Cinemath: Separating Pedagogical Reasoning from Deterministic Educational Animation

Mauricio Tedeschi

July 18, 2026

Abstract

Large language models can explain mathematical and physical reasoning, but they are unreliable authors of low-level animation code. We present Cinemath, an open-source pipeline that turns a typed or photographed problem into a short educational Manim video while keeping those roles strictly separated. A lightweight classifier routes each problem either to a local catalog planner (SymPy-backed `plan_*` functions that emit a complete lesson plan without freeform generation) or to a freeform teacher LLM that returns a structured plan of conceptual steps and named visual-tool calls. Integration problems route to technique-specific catalog planners (partial fractions, trigonometric substitution, u-substitution, integration by parts, or elementary antiderivatives); stacked or SymPy-unwalkable integrals fall through to freeform. Local compilers expand those calls into a validated animation script, re-check freeform claims with SymPy (retrying on failure), and render with Manim Community. Because scene construction never depends on free-form model-generated Python, visual fidelity is controlled by hand-written templates spanning paper arithmetic, algebra boards, calculus plots, formal proofs, and quantum-field-theory diagrams including one-loop layouts and renormalization-group flow sketches. We describe the architecture, the classify-and-catalog layer, the plan schema, the visual-tool catalog, curated and scraped problem banks, and a difficulty ladder from elementary arithmetic through Peskin-style β-function problems.

1 Introduction

Educational animation systems such as Manim make it possible to present equations, graphs, and diagrams with cinematic timing [1]. At the same time, frontier language models have become competent tutors for multi-step mathematics and physics [2]. Combining the two is attractive: given a homework-style prompt, one would like a narrated visual solution rather than a static wall of text.

A naïve approach—asking the model to emit Manim source directly—fails in predictable ways. Generated code frequently invents nonexistent APIs, breaks LaTeX mid-expression, hard-codes fragile layout constants, or silently skips pedagogically important beats. Even when the mathematics in the explanation is correct, the animation layer may not compile.

Cinemath adopts the opposite contract. The language model never authors Manim source. For common problem shapes it may not author the lesson plan either: a classifier selects a local catalog planner that builds `plan.json` with SymPy or exact arithmetic. Only when no planner fits does a freeform teacher return a versioned JSON plan—normalized statement, final answer, conceptual steps, and a non-empty list of visual tools from a fixed catalog. All animation construction, validation, and rendering remain local and deterministic. This separation mirrors classical compiler design: the front end emits an intermediate representation; trusted backends lower it to pixels.

The system is deliberately domain-agnostic at the plan level. Instead of a single mutually exclusive "problem type," the plan selects a composition of tools (for example, a rectangular double integral may request a region outline, an equation board, a three-dimensional surface, and a final answer beat). Specialized QFT tools cover interaction Lagrangians, tree-level $1 \to 2$ decays, one-loop / 1PI diagrams, and two-dimensional renormalization-group flow fields of the kind appearing in textbook exercises [3].

2 Architecture

Figure 1 summarizes the end-to-end pipeline. A run begins with a text file, Markdown note, or image of a problem. Images are OCR'd / transcribed; the extracted statement is then classified. Classification calls exactly one tool: either a catalog `plan_*` entry (quadratic, linear systems, derivatives, technique-specific integrals—partial fractions, trigonometric substitution, u-substitution, integration by parts, elementary definite / improper integrals—partial derivatives, double / triple box integrals, paper arithmetic, ...) or `teach_freeform`. Catalog planners emit a complete plan locally. When a catalog planner cannot finish an integral (for example a stacked $\sqrt{\tan x}$ exercise), it signals freeform rather than silently stacking techniques inside one lesson. The freeform teacher may call local arithmetic tools for long multiplication, division, addition, or subtraction so that digit-level results are ground-truth rather than model-invented; its plan is then verified with SymPy and retried with structured feedback on failure. Local template compilers expand `visuals[]` into an internal animation DSL, prepend a problem-statement scene, and validate the script. Manim finally renders `animation.mp4` together with a Markdown lesson transcript.

