Human-AI Hybrid Mathematical Discovery Workflow: Unifying Ramanujan's 17 Series for $\frac{1}{\pi}$

Michael Shalyt [1] **Elyasheev Leibtag** [1] **Shachar Weinbaum** [1] **Ido Kaminer** [1]

Abstract

We present a human-AI hybrid research workflow and apply it to a long-standing challenge: unifying Ramanujan's 17 remarkable series for $\frac{1}{\pi}$ (1914) under a single mathematical structure. The workflow combines human steering, a persistent LLM-based agent equipped with project-specific context, symbolic and numeric computational guardrails, and custom *skill.md* files. Special emphasis was placed on verification efficiency. Automated independent test patterns were employed, and generated artifacts were optimized for rapid human interpretation and evaluation. Both successful strategies and characteristic AI failure modes are discussed. This methodology yielded a new, elementary proof that all 17 formulas arise from a single algebraic template: a Conservative Matrix Field (CMF), where varying just a few parameters recovers each of Ramanujan's formulas. The agent and skill files have been open-sourced.

1. Problem and Setting

Ramanujan's 1914 paper (Ramanujan, 1914) lists 17 series for $\frac{1}{\pi}$, some converging at an extraordinary rate. These formulas - later extended by the Borwein brothers and the Chudnovsky brothers - underpin modern computation of π (Borwein et al., 1989). Despite a century of study, a unified structure for *all* 17 series has remained elusive.

CMFs were recently used to discover and organize formulas for mathematical constants (Elimelech et al., 2024). In essence, a CMF can be viewed as a discrete geometric space in which each linear trajectory (a ray) encodes a recurrence relation. Combined with specific initial conditions, this structure yields formulas for mathematical constants. A CMF can be defined by a D-finite generating function (any function satisfying a linear ODE with polynomial coefficients - in this work we focus on hypergeometric functions) and a small set of parameters (Weinbaum et al., 2025), together determining the constants appearing in the limit formulas and their convergence behavior. CMFs have emerged as a powerful unifying framework, condensing hundreds of formulas for π from diverse sources and across centuries into a family of trajectories in a CMF generated by the ${}_2F_1(a_1, a_2;\, b_1;\, z)$ hypergeometric function (Raz et al., 2025). The Ramanujan formulas, however, resisted this condensation - thus becoming the next unification goal. In principle, the trajectories that generate Ramanujan's formulas could be found by brute-force searching over a large number of configurations. In practice, such an approach is computationally intractable due to the size of the search space and the sensitivity of the evaluation (an "almost exact" formula is still false), so a more heuristic-based approach was needed.

We tackled this challenge through tight collaboration with a custom LLM-based assistant - the "Ramanujan Agent" - which was taught skills based on the group's papers, code, private notes, notebooks, previous experiments, and working norms. Proficient with a variety of symbolic tools - both custom group libraries and publicly available ones - it became the de facto interface between researchers and computational experimentation.

1.1. Related Work

Our approach is most akin to works that couple language models to symbolic or formal tools for grounding (rather than trusting generated text directly). COPRA executes LLM tactic proposals in Lean or Coq and feeds proof-state information back into later prompts (Thakur et al., 2024); Draft-Sketch-Prove turns informal proofs into formal sketches for automated provers (Jiang et al., 2023); LeanDojo/ReProver packages Lean interaction, retrieval, and benchmarks (Yang et al., 2023); and Baldur repairs generated Isabelle/HOL proofs using verifier feedback (First et al., 2023). AlphaGeometry similarly combines neural search with a symbolic geometry engine (Trinh et al., 2024). Our workflow is lighter-weight and more human-steered, but follows the same audit pattern.

[1]Technion - Israel Institute of Technology, Haifa 3200003, Israel.

Oral presentation at the 3rd *AI for Research Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026. Copyright 2026 by the author(s).

