Optimizing the performance of fusion reactors with transport in the loop

N. R. Mandell

Plasma Science and Fusion Center, MIT, Cambridge, USA

Massachusetts Institute of Technology

Oak Ridge Institute for Science and Education

PSFC MIT Plasma Science & Fusion Center

Fusion is hard

Fusion is hard

- Fusion is challenging because we need to achieve and maintain high pressures in the core of the reactor to reach self-sustaining “burning” conditions

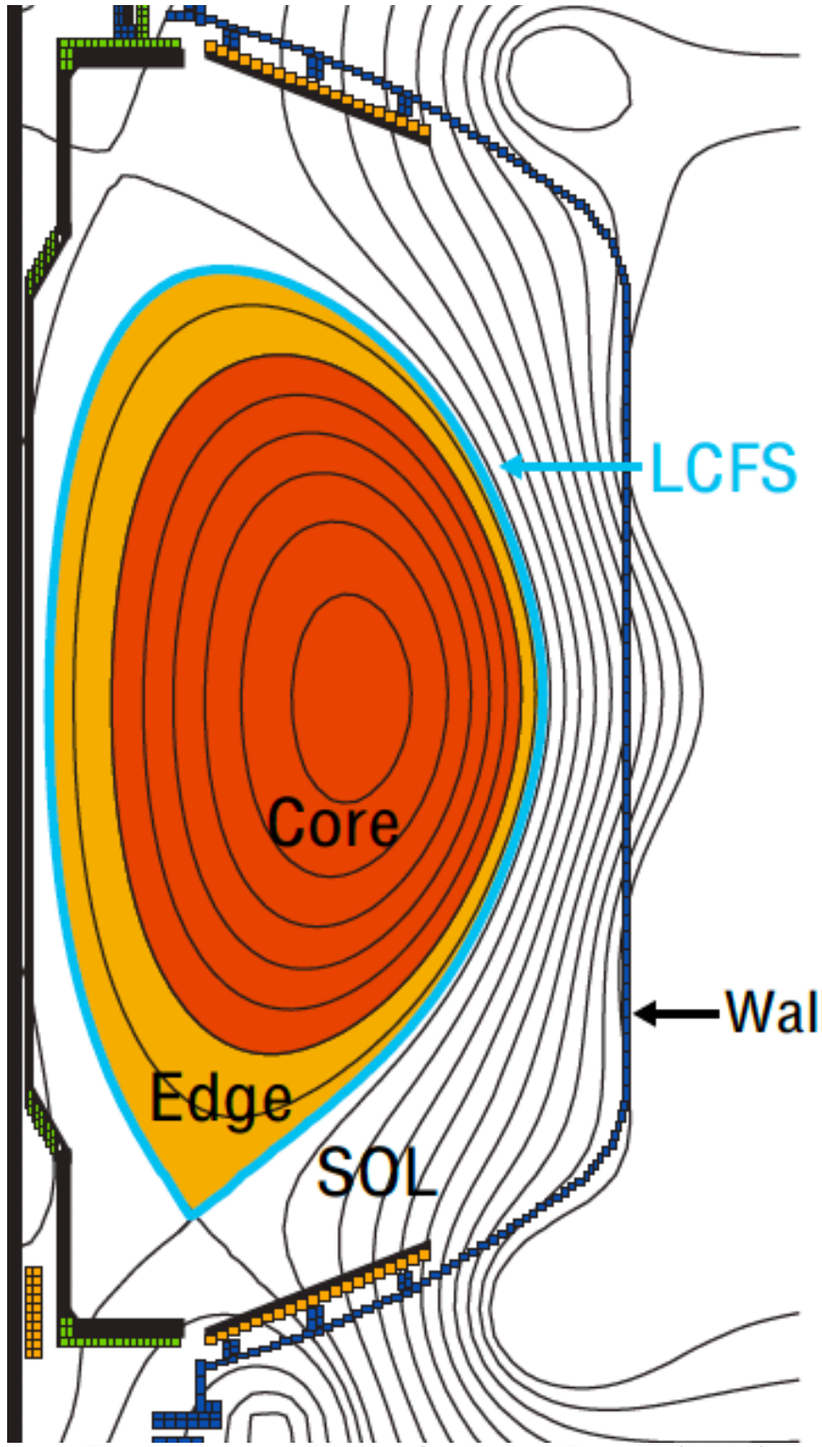


Figure adapted from Stoltzfus-Dueck (2009)

Fusion is hard

- Fusion is challenging because we need to achieve and maintain high pressures in the core of the reactor to reach self-sustaining “burning” conditions

- **Problem 1**: heat is lost too quickly from the core because of **turbulence**
  - Temperature gradients between hot core and cool walls are more than 1000x steeper than gradients handled by reentry tiles on spacecraft

- Nature doesn’t like temperature gradients that steep! Gradients drive instabilities that produce turbulence

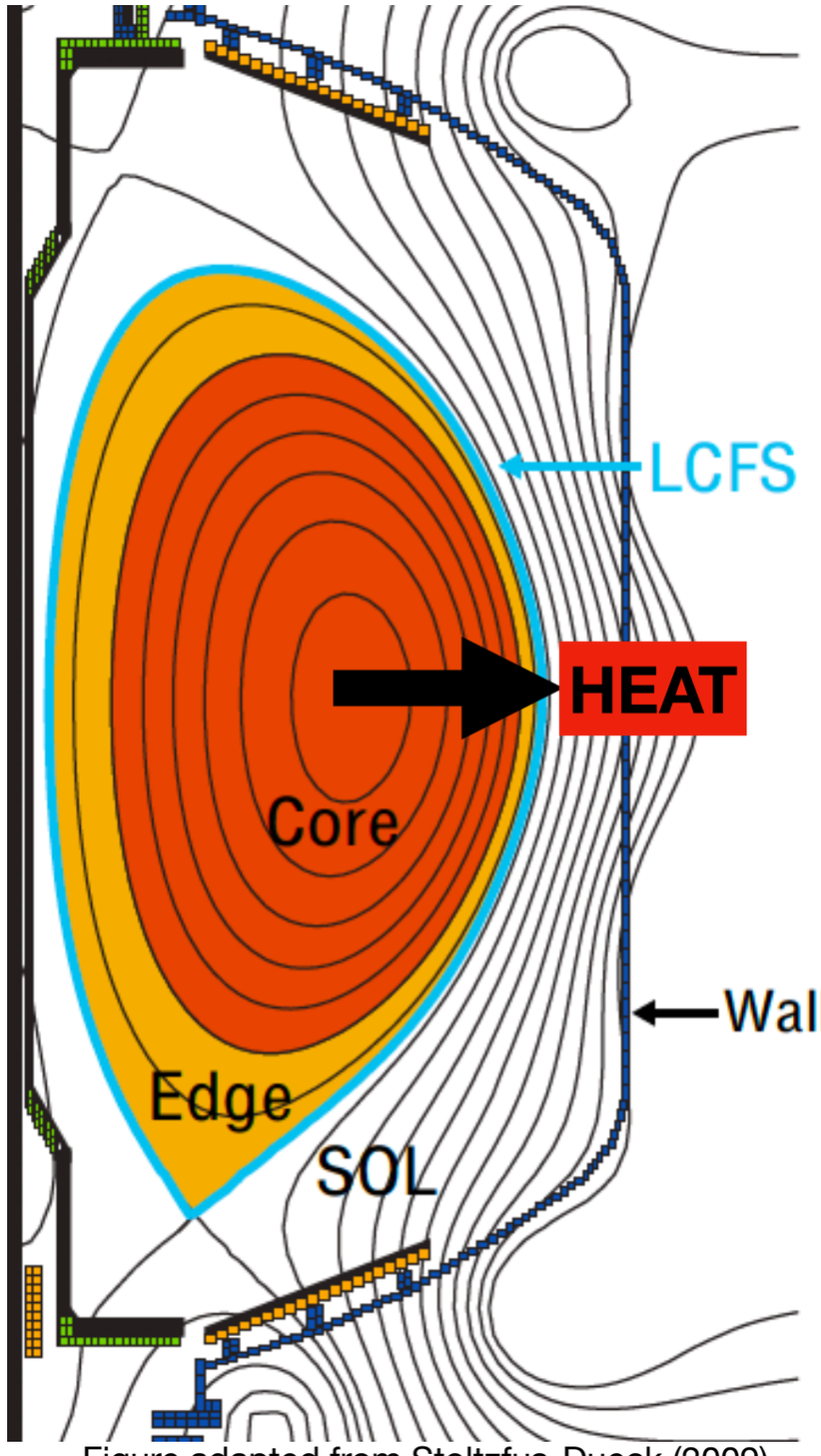


Figure adapted from Stoltzfus-Dueck (2009)

Fusion is hard

- Fusion is challenging because we need to achieve and maintain high pressures in the core of the reactor to reach self-sustaining “burning” conditions

- **Problem 1**: heat is lost too quickly from the core because of **turbulence**
  - Temperature gradients between hot core and cool walls are more than 1000x steeper than gradients handled by reentry tiles on spacecraft
  - Nature doesn’t like temperature gradients that steep! Gradients drive instabilities that produce turbulence

- Heat is exhausted from the core and handled in the boundary region, where the field lines are “open” and intersect the walls

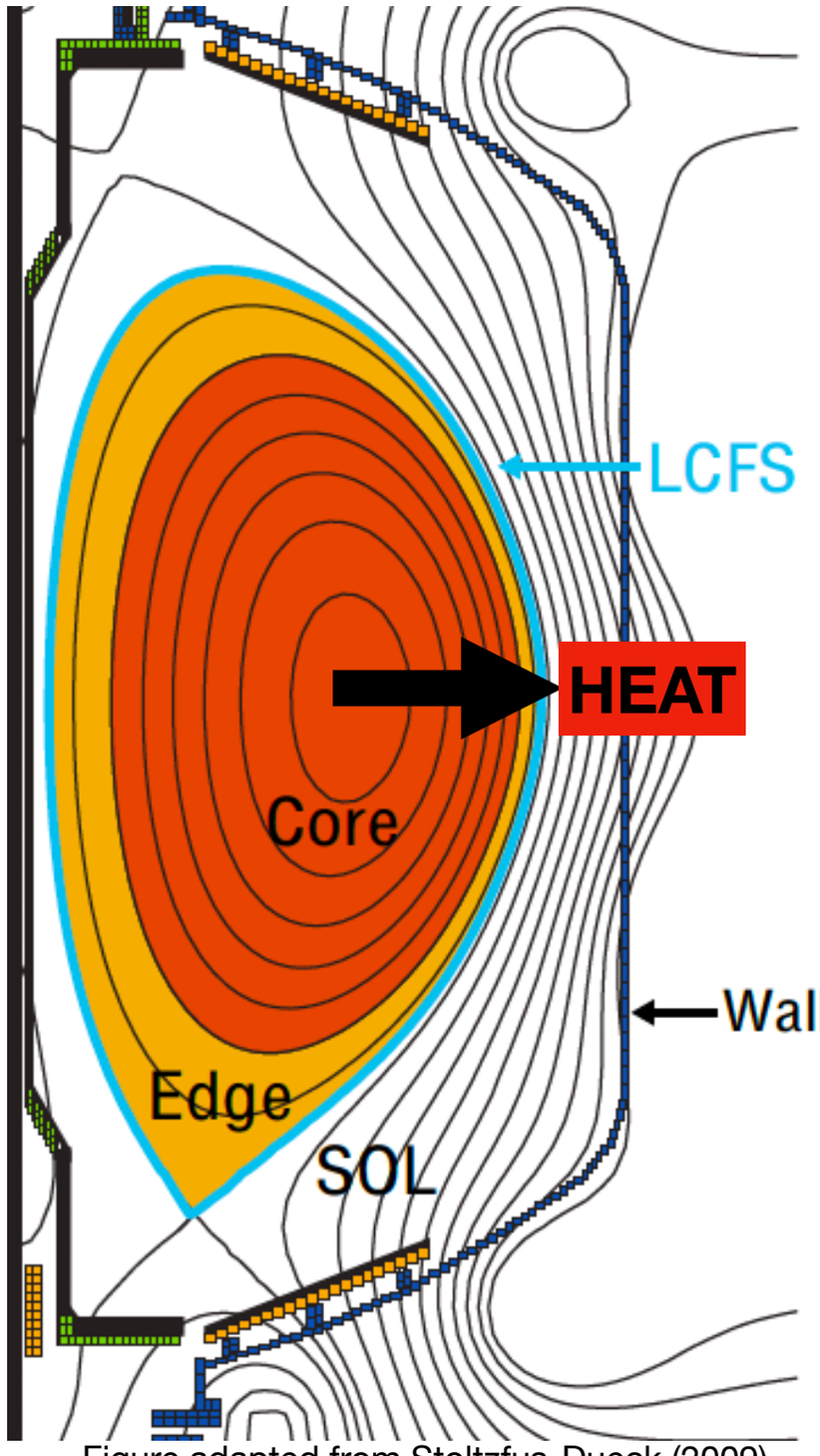


Figure adapted from Stoltzfus-Dueck (2009)

Fusion is hard

- Fusion is challenging because we need to achieve and maintain high pressures in the core of the reactor to reach self-sustaining “burning” conditions

- **Problem 1**: heat is lost too quickly from the core because of **turbulence**
  - Temperature gradients between hot core and cool walls are more than 1000x steeper than gradients handled by reentry tiles on spacecraft
  - Nature doesn’t like temperature gradients that steep! Gradients drive instabilities that produce turbulence

- Heat is exhausted from the core and handled in the boundary region, where the field lines are “open” and intersect the walls

- **Problem 2**: if the boundary plasma is too hot, the heat **exhaust** will be dangerous to the device walls

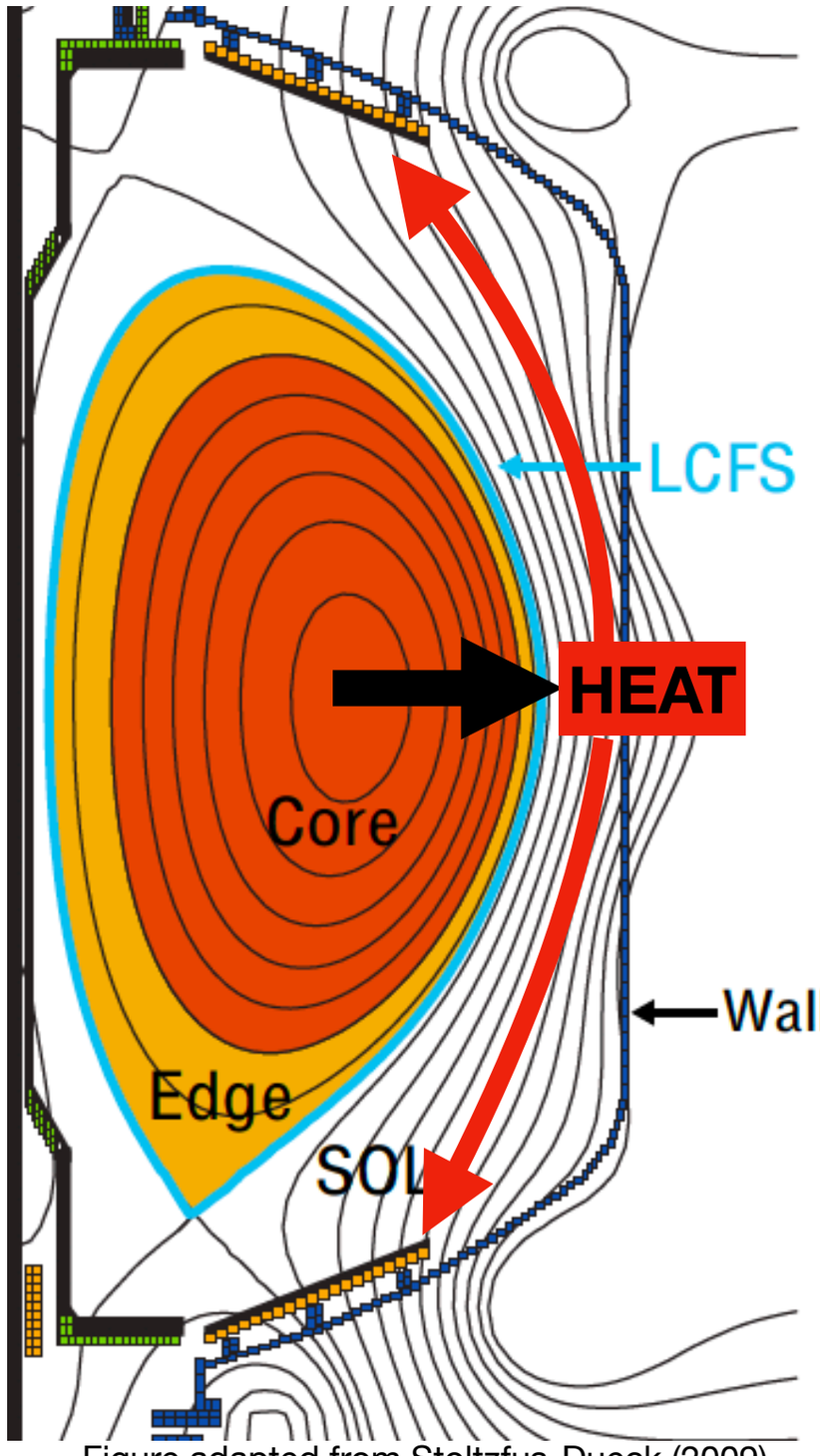
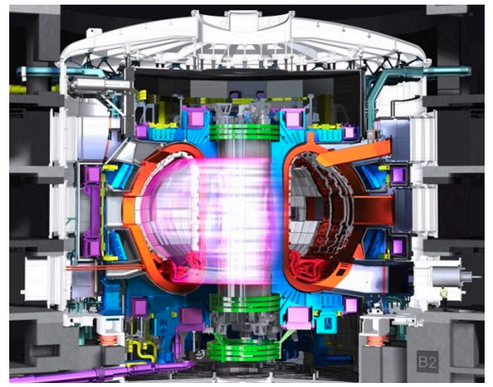


Figure adapted from Stoltzfus-Dueck (2009)

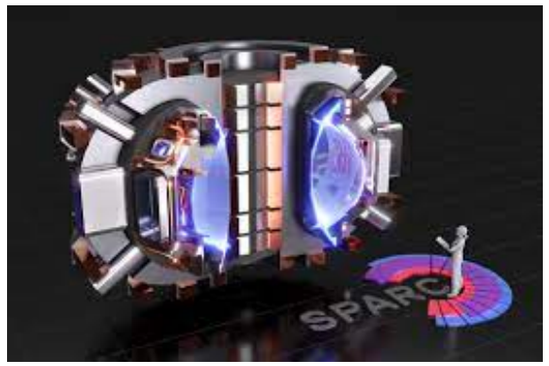
Fusion is hard

"A sustained burning plasma at high power density is required simultaneously with a solution to the power exhaust challenge: mitigating the extreme heat fluxes to materials surrounding the plasma." - T. Carter et al, FESAC Long Range Planning report

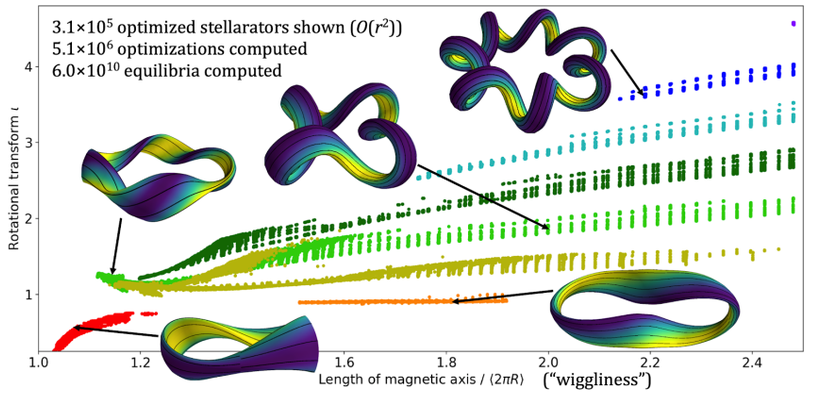
- Need a **whole-device** solution to these coupled core and boundary problems
  - We need the core to be 10x hotter than the surface of the sun without melting the walls
- So what to do?
  - Build bigger reactors (ITER, $$$): takes longer for heat to diffusively leak out
  - Use stronger magnetic fields (SPARC/ARC): diffusion length ~ $1/B^2$
  - **Find optimal designs** that simultaneously reduce turbulent heat losses and have favorable heat exhaust handling
    - Large design space, esp. for stellarators



ITER



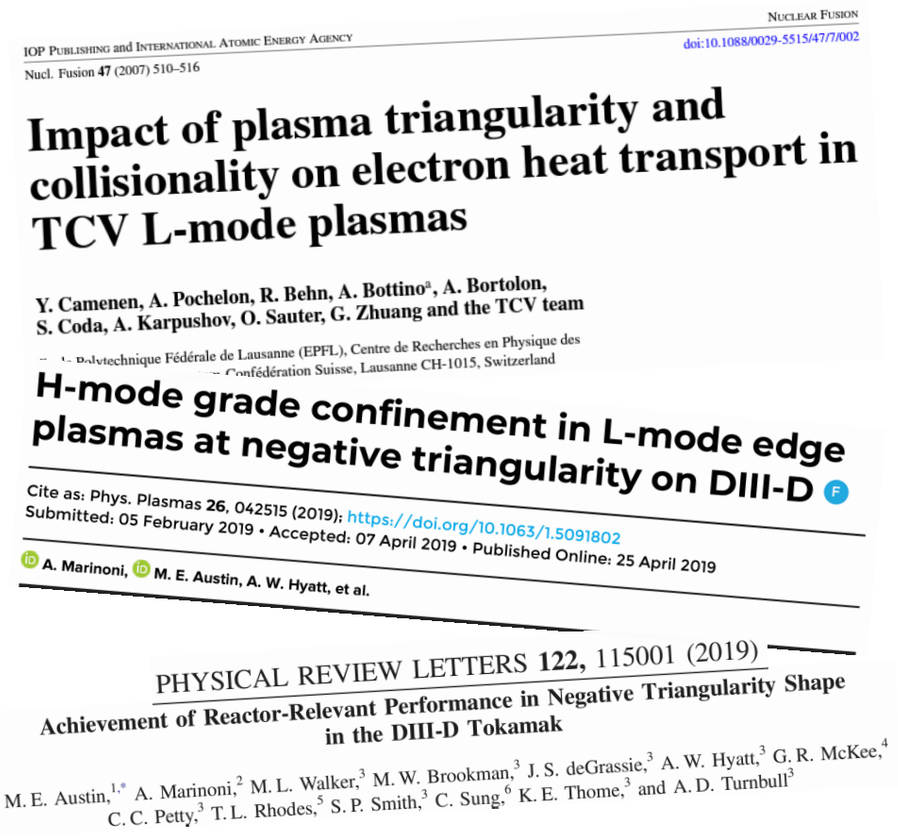
SPARC



Landreman

Improving the performance of fusion reactors.. with experiments

- In 60+ years of fusion research, we have learned a lot about how to optimize our fusion experiments
- An interesting recent advance is the success of negative triangularity tokamak configurations
  - Experiments on TCV and DIII-D have shown that negative triangularity improves core confinement while also eliminating dangerous boundary instabilities (ELMs), producing a potentially viable core-boundary solution
- This is promising, but we can't always rely on being able to modify current experiments or build new ones to identify promising new regimes


IOP Publishing and International Atomic Energy Agency
Nucl. Fusion 47 (2007) 510–516
Nuclear Fusion
doi:10.1088/0029-5515/47/7/002

Impact of plasma triangularity and collisionality on electron heat transport in TCV L-mode plasmas

Y. Camenen, A. Pochelon, R. Behn, A. Bottino[a], A. Bortolon, S. Coda, A. Karpushov, O. Sauter, G. Zhuang and the TCV team

Polytechnique Fédérale de Lausanne (EPFL), Centre de Recherches en Physique des Confédération Suisse, Lausanne CH-1015, Switzerland

H-mode grade confinement in L-mode edge plasmas at negative triangularity on DIII-D

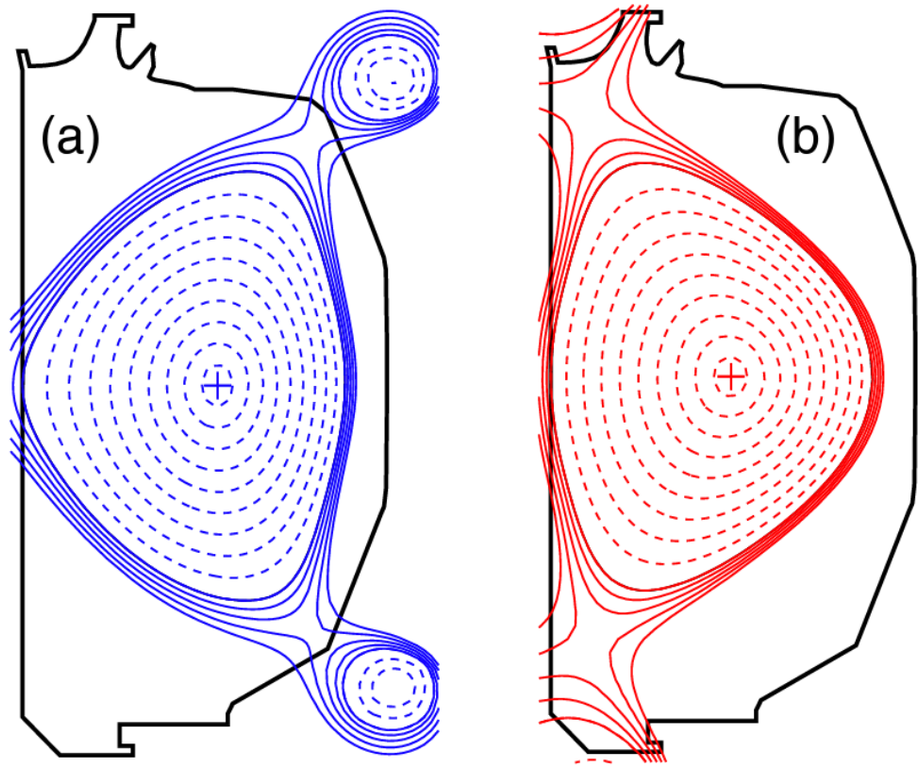
Cite as: Phys. Plasmas 26, 042515 (2019); https://doi.org/10.1063/1.5091802
Submitted: 05 February 2019 • Accepted: 07 April 2019 • Published Online: 25 April 2019

A. Marinoni, M. E. Austin, A. W. Hyatt, et al.

PHYSICAL REVIEW LETTERS 122, 115001 (2019)

Achievement of Reactor-Relevant Performance in Negative Triangularity Shape in the DIII-D Tokamak

M. E. Austin,[1,*] A. Marinoni,[2] M. L. Walker,[3] M. W. Brookman,[3] J. S. deGrassie,[3] A. W. Hyatt,[3] G. R. McKee,[4] C. C. Petty,[3] T. L. Rhodes,[5] S. P. Smith,[3] C. Sung,[6] K. E. Thome,[3] and A. D. Turnbull[3]



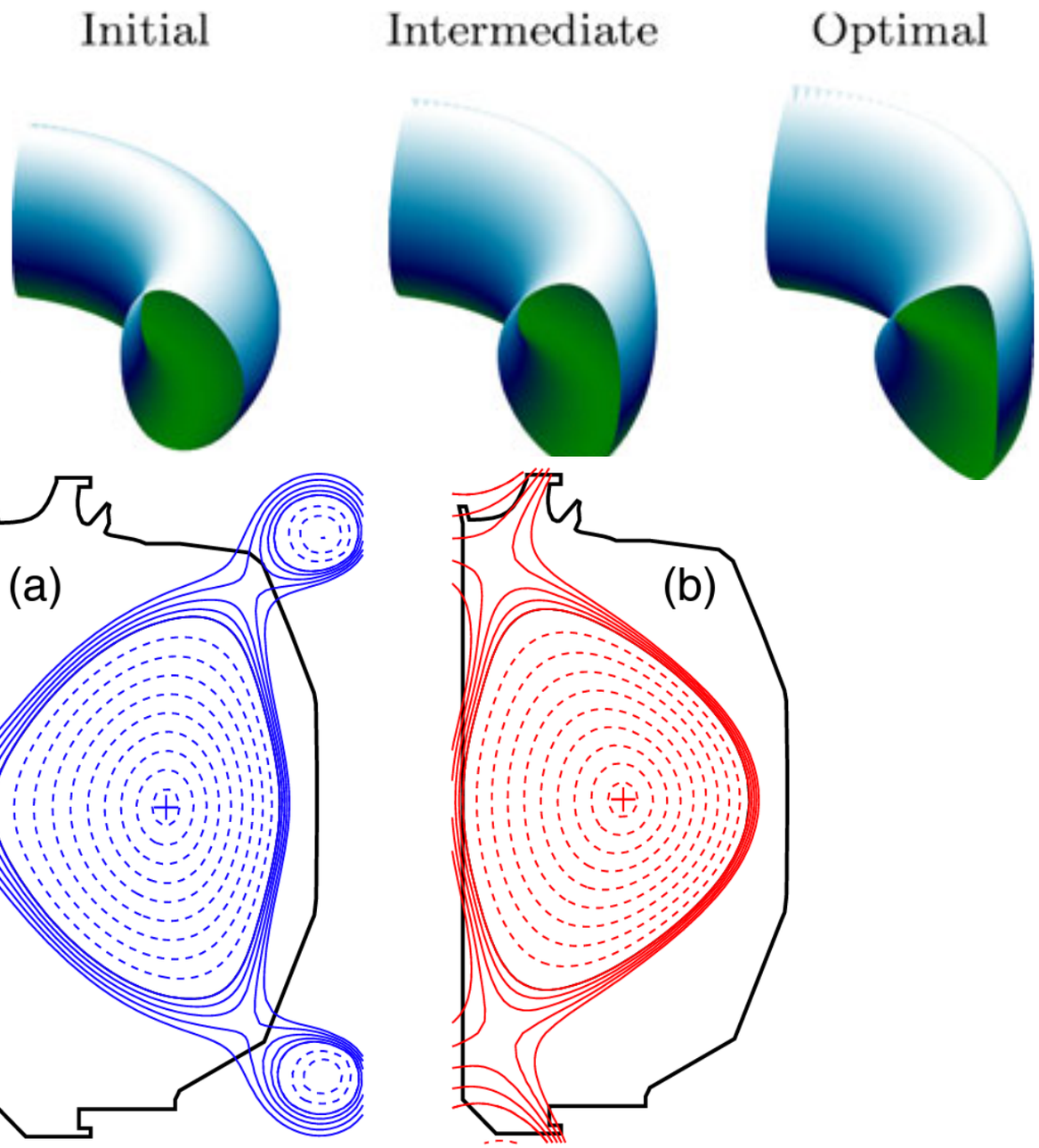
Improving the performance of fusion reactors.. with transport modeling

- Instead, we should think about how transport **modeling and optimization** could have foreseen the advantages of the negative triangularity regime
  - Highcock, Mandell et al (2018) demonstrated the ability to perform tokamak optimization with self-consistent MHD equilibria, gyro-*fluid* turbulence, and multi-scale transport solutions, all running inside an optimization loop
  - Optimization algorithm found negative triangularity to be optimal (more on this later)

J. Plasma Phys. (2018), *vol.* 84, 905840208 © Cambridge University Press 2018
doi:10.1017/S002237781800034X

Optimisation of confinement in a fusion reactor using a nonlinear turbulence model

E. G. Highcock[1,2,3,†], N. R. Mandell[4], M. Barnes[2] and W. Dorland[5]



IOP Publishing and International Atomic Energy Agency
Nucl. Fusion 47 (2007) 510–516
Nuclear Fusion
doi:10.1088/0029-5515/47/7/002

Impact of plasma triangularity and collisionality on electron heat transport in TCV L-mode plasmas

Y. Camenen, A. Pochelon, R. Behn, A. Bottino[a], A. Bortolon,
S. Coda, A. Karpushov, O. Sauter, G. Zhuang and the TCV team
...lytechnique Fédérale de Lausanne (EPFL), Centre de Recherches en Physique des
... Confédération Suisse, Lausanne CH-1015, Switzerland

H-mode grade confinement in L-mode edge plasmas at negative triangularity on DIII-D

Cite as: Phys. Plasmas 26, 042515 (2019); https://doi.org/10.1063/1.5091802
Submitted: 05 February 2019 • Accepted: 07 April 2019 • Published Online: 25 April 2019

A. Marinoni, M. E. Austin, A. W. Hyatt, et al.

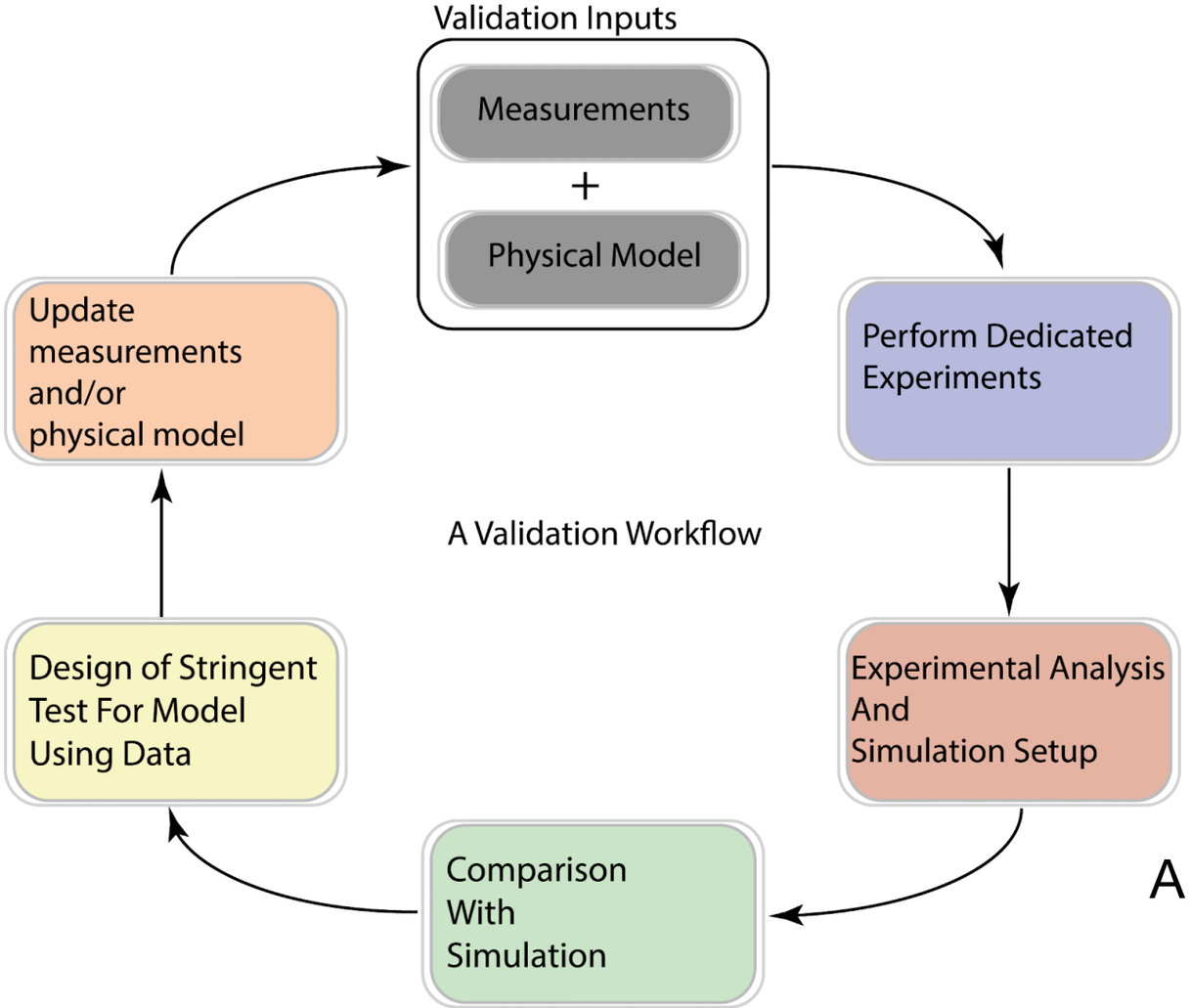
PHYSICAL REVIEW LETTERS 122, 115001 (2019)

Achievement of Reactor-Relevant Performance in Negative Triangularity Shape in the DIII-D Tokamak

M. E. Austin,[1,*] A. Marinoni,[2] M. L. Walker,[3] M. W. Brookman,[3] J. S. deGrassie,[3] A. W. Hyatt,[3] G. R. McKee,[4]
C. C. Petty,[3] T. L. Rhodes,[5] S. P. Smith,[3] C. Sung,[6] K. E. Thome,[3] and A. D. Turnbull[3]

Validation vs design

- Gyrokinetic turbulence models have been carefully **validated** against experimental measurements
  - This work is incredibly valuable, because we now know we can trust these models in many parameter regimes
- Validation exercises require precise, ultra-high-fidelity calculations and carefully-designed experiments
  - A single ultra-high-fidelity turbulence simulation could be 10M core-hours



A E White et al, 2019

Validation vs design

- Gyrokinetic turbulence models have been carefully **validated** against experimental measurements
  - This work is incredibly valuable, because we now know we can trust these models in many parameter regimes
- Validation exercises require precise, ultra-high-fidelity calculations and carefully-designed experiments
  - A single ultra-high-fidelity turbulence simulation could be 10M core-hours

Validation vs design

- Gyrokinetic turbulence models have been carefully **validated** against experimental measurements
  - This work is incredibly valuable, because we now know we can trust these models in many parameter regimes
- Validation exercises require precise, ultra-high-fidelity calculations and carefully-designed experiments
  - A single ultra-high-fidelity turbulence simulation could be 10M core-hours

- As we shift focus to **design/optimization**, validation-scale ultra-high-fidelity calculations are not practical
  - Can’t put a 10M CPU-hour simulation inside an optimization loop!
- Instead, to reduce cost we should think about trying to capture the physics that we know matters most, especially at reactor scales
  - Focus on **time to solution**
  - Example: In the core, reactor-scale turbulence will likely be dominated by “vanilla” ion-temperature-gradient turbulence and critical gradient effects that we have studied/ simulated for 20+ years

Whole-device modeling, optimization, and design

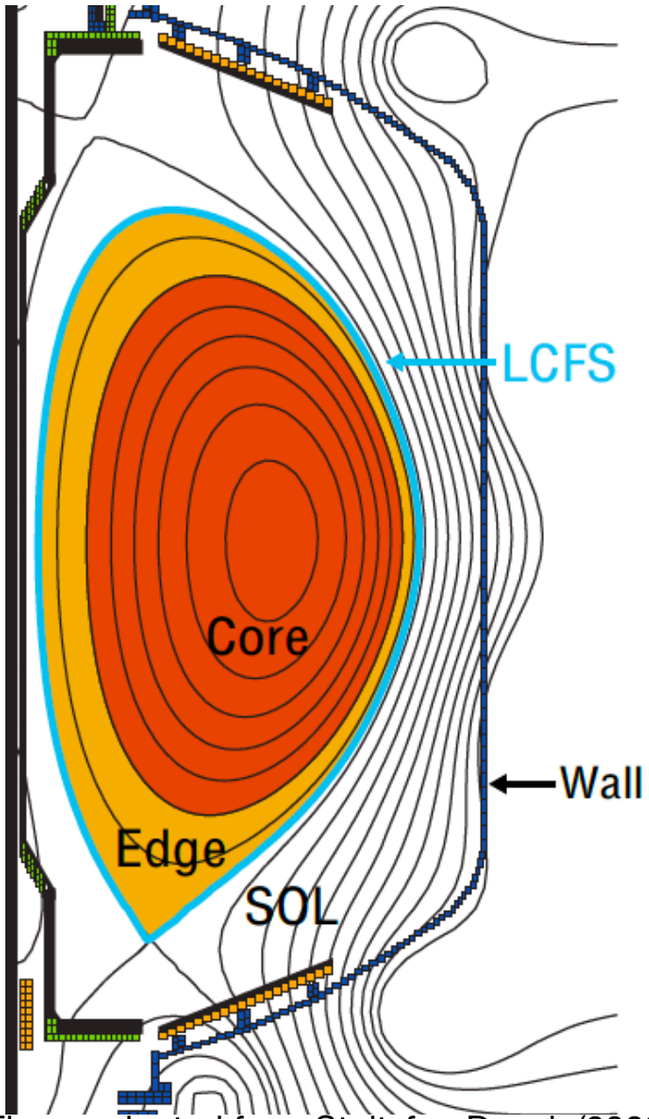


Figure adapted from Stoltzfus-Dueck (2009)

Whole-device modeling, optimization, and design

- **Whole-device turbulent transport modeling and optimization** are areas where **high-performance computing** can make a large impact on the success of fusion by enabling the **design** of more efficient fusion reactors. For a given reactor design, need to be able to model/predict:

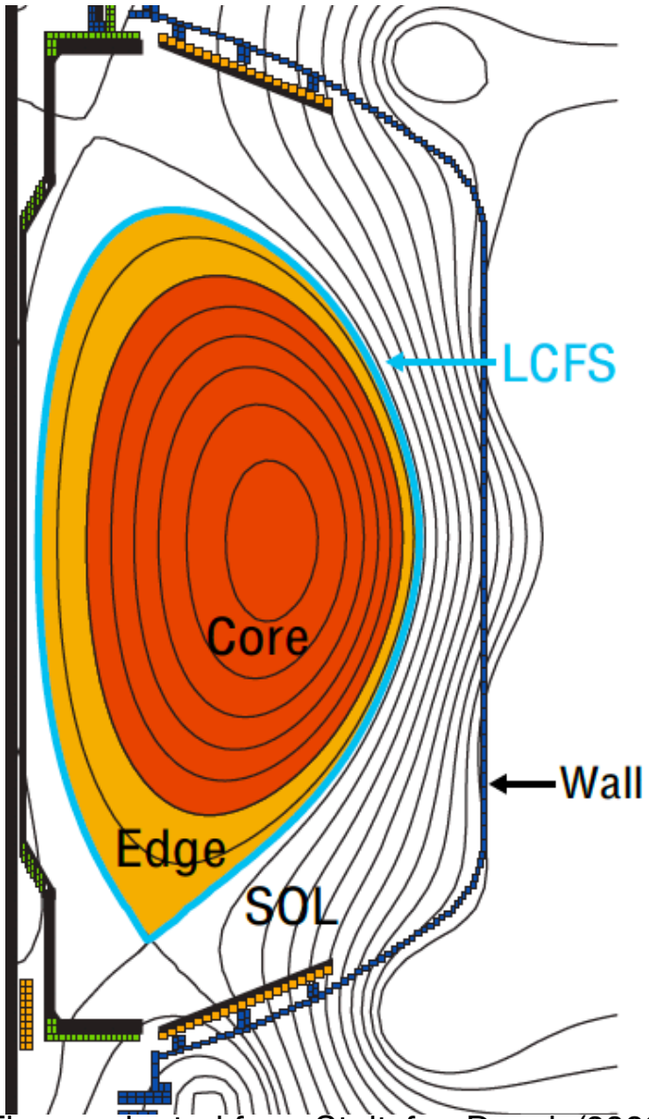


Figure adapted from Stoltzfus-Dueck (2009)

Whole-device modeling, optimization, and design

- **Whole-device turbulent transport modeling and optimization** are areas where **high-performance computing** can make a large impact on the success of fusion by enabling the **design** of more efficient fusion reactors. For a given reactor design, need to be able to model/predict:
  - Fusion power, which is determined by macroscopic profiles of density and temperature in the core

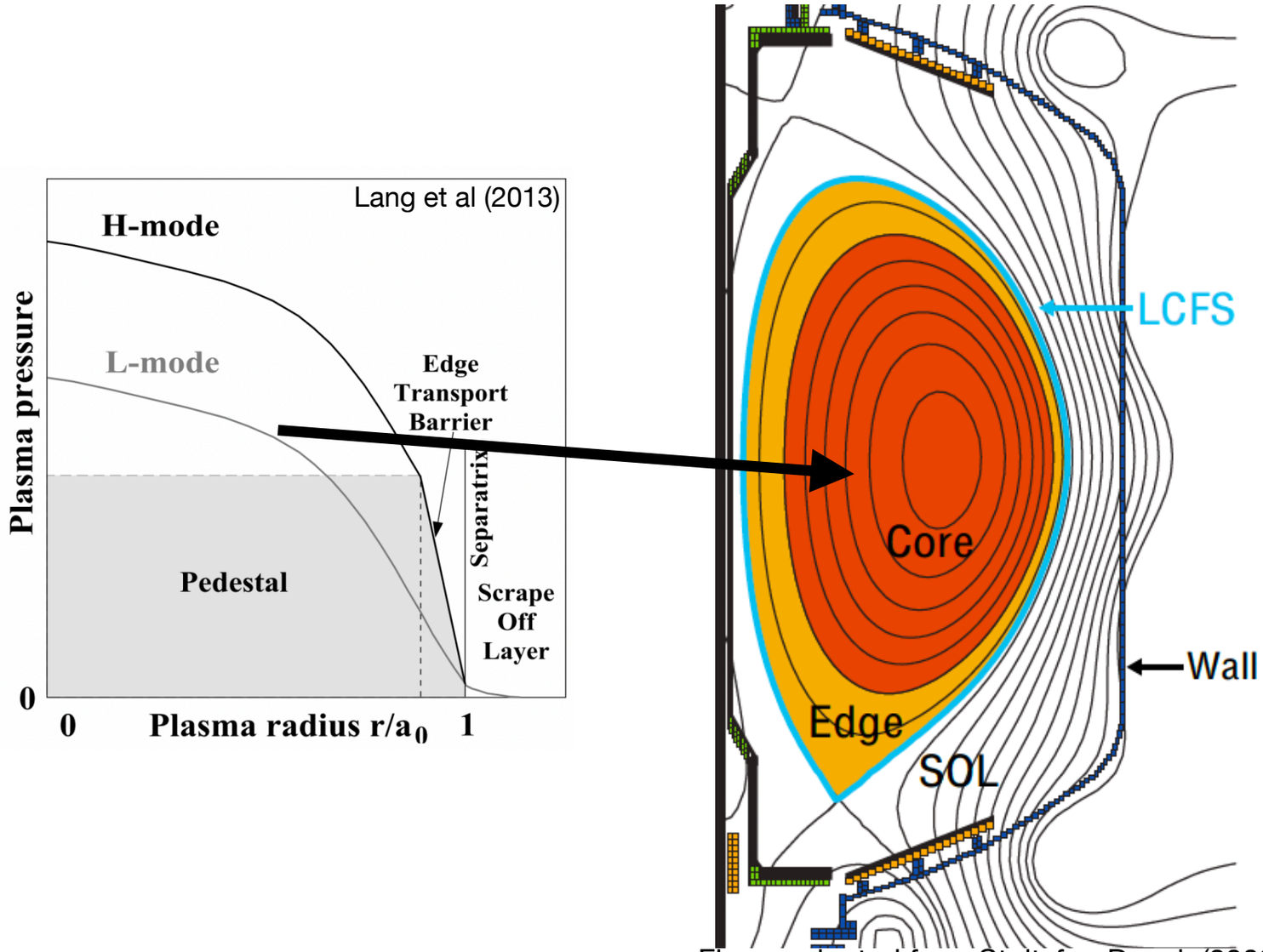


Figure adapted from Stoltzfus-Dueck (2009)

Whole-device modeling, optimization, and design

- **Whole-device turbulent transport modeling and optimization** are areas where **high-performance computing** can make a large impact on the success of fusion by enabling the **design** of more efficient fusion reactors. For a given reactor design, need to be able to model/predict:
  - Fusion power, which is determined by macroscopic profiles of density and temperature in the core
  - Boundary properties (pedestal height, heat load to walls, etc)

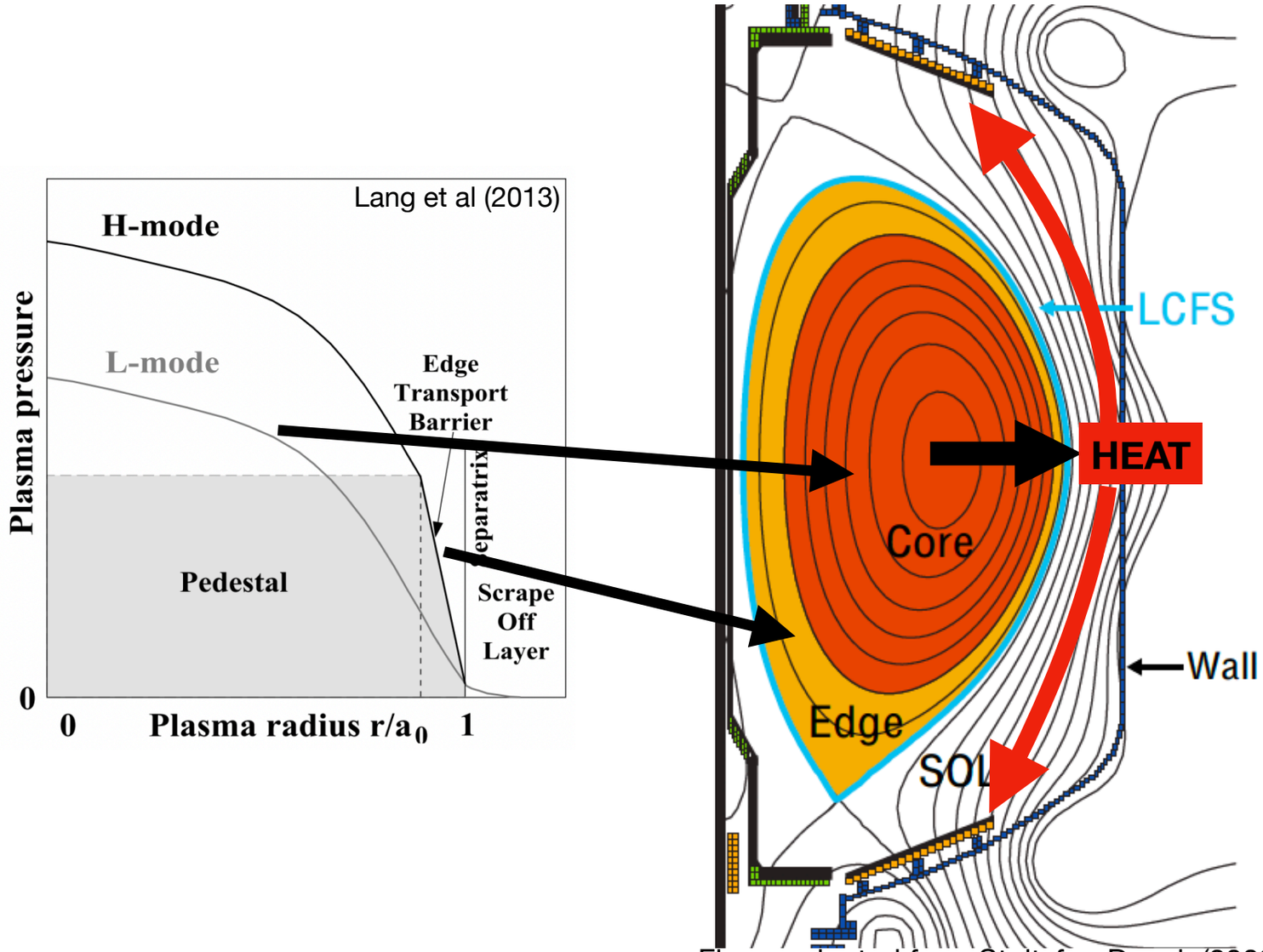


Figure adapted from Stoltzfus-Dueck (2009)

Whole-device modeling, optimization, and design

- **Whole-device turbulent transport modeling and optimization** are areas where **high-performance computing** can make a large impact on the success of fusion by enabling the **design** of more efficient fusion reactors. For a given reactor design, need to be able to model/predict:
  - Fusion power, which is determined by macroscopic profiles of density and temperature in the core
    - **Part 1:** Core turbulence modeling with **GX**, coupled to **Trinity** multi-scale transport solver
  - Boundary properties (pedestal height, heat load to walls, etc)

+

Trinity

(Part 1)

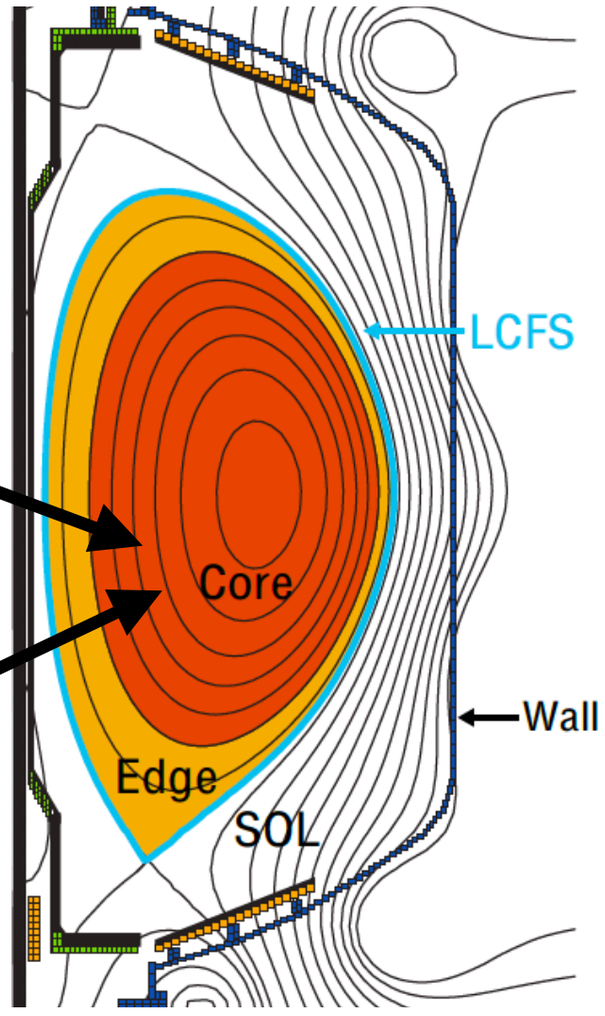


Figure adapted from Stoltzfus-Dueck (2009)

