

Literate Coding with AI Agents: A Methodology for Computational Analysis of Scientific Objects

Complexity Reduction through Structured Context, Executable Specification, and Closed-Loop Refinement

Gerd Graßhoff 

Humboldt-Universität zu Berlin, Deutsches Museum
gerd.grasshoff@hu-berlin.de

ABSTRACT

We present a computational research methodology that unifies Knuth’s literate programming with autonomous AI agent capabilities for the analysis of complex scientific objects. The central challenge in agent-assisted research is context fragmentation: mathematical derivations live in papers, algorithms in repositories, observational data in archives, and success criteria in the researcher’s head. When an AI agent operates across these disjoint silos, it inevitably generates code that is syntactically valid but physically or historically incoherent. We establish the Principle of Complexity Reduction: before delegating analytical tasks, the researcher constructs a single executable document containing domain theory, data schemas, quantitative thresholds, and code. The methodology defines three operational levels: (1) a living reference background under permanent revision, (2) a structured planning substrate, and (3) autonomous agentic execution with a closed feedback loop that updates the domain theory upon experimental failure. We demonstrate the methodology on the 3D reconstruction of ancient Hellenistic conic sundials.

KEYWORDS literate coding · AI agents · executable specification · reproducible research · complexity reduction · computational archaeology · scientific methodology

Table of contents

1 Introduction: The Problem of Context in AI-Assisted Research	2
2 The Three Levels of Agentic Operation	2
2.1 Level 1: The Living Reference Background	2
2.2 Level 2: The Structured Planning Substrate	3
2.3 Level 3: Autonomous Agentic Execution and Feedback	3
3 The Document as a Directed Acyclic Graph (DAG)	3
4 Case Study: Reconstructing Ancient Conic Sundials	3
4.1 Step 1: Embedding the Domain Theory (Level 1)	3
4.2 Step 2: Agentic Planning (Level 2)	3
4.3 Step 3: Execution, Anomaly Detection, and Theory Refinement (Level 3)	3
5 Epistemic Standards and Reproducibility	4
6 Conclusion	4
References	4
Bibliography	4

Supplement S7

Supplement S7 to the Ancient Sundials repository (Berlin Sundial Collaboration, ancient-sundials.lettire.ai). This draft documents how the repository’s measurement pipeline was built and checked. It is a preprint draft and has not been peer reviewed. Where a figure in the repository differs from a figure here, the repository, which is rebuilt from the current database, is authoritative.

1 Introduction: The Problem of Context in AI-Assisted Research

The computational analysis of complex scientific objects — such as 3D scans of fragmentary archaeological instruments, multi-spectral manuscripts, or astronomical spectra — requires the simultaneous coordination of four distinct knowledge domains:

1. **Domain Theory:** The physical laws, spherical astronomy, and mathematical theorems that govern the object [1].
2. **Observational Data:** Sensor point clouds, mesh representations, and coordinate metrology [2].
3. **Algorithms:** Numerical optimization routines, curve fitters, and signal filters [3].
4. **Success Criteria:** Quantitative tolerance thresholds derived from physical error models rather than convenience.

In conventional computational science, these four domains are scattered across disjoint media: journal articles, git repositories, remote databases, and undocumented researcher intuition [4], [5].

When large language model (LLM) agents are introduced into such fragmented workflows, they inherit this structural incoherence. An AI agent given access to code and data without the governing theoretical derivations will hallucinate plausible-sounding but geometrically invalid algorithms.

CONCEPT The Principle of Complexity Reduction

Before delegating an analytical problem to an AI agent, reduce the problem to its essential structure by constructing a single literate document that embeds: (1) the domain theory and derivations, (2) the observational data schema, (3) the quantitative success criteria, and (4) the runnable code blocks. The literate document is the agent's complete and bounded world.

