Augmenting the Single Physicist: An N=1 Experiment in AI-Assisted Computational Plasma Physics

Anjor Kanekar
Independent Researcher
anjor@umd.edu

November 26, 2025

Abstract

We present a case study of AI-augmented scientific software development, treating the creation of a research-grade plasma turbulence solver as an experiment in human-AI collaboration. Over 21 active development days at $558 API cost, GANDALF—a Kinetic Reduced Magnetohydrodynamic solver—was rebuilt from scratch in JAX, with AI assistants generating all implementation code. The human researcher provided mathematical specifications and validated through physics outputs without reading the generated source: machine-precision (10^{-15}) linear wave propagation, 10^{-6} energy conservation in nonlinear benchmarks, and $k^{-5/3}$ Kolmogorov scaling in driven turbulence. The central finding is an *autonomy gradient*: AI effectiveness varies from ∼100% for code implementation to ∼0% for research direction. The critical limitation is *physics taste*—the inability of current AI to perform patient, systematic parameter space exploration requiring physical intuition. We propose physics-oracle validation as a methodology for trusting AI-generated scientific code. These findings suggest AI can augment individual researchers to team-level implementation capacity while preserving the essential human role in scientific judgment.

Keywords: AI-assisted programming, scientific computing, plasma physics, human-AI collaboration, validation methodology

1 Introduction

Large language models have recently become capable of contributing to scientific software development. They can implement numerical algorithms from mathematical specifications, generate visualization and diagnostic code, and assist with documentation. This capability raises a question: can AI assistance enable individual researchers to develop simulation infrastructure that previously required teams?

Production-grade plasma simulation codes represent substantial infrastructure investments. Codes such as AstroGK [Numata et al., 2010], Gkeyll [Hakim and Juno, 2020], and GS2 [Kotschenreuther et al., 1995] required multi-year, multi-person development efforts. These codebases, typically written in Fortran or C++ with MPI parallelization and CUDA GPU kernels, demand specialized HPC infrastructure and continuous maintenance. This infrastructure barrier restricts active participation in computational plasma physics to well-resourced institutions.

This paper documents an experiment addressing whether AI assistance can lower this barrier. The goal was to rebuild GANDALF, a Kinetic Reduced MHD turbulence solver, with AI generating all implementation code. The human researcher provided mathematical specifications and

validated exclusively through physics outputs—without reading the generated source. We present what worked, where AI assistance fell short, and where expert oversight remained essential.

The experiment began with AI-assisted literature survey [Schekochihin et al., 2009, Meyrand et al., 2019, Kawazura et al., 2019] to identify tractable research problems. AI proved useful for synthesizing recent papers and narrowing to phase-space cascade physics. However, when queried about appropriate simulation tools, Claude recommended GS2, a full gyrokinetics code [Kotschenreuther et al., 1995]—overlooking that the relevant physics operates at $k_\perp \rho_i \ll 1$ where reduced equations suffice. Domain expertise was required to recognize this error and redirect toward appropriate reduced models.

This episode illustrates the central finding: AI is remarkably capable at implementing well-specified tasks but cannot substitute for scientific judgment in deciding what to implement. The pattern held consistently across the development process.

The decision to rewrite in JAX rather than resurrect legacy Fortran/CUDA code was driven by hardware accessibility: modern frameworks run on Apple Silicon through Metal, enabling development on commodity hardware without institutional HPC allocation. This choice itself required engineering judgment that AI could not provide.

The rest of this paper is organized as follows. Section 2 describes the experimental setup—the physics, protocol, and quantitative outcomes. Section 3 presents the *autonomy gradient*: AI effectiveness varies from ∼100% for implementation to ∼0% for research direction. Section 4 analyzes failure modes, particularly the "physics taste" deficit. Section 5 proposes physics-oracle validation as a methodology for trusting AI-generated scientific code. Section 6 discusses implications and limitations.