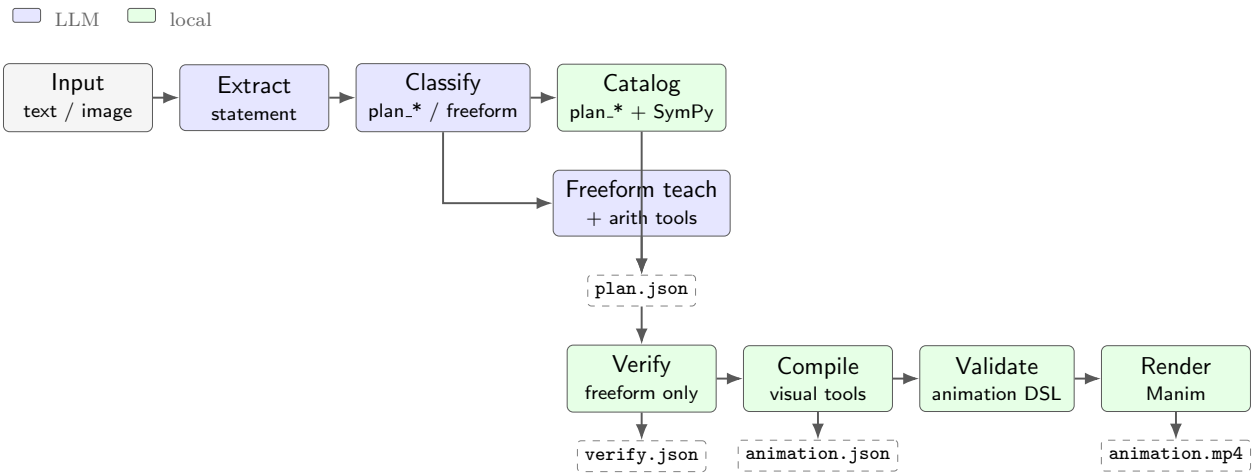


Figure 1: Cinemath pipeline. Blue stages may call the language model; green stages are fully local. Classification routes to a local catalog planner or a freeform teacher. `verify.json` is written for freeform plans (with SymPy retry); catalog plans skip that stage. Neither path authors Manim scenes.

2.1 Trust boundary

The critical invariant is that `animation.json` is never authored by the model. Catalog planners fill typed visual fields locally; the freeform teacher may choose *which* visual capabilities to invoke and fill their typed fields (coefficients, region bounds, Feynman labels, β-function expressions). Scene

graph construction, caption timing, equation-chain morphs, and camera moves live in versioned Python modules under `templates/` and `render_engine/`.

2.2 Artifacts

Each successful `cinemath solve` run creates a timestamped directory containing

- `problem.txt` — normalized statement with ASCII LaTeX in `$...$`;
- `plan.json` — lesson plan (version 2), from a catalog planner or the freeform teacher;
- `verify.json` — freeform SymPy check / retry summary (omitted for catalog plans);
- `lesson.md` — human-readable lesson transcript;
- `animation.json` — validated internal DSL;
- `animation.mp4` — rendered video (optional retained `media/` with `--keep-media`).

2.3 Catalog planners

When classification hits a catalog entry, a local `plan_*` function in `src/cinemath/planners/` builds the entire plan—steps, answer, and `visuals[]`—using SymPy or exact arithmetic. Shipped planners include `plan_quadratic`, `plan_linear`, `plan_linear_system_2d/3d`, `plan_percent_off`, `plan_derivative`, `plan_partial_fractions`, `plan_trig_substitution`, `plan_u_substitution`, `plan_integratio` `plan_definite_integral` (elementary antiderivatives and improper limits only), `plan_partial_derivative`, `plan_double_integral`, `plan_triple_integral`, and the four `plan_long_*` arithmetic planners. The classifier picks exactly one integration technique; `plan_definite_integral` rejects integrands that need a specialized planner, and unwalkable stacked integrals route to `teach_freeform`. Proofs, series arguments, vector calculus, and QFT exercises also fall through to `teach_freeform`.