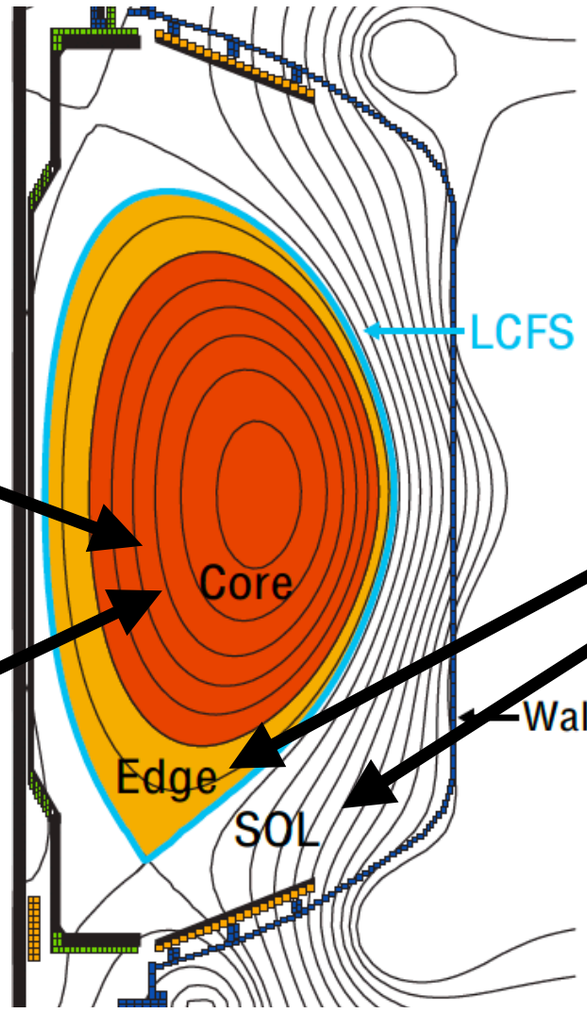
Whole-device modeling, optimization, and design

- **Whole-device turbulent transport modeling and optimization** are areas where **high-performance computing** can make a large impact on the success of fusion by enabling the **design** of more efficient fusion reactors. For a given reactor design, need to be able to model/predict:
  - Fusion power, which is determined by macroscopic profiles of density and temperature in the core
    - **Part 1:** Core turbulence modeling with **GX**, coupled to **Trinity** multi-scale transport solver
  - Boundary properties (pedestal height, heat load to walls, etc)
    - **Part 2:** Boundary turbulence modeling with **Gkeyll**

+

Trinity

(Part 1)



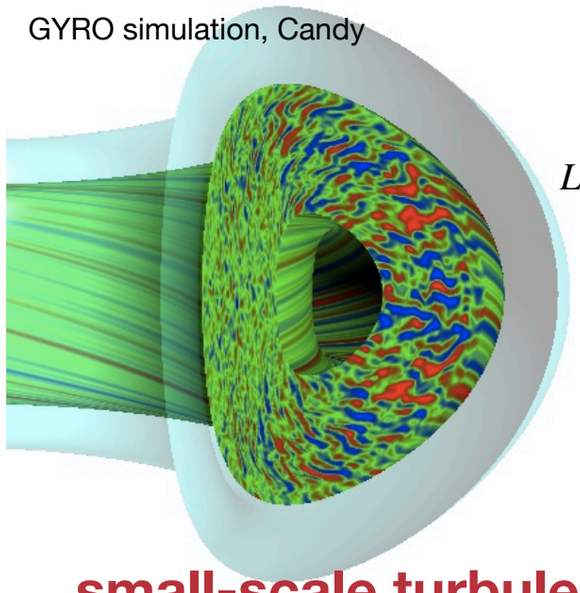
Gkeyll

(Part 2)

Figure adapted from Stoltzfus-Dueck (2009)

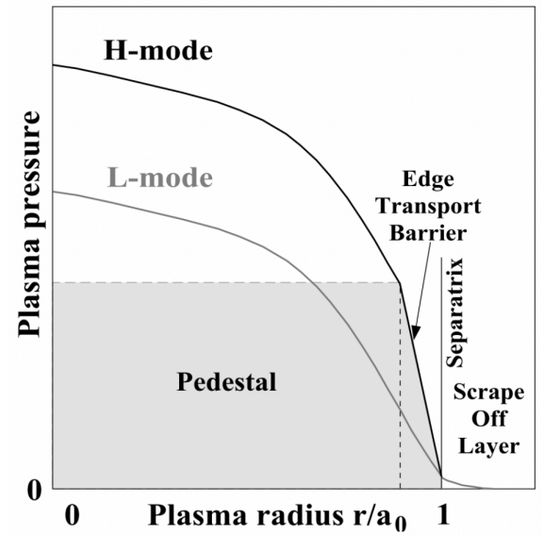
Fusion is hard; *modeling* a fusion plasma is hard, too

- A fusion reactor has an **enormous range of scales**. In the core, we have **turbulence at micro-scales** and background **profile evolution (transport) at macro-scales**, separated by several orders of magnitude



$L_{\text{turb}} \sim \rho_i \sim 0.1 - 0.5$ mm $\quad \sim 10^3 - 10^4 \quad$ $L_{\text{transp}} \sim a \gtrsim 1$ m

$\tau_{\text{turb}} \sim a/v_{\text{ti}} \sim 10\ \mu$s $\quad \sim 10^5 \quad$ $\tau_{\text{transp}} \sim \tau_E \sim 1$ s



small-scale turbulence **slowly-varying background profiles**

- Extreme temperatures require a **kinetic** description (instead of fluid), which means solving a PDE in 5D (3x+2v, after averaging over fast gyromotion to get **gyrokinetics**)
- Modeling the full range of scales with a brute-force 5D+time discretization would then require

$$N \sim \left(\frac{L_{\text{turb}}}{L_{\text{transp}}}\right)^3 \left(N_v \sim 10\right)^2 \sim 10^{14} \text{ phase-space grid points in 5D}$$

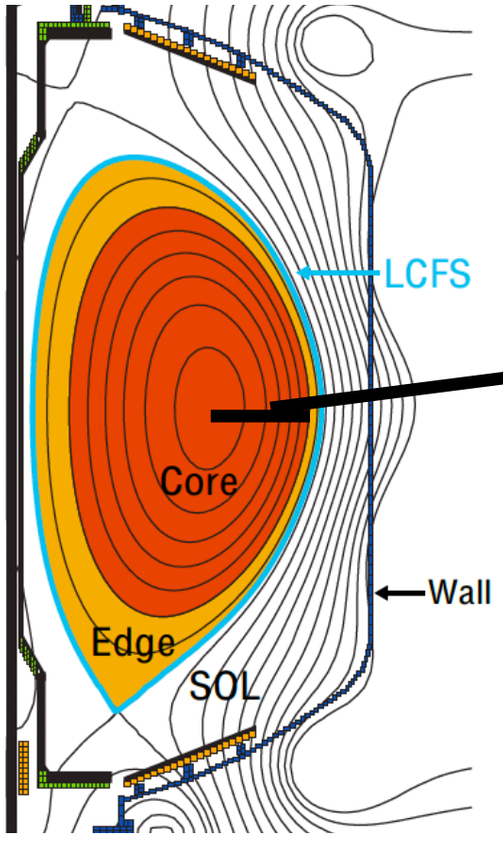
$$\text{NFLOP} \sim \mathcal{O}(N^2)\left(\frac{\tau_{\text{turb}}}{\tau_{\text{transp}}}\right) \sim 10^{33}, \text{ which would require } 10^{15} \text{ s on an exascale computer}$$

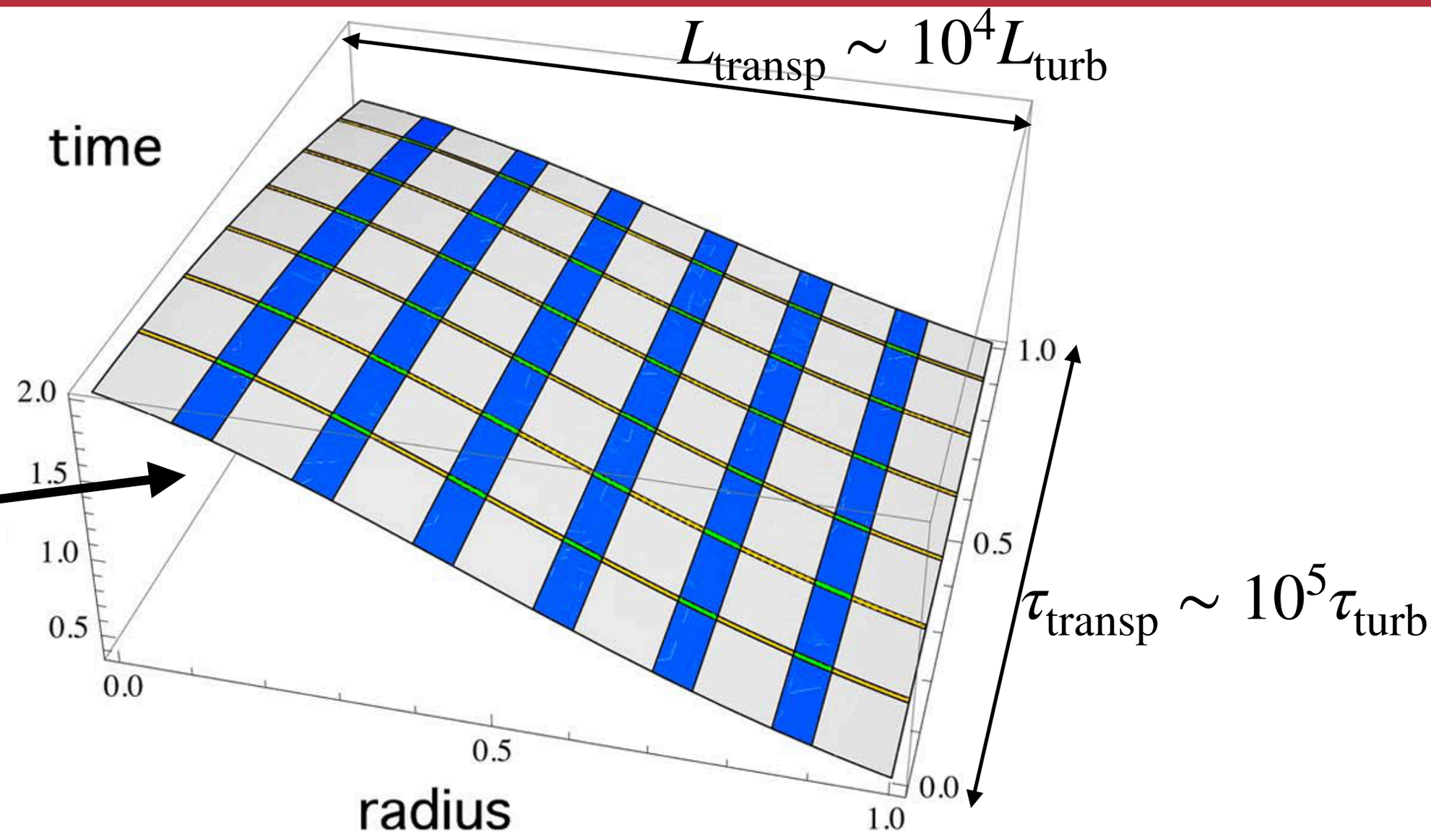
- These problems require **intense computational resources**, but also **clever multi-scale numerical algorithms and theory** to enable tractable calculations

Savings from using a multi-scale approach

Trinity

Barnes et al, 2010

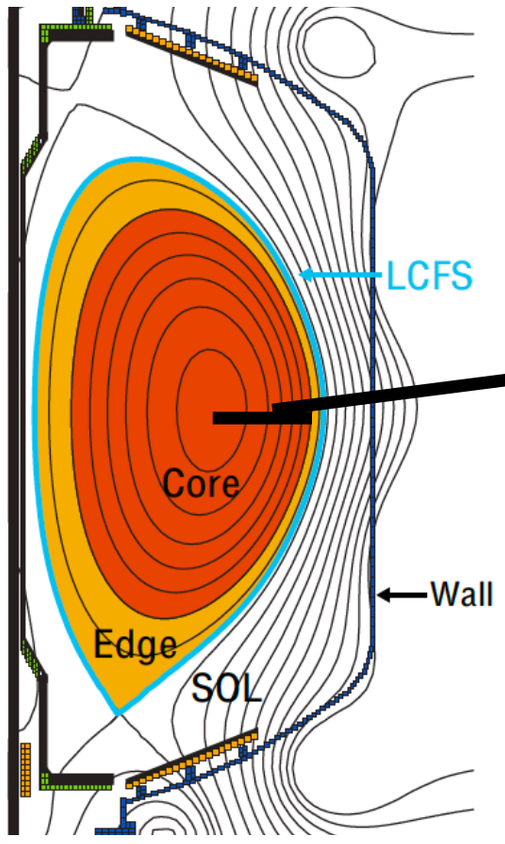


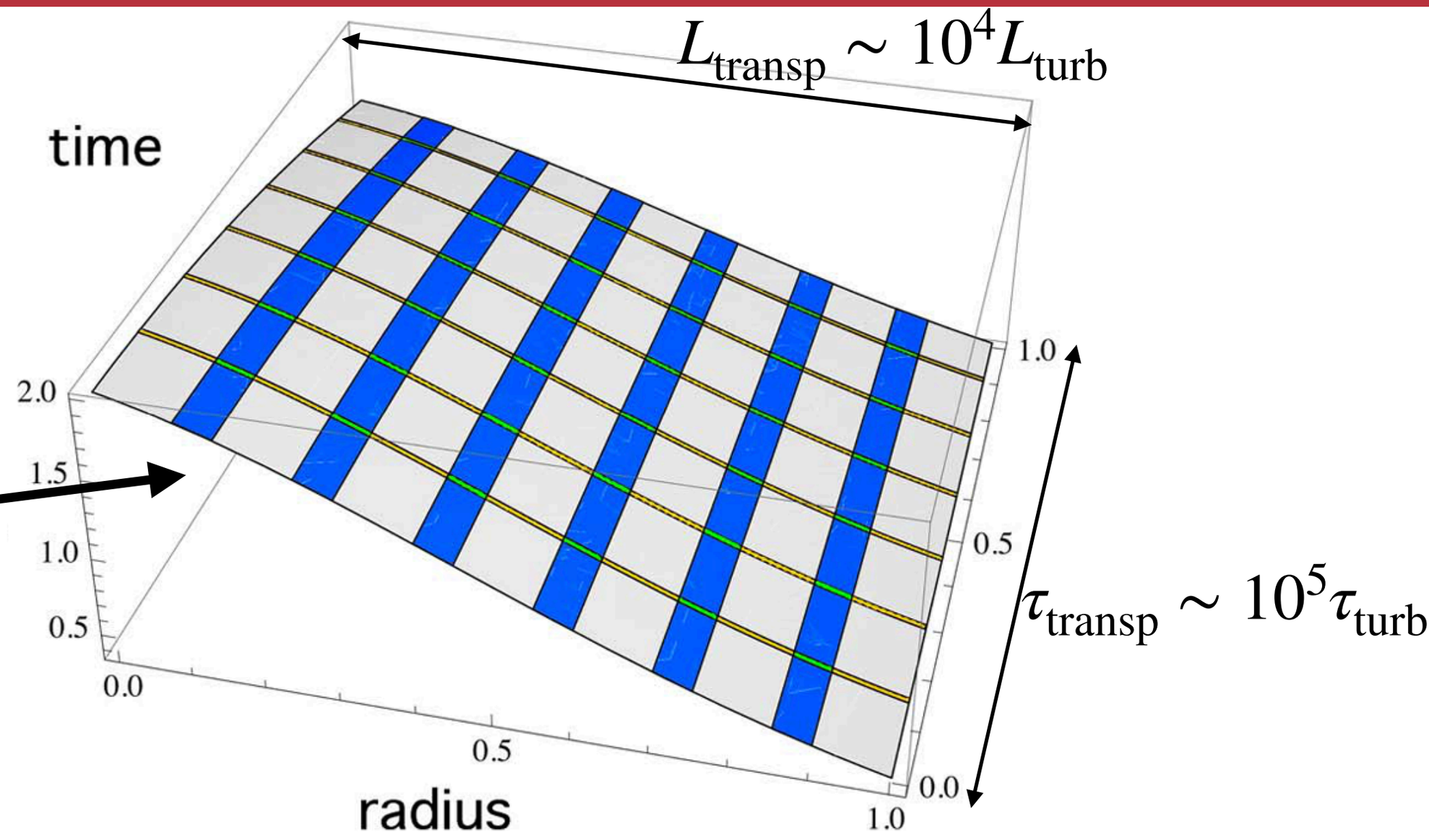


Savings from using a multi-scale approach

Trinity

Barnes et al, 2010



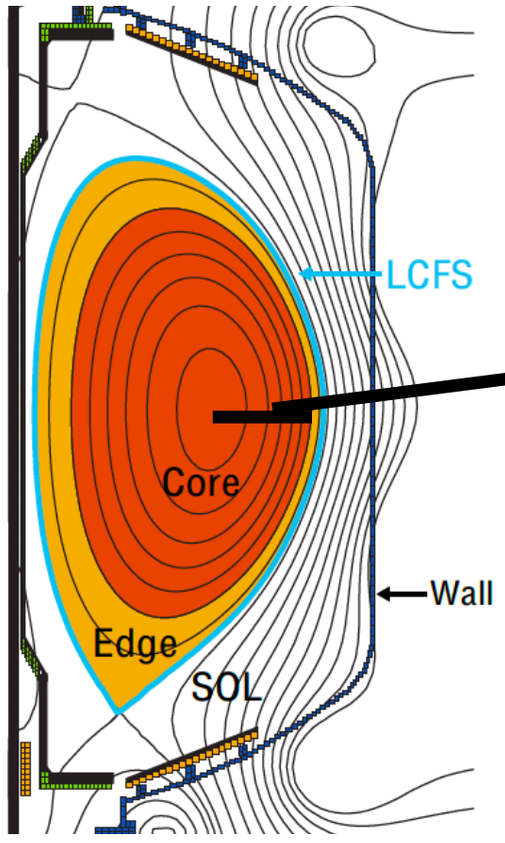


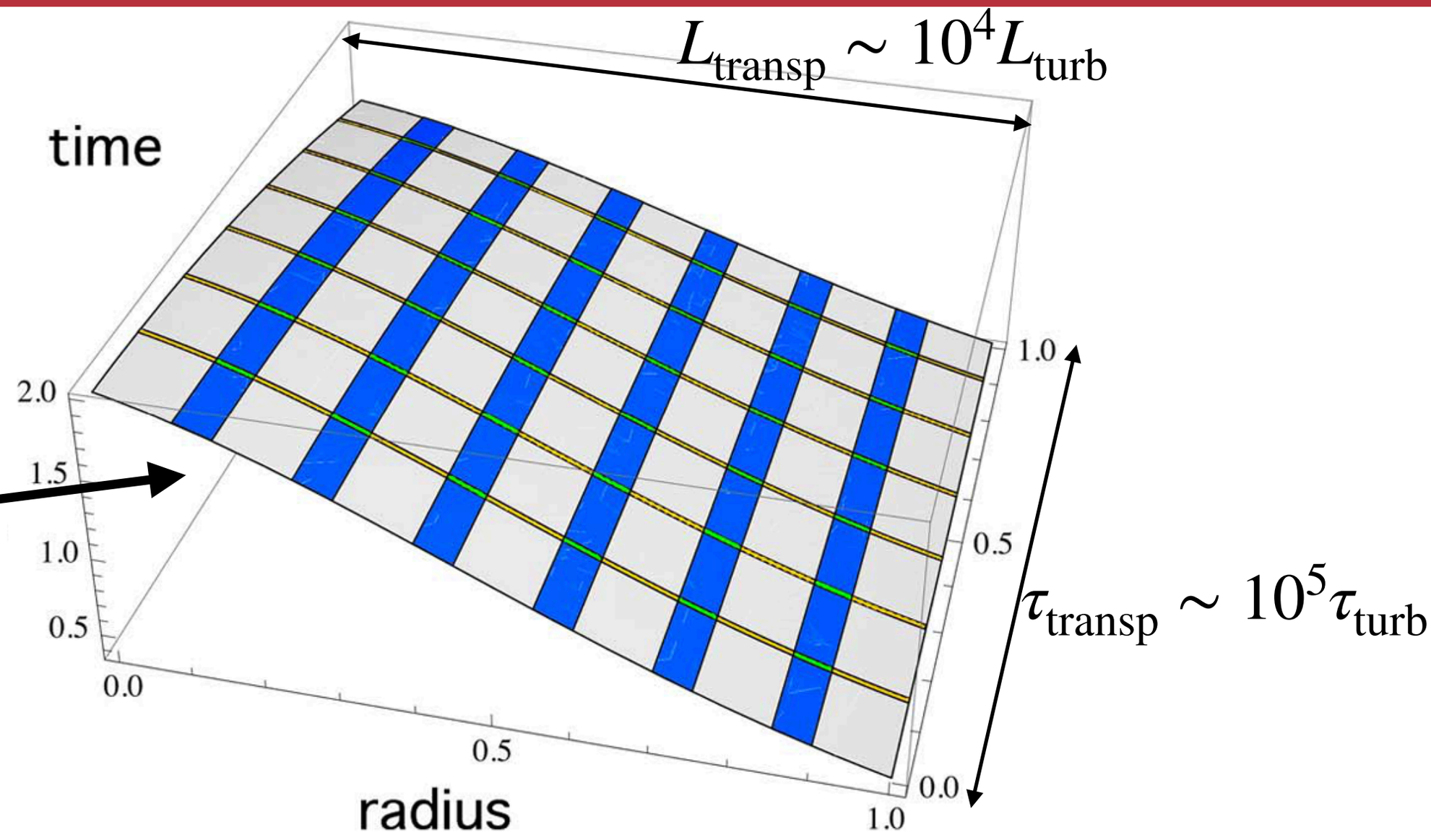
- Brute-force (global) approach: fine space-time mesh for entire domain, with $\Delta x \sim L_{\text{turb}}$ and $\Delta t \sim \tau_{\text{turb}}$, scales very poorly at reactor scale ($R/\rho_i \gg 1$)

Savings from using a multi-scale approach

Trinity

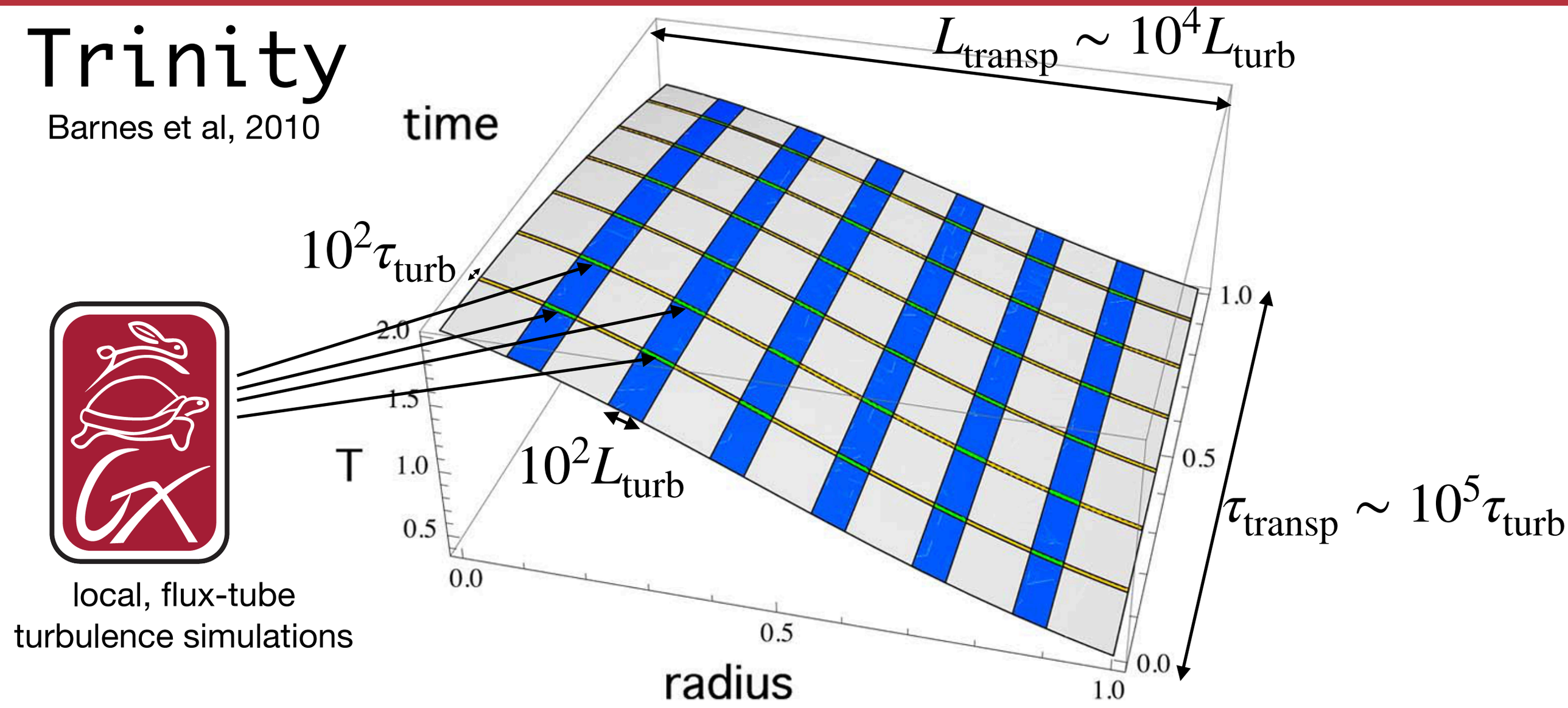
Barnes et al, 2010





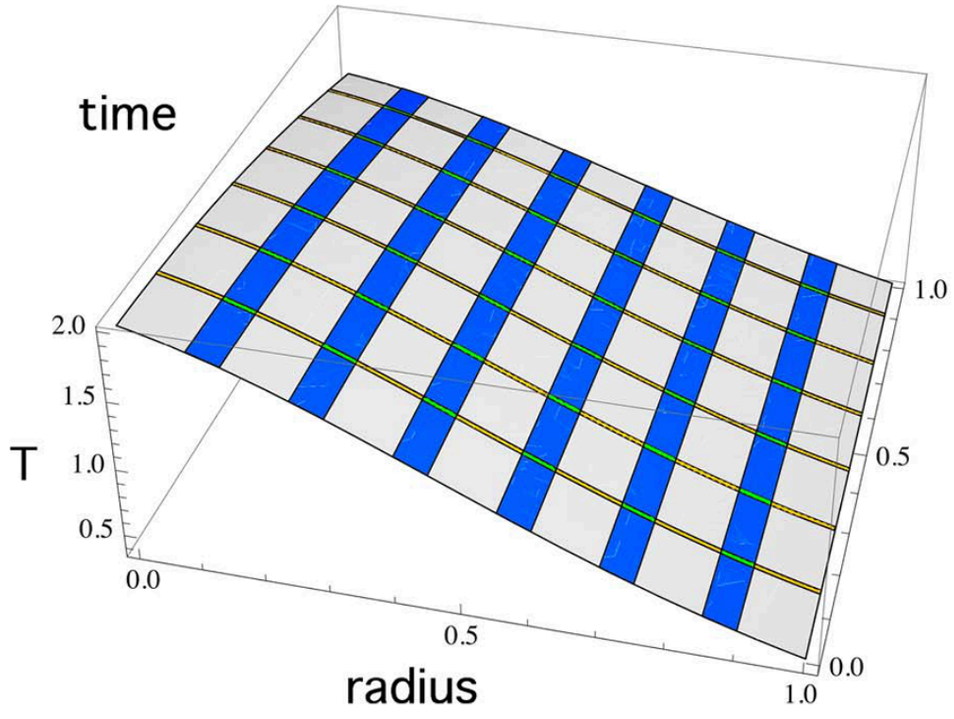
- Brute-force (global) approach: fine space-time mesh for entire domain, with $\Delta x \sim L_{\text{turb}}$ and $\Delta t \sim \tau_{\text{turb}}$, scales very poorly at reactor scale ($R/\rho_i \gg 1$)
- `Trinity` **multi-scale** approach (Barnes et al, 2010):

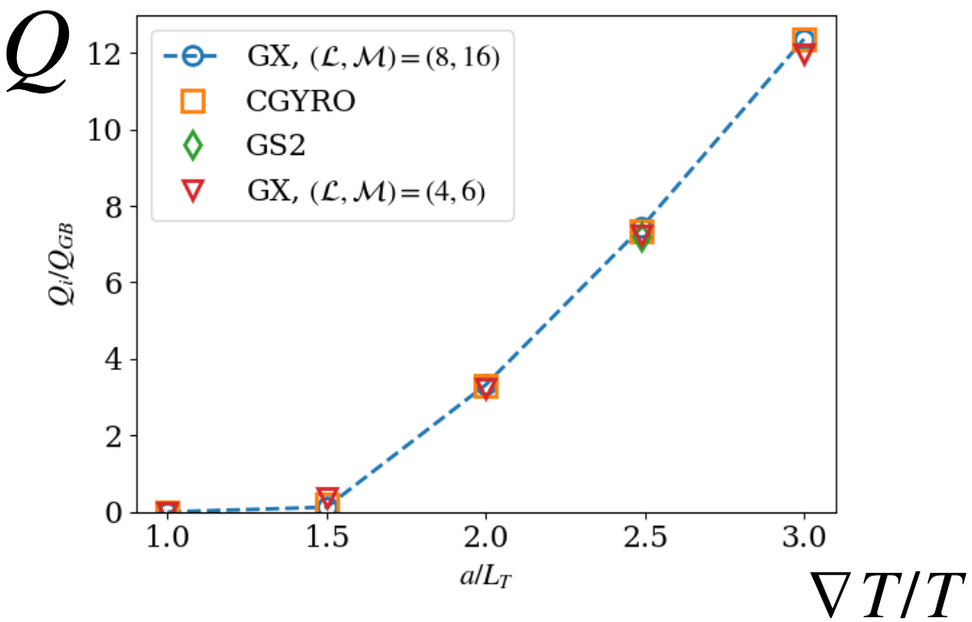
Savings from using a multi-scale approach



- Brute-force (global) approach: fine space-time mesh for entire domain, with $\Delta x \sim L_{\text{turb}}$ and $\Delta t \sim \tau_{\text{turb}}$, scales very poorly at reactor scale ($R/\rho_i \gg 1$)
- `Trinity` **multi-scale** approach (Barnes et al, 2010):
  - Can "zoom in" with small, local regions of fine grid (for turbulence) embedded in coarse grid (for slow profile evolution), in both space and time, drastically reducing the domain over which a fine space-time mesh is needed
  - Asymptotically valid as we approach reactor scales

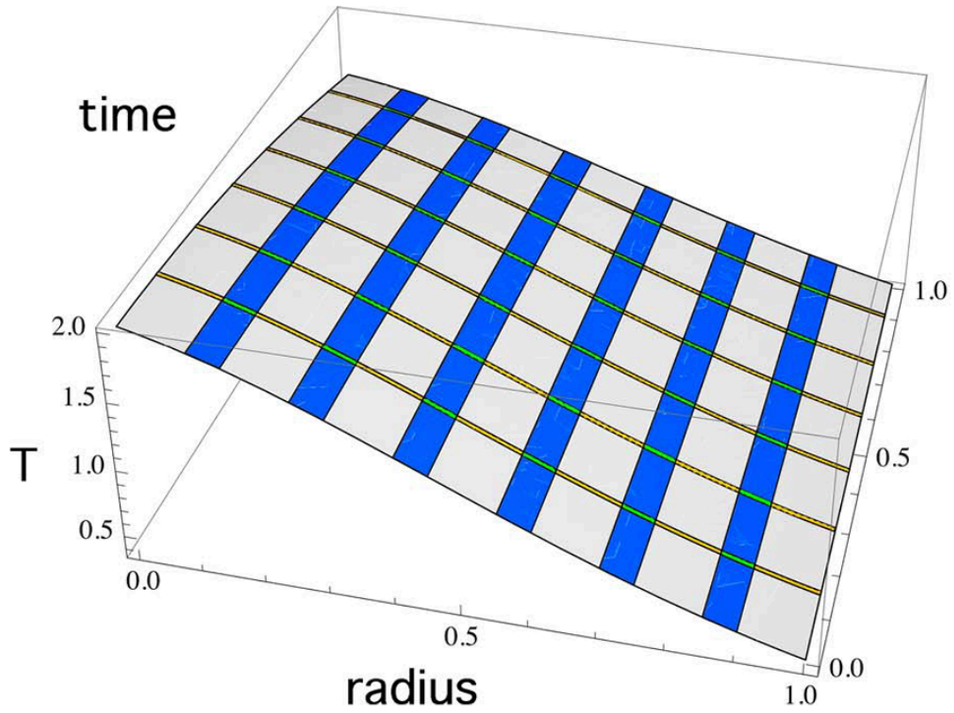
Multi-scale transport algorithm

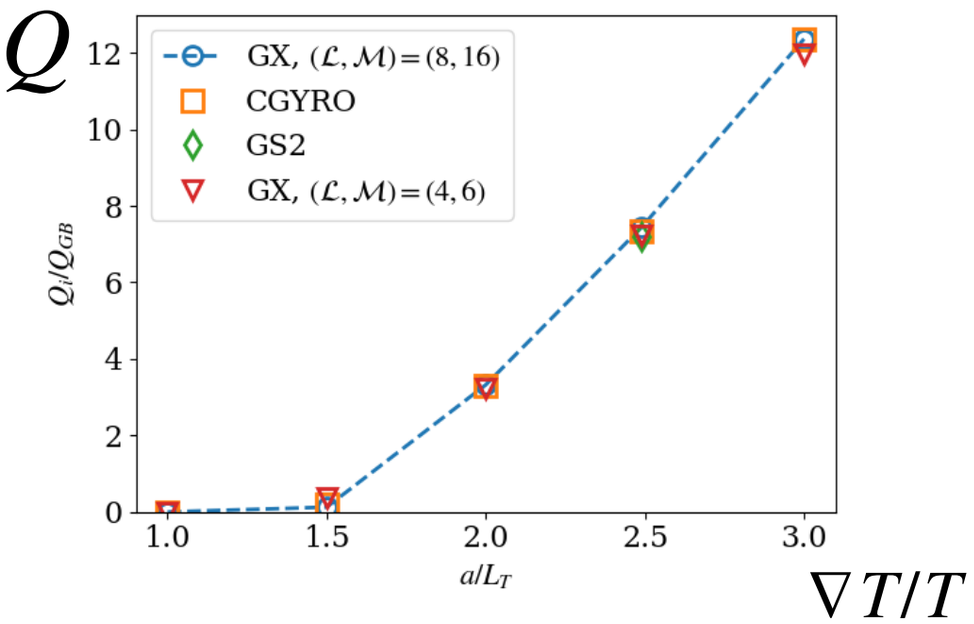




$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}\left(V'\overline{\langle Q_s\rangle}\right)=-\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

Multi-scale transport algorithm

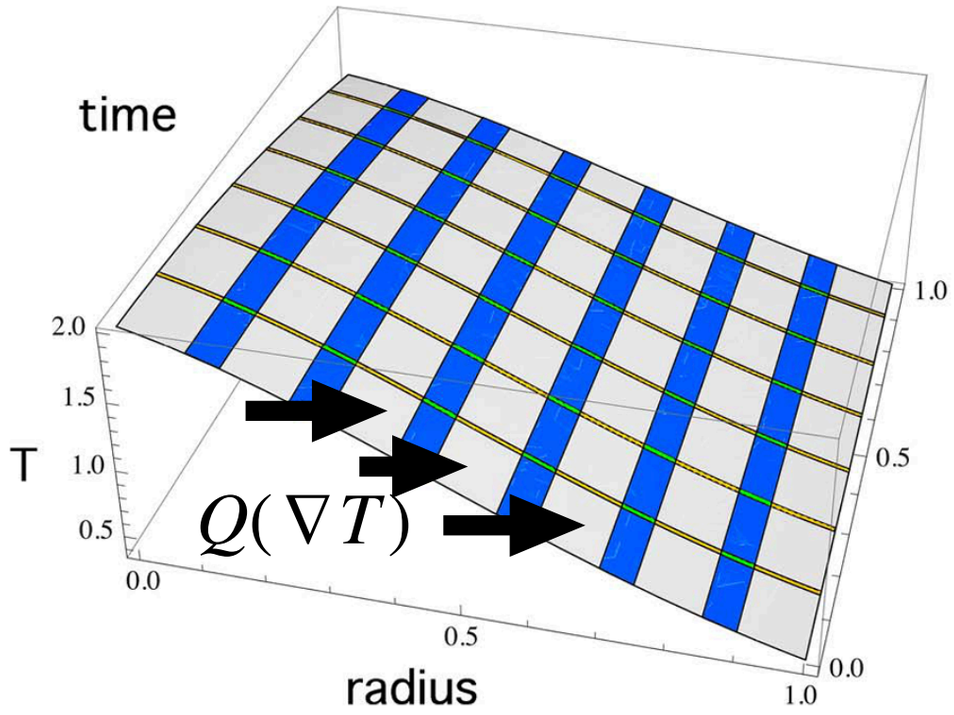


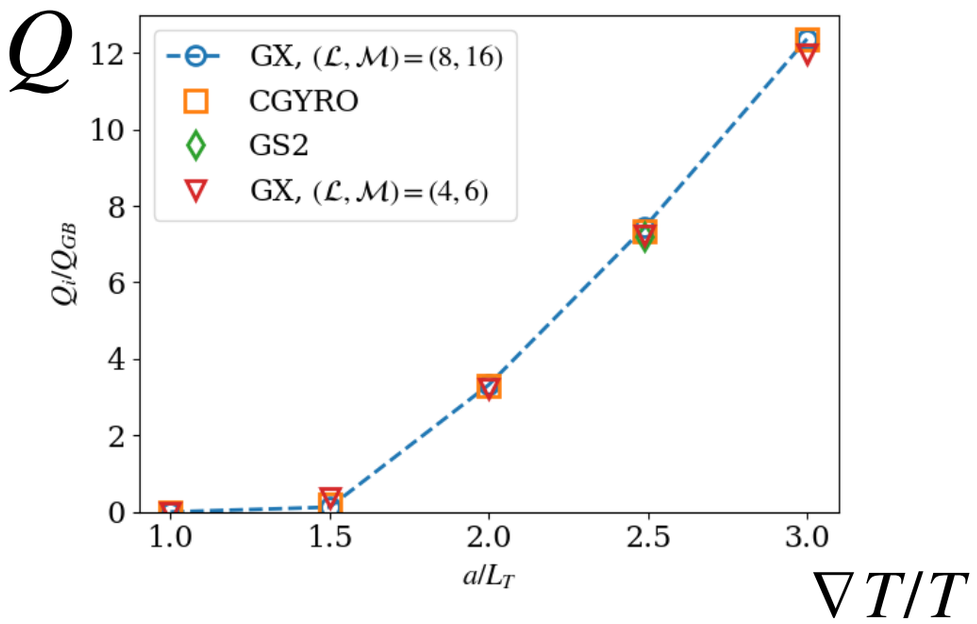


- The equilibrium profiles evolve due to diffusion-like equations

$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}\left(V'\overline{\langle Q_s\rangle}\right)=-\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

Multi-scale transport algorithm



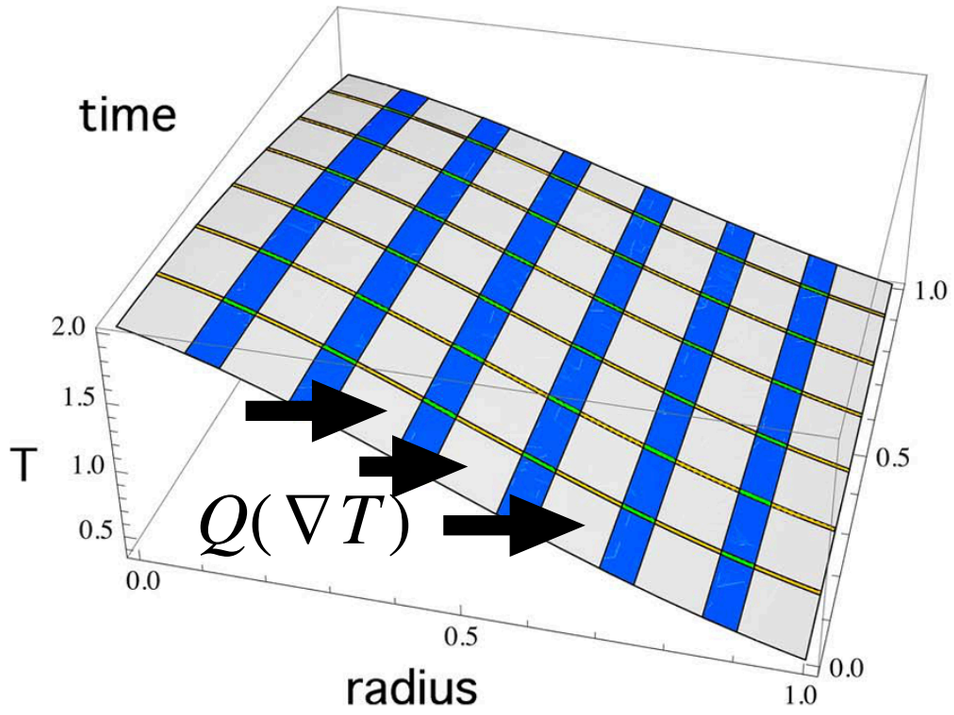


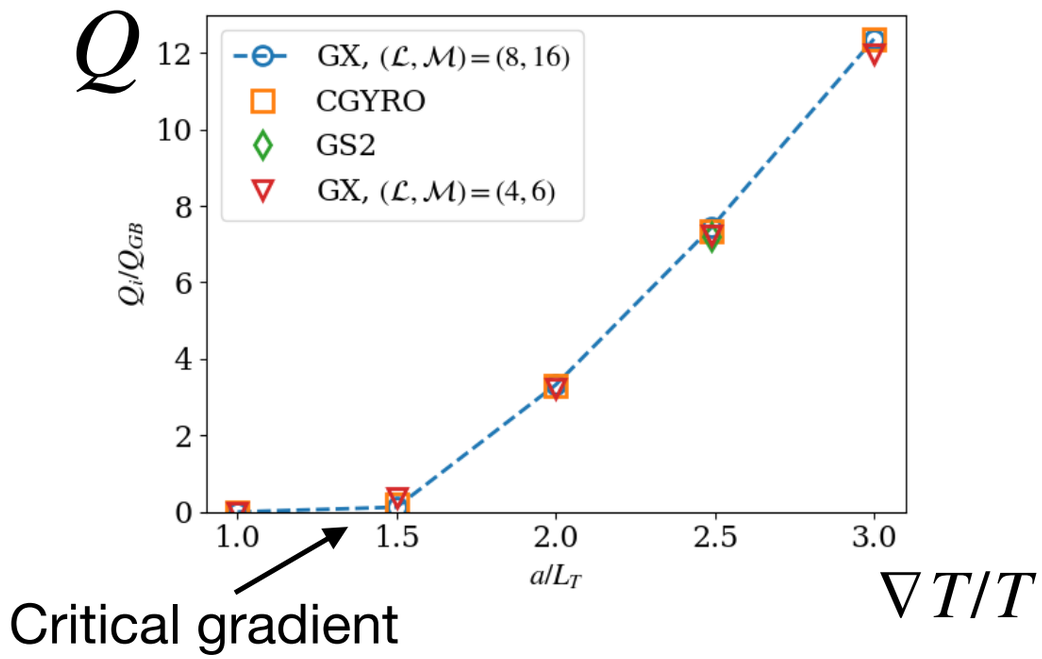
- The equilibrium profiles evolve due to diffusion-like equations

$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}\left(V'\overline{\langle Q_s\rangle}\right)=-\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

Turbulent heat flux ~ diffusion coefficient

Multi-scale transport algorithm



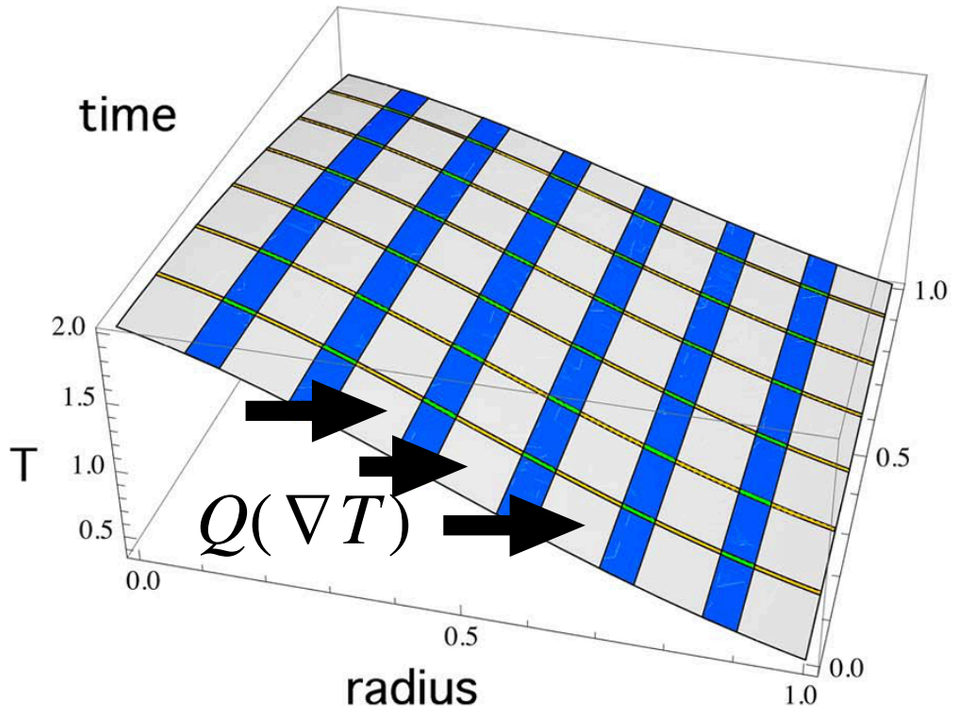


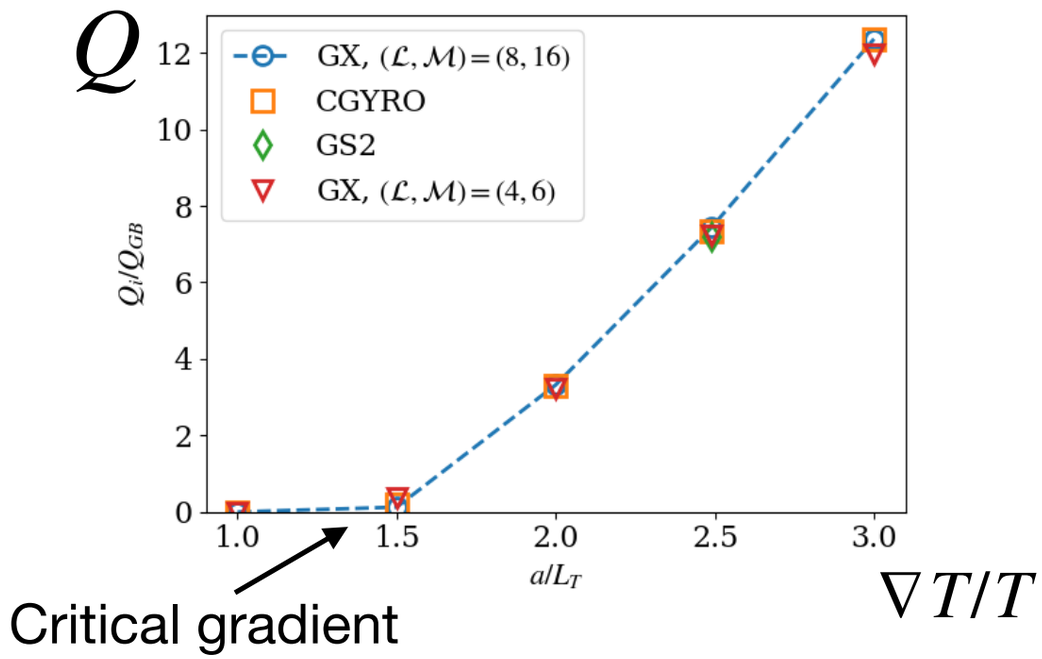
- The equilibrium profiles evolve due to diffusion-like equations

$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}\left(V'\overline{\langle Q_s\rangle}\right)=-\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

Turbulent heat flux ~ diffusion coefficient

Multi-scale transport algorithm





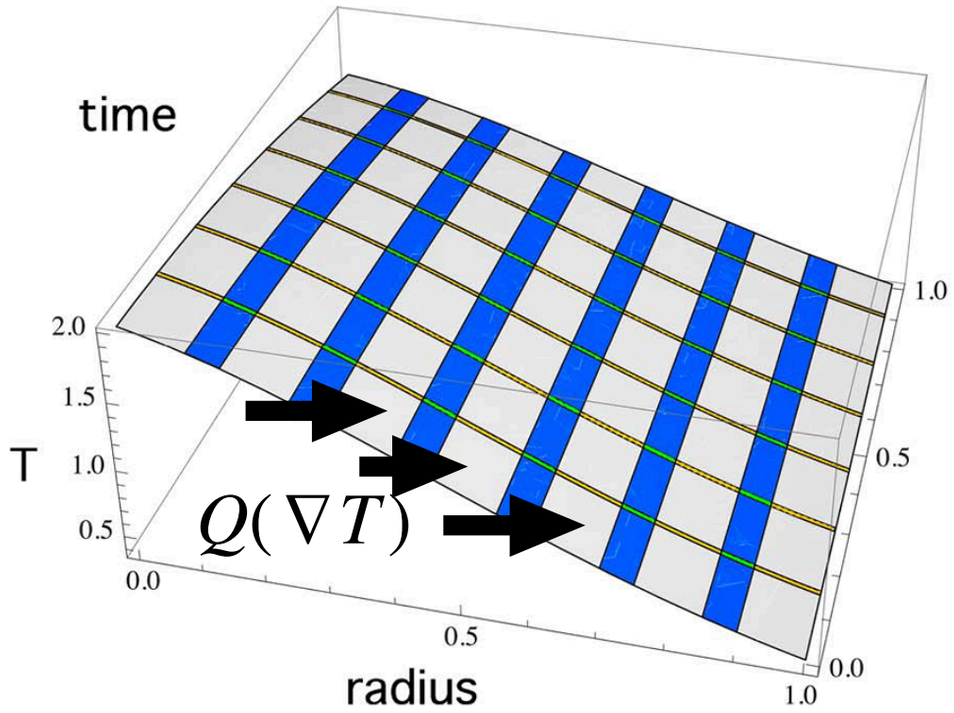
- The equilibrium profiles evolve due to diffusion-like equations

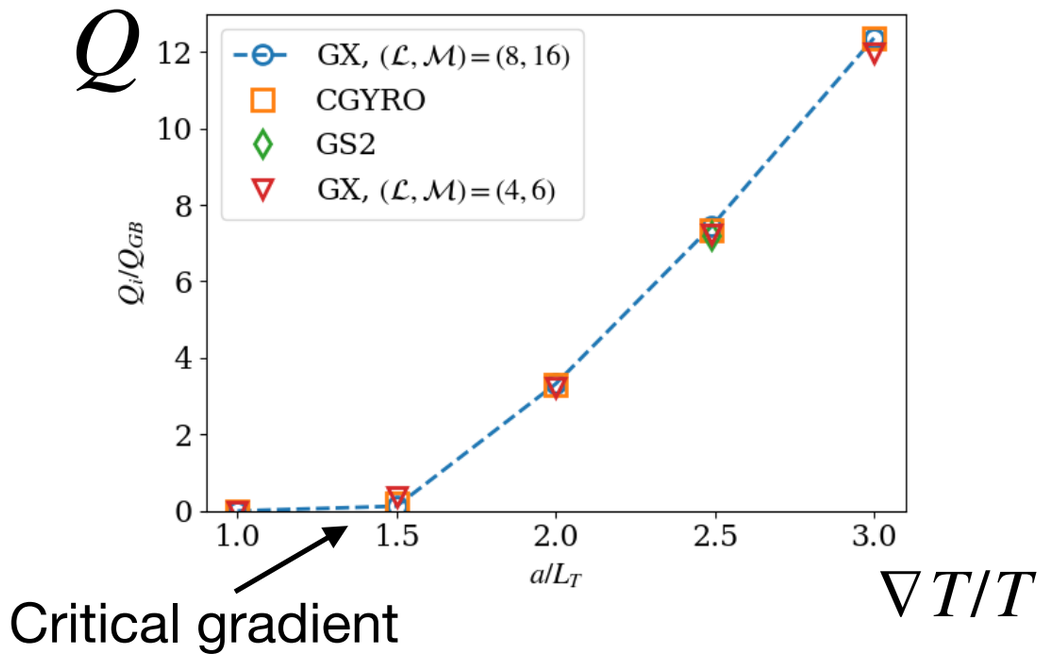
$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}\left(V'\overline{\langle Q_s\rangle}\right)=-\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

Turbulent heat flux ~ diffusion coefficient

- `Trinity` algorithm:

Multi-scale transport algorithm





- The equilibrium profiles evolve due to diffusion-like equations

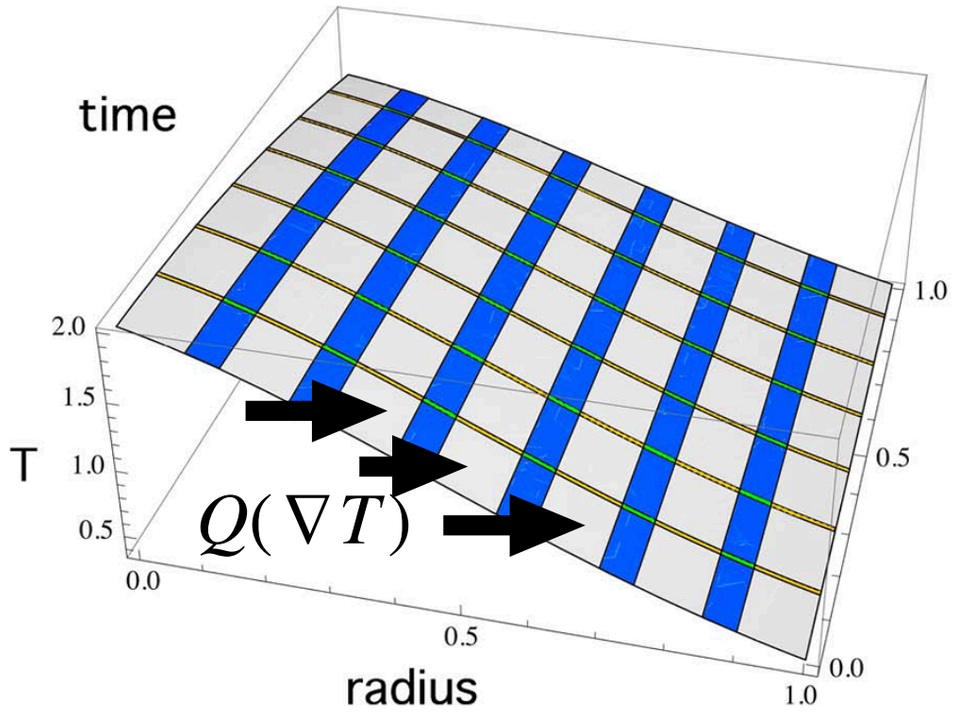
$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}(V'\overline{\langle Q_s\rangle}) = -\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

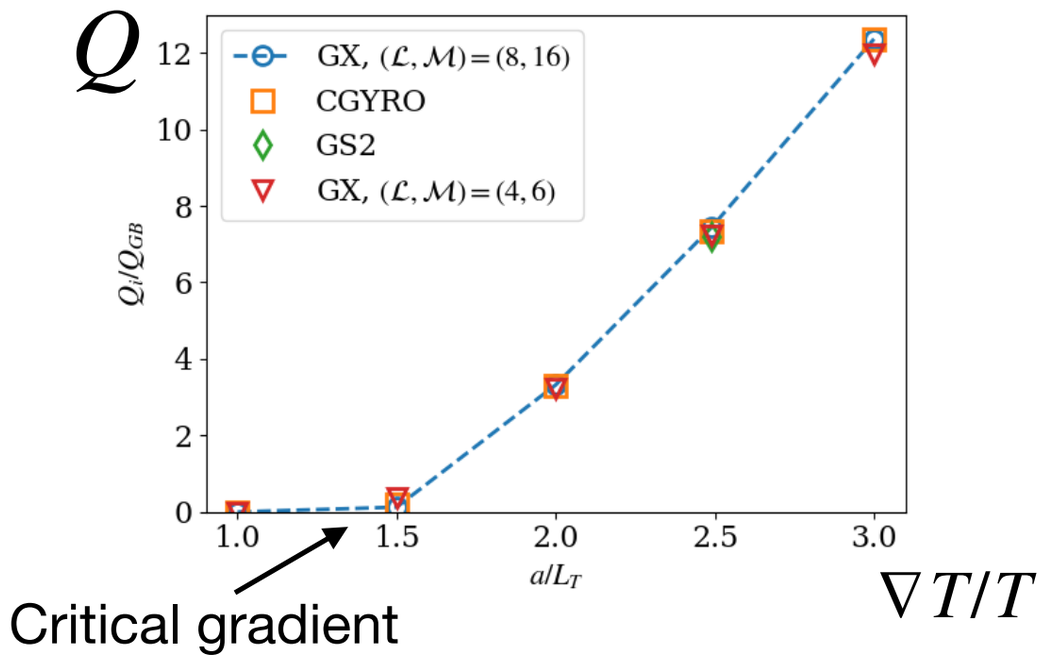
Turbulent heat flux ~ diffusion coefficient

- `Trinity` algorithm:

1. Given the equilibrium profiles on a coarse radial grid at time t^n, compute the turbulent heat flux through radial cell boundaries, $Q(r_i, t^n)$, with GX calculations

Multi-scale transport algorithm





- The equilibrium profiles evolve due to diffusion-like equations

$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}\left(V'\overline{\langle Q_s\rangle}\right)=-\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

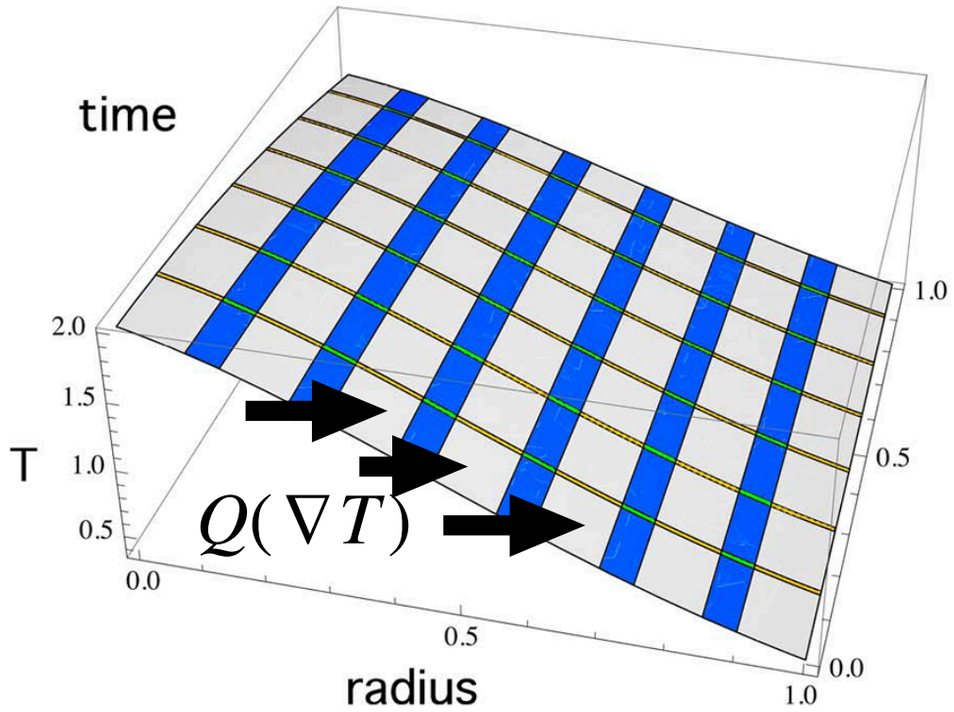
Turbulent heat flux ~ diffusion coefficient

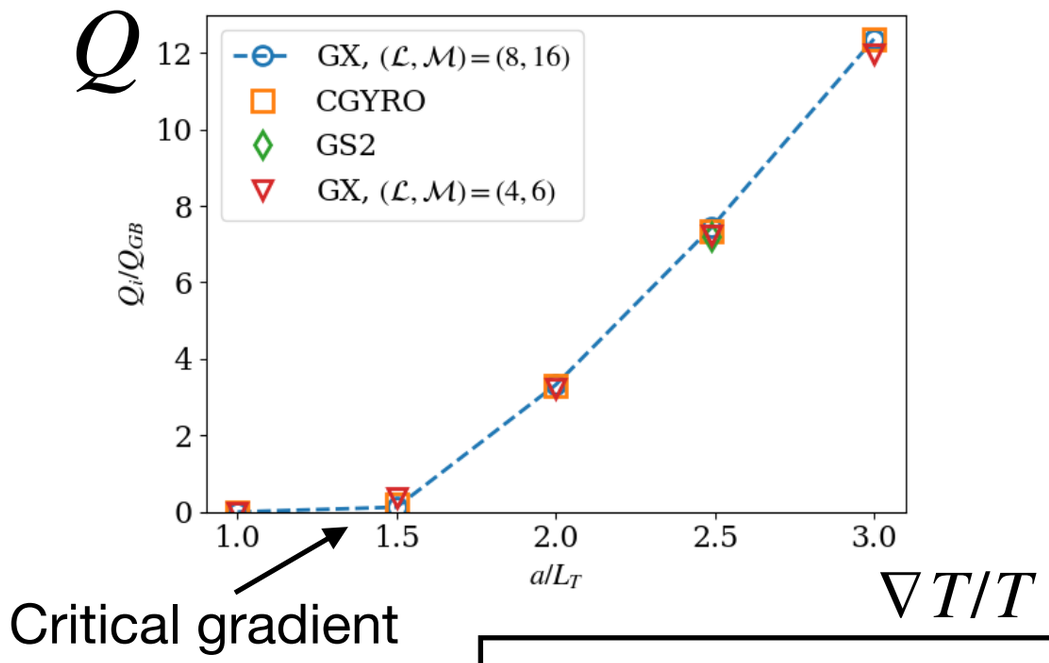
- `Trinity` algorithm:

1. Given the equilibrium profiles on a coarse radial grid at time t^n, compute the turbulent heat flux through radial cell boundaries, $Q(r_i, t^n)$, with GX calculations

2. Use the turbulent fluxes to advance the equilibrium profiles $t^n \rightarrow t^n + \Delta t_{\text{transp}}$ on transport timescale (~s)

Multi-scale transport algorithm





How do we compute Q, and how do we do it as efficiently as possible?

- The equilibrium profiles evolve due to diffusion-like equations

$$\frac{3}{2}\frac{\partial p_s}{\partial t}+\frac{1}{V'}\frac{\partial}{\partial \psi}\left(V'\overline{\langle Q_s\rangle}\right)=-\overline{\langle H_s\rangle}+\frac{3}{2}n_s\sum_u \nu_{su}^{\varepsilon}(T_u-T_s)+\overline{\langle S_p\rangle}$$

Turbulent heat flux ~ diffusion coefficient

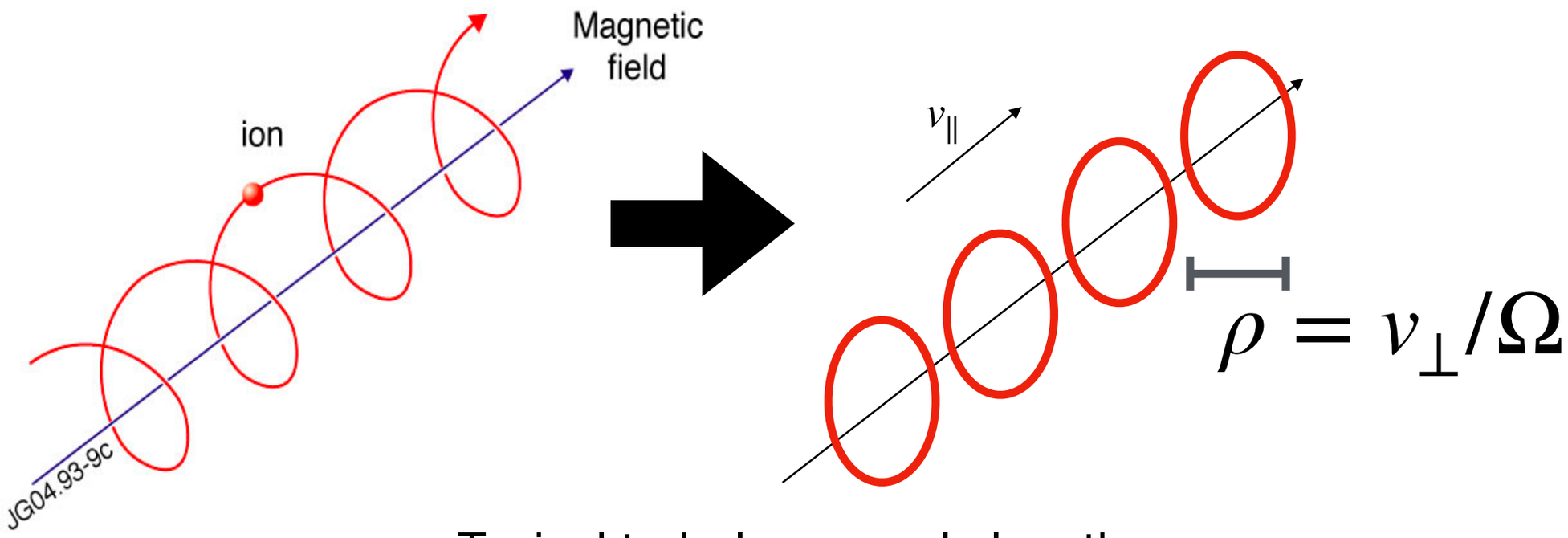
- `Trinity` algorithm:

1. Given the equilibrium profiles on a coarse radial grid at time t^n, compute the turbulent heat flux through radial cell boundaries, $Q(r_i, t^n)$, with GX calculations

2. Use the turbulent fluxes to advance the equilibrium profiles $t^n \rightarrow t^n + \Delta t_{\text{transp}}$ on transport timescale (~s)

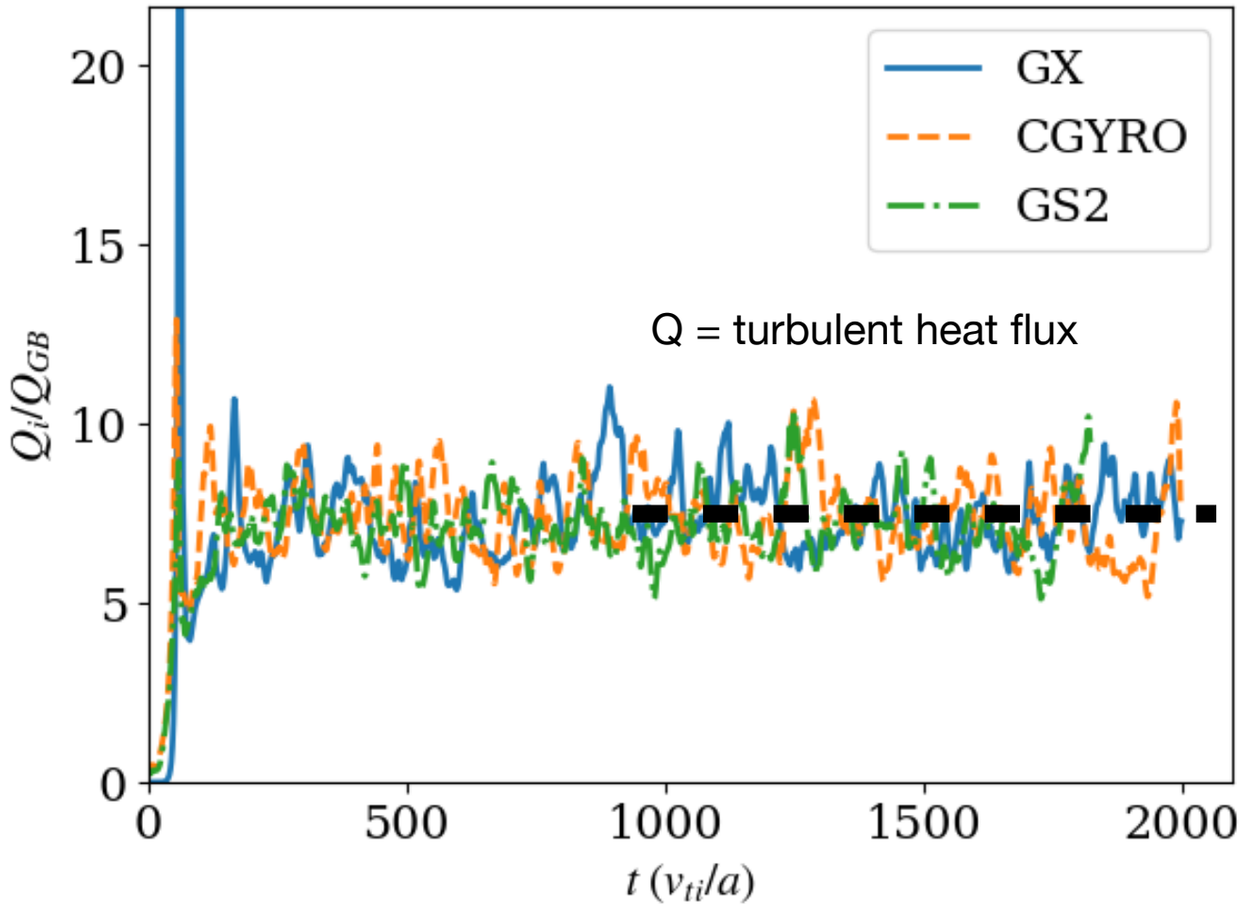
Modeling turbulence in fusion plasmas

- Turbulence in a fusion plasma (both core and boundary) is well-described by **gyrokinetics**
  - A kinetic description (where particle positions *and* velocities are tracked) is necessary because fusion plasmas are very collisionless
  - A naive kinetic description would involve a 6D phase space, tracking PDF $f(x, y, z, v_x, v_y, v_z)$
    - E.g. Vlasov-Boltzmann-Maxwell system
  - We can reduce the dimensionality by one velocity dimension by averaging out the high frequency particle gyration, tracking PDF $f(x, y, z, v_{\|}, v_{\perp})$
- Effectively a transformation from discrete charged particles → rings of charge



Typical turbulence scale length
~ ion gyroradius ρ_i

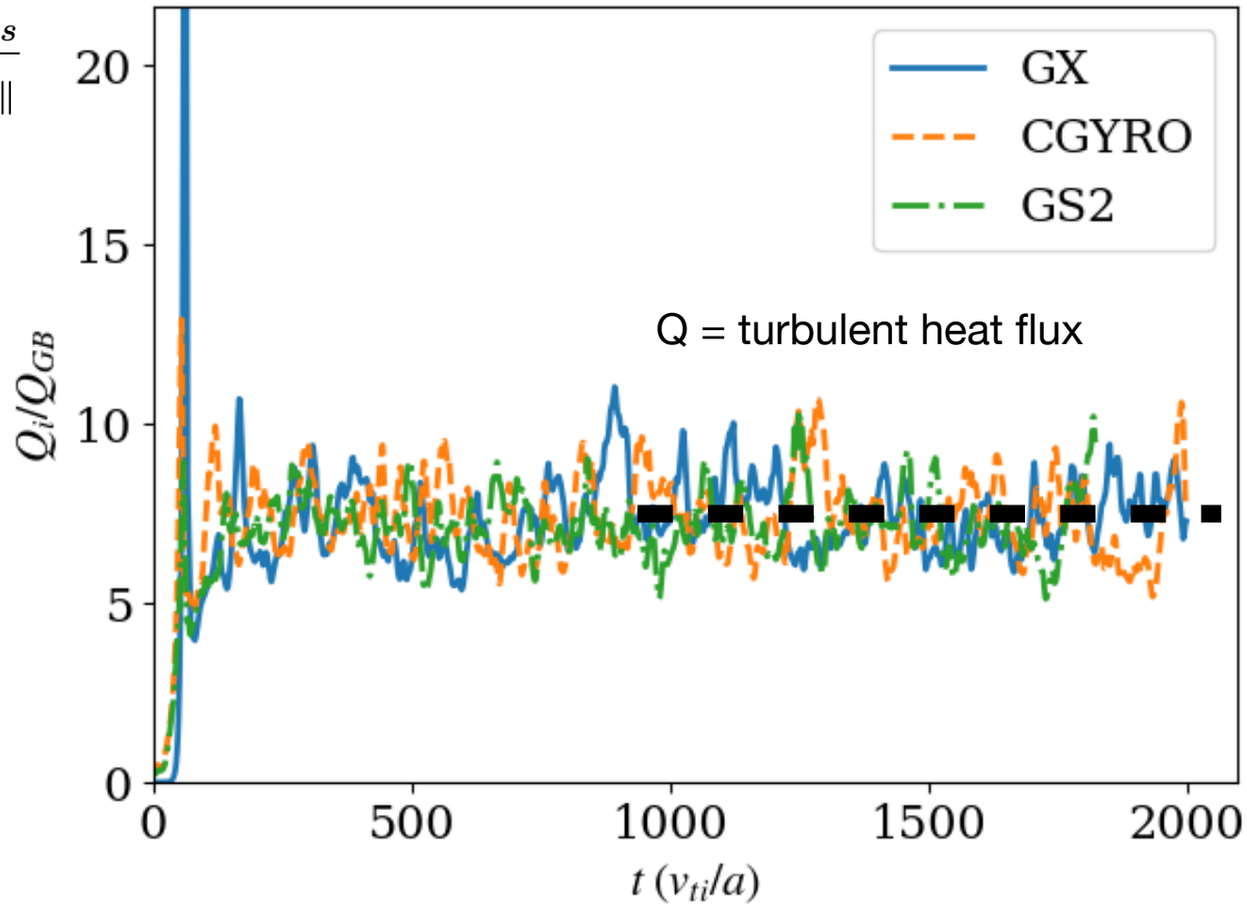
Computing the turbulent fluxes



Computing the turbulent fluxes

- The gyrokinetic (GK) equation is a 5D nonlinear PDE modeling turbulent fluctuations driven by gradients in the (fixed) background

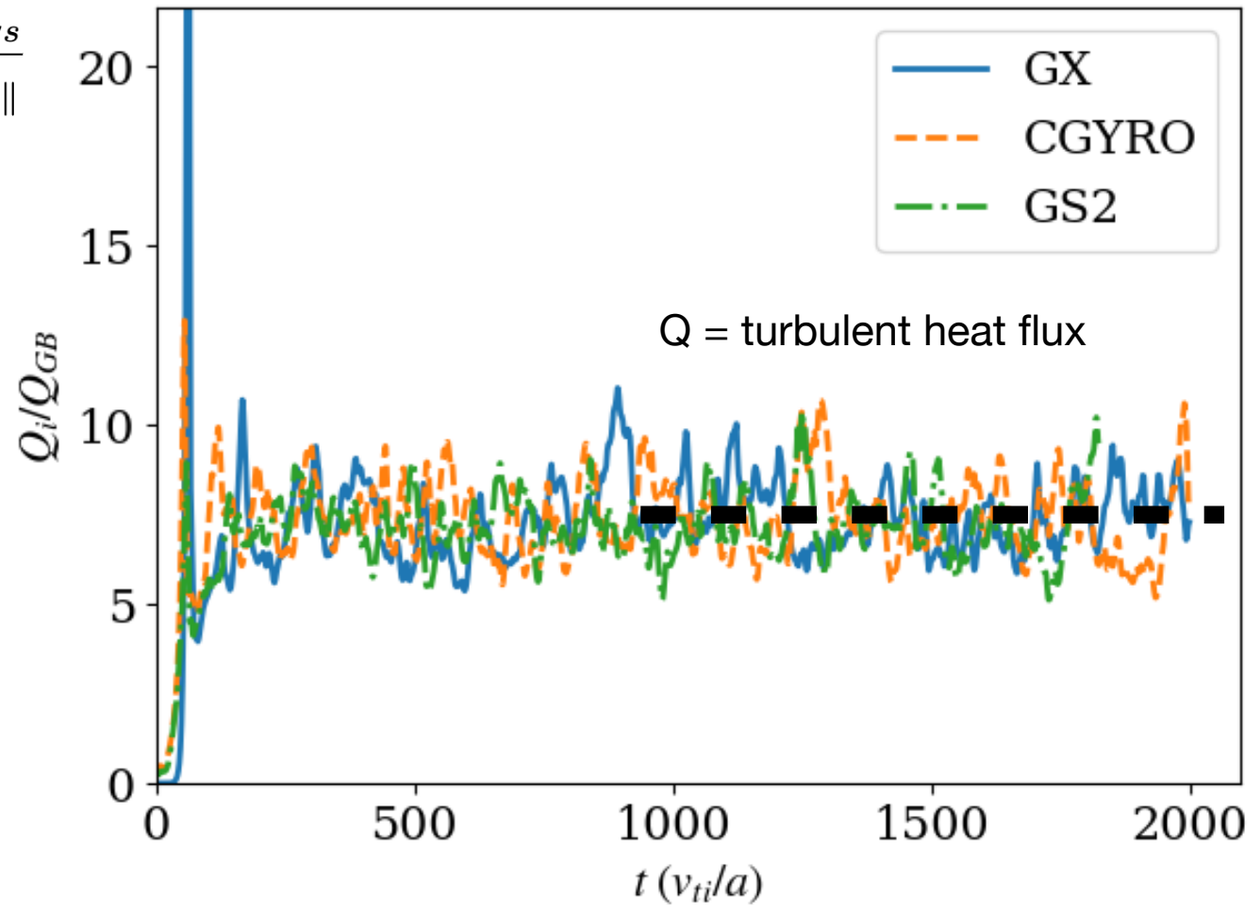
$$\frac{\partial h_s}{\partial t} + \left[v_{ts} v_\parallel \hat{\mathbf{b}} + \langle \mathbf{v}_\chi \rangle_{\mathbf{R}} + \frac{\tau_s}{Z_s} \mathbf{v}_d \right] \cdot \nabla h_s - v_{ts} \mu \left(\hat{\mathbf{b}} \cdot \nabla B \right) \frac{\partial h_s}{\partial v_\parallel}$$
$$= \frac{Z_s}{\tau_s} F_{Ms} \frac{\partial \langle \chi \rangle_{\mathbf{R}}}{\partial t} - \langle \mathbf{v}_\chi \rangle_{\mathbf{R}} \cdot \nabla \big|_E F_{Ms} + C(h_s).$$



Computing the turbulent fluxes

- The gyrokinetic (GK) equation is a 5D nonlinear PDE modeling turbulent fluctuations driven by gradients in the (fixed) background

$$\frac{\partial h_s}{\partial t} + \left[v_{ts} v_\parallel \hat{\mathbf{b}} + \langle \mathbf{v}_\chi \rangle_{\mathbf{R}} + \frac{\tau_s}{Z_s} \mathbf{v}_d \right] \cdot \nabla h_s - v_{ts} \mu \left(\hat{\mathbf{b}} \cdot \nabla B \right) \frac{\partial h_s}{\partial v_\parallel}$$
$$= \frac{Z_s}{\tau_s} F_{Ms} \frac{\partial \langle \chi \rangle_{\mathbf{R}}}{\partial t} - \langle \mathbf{v}_\chi \rangle_{\mathbf{R}} \cdot \nabla \big|_E F_{Ms} + C(h_s).$$



- The turbulent heat flux Q that is needed by `Trinity` is given by a velocity integral of h, time-averaged over the turbulent statistically-steady state

$$Q_s = \left\langle \int \mathrm{d}^3 \boldsymbol{v} \frac{m_s v^2}{2} \left[\left(\frac{c}{B} \hat{\boldsymbol{b}} \times \nabla \langle \varphi \rangle_{\mathbf{R}} \right) \cdot \nabla \psi \right] \langle h_s \rangle_{\mathbf{r}} \right\rangle_{\psi, t}$$

Computing the turbulent fluxes

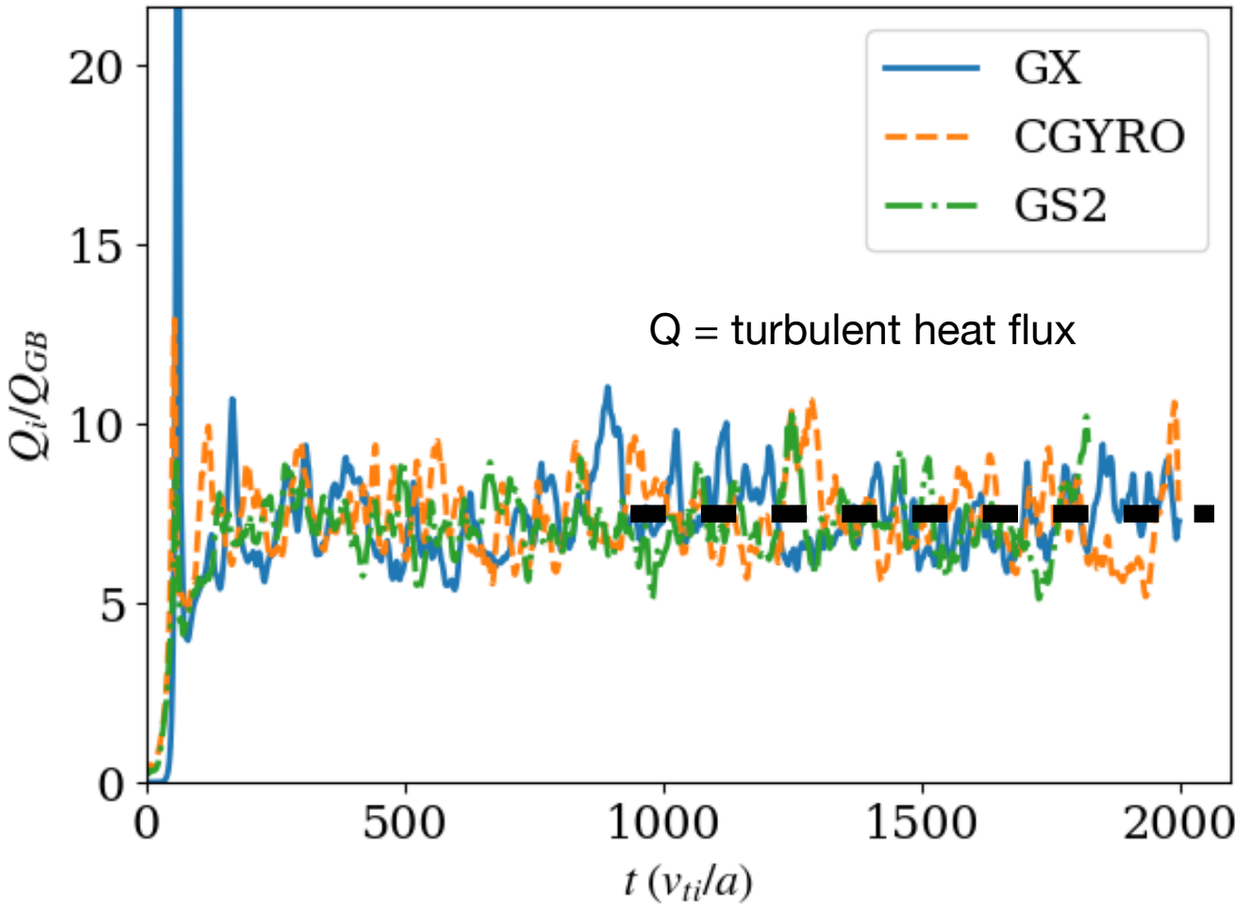
$$\frac{\partial h_s}{\partial t} + \left[v_{ts} v_\parallel \hat{\mathbf{b}} + \langle \mathbf{v}_\chi \rangle_\mathbf{R} + \frac{\tau_s}{Z_s} \mathbf{v}_d \right] \cdot \nabla h_s - v_{ts} \mu \left(\hat{\mathbf{b}} \cdot \nabla B \right) \frac{\partial h_s}{\partial v_\parallel}$$
$$= \frac{Z_s}{\tau_s} F_{Ms} \frac{\partial \langle \chi \rangle_\mathbf{R}}{\partial t} - \langle \mathbf{v}_\chi \rangle_\mathbf{R} \cdot \nabla \big|_E F_{Ms} + C(h_s).$$

- The GK equation can be solved with standard PDE methods (in 5D!): finite difference, finite element, spectral, discontinuous Galerkin, etc

- However, most GK codes are often too expensive (~10k-100k CPU-hours) for use inside Trinity or in an optimization loop, where O(100)-O(1000) or more flux evaluations could be needed

- This has led to the use of lower-fidelity models for the turbulence calculations to accelerate transport modeling

    - But difficult to project to reactor-scale without first-principles



Gyrokinetic vs Gyrofluid

- We didn't always have the computing power we have now. In the before times (~25 years ago), a single nonlinear gyrokinetic simulation was too expensive.

- Progress was made by the development of gyrofluid models by Hammett, Dorland, Beer, Snyder, Waltz, etc in the 90s

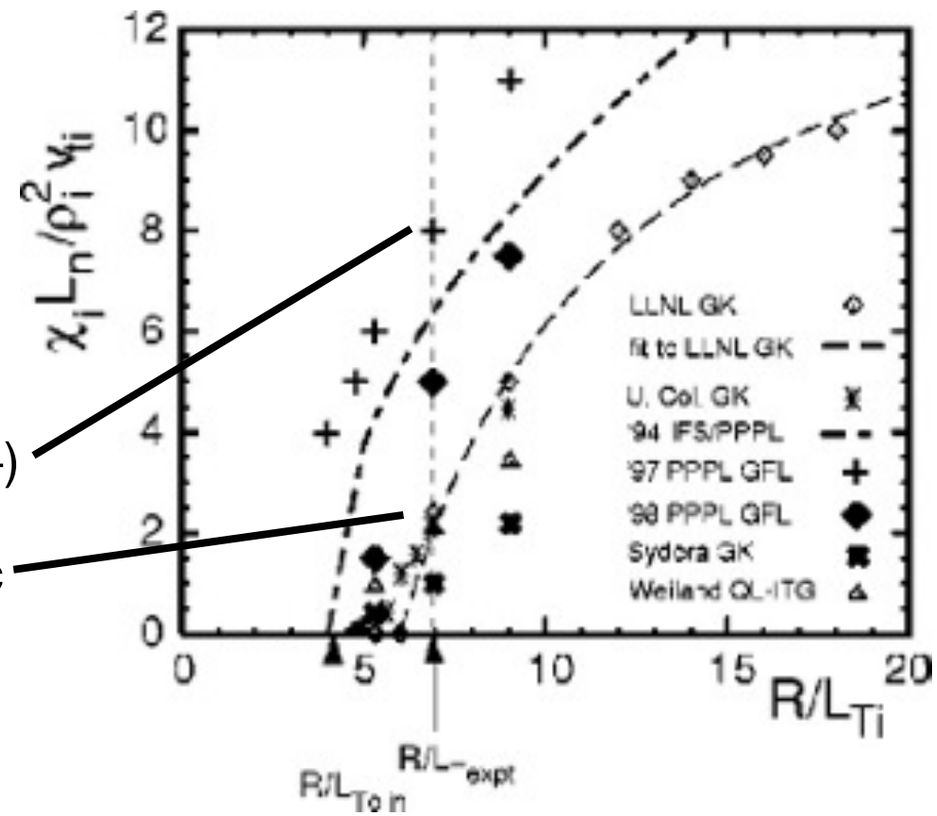
  - Take moments of the gyrokinetic equation to get fluid equations (5D → 3D) for density, momentum, temperature, etc

  - Kept only ~6 moments → closure problem. In Beer-Hammett model, closures designed to fit kinetic dispersion relation (e.g. Hammett-Perkins Landau damping closure)

- However, gyrofluid models were inaccurate for nonlinear cases because of incomplete physics (e.g. zonal flows over-damped by closures, leading to 2-3x larger heat fluxes than GK, no Dimits shift)

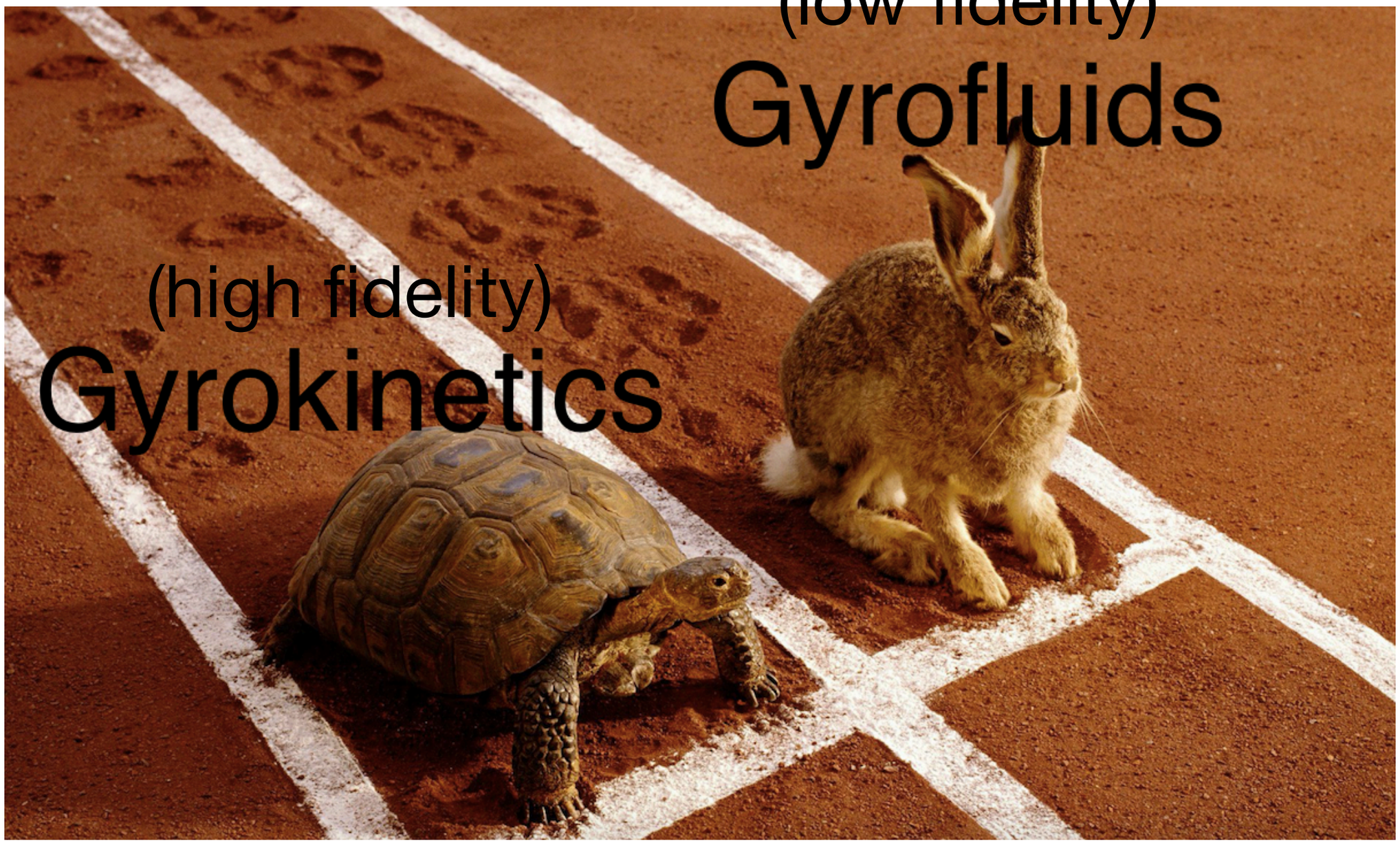
Beer-Hammett gyrofluid (+)

gyrokinetic

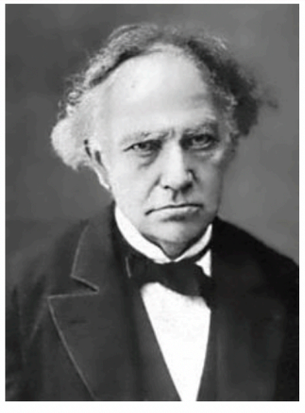


Dimits et al, 2000
origins of "Cyclone base case"
and "Dimits shift"

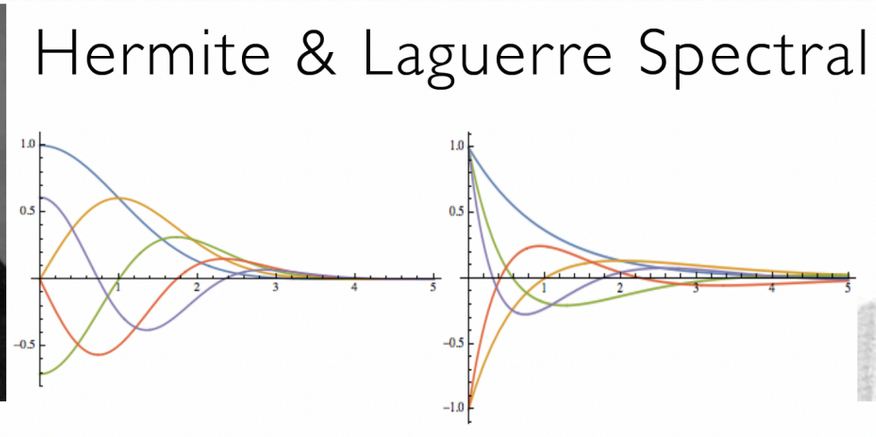
A classic problem…

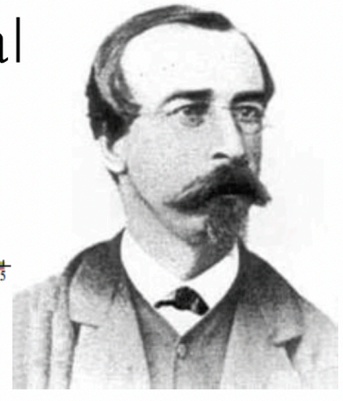


Laguerre-Hermite pseudo-spectral velocity formulation



Msr. Hermite





Msr. Laguerre

- **Why stop at 6 gyrofluid moments?** Mandell et al (2018) developed a formulation to **extend the gyrofluid model** to arbitrary number of moments, which is **mathematically equivalent to full gyrokinetics**
- Expand the velocity dependence of distribution function in Laguerre and Hermite polynomial basis

$$h(\vec{x}, v_\parallel, \mu) = \sum_{\ell=0} \sum_{m=0} \frac{(-1)^\ell}{\sqrt{m!}} \mathrm{L}_\ell(\mu B) \mathrm{He}_m(v_\parallel) e^{-v_\parallel^2/2 - \mu B} H_{\ell,m}(\vec{x})$$

- The spectral "modes" $H_{\ell,m}$ are 3D (gyro)fluid moments, and can be defined via projection onto basis:

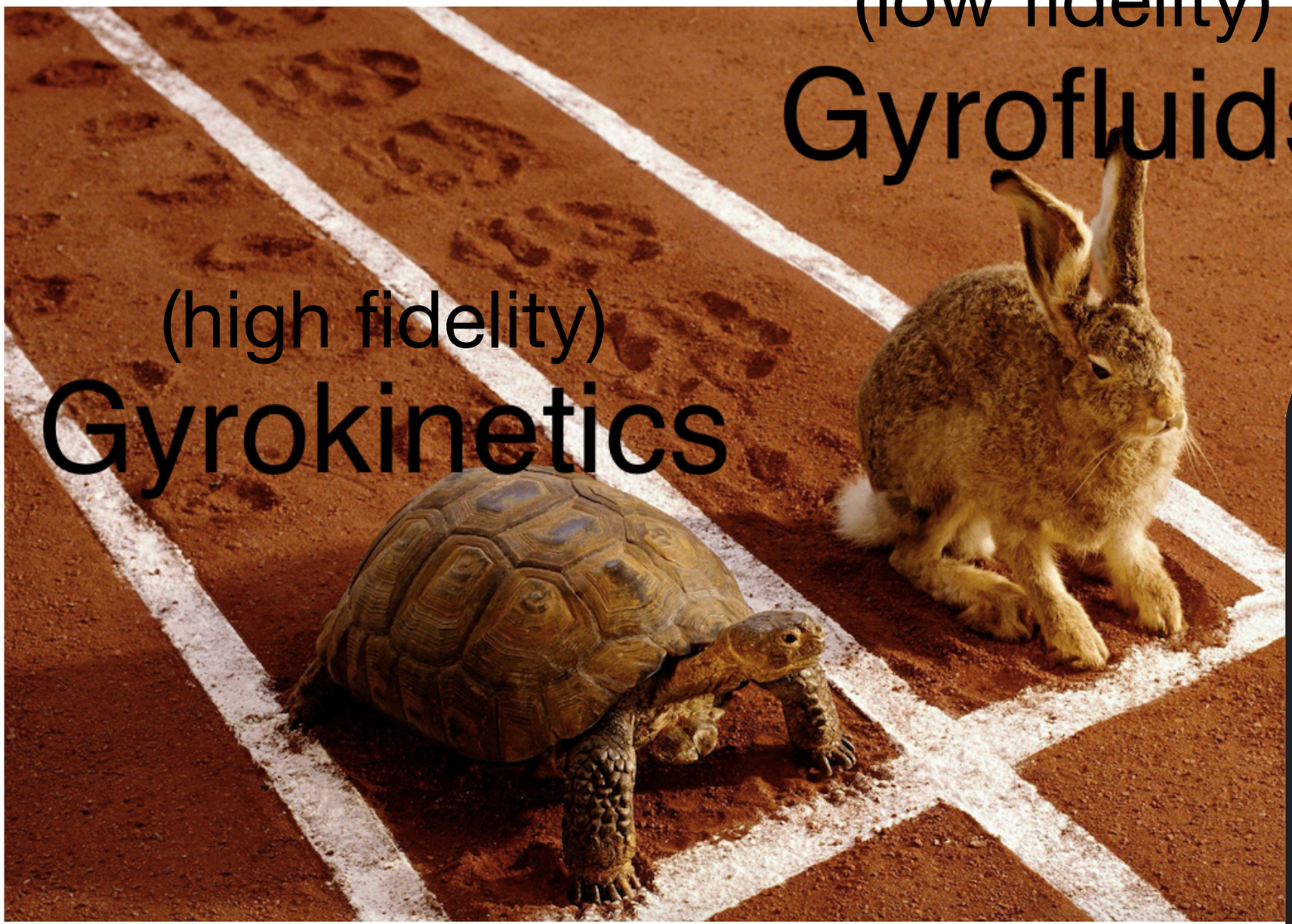
$$H_{\ell,m}(\vec{x}) = \int d^3v \, \frac{(-1)^\ell}{\sqrt{m!}} \mathrm{L}_\ell(\mu B) \mathrm{He}_m(v_\parallel) h(\vec{x}, v_\parallel, \mu)$$

- The lowest order moments are directly related to the 6 gyrofluid moments from the (relatively successful) models from the 90s

$$(H_{0,0}, H_{1,0}, \sqrt{2} H_{2,0}, \sqrt{3} H_{3,0}, H_{0,1}, H_{1,1}) \rightarrow (n, u_\parallel, T_\parallel, q_\parallel, T_\perp, q_\perp)$$

- Spectral velocity representation → arbitrary number of coupled fluid moment evolution equations → **flexible fidelity** between **gyrofluid at low resolution** and **gyrokinetic at high resolution**

Winning the race



(flexible fidelity)

What is GX?

- **GX** is a code for solving the nonlinear gyrokinetic system for low-frequency turbulence in magnetized plasmas, particularly tokamaks and stellarators

- GX uses Fourier-Laguerre-Hermite (pseudo-)spectral methods

- Main target is device **design** at reactor scale (rather than physics exploration or validation)

  - Radially-local flux-tube approach is optimal in reactor limit of small $\rho_* \equiv \rho_i/R$

  - Simple Dougherty collision operator that is very efficient in Laguerre-Hermite basis

- GX is a **GPU-native** code

  - Every algorithmic design decision made to target GPUs, the fastest computational platforms available today

    - Single precision; fast transforms (e.g. FFTs)

https://gx.readthedocs.io

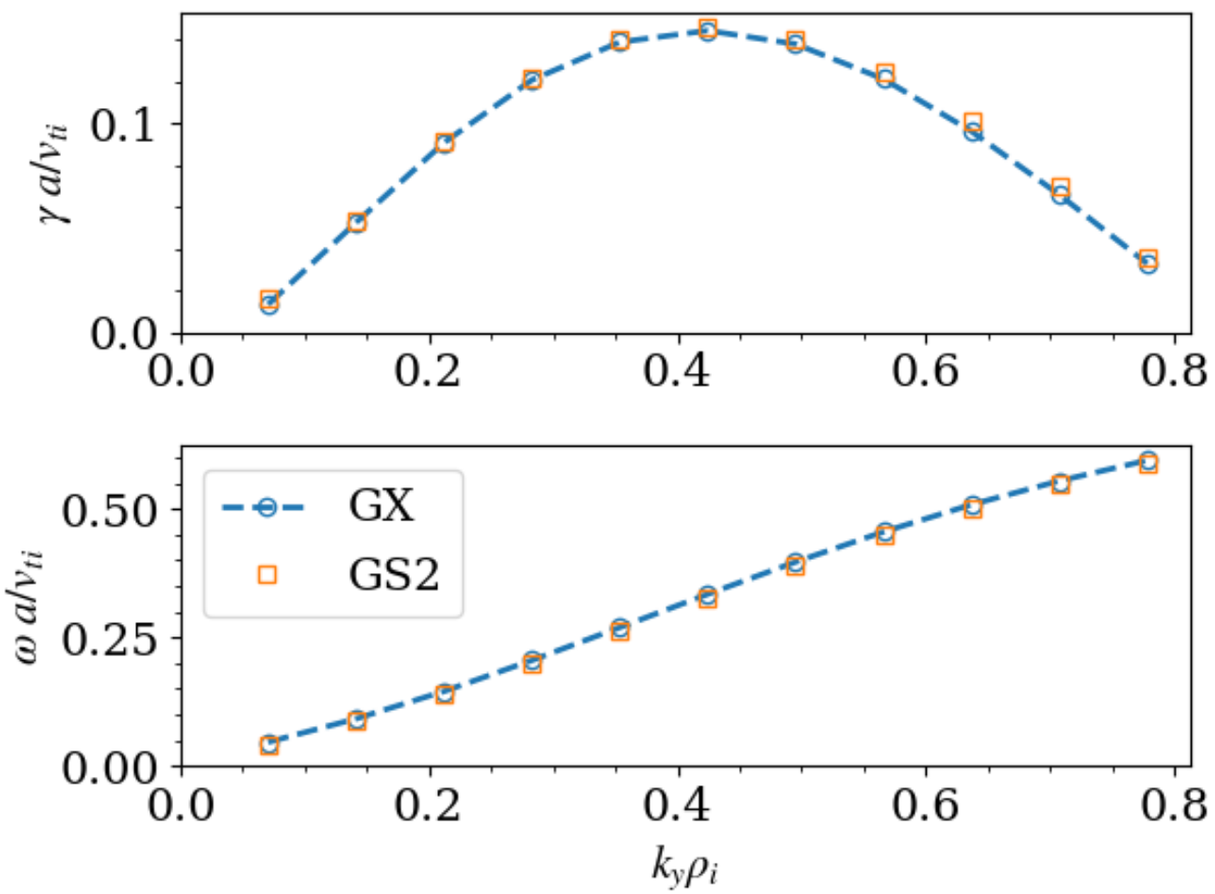
- GX can produce useful turbulence simulations in **minutes** on one or a few GPUs

- Open source (pre-release): https://bitbucket.org/gyrokinetics/gx

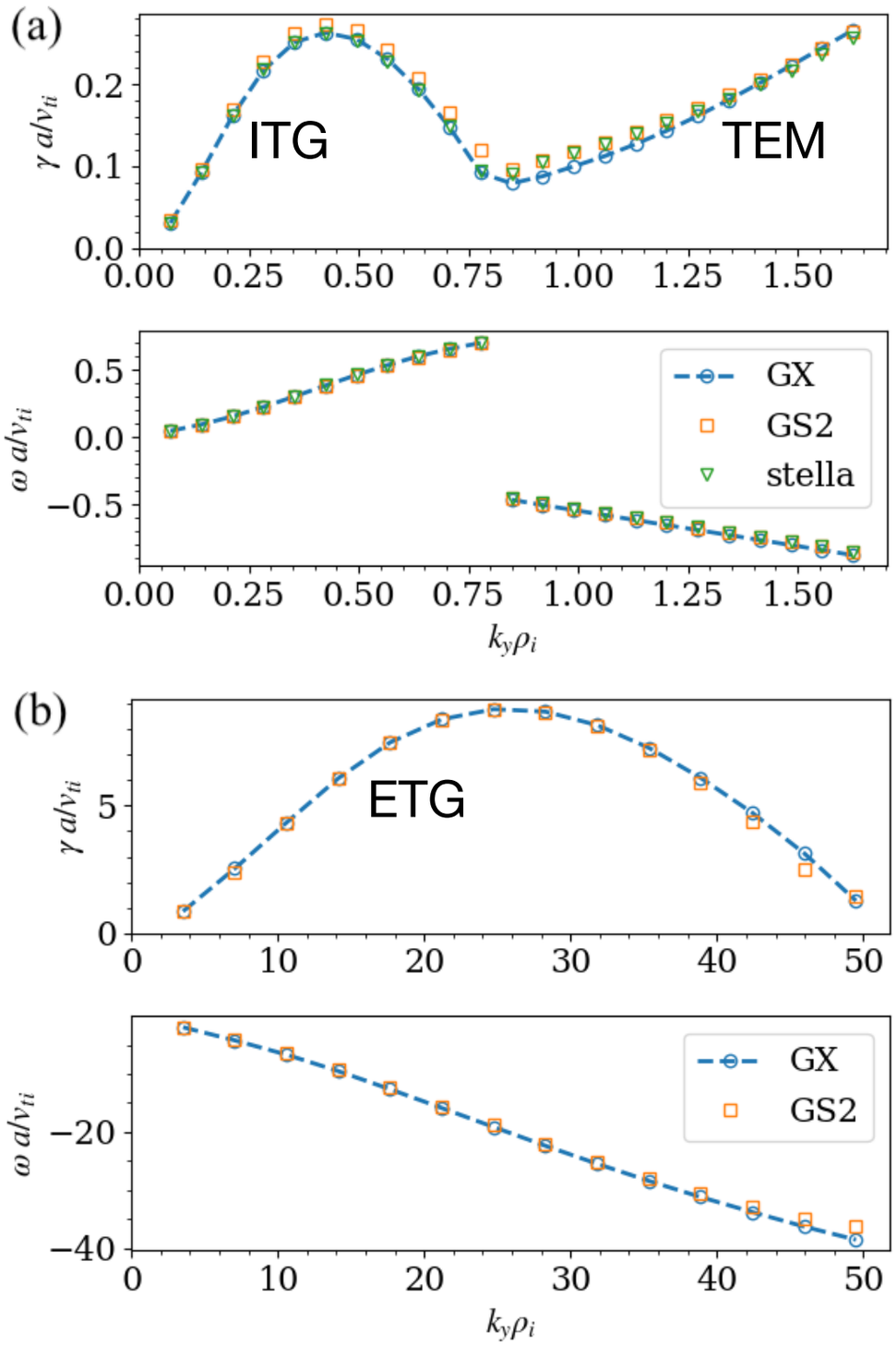
Linear benchmarking: ITG instability (Cyclone)