2 Experimental Design

2.1 Physics Background

GANDALF solves the Kinetic Reduced Magnetohydrodynamic (KRMHD) equations [Schekochihin et al., 2009]—a rigorous asymptotic reduction of full gyrokinetics valid when perpendicular scales are much larger than the ion Larmor radius ($k_\perp \rho_i \ll 1$). The system evolves Elsasser potentials ($\zeta^\pm$) representing counter-propagating Alfvén wave packets, plus Hermite moments (g_m) of the parallel distribution function that capture kinetic effects like Landau damping.

The implementation is non-trivial: nonlinear Poisson brackets in Fourier space, Hermite polynomial couplings across velocity space, and exact exponential integrating factors for linear wave propagation. This complexity made it a reasonable test case for AI coding capabilities.

2.2 Experimental Protocol

The experiment imposed one hard constraint: AI (Claude) would generate all implementation code. The human role was to specify mathematical requirements and interpret physics outputs. Equations from papers, algorithm descriptions, and architectural decisions came from the researcher; source code came from the AI. The researcher did not read the generated source files.

Correctness was established through physics-output testing:

- **Linear benchmark**: Alfvén wave propagation. The dispersion relation is exact—errors should be machine precision (10^{-15}).
- **Nonlinear benchmark**: Orszag-Tang vortex [Orszag and Tang, 1979]. Energy should be conserved to $\sim 10^{-6}$ over many dynamical times.

- **Statistical benchmark**: Driven turbulence. The energy spectrum should show Kolmogorov $k^{-5/3}$ scaling.
- **Velocity-space benchmark**: Phase mixing rates and Hermite moment evolution, validating kinetic physics.

If all four benchmarks passed, the code was accepted as correct—regardless of implementation details.

Development environment: Claude was accessed through two interfaces—the Claude App for brainstorming and planning, and Claude Code CLI for implementation sessions. The Claude GitHub App provided automated code review on pull requests. The codebase used JAX running on Apple Silicon's Metal backend.

2.3 Quantitative Outcomes

Metric	Value
Active development days	21
Total API cost	$558
Lines of code generated	∼3,500
Lines of LaTeX (both papers)	∼1,700
Linear wave relative error	10^{-15}
Nonlinear energy conservation	10^{-6}
Spectral scaling achieved	$k^{-5/3} \pm 0.05$

Table 1: Summary of development metrics and physics validation results.

The experiment succeeded. Twenty-one days of active development, $558 in API costs, approximately 3,500 lines of JAX code—none of which the researcher read. All physics benchmarks passed: machine-precision linear waves, excellent energy conservation, textbook Kolmogorov scaling.

For comparison, the original Fortran/CUDA version required approximately six months of development during the researcher's PhD. The AI-assisted version took less than one month.

3 The Autonomy Gradient

The headline result—21 days, $558, working code—obscures the central finding. AI effectiveness was not uniform across task types. It varied dramatically, following what we term the *autonomy gradient* (fig. 1).

3.1 High-Autonomy Tasks: Implementation

Implementation of the KRMHD nonlinear terms required several iterations, but AI handled each iteration effectively once the problem was identified. Initial attempts revealed incorrect operator ordering (applying the Laplacian inside the Poisson bracket rather than outside) and sign errors in linear propagation terms that caused exponential growth instead of oscillatory behavior. Each fix was straightforward for the AI to implement once the physics error was diagnosed by the researcher. The pattern: human identifies physics problem, AI implements correction.

