Using Advanced LLMs to Improve Numerics in a High-Fidelity Gyrokinetic Simulation Code: a Case Study

Gregory W. Hammett[1,3], Manaure Francisquez[1], Ammar Hakim[2]
Antoine C. D. Hoffmann[1], Maxwell H. Rosen[3]

[1] Princeton Plasma Physics Laboratory (PPPL)
[2] Lanyon AI
[3] Princeton University

hammett@pppl.gov

w3.pppl.gov/~hammett

APS-DPP, Chicago, Nov. 2-6, 2026

Abstract

We have used advanced LLMs (ChatGPT, Claude, and Gemini) to improve algorithms in the Gkeyll code for nonlinear 5D gyrokinetic simulations of plasma turbulence and 4D axisymmetric simulations of kinetic plasma-divertor interactions. Gkeyll uses a special discontinuous Galerkin algorithm that conserves energy for Hamiltonian systems in continuous time, like the Arakawa method, even with upwinding [1,2]. This algorithm works well in many regimes, but we found special cases that exhibit a subtle numerical instability. ChatGPT 5.5/5.6 at the highest Pro reasoning levels analyzed this in detail, proposed several algorithmic improvements, and eliminated the instability. [Later used ChatGPT 6 Astra Max.] It also showed how to implement a nonlinear Boltzmann relation $e\,\phi(z) = T\ln\left(n_e(z)/C\right)$ with positivity limits while conserving energy. ChatGPT's analysis appears comparable in technical level to current research papers, drawing on adjoint properties and other general mathematical and physical principles. This case study demonstrates how LLMs can improve the robustness of high-fidelity HPC codes needed to extend ML-based models to new parameter regimes.

[1] A. Hakim et al., 2019, arXiv:1908.01814
[2] M. Francisquez et al., 2026, https://doi.org/10.1016/j.jcp.2026.114852

Acknowledgments

This work was supported by a DOE Distinguished Scientist award, the CEDA SciDAC project, and by the DOE Office of Science, Office of Fusion Energy Sciences, as part of the Genesis Mission National Science and Technology Challenges, under contract DE-AC02-09CH11466 at the Princeton Plasma Physics Laboratory.

This project used AI LLMs in carrying out research and preparing the presentation, but I take responsibility for the results.

Using AI to Discover and fix a Numerical Streaming Instability

- Gkeyll uses a special form of DG that conserves energy exactly for Hamiltonian problems (in the continuous time limit, like the Arakawa method), and ensures that entropy decays (with upwinding) or conserved (with central fluxes). This works well in many cases, but occasionally we have found oscillations or unusual features in the solution in some places in certain parameter regimes.
- We isolated a 1x-1v test problem that illustrated unusual oscillations, and after giving these results and some of our algorithms papers to ChatGPT 5.6 Pro, and iterating a few times, it identified 2 types of subtle numerical instabilities that can occur in some parameter regimes.
- This moderate streaming instability occurs in a cold-ion regime with $v_{ti} \ll u_0 < c_{s0}/2$ so that the numerical instabilty can overcome ion Landau damping. (Here v_{ti} is the ion thermal velocity, u_0 is the streaming velocity, and $c_{s0} = \sqrt{T_e/m_i}$ is the cold ion sound speed.)
- This instability can affect many algorithms with upwinding or numerical diffusion (finite difference, finite volume, Semi-Lagrangian? [Mawathage et al. this conference, DPP-FP11/74], ...)
- Fixing this instability improves Gkeyll's robustness, works well over a wider range of parameters.

Paradigm problem: 1x-1v Ions and adiabatic electrons.