Cyclone base case parameters with Miller geometry

Adiabatic electrons



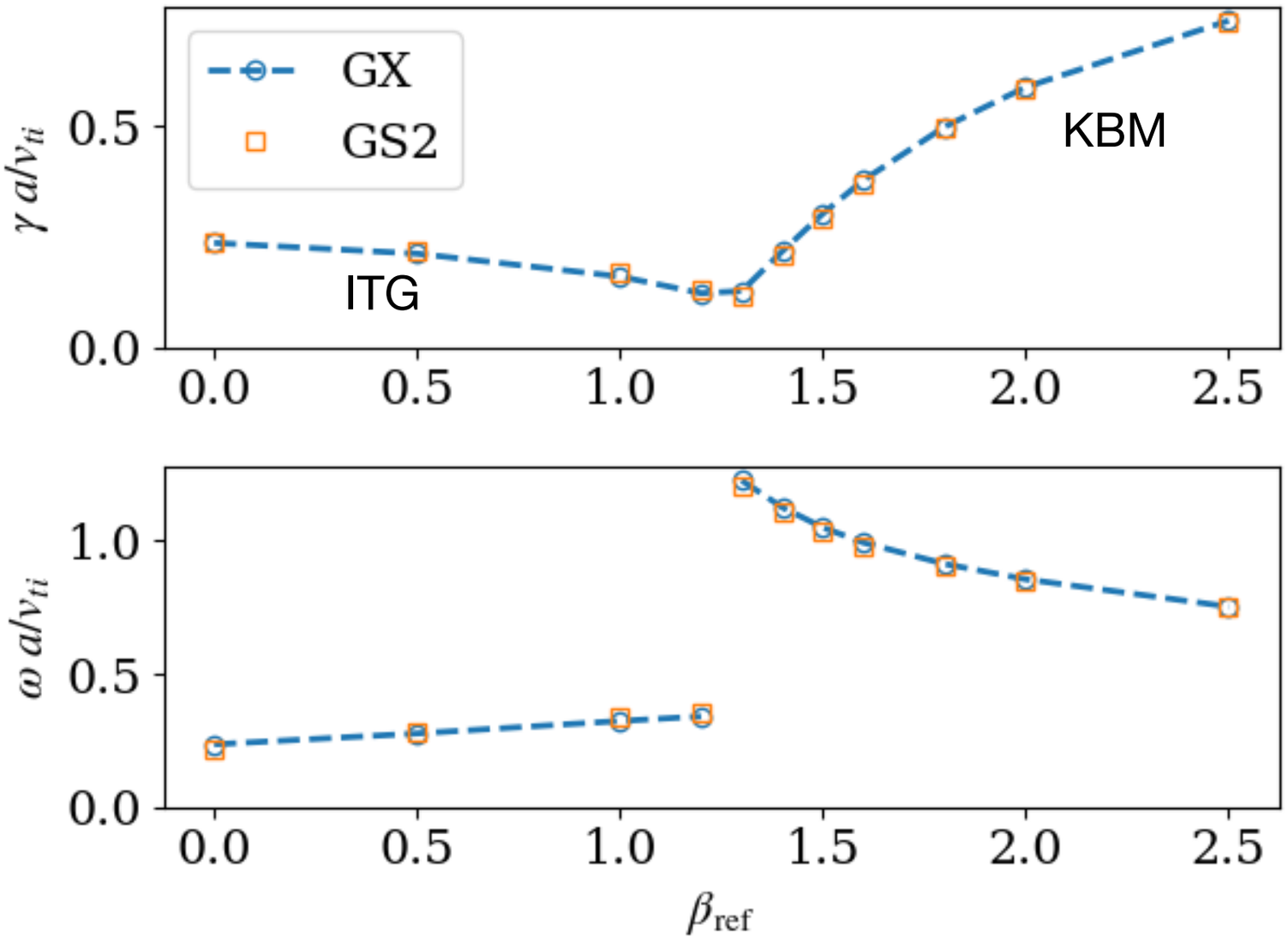
Kinetic electrons



Mandell et al, arXiv:2209.06731 (2022)

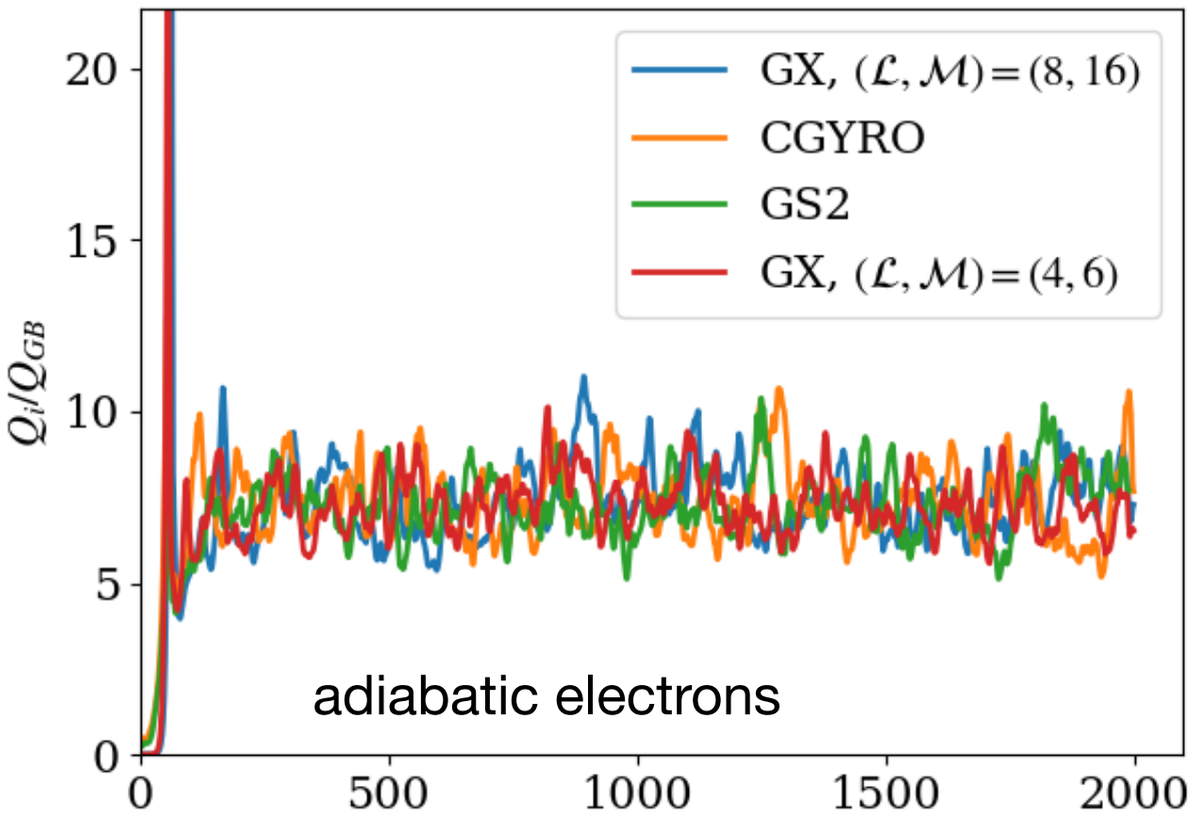
Linear benchmarking: ITG-KBM transition

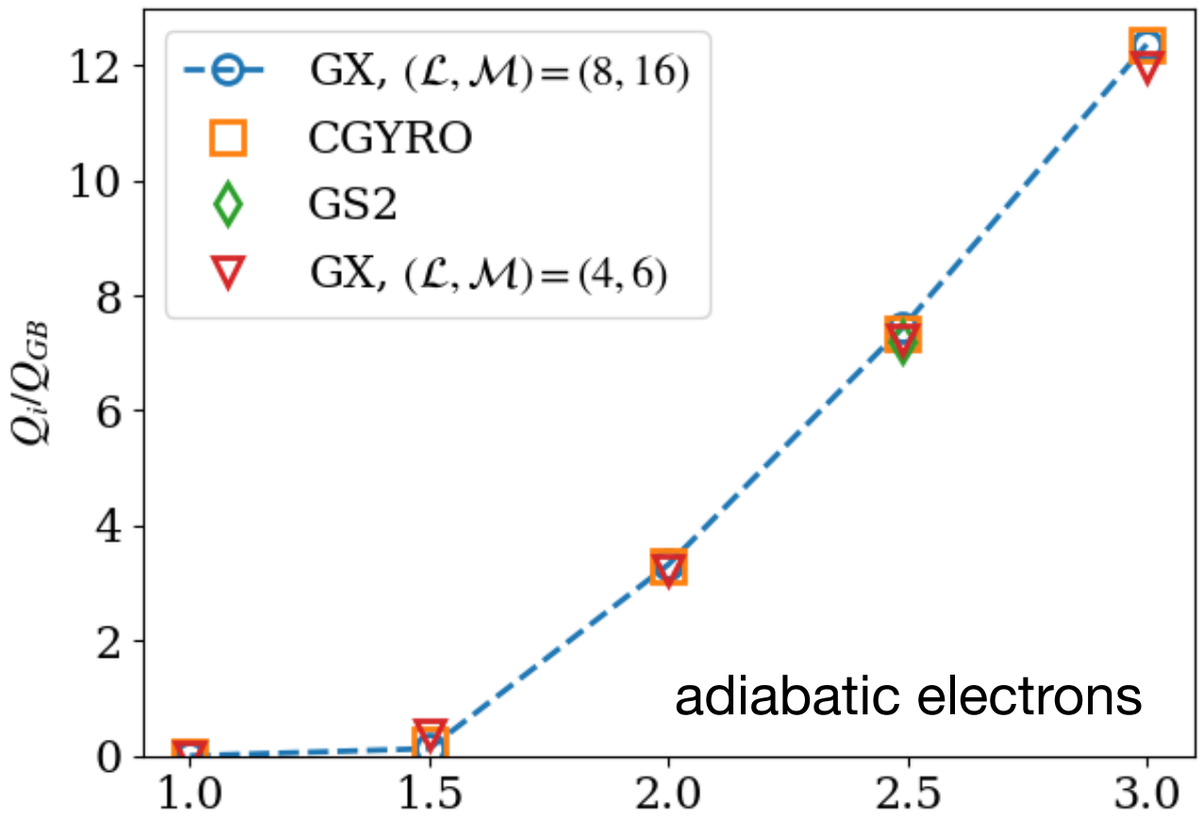
Cyclone base case parameters with Miller geometry



Mandell et al, arXiv:2209.06731 (2022)

Nonlinear tokamak benchmark (Cyclone, adiabatic electrons)





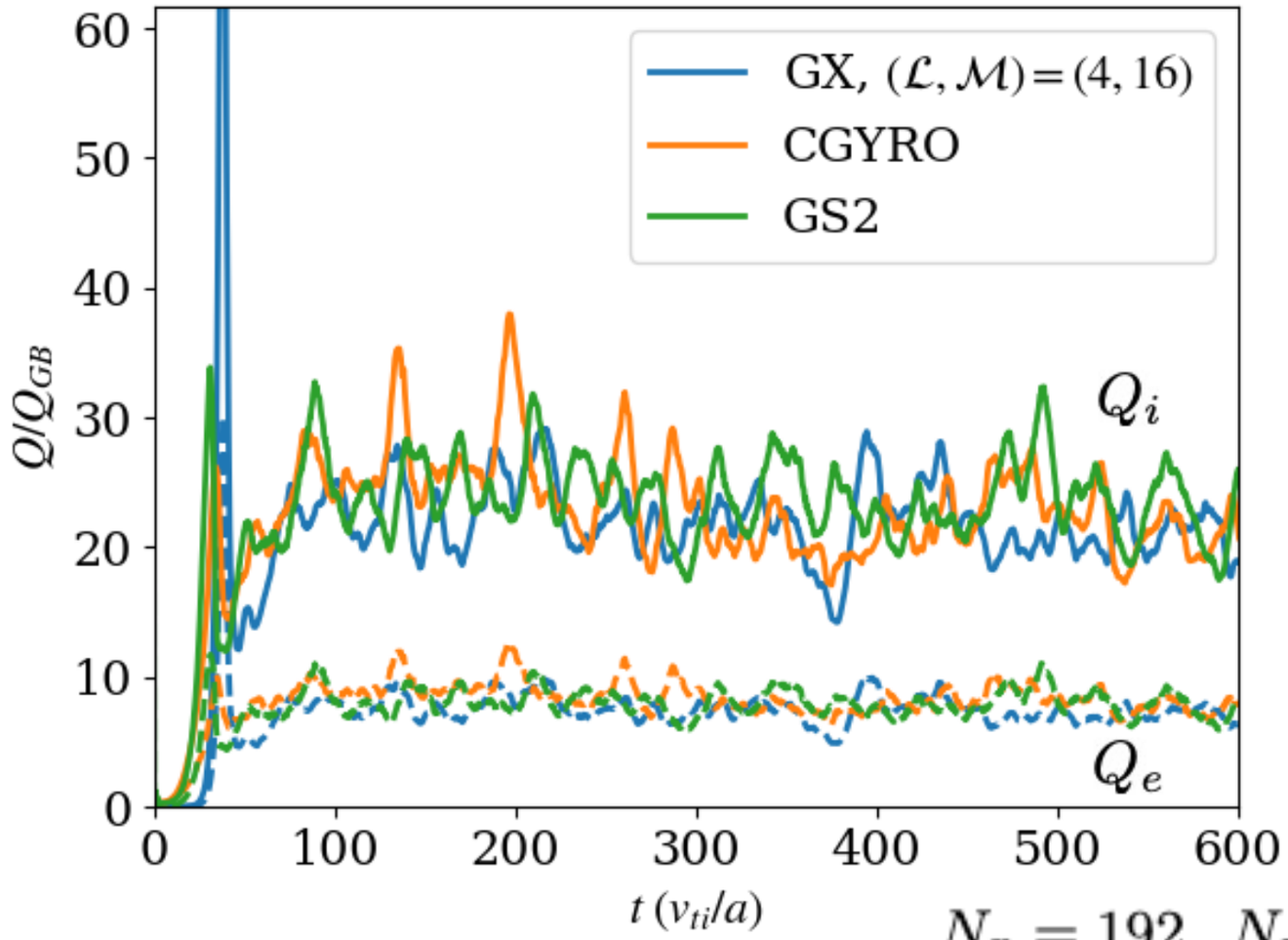
$$N_x = 192,\ N_y = 64,\ N_z = 24$$

$\mathcal{L}$	$\mathcal{M}$	Q_i/Q_{GB}	wallclock (min)	time/timestep (s)
16	32	7.4 ± 0.9	467	0.14
⋆ 8	16	7.3 ± 0.9	61	0.037
6	12	7.5 ± 0.7	31	0.023
4	8	7.4 ± 0.9	9	0.011
⋆ 4	6	7.3 ± 0.8	6	0.0092

For nonlinear cases, Laguerre-Hermite velocity-space convergence is quite remarkable

Mandell et al, arXiv:2209.06731 (2022)

Nonlinear tokamak benchmark (Cyclone, kinetic electrons)



kinetic electrons

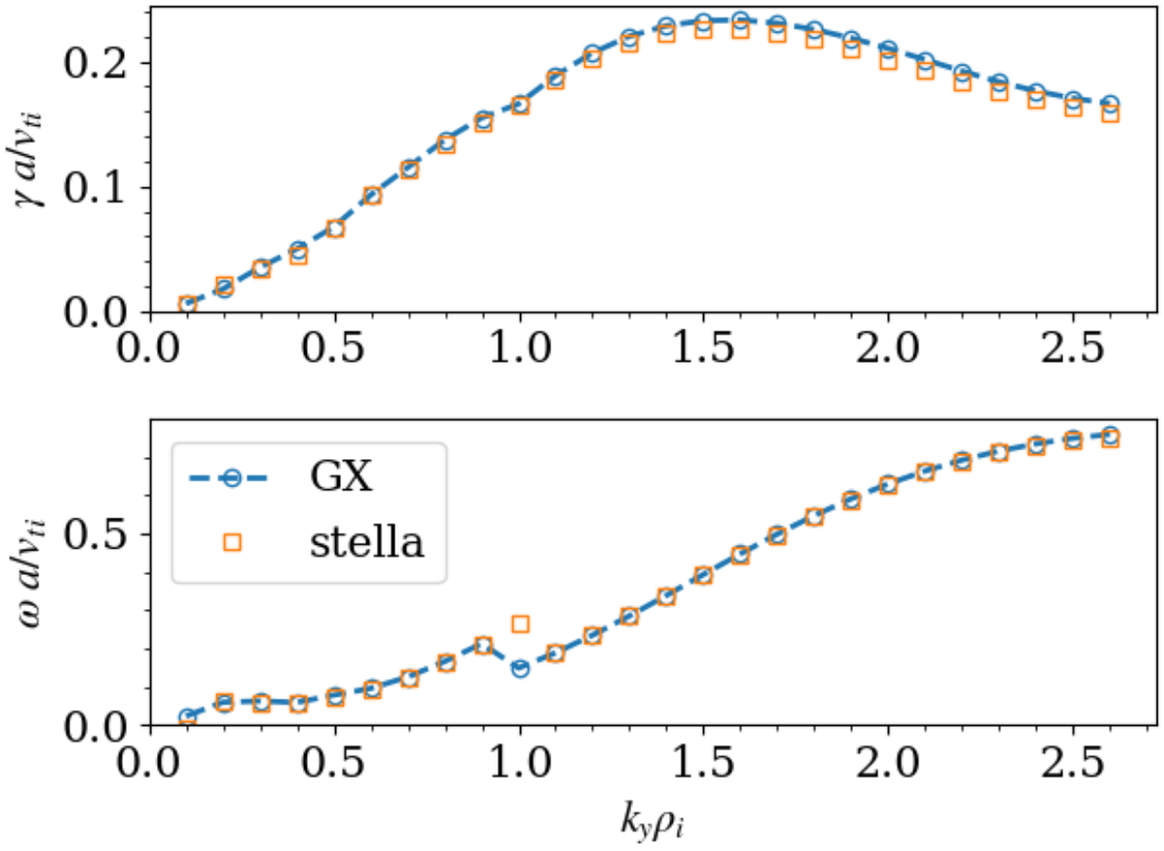
$N_x = 192,\ N_y = 64,\ N_z = 32$

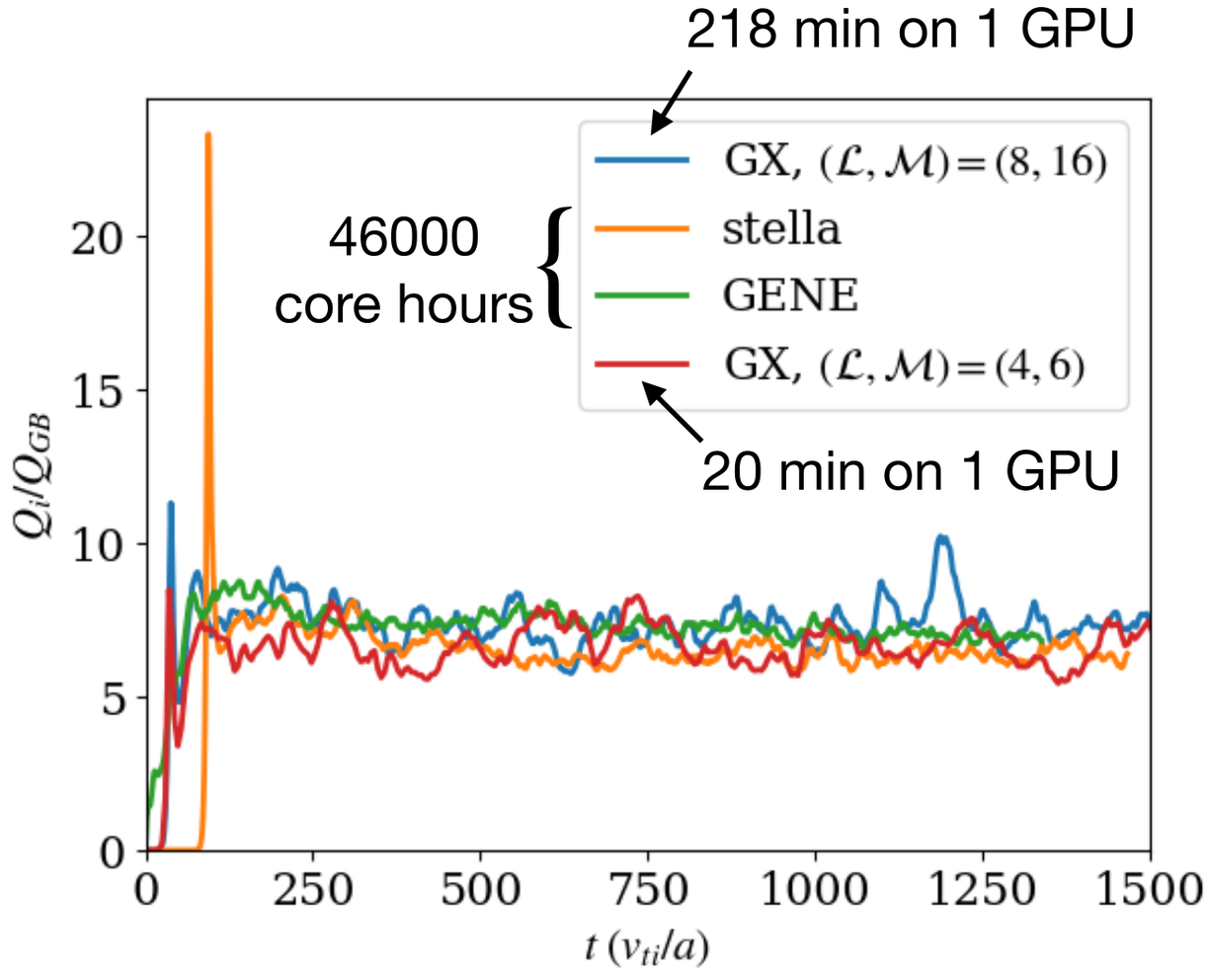
$\mathcal{L}$	$\mathcal{M}$	Q_i/Q_{GB}	Q_e/Q_{GB}	wallclock (min)	time/timestep (s)	# of GPUs
16	32	25.1 ± 2.3	7.9 ± 0.8	1716	0.086	8
8	32	26.6 ± 4.0	8.5 ± 1.3	904	0.045	8
4	32	24.3 ± 2.4	8.1 ± 0.8	581	0.029	8
8	16	23.6 ± 2.5	7.6 ± 0.8	383	0.038	4
$\star$ 4	16	22.4 ± 3.1	7.6 ± 1.0	218	0.022	4

Mandell et al, arXiv:2209.06731 (2022)

Stellarator benchmarks: W7-X (adiabatic electrons)

W7-X “bean” flux-tube stellarator benchmark
(stella and GENE data from Gonzalez et al, 2021)

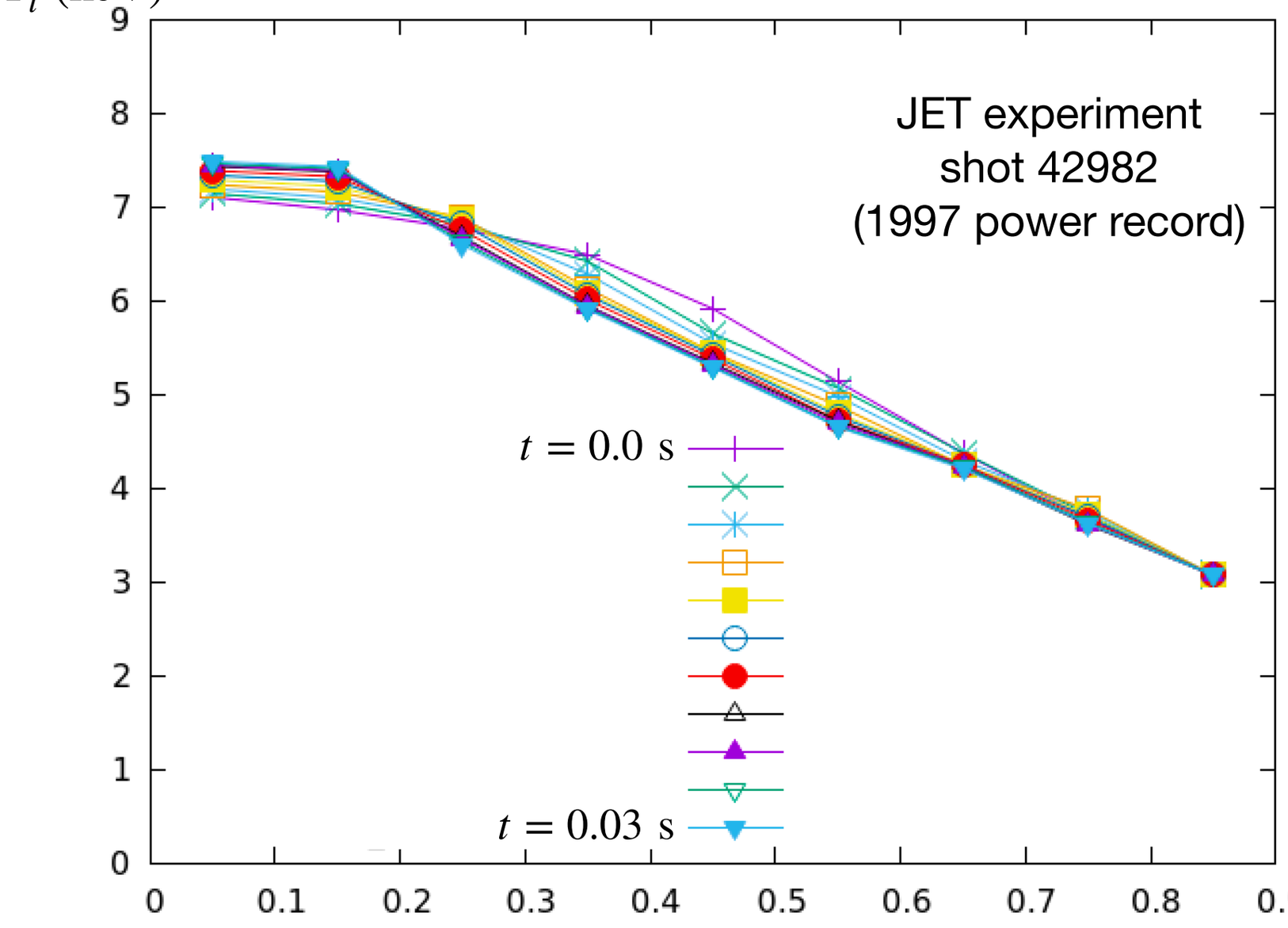




Mandell et al, arXiv:2209.06731 (2022)

Preliminary GX + Trinity simulation

- The speed of GX makes once-daunting coupled turbulence-transport simulations manageable



8 radial cells

16 GX instances (2 per cell for a Jacobian calculation)

10 timesteps, 18 total evaluations (including intermediate Newton iterations)

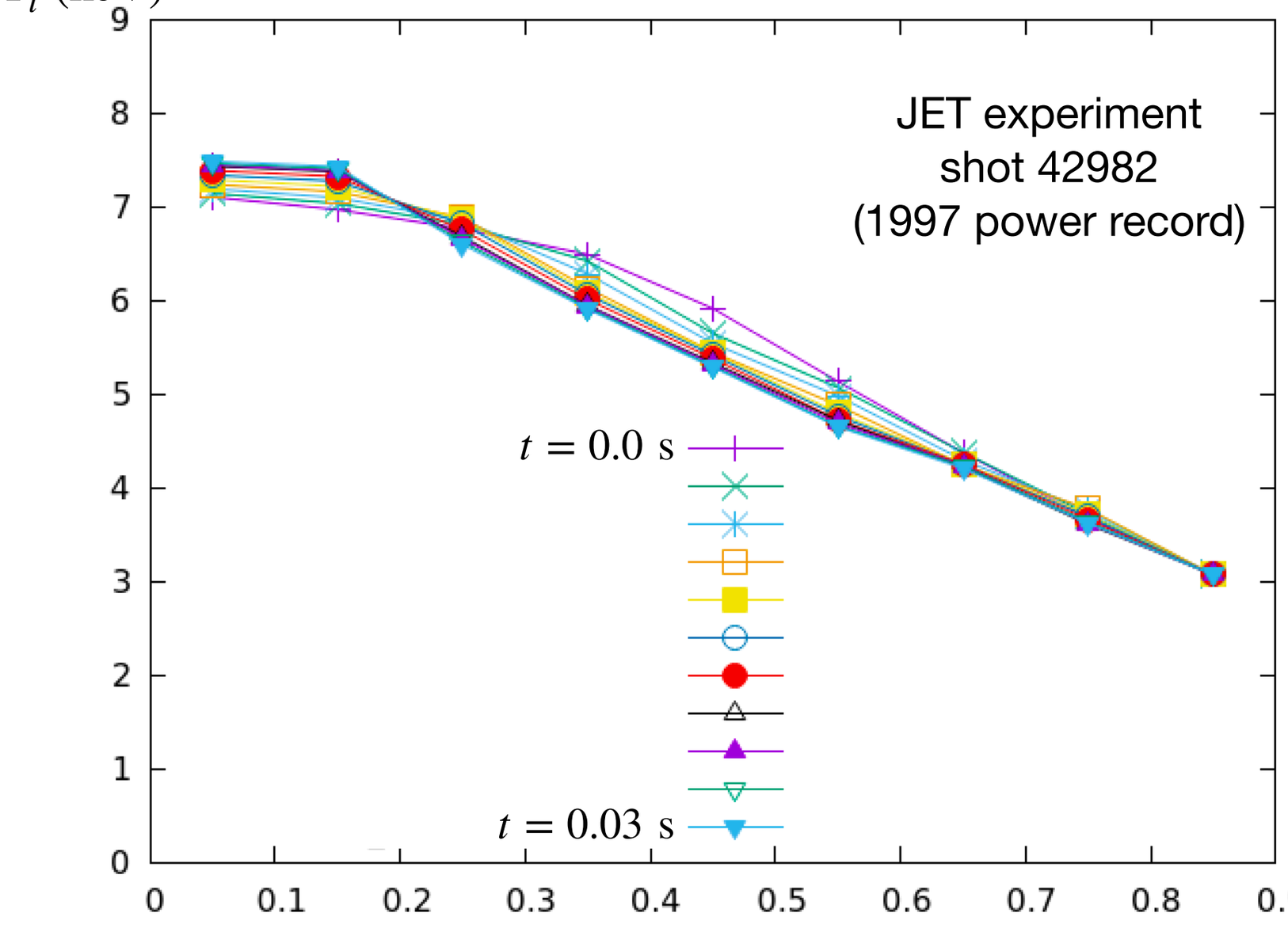
18 x 16 = 288 nonlinear turbulence evaluations

~2 hours total on 16 GPUs (V100)
(would be ~70% faster on A100)

(Might have taken CGYRO+Trinity ~12 hours on ~4000 cores)

Preliminary GX + Trinity simulation

- The speed of GX makes once-daunting coupled turbulence-transport simulations manageable



8 radial cells

16 GX instances (2 per cell for a Jacobian calculation)

10 timesteps, 18 total evaluations (including intermediate Newton iterations)

18 x 16 = 288 nonlinear turbulence evaluations

~2 hours total on 16 GPUs (V100)
(would be ~70% faster on A100)

(Might have taken CGYRO+Trinity ~12 hours on ~4000 cores)

GX+Trinity transport model can be used to predict and optimize core profiles for future fusion reactor designs!

Summary of GX + Trinity work, and future opportunities

- GX is a **fast, flexible-fidelity turbulence code** for the core of fusion plasmas
  - Takes advantage of advances in both **algorithms (spectral) and hardware (GPU)**
  - Ideal tool for scoping/design calculations in large parameter spaces
  - Ongoing work: Investigating algorithms/models to reduce timestep restriction from fast electron dynamics
    - Implicit-explicit schemes, analytically removing fast electron motion via mass-ratio expansion, bounce/flux-surface averaging, etc
- Coupling GX to Trinity enables **transport time-scale simulation for reactor scales at manageable cost**
  - Multi-scale algorithm is many orders of magnitude better than brute-force approach
  - Enables high-fidelity core transport calculations in O(1 hour) on O(10) GPUs, allowing use in optimization frameworks
  - Even higher fidelity calculations (e.g. evolving more transport channels, higher resolution, more species, etc) in O(1 day) on O(100) of GPUs

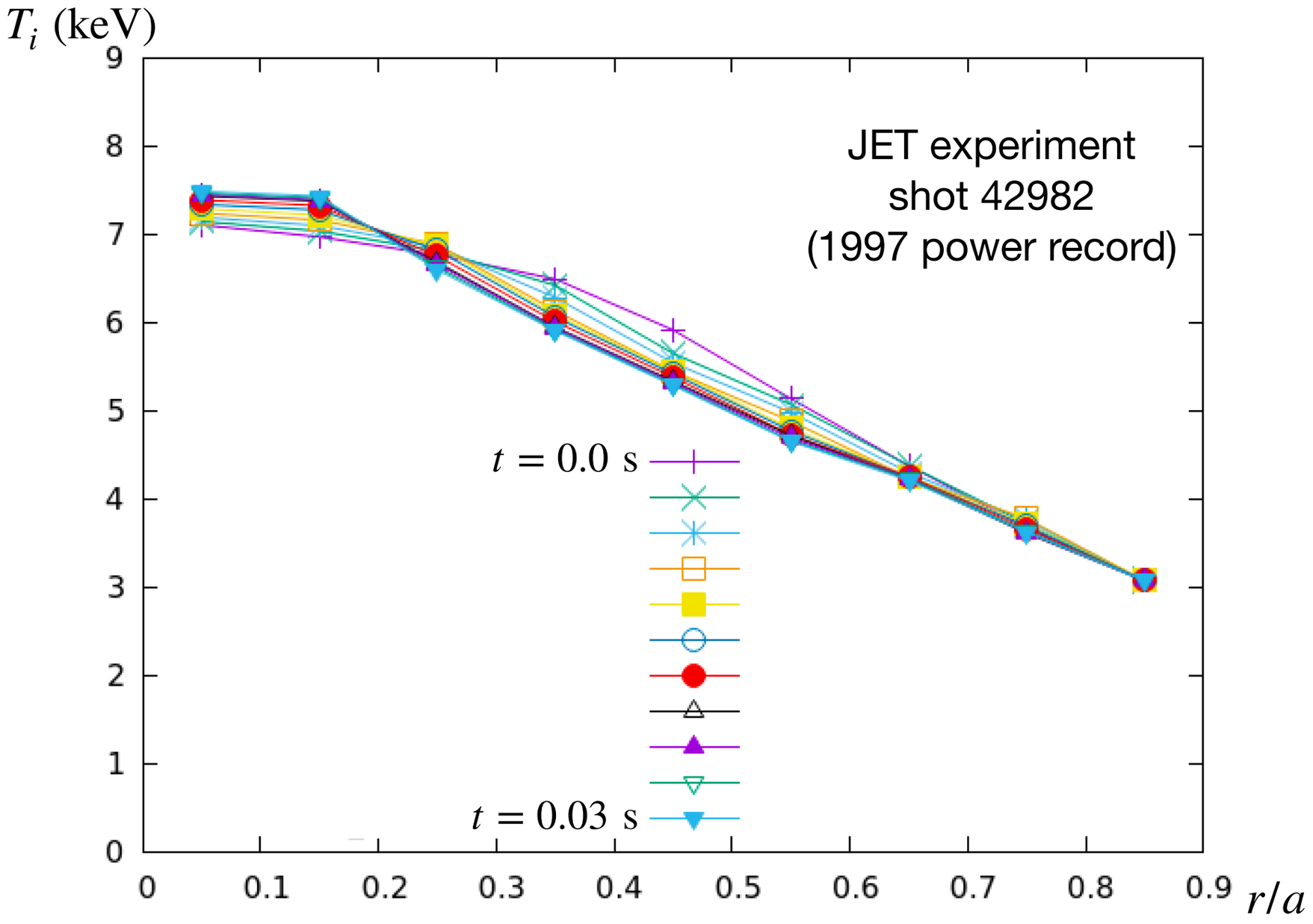
+

Trinity

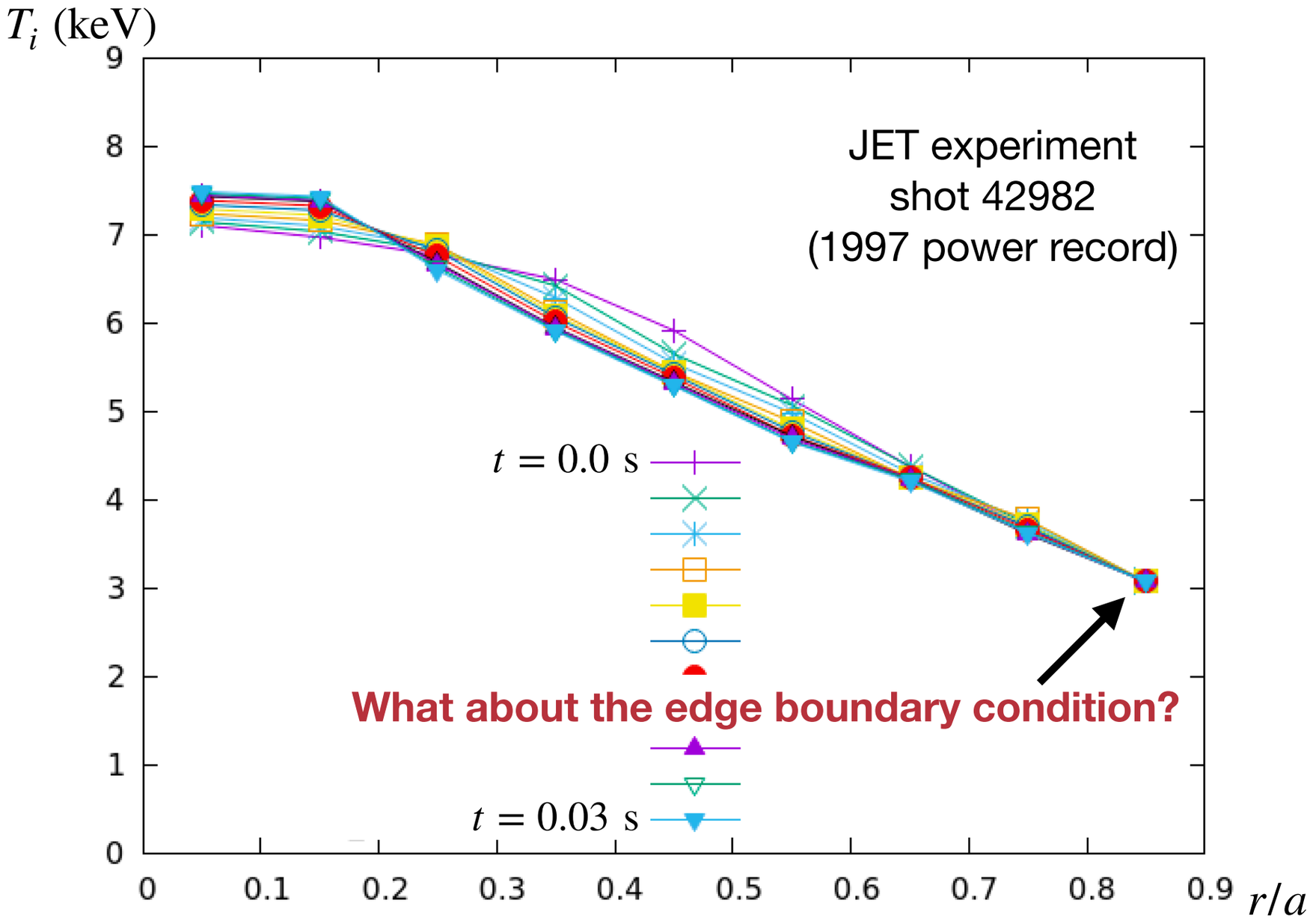
https://gx.readthedocs.io

https://bitbucket.org/gyrokinetics/gx

Thinking on the edge



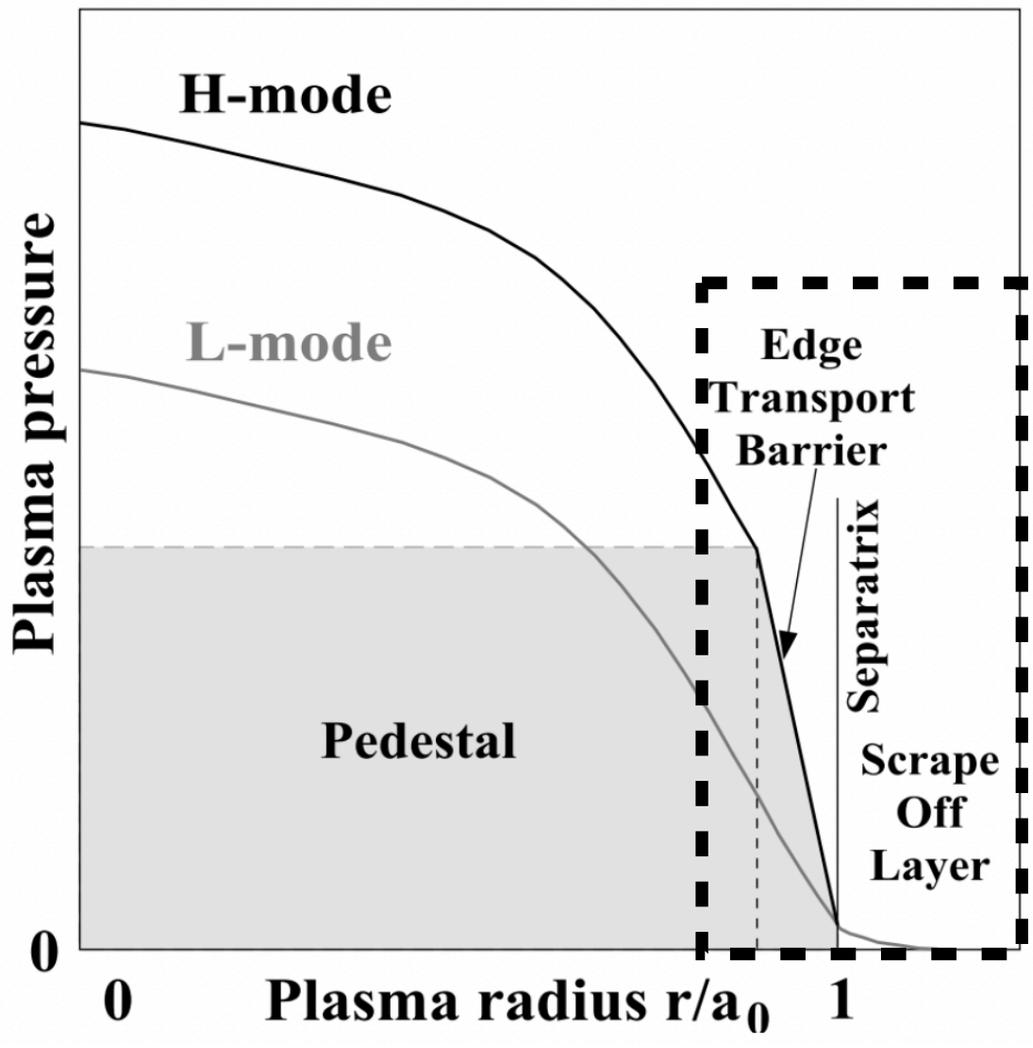
Thinking on the edge



The boundary is crucial to reactor performance

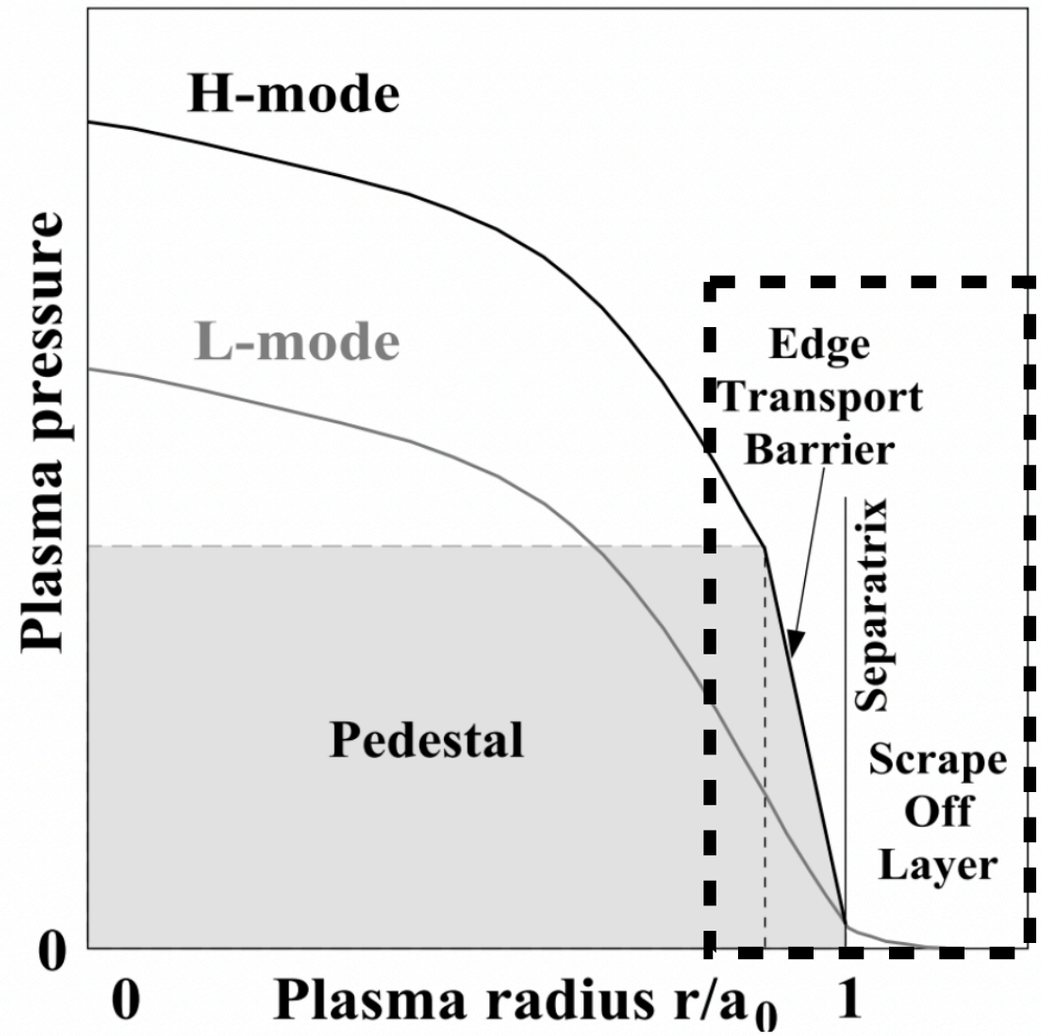
The boundary is crucial to reactor performance

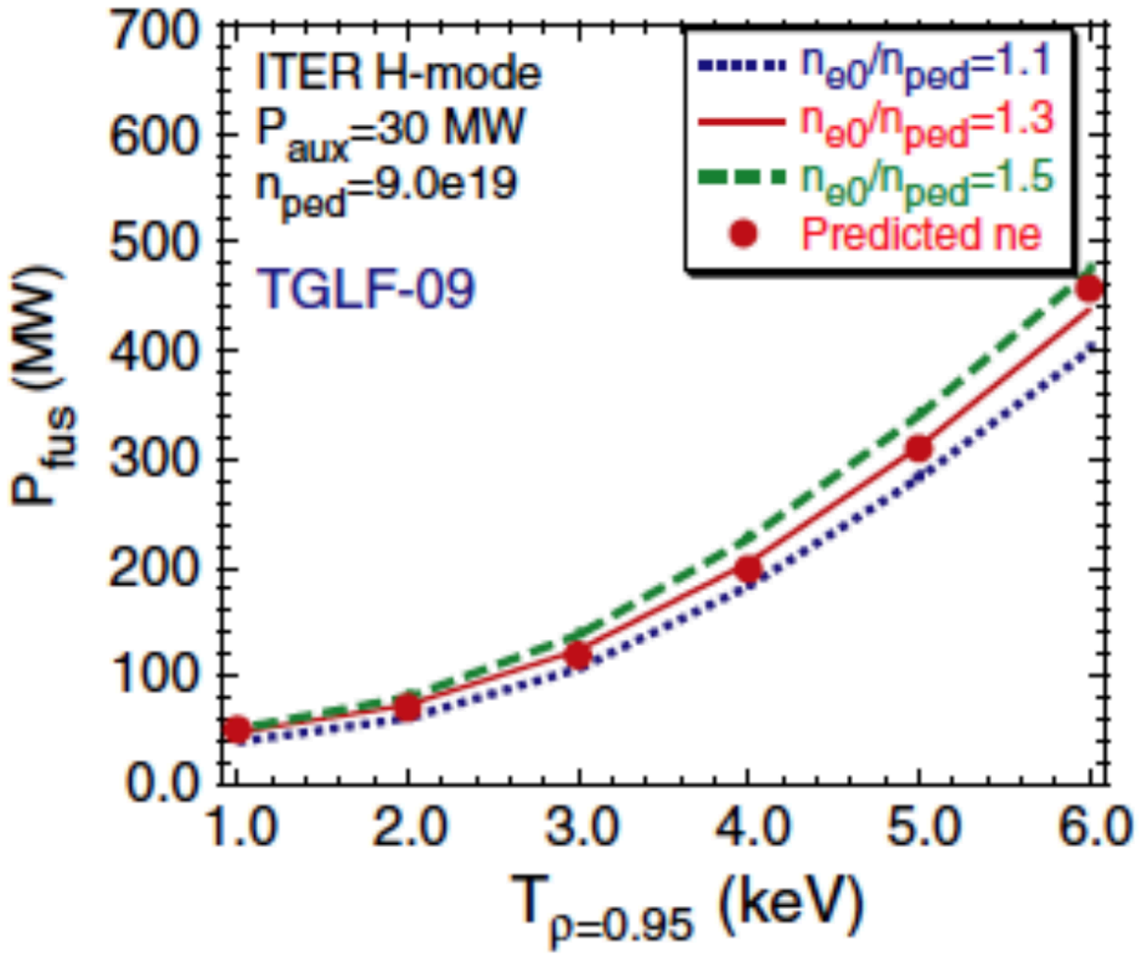
- In today's fusion experiments, achieving good performance usually requires H-mode
  - H-mode occurs when a transport barrier forms at the edge of the core, enabling a steep-gradient region that lifts up the profiles in the core (as if they were standing on a "pedestal")



The boundary is crucial to reactor performance

- In today's fusion experiments, achieving good performance usually requires H-mode
  - H-mode occurs when a transport barrier forms at the edge of the core, enabling a steep-gradient region that lifts up the profiles in the core (as if they were standing on a "pedestal")