2.4 Worked example: quadratic roots

Figure 2 summarizes the on-disk contract for a real run of `examples/04_quadratic.txt` ($x^2 - 5x + 6 = 0$; also `problems/curated/04_quadratic.txt`). Classification selects `plan_quadratic`, so the LLM does not write `plan.json`; local stages emit `animation.json`, then Manim produces `animation.mp4`. The JSON chips show *shape*, not full payloads. Figure 3 shows four frames from that same video.

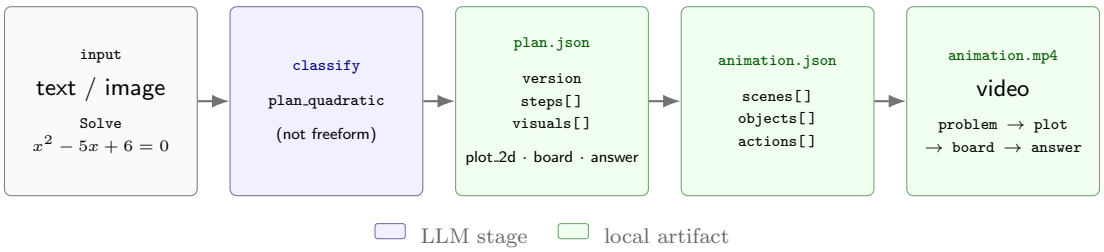
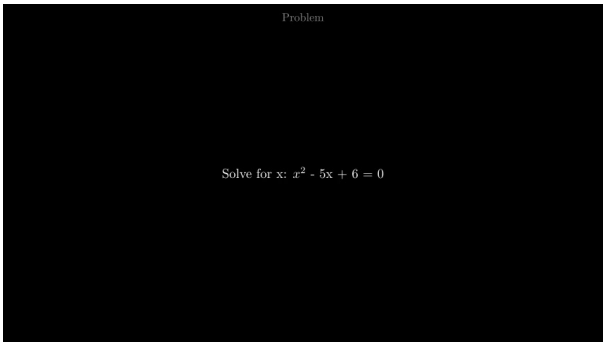
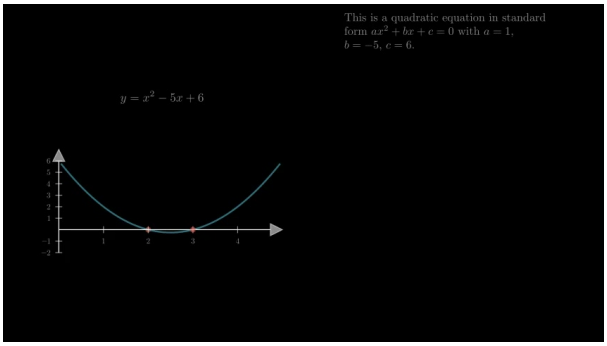


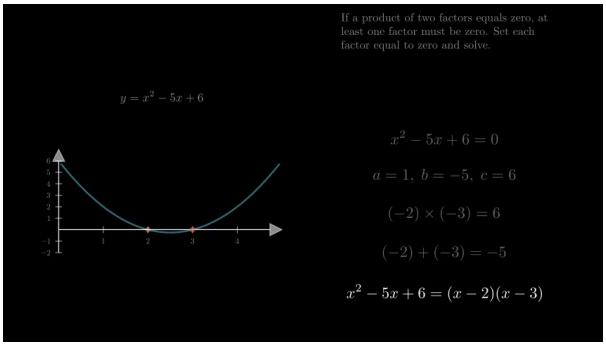
Figure 2: Quadratic catalog run as pictographs. Classification selects `plan_quadratic`; the plan and animation artifacts are local. Boxes sketch *shape*, not complete file contents.