Ion acoustic waves with a background mean flow, 1x-1v kinetic ions, linearized adiabatic/Boltzmann electrons, quasineutality

Free energy with a shifted Maxwellian

Kinetic ions with hyperdiffusion, quasineutral Boltzmann electrons:

$$\partial_t \widetilde{f} + v\partial_z \widetilde{f} = \frac{e}{m_i}(\partial_z \phi)\,\partial_v F_0 - D(v)\partial_z^4 \widetilde{f}, \qquad n_0 e\phi/T_{e0} = \int dv \widetilde{f}.$$

$$\partial_t \widetilde{f} + v\partial_z \widetilde{f} = \frac{e}{T_i}\widetilde{E}\,(v-u_0)F_0 - D(v)\partial_z^4 \widetilde{f}, \qquad .$$

Multiply by $T_i \widetilde{f}/F_0$, integrate over z, v, and use density eq. $\partial_t \widetilde{n} = \ldots$ (including hyperdiffusion) :

$$W = \frac{T_i}{2}\int dz\,dv\,\frac{\widetilde{f}^2}{F_0} + \frac{T_{e0}}{2n_0}\int dz\,\widetilde{n}^2.$$

($dW/dt \leq 0$ prevents unstable modes)

$$\frac{dW}{dt} = -u_0 e \int dz\,\widetilde{n}\,\widetilde{E} - T_i \int dz\,dv\,\frac{D(v)}{F_0}(\partial_z^2 \widetilde{f})^2$$
$$-e\int dz\,(\partial_z^2\widetilde{\phi})\int dv\,D(v)\partial_z^2\widetilde{f}.$$

Two possible sources of numerical instability

1. Hyperdiffusion can drive electron compression. For constant $D(v) = D_0$,

$$-e \int dz\, (\partial_z^2 \widetilde{\phi}) \int dv\, D_0 \partial_z^2 \widetilde{f} = -\frac{D_0 T_{e0}}{n_0} \int dz\, (\partial_z^2 \widetilde{n})^2 \leq 0.$$

For $D(v)$, this term has no definite sign and can drive instability. (Modifying hyperdiffusion to damp only non-Boltzmann part, $\widetilde{f} + e\phi F_0/T_i$, (Catto 1978), restores definite damping.)

2. A momentum defect couples to the mean flow. Analytically, $\widetilde{n} \propto \phi$ so

$$-u_0 \int dz\, \widetilde{n} \widetilde{E} \propto u_0 \int dz\, \widetilde{\phi} \partial_z \widetilde{\phi} = 0.$$

But in original DG algorithm $\widetilde{n}$ is only weakly proportional to ϕ, in the space of continuous functions, not in the space of discontinuous functions where $\widetilde{E}$ lives.

Projected field energy: $W_{e,h} = \frac{e}{2} \int dz\, \widetilde{n}_h \widetilde{\phi}_h$. Use a self-adjoint response and consistent cancellation of field work.

Cold-ion moments with velocity-dependent diffusion

First-order upwinding gives $D(v) = \alpha v$ for $v > 0$, with $\alpha = \Delta x/2$:

$$\partial_t f + v\partial_x f = \alpha v \partial_x^2 f + \frac{e}{m_i}(\partial_x \phi)\partial_v f, \qquad n = n_0 \exp(e\phi/T_{e0}).$$

Take density and momentum moments with the cold-ion closure, $\int v^2 f \, dv = nu^2$:

$$\partial_t n + \partial_x(nu) = \alpha \partial_x^2 (nu),$$
$$\partial_t (nu) + \partial_x(nu^2) = \alpha \partial_x^2 (nu^2) - c_s^2 \partial_x n.$$

Linearize about $F_0 = n_0 \delta(v - u_0)$, $u_0 > 0$, with $n = n_0 + \tilde{n}$, $u = u_0 + \tilde{u}$:

$$(\partial_t + u_0\partial_x)\tilde{n} + n_0 \partial_x \tilde{u} = \alpha u_0 \partial_x^2 \tilde{n} + \alpha n_0 \partial_x^2 \tilde{u},$$
$$(\partial_t + u_0\partial_x)\tilde{u} = -\frac{c_s^2}{n_0}\partial_x \tilde{n} + \alpha u_0 \partial_x^2 \tilde{u}.$$

The extra diffusion term in continuity retains the velocity dependence of $D(v)$. Freezing D at αu_0 would omit this term.