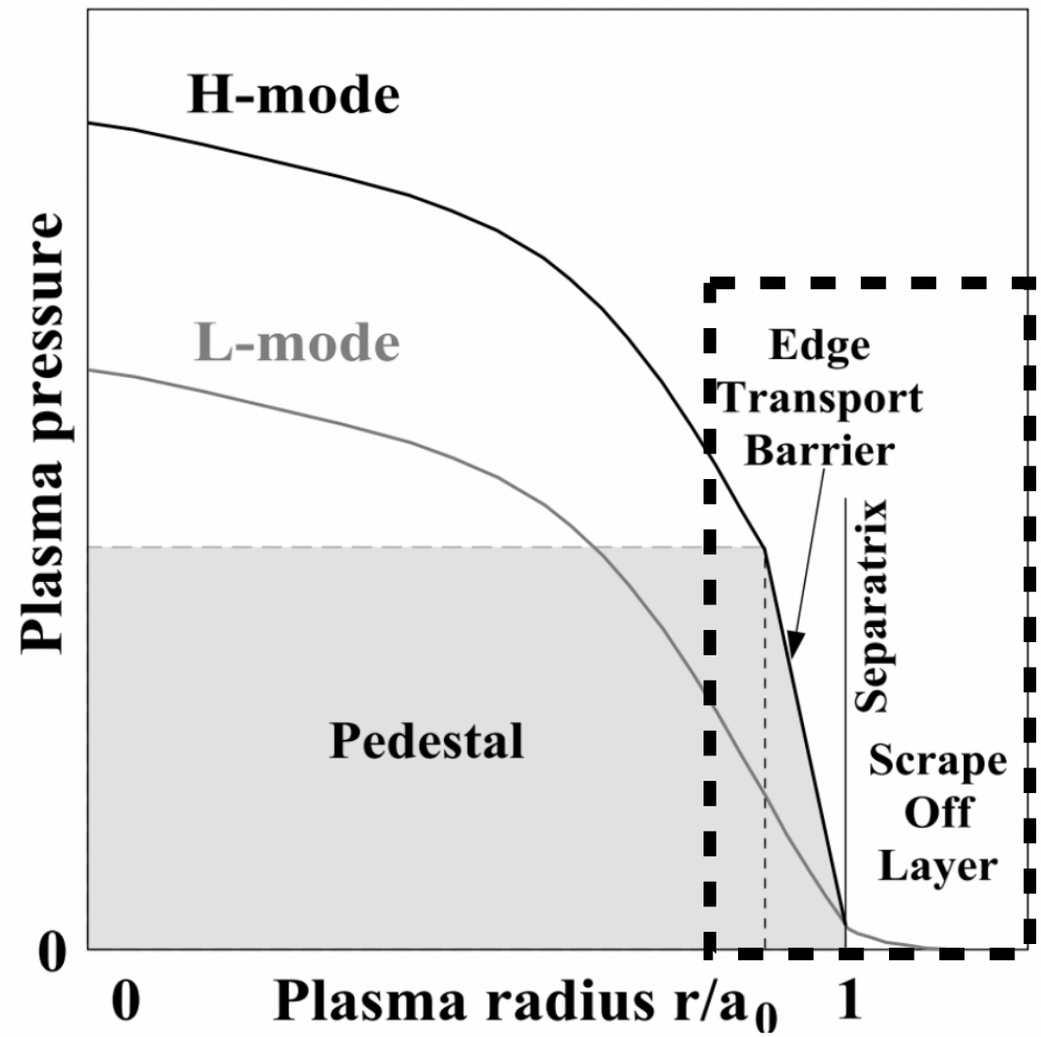


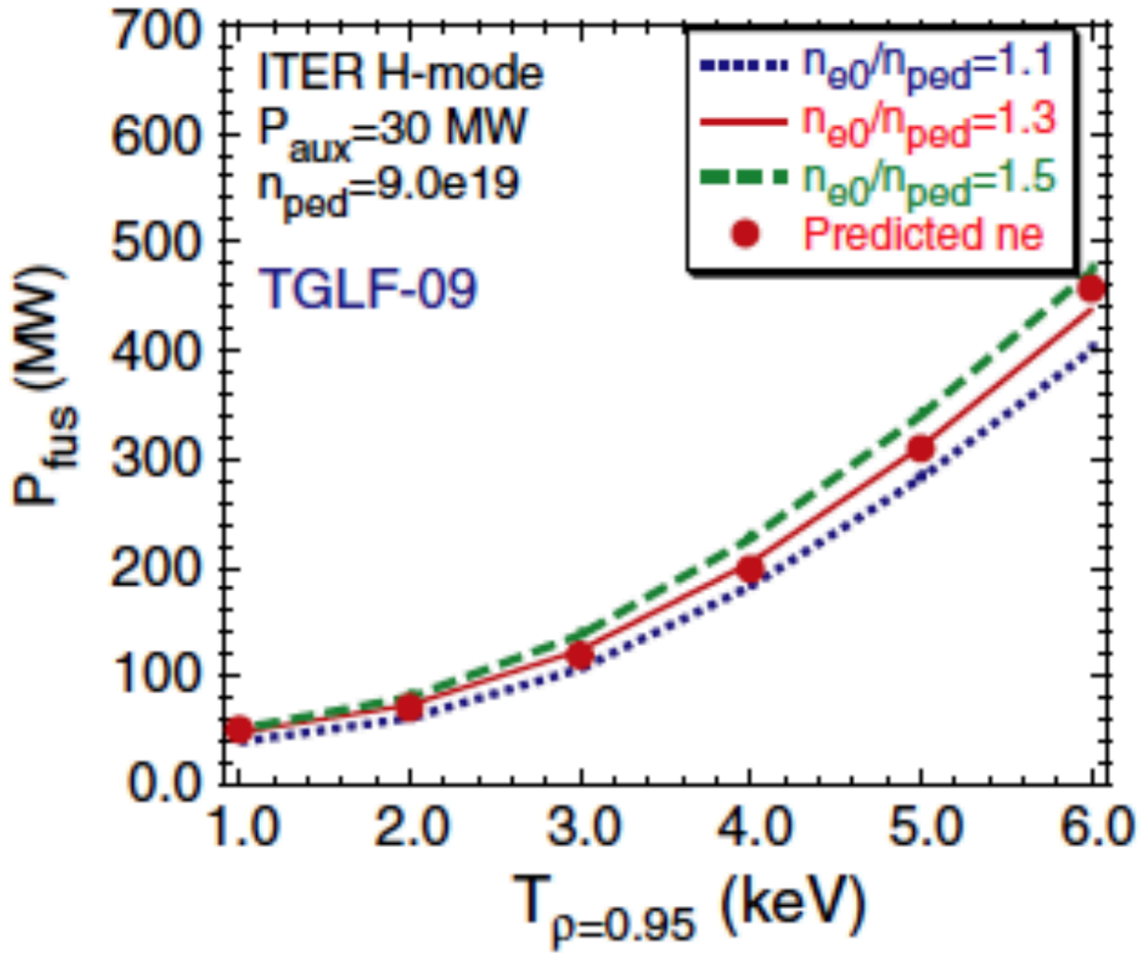


Kinsey et al. Nucl. Fus. 2011

The boundary is crucial to reactor performance

- In today's fusion experiments, achieving good performance usually requires H-mode
  - H-mode occurs when a transport barrier forms at the edge of the core, enabling a steep-gradient region that lifts up the profiles in the core (as if they were standing on a "pedestal")
  - The formation of the H-mode transport barrier (the L-H transition) is still not well-understood

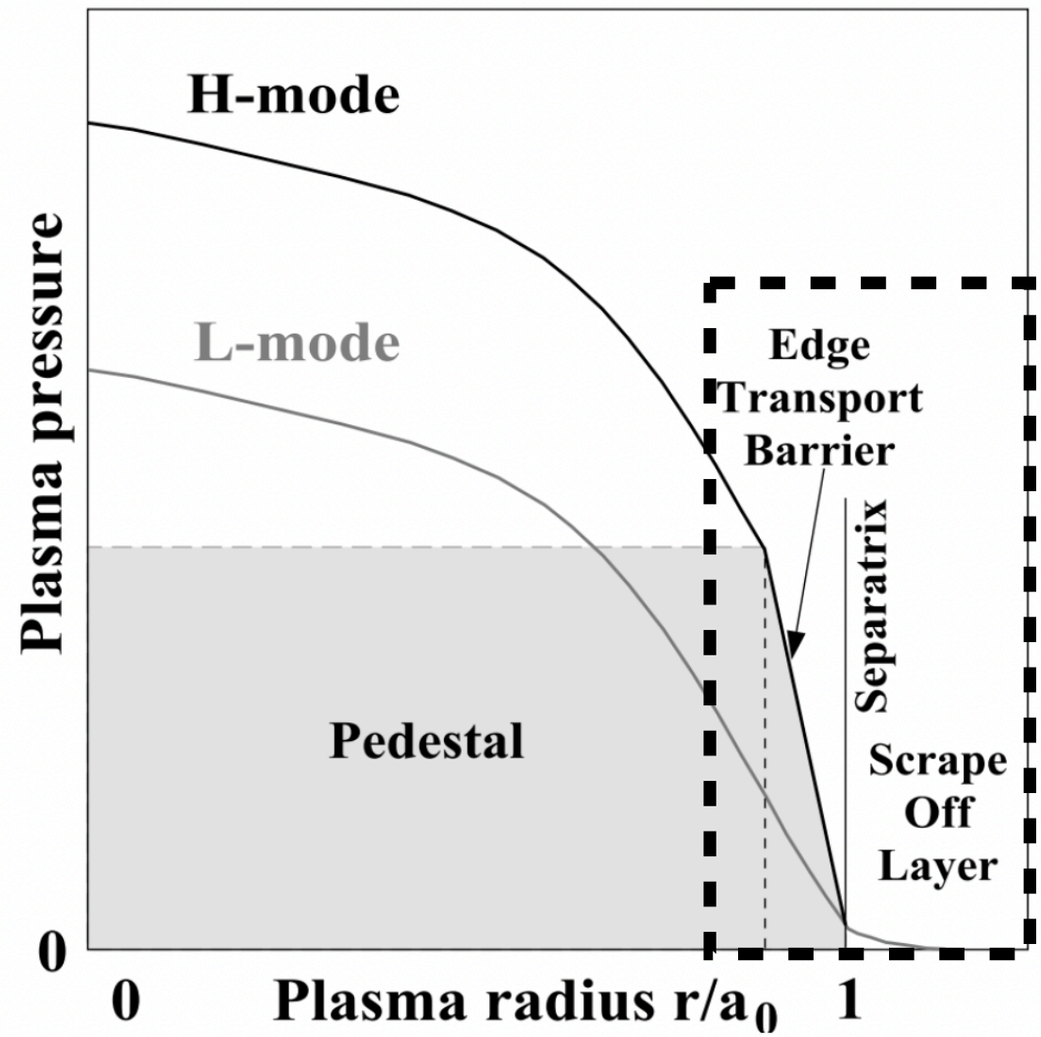


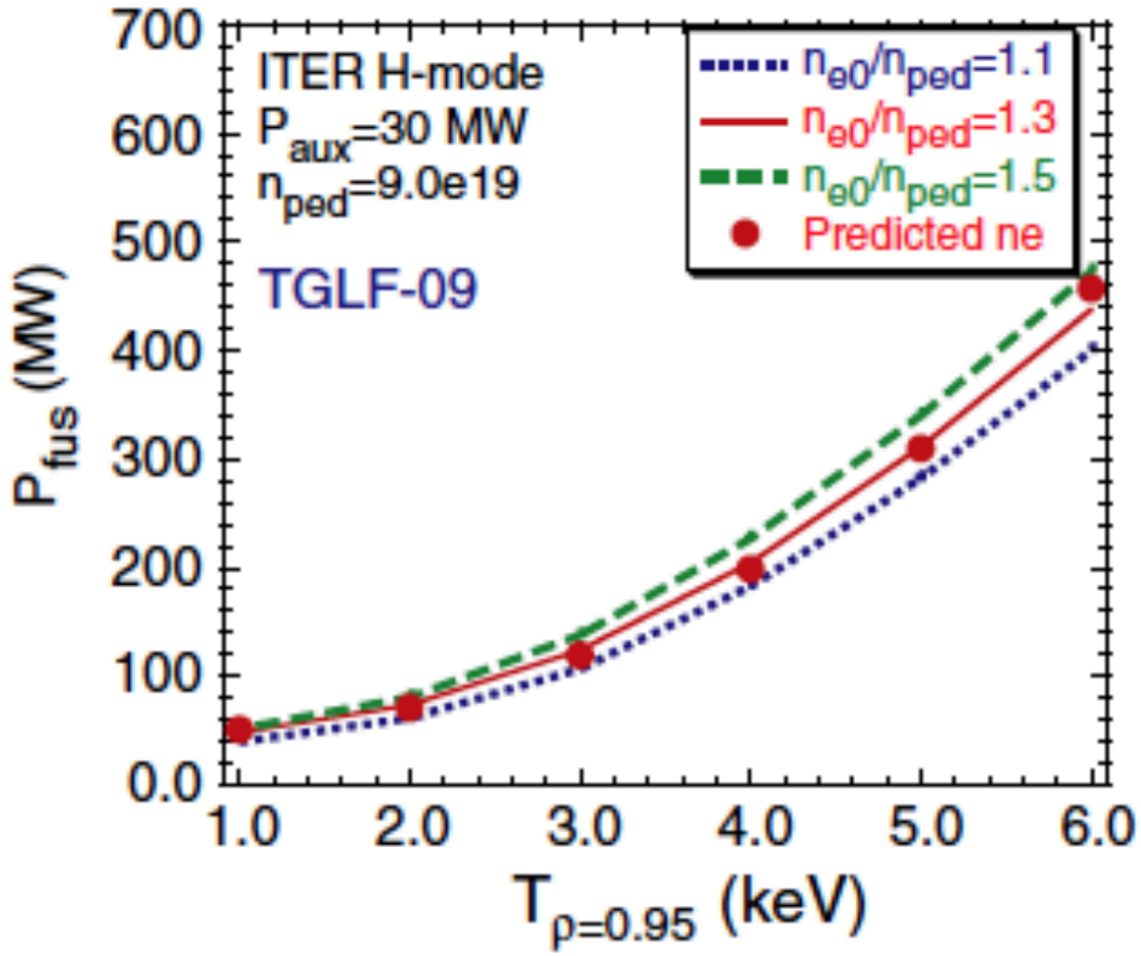


Kinsey et al. Nucl. Fus. 2011

The boundary is crucial to reactor performance

- In today's fusion experiments, achieving good performance usually requires H-mode
  - H-mode occurs when a transport barrier forms at the edge of the core, enabling a steep-gradient region that lifts up the profiles in the core (as if they were standing on a "pedestal")
  - The formation of the H-mode transport barrier (the L-H transition) is still not well-understood
  - In some cases we may want to avoid H-mode, e.g. to avoid ELMs. Need to be able to model/predict this too.

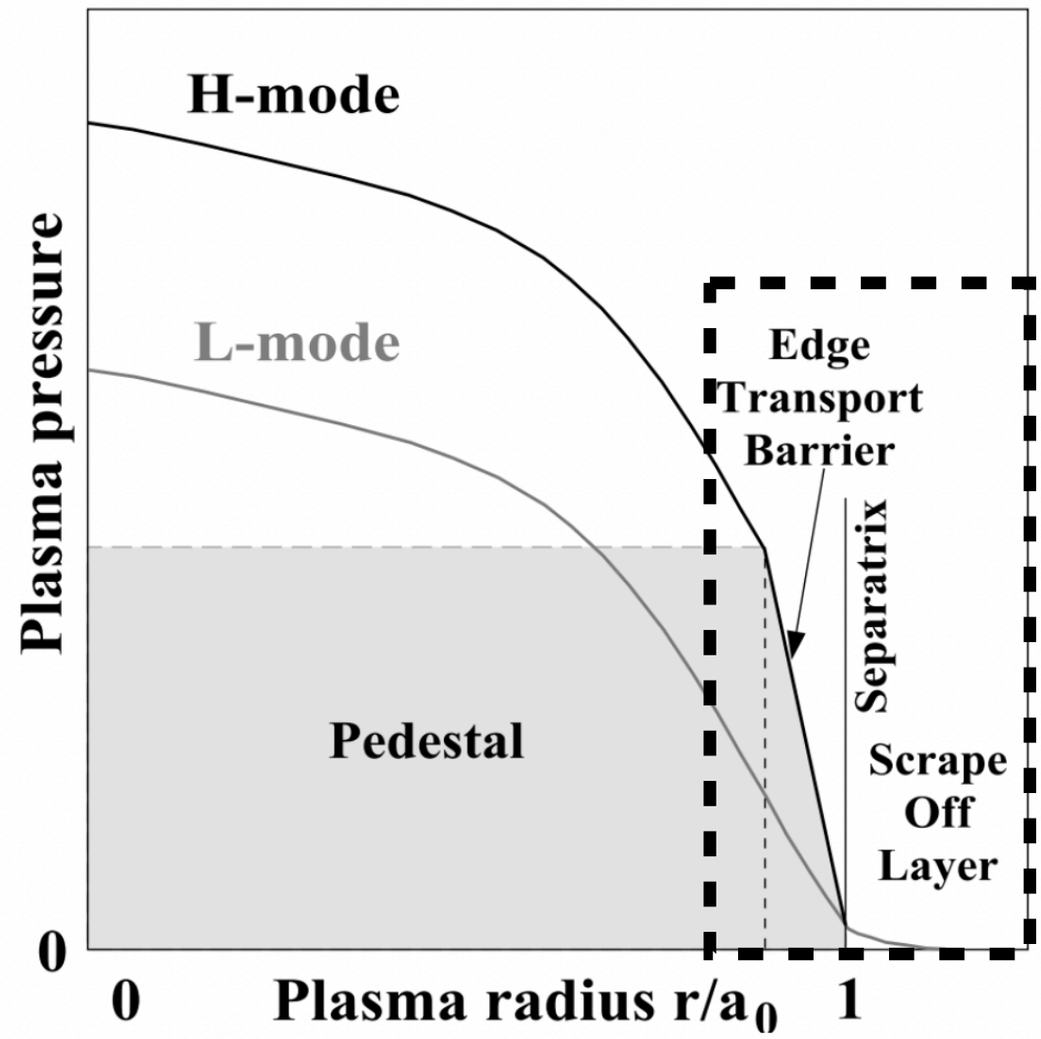


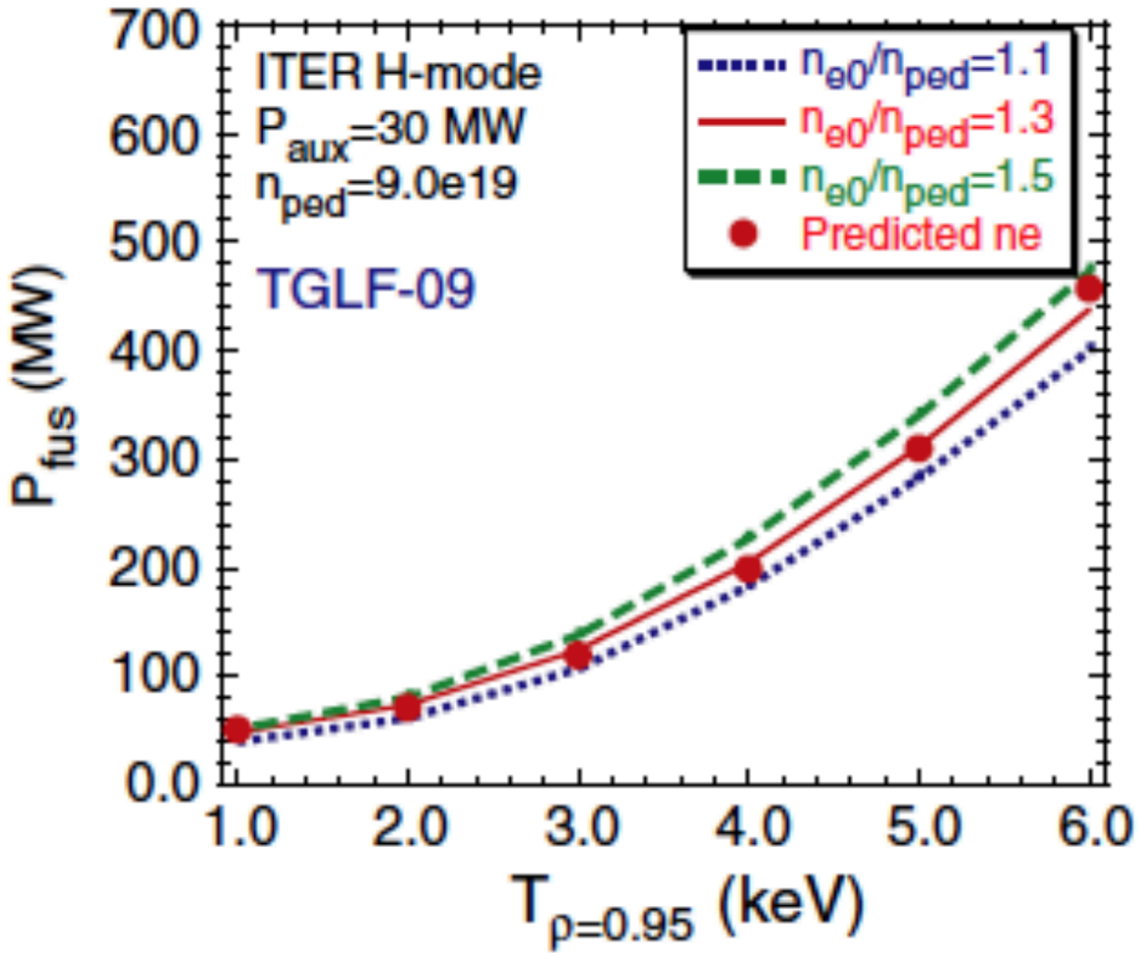


Kinsey et al. Nucl. Fus. 2011

The boundary is crucial to reactor performance

- In today's fusion experiments, achieving good performance usually requires H-mode
  - H-mode occurs when a transport barrier forms at the edge of the core, enabling a steep-gradient region that lifts up the profiles in the core (as if they were standing on a "pedestal")
  - The formation of the H-mode transport barrier (the L-H transition) is still not well-understood
  - In some cases we may want to avoid H-mode, e.g. to avoid ELMs. Need to be able to model/predict this too.
- **Need to be able to confidently predict/optimize edge temperature of reactor designs**





Kinsey et al. Nucl. Fus. 2011

The boundary heat exhaust problem is a potential show-stopper

- Heat exhausted in boundary could damage divertor plates if heat flux width is too narrow
    - Major problem at reactor scale
- Turbulence in the boundary could help by broadening the width of the heat flux channel
- Several of the current state-of-the-art boundary simulation codes (like SOLPS) are 2D fluid codes with a crude empirical diffusion model to represent turbulence
    - Good for interpretive modeling in support of current experiments, but not predicting future experiments, especially hotter reactor-scale devices where fluid approximations will break down
- **Need first-principles models to model/optimize turbulent broadening of boundary heat flux**

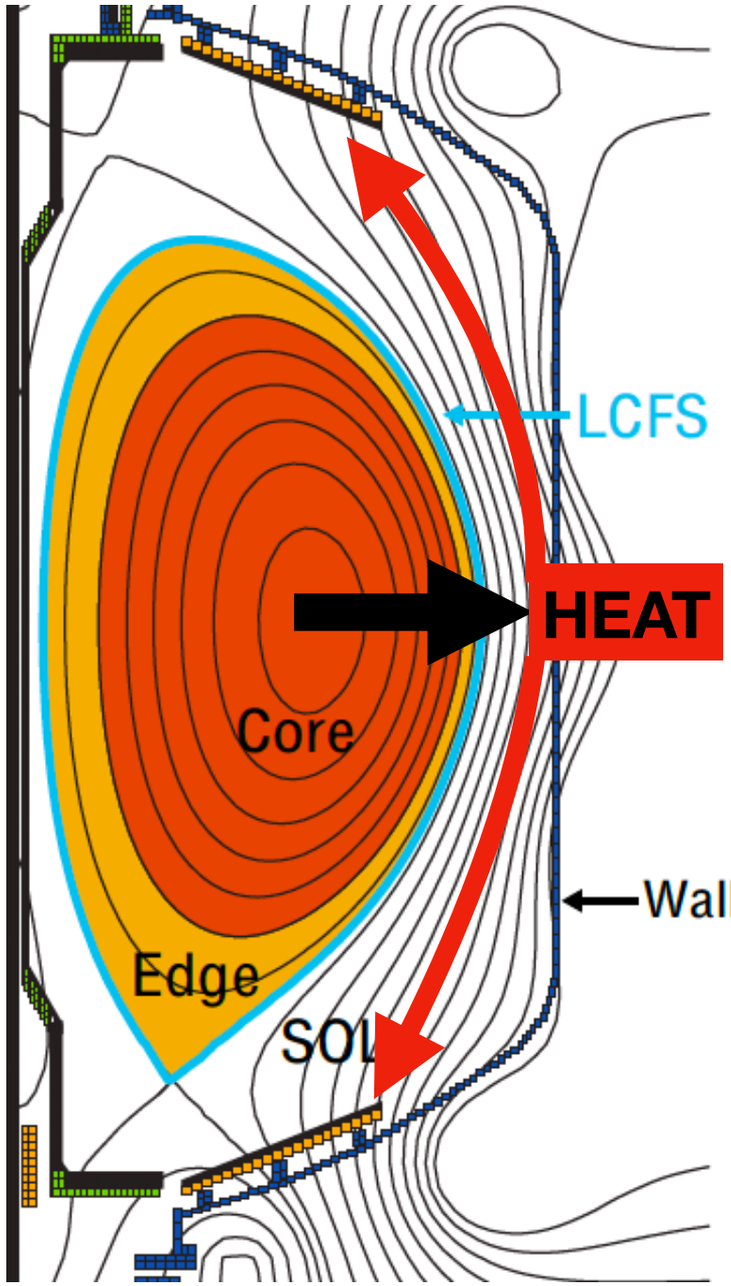


Figure adapted from Stoltzfus-Dueck (2009)

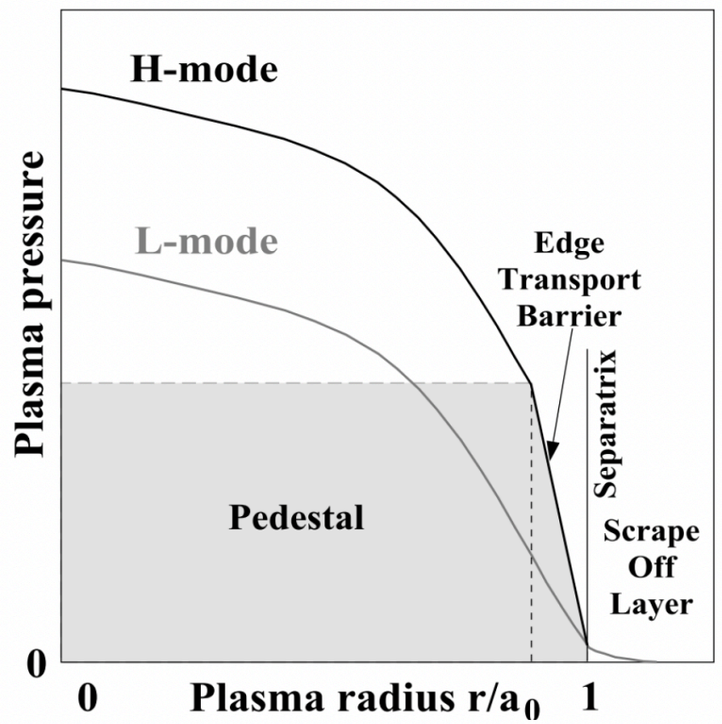
Modeling the boundary presents new challenges

- Because of the steep gradients in the boundary, there is not much scale separation between turbulence and transport
  - Can't use a multi-scale approach like we did in the core
- Additional complications: boundary plasma has large-amplitude fluctuations, open field lines, plasma-wall interactions, X-point geometry, neutral/atomic physics, transition from kinetic to fluid regimes → **need specialized gyrokinetic codes for the boundary**
- Electromagnetic effects also important, especially in steep-gradient region, but including self-consistent magnetic fluctuations very challenging → until recently, gyrokinetic simulations on open field lines neglected magnetic fluctuations

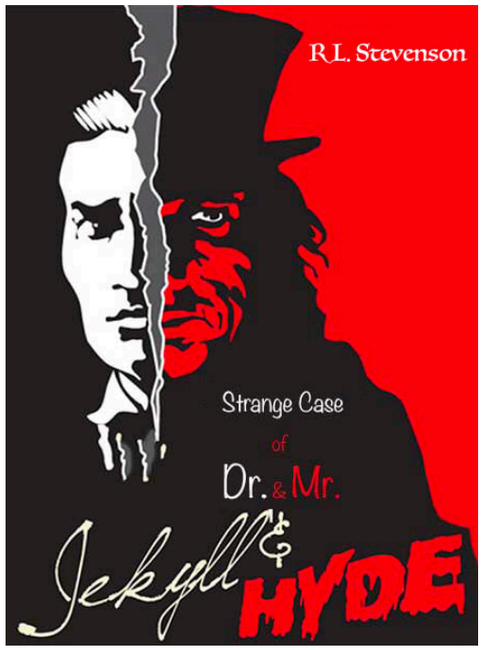
Boundary:

$L_{\text{turb}} \sim \rho_i \sim 0.5$ mm $L_{\text{transp}} \sim L_p \sim 1$ cm

$\tau_{\text{turb}} \sim 1\ \mu$s $\tau_{\text{transp}} \sim \tau_{\text{ELM}} \sim 0.01 - 0.1$ s



Gkeyll: a gyrokinetic model for the boundary



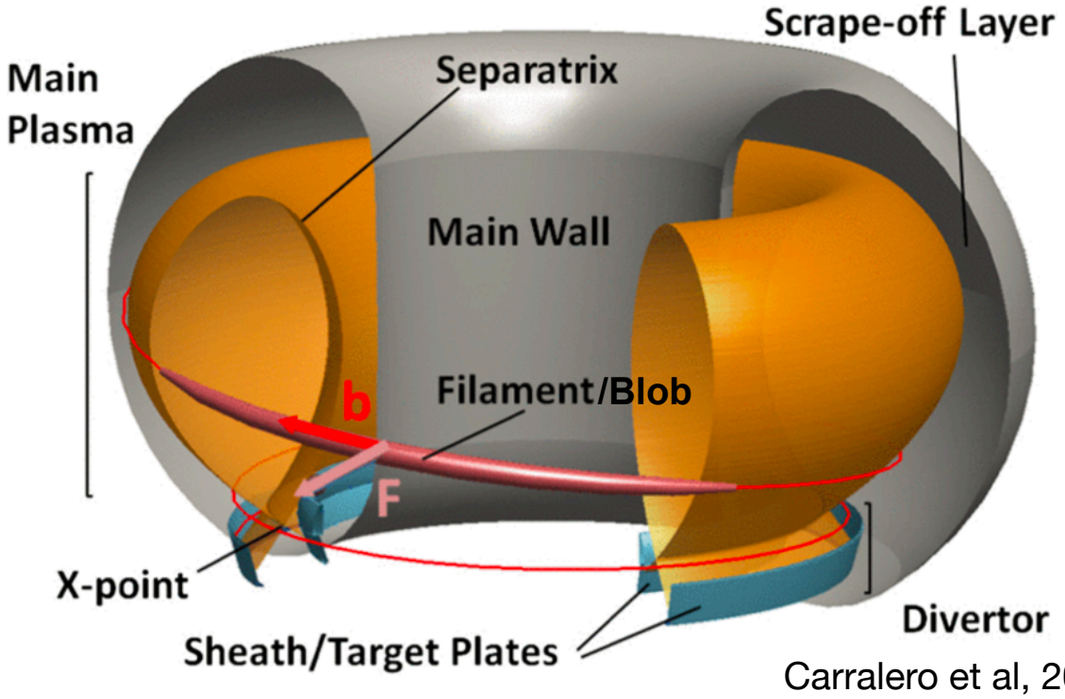
https://github.com/ammarhakim/gkyl/

https://gkyl.readthedocs.io

https://gkyl.readthedocs.io/en/latest/gkyl/pubs.html

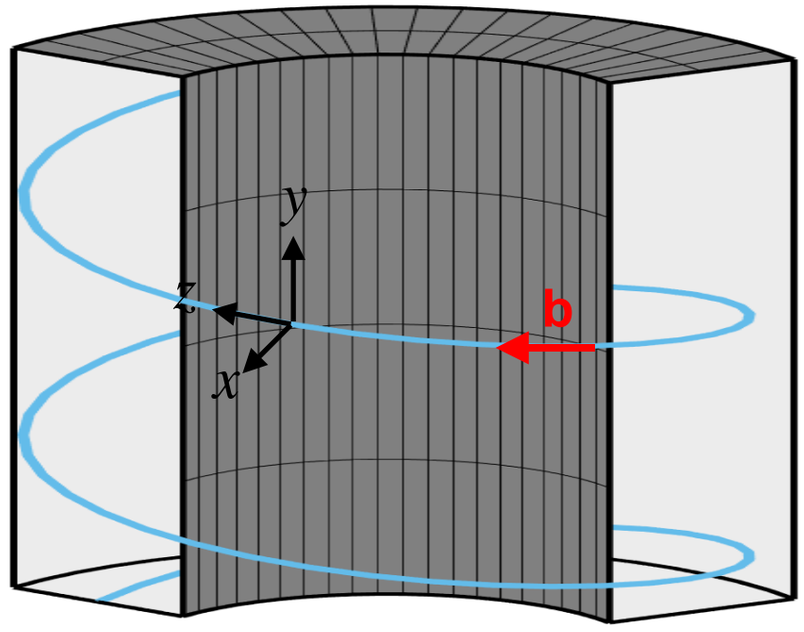
- Gkeyll was first successful continuum (i.e. not particle-in-cell) GK code on open field lines
  - Specialized to handle large-amplitude fluctuations, plasma interaction with walls, etc
- We developed an energy-conserving discontinuous Galerkin (DG) discretization scheme for Hamiltonian systems (like GK)
  - DG methods are highly local, highly parallelizable, allow high-order accuracy, and enforce local conservation laws
  - Hakim, Hammett, Shi, Mandell, arXiv:1908.01814 (2019)
- Developed a novel scheme to add magnetic fluctuations to Gkeyll, creating a first-of-a-kind electromagnetic gyrokinetic model on open field lines
  - **N. R. Mandell et al, JPP 2020**
  - **A. Hakim, N. R. Mandell, et al, PoP 2020**
  - **N. R. Mandell, Princeton Ph.D. thesis 2021 (arXiv:2103.16062)**
- **Key advance:** Numerical scheme designed to avoid troublesome numerical headaches found in many EMGK codes for 20+ years (Ampere cancellation)
- **Key advance:** Developed boundary conditions to stably model open-field-line effects on magnetic fluctuations, which can modify electromagnetic instabilities (this caused problems in previous efforts)

Modeling the NSTX SOL with Gkeyll



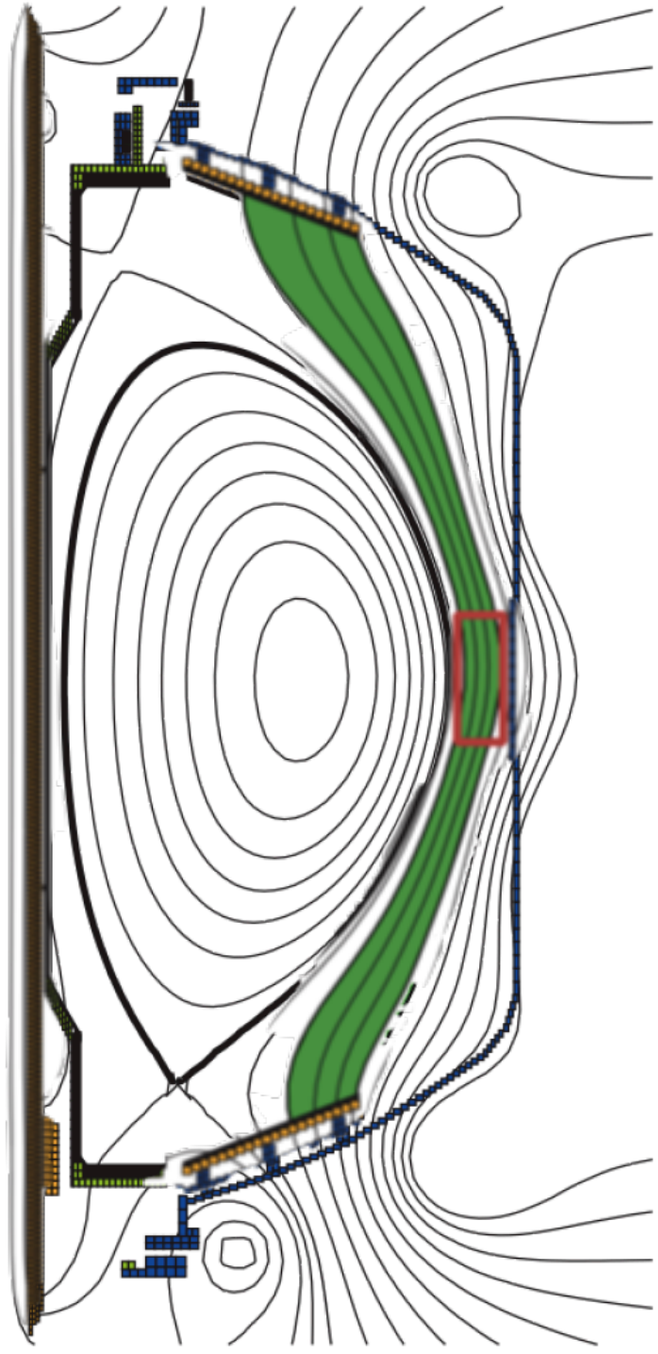
Carralero et al, 2015

≈

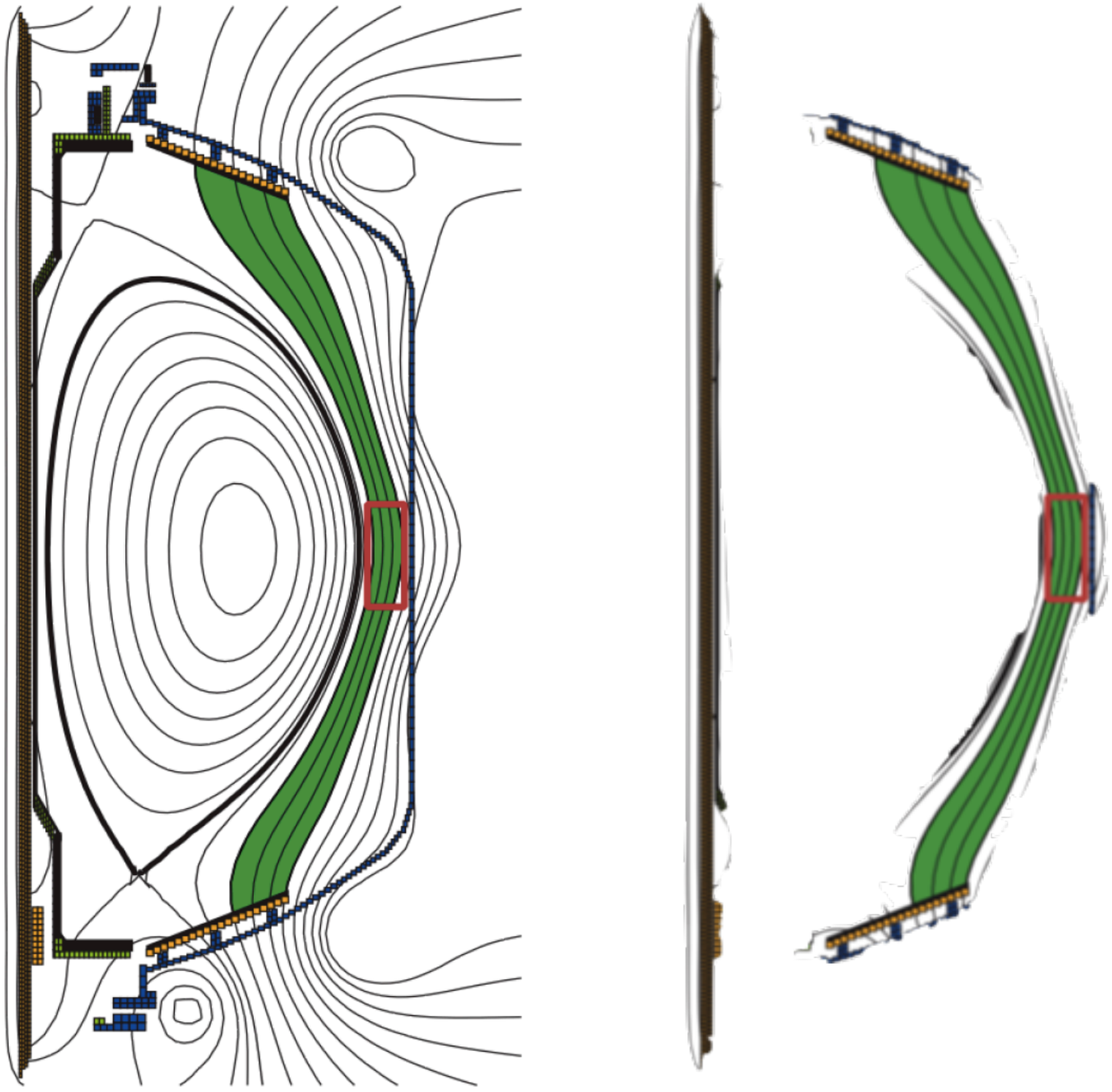


- For now, we take a simple helical model of NSTX SOL
  - Simple magnetized torus (SMT) geometry (like Helimak or TORPEX)
  - Field-aligned simulation domain that follows field lines from bottom divertor plate, around the torus, to the top divertor plate
  - All bad curvature interchange instability, blob dynamics
  - Parameters from NSTX H-mode SOL

Modeling the NSTX SOL with Gkeyll

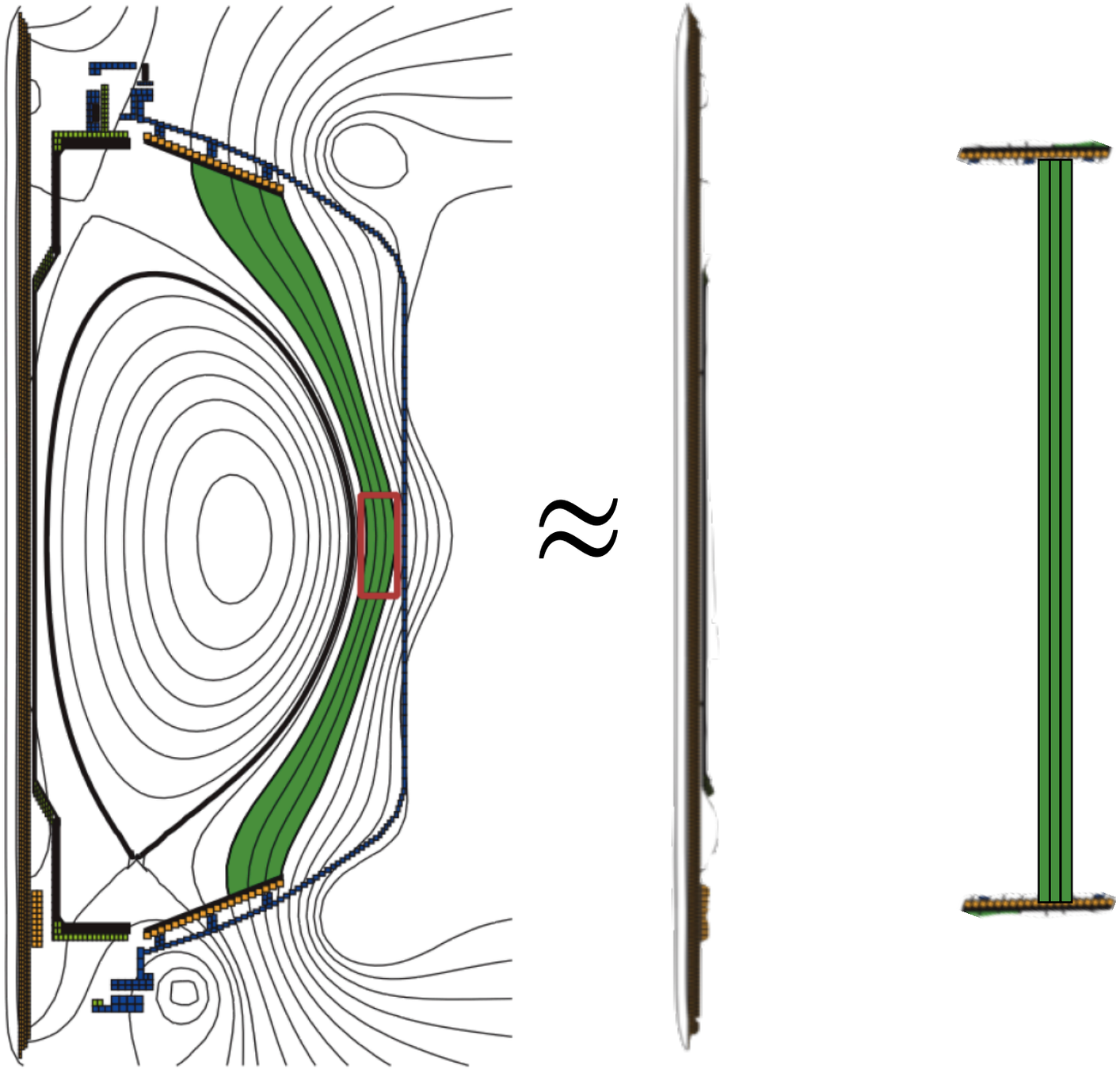


Modeling the NSTX SOL with Gkeyll



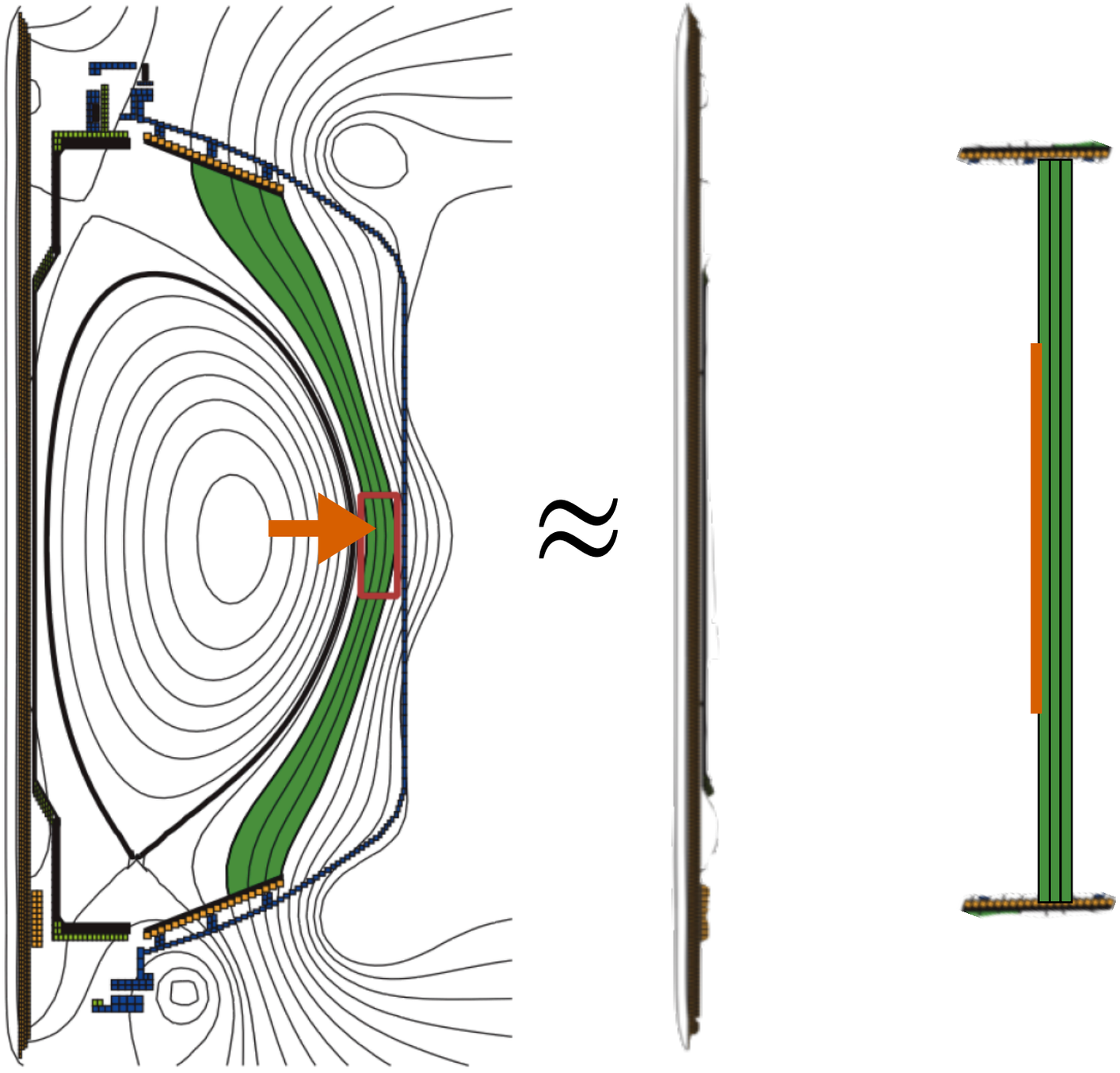
- Open-field-line region only

Modeling the NSTX SOL with Gkeyll



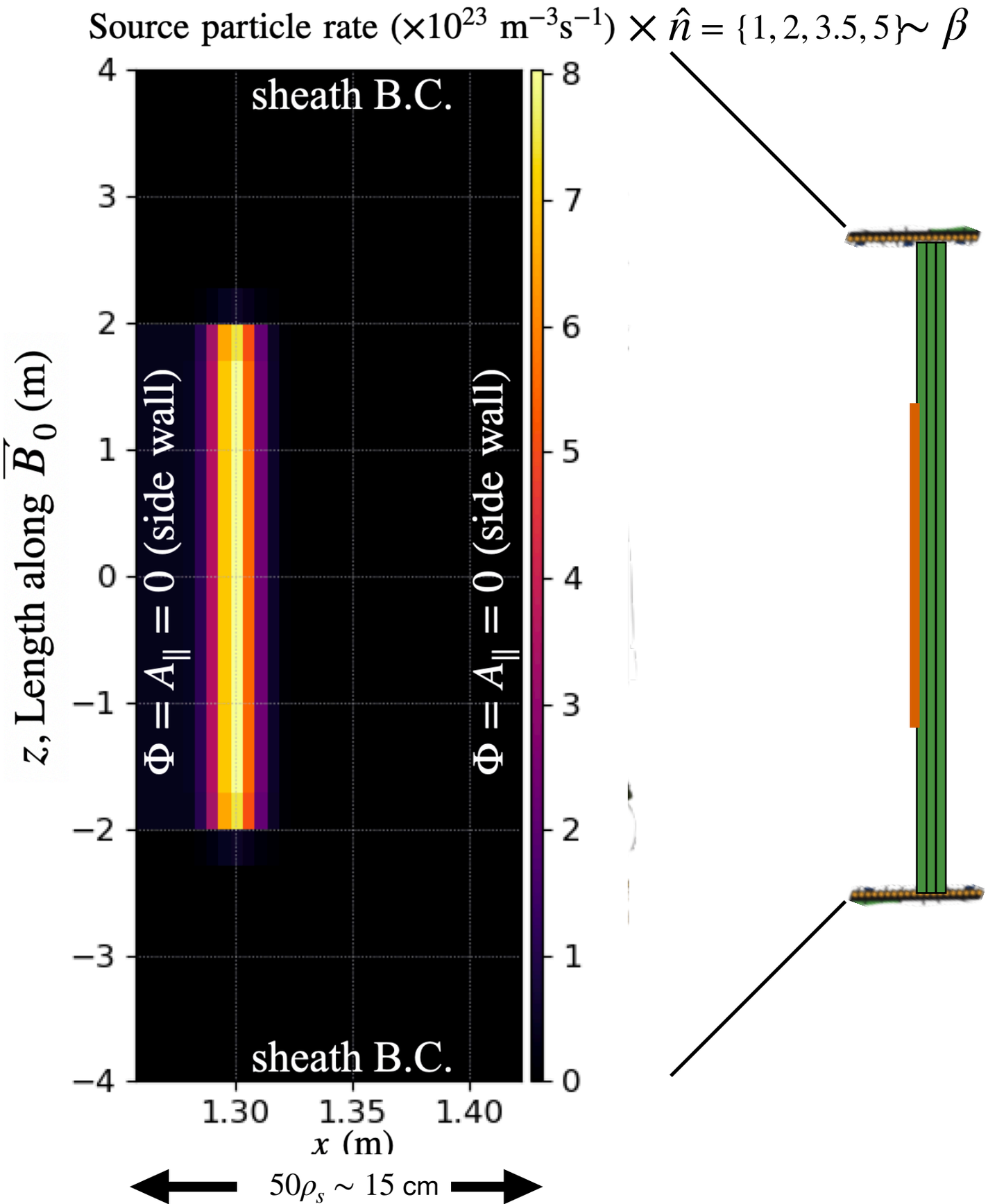
- Open-field-line region only
- Simplified helical geometry with vertical flux surfaces, const curvature and no shear

Modeling the NSTX SOL with Gkeyll



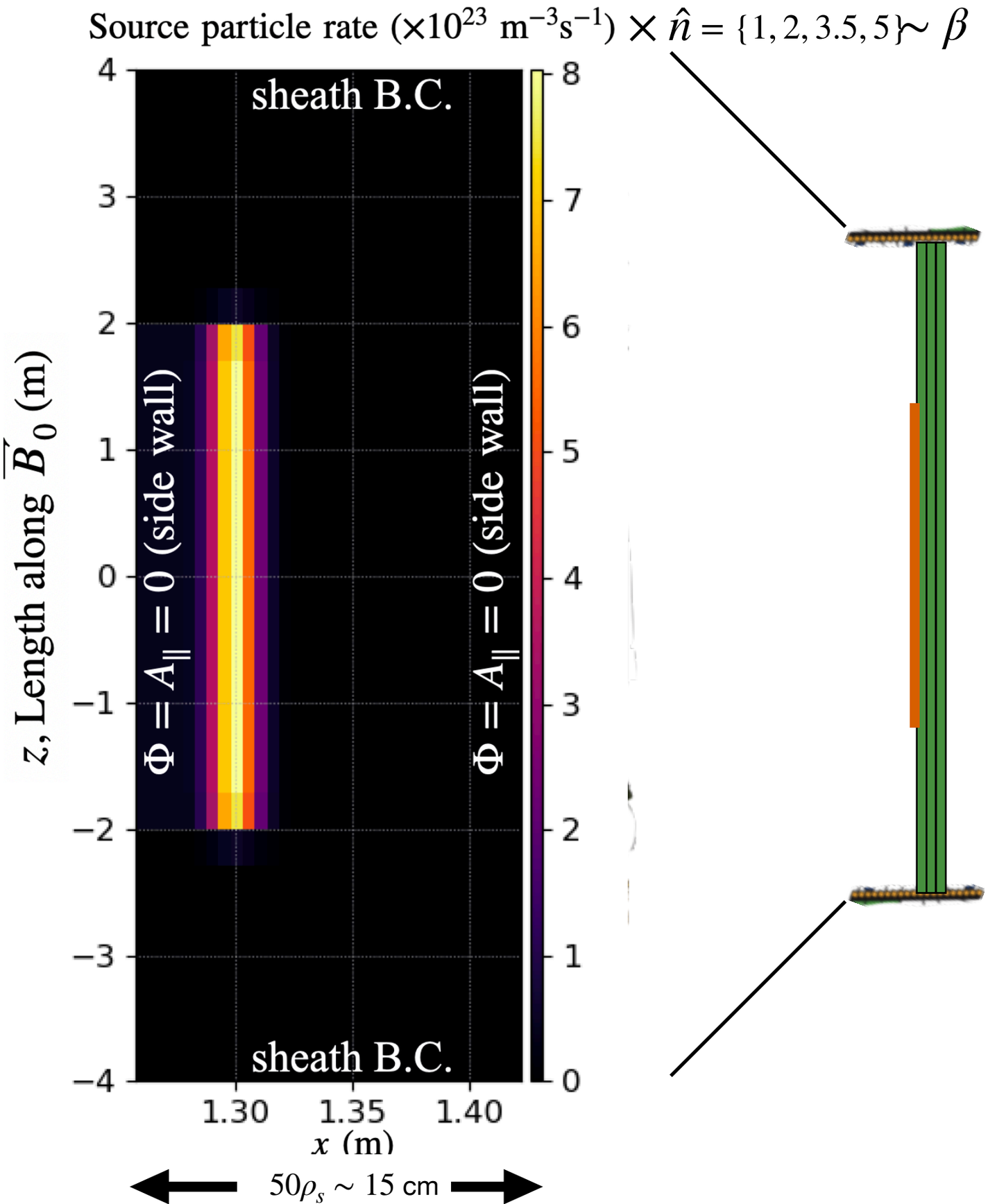
- Open-field-line region only
- Simplified helical geometry with vertical flux surfaces, const curvature and no shear
- Model flux of heat and particles across separatrix with source

