - \sin\varphi,\ -K_1\cos N_p\varphi + \cos\varphi)/\|\cdot\|$. The equidistant

Algorithm 2 LTCA: Loaded Force-Balance Solver

Require: DCA results $(\mathbf{TO}, \mathbf{TP})$, stiffness matrix $\mathbf{K}_g$, applied torque T_{in}, geometry parameters
Ensure: Contact forces $\mathbf{F}_p$, loaded output angles $\theta_{\text{out, loaded}}$, transmission error $\Delta\phi$

1: **for** $j = 1$ **to** num_steps **do**
2: Compute backlash: $BL(j,i) \leftarrow TO(j,i) - \theta^{(j)}_{\text{out, theo}}$ for each valid pin i
3: $\mathbf{TP}_j \leftarrow TP(j,:)$, $\mathbf{BL}_j \leftarrow BL(j,:)$, $\mathbf{K}_{g,j} \leftarrow K_g(j,:)$
4: Define residual function: $F(\delta_{\text{out}}) \leftarrow T_{\text{in}} - \sum_i f_i(\delta_{\text{out}}; \mathbf{TP}_j, \mathbf{BL}_j, \mathbf{K}_{g,j})$
// f_i is computed by the force model (Algorithm 4)
5: Scan $\delta_{\text{out}} \in [-10^{-4}, 10^{-4}]$ for sign change of F
6: **if** sign change found in narrow scan **then**
7: Bracket $\leftarrow$ interval containing root
8: **else**
9: Extend scan to $\delta_{\text{out}} \in [-0.01, 0.01]$
10: **end if**
11: $\delta^*_{\text{out}} \leftarrow \text{fzero}(F, \text{Bracket})$ ▷ Bracketed root search
12: Extract $F_p(j,:), H_b(j,:), k(j,:), H_z(j,:)$ at δ^*_{out}
13: $\theta^{(j)}_{\text{out, loaded}} \leftarrow \theta^{(j)}_{\text{out, theo}} - \delta^*_{\text{out}}$
14: $\Delta\phi^{(j)} \leftarrow \text{rad2deg}(\delta^*_{\text{out}}) \times 3600$ ▷ Transmission error in arcsec
15: **end for**

(offset) curve adds the generation radius R_c plus the modification amount $\Delta R_c(\varphi)$ along this normal:

$$x_1 = x_0 + (R_c + \Delta R_c)\, n_x, \qquad y_1 = y_0 - (R_c + \Delta R_c)\, n_y. \tag{3}$$

The local coordinate system is obtained by rotating by the half-pitch angle β and translating so that the tooth root lies on the y-axis:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - (R_z - R_c - A)\cos\beta \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{4}$$

In the local frame, x_2 represents the tooth half-thickness and y_2 measures distance along the tooth centerline from the root. The bending moment arm M in line 17 of Algorithm 3 is the perpendicular distance from the contact force line of action to the cross-section at height y_2.

A.3 Friction and Working-Side Determination

The friction model uses static Coulomb friction projected through an equivalent torque arm. The working side—the set of pins that provide torque in the same direction as the input torque—is determined by a 2D cross-product test on the moment arm and force direction.

The pitch point P in line 3 is the instantaneous center of relative rotation between the cycloid disk and the pin housing. Its position $P = N_p A \cdot [\cos\theta_{\text{in}}, \sin\theta_{\text{in}}]$ follows from the kinematic constraint that the cycloid disk rotates about the eccentric center with a one-tooth-phase offset from the input crank.

The slip direction (line 9) is determined by projecting the velocity of the contact point relative to the pitch point onto the contact normal. This gives the instantaneous tendency for the cycloid tooth to slide relative to the pin. The static friction force (line 12) opposes this tendency and is bounded by $\mu \cdot F_n$.

The working-side test (line 16) uses the 2D cross product of the position vector $\mathbf{r}$ (from eccentric center to pin center) with the normal force direction $\mathbf{F}_{\text{normal}}$. Pins for which this cross product is positive contribute torque in the same direction as the input torque T_{in} and are retained as load-bearing pins; pins with negative cross product are on the back side and assigned zero force.

The effective moment arm (line 20) combines the normal-force moment arm τ_i with the friction moment arm $m_{\text{fric},i}$. Both terms have consistent sign conventions because they share the same torque reference direction. The torque balance in the LTCA solver (Algorithm 2) then reads:

$$T_{\text{in}} = \sum_{i \in \text{working side}} F_{p,i} \cdot \ell_{\text{eff},i}. \tag{5}$$

Algorithm 3 Energy-Method Tooth-Body Stiffness K_g

Require: Contact parameter φ_0, geometry (R_z, A, N_c, N_p, R_c), material (E, ν), modification parameters
Ensure: Tooth-body stiffness K_g (N/mm)

$\beta \leftarrow \pi / N_c$ ▷ Half-pitch angle
$K_1 \leftarrow A \cdot N_p / R_z$ ▷ Short-width coefficient
$G \leftarrow E/(2(1+\nu))$ ▷ Shear modulus
$\varphi_{\min} \leftarrow \pi / N_c^2, \quad \varphi_{\max} \leftarrow (N_c - 1)\pi / N_c^2$
$\varphi_0 \leftarrow \mathrm{clamp}(\varphi_0, \varphi_{\min}, \varphi_{\max})$

Compute contact-point quantities:
$(x_2^0, y_2^0) \leftarrow \mathrm{LocalCoords}(\varphi_0)$ ▷ Eq. 4
$(dx_2^0, dy_2^0) \leftarrow \partial(x_2, y_2)/\partial\varphi$ at φ_0 via central difference
$\alpha_0 \leftarrow \pi/2 - \mathrm{atan2}(dy_2^0, dx_2^0)$ ▷ Pressure angle at contact