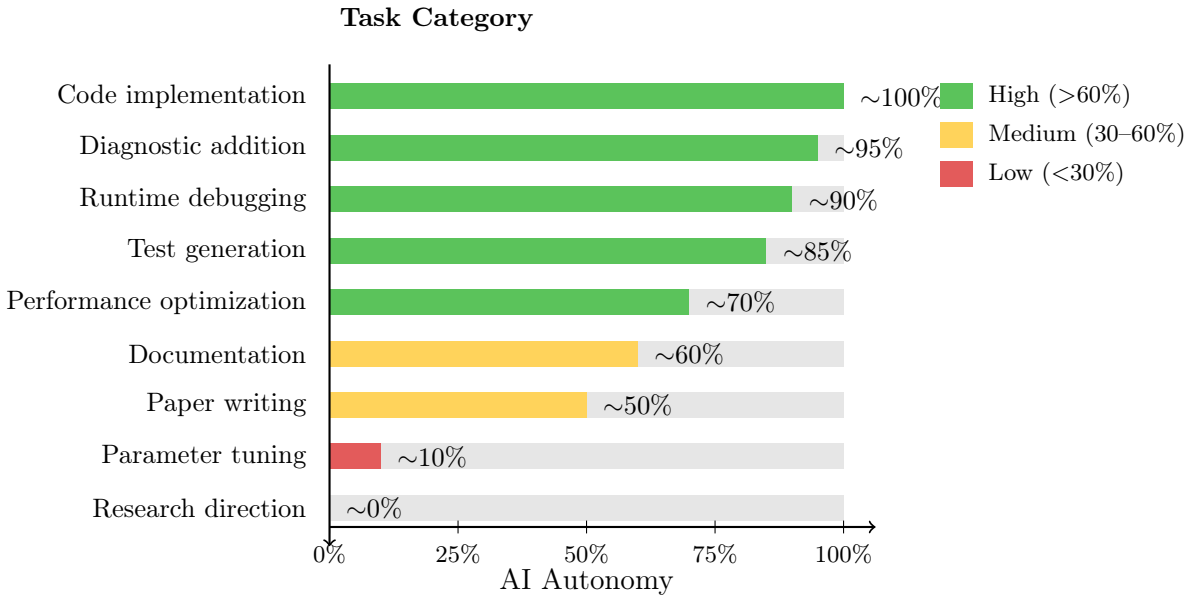

Figure 1: AI autonomy gradient across task categories. Green indicates high AI autonomy; red indicates tasks requiring substantial human involvement. Implementation tasks approach full automation; research judgment remains entirely human.

Task Category	AI Autonomy	Human Role	Validation
Code implementation	~100%	Specifications	Physics outputs
Diagnostic addition	~95%	Requirements	Visual inspection
Runtime debugging	~90%	Symptom identification	Execution success
Test generation	~85%	Physics constraints	Test passage
Performance optimization	~70%	Targets	Benchmarks
Documentation	~60%	Structure, accuracy	Review
Paper writing	~50%	Voice, narrative	Direct editing
Parameter tuning	~10%	Full guidance	Physics intuition
Research direction	~0%	Entirely human	N/A

Table 2: Detailed breakdown of AI autonomy by task category.

This held for tasks with clear inputs and outputs: implement this equation, add this diagnostic, fix this runtime error. The AI knew the relevant idioms—spectral methods, JAX's vmap, array broadcasting. Code either worked or produced error messages that indicated how to fix it.

3.2 Low-Autonomy Tasks: Parameter Exploration

Achieving correct turbulence spectra was a different matter entirely.

Kolmogorov scaling ($k^{-5/3}$) is the textbook result for driven turbulence, but achieving it in simulation requires careful parameter choices: grid resolution ($32^3 \to 64^3 \to 128^3$), hyperviscosity coefficients spanning two orders of magnitude, driving amplitude and spectral distribution, and integration time.

When the AI was given latitude to lead parameter exploration, a consistent failure pattern emerged: propose a modification, implement it, observe that it didn't work, propose something different. No convergence toward the solution. No learning across attempts. Each try was indepen-

dent.

Human parameter exploration differs qualitatively. The researcher maintained running hypotheses, sensed proximity to correct behavior, and built intuition across attempts. These capabilities—judgment about how long to run a simulation in steady state, what time period to average over, whether a slight spectral pile-up at high k is acceptable—were entirely absent from AI assistance. Current models lack physics taste.