Dispersion relation and numerical growth

Use $(\widetilde{n}, \widetilde{u}) \propto \exp(ikx - i\omega t)$, $k > 0$, with $\gamma = \operatorname{Im}\omega$. Eliminating the amplitudes gives:

$$\boxed{\left(\omega - ku_0 + i\alpha u_0 k^2\right)^2 = k^2 c_s^2 (1 - i\alpha k).}$$

Its exact roots, with no expansion in u_0/c_s, are

$$\boxed{\omega_\pm = ku_0 - i\alpha u_0 k^2 \pm kc_s\sqrt{1 - i\alpha k}.}$$

For $k\Delta x \ll 1$, expand $\sqrt{1 - i\alpha k} \simeq 1 - i\alpha k/2$:

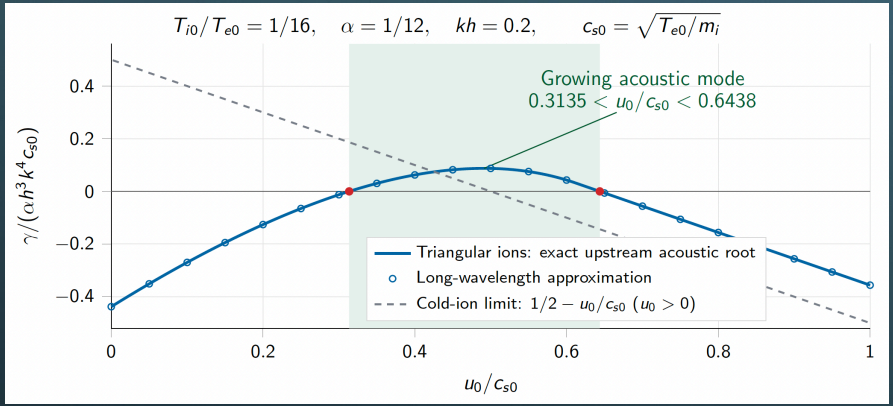
$$\omega_\pm = k(u_0 \pm c_s) - i\alpha k^2 \left(u_0 \pm \frac{c_s}{2}\right) + O(\alpha^2 k^3 c_s).$$

The upstream branch has phase speed $u_0 - c_s$ and, to leading order,

$$\boxed{\gamma_- \simeq \alpha k^2 \left(\frac{c_s}{2} - u_0\right), \qquad 0 < u_0 < \frac{c_s}{2} \text{ gives growth.}}$$

For $0 < u_0 \ll c_s$, $\gamma_- \simeq c_s \Delta x\, k^2/4$. The downstream branch damps.

Growth rate including finite ion temperature



With finite ion temperature T_i, physical Landau damping stabilizes this mode for small u_0, but there is still a numerical instability for intermediate flow velocities u_0.

Centered DG repair: retain the complete density response

Periodic z, kinetic ions and Boltzmann electrons; centered fluxes and no diffusion.

The original continuous ϕ_h^C satisfies quasineutrality only against continuous test functions. Since E_h^C is discontinuous, this does not imply

$$\sum_j \int_{I_j} \widetilde{n}_h E_h^C \, dz = 0.$$

Use the full DG density to determine a discontinuous potential:

$$\widetilde{n}_h = \int \widetilde{f}_h \, dv, \qquad e\phi_h = \frac{T_{e0}}{n_0} \widetilde{n}_h \quad \text{in every cell and every DG mode.}$$

For cell width h and $\xi = 2(z - z_j)/h \in [-1, 1]$,

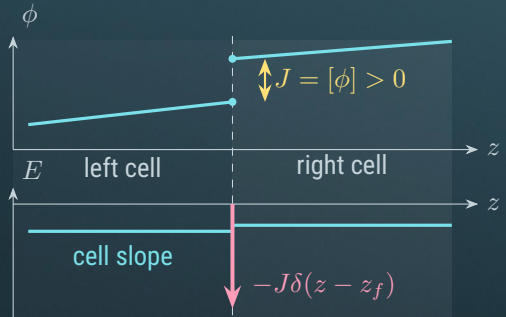
$$\widetilde{f}_j = \widetilde{f}_{j0}(v) + \widetilde{f}_{j1}(v)\xi, \qquad \phi_j = a_j + b_j \xi, \qquad \begin{pmatrix} a_j \\ b_j \end{pmatrix} = \frac{T_{e0}}{e n_0} \int dv \begin{pmatrix} \widetilde{f}_{j0} \\ \widetilde{f}_{j1} \end{pmatrix}.$$

The force must include both the cell slopes and the interface jumps of ϕ_h. The resulting E_h has both constant and linear components.