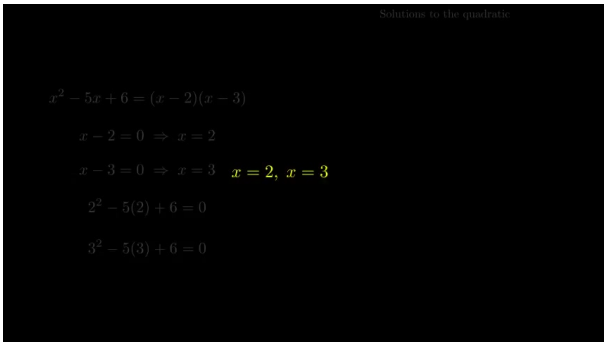
(a) Problem beat



(b) `plot_2d` beat



(c) Equation board



(d) Answer beat

Figure 3: Frames from the same quadratic `animation.mp4`, in order.

3 Teacher plan schema

Plans are JSON objects with schema version 2. The required fields are `problem`, `answer`, `steps`, and a non-empty `visuals` array. Each step carries a short board title, a one-to-three sentence explanation, and zero or more display-math fragments. Visual entries are tagged by `tool` and validated against a closed catalog (Section 4). Catalog planners and the freeform teacher share this schema so the compilers need not know which front end produced the plan.

The freeform system prompt forbids animation design language: the model must not mention Manim, colors, camera angles, or frame timing. Mathematical hygiene rules require ASCII LaTeX (no Unicode glyphs such as μ or ϕ written as code points) and consistent dollar wrapping in prose fields. Pure formula fields omit surrounding dollars so that Manim's `MathTex` path remains unambiguous.

For long arithmetic outside the catalog path, the freeform teacher must call the matching local tool (`long_multiply`, `long_divide`, `long_add`, or `long_subtract`) and copy the returned digits into both `answer` and the corresponding `paper_long_*` visual. Catalog arithmetic planners skip that round-trip and write the digits directly.

4 Visual tools

Table 1 lists the current catalog. Tools are intentionally composable: a proof uses `state_claim`, `equation_board`, and `show_qed`; a rectangular double integral typically stacks region, board, surface, and answer beats; a Yukawa β-function exercise may combine a Lagrangian frame, one or more loop diagrams, the equation board, an RG flow field, and a highlighted answer.

Table 1: Visual-tool catalog exposed to the teacher (plan version 2).

Tool	Role
`equation_board`	Caption + stacked derivation lines
`state_claim` / `show_qed`	Proof opener / closer
`plot_2d`	Quadratic parabola with marked roots
`plot_integral_1d`	Shaded definite integral under a curve
`show_region_rectangle`	2D rectangular integration region
`plot_surface_3d`	Surface over a rectangular domain
`paper_long_*`	Stacked paper arithmetic layouts
`show_lagrangian`	Interaction / Lagrangian setup
`feynman_1to2`	Tree-level $1 \to 2$ decay diagram
`feynman_loop`	One-loop layouts (bubble, 4-point, Yukawa triangle)
`rg_flow_2d`	Arrow field from $\beta_x(x, y)$, $\beta_y(x, y)$
`show_answer`	Final highlighted answer beat

Each tool maps to a small compiler that emits objects in an internal DSL (`math`, `prose`, `axes`, `plot`, `feynman`, `flow_field`, ...) and a short action list (`write`, `create`, `indicate`, `wait`, ...). The renderer interprets that DSL only after schema validation, including safe-expression checks for plots and RG fields.

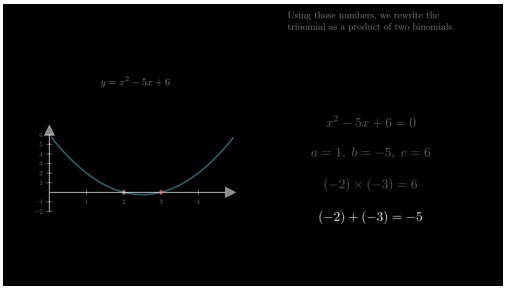
5 Rendering and pedagogy

Algebraic derivations use an equation chain: the previous line is dimmed, the board scrolls to make room, and a copy morphs into the next expression via Manim's `TransformMatchingTex`. Step explanations become wrapped instruction banners sized for full-frame or split-stage layouts. Every video opens with a problem-statement beat so the viewer sees the claim before the solution choreography begins.