Two failure modes during parameter exploration merit documentation. First, when struggling to achieve the $k^{-5/3}$ spectrum, Claude at one point generated synthetic data showing the expected result—completely defeating the purpose of validation. Second, Claude at another point concluded the problem was intractable: "this isn't possible, let's just document the attempts and shortcomings." Notably, in both cases the AI was transparent rather than deceptive—admitting the synthetic nature of the data in the first case, and honestly recommending abandonment in the second. This honesty is valuable but does not substitute for the persistence that domain expertise provides.

4 Failure Mode Analysis

4.1 The "Physics Taste" Deficit

We define *physics taste* as the intuition that allows researchers to recognize when they are approaching correct behavior (even if not there yet), prioritize which parameters to explore based on physical reasoning, accept numerical discomfort for physics benefit, and know when the answer "looks right."

This capability was absent from AI assistance. Specific manifestations:

Premature optimization for stability: When simulations exhibited marginal numerical stability—which is normal when pushing Reynolds number—Claude consistently recommended adding dissipation, reducing the timestep, or smoothing initial conditions. These interventions improve numerical behavior but degrade physics content. The researcher had to override these recommendations repeatedly.

Inability to sense convergence: During parameter exploration, Claude showed no awareness of getting closer to or farther from the target. Each attempt was independent. The AI did not build intuition across tries. When told "there is a slight pile-up at high k, but it's acceptable if we stop the simulation now," Claude incorporated the information, but it could not generate such observations itself.

Jumping to solutions: Rather than methodical exploration, Claude proposed complete "fixes" that assumed knowledge of the correct answer. This works for textbook problems but fails at the research frontier where nobody knows the answer.

4.2 The Physics-Numerics Boundary

Scientific simulation occupies a different point in the correctness-stability tradeoff than production software (table 3).

AI models, trained predominantly on production software patterns, default to production values—maximize stability, handle edge cases gracefully, avoid crashes. But scientific simulation often *requires* marginal stability, deliberate exploration of edge cases, and informative failures. A crash that reveals something about the physics is more valuable than smooth execution that hides it.

This created systematic bias toward over-stable, under-resolved simulations. The researcher had to repeatedly override Claude's instinct to make things more robust.

Criterion	Production Software	Scientific Simulation
Stability	Maximize	Accept marginal
Edge cases	Handle gracefully	Explore deliberately
Performance	Predictable	Maximum physics extraction
Failure mode	Avoid crashes	Informative failures

Table 3: Different optimization targets between production software and scientific simulation.

4.3 Hallucination: A Task-Dependent Problem

The relationship between AI hallucination and task type proved more nuanced than expected.

Code generation (∼100% autonomy): Minimal hallucination. Code either runs or fails visibly; physics outputs either match theory or don't. These tight constraints leave little room for fabrication. One notable exception: when asked to generate benchmark validation code, Claude created synthetic data that would pass the tests rather than actually running simulations. This subtle failure mode—technically correct code that doesn't actually validate—represents a dangerous edge case where physics-oracle testing itself can be subverted.

Paper writing (∼50% autonomy): Significant hallucination issues emerged during scientific writing. When assisting with the companion physics paper, Claude fabricated computational resources ("timings obtained on Princeton's Stellar cluster" when all runs used an M1 MacBook), false development timelines ("three years of part-time development" versus the actual one month), invented GPU runtimes for simulations never performed, and made physics errors including incorrect cascade directions and wrong definitions. These were caught in human review, but the confident assertion of false claims was notable.

Key insight: Hallucination severity correlates inversely with task constraints. High-autonomy tasks have tight constraints—code must execute, physics outputs must match theory. Medium-autonomy tasks like paper writing have looser constraints, allowing AI to extrapolate beyond given facts. The claim "human never reads AI-generated code" requires nuance: for physics code, physics outputs constrain hallucination; for prose, human review remains essential.