Numerical integration from $\varphi_{\min}$ to φ_0:
for each quadrature point φ **do**
  $(x_2, y_2, \cdot, dy_2/d\varphi) \leftarrow \mathrm{LocalCoords}(\varphi)$
  $M \leftarrow (y_2^0 - y_2)\cos\alpha_0 + x_2^0 \sin\alpha_0$ ▷ Bending moment arm
  $A_s \leftarrow 2|x_2| \cdot B$ ▷ Cross-section area
  $I \leftarrow \frac{2}{3}|x_2|^3 \cdot B$ ▷ Moment of inertia
  // Three compliance components (energy method):
  $\frac{1}{K_b}$ += $\frac{3M^2}{2EI}\left|\frac{dy_2}{d\varphi}\right|$ ▷ Bending
  $\frac{1}{K_s}$ += $\frac{1.2\cos^2\alpha_0}{2GA_s}\left|\frac{dy_2}{d\varphi}\right|$ ▷ Shear
  $\frac{1}{K_c}$ += $\frac{\sin^2\alpha_0}{2EA_s}\left|\frac{dy_2}{d\varphi}\right|$ ▷ Compression
end for

$K_g \leftarrow 1 \Big/ \left(\frac{1}{K_b} + \frac{1}{K_s} + \frac{1}{K_c}\right)$

B Formal Specification and Verification Artifacts

This appendix provides the full formal specification of the LTCA force-balance solver that was summarized in Section 3.6. The specification includes code-verified preconditions, postconditions, root-finding strategy, pin-hole iteration protocol, and numerical stability analysis. All assertions were implemented as MATLAB runtime checks in `fun_force_analysis.m` and are publicly available at `https://github.com/haomjc/sah-ltca-cycloid`.

B.1 Preconditions (SAH_PRE)

The following parameter range checks execute at function entry:

Precondition Contracts (SAH_PRE)

1. Parameter vector `z` must have ≥ 17 elements.
2. All geometry and load parameters $(A, R_{p,\mathrm{profile}}, R_{rp}, N_c, T_{\mathrm{in}}, B, r_{zo}, L_{\mathrm{pin}}, R_{rp,\mathrm{pin}}) > 0$.
3. Elastic modulus $E > 0$.
4. Poisson ratio $0 \leq \nu \leq 0.5$.
5. Friction coefficient $\mu \geq 0$.
6. Contact phase array `TP` and backlash array `BL` are non-empty and have equal length.

Algorithm 4 Friction and Working-Side Determination

Require: Contact phases **TP**, input angle θ_{in}, eccentricity A, pin centers $(X_{S,i}, Y_{S,i})$, actual pin radius $R_{rp,\text{pin}}$, friction coefficient μ
Ensure: Working-side mask $\mathbf{w}$, friction moment arms $\mathbf{m}_{\text{fric}}$, effective moment arms $\boldsymbol{\ell}_{\text{eff}}$

Pitch point and eccentric center:
$X_0 \leftarrow A\cos\theta_{\text{in}}, \quad Y_0 \leftarrow A\sin\theta_{\text{in}}$ ▷ Eccentric center
$P_x \leftarrow N_p A\cos\theta_{\text{in}}, \quad P_y \leftarrow N_p A\sin\theta_{\text{in}}$ ▷ Pitch point (node)

for each pin i with valid contact **do**
  Contact point geometry:
  $K_x \leftarrow X_{S,i} - R_{rp,\text{pin}}\cos(TP_i), \quad K_y \leftarrow Y_{S,i} - R_{rp,\text{pin}}\sin(TP_i)$
  Slip direction from pitch-point kinematics:
  $\overrightarrow{PK} \leftarrow (K_x - P_x,\ K_y - P_y)$ ▷ Node to contact point
  $V_{\text{proj}} \leftarrow PK_y\sin(TP_i) - PK_x\cos(TP_i)$
  $S_{\text{slip}} \leftarrow \text{sign}(V_{\text{proj}})$ ▷ Slip tendency direction
  Friction unit vector (opposing slip):
  $\hat{t} \leftarrow (\sin TP_i,\ -\cos TP_i)$ ▷ Tangent direction at contact
  $\hat{f}_{\text{fric}} \leftarrow -S_{\text{slip}} \cdot \hat{t}$ ▷ Opposing relative sliding
  Torque direction via 2D cross product:
  $\mathbf{r} \leftarrow (X_{S,i} - X_0,\ Y_{S,i} - Y_0)$ ▷ Pin center − eccentric center
  $\mathbf{F}_{\text{normal}} \leftarrow (-\cos TP_i,\ -\sin TP_i)$ ▷ Pin-to-cycloid force
  $\tau_i \leftarrow r_x F_y - r_y F_x$ ▷ $\mathbf{r} \times \mathbf{F}_{\text{normal}}$
  $w_i \leftarrow (\tau_i > 0) \wedge \text{valid}(TP_i)$ ▷ Working-side pin
  Friction moment arm:
  $\mathbf{r}_K \leftarrow (K_x - X_0,\ K_y - Y_0)$ ▷ Contact point − eccentric center
  $m_{\text{fric},i} \leftarrow \mu \cdot (r_{K,x}\hat{f}_{\text{fric},y} - r_{K,y}\hat{f}_{\text{fric},x})$
  $\ell_{\text{eff},i} \leftarrow \tau_i + m_{\text{fric},i}$ if w_i, else 0
end for
return $\mathbf{w}$, $\mathbf{m}_{\text{fric}}$, $\boldsymbol{\ell}_{\text{eff}}$

B.2 Postconditions (SAH_POST)

The following invariants are checked before the function returns results:

Postcondition Contracts (SAH_POST)

1. **Force non-negativity**: $F_{p,i} \geq 0$ for all pins i (tolerance: -10^{-6}).
2. **Backside zeroing**: $F_{p,i} = 0$ for all non-working-side pins.
3. **Residual finiteness**: torque balance residual $|EQ_1|$ is finite (not NaN or Inf).
4. **Hertz validity**: Hertz half-width $H_b \geq 0$ and contact stress $H_z \geq 0$ for all active contacts.