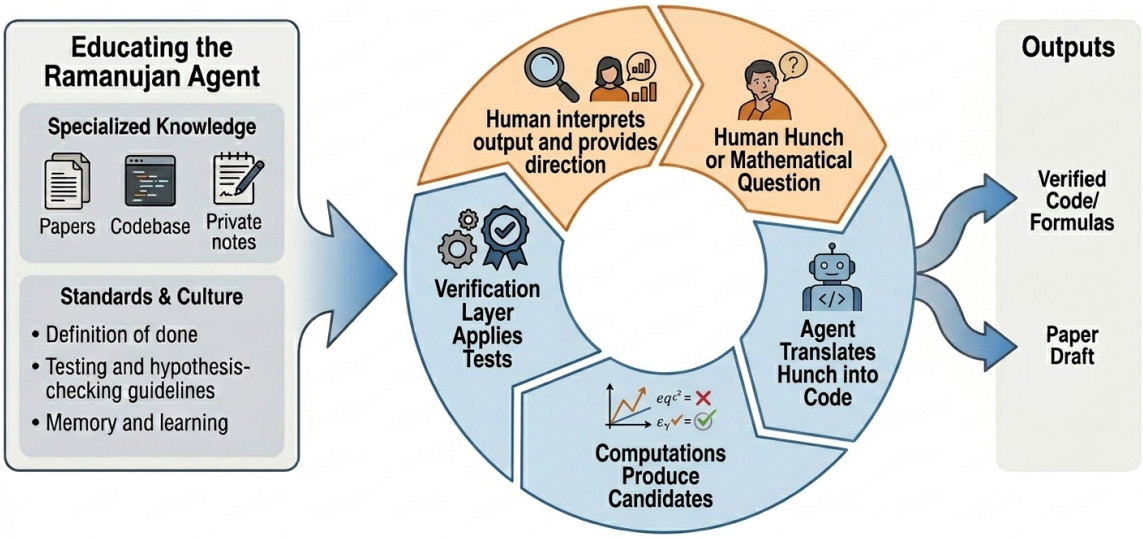

Figure 1. **Human-AI discovery loop.** A researcher proposes a "hunch"; the agent converts it into symbolic or numeric experiments; candidate results are tested; the validated or failed outcome returns to the researcher for interpretation and the next research direction.

2. Human-AI Hybrid Discovery Workflow

The workflow has four main components:

1. **Human researcher**: defined the problem, proposed candidate structures ("hunches"), interpreted results, rejected unpromising directions, and supplied the "mathematical taste".
2. **LLM-powered agent**: acted as an experimental collaborator and interface to symbolic computation. It generated formula-search code, handled low-level implementation (e.g., multiprocessing), created summaries, graphs and other human-friendly outputs. Most notably, adapted proof-of-concept results from a single formula to all 17. This work was done with OpenAI's Codex, powered by GPT-5.4 High, but the methodology and skills are model-agnostic.
3. **Symbolic computation layer**: packages like SymPy and the custom CMF library ramanujantools (Ramanujan Machine Group, 2025).
4. **Skills and definitions**: a collection of *skill.md* files containing condensed and AI-native summaries of the relevant technical and mathematical background, as well as workflow standards (e.g. testing, guardrails and "definition-of-done"). See Section 2.1.

Fig. 1 summarizes the loop. The workflow was iterative: human questions and ideas (e.g., a candidate CMF-generating function, an equivalence between formulas) were turned by the agent into scripts, experiments, symbolic checks, and short explorations (e.g., scanning thousands of trajectories and initial conditions, profiling candidate formulas). The agent then tested the output (numerically and by additional custom guardrails) before returning it to the human - in the form of a detailed report and graphical output - who interpreted the results and fed further insights and directions into the next cycle.

2.1. Ramanujan Agent "Education"

A central design choice in constructing the Ramanujan Agent was to separate background education from the immediate research task. The skills were organized as reusable layers of context: one layer for the mathematical background of the group, one for the structure and conventions of the codebase, and further layers for validation standards and "definitions of done" (see the "Ramanujan Agent Skill Design Principles" box below). This separation has two practical advantages.