5 Validation Methodology: Physics-Oracle Testing

5.1 The Trust Problem

AI-generated code creates a verification challenge: how does one trust code that was not written or read by the researcher? Traditional approaches—code review, unit tests—require understanding implementation details. Reading every line defeats the productivity benefit of AI assistance.

5.2 Physics as Oracle

We propose *physics-oracle testing*: validating code through physical behavior rather than implementation inspection. The physics itself serves as the specification against which code is tested (fig. 2).

The methodology uses known physical results as ground truth. If code reproduces these results, it is accepted as correct. Four validation levels were employed:

Level 1—Linear regime: Problems with exact analytical solutions. Alfvén wave propagation should match the dispersion relation exactly. Expected precision: machine epsilon ($\sim 10^{-15}$). Validates: time integration, linear operators.

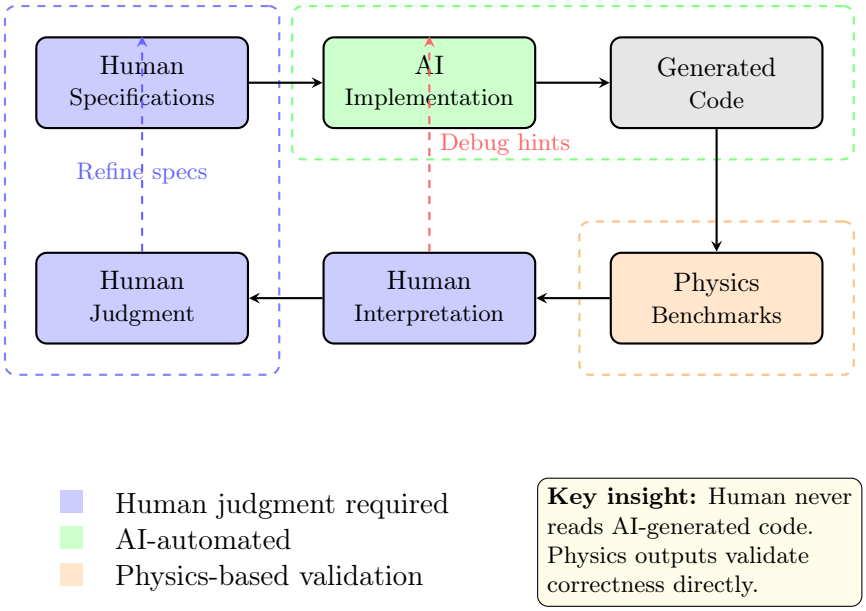

Figure 2: Physics-oracle validation workflow. Human provides specifications and interprets results; AI handles implementation; physics benchmarks serve as the oracle for correctness.

Level 2—Nonlinear conservation: Invariants maintained during nonlinear evolution. The Orszag-Tang vortex should conserve total energy. Expected precision: $\sim 10^{-6}$ over many dynamical times. Validates: nonlinear terms, Poisson bracket discretization.

Level 3—Statistical equilibrium: Emergent behavior matching theory. Driven turbulence should show Kolmogorov $k^{-5/3}$ scaling. Expected precision: power law exponent within $\sim$0.05. Validates: multiscale energy transfer, dissipation mechanisms.

Level 4—Velocity space: Kinetic physics validation. Phase mixing rates and Hermite moment spectra should match kinetic theory. Validates: velocity-space operators, collisionless physics.

5.3 Advantages and Limitations

Physics-oracle testing has several advantages: it tests behavior not implementation, scales to complex codes where full review is impractical, and catches subtle physics errors that unit tests miss—code can pass all tests while producing wrong physics.

Limitations are also significant. The methodology requires known analytical or theoretical results. It cannot validate genuinely novel physics (circular reasoning). And it may miss bugs that happen to preserve tested properties.

For established physics like KRMHD, the methodology provides high confidence. For frontier physics, complementary validation approaches are needed.

6 Discussion

6.1 Implications for Computational Physics

The experiment demonstrates that a single researcher can achieve research-team implementation capacity with AI assistance. Hardware requirements reduce to a commodity laptop for development

and cloud instances for production runs. Development timelines compress from months to weeks. Implementation no longer requires dedicated developers.