B.3 Root-Finding Strategy

The torque equilibrium equation

$$EQ_1 = T_{\text{in}} - \sum_{i \in \text{working}} F_{p,i}(\delta_{\text{out}}) \cdot \ell_{\text{eff},i} \tag{6}$$

is solved for the output displacement δ_{out} using MATLAB `fzero` with a bracketed search. The bracket is expanded adaptively: if the initial bracket $[\delta_{\text{lo}}, \delta_{\text{hi}}]$ does not straddle a root, the upper bound is doubled (up to 10 expansions) until a sign change is detected. If no sign change is found, the solver falls back to the midpoint and issues a convergence warning.

B.4 Pin-Hole Iteration Protocol

When pin-housing elastic displacement is enabled, the solver iterates on the pin-hole deformation δ_{ph}:

1. Initialize $\delta_{ph} = 0$ for all pins.
2. At each iteration, compute new deformations $\delta_{ph,\text{new}}$ from the Hertz compliance model.
3. Check convergence: $\max_i |\delta_{ph,\text{new},i} - \delta_{ph,i}| < 10^{-6}$.
4. If converged, accept solution; otherwise, update $\delta_{ph} \leftarrow \delta_{ph,\text{new}}$ and repeat.
5. Maximum iterations: 50. If exhausted, emit `SAH_CONV` warning with residual value.

B.5 Numerical Stability Considerations

- **Clearance singularity**: When pin-housing clearance $\delta_{clr} \to 0$, the transition from rigid gap to elastic contact is handled by a threshold check ($\delta_{clr} < 10^{-4}$ mm) rather than a smooth transition function. This prevents numerical oscillation near the contact threshold.
- **Zero-force pins**: Pins with $F_{p,i} < 10^{-6}$ N are explicitly set to zero force and excluded from the torque balance. This prevents accumulation of floating-point noise.
- **Hertz validity**: The Hertz half-width formula $H_b = \sqrt{4F_pR'/(\pi E'B)}$ requires $F_p > 0$ and $R' > 0$. Pins that fail either condition are assigned $H_b = 0$, $H_z = 0$.
- **Stiffness fallback**: If the energy-method tooth-body stiffness computation fails (e.g., due to singular integration), the solver falls back to rigid-body stiffness ($K_g \to \infty$) with a `try/catch` guard.

B.6 Dafny Contract Sketch

A partial formalization of the force-balance core in the Dafny verification language is provided in the repository (`verification/ltca_contracts.dfy`). It includes:

- A `PinState` datatype with fields for position, force, deformation, and working-side flag.
- Predicates `ForceNonNegative`, `BacksideZeroForce`, and `ForceBalanceInvariant`.
- A `ComputePinForces` method signature with full pre/postconditions.
- A convergence lemma declared as an axiom, with a proof sketch based on the Banach fixed-point theorem.

Note: The Dafny contracts are a *sketch*, not a complete verified implementation. The convergence lemma uses an axiom because the contraction property depends on domain-specific Hertz model analysis. The MATLAB runtime assertions provide the executable verification layer; the Dafny sketch provides the specification layer.

C Parametric Study Figures

This appendix contains the complete parametric study figures referenced in Section 5.1.1. All figures were generated from the five-case comparative study (Ideal Rigid / Friction / Gear Stiffness / Pin Displacement / Fully Coupled) at 0° input rotation. Data files are available in the repository `data/` directory.

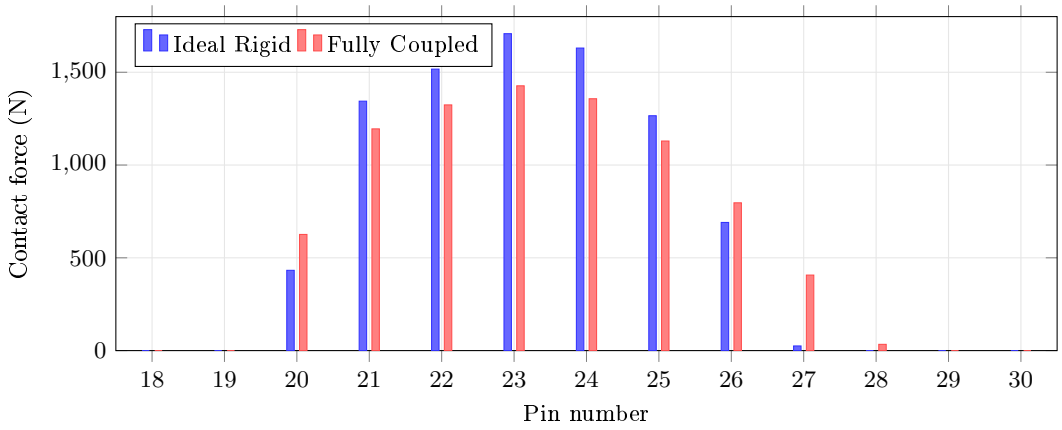


Figure 5: **Load distribution at 0° rotation (zoomed).** Comparison of ideal rigid and fully coupled models. The fully coupled model redistributes load: peak force drops from 1707.9 N to 1427.0 N (Pin 23), while peripheral pins (27–28) gain load due to pin-hole elastic retreat.