Modeling the NSTX SOL with Gkeyll



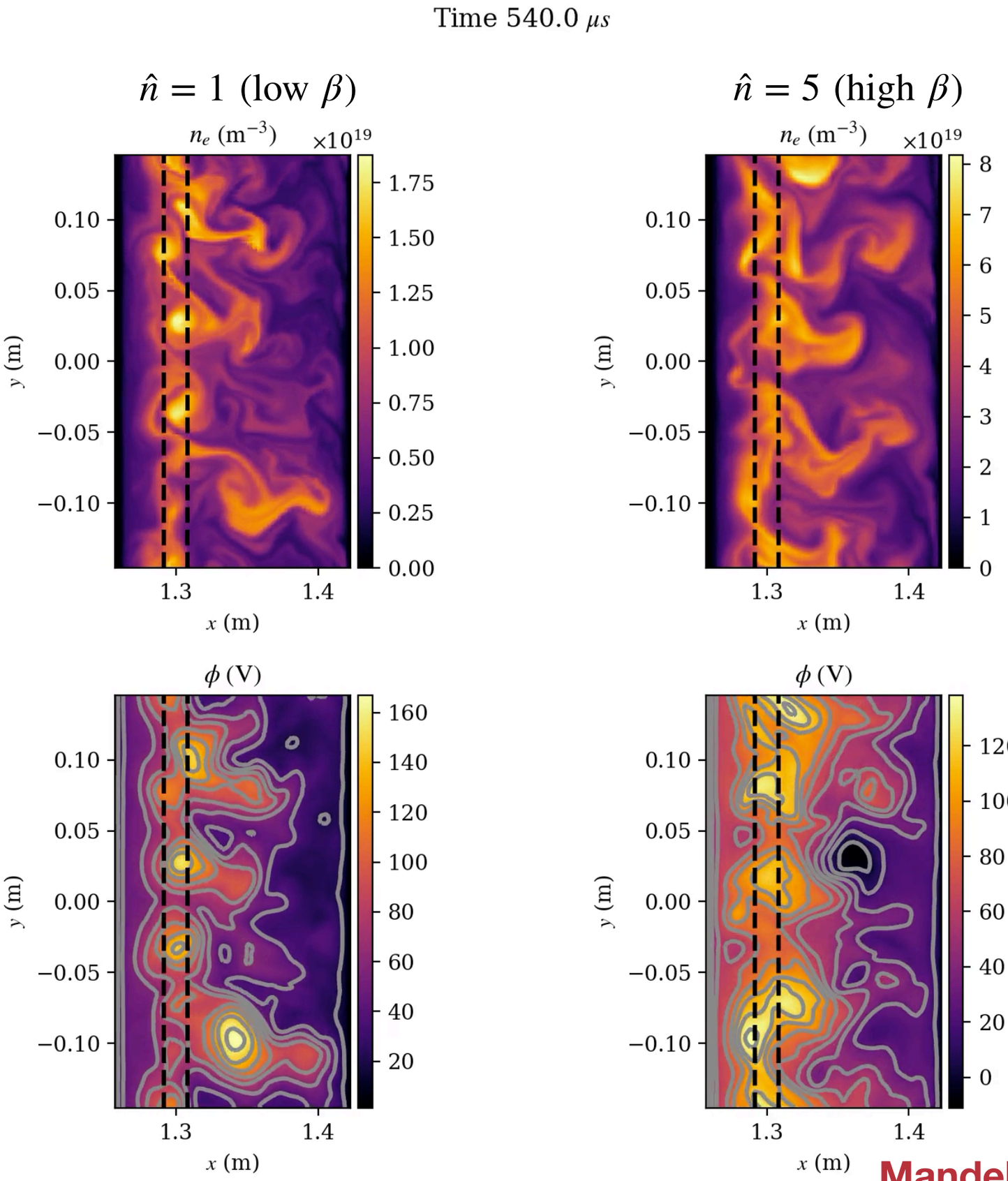
- Open-field-line region only
- Simplified helical geometry with vertical flux surfaces, const curvature and no shear
- Model flux of heat and particles across separatrix with source
  - To examine EM effects, do a parameter scan of β at fixed temperature ($T_{\text{src}} = 70$ eV) by scaling source particle rate by factor $\hat{n}$
- Base case ($\hat{n} = 1$) corresponds to "nominal" experimental heating power, with $P_{\text{SOL}} = 5.4$ MW

Modeling the NSTX SOL with Gkeyll

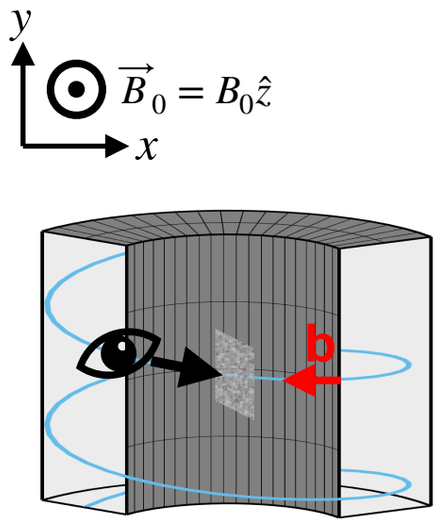


- Open-field-line region only
- Simplified helical geometry with vertical flux surfaces, const curvature and no shear
- Model flux of heat and particles across separatrix with source
  - To examine EM effects, do a parameter scan of β at fixed temperature ($T_{\text{src}} = 70$ eV) by scaling source particle rate by factor $\hat{n}$
  - Base case ($\hat{n} = 1$) corresponds to "nominal" experimental heating power, with $P_{\text{SOL}} = 5.4$ MW
- Boundary conditions:
  - perfectly conducting walls ($\phi = A_\parallel = 0$) in radial direction, *x*
  - periodic in binormal direction, *y*
  - conducting sheath model BC along field line, *z*
    - reflects low energy electrons, allows sheath current fluctuations

Simulation results: midplane electron density and electrostatic potential



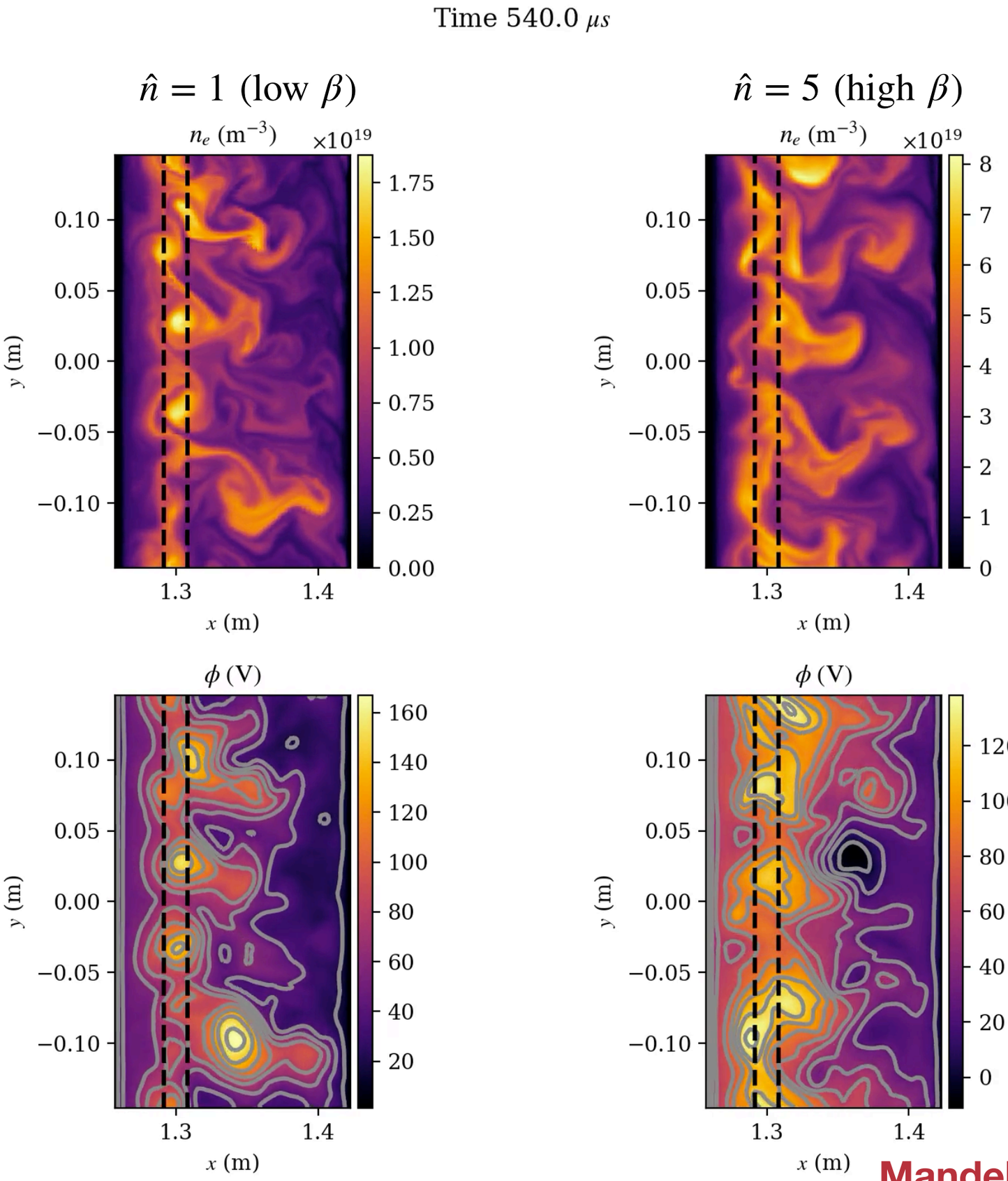
- Density blobs propagate radially outwards (interchange instability)
- Similar density structures in both low β ($\hat{n} = 1$) and high β ($\hat{n} = 5$) cases
- In low β ($\hat{n} = 1$) case, tendency towards monopole potential structures causes blobs to spin†



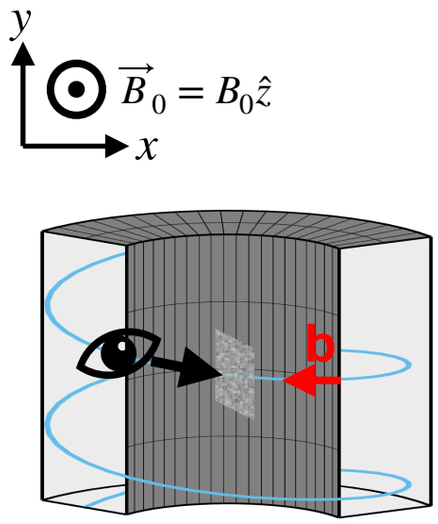
† Angus et al, Phys. Plasmas (2012)

Mandell et al, PoP (2022)

Simulation results: midplane electron density and electrostatic potential



- Density blobs propagate radially outwards (interchange instability)
- Similar density structures in both low β ($\hat{n} = 1$) and high β ($\hat{n} = 5$) cases
- In low β ($\hat{n} = 1$) case, tendency towards monopole potential structures causes blobs to spin†



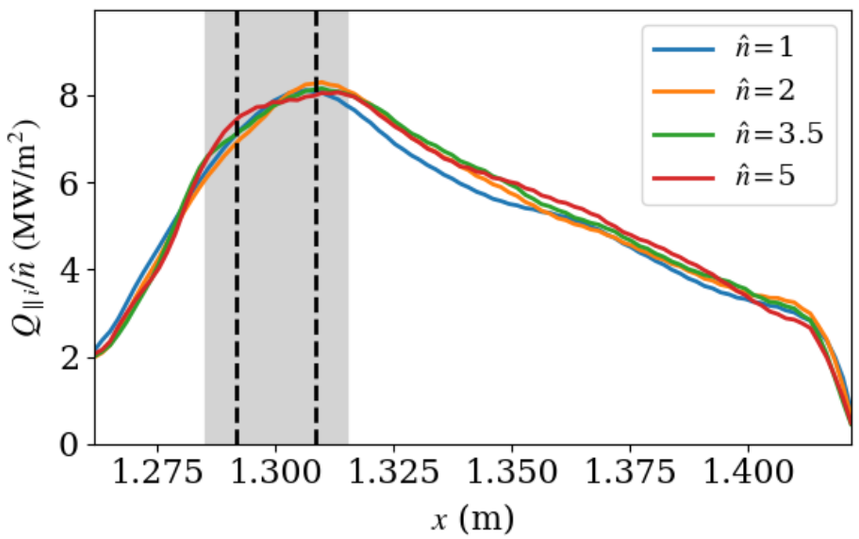
† Angus et al, Phys. Plasmas (2012)

Mandell et al, PoP (2022)

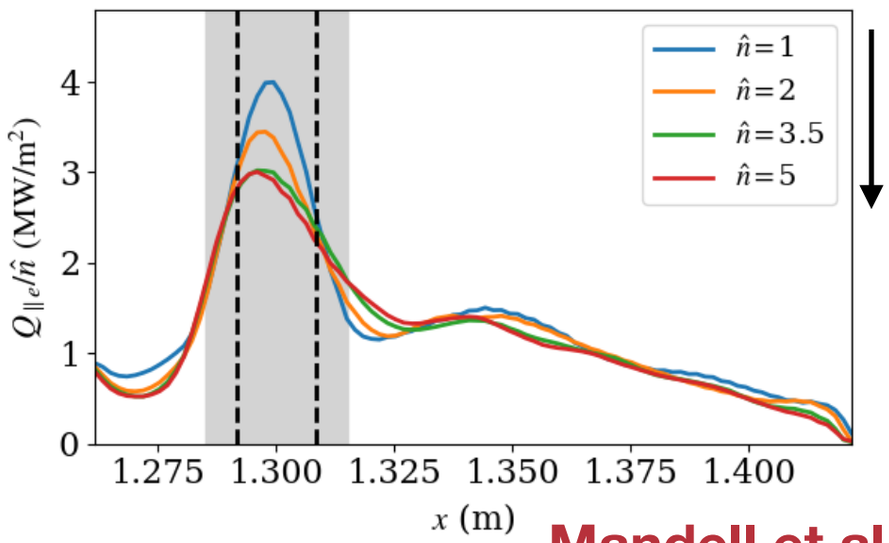
Increased cross-field transport broadens electron heat flux to endplates

Heat flux to endplate: $Q_{\|}(x) = \left\langle \int d^3v \; v_{\|}(mv^2/2 + q\Phi)\Big|_{z=z_{\text{end}}} \right\rangle_{y,t}$

ions



electrons



increasing β

Mandell et al, PoP (2022)

- Noticeable broadening of radial profile of electron heat flux to endplates as $\beta \sim \hat{n}$ increases, peak decreases ~25%

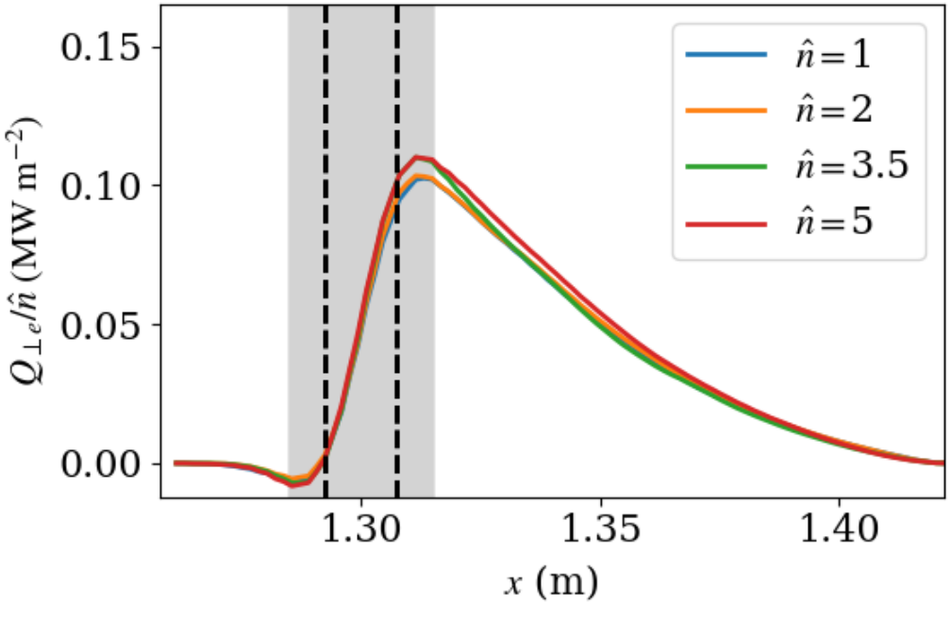
- Electron cross-field (perpendicular) heat flux slightly increases by ~10% in high β cases, consistent with broadening

- Differences in heat flux profiles between electrons and ions reflect differences in the competition between parallel and perpendicular transport

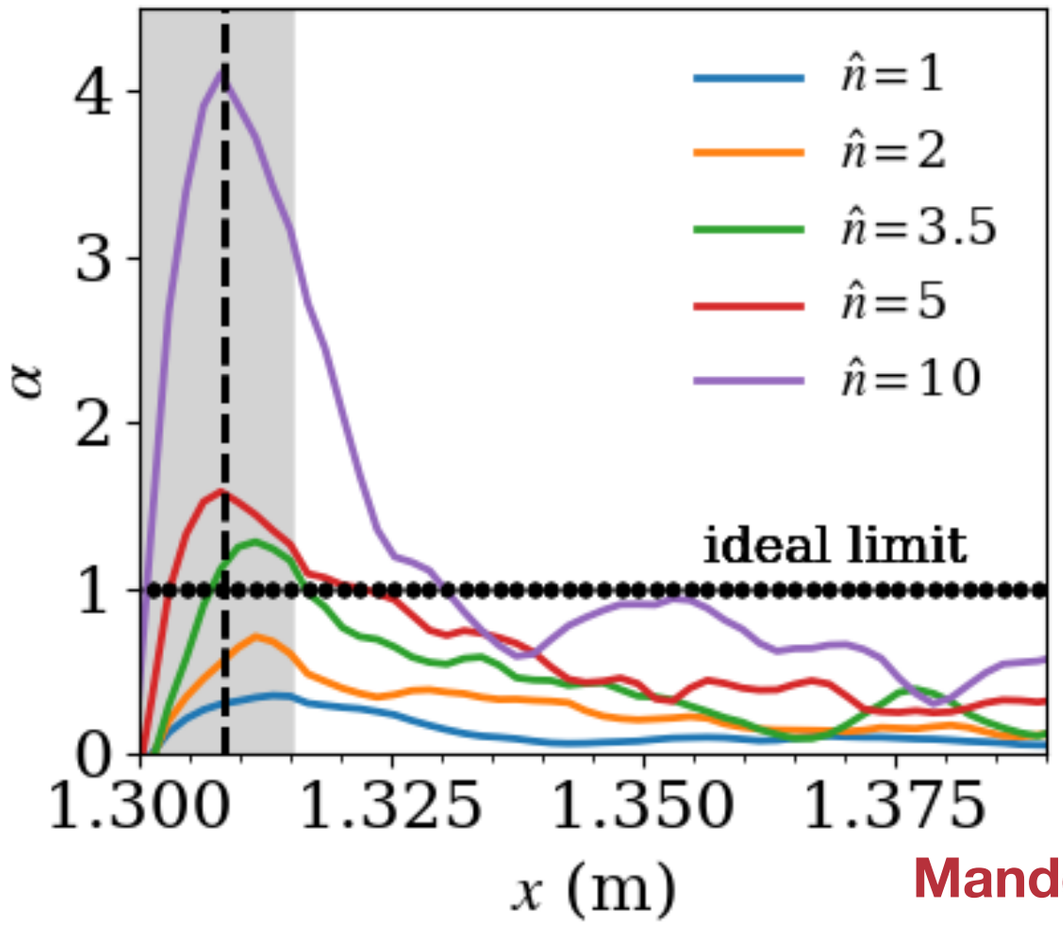
    - Electrons flow much more quickly to endplates, so there is less perp. spreading of heat

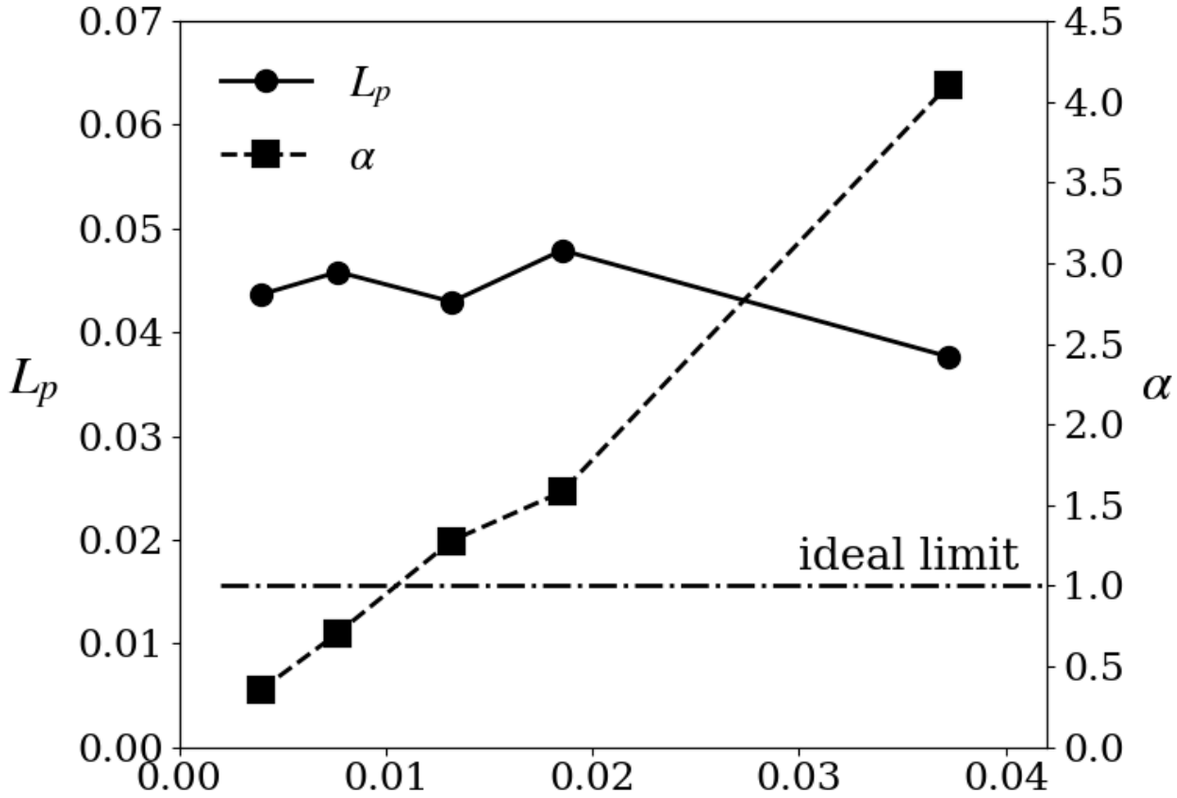
Perp. heat flux near midplane:

$Q_{\perp} = \langle \tilde{p}\tilde{v}_r \rangle + \langle \tilde{q}_{\|}\tilde{b}_r \rangle$



Exceeding the ideal ballooning limit?





Mandell & Hammett, in prep

- Examining the ideal ballooning parameter, $\alpha = L_\parallel^2 \beta / (\pi^2 R L_p)$, the ideal limit at $\alpha = 1$ is exceeded in the steep-gradient region of the high-beta simulations

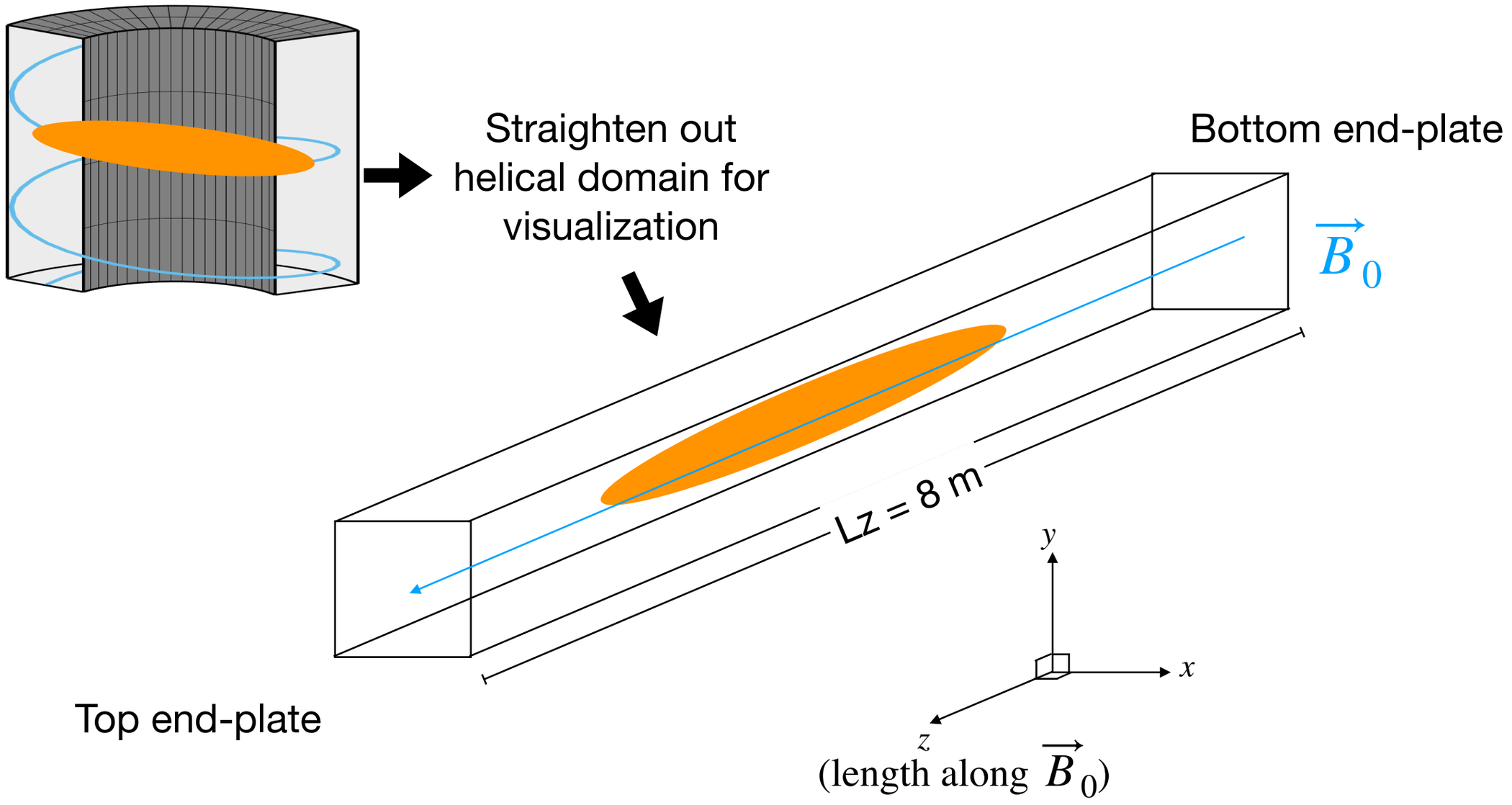
- Meanwhile, the pressure gradient scale-length (L_p, where $\nabla p \sim p/L_p$) is relatively constant as β is increased

- This is surprising! Might expect that as heating is added and the pressure profile approaches the ballooning limit, the turbulence will become stronger and broaden the profile so as to increase L_p to keep $\alpha \propto \beta / L_p < 1$ below the ballooning limit

  - We have recently developed a novel theoretical explanation (details in back-up slides if interested)

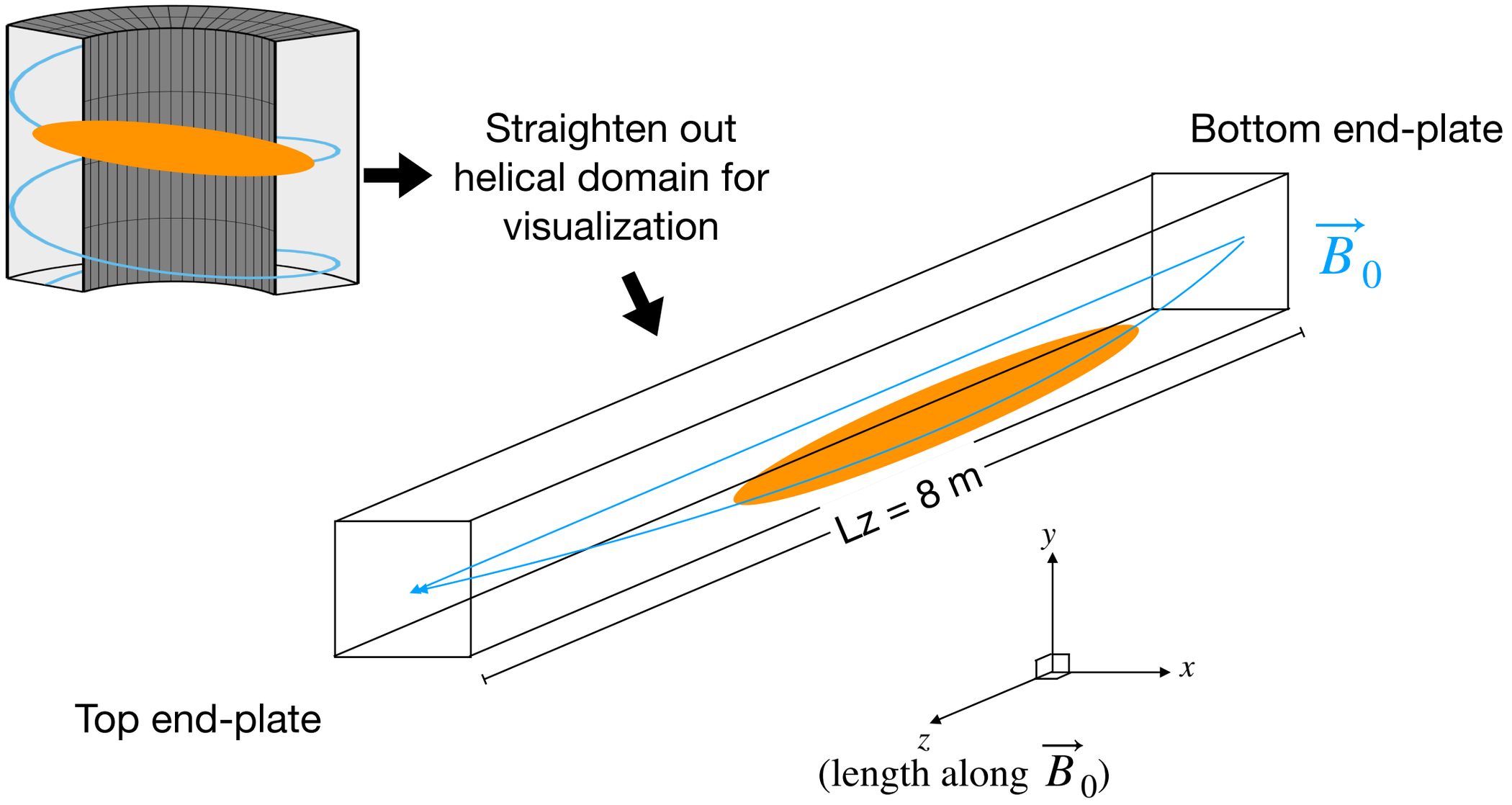
Interaction between magnetic field lines and plasma

- Magnetic field lines in plasma behave like taut strings
- The field lines can be “plucked” by plasma motion

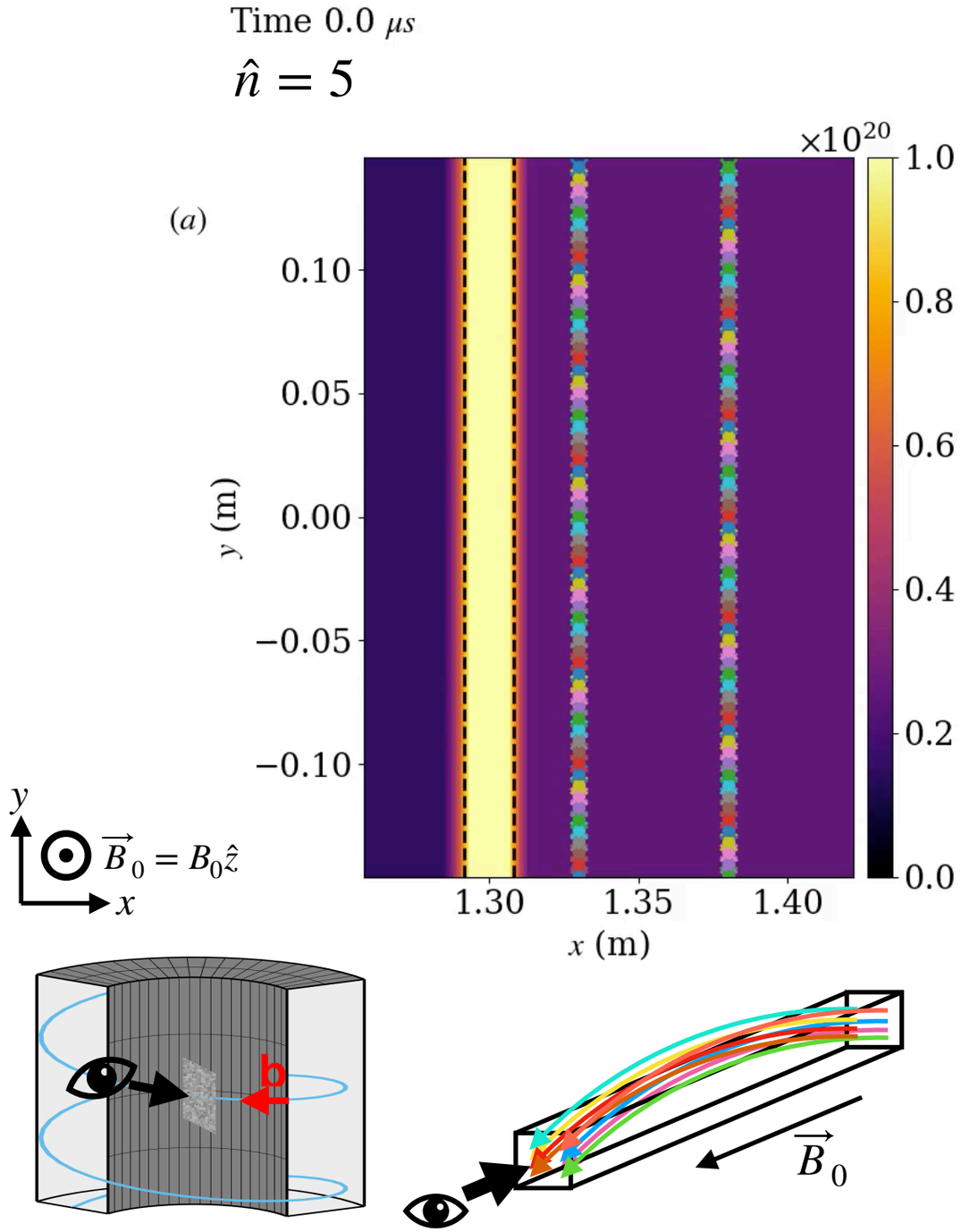


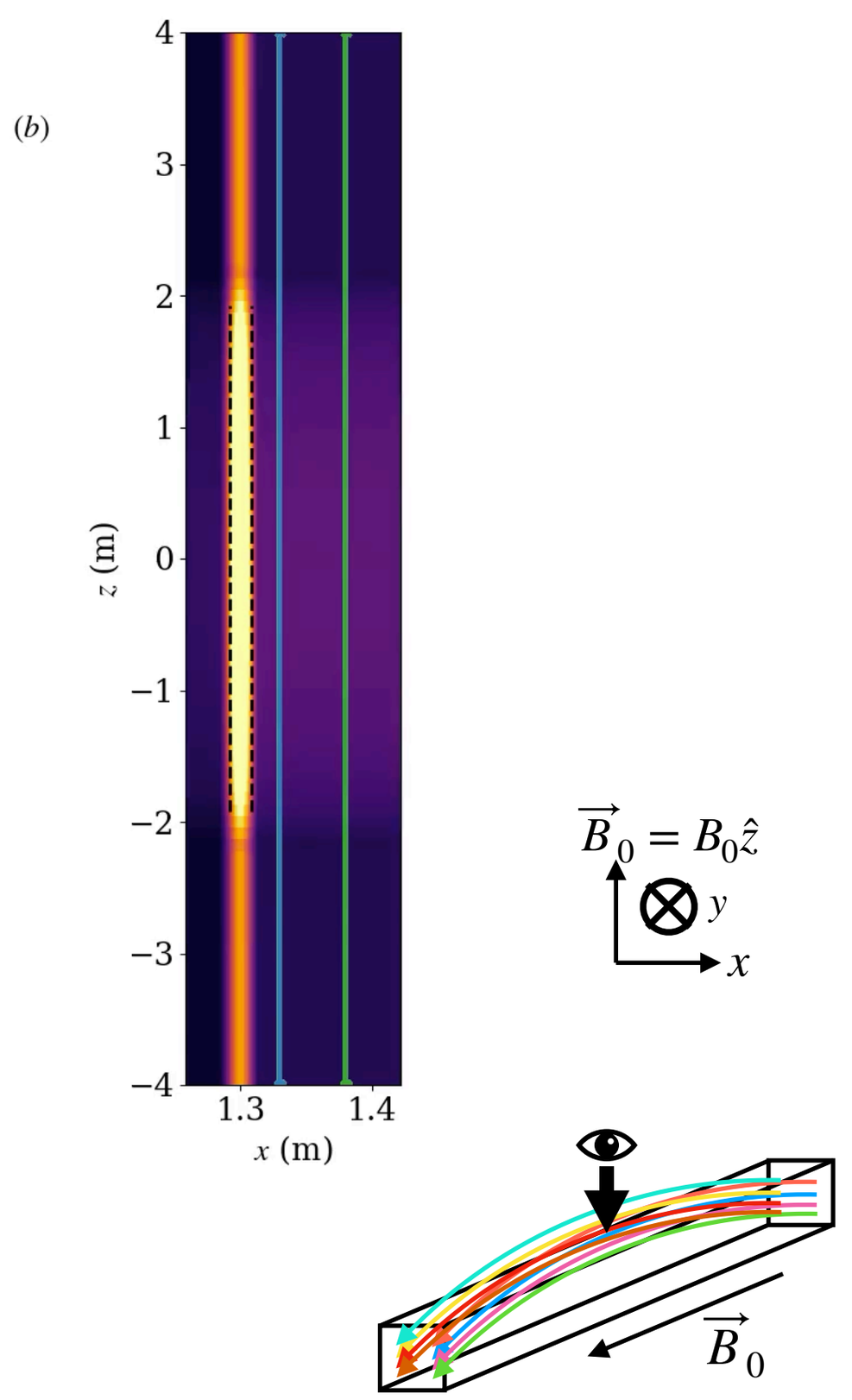
Interaction between magnetic field lines and plasma

- Magnetic field lines in plasma behave like taut strings
- The field lines can be “plucked” by plasma motion

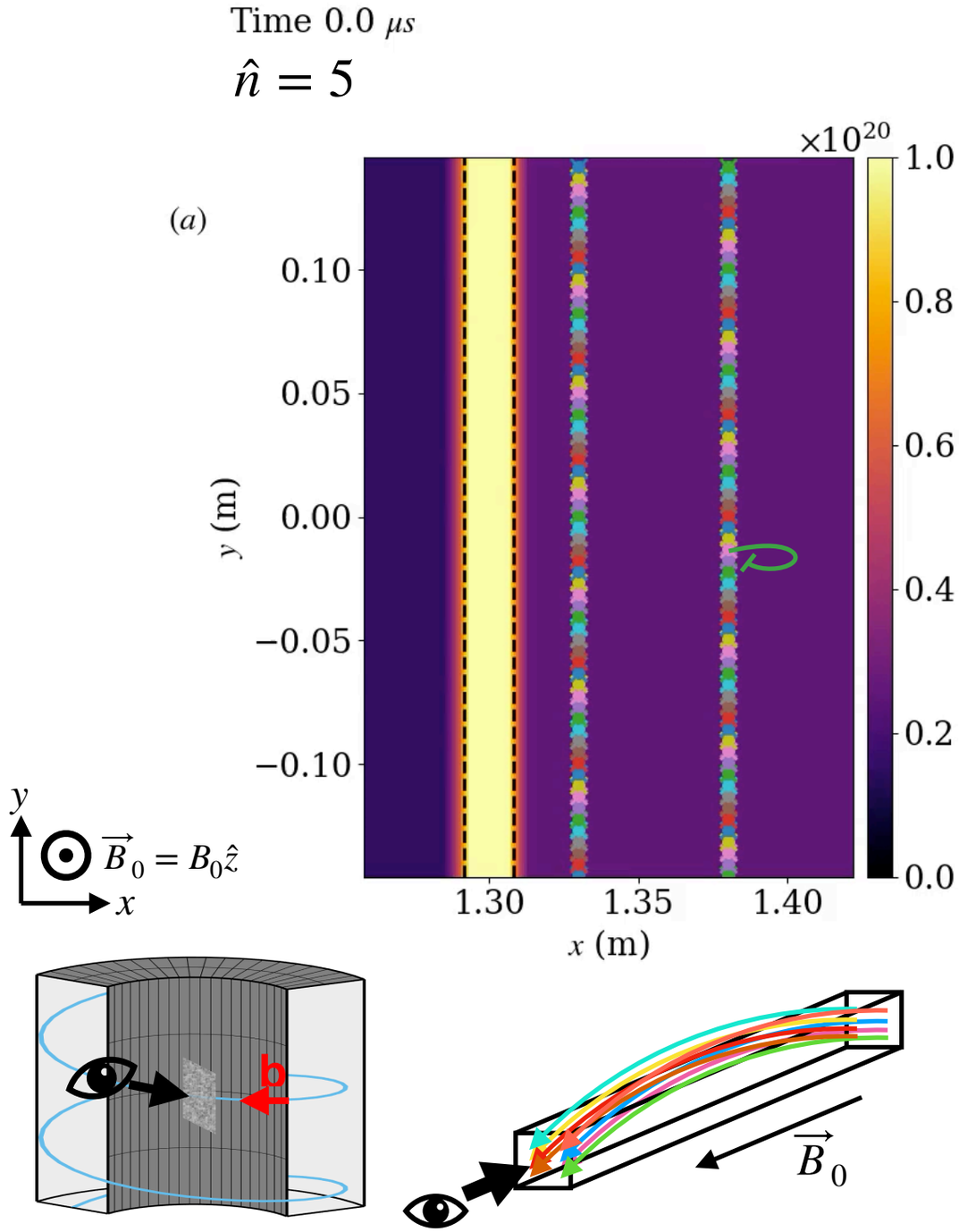


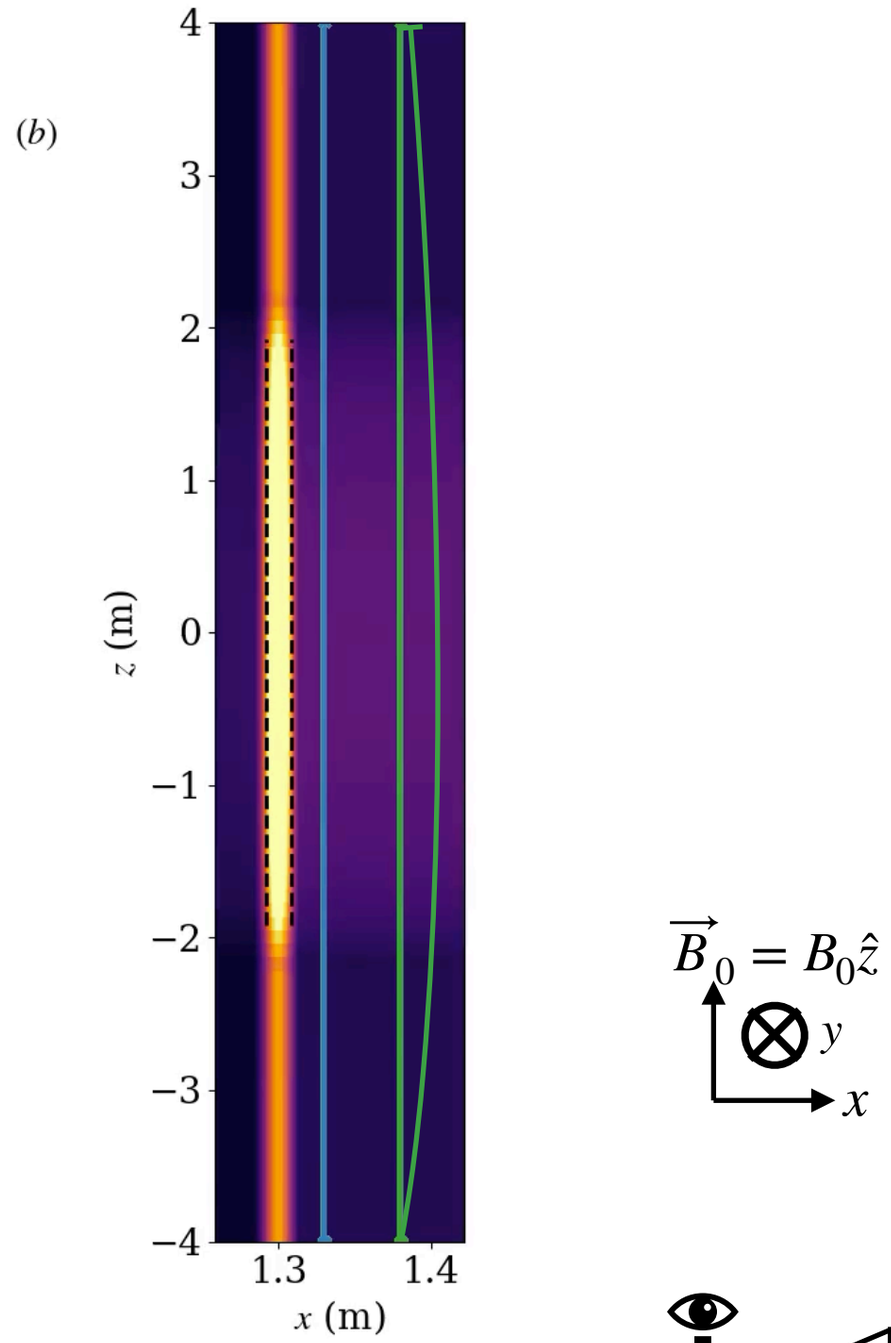
Field line bending and ballooning with partial line-tying



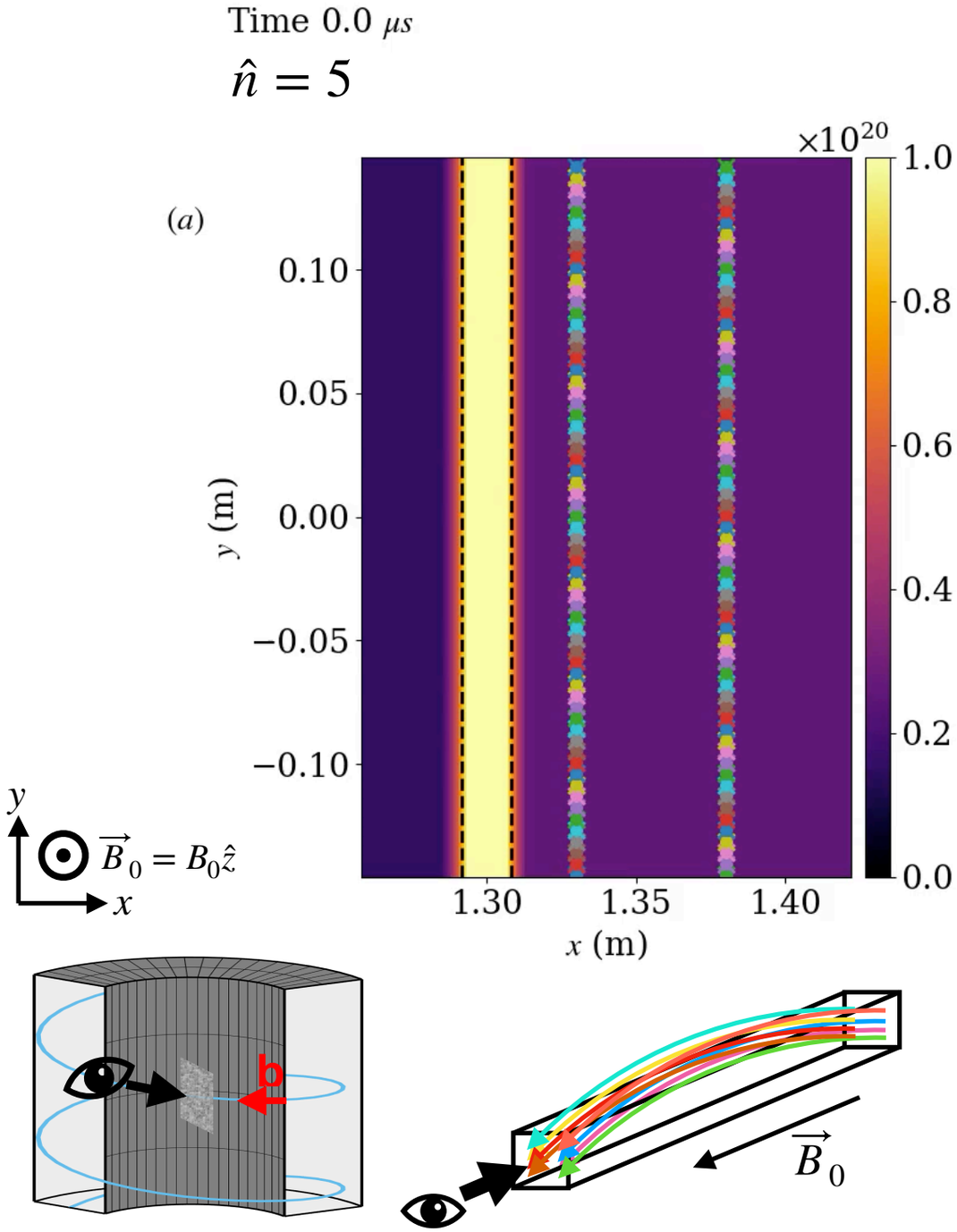


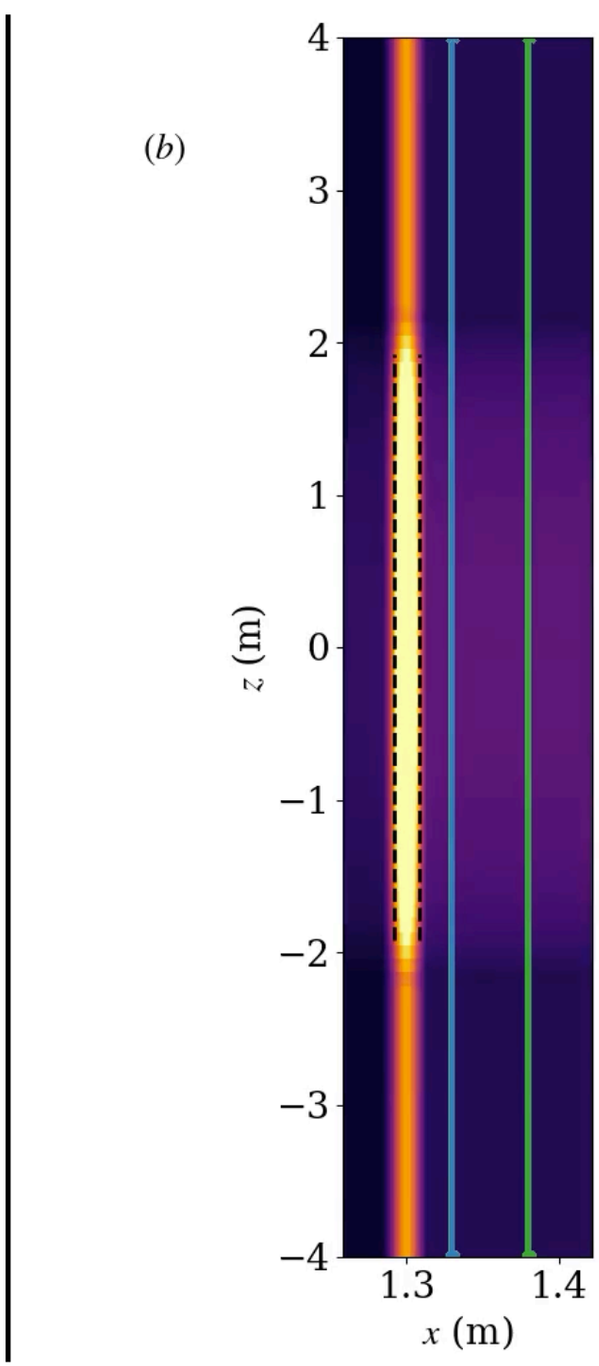
Field line bending and ballooning with partial line-tying

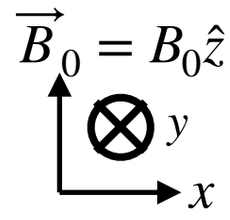


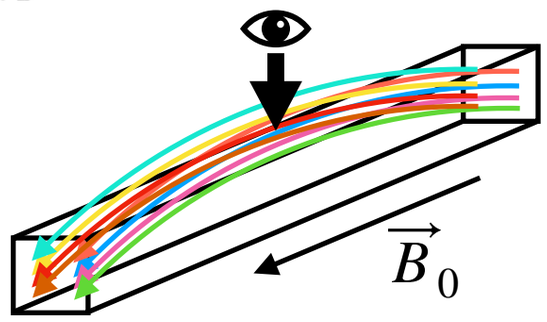


Field line bending and ballooning with partial line-tying









- Blobs bend/stretch magnetic field lines
- Finite sheath conductivity allows footpoints to slip at endplates, so line-tying is only partial
- Abrupt jumps → reconnection?
- Finite $k_{\|}$ → ballooning