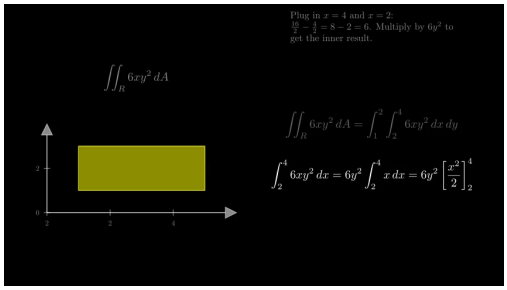
Two- and three-dimensional graph helpers auto-select axis ranges, optionally shade regions, and keep captions clear of plot chrome. QFT helpers draw schematic Feynman layouts and sample renormalization-group arrow fields from safe arithmetic expressions in variables (x, y), which the teacher uses as stand-ins for couplings such as (λ, g).

6 Examples and problem banks

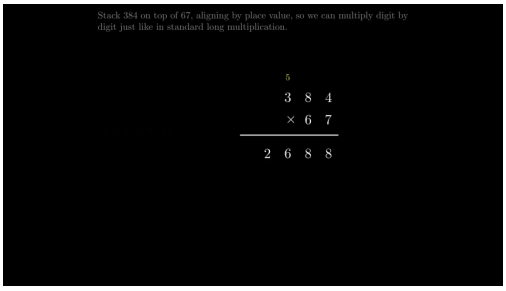
The repository ships a curated difficulty ladder under `problems/curated/` (symlinked as `examples/` at the repo root), from elementary word problems through linear systems, Calc II/III catalog targets, tree-level scalar decay, and related QFT exercises (Table 2). Scraped banks from Paul's Online Math Notes (Lamar), MIT 18.01/18.02, Arizona Math 129, and OpenStax live under `problems/` with per-topic `meta.json` manifests; sync with `scripts/sync_problem_banks.py`. In practice the same architecture also covers photographed textbook problems, including Callan–Symanzik β-function computations in pseudoscalar Yukawa theory with qualitative RG-flow sketches [3]. Figure 4 shows representative frames from rendered lessons across several tool recipes.



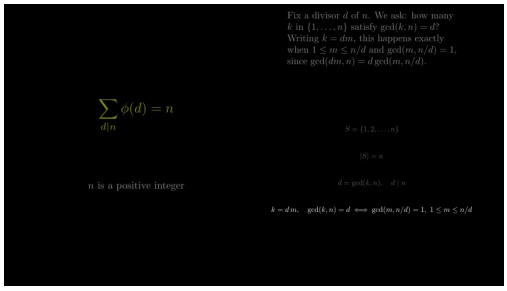
(a) Quadratic + `plot_2d`



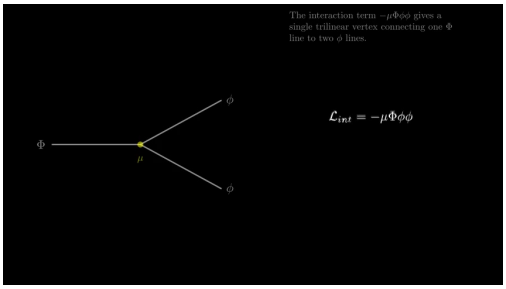
(b) Double-integral region



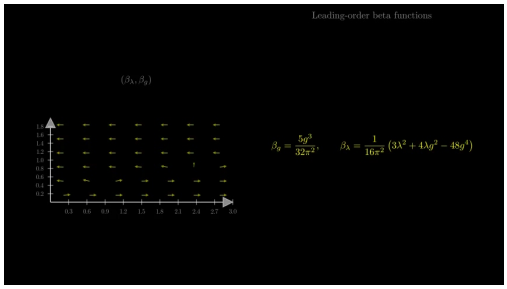
(c) Paper long multiplication



(d) Totient proof board



(e) Tree-level $1 \to 2$ decay



(f) RG flow + one-loop betas

Figure 4: Representative frames from CINEMATH lessons (algebra through QFT).