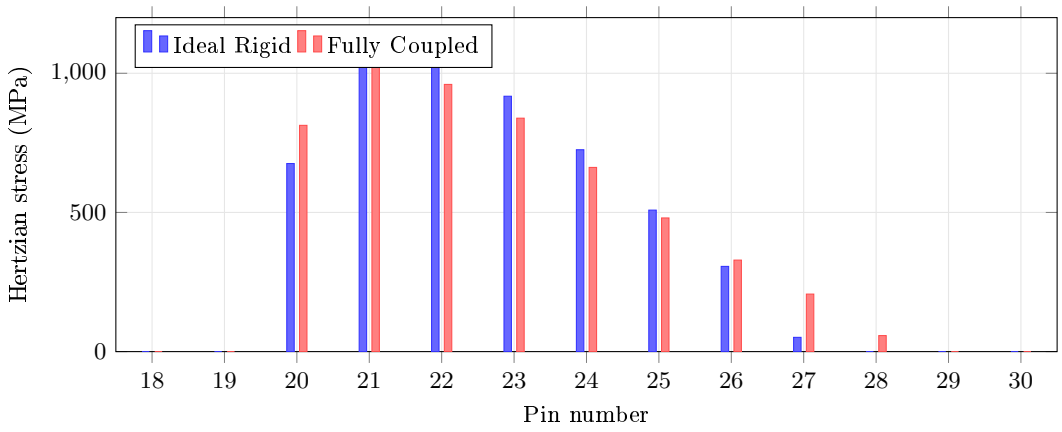


Figure 6: **Hertzian contact stress at 0° rotation (zoomed).** The fully coupled model reduces peak stress from 1096.9 MPa (Pin 21) to 1034.2 MPa, while stress at peripheral pins (27–28) increases due to load redistribution.

D Future Research Directions

This appendix expands on the directions outlined in Section 12. Each subsection explores a specific bottleneck on the path from static harnesses to self-evolving research systems.

D.1 Self-Evolution Without Retraining

Current AI systems improve only when retrained on new data. Between training cycles, the model is frozen: it cannot learn from successful interactions, corrected mistakes, or domain-specific patterns encountered during use. This creates a fundamental mismatch with research practice, where expertise accumulates continuously through experiment, failure, and revision.

Several partial solutions exist. *Retrieval-augmented generation* (RAG) allows a model to consult external documents at inference time, but the model weights themselves remain unchanged. *In-context learning* enables the model to adapt within a single conversation, but the adaptation is lost when the session ends. *Fine-tuning on interaction logs* can encode session-level improvements, but requires explicit training infrastructure and risks catastrophic forgetting.

The Semantic Artifact Harness contributes to this problem by producing *persistent, structured interaction*

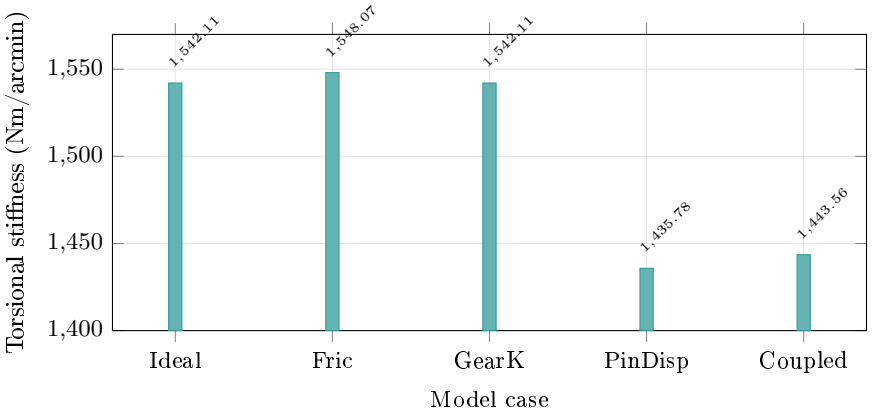


Figure 7: **Torsional stiffness comparison.** Pin-hole elastic displacement is the dominant compliance source, reducing stiffness from 1542.1 to 1435.8 Nm/arcmin (PinDisp case). Friction slightly increases stiffness (1548.1 Nm/arcmin) due to load redistribution. The fully coupled case (1443.6 Nm/arcmin) is close to PinDisp alone.

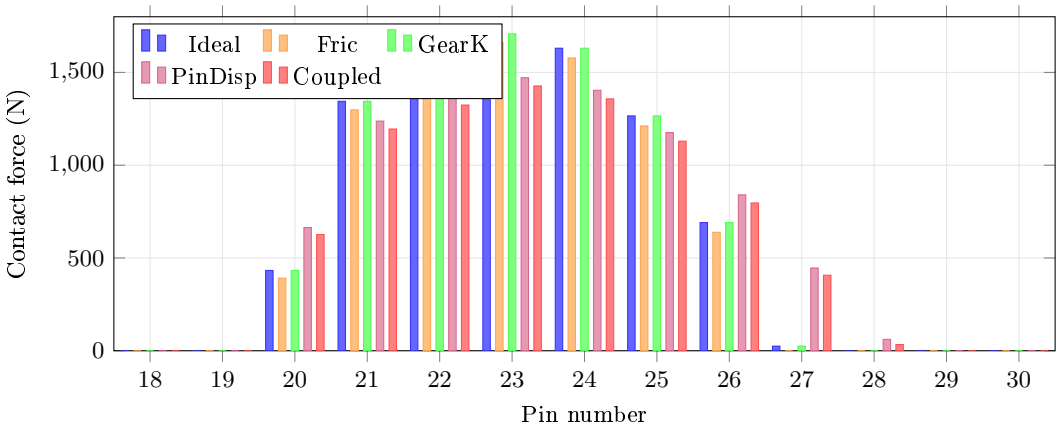


Figure 8: **Five-case load distribution comparison (zoomed).** Pin-hole displacement (Case 4) shifts load toward peripheral pins and reduces peak force. Friction (Case 2) has a smaller effect. Gear stiffness alone (Case 3) produces results identical to the ideal rigid model at 0° rotation.

records: correction logs, model-evolution traces, and claim-evidence maps that survive across sessions. These records form a natural training signal for future adaptive systems. The key open question is whether locally maintained adapters—small, task-specific weight updates—can encode interaction experience without the cost and risk of full retraining.