A jump in potential produces an interface electric field

Let $[\phi] = \phi^+ - \phi^-$, with $-$ and $+$ denoting left and right traces. $E = -\partial_z \phi$ contains delta functions:

$$E = -\partial_z \phi_h = -(\partial_z \phi_h)_{\text{cell}} - \sum_{\text{faces } f} [\phi]_f \, \delta(z - z_f).$$



Arrow denotes integrated weight $-J$, not a finite height.

A DG test w has two interface traces. Centered fluxes use their average:

$$\{w\} = \frac{w^- + w^+}{2}.$$

One interface contributes

$$\int w E_{\text{jump}} \, dz = -\frac{J}{2}(w^- + w^+).$$

The weight $-J$ is shared equally. For continuous w, recover $-Jw(z_f)$.

Project the interface force onto both linear basis functions

Project a unit delta at the cell's right edge as $\ell_R = A + B\xi$:

$$\int_{I_j} w\ell_R \, dz = w(1), \quad w = 1, \xi \quad \Longrightarrow \quad hA = 1, \quad \frac{h}{3}B = 1.$$

The projected delta functions are therefore

$$\ell_R(\xi) = \frac{1+3\xi}{h}, \qquad \ell_L(\xi) = \frac{1-3\xi}{h}.$$

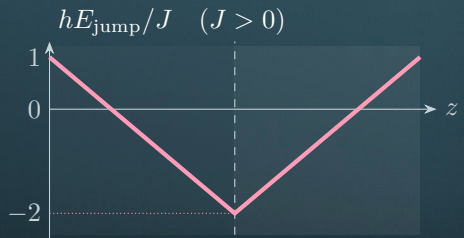
At one interface, centered projection gives

$$E_{\text{jump}} = \begin{cases} -\dfrac{J}{2h}(1+3\xi_{\text{left}}), & \text{left cell,} \\ -\dfrac{J}{2h}(1-3\xi_{\text{right}}), & \text{right cell.} \end{cases}$$

Each cell has $\int E_{\text{jump}}\, dz = -J/2$.

Both signs needed to match the linear test-function moment.

Shown: one interface. Add the cell slope and the other interface.



The complete force: compute both electric-field coefficients

Advance the linearized kinetic equation using the centered DG derivative D_c:

$$\partial_t \widetilde{f}_h + v D_c \widetilde{f}_h = -\frac{e}{m_i} E_h \partial_v F_0, \qquad E_h = -D_c \phi_h, \qquad \widehat{\widetilde{f}} = \frac{\widetilde{f}^- + \widetilde{f}^+}{2}.$$

Use centered potential traces $\widehat{\phi} = (\phi^- + \phi^+)/2$. In each cell, define field weakly for $w = 1, \xi$:

$$\int_{I_j} w E_j \, dz = \int_{I_j} (\partial_z w) \phi_j \, dz - w(1)\widehat{\phi}_R + w(-1)\widehat{\phi}_L.$$

For $\phi_j = a_j + b_j \xi$, the complete field is $E_j = E_{j0} + E_{j1}\xi$, with

$$\boxed{E_{j0} = -\frac{\widehat{\phi}_R - \widehat{\phi}_L}{h}, \qquad E_{j1} = -\frac{3}{h}(\widehat{\phi}_R + \widehat{\phi}_L - 2a_j).}$$