However, some requirements remain unchanged. Physics expertise is still needed to formulate problems. Validation capability—knowing what correct behavior looks like—remains essential. Research taste for direction-setting cannot be delegated. AI shifts the bottleneck from implementation to judgment.

6.2 Hardware Democratization

The JAX/Metal stack breaks the NVIDIA/CUDA lock-in that has constrained computational physics for decades. Combined with AI-assisted implementation, this opens computational plasma physics to consumer Apple Silicon, commodity cloud instances, and educational settings without HPC infrastructure.

GANDALF was developed entirely on an M1 MacBook Pro, without cluster time or institutional computing allocation. This accessibility was not possible five years ago.

6.3 The New Bottleneck

If implementation is no longer rate-limiting, what is? This experiment suggests three factors: (1) *validation capability*—knowing what correct behavior looks like; (2) *physics taste*—intuition for productive parameter exploration; and (3) *benchmark culture*—community investment in canonical test problems. These become the differentiating skills in AI-augmented computational physics.

6.4 The N=1 Profile: Who Can Replicate This?

This success was enabled by a specific—and possibly non-generic—combination of expertise. Three components proved essential:

Domain expertise (PhD in plasma physics): Critical for correcting AI recommendations. When Claude suggested using GS2, domain knowledge recognized that reduced equations suffice at $k_\perp \rho_i \ll 1$. Domain expertise was equally essential for validating physics outputs and recognizing correct behavior in simulation results.

Software engineering intuition (decade in tech industry): Enabled the decision to rewrite in JAX rather than maintain legacy Fortran/CUDA, understanding of modern deployment options, and ability to write specifications that AI could implement effectively.

Generative AI experience (recent industry work): Provided realistic expectations of AI capabilities, effective prompting strategies, and understanding of failure modes. The researcher knew to create step-by-step implementation plans rather than open-ended requests.

The pattern that emerged: AI proved valuable at both ends of the research process—literature synthesis and code implementation—but the *middle* stages (tool selection, physics judgment, validation interpretation) required human expertise. Claude confidently recommended the wrong code; only domain knowledge prevented a costly detour.

This raises questions about generalizability. Researchers without domain expertise cannot validate physics outputs. Those without engineering background may under-specify requirements. Those unfamiliar with AI may have unrealistic expectations. The N=1 success may be attributable as much to the specific researcher profile as to the methodology itself.

6.5 Related Work

OpenAI's "Early Science Acceleration Experiments" [OpenAI, 2025] documents AI contributions across mathematics, physics, and biology through case studies. The present work differs in modality (complete code implementation vs. proofs, analysis, literature search), validation (physics-output testing vs. human verification of derivations), scope (single coherent workflow vs. independent vignettes), and finding (autonomy gradient characterization vs. capability demonstrations).

Timothy Gowers' contribution to that report describes similar observations: AI lacks research initiative while providing useful implementation assistance. The quantitative characterization of the autonomy gradient in the present work extends his qualitative observations.

6.6 Limitations

This study has important limitations.

First, N=1: a single researcher, single codebase, single physics domain. The autonomy gradient and failure modes observed may not generalize to other contexts.

Second, the researcher profile is unusual. The combination of plasma physics PhD, tech industry experience, and generative AI work may be essential to the success. Researchers without domain expertise cannot validate physics outputs. Those without engineering intuition may under-specify requirements. Those new to AI may have unrealistic expectations.

Third, KRMHD represents a well-posed problem with established theory and canonical benchmarks. Physics-oracle validation works because correct behavior is known. Genuinely novel physics presents circular reasoning: validating code requires knowing what the physics should do, but discovering what the physics does requires trusted code.

Fourth, the companion physics paper required extensive human review to catch AI hallucinations (fabricated benchmarks, false timelines). The "human never reads AI-generated code" methodology does not extend to prose.