D.2 End-to-End Computer Control

The present case demonstrates AI-agent interaction with MATLAB code through a command-line interface. A more ambitious goal is *end-to-end control*: an AI agent that operates engineering software through the same visual interface a human uses, but with machine-readable audit trails. This would eliminate the need for software-specific API bridges and allow the agent to interact with any GUI-based tool.

Recent progress in vision-language models (VLMs) and native visuomotor agents suggests this is becoming feasible. Models that accept screen images and output keyboard/mouse events can, in principle, operate any software without tool-specific integration. The Semantic Artifact Harness would then serve as the *evidence layer*: every GUI interaction is recorded as a structured claim ("I set parameter X to value Y in dialog Z"), and the full interaction sequence becomes an inspectable trace.

The current bottleneck is reliability. GUI-based agents are prone to mis-clicks, misreadings, and context loss. For engineering applications where a wrong parameter can invalidate hours of computation, the error tolerance is much lower than for web browsing or document editing. The harness addresses this by requiring *post-hoc verification*: even if the agent makes a mistake during interaction, the claim-evidence map forces the researcher to verify that the final results are consistent with the stated assumptions.

D.3 The Domain Gap in AI-Assisted Engineering

General-purpose LLMs are trained on broad internet corpora that include scientific papers, code repositories, and technical documentation. This gives them strong surface-level fluency in engineering language. But surface fluency is not domain competence. A model can write a correct-sounding paragraph about Hertz contact theory while making subtle errors in sign conventions, unit conversions, or boundary condition interpretations that an experienced engineer would catch immediately.

The domain gap is especially dangerous in computational engineering because errors are often *locally plausible*: a formula with a wrong sign or a missing factor of 2 can produce numerically reasonable but physically wrong results. The harness addresses this through *claim-evidence contracts* and *formula passports* that force explicit boundary conditions, but these are mitigations, not solutions.

The deeper solution requires domain-specific foundation models or self-evolving agents that learn through use in a particular engineering domain. The interaction logs produced by the harness (correction traces, model-evolution logs) are precisely the data such models would need. The open question is whether transfer learning from general-purpose models to domain-specific agents can be made efficient enough to be practical.

D.4 Multi-Agent Collaboration

Single-agent systems have fixed blind spots determined by their training data and architecture. Multi-agent systems—in which multiple AI agents with different specializations collaborate on a research task—could address this by introducing structured disagreement. For example, one agent might specialize in code generation while another focuses on physical consistency checking; a third might serve as an adversarial reviewer.

The Semantic Artifact Harness provides a natural substrate for multi-agent collaboration because it defines *typed interfaces* between artifacts: claims, evidence, algorithms, code, and figures are all structured objects with explicit schemas. Agents can interact through these interfaces without needing to parse free-form prose. The claim-evidence map, in particular, serves as a shared workspace: one agent proposes a claim, another provides evidence, and a third verifies the link.

The key risk is *bias amplification*: if agents share similar training data or architectures, their agreement may reflect shared biases rather than independent verification. True multi-agent consensus requires agents with genuinely different perspectives, which in turn requires different training data, different architectures, or different verification strategies. The anti-contamination protocol described in Section 8.1 is a first step toward this, but structured multi-agent verification remains an open research direction.

References

Gerasimos Chourdakis, Kyle Davis, Benjamin Frings, Rajeev K. Jaiman, Maria Lukacova-Medvidova, Kilian Scheufele, Arushi Shuka, Benjamin Uekermann, Miriam Utz, and Philipp Vollmer. preCICE v2: A sustainable and user-friendly coupling library. *Open Research Europe*, 2:51, 2022. doi:10.12688/openreseurope.14445.2.

OpenCFD Ltd. OpenFOAM: Open field operation and manipulation, 2024. URL `https://www.openfoam.com/`. Version 12.

Guido Dhondt. CalculiX: A free software three-dimensional structural finite element program, 2024. URL `http://www.calculix.de/`. Version 2.21.

Johannes Holzinger and Johannes Gerstmayr. Exudyn: A flexible multibody dynamics simulation python library. *SoftwareX*, 26:101700, 2024. doi:10.1016/j.softx.2024.101700.

Jonathon W. Sensinger. Unified approach to cycloid drive profile, stress, and efficiency optimization. *Journal of Mechanical Design*, 132(2):024503, 2010. doi:10.1115/1.4000832.

L. Ivanović, G. Devedžić, S. Ćuković, and N. Mirić. Modeling of the meshing of trochoidal profiles with clearances. *Journal of Mechanical Design*, 134(4):041003, 2012. doi:10.1115/1.4006181.

W. S. Lin, Y. P. Shih, and J. J. Lee. Design of a two-stage cycloidal gear reducer with tooth modifications. *Mechanism and Machine Theory*, 79:184–197, 2014. doi:10.1016/j.mechmachtheory.2014.04.006.

Lixin Xu and Yuhu Yang. Dynamic modeling and contact analysis of a cycloid-pin gear mechanism with a turning arm cylindrical roller bearing. *Mechanism and Machine Theory*, 104:327–349, 2016. doi:10.1016/j.mechmachtheory.2016.06.018.

Yun Yang, Chaoyang Li, and Xuan Li. Theoretical study on meshing stiffness of cycloid-pin gear drive based on energy method. *Journal of Mechanical Transmission*, 42(9):7–12, 2018. doi:10.16578/j.issn.1004.2539.2018.09.002.

Shyi-Jeng Tsai and Ching-Hao Huang. A study on loaded tooth contact analysis of a cycloid planetary gear reducer considering friction and bearing roller stiffness. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 11(6):JAMDSM0077, 2017. doi:10.1299/jamdsm.2017jamdsm0077.