First, the same background skill can support nearby research problems without being rewritten. In the present case, the move from Ramanujan's 17 formulas to other formulas in the same category is a natural continuation, for example π formulas from the RISC collection (Paule et al., 2025). More generally, the same mathematical vocabulary, codebase knowledge, and verification habits apply to problems that belong to broader topics (e.g., Diophantine approximation, experimental math).

Second, fundamental skills can be trained and validated gradually, rather than being judged only by success on a single target problem. The process is analogous to onboarding a new student: the agent is first given papers, notes, code, and terminology, and is then asked to summarize, explain, and apply what it has learned. This "educational" phase exposed gaps that would be difficult to detect from task performance alone. Early versions of the agent often produced partial summaries, missed implicit assump-

Ramanujan Agent Skill Design Principles

- Encode the **internal scientific background**: papers, notes, drafts, talks, documentation, submissions, and terminology. This is the conceptual foundation for the agent's work, and also establishes a shared vocabulary and shorthand between human and agent.
- Teach **codebase navigation**: use git repositories and history, Colab notebooks, scripts, READMEs, inline comments, and module descriptions as reference materials. Splitting the theoretical background and the codebase understanding into separate skills can help limit the agent's context load.
- Require **validation for every claim**: code-level (function unit tests, independent numerical checks for formula convergence, etc.), math-level (e.g., check known edge cases, motivate each step, cite literature when applicable), and review-level (e.g., human-readable reports and graphs, complete summaries of full vs. partial success). These are the bulk of the guardrails designed to prevent the agent from producing plausible but incorrect results.
- Define **"definitions of done"**: for each type of task or objective (e.g., finding a formula for a constant, constructing a proof, drafting a paper) explicitly list the criteria for completion that must be double-checked.
- Preserve **memory**: structured summaries of discoveries, instructions, and failed experiments - enabling continuity across sessions, agents, and models.

tions, or failed to understand terminology that was internal to the group or a specific field (see Section 4). Iterative questioning therefore served as a validation mechanism: the human researcher challenged the agent on specific definitions, asked it to explain the role of particular objects, and corrected missing or misleading statements. In some cases, this process revealed that certain necessary information had never been explicitly documented, but existed as informal group knowledge - improving internal research documentation as a side effect.

The same principle applies to the codebase skill. Since many parts of the software stack already contain unit tests and reproducible examples, the agent's codebase understanding can be evaluated by system tests rather than by single-function tasks. A useful test is to give the agent a previously solved research task that requires combining several functions or logical steps, and compare its output with a verified result. Such tests evaluate not only whether the agent can call the right functions, but also whether it understands the scientific terms and shorthand used by human researchers. This shared operational vocabulary is essential for making the agent useful as a research collaborator rather than merely as a code-generation interface.

The created agent and skills are open source and can be installed from: [URL redacted for anonymity, see supplemental material].

3. Result: Unifying Ramanujan's Formulas

The central outcome is a proven mapping between each of Ramanujan's 17 formulas and a trajectory in a CMF. Notably, every formula was shown to emerge from CMFs constructed from the same generator function - the ${}_4F_3(a_1, a_2, a_3, a_4;\, b_1, b_2, b_3;\, z)$ generalized hypergeometric function - and all following the same trajectory direction. The formulas differ only in the starting point (a shift in the hypergeometric parameters), the evaluation point z, and the initial conditions for the generated recurrence (see Table 1). This unified pattern not only offers a plausible hint at how the formulas may have been originally discovered, but more importantly, suggests that the CMF can be used to yield even more efficient formulas for π.

3.1. Mathematical Verification

In addition to the agent validation requirements discussed above, the final results were especially scrutinized. The verification strategy compares independently constructed objects. For each formula, one side is obtained directly from the summand, which determines a recurrence for partial sums or related tail quantities, without using the CMF construction. The other side is obtained from the ${}_4F_3$ CMF trajectory. Starting at $I = (1, a_2, a_3, a_4; 1, 1, 1)$ and advancing along the vector $(0, 1, 1, 1; 1, 1, 1)$, the symbolic pipeline derives a recurrence matrix. A trajectory is accepted only when this recurrence agrees with the recurrence from the original summand after normalization and scaling. This verifies an identity between functions of n (the number of summands taken in Ramanujan's series or number of steps down the trajectory in the CMF) - proving the embedding of the series in the CMF. While in practice there was a human mathematician involved in the process of validation, in the future - as the agent, skills, and guardrails are successfully applied to other mathematical problems - this involvement can become less necessary.