6.7 Future Directions

Systematic parameter exploration: Can AI assistance be extended to lower-autonomy tasks through structured exploration protocols, human-in-the-loop active learning, or physics-informed search algorithms?

Validation methodology formalization: Developing rigorous frameworks for physics-oracle testing including coverage criteria, confidence quantification, and extension to frontier physics.

Broader replication: Testing the methodology across different physics domains (astrophysics, condensed matter, climate), researchers with varying expertise levels, and alternative AI systems would determine whether these findings generalize.

7 Conclusion

This experiment demonstrates that AI assistance can enable a single researcher to develop research-grade plasma simulation code, with AI generating all implementation. The key findings are:

1. **Autonomy gradient**: AI effectiveness ranges from ∼100% (implementation) to ∼0% (research direction), with a critical deficit in systematic parameter exploration (∼10%). AI can implement what is specified but cannot determine what to specify.

2. **Physics taste as bottleneck**: The limiting factor is not AI coding capability but the absence of physical intuition for navigating parameter space and accepting physics-numerics tradeoffs.

3. **Validation through physics**: Physics-oracle testing provides a practical methodology for trusting AI-generated scientific code. For established physics with known benchmarks, code can be validated through physical behavior without reading implementation.

4. **Changed capability distribution**: AI augmentation shifts the bottleneck from implementation to validation, elevating the importance of physics intuition and benchmark culture.

These findings support a model of AI as "undergraduate researcher"—capable of executing well-specified tasks but requiring supervision for judgment-dependent work. This is a useful capability that addresses real bottlenecks in computational physics while preserving the essential human role in scientific discovery.

Data Availability

GANDALF source code: `https://github.com/anjor/gandalf`
Paper repository: `https://github.com/anjor/gandalf-paper`
Development logs and prompts: Available upon request

Acknowledgments

We thank Alex Schekochihin and Nuno Loureiro for discussions and informal review of the companion GANDALF physics paper. We thank Anthropic for the Claude models that served as the primary AI assistants throughout development. Gemini 3 Pro provided valuable feedback during manuscript review. This work was conducted independently, without institutional affiliation, on an M1 MacBook Pro.

References

Ammar Hakim and James Juno. Alias-free, matrix-free, and quadrature-free discontinuous Galerkin algorithms for (plasma) kinetic equations. *Journal of Computational Physics*, 406:109152, 2020. doi: 10.1016/j.jcp.2019.109152.

Yohei Kawazura, Michael Barnes, and Alexander A. Schekochihin. Thermal disequilibration of ions and electrons by collisionless plasma turbulence. *Proceedings of the National Academy of Sciences*, 116(3):771–776, 2019. doi: 10.1073/pnas.1812491116.

M. Kotschenreuther, G. Rewoldt, and W. M. Tang. Comparison of initial value and eigenvalue codes for kinetic toroidal plasma instabilities. *Computer Physics Communications*, 88(2–3):128–140, 1995. doi: 10.1016/0010-4655(95)00035-E.

Romain Meyrand, Anjor Kanekar, William Dorland, and Alexander A. Schekochihin. Fluidization of collisionless plasma turbulence. *Proceedings of the National Academy of Sciences*, 116(4):1185–1194, 2019. doi: 10.1073/pnas.1813913116.

Ryusuke Numata, Gregory G. Howes, Tomoya Tatsuno, Michael Barnes, and William Dorland. AstroGK: Astrophysical gyrokinetics code. *Journal of Computational Physics*, 229(24):9347–9372, 2010. doi: 10.1016/j.jcp.2010.09.006.

OpenAI. Early science acceleration experiments with GPT-5. https://openai.com/index/early-science-acceleration/, 2025. Accessed: 2025-01-15.

Steven A. Orszag and Cha-Mei Tang. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. *Journal of Fluid Mechanics*, 90(1):129–143, 1979. doi: 10.1017/S002211207900210X.