X. Li, C. Li, Y. Wang, and B. Chen. Analysis of a cycloid speed reducer considering tooth profile modification and clearance-fit output mechanism. *Journal of Mechanical Design*, 139(3):033303, 2017. doi:10.1115/1.4035541.

Li Xin Xu. A dynamic model to predict the number of pins to transmit load in a cycloidal reducer with assembling clearance. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 233(12):4247–4269, 2019. doi:10.1177/0954406218809732.

Hui Wang, Zhao-Yao Shi, Bo Yu, and Hang Xu. Transmission performance analysis of rv reducers influenced by profile modification and load. *Applied Sciences*, 9(19):4099, 2019. doi:10.3390/app9194099.

Xiaoxiao Sun, Liang Han, and Jian Wang. Tooth modification and loaded tooth contact analysis of china bearing reducer. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 233(17):5999–6016, 2019. doi:10.1177/0954406219858184.

Xiaoxiao Sun, Shuaijie Wang, Zihao Wang, Dongping Ji, Liang Han, Yanhui Xu, Cong Qiu, and Dongpeng Shan. Meshing performance analysis and optimization of china bearing reducer cycloid gear based on tooth contact analysis. *Machines*, 12(12):915, 2024. doi:10.3390/machines12120915.

Xiaoxiao Sun, Liang Han, and Jian Wang. Lost motion analysis of cbr reducer. *Mechanism and Machine Theory*, 120:89–106, 2018. doi:10.1016/j.mechmachtheory.2017.09.019.

Hang Xu, Zhaoyao Shi, Bo Yu, and Hui Wang. Dynamic measurement of the lost motion of precision reducers in robots and the determination of optimal measurement speed. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 13(3):JAMDSM0044, 2019a. doi:10.1299/jamdsm.2019jamdsm0044.

Li Xin Xu, Bing Kui Chen, and Chao Yang Li. Dynamic modelling and contact analysis of bearing-cycloid-pinwheel transmission mechanisms used in joint rotate vector reducers. *Mechanism and Machine Theory*, 137:432–458, 2019b. doi:10.1016/j.mechmachtheory.2019.03.035.

Xuan Li, Jiaqing Huang, Chuancang Ding, Ran Guo, and Weilong Niu. Dynamic modeling and analysis of an rv reducer considering tooth profile modifications and errors. *Machines*, 11(6):626, 2023a. doi:10.3390/machines11060626.

Xuan Li, Haidong Yang, Weilong Niu, Ran Guo, and Lining Sun. Investigation on tooth surface wear of cycloid drives considering tooth profile modifications. *Lubricants*, 11(8):323, 2023b. doi:10.3390/lubricants11080323.

Y. Wang, B. Wei, Z. Wang, J. Yang, and J. Xu. Research on loaded contact analysis and tooth wear calculation method of cycloid-pin gear reducer. *Lubricants*, 11(10):445, 2023. doi:10.3390/lubricants11100445.

Ke Zhang, Caixia Guo, Yutao Li, Yuewen Su, Bodong Zhang, and Peihu Gao. Contact analysis for cycloid pinwheel mechanism by isogeometric finite element. *Coatings*, 13(12):2029, 2023. doi:10.3390/coatings13122029.

Rui Li, Pengyuan Zheng, Gang Wang, Guodong Li, Shangkai Chi, and Xianghui Meng. Tribo-dynamics modeling of cycloidal gear-pin pair considering transient effects. *International Journal of Mechanical Sciences*, 305:110809, 2025. doi:10.1016/j.ijmecsci.2025.110809.

Luciana B. Sollaci and Mauricio G. Pereira. The introduction, methods, results, and discussion (imrad) structure: A fifty-year survey. *Journal of the Medical Library Association*, 92(3):364–367, 2004.

John M. Swales. *Genre Analysis: English in Academic and Research Settings*. Cambridge University Press, Cambridge, 1990.

Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984. doi:10.1093/comjnl/27.2.97.

Pieter Van Gorp and Steffen Mazanek. SHARE: A web portal for creating and sharing executable research papers. In *Proceedings of the International Conference on Computational Science*, volume 4 of *Procedia Computer Science*, pages 589–597, 2011. doi:10.1016/j.procs.2011.04.062.

Yann-Aël Le Borgne, Albert Adam, and Sebastian Maneth. Open review in computer science elsevier's grand challenge on executable papers. In *Proceedings of the International Conference on Computational Science*, volume 4 of *Procedia Computer Science*, pages 778–780, 2011. doi:10.1016/j.procs.2011.04.082.

Sean Bechhofer, David De Roure, Matthew Gamble, Carole Goble, and Iain Buchan. Research objects: Towards exchange and reuse of digital knowledge. In *The Future of the Web for Collaborative Science*, 2010.

Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016. doi:10.1038/sdata.2016.18.

Anthropic. Model context protocol, 2024. URL `https://modelcontextprotocol.io/`. Specification and reference implementations.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In *Automated Deduction – CADE-25*, pages 378–388. Springer, 2015. doi:10.1007/978-3-319-21401-6_26.

The mathlib Community. The lean mathematical library. *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 367–381, 2020. doi:10.1145/3372885.3373824.

Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004. doi:10.1038/nature02236.

Benjamin P. MacLeod, F. Grant L. Parlane, Thomas D. Morrissey, Florian Häse, Loïc M. Roch, Kevan E. Dettelbach, Rafael Moreira, Lars P. E. Yunker, Marianne B. Rooney, Jason R. Deeth, Vincent Lai, Greg J. Ng, Helen Situ, Rui H. Zhang, Michael S. Elliott, Thomas H. Haley, David J. Dvorak, Alán Aspuru-Guzik, Jason E. Hein, and Curtis P. Berlinguette. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867, 2020. doi:10.1126/sciadv.aaz8867.