4. Failure Modes and Lessons

In a goal-directed research task, an agent tends to suppress failures - unsuccessful candidates, nonconverging trajectories, numerical inconsistencies, etc. - because they do not directly answer the request. This behavior is natural, but undesirable in scientific exploration. Reporting and storing negative results prevents repeated scans, and they can also reveal structure in the search space: for example, repeated failures in a subset of CMF trajectories may indicate that this search space is unlikely to contain the desired represen-

Table 1. **Mapping of Ramanujan's formulas to CMF trajectories.** The initial point is $I = (1, a_2, a_3, a_4; 1, 1, 1)$, determined by the shift. A, B are recurrence seed values.

#	formula	shift (a_2, a_3, a_4)	z	A	B
1	$\frac{4}{\pi} = \sum_{n\geq 0}(6n+1)\binom{2n}{n}^3 \frac{1}{2^{6n}}$	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	$\frac{1}{4}$	1	6
2	$\frac{16}{\pi} = \sum_{n\geq 0}(42n+5)\binom{2n}{n}^3 \frac{1}{2^{12n}}$	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	$\frac{1}{2^6}$	5	42
	...				
16	$\frac{2}{\pi\sqrt{11}} = \sum_{n\geq 0}(280n+19)\binom{2n}{n}^2\binom{4n}{2n}\frac{1}{2^{8n}99^{2n+1}}$	$(\frac{1}{4}, \frac{1}{2}, \frac{3}{4})$	$\frac{1}{99^2}$	19	280
17	$\frac{1}{2\pi\sqrt{2}} = \sum_{n\geq 0}(26390n+1103)\binom{2n}{n}^2\binom{4n}{2n}\frac{1}{2^{8n}99^{4n+2}}$	$(\frac{1}{4}, \frac{1}{2}, \frac{3}{4})$	$\frac{1}{99^4}$	1103	26390

tation. Some of these judgments were made by the human researcher, but simpler aggregations can be delegated to the agent, provided that it is explicitly instructed to save failed cases, summarize them, and extract tentative conclusions.

The more subtle, and potentially more dangerous, failures were those initially reported as successes. These cases motivated several of the validation criteria described above. For instance, while scanning CMFs for formulas equivalent to Ramanujan's, the agent found trajectories that converged to the same limiting constant and belonged to a hypergeometric family, but were nevertheless not equivalent to the original Ramanujan formulas (numerical convergence to the correct limit is not, by itself, a sufficient certificate of mathematical equivalence). The mismatch was detected by the human mathematician using a different dynamical metric (Shalyt et al., 2024) of the sequence, namely the irrationality measure (a common metric in number theory, measuring how well a real number can be approximated by rationals). This criterion was then added to the validation process. The example illustrates why multiple independent checks are necessary to verify a candidate formula - and how an agent will apply the minimal possible criteria to believe it succeeded in the task.

A separate failure mode appeared in the construction of skills. After the mathematical results were verified, the agent was also used to produce a first draft of a paper describing the results and future directions. While the generated draft was imperfect, the process of iterative human supervisor feedback and revisions is representative of scientific paper writing. It was therefore natural to ask whether this process could be distilled into a reusable skill. However, a naive instruction to "take this process that we just did and create a skill" produced an overfit result. Despite explicit (but evidently insufficient) requests to generalize the skill, the description still tied the procedure to this specific project:

```
description:  "Write and edit the LaTeX
paper for the Ramanujan 1914 CMF keys
project.  USE FOR: editing paper sections,
adding citations, updating tables/figures,
maintaining consistent notation, LaTeX
formatting.  DO NOT USE FOR: running code,
modifying computational scripts."
```

This failure is conceptually important: the agent successfully summarized the immediate process, but did not abstract it to the intended level. The resulting skill encoded accidental details of the project rather than the general principles of mathematical paper construction.