Both coefficients enter the corresponding distribution-function modes:

$$\left(\partial_t \widetilde{f}_{j0}\right)_E = -\frac{e}{m_i} E_{j0} \partial_v F_0, \qquad \left(\partial_t \widetilde{f}_{j1}\right)_E = -\frac{e}{m_i} E_{j1} \partial_v F_0.$$

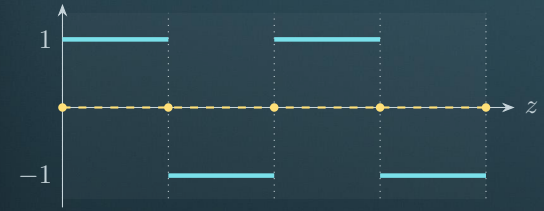
This field calculation uses only the cell and its two neighbors. The Boltzmann response requires no global field solve.

Even piecewise-constant potential can require a linear force

Consider alternating cell averages: $a_j = (-1)^j A$, $b_j = 0$. The centered potential trace vanishes at every face, but

$$\widehat{\phi}_L = \widehat{\phi}_R = 0 \quad \Longrightarrow \quad \boxed{E_{j0} = 0, \qquad E_{j1} = \frac{6a_j}{h}.}$$

DG potential ϕ_h/A



Centered face values and continuous P^1 projection: zero

Complete DG field hE_h/A



Interface jumps supply the missing linear force.

For this density pattern, a continuous P^1 field solve returns $\phi_h^C = 0$. The full DG response retains a nonzero restoring force.

The discrete free energy is conserved for any mean flow

Shifted Maxwellian: $\partial_v F_0/F_0 = -m_i(v-u_0)/T_i$. Periodic centered DG gives

$$\int dz\, w D_c g = -\int dz\, g D_c w.$$

Multiply by $T_i \widetilde{f}_h/F_0$, integrate, and use $\partial_t \widetilde{n}_h = -D_c\widetilde{\Gamma}_h$, $\widetilde{\Gamma}_h = \int v\widetilde{f}_h\, dv$:

$$\boxed{W_h = W_{i,h} + W_{e,h} = \frac{T_i}{2}\int dz\, dv\, \frac{\widetilde{f}_h^2}{F_0} + \frac{T_{e0}}{2n_0}\int dz\, \widetilde{n}_h^2.}$$

$$\frac{dW_{i,h}}{dt} = e\int dz\, E_h(\widetilde{\Gamma}_h - u_0\widetilde{n}_h), \qquad \frac{dW_{e,h}}{dt} = -e\int dz\, E_h\widetilde{\Gamma}_h.$$

$$\boxed{\frac{dW_h}{dt} = -u_0 e\int dz\, \widetilde{n}_h E_h = \frac{u_0 T_{e0}}{n_0}\int dz\, \widetilde{n}_h D_c\widetilde{n}_h = 0.}$$

Positive W_h rules out linear exponential growth for any u_0.

Linear spatial semidiscretization, with exact velocity moments and consistent spatial quadrature. Centered fluxes can leave neutral grid modes; they add no damping.

AI / LLMs are great tools but still have limits.

- Extremely Impressive, but still not perfect. Doesn't make simple direct mistakes (able to do complicated math very well), but sometimes doesn't consider broader implications of what it's doing.
  - Said that numerical instability can be avoided if $\mathrm{d}/\mathrm{d}z$ and ∇^2 commute, and proposed that ∇^2 be calculated with 2 applications of $\mathrm{d}/\mathrm{d}z$. But that means $\mathrm{d}^2/\mathrm{d}z^2 = \ldots$ couples only every other grid point, and leads to checkerboard pattern of undamped errors and inefficiencies in wasting half of the grid resolution (or a factor of 64 in 5D+time). When asked about this, it acknowledged the problem, and said that its approach was overly restrictive and simplistic, and there are better ways to have the discretized $\mathrm{d}/\mathrm{d}z$ and ∇^2 operators commute…

Practical advice for using AI LLMs (1/3)

In ChatGPT Settings, Personalization, give it background info on you to give it context, so it knows you aren't a high school student and that you aren't asking for creative fiction.