This principle builds on Donald Knuth's classical *literate programming* [6]. However, where traditional literate programming focused on interleaving code with software documentation for human readers, **literate coding for AI agents** extends the document into a living, *executable specification* that actively constrains and guides autonomous agent reasoning.

2 The Three Levels of Agentic Operation

The literate coding methodology establishes three hierarchically coupled modes of interaction between the researcher, the document, and the AI agent:

METHOD The three-level operational model

- Level 1 (Reference Background): Domain theory under permanent revision.
- Level 2 (Planning Substrate): Formal derivation of agent execution plans.
- Level 3 (Agentic Action): Autonomous code execution and closed-loop validation.

2.1 Level 1: The Living Reference Background

At Level 1, the literate Quarto document serves as the authoritative repository of domain definitions, coordinate conventions, and analytical proofs. Unlike a static published paper, the Level 1 specification is under **permanent revision**:

- When an analytical experiment at Level 3 fails, the failure is diagnosed to determine whether the error lies in code or in an oversimplified theoretical assumption.
- If the theory was deficient, the Level 1 document is directly updated.
- This revision immediately updates the agent's prompt context, preventing repeated hallucinations and enforcing newly discovered physical constraints.

2.2 Level 2: The Structured Planning Substrate

At Level 2, the AI agent consumes the Level 1 specification to synthesize an actionable plan before generating executable code:

- The agent maps each high-level research goal to specific mathematical constraints in the text.
- It verifies that the required data formats and coordinate frames are aligned.
- It establishes quantitative acceptance thresholds (e.g. “Fit RMS ≤ 0.4 mm governed by the ± 0.3 mm scan noise floor”).

2.3 Level 3: Autonomous Agentic Execution and Feedback

At Level 3, the agent executes the plan in a sandboxed runtime environment:

- It runs Python or Julia scripts against the 3D data.
- It compares numerical outputs against the Level 2 criteria.
- If a criterion fails, it logs an explicit diagnostic trace.
- The closed feedback loop either adjusts algorithmic parameters or triggers a theoretical revision at Level 1.

3 The Document as a Directed Acyclic Graph (DAG)

An executable specification is structured as a directed acyclic graph of formal dependencies:

$$\begin{array}{ccccccc} \text{Domain Theory } \mathcal{T} & \xrightarrow{\text{constrains}} & \text{Methods } \mathcal{M} & \xrightarrow{\text{implements}} & \text{Executable Code } \mathcal{C} & \xrightarrow{\text{produces}} & \text{Results } \mathcal{R} \\ & & & & & & \\ & & \text{Domain Theory } \mathcal{T} & \xrightarrow{\text{derives}} & \text{Criteria } \mathcal{K} & \xrightarrow{\text{evaluates}} & \text{Results } \mathcal{R} \xrightarrow{\text{feedback}} \mathcal{T} \end{array}$$

Reading. The theoretical domain model $\mathcal{T}$ constrains the algorithmic methods $\mathcal{M}$ and derives the quantitative success criteria $\mathcal{K}$. Code $\mathcal{C}$ executes $\mathcal{M}$ on data to produce results $\mathcal{R}$, which are evaluated against $\mathcal{K}$. Discrepancies feed back directly into revising $\mathcal{T}$.

4 Case Study: Reconstructing Ancient Conic Sundials

We illustrate the complete three-level methodology on the reconstruction of ancient Hellenistic conic sundials from photogrammetric 3D surface scans [2].

4.1 Step 1: Embedding the Domain Theory (Level 1)

The literate specification incorporates the complete construction geometry from G. Graßhoff [1]:

- Coordinate systems: East–North–Up world frame and equatorial polar frame.
- Forward shadow projection: ray–cone intersection quadratic equations.
- The equinox invariant: $\ell_E = d$.
- The apex-free marking ratio $Q = (L_S - L_W)/(L_S - L_E)$ and its exact inversion $\alpha = \arctan((2 - Q)/(Q \tan \varepsilon))$.