A related failure occurred during an attempt at automated revision. Once an initial draft existed, several LLMs were asked to critique it, identifying missing details, incoherent transitions, and unsupported claims. Their responses were aggregated to extract common criticisms and address only points on which multiple models agreed, with the aim of reducing model-specific preferences and stylistic noise. In practice, however, the resulting versions were less coherent and more cluttered: the agent reintroduced fragments that had been removed in earlier drafts without rebuilding the exposition around them, some mathematical details appeared before the relevant objects were defined, etc. In our experience, such iterations can therefore lead to a form of revision collapse: additional feedback cycles increasingly degrade the paper's self-contained structure unless the agent is constrained by explicit narrative, outline, and dependency structures.

5. Conclusion

This case study shows a practical division of labor in hybrid scientific discovery: the human provides problem selection and results interpretation, while the agent provides rapid experimentation and symbolic/numeric validation. The agent did not replace mathematical judgment or taste; rather, it made large numbers of experiments and checks cheap enough to perform, ultimately leading to a strong mathematical achievement.

The resulting time savings are best illustrated by two examples. First, the human mathematician supplied the broad idea that Ramanujan's series should be expressible as hypergeometric sums, based on their form and limiting behavior. The agent then searched for the specific hypergeometric parameters for each of the 17 formulas, turning a general structural clue into a concrete mapping of parameters per formula. Second, a human-guided proof-of-concept proved a single successful case of an embedding of a Ramanujan series in a CMF. Once this special case was validated, the agent was asked to apply the same logic and algorithm to the remaining formulas; this converted a single-case prototype into proofs for the entire collection. In both cases, the task was completed in hours, instead of days in a manual workflow.

The presented blueprint for human-AI hybrid research workflow substantially lowers the cost of early-stage creativity: it makes an initial hunch or idea - potentially too immature to justify heavy manual investment - cheap to test, refine, and validate through a prepared agent, thereby further democratizing science.

References

J. M. Borwein, P. B. Borwein, and D. H. Bailey. Ramanujan, modular equations, and approximations to pi or how to compute one billion digits of pi. *The American Mathematical Monthly*, 96(3):201–219, 1989. doi: 10.1080/00029890.1989.11972169.

R. Elimelech et al. Algorithm-assisted discovery of an intrinsic order among mathematical constants. *Proceedings of the National Academy of Sciences*, 121(25):e2321440121, 2024. doi: 10.1073/pnas.2321440121. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2321440121`.

E. First, M. N. Rabe, T. Ringer, and Y. Brun. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1229–1241. Association for Computing Machinery, 2023. doi: 10.1145/3611643.3616243.

A. Q. Jiang, S. Welleck, J. P. Zhou, T. Lacroix, J. Liu, W. Li, M. Jamnik, G. Lample, and Y. Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations*, 2023.

P. Paule et al. Ramanujan-type series. *RISC Report Series*, (25-01), 2025.

S. Ramanujan. Modular equations and approximations to π. *Quarterly Journal of Mathematics*, 45:350–372, 1914.

Ramanujan Machine Group. ramanujantools, 2025. URL `https://zenodo.org/doi/10.5281/zenodo.15747099`.

T. Raz et al. From Euler to AI: Unifying formulas for mathematical constants. In *Advances in Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=cNqMAmpZh4`.

M. Shalyt, U. Seligmann, I. Beit Halachmi, O. David, R. Elimelech, and I. Kaminer. Unsupervised discovery of formulas for mathematical constants. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 113156–113190, 2024.

A. Thakur, G. Tsoukalas, Y. Wen, J. Xin, and S. Chaudhuri. An in-context learning agent for formal theorem-proving. In *First Conference on Language Modeling*, 2024.

T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. doi: 10.1038/s41586-023-06747-5.

S. Weinbaum et al. On conservative matrix fields: Continuous asymptotics and arithmetic, 2025. URL `https://arxiv.org/abs/2507.08138`.

K. Yang, A. M. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. J. Prenger, and A. Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.