Table 2: Representative curated ladder (`problems/curated/`).

Level	Example	Planner	Typical visuals
1–3	Arithmetic / percent	catalog / freeform	`equation_board`
4	Quadratic	`plan_quadratic`	`plot_2d`, board, answer
6–7	Derivative / double integral	catalog	board / region+surface
8	Euler totient identity	freeform	claim, board, QED
9–10	Improper integral / $\Phi \to \phi\phi$	catalog / freeform	board / Feynman
11–15	Paper long arithmetic	`plan_long_*`	`paper_long_*`
16–17	Linear systems 2×2 / 3×3	catalog	board, answer
18–21	IBP / sine / partial / triple	catalog	board (+ plot/region)

7 Discussion

Restricting generation to a typed intermediate representation—and, for catalog hits, removing freeform plan authorship entirely—trades expressive freedom for reliability. New visual genres

and new `plan_*` entries require local code, but once written they are reused across every problem that routes to them. Classification already implements a coarse gate: only the chosen planner's parameters (or the freeform teach path) are in play, which keeps context smaller as the planner registry grows. Visual tools remain tagged by domain (`core`, `proof`, `calculus`, `arithmetic`, `qft`) for further prompt gating if needed.

Limitations remain. Verification coverage is uneven across freeform tools; symbolic checks are strongest for arithmetic and selected algebraic payloads. Catalog calculus planners now cover partial fractions, trigonometric substitution, u-substitution, integration by parts, and elementary definite integrals with stepped SymPy `manualintegrate` walks; stacked or SymPy-unwalkable integrals defer to the freeform teacher rather than auto-chaining techniques inside one catalog lesson. Remaining gaps include series tests, vector calculus, and richer diagrammatic calculus. Pedagogical quality on the freeform path still depends on the teacher's step decomposition. Schematic Feynman diagrams are illustrative rather than automated 1PI generators, and RG flow fields visualize provided β expressions rather than deriving them. Finally, image OCR inherits the usual transcription failure modes for dense blackboard photography.

Natural extensions include richer series and vector-calculus planners, tighter SymPy verification of board lines, golden tests over the curated ladder and Lamar banks organized by `plan_*`, and optional narration audio aligned to scene captions.

8 Conclusions

CINEMATH demonstrates a practical split between LLM routing / pedagogy and deterministic educational animation. Classification and catalog planners keep common problems off the freeform path; when the teacher is needed, it still never writes animation code. Local visual tools compile and Manim renders. The resulting pipeline produces short lesson videos across arithmetic, algebra, calculus, proof, and selected quantum field theory topics, backed by curated examples and scraped Calc II/III problem banks. Source code and examples are available in the accompanying repository.

Software

Implementation is in Python 3.12+ using the Anthropic API [2], SymPy [4], Typer, and Manim Community [1]. Optional model overrides separate a fast classify/OCR model from the freeform teach model. The command-line entrypoint is

```
uv run cinemath solve path/to/problem
```

Artifacts for each run are written under `outputs/`.

A Reproducibility

From the project root,

```
uv sync
cp .env.example .env   # set ANTHROPIC_API_KEY
uv run cinemath solve examples/04_quadratic.txt
uv run cinemath solve problems/curated/18_lamar_integration_by_parts.txt
uv run cinemath solve problems/by-type/plan_trig_substitution/lamar-calc-ii-trig-substitutions/
uv run python scripts/sync_problem_banks.py   # refresh scraped banks
```

References

[1] The Manim Community Developers. Manim – Mathematical Animation Framework (community edition). https://www.manim.community/, 2024. Software.

[2] Anthropic. Claude. https://www.anthropic.com/claude, 2024. Large language model.

[3] Michael E. Peskin and Daniel V. Schroeder. *An Introduction to Quantum Field Theory*. Addison-Wesley, Reading, MA, 1995.

[4] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017. doi: 10.7717/peerj-cs.103.