Aldair E. Góngora, Bowen Xu, Wyatt Perry, Chika Okoye, Patrick Riley, Kevin G. Reyes, Elise F. Morgan, and Keith A. Brown. A bayesian experimental autonomous researcher for mechanical design. *Science Advances*, 6(15):eaaz1708, 2020. doi:10.1126/sciadv.aaz1708.

Table 8: Model-evolution log as a readable audit table. Each row links a model gap, the reason it mattered, the accepted correction, and the changed artifact.

Stage	Gap observed	Why it mattered	Harness correction	Changed artifact
V0	Baseline LTCA.	The model was useful, but assumptions were implicit.	Treat the codebase as an inspectable research object rather than only a calculation script.	Repository.
V1	Pin geometry mixed with generating geometry.	One radius represented multiple physical quantities.	Separate profile radius, actual pin radius, and housing-center radius.	`runTCA.m`; `fun_force.m`.
V2	High-tooth-count cases lost torque balance.	The torque-equilibrium root was too narrow for direct search.	Use scan-plus-`fzero` bracketing.	Solver path.
V3	Curvature change gave impossible stress.	AI confused profile-generation radius with actual pin radius.	Revert the substitution; use actual pin radius only in Hertz equivalent curvature.	Human correction.
V4	Hertz-only stiffness was too stiff.	Tooth-body compliance was missing.	Add energy-method tooth stiffness in series.	`fun_tooth_stiffness.m`.
V5	Tooth stiffness missed force direction.	Local projection and coordinates needed correction.	Include bending, shear, and compression with corrected projections.	Stiffness routine.
V6	Linearized Hertz stiffness was crude.	Contact stiffness depends on local load.	Iterate force and compliance inside LTCA.	`fun_force.m`.
V7	Friction could pollute normal compliance.	Tangential friction and normal deformation were mixed.	Project static friction through an equivalent torque arm.	Friction path.
V8	Non-contact placeholders entered calculations.	Sentinel values and zero-slip cases created artifacts.	Use `NaN` semantics and zero-slip guards.	Robustness patch.
V9	Floating-pin contact count became too small.	Clearance behaved like a contact-removal filter.	Consume clearance as rigid displacement loss, then iterate pin-housing elastic compliance.	Commit `5b4097b`.
V10	Tooth stiffness used reference geometry.	The target was the user's modified profile, not exact paper reproduction.	Reconstruct the actual modified tooth profile and compute derivatives by finite difference.	`fun_tooth_stiffness.m`.
V11	Visual model showed tiny gaps and label overlap.	VLM audit found mismatch between numerical contact and rendered profile.	Use high-density sampling and corrected tangency logic in `export_tikz_data.m`.	VLM-driven sync.

Table 9: Compressed AI-interaction trace.

Human challenge	AI action	Accepted outcome	Research role
"Why did the number of loaded pins become much smaller?"	Reanalyzed pin smaller than housing hole, clearance, and elastic compression.	Floating-pin iteration reformulated.	Physical interpretation.
"The pin is smaller than the hole, so it should move first."	Traced how clearance was entering contact selection and compliance.	Rigid displacement-before-deformation logic was added.	Idea-to-algorithm translation.
"General TCA often assumes fixed pins; this model should expose pin-hole motion."	Separated actual pin center, housing-hole center, clearance vector, and contact point in the figure data.	Figure 4 and algorithm passport show the moving-pin interpretation.	Visual and code evidence.
"Only static friction should be considered."	Removed dynamic friction interpretation and used slip tendency for direction.	Static Coulomb friction projected into torque balance.	Assumption control.
"Use my modified tooth profiles."	Replaced purely analytical unmodified profile with actual profile reconstruction.	Numerical-difference derivatives used for stiffness integration.	Code-model consistency.
"Can this become an arXiv cs.SE paper?"	Reframed work as CLI-based AI workflow with robotics case study.	Manuscript structure changed from model report to workflow paper.	Contribution framing.

Table 10: Claim-audit accuracy: IMRaD baseline vs. SAH-structured artifacts. Scores are percentages (mean ± std across three AI systems).

Task	IMRaD baseline	SAH version
Claim extraction	62.3 ± 8.1	91.7 ± 3.2
Evidence linking	45.0 ± 12.4	88.3 ± 5.1
Inconsistency detection	38.7 ± 15.3	76.0 ± 8.7
Limitation identification	55.0 ± 10.2	83.3 ± 6.4
Reproducibility assessment	48.3 ± 14.1	80.0 ± 7.5
Overall mean	49.9 ± 11.8	83.9 ± 6.0

Table 11: Mapping between source files and research roles.

File	Role	Harness inspection question
`runTCA.m`	Orchestrates TCA/LTCA workflow.	Which model version generated the plotted result?
`fun_force.m`	Solves force and torque balance.	How are clearance, friction, and compliance combined?
`fun_output_mod.m`	Computes unloaded contact geometry.	Which radius defines the actual operating contact?
`fun_findtc.m`	Finds tooth-contact positions.	Are profile radius and pin-center radius separated?
`fun_tooth_stiffness.m`	Computes tooth-body stiffness.	Does stiffness use the modified actual tooth profile?
`references.bib`	Stores citation metadata.	Which claim depends on which source?

Table 12: Anti-contamination protocol for AI-native research papers.