- "Occupation": Computational plasma physicist and plasma physics professor, focused on fusion energy, gyrokinetics, and numerical methods.
- "More about you" (not all my wording, based on its suggestions): I am technically sophisticated and prefer answers aimed at a graduate-level physicist. I value mathematical correctness, clearly stated assumptions, dimensional checks, and concise tutorial explanations. I often need rigorous derivations, exam/tutorial-style solutions, and literature assessment. I do not need elementary background unless it is needed to avoid a common mistake.

Also, in ChatGPT, Settings, Data Control:

- turn off "Improve the model for everyone" to prevent OpenAI from using your ideas and prevent your information from leaking to others. Similar options in Claude Settings, Privacy.

Practical advice for using AI LLMs (2/3)

In Settings, Personalization, I set extra instructions used for all chats to (part 1/2)

You are primarily to act as a physics research assistant to a plasma physicist at a major research university, although I will occasionally ask you broader non-physics questions also.

Use LaTeX for mathematical notation, including inline symbols, rather than Unicode substitutes or plain-text approximations. For inline math use `\(...\)` delimiting. For display math (on its own line), use
`$$`
`...`
`$$`
for delimiting, with each delimiter on its own line. Keep Markdown formatting outside math environments. Unless I request source code, present equations as rendered math rather than in code blocks.

When creating Markdown files, or providing Markdown source intended for GitHub or VS Code's Markdown Preview, use `$...$` for inline math and use
`$$`
`...`
`$$`
for display math.

Practical advice for using AI LLMs (2/3)

In Settings, Personalization, I set extra instructions used for all chats to (part 2/2):

For derivations or exam-style solutions, show the logical steps a strong graduate student should write, not just a final result. State assumptions, define symbols, check limiting cases or units when useful. If a problem statement is ambiguous, identify and flag ambiguities and choose a reasonable interpretation before proceeding. Prefer a clear blackboard-style derivation with short explanatory text between equations.

When appropriate, you can first try to work through answers to physics problems using your analytical reasoning ability, but you should use Python (and other computational tools like SymPy or logic tools like Lean) to check results or do calculations that are more easily done with them, such as long derivations, algebra, integrals, or plots. If you are speculating at points, make that clear, and make it clear when you are uncertain about a result or when there isn't major agreement among experts.

Prompting with normal language (1/2):

I just use normal language as if I was talking with a colleague who was an expert comptuational plasma physicist. As an example, below is my initial prompt. This was followed by dozens of prompts and responses as we iteratively worked through this problem and I asked ChatGPT to explain what it was finding, asked questions about its results, asked for variations with different assumptions, asked for plots to be made, etc.

```
Attached is a draft paper involving our group which develops the Gkeyll code.
As illustrated in Fig. C8, there appears to be a numerical instability in
Gkeyll for the case of constant B(z) = B_0 even in the simple limit of
Boltzmann electrons.  The COGENT code doesn't see this numerical instability.
Another option is that this is weak real instability of the equations used in
this limit, but that COGENT has more numerical damping than GKEYLL so it
doesn't see it.   (I might sometimes refer to this question as whether it is
a "numerical instability", introduced by the discretization, or whether it
is a "physical instability", a real instability of the mathematical equations
being solved in this limit.  I realize that there might be unphysical features
of the mathematical equations in a particular model.  If those unphysical
features cause an instability, that's also useful to know.)
```

Prompting with normal language (2/2):

```
Another option might be that COGENT is using a different boundary condition
along the field line.  Can you do a search for papers that describe the
numerical methods in COGENT to see if it treats the boundary condition
differently?  I think the boundary condition on the distribution function
is simple, $F(v,L) = 0$ for $v<0$ at the right boundary $z=L$.  Perhaps
there is a difference in the way the electric field is solved for?  I've
forgotten what Gkeyll is doing to turn a discontinuous $n_i(z)$ into a
continuous $\phi9z)$.  Could that be the source of a problem?  Can you
do a search for Gkeyll papers on how the Boltzmann response is handled,
perhaps in a paper by Mana Francisquez on Gkeyll simulations of mirrors.
```