4.2 Step 2: Agentic Planning (Level 2)

An AI sub-agent parses the Quarto document and constructs a 5-step execution plan:

1. Ingest PLY point clouds for Dialface ObjID 18 (British Museum inv. 2547).
2. Slice cross-sectional bands and extract medial groove centrelines.
3. Fit 3D circular arcs to hour lines and coaxial circles to day-lines.
4. Extract noon marks P_S, P_E, P_W and apply closed-form Q -inversion.
5. Combine front plane, cone axis, and solstitial hour divisions into consensus latitude φ .

4.3 Step 3: Execution, Anomaly Detection, and Theory Refinement (Level 3)

During execution, the agent encounters two major physical anomalies:

1. **Straight Line Failure:** Straight lines leave systematic bow residuals of 2.2–7.6 mm on outer hour lines. The agent diagnoses that ancient artisans carved hour lines as circular arcs (0.20–0.71 mm RMS), triggering a Level 1 update in the document.
2. **Depth Notch Failure:** The minimum depth estimator (argmin_z) diverges due to ± 0.3 mm scan noise. The agent substitutes the medial-line filter and updates the specification.

OPERATION

Literate Coding Agent Closed-Loop Execution

2.1 s

Q

Execute reconstruction cascade on ObjID 18 with autonomous anomaly handling

1	spec-reader	Reads domain theory from conic-sundial-geometry.qmd and extracts constraint catalog
2	code-generator	Synthesizes Python pipeline utilizing MedialSliceExtractor and ApexFreeInverter
3	test-runner	Evaluates fit residuals against $\sigma_0 = 0.3$ mm threshold; detects straight-line bow anomaly
4	theory-updater	Amends Level 1 specification to include circular arc model class and re-runs fit
5	consensus-builder	Verifies exact closure: $\alpha = 34.40^\circ$, $d = 101.3$ mm, $\varphi = 41.45^\circ$ ($s = 53; 02$)

→ All criteria passed; Level 1 specification amended; parameter deduction verified within 0.6 standard errors

5 Epistemic Standards and Reproducibility

The literate coding methodology enforces three foundational epistemic guarantees:

1. **Traceability:** Every parameter in the final paper is programmatically derived from raw observation data through documented code blocks.
2. **Constraint Enforcement:** AI agents are forbidden from introducing free parameters that violate the labeled constraints in the Level 1 catalog.
3. **Reproducibility:** Re-rendering the literate document executes the entire computational pipeline from 3D point cloud to publication-ready PDF tables and figures.

6 Conclusion

As autonomous AI agents assume greater responsibility in scientific data analysis, prompt engineering must be replaced by formal, executable specifications. Literate coding bridges the gap between high-level domain theory and low-level code execution. By embedding mathematical derivations, data contracts, and quantitative criteria into a unified living document, scientific discoveries can be conducted with unprecedented speed while maintaining absolute epistemological rigor.

References

Bibliography

- [1] G. Graßhoff, “Construction Geometry of the Conic Sundial with Horizontal Gnomon,” 2026.
- [2] Berlin Sundial Collaboration, “Dialface ID 18: Surfaces extracted from model BSDP0300 (ObjID 18).” Edition Topoi, 2014.
- [3] Y. Bard, *Nonlinear Parameter Estimation*. New York: Academic Press, 1974.
- [4] J. F. Claerbout and M. Karrenbach, “Electronic Documents Give Reproducible Research a New Meaning,” *SEG Technical Program Expanded Abstracts*, pp. 601–604, 1992, doi: 10.1190/1.1822162.

-
- [5] R. D. Peng, “Reproducible Research in Computational Science,” *Science*, vol. 334, no. 6060, pp. 1226–1227, 2011, doi: 10.1126/science.1213847.
 - [6] D. E. Knuth, “Literate Programming,” *The Computer Journal*, vol. 27, no. 2, pp. 97–111, 1984, doi: 10.1093/comjnl/27.2.97.