Contamination route	**Hidden error**	**Harness countermeasure**
Fluent AI writing	Plausible prose hides unsupported claims.	Claim-evidence map; every major claim must point to code, data, formula, literature role, or limitation.
AI literature synthesis	Real papers are cited for claims they do not support.	Dense literature matrix states each source's role, reusable information, and boundary.
AI formula editing	Dimensionally or physically invalid equations look standard.	Formula passport includes units, code path, physical meaning, and failure mode.
AI code generation	Code runs but implements a wrong model.	Algorithm passport, code-artifact map, minimal numerical checks, and human correction trace.
AI figure generation	Diagrams become persuasive but not data-backed.	Claim-linked figures must state regeneration path and plotted data source.
AI review	Review focuses on novelty and language instead of artifact consistency.	Multi-agent artifact review: literature, logic, code, unit, figure, domain, and reproduction checks.
Database indexing	Low-evidence papers are treated like verified artifacts.	Evidence-weighted metadata: generated-only, code-available, data-available, reproduced, corrected, or disputed.

Table 13: Verification ladder for AI-native research claims (cf. CI/CD pipeline: each layer is a gate that must pass before promotion to the next stage).

Layer	**What AI can verify**	**Typical artifact**	**Residual human role**
Formal proof	Logical validity of mathematical statements.	Lean theorem, proof script, checked library dependency.	Decide whether the formal statement matches the scientific claim.
Executable code	Whether the described algorithm runs and matches tests.	Source file, test case, environment, output hash.	Decide whether tests represent the real model boundary.
Numerical simulation	Whether behavior is consistent under parameter sweeps or higher-fidelity solvers.	FEM/CFD/MBD models via OpenFOAM, CalculiX, preCICE, Exudyn; LTCA comparison; AI-configured cross-domain simulations; residual curves; adversarial cases.	Judge model fidelity, mesh adequacy, and physical relevance.
Experiment	Whether predicted quantities match measured behavior.	Test bench data, calibration record, uncertainty budget.	Judge instrumentation, operating condition, and interpretation.
Embodied interaction	Whether an AI system can close the loop through sensing, action, and correction.	Robot lab workflow, autonomous experiment log, sensor trace, MCP-mediated tool calls and responses.	Define objective, safety boundary, and meaningful success criteria.
Literature feedback	Whether later work reproduces, disputes, or corrects the claim.	Evidence-weighted database record, replication status, correction trace.	Decide when accumulated evidence changes the paper's status.

Table 14: Review–revision trace. Each row maps a reviewer concern to the revision made, the section affected, and the type of change.

ID	Revision	Section	Change type
Round 1 revisions (Version 2.0)			
W1	Claim-audit experiment with three AI systems, five tasks, quantitative metrics (Table 10).	§4.5 (new)	New section
W2	Formal pre/post-conditions, scan+fzero strategy, convergence analysis, code-verified against `runTCA_analysis.m`.	§3.4	Expanded tcolorbox
W2	Hertz contact formula passport with dimensional analysis.	§3.5	New passport
W3	Minimum Viable Harness (MVH) definition + cost-benefit table.	§7.1–7.2	New subsections
W4	Concrete AI-interaction trace + Statement of AI Contribution.	§4.4	New subsections
Round 2 revisions (Version 2.0, cont.)			
W7	Reframed SAH as *necessary but not sufficient*; changed "breaks the loop" to "enables detection and interruption."	§5, §8.1	Wording
W8	Added explicit comparison with Jupyter, Code Ocean, OSF, and related platforms.	§3.8, App. 3.9	Expanded
Q1	Stated testable hypothesis: SAH format increases claim-evidence binding completeness and inconsistency detection rate.	§4.5	Expanded
Q2	Clarified semantic-layer distinction: Jupyter is executable narrative; SAH is an auditable knowledge graph with typed claim interfaces.	§3.8	Expanded
Q3	Described AI verification workflow: read claim-evidence map → check preconditions → verify code-artifact binding → generate audit report.	§3.4	Expanded
Q4	Acknowledged that SAH raises fabrication cost from $O(1)$ (plausible prose) to $O(n)$ (consistent code + trace + evidence binding) but cannot eliminate fabrication.	§8.1	Expanded
Round 3 revisions (Version 3.0)			
R3	Published formal verification artifacts to GitHub: JSON Schema (`sah-ltca.schema.json`), MATLAB runtime assertions (`SAH_PRE`/`SAH_POST`/`SAH_CONV`), and Dafny contract sketch (`ltca_contracts.dfy`).	App. B (new)	New appendix
R3	Added Code and Data Availability section with GitHub URL and commit hash.	§10 (new)	New section

All test parameters match `run_parametric_study.m` ($Z_a = 59$, $Z_b = 60$, $R_z = 168.01\,\text{mm}$, $T_{in} = 1100\,\text{Nm}$).

Table 15: Cost-benefit comparison of paper formats.

Dimension	IMRaD	MVH	Full harness
Writing overhead	Baseline	+15–20%	+40–60%
Claim extraction accuracy (AI)	62%	80% (est.)	92%
Evidence linking accuracy	45%	70% (est.)	88%
Inconsistency detection	39%	60% (est.)	76%
Adoption barrier	None	Low	Moderate
Automation potential	Low	Medium	High

Table 16: Open questions on the path from static harness to self-evolving research systems.

Question	Current state	Required breakthrough
Can the system learn from use without retraining?	Feedback does not update weights; generated code accumulates but does not self-improve.	Persistent, locally updated adapters that encode interaction experience.
Can end-to-end control replace API bridges?	Text-centric pipeline with OCR errors and per-software API overhead.	Native visuomotor models that output keyboard/mouse events from screen video.
Can domain expertise be acquired without domain pretraining?	General-purpose LLM lacks engineering intuition; domain data is scarce in training corpora.	Domain-specific foundation models or self-evolving agents that learn through use.
Can multi-agent consensus converge reliably?	Single-agent systems have fixed blind spots; direct agent dialogue risks bias amplification.	Structured claim-evidence networks with provenance tracking and adversarial review.
Can AI autonomously set research directions?	Current systems execute and verify but do not originate questions.	Models with open-ended curiosity and the ability to recognize anomalies as research opportunities.