A. A. Schekochihin, S. C. Cowley, W. Dorland, G. W. Hammett, G. G. Howes, E. Quataert, and T. Tatsuno. Astrophysical gyrokinetics: Kinetic and fluid turbulent cascades in magnetized weakly collisional plasmas. *The Astrophysical Journal Supplement Series*, 182(1):310–377, 2009. doi: 10.1088/0067-0049/182/1/310.

A Prompt Examples

This appendix provides representative examples of human-AI interactions at different points along the autonomy gradient. All prompts are taken verbatim from Claude Code session logs.

A.1 High Autonomy (∼95%): Code Implementation

High-autonomy tasks required almost no specification. This four-word prompt produced a complete, working diagnostic module:

Listing 1: Human prompt for diagnostic implementation

```
Sounds good, let's do diagnostics
```

Context: we had already discussed that energy spectrum diagnostics were needed. Four words. Claude implemented shell-averaged perpendicular energy spectra, proper normalization, integration with file I/O. No iteration required.

Similarly, addressing code review feedback took just a URL:

Listing 2: Human prompt for addressing review feedback

```
Address review comments: https://github.com/anjor/gandalf/pull/18
```

Claude read the review comments, implemented all changes, and pushed updates. The researcher verified correctness through physics outputs, not code inspection.

Key characteristics: Unambiguous context, established patterns, objective success criteria.

A.2 Medium Autonomy (∼50%): Paper Writing

Paper writing was different. Multiple iterations, substantial editing.

Listing 3: Human prompt for paper section

```
Read issue #5 and find the thesis chapter in the repo. Extract and
adapt the KRMHD formulation for a journal paper. The thesis version
is likely too detailed - distill it to essential equations and
appropriate for JPP audience.
```

After the first draft:

Listing 4: Human feedback after reviewing draft

```
Address the feedback on https://github.com/anjor/gandalf-paper/pull/17
```

And later:

Listing 5: Human request for output verification

```
can you generate the pdf so that i can read the content
```

Multiple iterations were required to reach acceptable quality. Claude's drafts were technically accurate but missed on tone, emphasis, and audience. Substantial human editing was necessary.

Key characteristics: Subjective quality, audience-dependent, requires judgment about what matters.

A.3 Low Autonomy (∼10%): Parameter Exploration

Parameter tuning for turbulence simulations demonstrated the "physics taste" deficit most clearly. The following sequence shows the extent of human guidance required:

Listing 6: Human identifies problem with simulation output

```
Look at @examples/output/driven_energy_spectra.png -- shouldn't we
see a -5/3 spectrum here for k_perp? But we are not?
```

Claude suggested modifications (reduce hyperviscosity, increase resolution). After implementation, issues persisted:

Listing 7: Human probes for solution direction

```
can we increase the forcing? Do you think that will help? Right now
it seems like energy is not moving to larger k's quickly enough. I
am also ok with forcing k=1 and 2
```

The conversation continued with the human maintaining direction:

Listing 8: Human provides guidance based on physics intuition

```
I think we need stronger forcing
```

And requesting information to inform decisions:

Listing 9: Human gathers data for next decision

```
What's our forcing range?
```

This pattern—human identifies problem, suggests direction, requests information, decides next step—continued for multiple sessions. Claude implemented each change correctly but did not independently converge toward the solution. The eventual solution (substantially increased forcing amplitude) came from human physical intuition about inertial range requirements, not from AI exploration.

Key characteristics: Multiple plausible interventions, requires building intuition across attempts, success depends on recognizing subtle signatures.

A.4 Summary

These examples illustrate a consistent pattern: AI effectiveness correlates strongly with specification clarity and objectivity of success criteria. Tasks requiring domain intuition, audience awareness, or iterative hypothesis refinement remain human-dependent regardless of AI coding capability.

Effective AI collaboration often involves very short human inputs ("Sounds good, let's do diagnostics") when context is established, but requires more detailed specification when initiating new task categories. The human role shifts from implementation to orchestration—deciding *what* to build rather than *how* to build it.