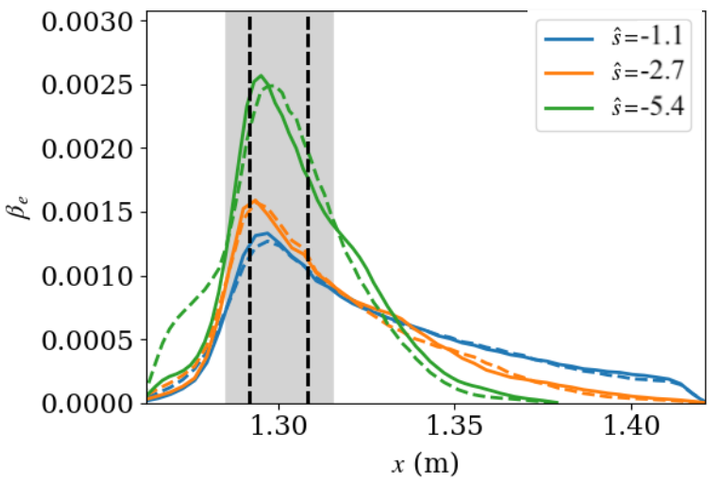
Ongoing Gkeyll development work to enable predictive capability

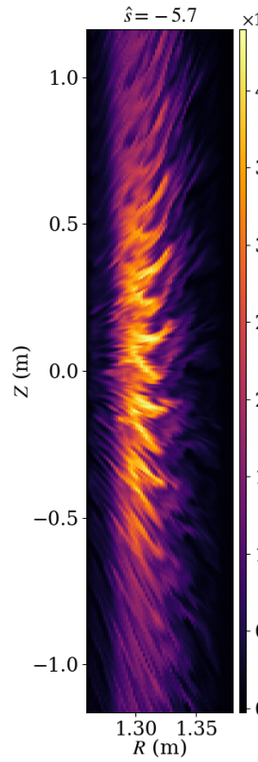
Ongoing Gkeyll development work to enable predictive capability

- A number of key developments in progress that will improve the realism of Gkeyll boundary physics simulations, allowing validation of the code with experimental results and enabling reactor-relevant predictive modeling

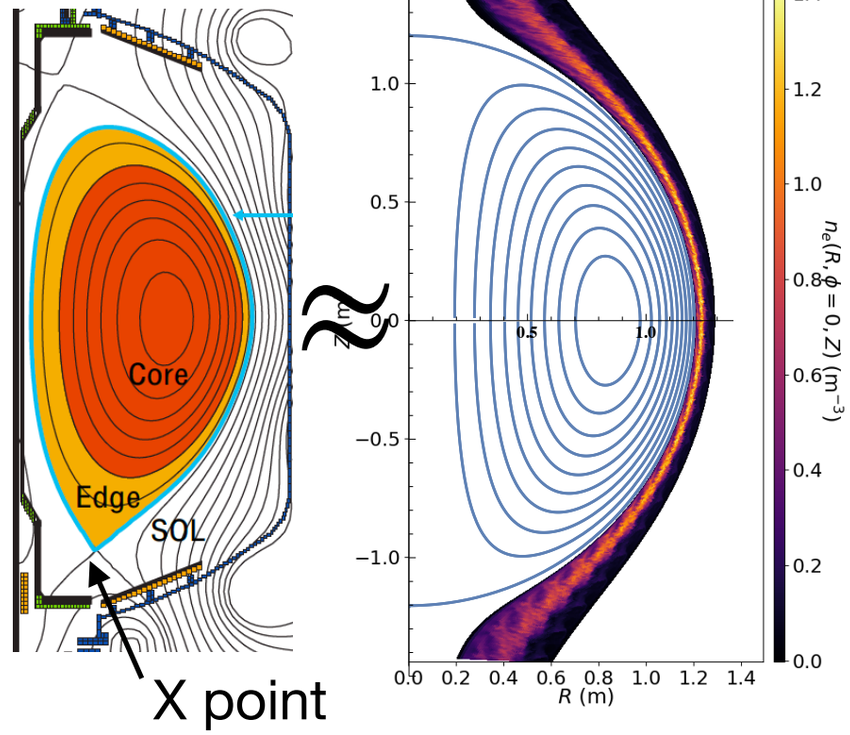
Ongoing Gkeyll development work to enable predictive capability

- A number of key developments in progress that will improve the realism of Gkeyll boundary physics simulations, allowing validation of the code with experimental results and enabling reactor-relevant predictive modeling
  - Can now model open-field-line geometry with magnetic shear and realistic shaping
    - Magnetic shear reduces SOL transport (Mandell et al, PPCF 2022)
    - **Challenge**: the X point is a coordinate singularity in the usually-efficient field-aligned coordinate system we prefer, but X point is needed for experimentally-relevant investigations of heat flux width and advanced divertor configurations



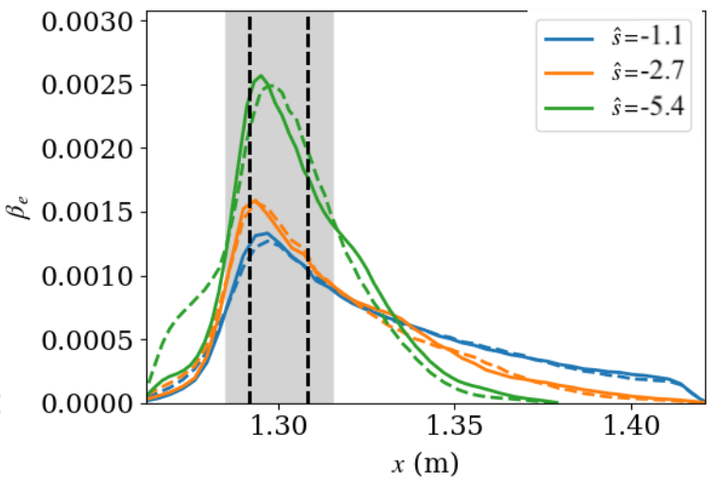


N. R. Mandell et al, PPCF 2022

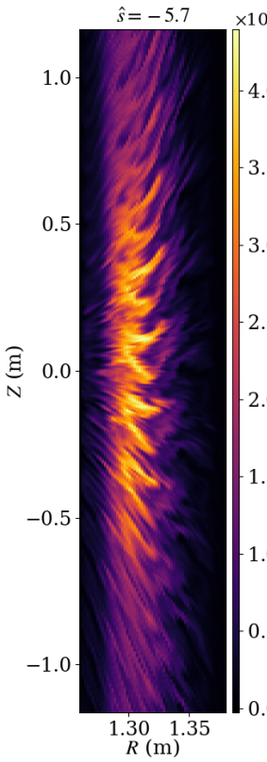


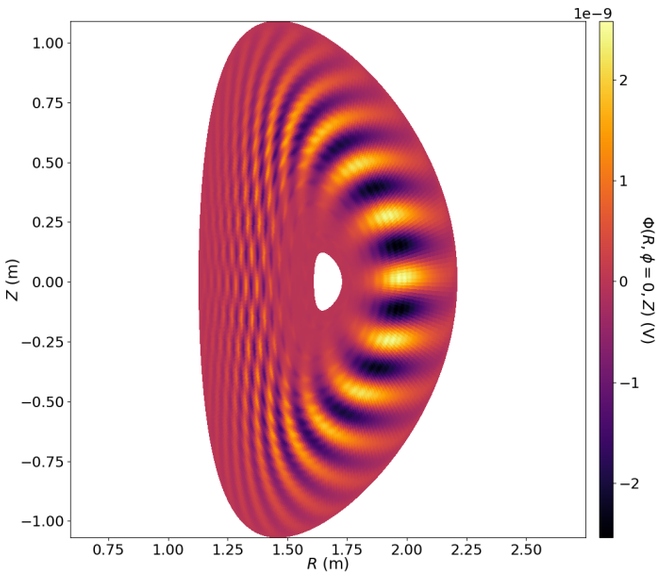
Ongoing Gkeyll development work to enable predictive capability

- A number of key developments in progress that will improve the realism of Gkeyll boundary physics simulations, allowing validation of the code with experimental results and enabling reactor-relevant predictive modeling
  - Can now model open-field-line geometry with magnetic shear and realistic shaping
    - Magnetic shear reduces SOL transport (Mandell et al, PPCF 2022)
    - **Challenge**: the X point is a coordinate singularity in the usually-efficient field-aligned coordinate system we prefer, but X point is needed for experimentally-relevant investigations of heat flux width and advanced divertor configurations
  - Closed + open field line simulations now possible (Francisquez et al, 2021)
    - Enables studies of coupled dynamics of edge/pedestal and boundary heat exhaust

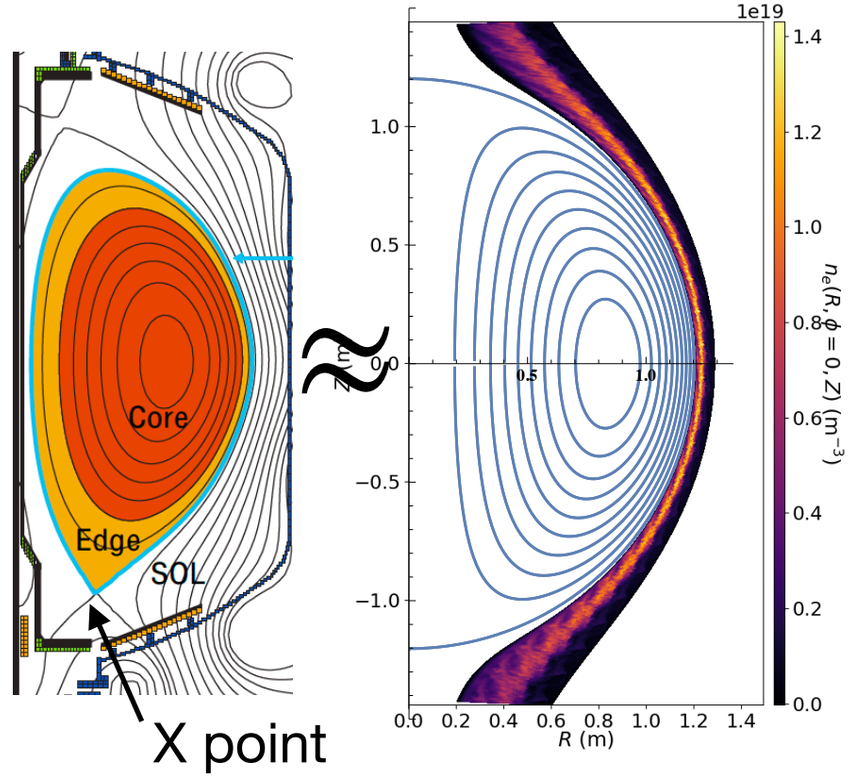


N. R. Mandell et al, PPCF 2022



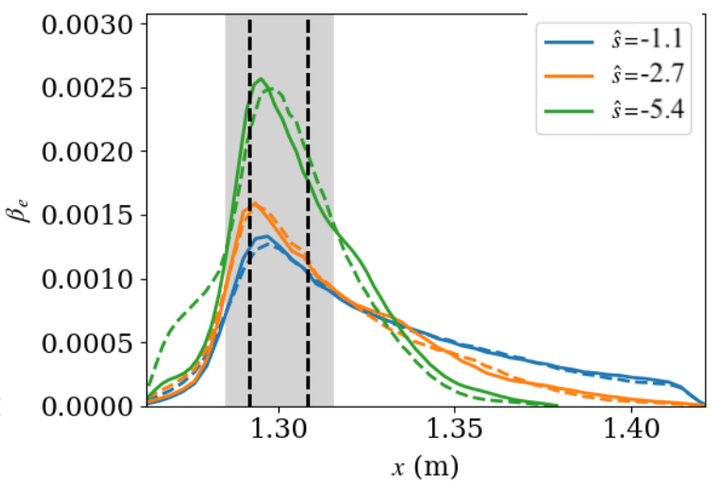


Francisquez et al (2021)

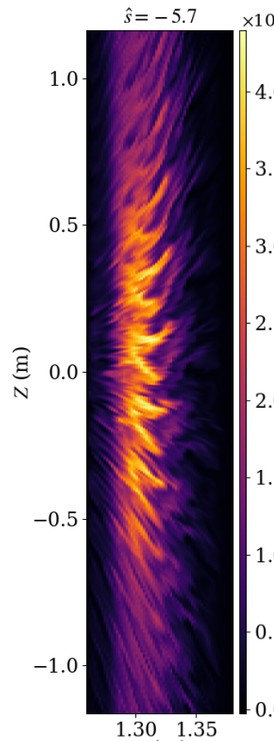


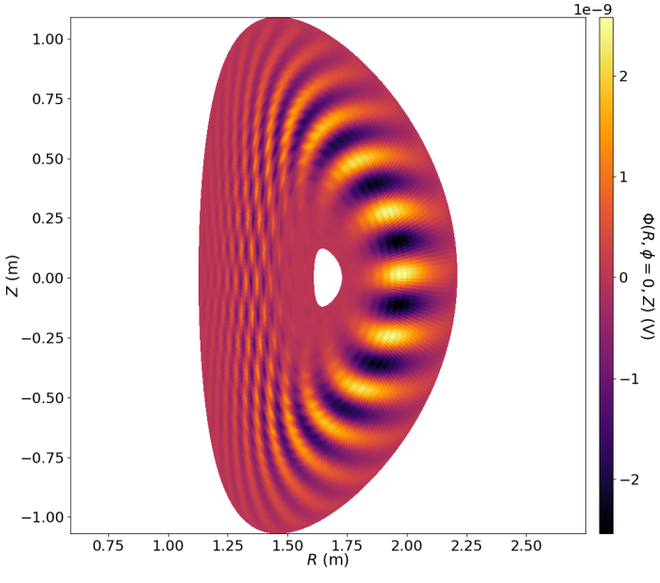
Ongoing Gkeyll development work to enable predictive capability

- A number of key developments in progress that will improve the realism of Gkeyll boundary physics simulations, allowing validation of the code with experimental results and enabling reactor-relevant predictive modeling
  - Can now model open-field-line geometry with magnetic shear and realistic shaping
    - Magnetic shear reduces SOL transport (Mandell et al, PPCF 2022)
    - **Challenge**: the X point is a coordinate singularity in the usually-efficient field-aligned coordinate system we prefer, but X point is needed for experimentally-relevant investigations of heat flux width and advanced divertor configurations
  - Closed + open field line simulations now possible (Francisquez et al, 2021)
    - Enables studies of coupled dynamics of edge/pedestal and boundary heat exhaust
  - Neutral modeling (Bernard et al, PoP 2022)
    - Important for accurate modeling of boundary fueling and recycling
    - Enables studies of detachment, a possible solution to the heat exhaust problem at reactor scale

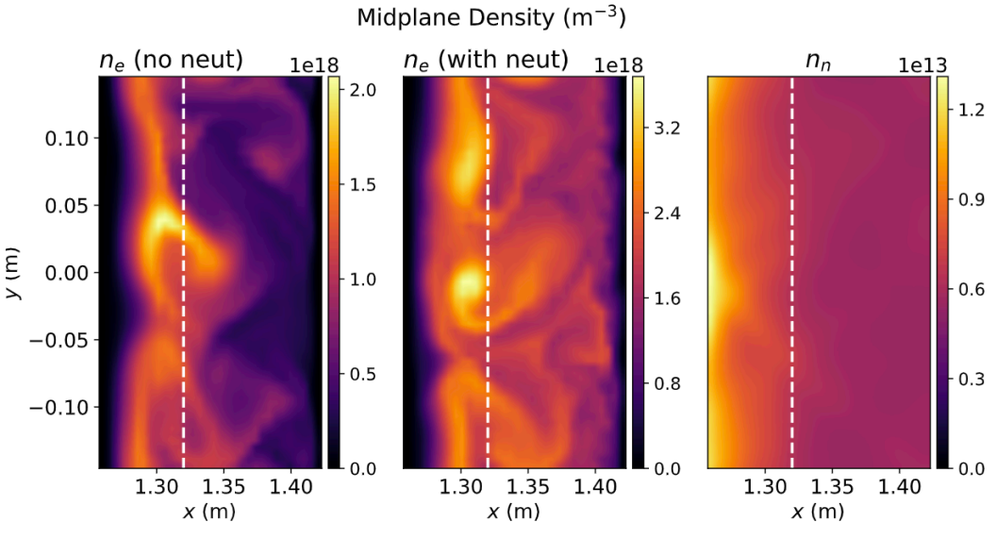


N. R. Mandell et al, PPCF 2022

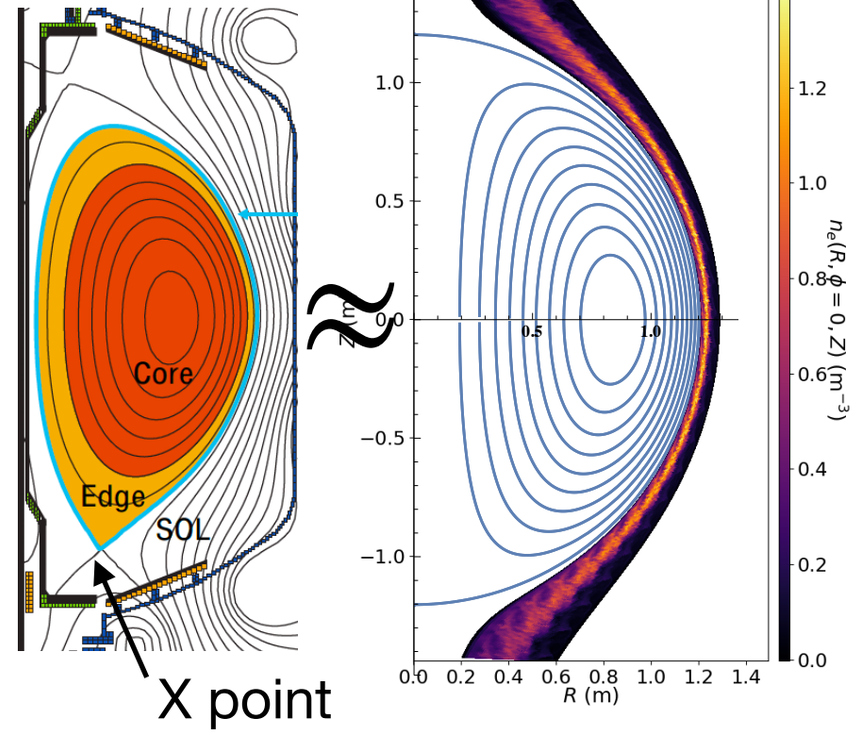




Francisquez et al (2021)

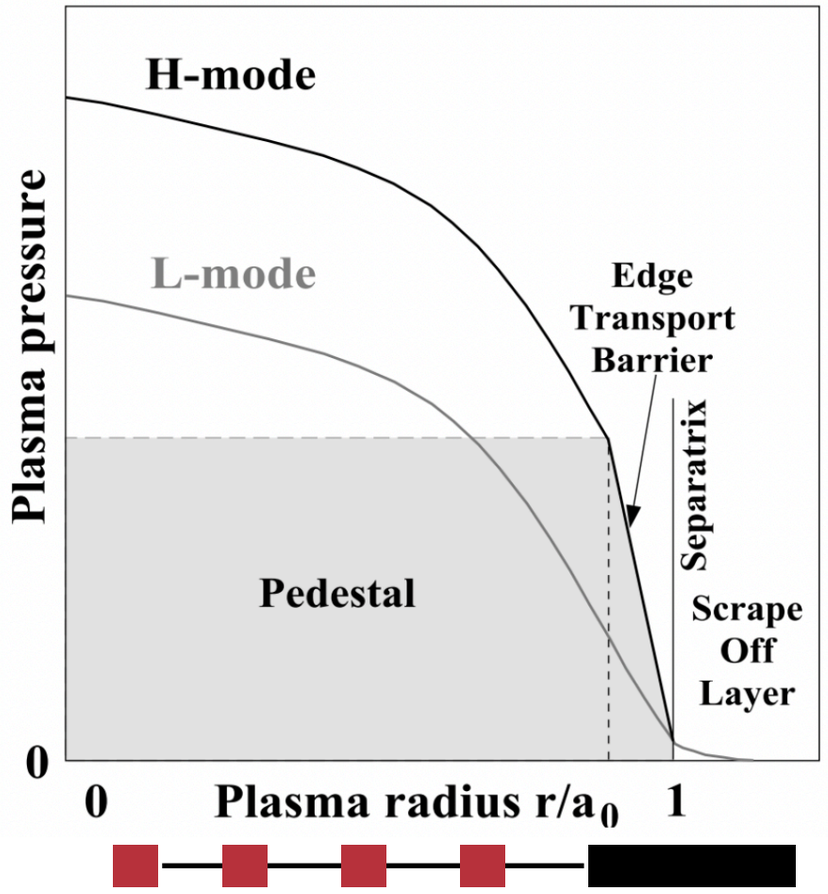


Bernard et al (2022)



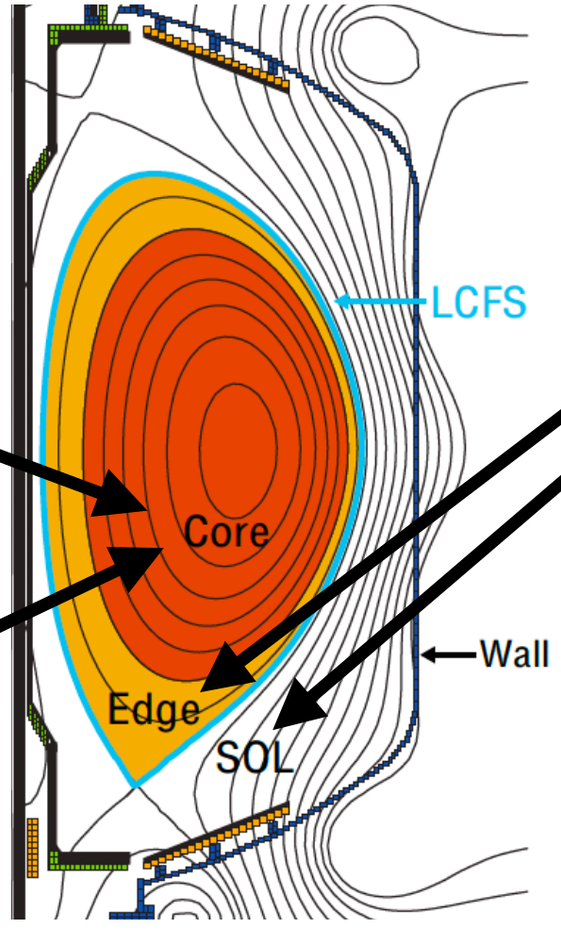
A whole-device transport model

- We can develop a whole-device transport model with GX+Trinity in the core and Gkeyll in the boundary
  - Will be able to directly study impact of changes in core confinement on boundary exhaust
  - Will be able to directly study impact of boundary exhaust solutions (like detachment) on core confinement
- Open questions:
  - How tightly coupled are the core and the edge? Do we need to run a Gkeyll simulation every Trinity timestep, or can we update the edge less frequently?



+

Trinity



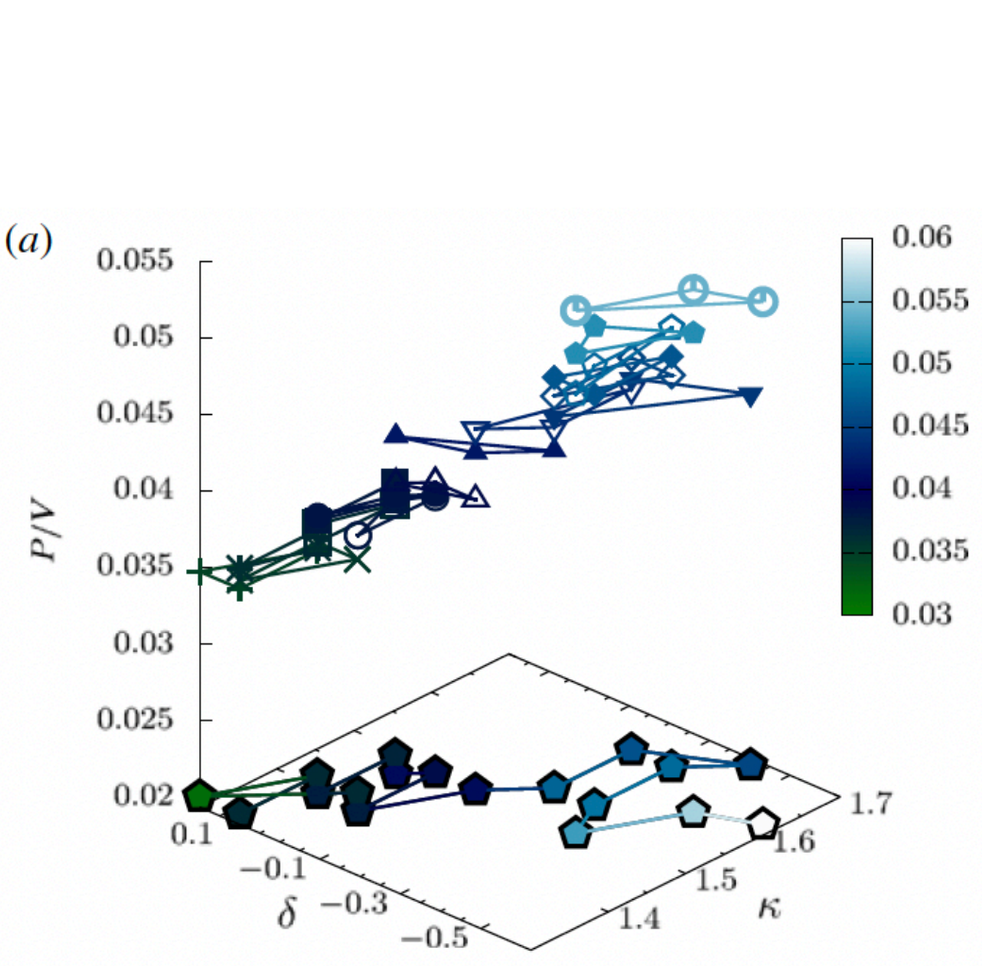
Gkeyll

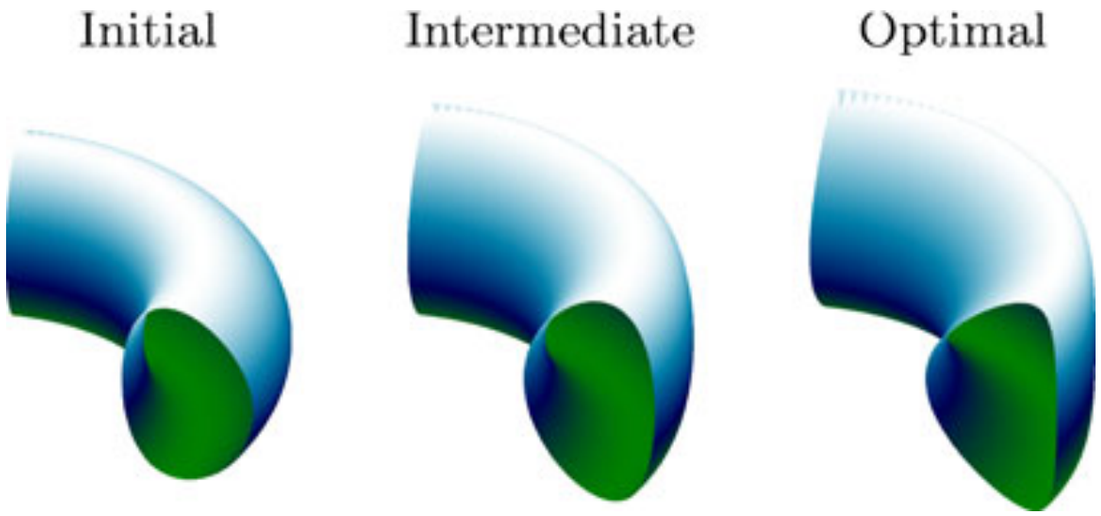
The goal: whole-device transport optimization

- Each whole-device calculation can compute key figures of merit to be optimized for a fusion reactor design
  - Total fusion power, energy confinement time, heat load to walls, etc
- In a tokamak core, there are ~20 shaping parameters that could be varied simultaneously; order of magnitude more in a stellarator
- In the boundary, could start with a few candidate divertor configurations
- Parallel optimization algorithm could run O(10)-O(100) shapes simultaneously

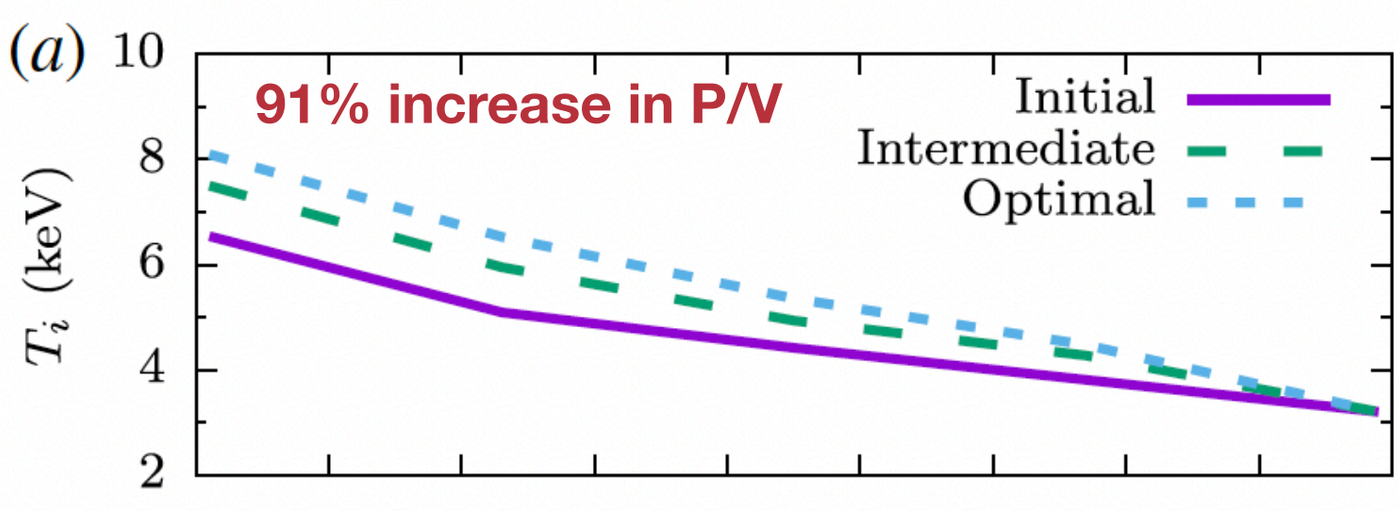
Transport optimization is possible!

- Preliminary design study (**Highcock, Mandell, Barnes, Dorland (2018)**) found **negative triangularity** to be optimal in core
  - Used a lower-fidelity precursor to GX (gyrofluid model w/ some GK extensions), coupled to Trinity
  - **18 shape design evaluations = 8680 nonlinear calculations = 3000 GPU node hours**
  - **91% improvement in fusion power per unit volume from going to negative triangularity**
  - Consistent with TCV and DIII-D experimental results showing confinement improvement with negative triangularity



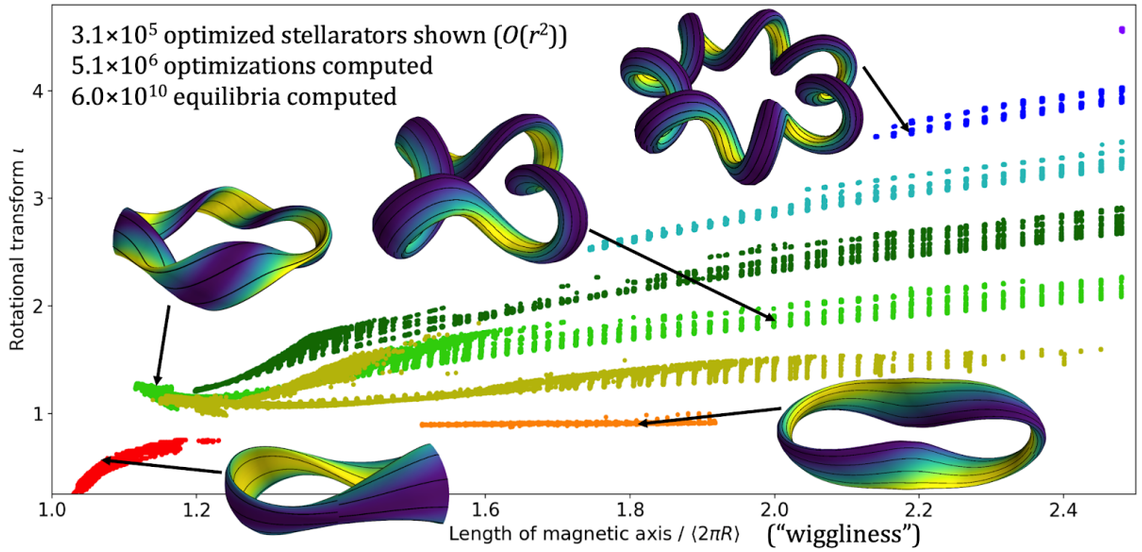


Highcock, Mandell, Barnes, Dorland (2018)

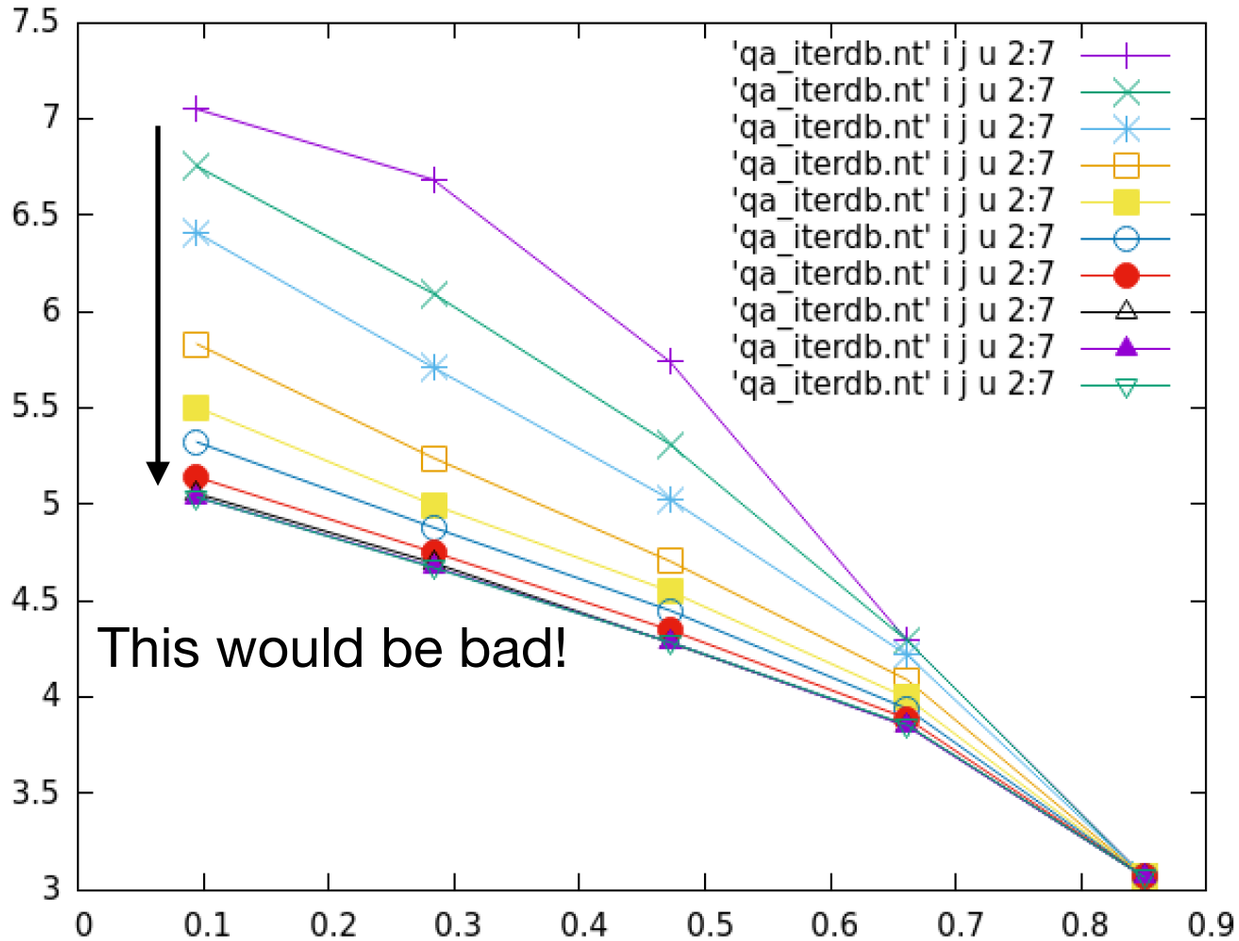


Need for transport in the optimization loop

- There are several groups working on stellarator optimization
  - Focus is primarily on optimizing neoclassical transport, coil complexity, etc; turbulent transport not currently included
  - Landreman et al showed an optimized quasi-axisymmetric (QA) stellarator at Sherwood conference. What if we used this design to build a JET-sized QA stellarator?
- Trinity+GX calculation shows that critical gradient is much smaller than in JET → poor performance!
- **This shows the need for putting transport in the optimization loop!**
- Collaborations with SIMSOPT and DESC groups have begun

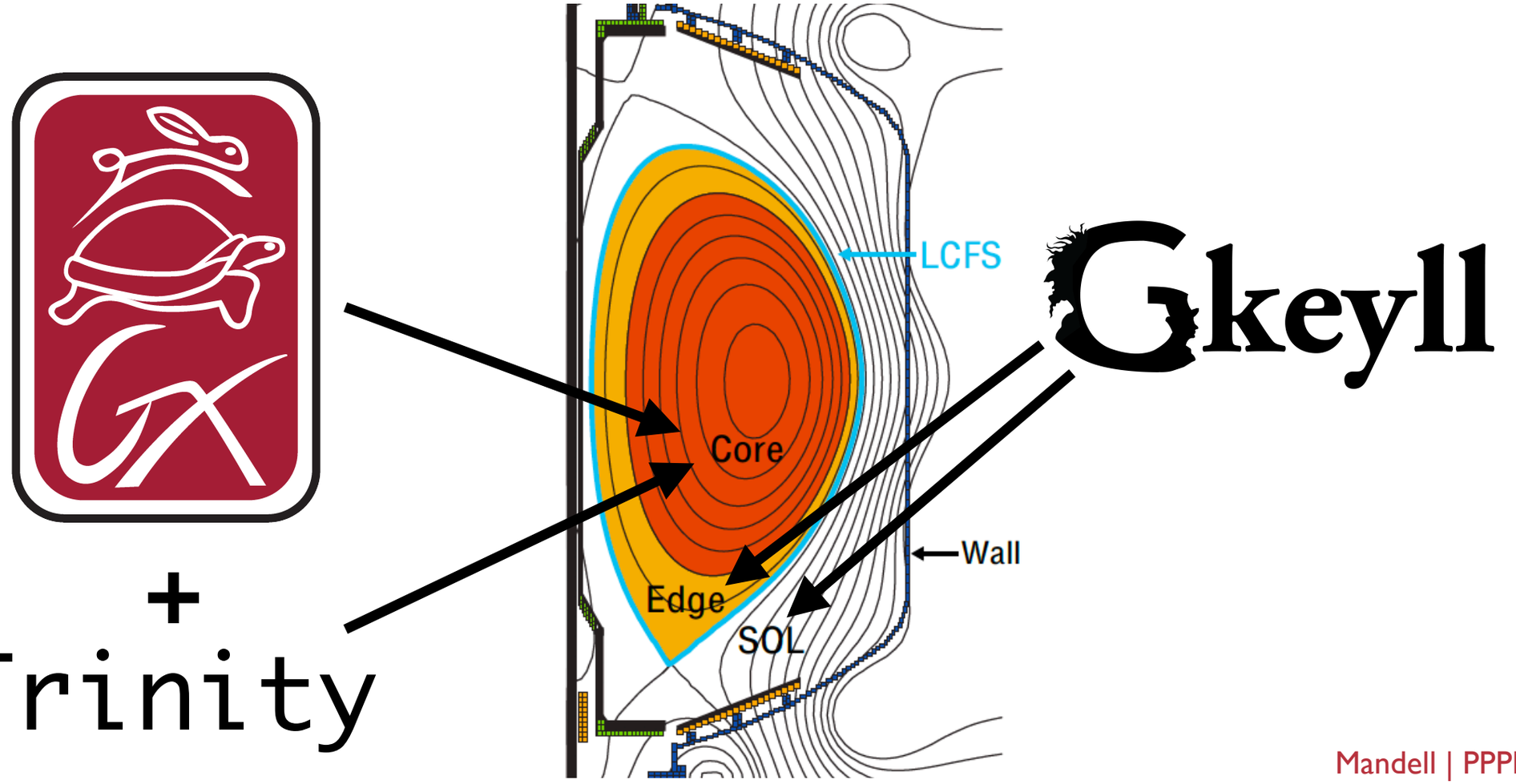


Landreman, SIMSOPT (Hidden Symmetries)



Summary

- I have presented a whole-device framework for modeling turbulence and transport in fusion reactors
  - **GX** models **core turbulence** with **spectral methods** on **GPUs**, and couples to a transport solver like **Trinity** to form a **multi-scale core transport model**
  - **Gkeyll** models **boundary turbulence** with **discontinuous Galerkin methods** and a first-of-a-kind **kinetic** scheme that includes **magnetic fluctuations**
- Still work to do, but we can achieve **whole-device transport modeling and optimization** to **design better fusion reactors**



Acknowledgements

This work is funded by the DOE FES Postdoctoral Fellow program, administered by ORISE.

Massachusetts Institute of Technology

OAK RIDGE INSTITUTE FOR SCIENCE AND EDUCATION

PPPL PRINCETON PLASMA PHYSICS LABORATORY

GX group:

- Bill Dorland
- Ian Abel
- Nate Barbour
- Braden Buck
- Rahul Gaur
- Patrick Kim
- Matt Landreman
- Jason Parisi
- Tony Qian
- … and others!

Gkeyll group:

- Ammar Hakim
- Greg Hammett
- Tess Bernard
- Petr Cagas
- Mana Francisquez
- Jimmy Juno
- … and others!

https://bitbucket.org/gyrokinetics/gx

https://gx.readthedocs.io

https://github.com/ammarhakim/gkyl/

https://gkyl.readthedocs.io/

Backup

GX approach: Laguerre-Hermite pseudo-spectral velocity formulation

Hermite-Laguerre transform of gyrokinetic equation:

$$\frac{\partial G'_{\ell,m}}{\partial t} + N_{\ell,m} + v_{ts}\nabla_\parallel \left(\sqrt{m+1}\, H_{\ell,m+1} + \sqrt{m}\, H_{\ell,m-1}\right)$$
$$+ v_{ts}\Big[-(\ell+1)\sqrt{m+1}\, H_{\ell,m+1} - \ell\sqrt{m+1}\, H_{\ell-1,m+1}$$
$$+ \ell\sqrt{m}\, H_{\ell,m-1} + (\ell+1)\sqrt{m}\, H_{\ell+1,m-1}\Big]\nabla_\parallel \ln B$$
$$+ i\omega_{ds}^{\kappa}\Big[\sqrt{(m+1)(m+2)}\, H_{\ell,m+2} + (2m+1)H_{\ell,m} + \sqrt{m(m-1)}\, H_{\ell,m-2}\Big]$$
$$+ i\omega_{ds}^{\nabla B}\Big[(\ell+1)\, H_{\ell+1,m} + (2\ell+1)H_{\ell,m} + \ell\, H_{\ell-1,m}\Big]$$
$$= D_{\ell,m} + C_{\ell,m}$$

Drive terms:

$$D_{\ell,m=0} = i\omega_* \left[\frac{1}{L_{ns}}\mathcal{J}_\ell + \frac{1}{L_{Ts}}\left[\ell\mathcal{J}_{\ell-1} + 2\ell\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\right]\Phi$$

$$D_{\ell,m=1} = v_{ts} i\omega_* \left[\frac{1}{L_{ns}}\mathcal{J}_\ell + \frac{1}{L_{Ts}}\left[\ell\mathcal{J}_{\ell-1} + (2\ell+1)\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\right] A_\parallel$$

$$D_{\ell,m=2} = \frac{1}{\sqrt{2}} i\omega_* \frac{1}{L_{Ts}}\mathcal{J}_\ell \Phi$$

$$D_{\ell,m=3} = v_{ts}\sqrt{\frac{3}{2}} i\omega_* \frac{1}{L_{Ts}}\mathcal{J}_\ell A_\parallel$$

$$D_{\ell,m>3} = 0,$$

Collisions (Dougherty model):

$$C_{\ell,0} = -\nu\,(b+2\ell+0)\, H_{\ell,0}$$
$$+ \nu\left(\sqrt{b}\,(\mathcal{J}_\ell + \mathcal{J}_{\ell-1})\,\bar{u}_\perp + 2\left[\ell\mathcal{J}_{\ell-1} + 2\ell\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\bar{T}\right),$$
$$C_{\ell,1} = -\nu\,(b+2\ell+1) H_{\ell,1} + \nu\,\mathcal{J}_\ell \bar{u}_\parallel,$$
$$C_{\ell,2} = -\nu\,(b+2\ell+2)\, H_{\ell,2} + \nu\sqrt{2}\,\mathcal{J}_\ell \bar{T},$$
$$C_{\ell,m} = -\nu(b+2\ell+m) H_{\ell,m}, \qquad (m>2)$$

Field equations:

$$\sum_s \frac{Z_s^2 n_s}{\tau_s}\left(1 - \sum_{\ell=0}\mathcal{J}_\ell^2\right)\Phi = \sum_s Z_s n_s \sum_{\ell=0}\mathcal{J}_\ell G'_{\ell,0}$$

$$\left(k_\perp^2 + \frac{\beta_{\rm ref}}{2}\sum_s \frac{Z_s^2 n_s}{m_s}\sum_{\ell=0}\mathcal{J}_\ell^2\right) A_\parallel = \frac{\beta_{\rm ref}}{2}\sum_s Z_s n_s v_{ts}\sum_{\ell=0}\mathcal{J}_\ell G'_{\ell 1}$$

GX approach: Laguerre-Hermite pseudo-spectral velocity formulation

Hermite-Laguerre transform of gyrokinetic equation:

$$\frac{\partial G'_{\ell,m}}{\partial t} + N_{\ell,m} + v_{ts}\nabla_\parallel \left(\sqrt{m+1}\, H_{\ell,m+1} + \sqrt{m}\, H_{\ell,m-1}\right)$$

parallel phase-mixing (Landau damping)

$$+ v_{ts}\Big[-(\ell+1)\sqrt{m+1}\, H_{\ell,m+1} - \ell\sqrt{m+1}\, H_{\ell-1,m+1} + \ell\sqrt{m}\, H_{\ell,m-1} + (\ell+1)\sqrt{m}\, H_{\ell+1,m-1}\Big]\nabla_\parallel \ln B$$

$$+ i\omega_{ds}^{\kappa}\left[\sqrt{(m+1)(m+2)}\, H_{\ell,m+2} + (2m+1)H_{\ell,m} + \sqrt{m(m-1)}\, H_{\ell,m-2}\right]$$

$$+ i\omega_{ds}^{\nabla B}\left[(\ell+1)\, H_{\ell+1,m} + (2\ell+1)H_{\ell,m} + \ell\, H_{\ell-1,m}\right]$$

$$= D_{\ell,m} + C_{\ell,m}$$

Drive terms:

$$D_{\ell,m=0} = i\omega_* \left[\frac{1}{L_{ns}}\mathcal{J}_\ell + \frac{1}{L_{Ts}}\left[\ell\mathcal{J}_{\ell-1} + 2\ell\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\right]\Phi$$

$$D_{\ell,m=1} = v_{ts} i\omega_* \left[\frac{1}{L_{ns}}\mathcal{J}_\ell + \frac{1}{L_{Ts}}\left[\ell\mathcal{J}_{\ell-1} + (2\ell+1)\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\right] A_\parallel$$

$$D_{\ell,m=2} = \frac{1}{\sqrt{2}} i\omega_* \frac{1}{L_{Ts}}\mathcal{J}_\ell \Phi$$

$$D_{\ell,m=3} = v_{ts}\sqrt{\frac{3}{2}} i\omega_* \frac{1}{L_{Ts}}\mathcal{J}_\ell A_\parallel$$

$$D_{\ell,m>3} = 0,$$

Collisions (Dougherty model):

$$C_{\ell,0} = -\nu\,(b+2\ell+0)\, H_{\ell,0} + \nu\left(\sqrt{b}\,(\mathcal{J}_\ell + \mathcal{J}_{\ell-1})\,\bar{u}_\perp + 2\left[\ell\mathcal{J}_{\ell-1} + 2\ell\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\bar{T}\right),$$

$$C_{\ell,1} = -\nu\,(b+2\ell+1) H_{\ell,1} + \nu\,\mathcal{J}_\ell \bar{u}_\parallel,$$

$$C_{\ell,2} = -\nu\,(b+2\ell+2)\, H_{\ell,2} + \nu\sqrt{2}\,\mathcal{J}_\ell \bar{T},$$

$$C_{\ell,m} = -\nu(b+2\ell+m) H_{\ell,m}, \qquad (m>2)$$

Field equations:

$$\sum_s \frac{Z_s^2 n_s}{\tau_s}\left(1 - \sum_{\ell=0}\mathcal{J}_\ell^2\right)\Phi = \sum_s Z_s n_s \sum_{\ell=0}\mathcal{J}_\ell G'_{\ell,0}$$

$$\left(k_\perp^2 + \frac{\beta_{\rm ref}}{2}\sum_s \frac{Z_s^2 n_s}{m_s}\sum_{\ell=0}\mathcal{J}_\ell^2\right) A_\parallel = \frac{\beta_{\rm ref}}{2}\sum_s Z_s n_s v_{ts}\sum_{\ell=0}\mathcal{J}_\ell G'_{\ell 1}$$

GX approach: Laguerre-Hermite pseudo-spectral velocity formulation

Hermite-Laguerre transform of gyrokinetic equation:

$$\frac{\partial G'_{\ell,m}}{\partial t} + N_{\ell,m} + v_{ts}\nabla_\parallel \left(\sqrt{m+1}\, H_{\ell,m+1} + \sqrt{m}\, H_{\ell,m-1}\right)$$

parallel phase-mixing (Landau damping)

$$+ v_{ts}\Big[-(\ell+1)\sqrt{m+1}\, H_{\ell,m+1} - \ell\sqrt{m+1}\, H_{\ell-1,m+1}$$
$$+ \ell\sqrt{m}\, H_{\ell,m-1} + (\ell+1)\sqrt{m}\, H_{\ell+1,m-1}\Big]\nabla_\parallel \ln B$$
$$+ i\omega_{ds}^{\kappa}\left[\sqrt{(m+1)(m+2)}\, H_{\ell,m+2} + (2m+1)H_{\ell,m} + \sqrt{m(m-1)}\, H_{\ell,m-2}\right]$$

toroidal phase-mixing

$$+ i\omega_{ds}^{\nabla B}\left[(\ell+1)\, H_{\ell+1,m} + (2\ell+1)H_{\ell,m} + \ell\, H_{\ell-1,m}\right]$$
$$= D_{\ell,m} + C_{\ell,m}$$

Drive terms:

$$D_{\ell,m=0} = i\omega_* \left[\frac{1}{L_{ns}}\mathcal{J}_\ell + \frac{1}{L_{Ts}}\left[\ell\mathcal{J}_{\ell-1} + 2\ell\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\right]\Phi$$
$$D_{\ell,m=1} = v_{ts} i\omega_* \left[\frac{1}{L_{ns}}\mathcal{J}_\ell + \frac{1}{L_{Ts}}\left[\ell\mathcal{J}_{\ell-1} + (2\ell+1)\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\right] A_\parallel$$
$$D_{\ell,m=2} = \frac{1}{\sqrt{2}} i\omega_* \frac{1}{L_{Ts}}\mathcal{J}_\ell \Phi$$
$$D_{\ell,m=3} = v_{ts}\sqrt{\frac{3}{2}} i\omega_* \frac{1}{L_{Ts}}\mathcal{J}_\ell A_\parallel$$
$$D_{\ell,m>3} = 0,$$

Collisions (Dougherty model):

$$C_{\ell,0} = -\nu\,(b+2\ell+0)\, H_{\ell,0}$$
$$+ \nu\left(\sqrt{b}\,(\mathcal{J}_\ell + \mathcal{J}_{\ell-1})\,\bar{u}_\perp + 2\left[\ell\mathcal{J}_{\ell-1} + 2\ell\mathcal{J}_\ell + (\ell+1)\mathcal{J}_{\ell+1}\right]\bar{T}\right),$$
$$C_{\ell,1} = -\nu\,(b+2\ell+1) H_{\ell,1} + \nu\,\mathcal{J}_\ell \bar{u}_\parallel,$$
$$C_{\ell,2} = -\nu\,(b+2\ell+2)\, H_{\ell,2} + \nu\sqrt{2}\,\mathcal{J}_\ell \bar{T},$$
$$C_{\ell,m} = -\nu(b+2\ell+m) H_{\ell,m}, \qquad (m>2)$$

Field equations:

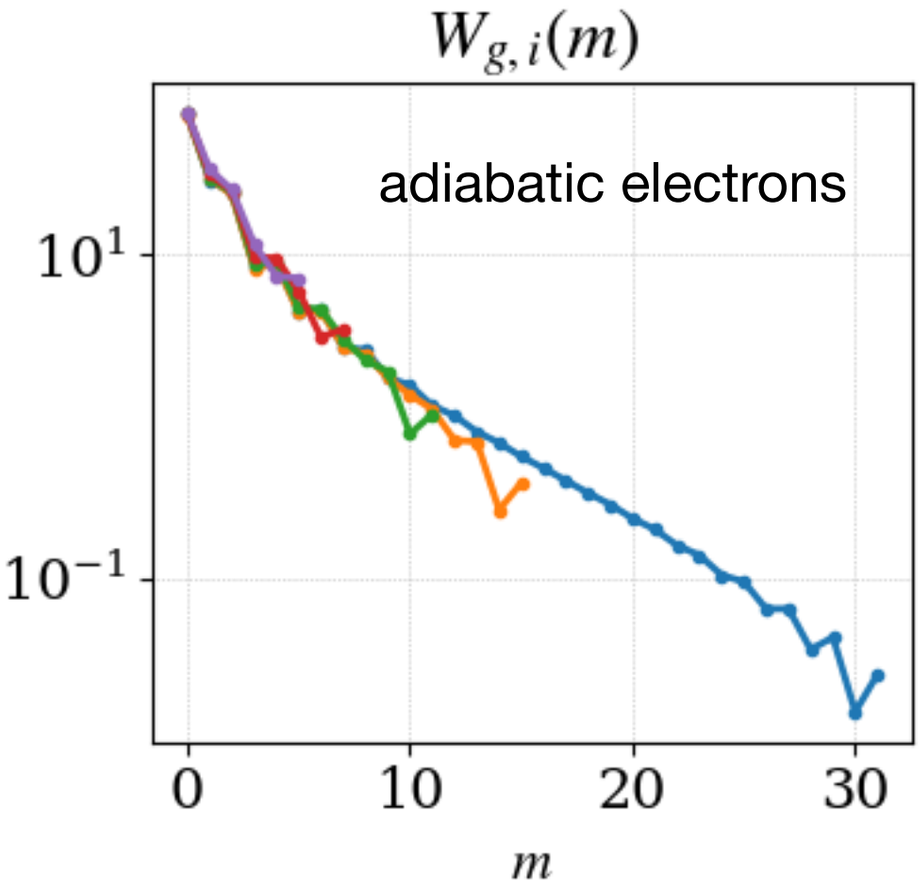
$$\sum_s \frac{Z_s^2 n_s}{\tau_s}\left(1 - \sum_{\ell=0}\mathcal{J}_\ell^2\right)\Phi = \sum_s Z_s n_s \sum_{\ell=0}\mathcal{J}_\ell G'_{\ell,0}$$

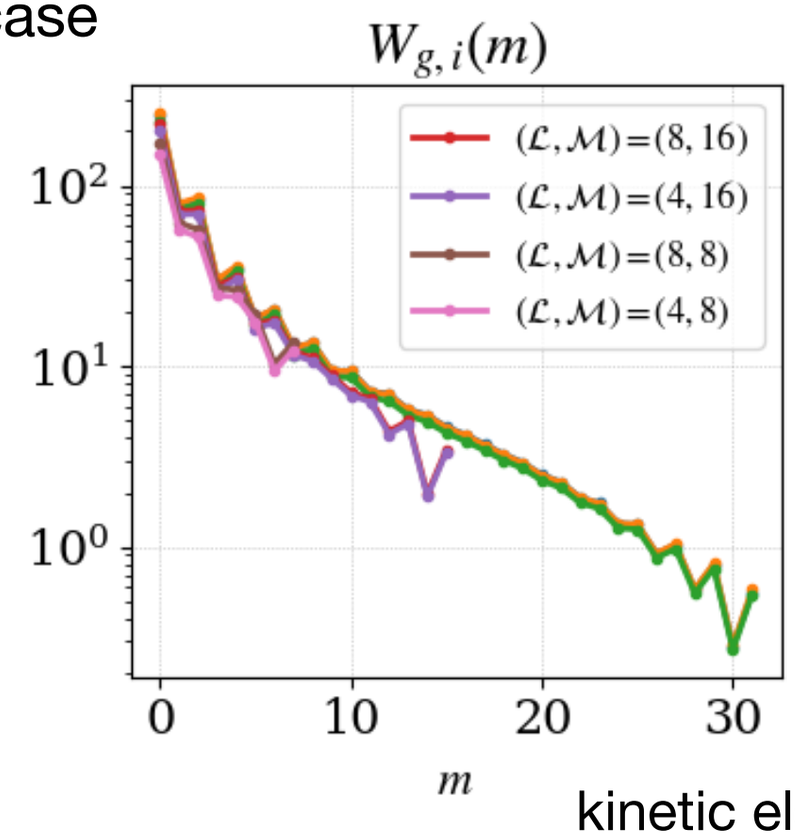
$$\left(k_\perp^2 + \frac{\beta_{\rm ref}}{2}\sum_s \frac{Z_s^2 n_s}{m_s}\sum_{\ell=0}\mathcal{J}_\ell^2\right) A_\parallel = \frac{\beta_{\rm ref}}{2}\sum_s Z_s n_s v_{ts}\sum_{\ell=0}\mathcal{J}_\ell G'_{\ell 1}$$

Hermite spectra

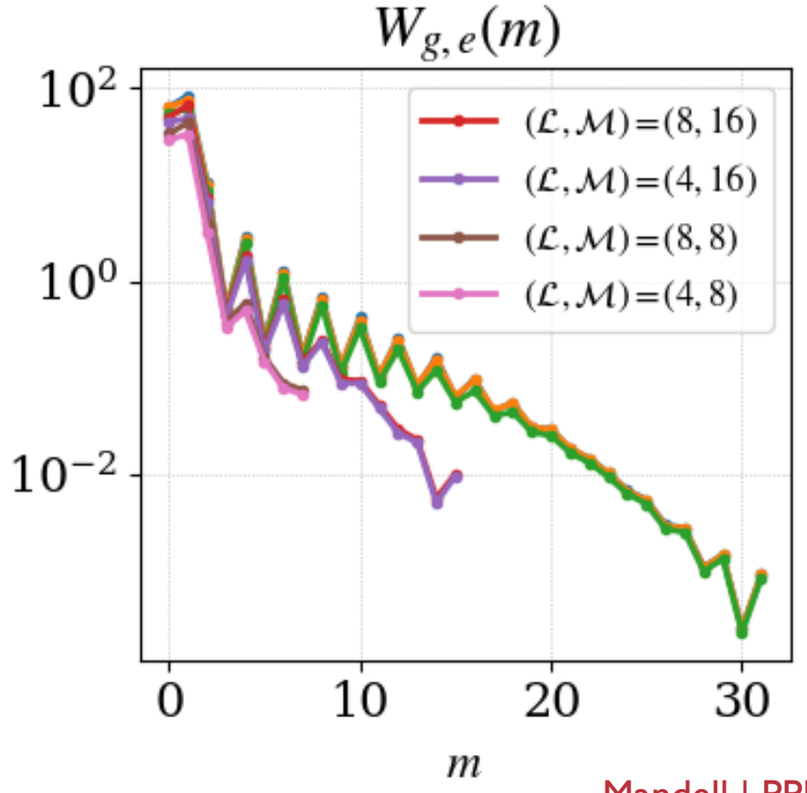
- Need more Hermite resolution in kinetic electron case

$$W_{g,s}(m) = \sum_{\ell} \int |G^{s}_{\ell,m}|^2 \, dx \, dy \, dz$$





kinetic electrons



Multi-GPU scaling

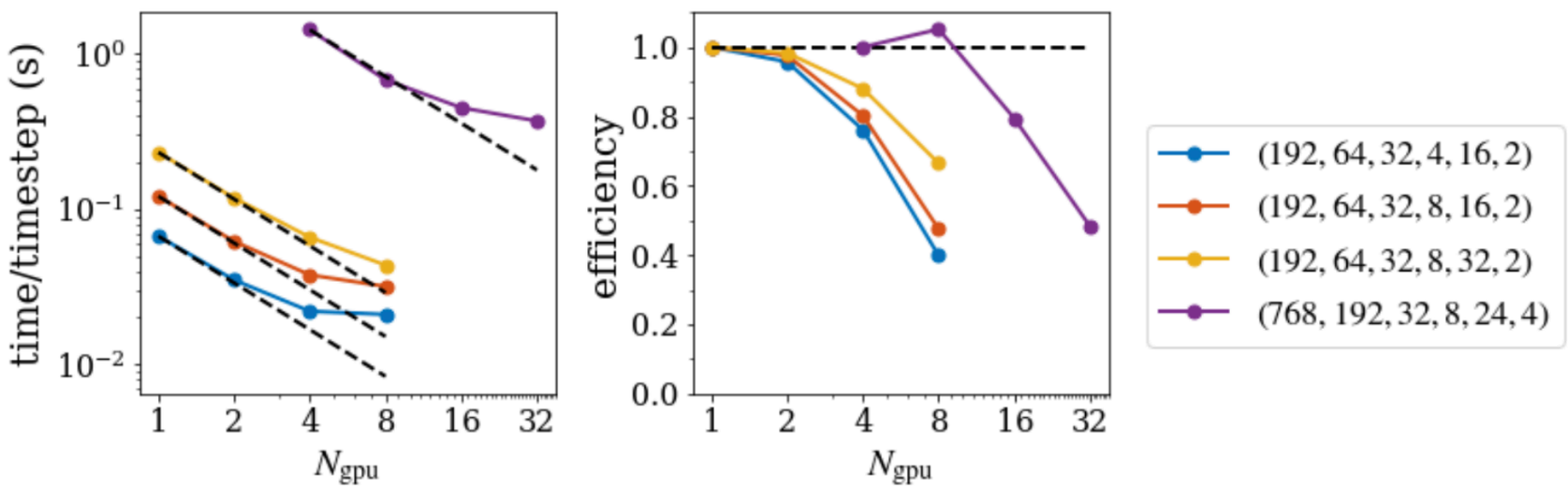


FIGURE 11. Strong scaling study for the nonlinear CBC with kinetic electrons showing the time per timestep (a) and the scaling efficiency (b) as a function of number of GPUs used, with fixed resolution for each curve. In each case the ideal scaling is shown with a dashed line. The resolution parameters are listed in the legend as $(N_x, N_y, N_z, \mathcal{L}, \mathcal{M}, N_{\text{species}})$.

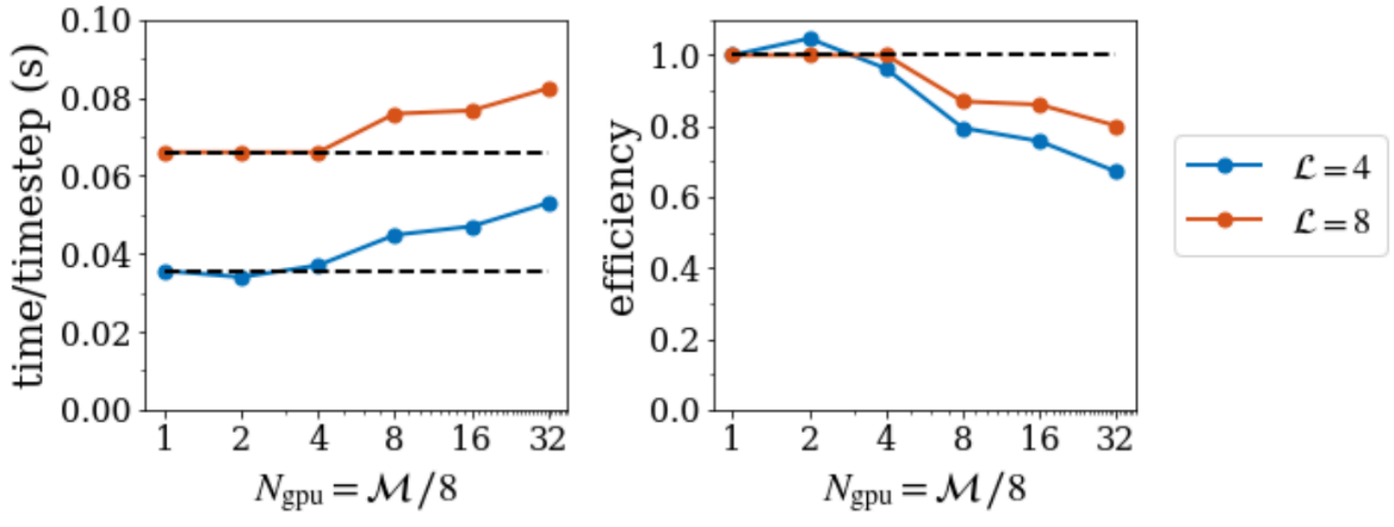
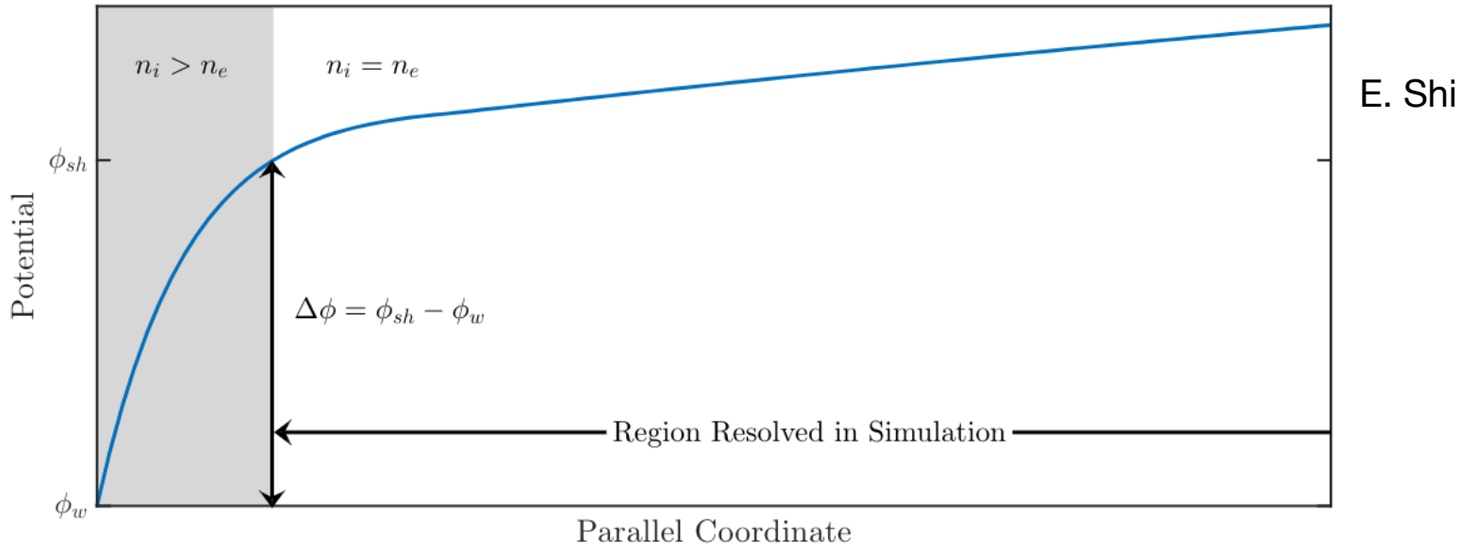


FIGURE 12. Weak scaling study for the nonlinear CBC with kinetic electrons showing the time per timestep (a) and the scaling efficiency (b) as a function of number of GPUs used, with the number of Hermite modes $\mathcal{M}$ scaling with the number of GPUs as $\mathcal{M} = 8N_{\text{gpu}}$. We show results for both $\mathcal{L} = 4$ (blue) and $\mathcal{L} = 8$ (red). The ideal scaling is shown with dashed lines. The other resolution parameters are: $N_x = 192,\ N_y = 64,\ N_z = 32,\ N_{\text{species}} = 2$.

Conducting-sheath boundary condition



E. Shi

- Need to model non-neutral sheath using BCs (GK assumes quasi-neutrality, cannot resolve sheath)

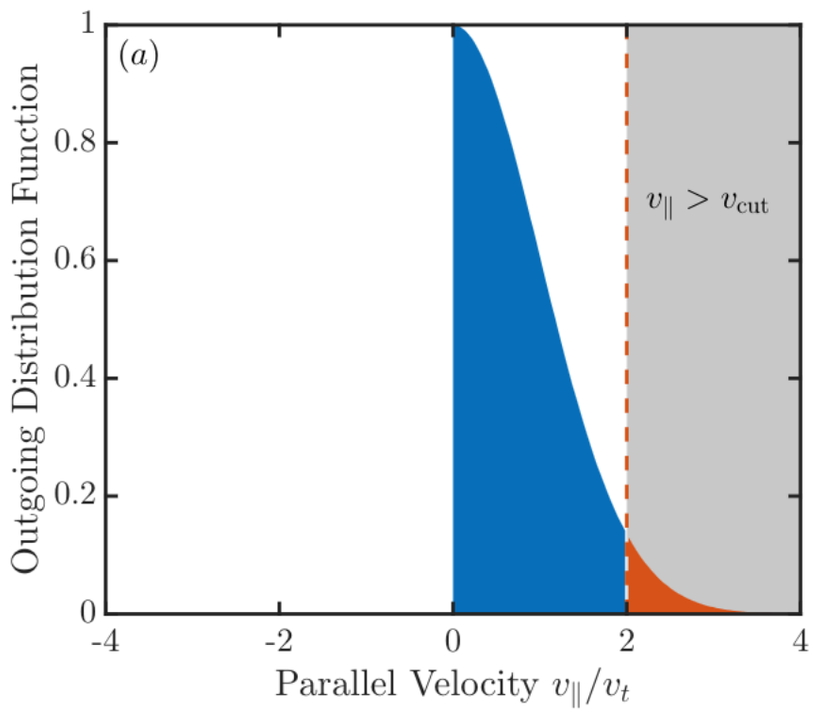
- Sheath potential should reflect low energy electrons

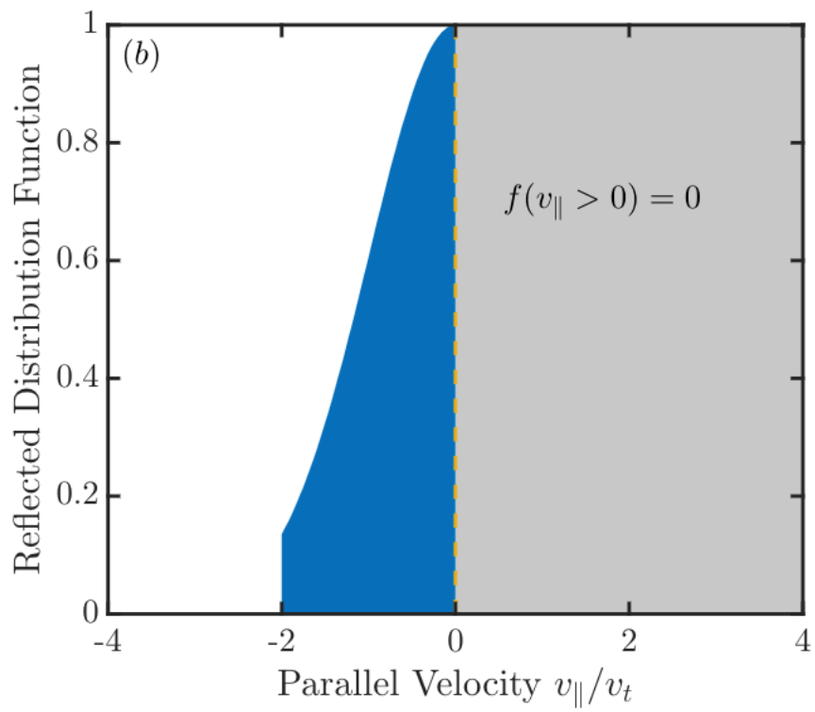
- Solve Poisson equation on z boundary to get $\phi_{sh}(x,y) \doteq \phi(z = z_{sh})$, then use $\Delta\phi = \phi_{sh} - \phi_w$ to reflect electrons with $mv_{\parallel}^2/2 < |e|\Delta\phi$

$$-\nabla_\perp \cdot \sum_s \frac{m_s n_{0s}}{B^2} \nabla_\perp \phi(z = z_{sh}) = \sum_s q_s \int d^3v \; f_s(z = z_{sh})$$

- Potential self-consistently relaxes to ambipolar-parallel-outflow state, and allows local currents in and out of wall (unlike “logical” sheath model)

Sheath boundary condition for electrons





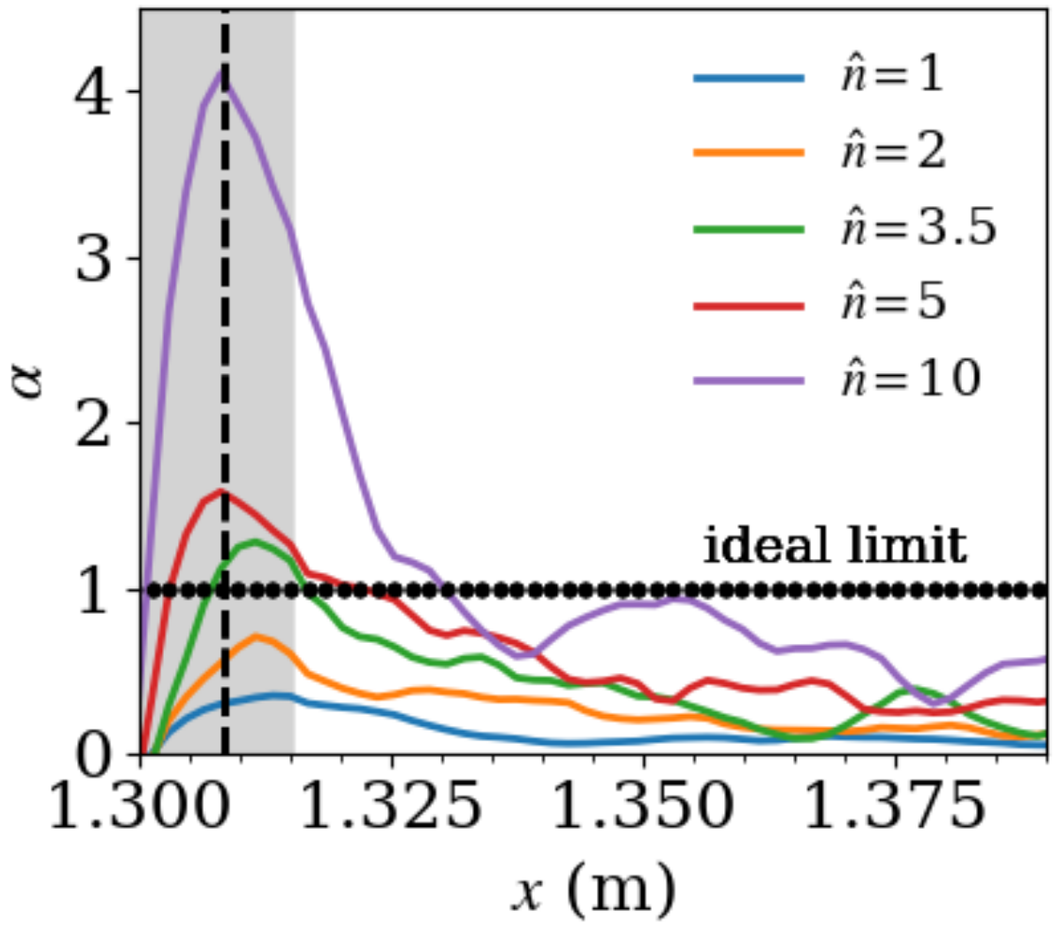
E. Shi

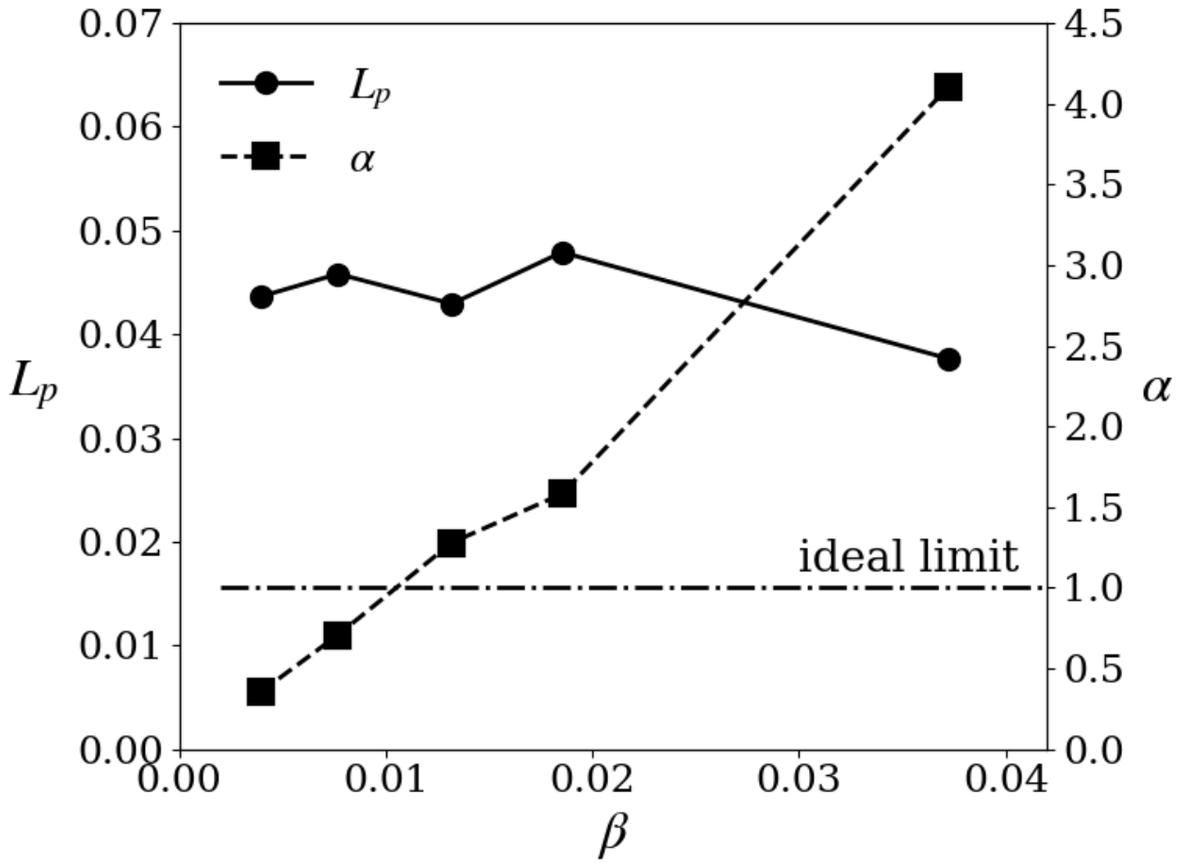
(a) Outgoing electrons with $v_{||} > v_{cut} = \sqrt{2e\Delta\phi/m}$ are lost into the wall

(b) Rest of outgoing electrons $0 < v_{||} < v_{cut}$ are reflected back into plasma

Ions: Assuming positive sheath potential (relative to wall), all ions are lost

Exceeding the ideal ballooning limit





- Examining the ideal ballooning parameter, $\alpha = L_{\parallel}^2 \beta/(\pi^2 R L_p)$, the ideal limit at $\alpha = 1$ is exceeded in the steep-gradient region of the high-beta simulations

- Meanwhile, the pressure gradient scale-length (L_p, where $\nabla p \sim p/L_p$) is relatively constant as β is increased

- This is surprising! Might expect that as heating is added and the pressure profile approaches the ballooning limit, the turbulence will become stronger and broaden the profile so as to increase L_p to keep $\alpha \propto \beta/L_p < 1$ below the ballooning limit

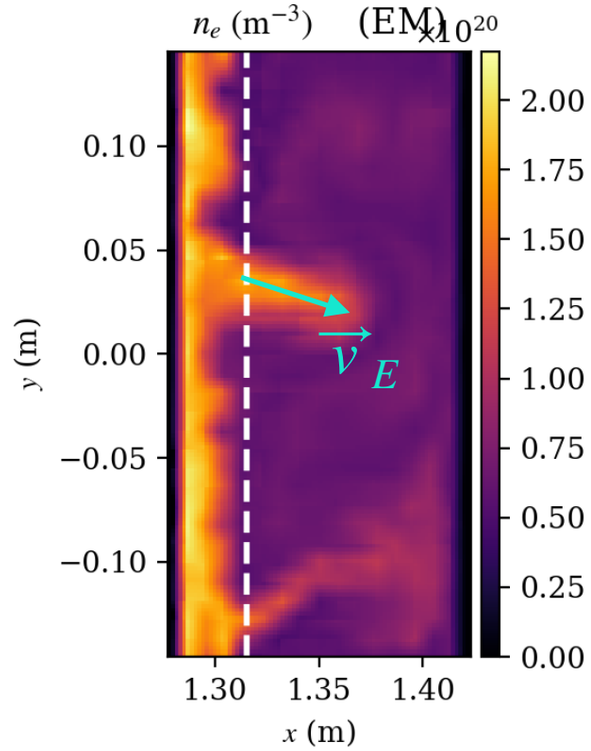
    - Theoretical explanation for this surprising result is subject of next several slides

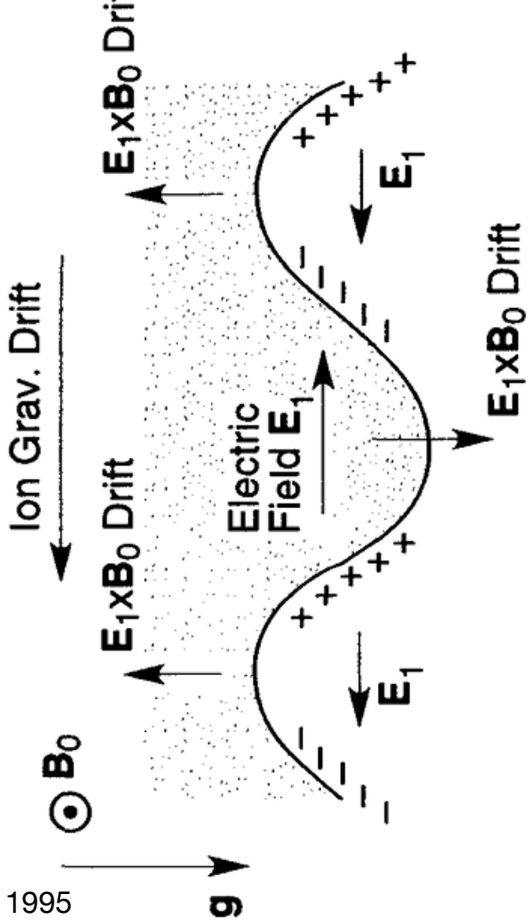
Interchange instability

- The main instabilities in the SOL are expected to be of the interchange type
- Bad curvature in a tokamak can give an effective gravity from the curvature drift, resulting in the interchange (or flute) instability

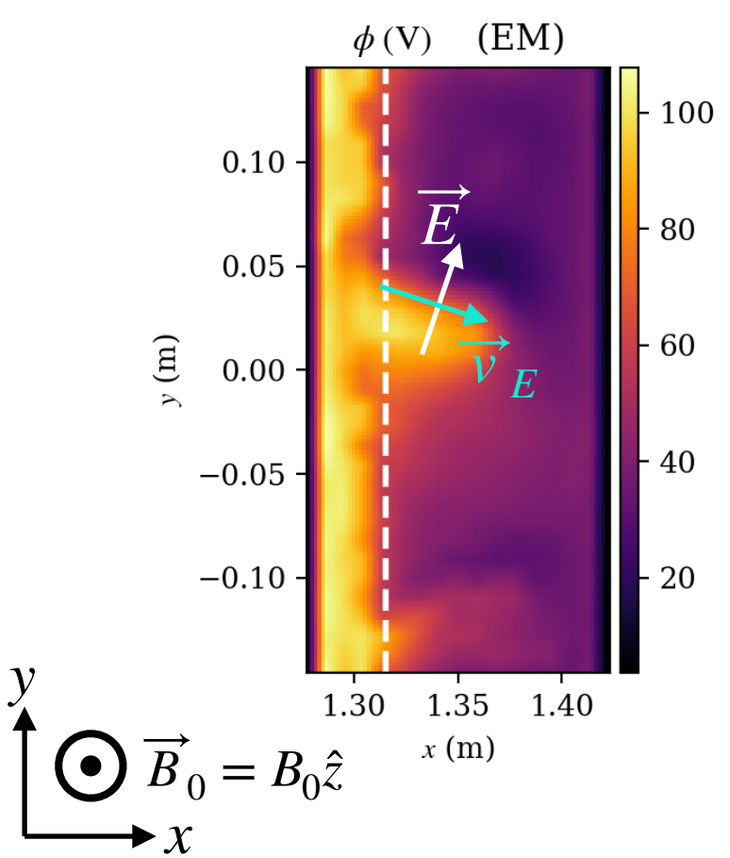
$$\mathbf{v}_\mathrm{d} = \frac{M}{e}\left(\frac{v_\perp^2}{2} + v_\parallel^2\right)\frac{\mathbf{R}_\mathrm{c}\times\mathbf{B}}{R_\mathrm{c}^2 B^2} \qquad g = \left(\frac{v_\perp^2}{2} + v_\parallel^2\right)\frac{1}{R_\mathrm{c}}$$

- Ideal interchange instability gives $k_\parallel = 0$ modes with $\gamma_\mathrm{int} = \dfrac{\sqrt{2}c_s}{\sqrt{RL_p}}$





Goldston & Rutherford, 1995



Ideal MHD interchange-ballooning equation

- If we also allow the field lines to bend, we will get ballooning-type instabilities. Line-bending is stabilizing because it takes energy to bend the field lines.

- Ideal MHD interchange-ballooning equation with diamagnetic effects in helical geometry without shear:

$$\frac{\partial^2 \varphi}{\partial z^2} + \alpha \frac{\pi^2}{L_z^2} \left(\frac{-\gamma(\gamma - i\omega_*)}{\gamma_{\rm int}^2} + \frac{k_y^2}{k_\perp^2} \right) \varphi = 0$$

$$\alpha = \frac{L_\parallel^2 \beta}{R L_p} \qquad \gamma_{\rm int} = \frac{\sqrt{2} c_s}{\sqrt{R L_p}}$$

$$\omega_* = -k_y \rho_i v_{ti} / L_p$$

- This equation contains the ideal interchange mode: take $k_\parallel = \omega_* = k_x = 0$, which gives $\gamma = \gamma_{\rm int}$

- To get something other than ideal interchange, need something with finite $k_\parallel$, at least somewhere

  - If field lines are perfectly tied at the ends, so that $\varphi(z = \pm L_z/2) = 0$, the allowable parallel wavelengths are $k_\parallel = \ell \pi / L_z$, which gives (taking $\ell = 1$ and the limit $\omega_* = k_x = 0$)

$$\gamma = \gamma_{\rm int} \sqrt{1 - 1/\alpha}$$

    This is the standard ideal ballooning mode, which is unstable for $\alpha > 1$

Modification due to sheath boundary conditions

- Now consider the case where the field lines end on conducting plates, so that a sheath forms

- At zeroth order we have the Bohm sheath potential with $j_\parallel = 0$, but a fluctuation in the sheath potential will result in a fluctuating electron current into the sheath [1,2]

$$\tilde{j}_{\parallel e} = \sigma e \int_{v_{c0}}^{v_{c0}+\tilde{v}_c} dv_\parallel \, v_\parallel F_e = \sigma n_e e c_s \frac{e\tilde{\phi}}{T_{e0}} \qquad (\sigma = \pm 1)$$

- We can use this as a boundary condition for the ballooning equation

$$\mu_0 j_\parallel = k_\perp^2 A_\parallel$$

$$-\frac{\partial \varphi}{\partial z} - \frac{\partial A_\parallel}{\partial t} = 0 \qquad \Rightarrow \qquad -\frac{\partial \varphi}{\partial z}\Big|_{\pm L_z/2} = \gamma \frac{\mu_0}{k_\perp^2} j_\parallel \Big|_{\pm L_z/2} = \pm\gamma \frac{c_s}{k_\perp^2 \rho_s^2 v_A^2} \varphi \Big|_{\pm L_z/2}$$

- Now the system for the interchange-ballooning mode with sheath effects is

$$\frac{\partial^2 \varphi}{\partial z^2} + \alpha \frac{\pi^2}{L_z^2} \left(-\hat{\gamma}(\hat{\gamma} - i\hat{\omega}_*) + \frac{k_y^2}{k_\perp^2} \right) \varphi = 0$$

$$-\frac{\partial \varphi}{\partial z}\Big|_{\pm L_z/2} = \pm \hat{\gamma} \hat{\omega}_s \frac{\alpha}{2} \frac{\pi^2}{L_z} \varphi \Big|_{\pm L_z/2}$$

$$\hat{\gamma} = \frac{\gamma}{\gamma_{\text{int}}} \qquad \alpha = \frac{L_\parallel^2 \beta}{R L_p}$$

$$\hat{\omega}_* = \frac{\omega_*}{\gamma_{\text{int}}} = \frac{-k_y \rho_s}{1 + \frac{T_e}{T_i}} \sqrt{\frac{R}{2L_p}}$$

$$\hat{\omega}_s = \frac{\omega_s}{\gamma_{\text{int}}} = \frac{2c_s}{k_\perp^2 \rho_s^2 L_\parallel \gamma_{\text{int}}}$$

where we have defined a new parameter $\hat{\omega}_s$, which is effectively a measure of sheath conductivity

- Small $\hat{\omega}_s$ (insulating sheath) allows field lines to slip at the sheath entrance

- Large $\hat{\omega}_s$ (conducting sheath) gives strong line tying to the endplates

[1] Kerner et al 1997 Plasma Phys. Control. Fusion 39 1101
[2] Myra et al 1997 Phys. Plasmas 4 1330

Sheath-modified interchange-ballooning instability

$$\frac{\partial^2\varphi}{\partial z^2} + \alpha\frac{\pi^2}{L_z^2}\left(-\hat{\gamma}(\hat{\gamma} - i\hat{\omega}_*) + \frac{k_y^2}{k_\perp^2}\right)\varphi = 0, \qquad \hat{\omega}_s = \frac{\omega_s}{\gamma_{\rm int}} = \frac{2c_s}{k_\perp^2\rho_s^2 L_\parallel \gamma_{\rm int}}$$ ~ sheath conductivity

$$-\frac{\partial\varphi}{\partial z}\Big|_{\pm L_z/2} = \pm\hat{\gamma}\hat{\omega}_s\frac{\alpha}{2}\frac{\pi^2}{L_z}\varphi\Big|_{\pm L_z/2}$$

- Let us examine various limits in the cold ion ($\omega_* \ll 1$) and local ($k_x \ll k_y$) limit.

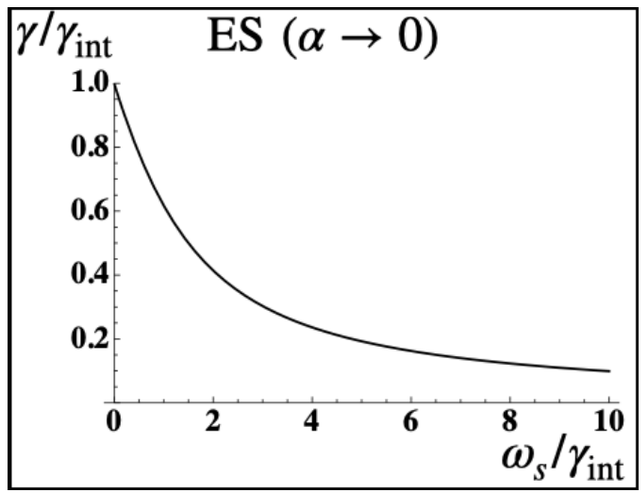
- Ideal interchange, $\hat{\gamma} = 1$, is recovered in the zero sheath current (high sheath resistivity) limit ($\omega_s \ll \gamma_{\rm int}$), which results in $k_\parallel = 0$ from the boundary condition

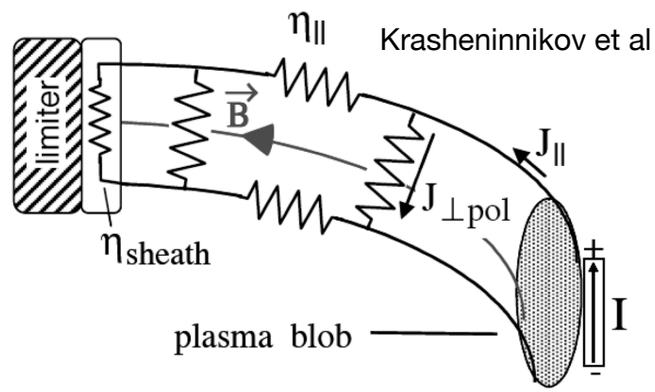
  - This is effectively the logical-sheath boundary condition (which imposes no net sheath current, $j_\parallel = 0$)

- Electrostatic limit ($\alpha \to 0$, no line bending)

  - The dispersion relation reduces to

$$\hat{\gamma} = -\frac{\hat{\omega}_s}{2} + \sqrt{\frac{\hat{\omega}_s^2}{4} + 1}$$





  - Sheath current stabilizes the interchange mode (connection to sheath shorts out charge polarization in pressure perturbation)

  - This sheath stabilization requires current fluctuations into sheath, so it is not captured by logical-sheath ($j_\parallel = 0$) BC

Sheath-modified interchange-ballooning instability

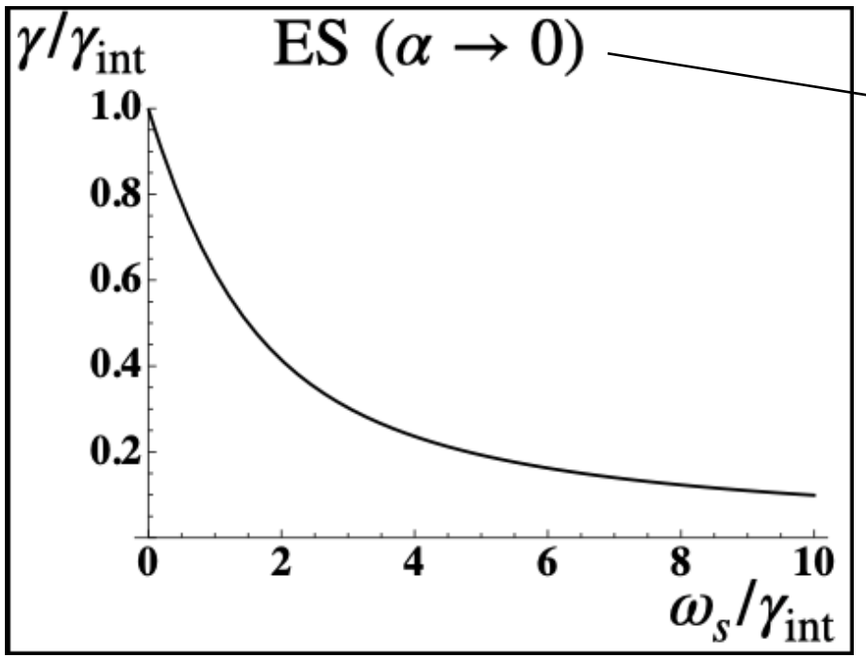
$$\frac{\partial^2\varphi}{\partial z^2} + \alpha\frac{\pi^2}{L_z^2}\left(-\hat{\gamma}(\hat{\gamma} - i\hat{\omega}_*) + \frac{k_y^2}{k_\perp^2}\right)\varphi = 0, \qquad \hat{\omega}_s = \frac{\omega_s}{\gamma_{\rm int}} = \frac{2c_s}{k_\perp^2\rho_s^2 L_\parallel \gamma_{\rm int}}$$ ~ sheath conductivity

$$-\frac{\partial\varphi}{\partial z}\Big|_{\pm L_z/2} = \pm\hat{\gamma}\hat{\omega}_s\frac{\alpha}{2}\frac{\pi^2}{L_z}\varphi\Big|_{\pm L_z/2}$$

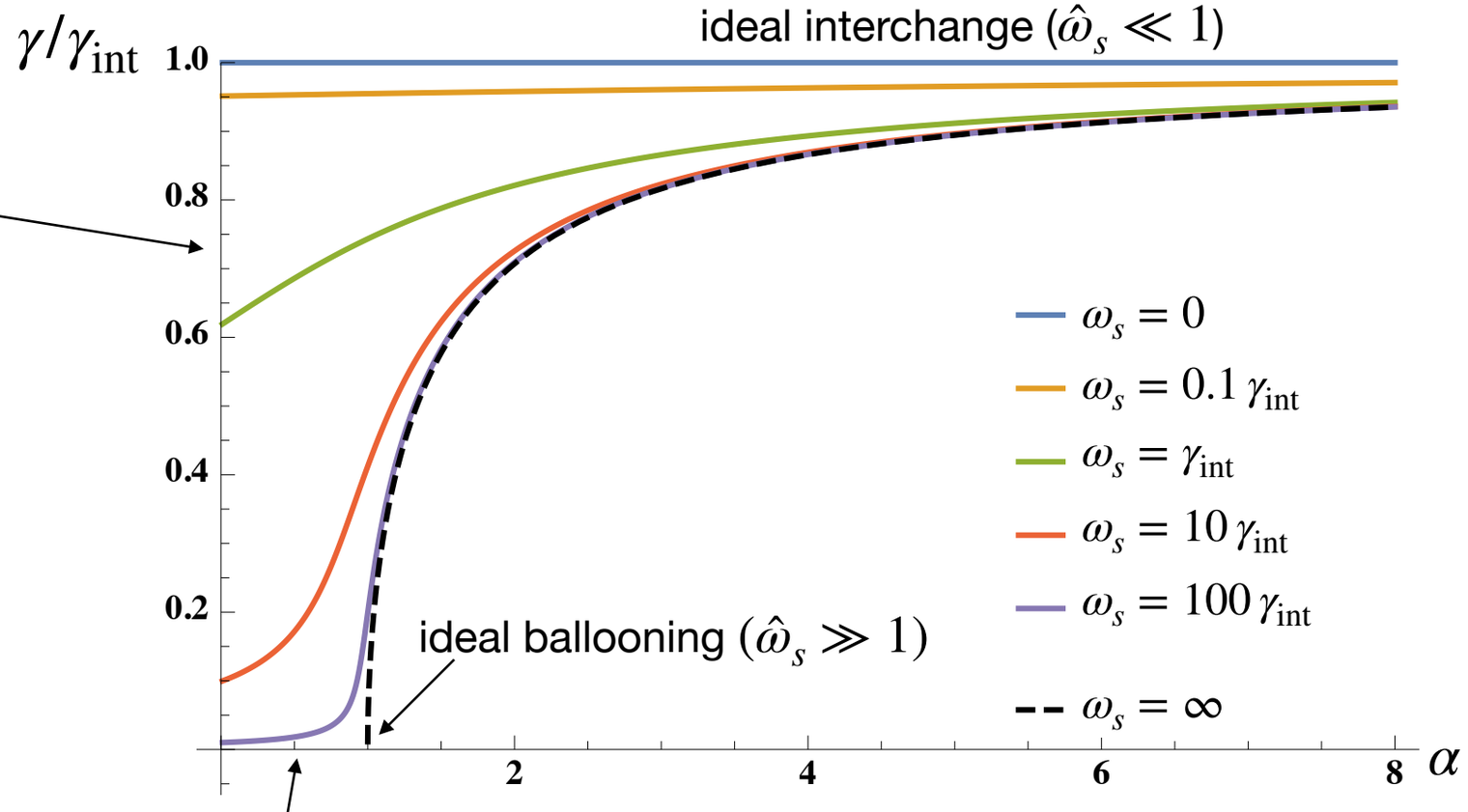
- Let us examine various limits in the cold ion ($\omega_* \ll 1$) and local ($k_x \ll k_y$) limit

- Ideal MHD (ballooning) limit: take $\hat{\omega}_s \to \infty$, which gives perfect line-tying. This gives $k_\parallel = \ell\pi/L_z$, and we recover

$$\hat{\gamma} = \sqrt{1 - 1/\alpha}$$



sheath stabilization of interchange in electrostatic limit



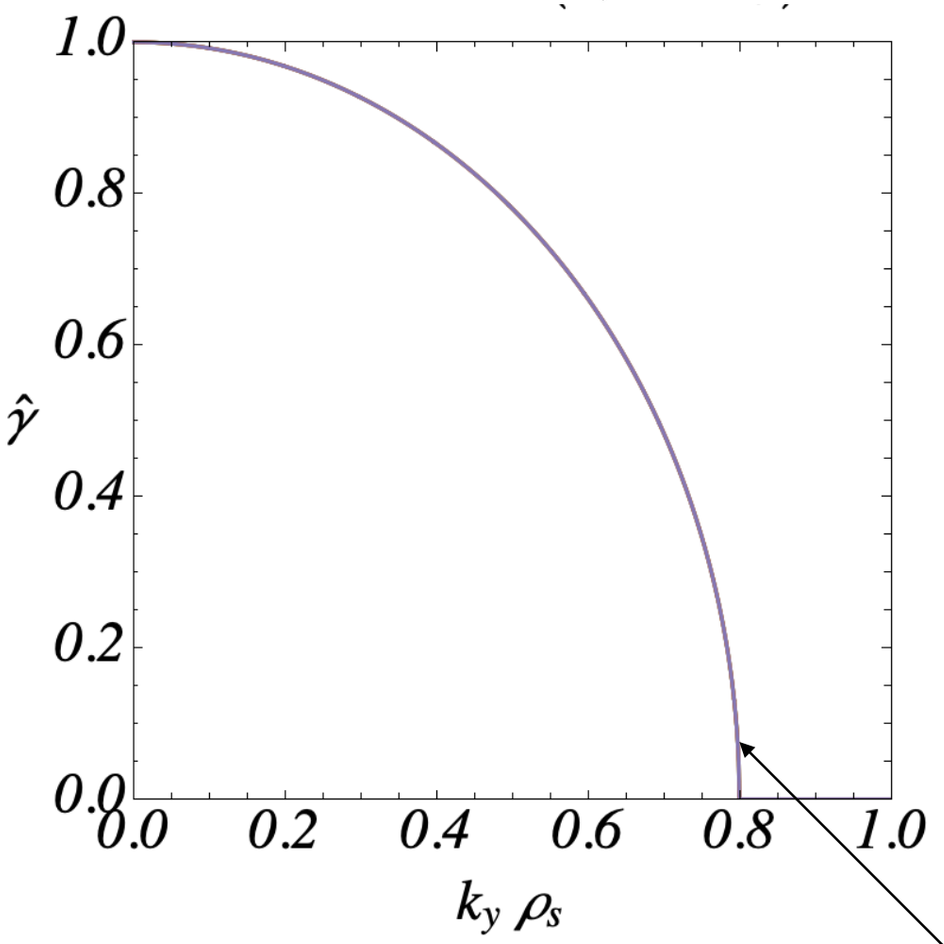
for finite $\hat{\omega}_s$, there is always some interchange-like instability due to incomplete line-tying

Including diamagnetic effects

- FLR effects from diamagnetic terms (ω_*) give stabilization at large $k_y\rho_s$
  - Showing $\hat{\omega}_* = -2.5 k_y \rho_s$, which comes from simulation parameters
- Finite sheath current ($\hat{\omega}_s$) stabilizes interchange-like mode for small $k_y\rho_s$, but only below the ballooning limit ($\alpha \leq 1$)
- From this, would still expect small $k_y\rho_s$ modes to drive significant transport upon crossing $\alpha = 1$

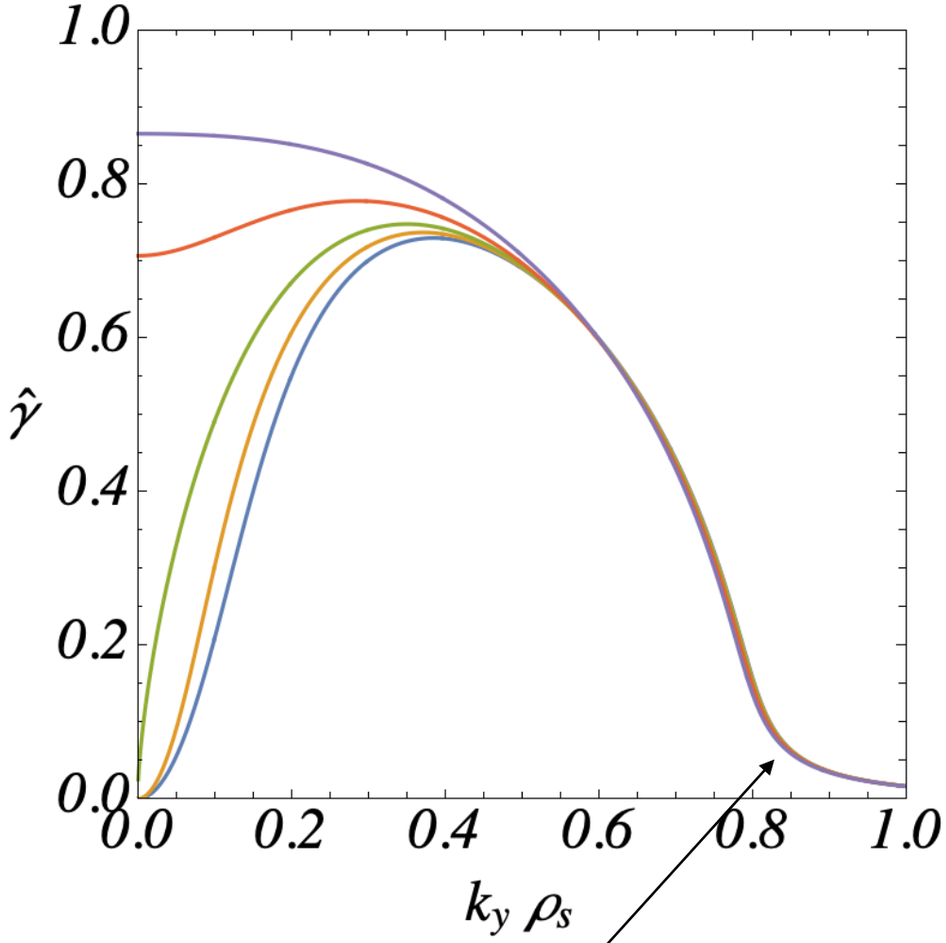
Logical ($j_{||} = 0$) sheath

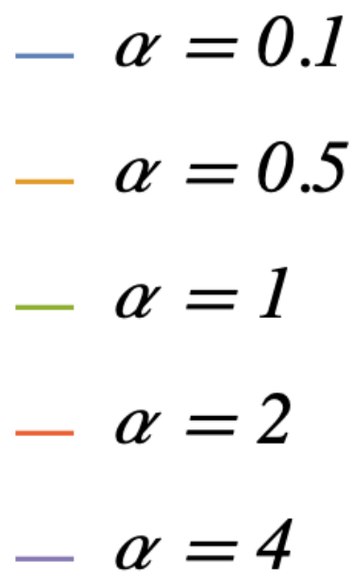
Local limit ($k_x \ll k_y$), $\hat{\omega}_s = 0$



Conducting ($j_{||}$ = finite) sheath

Local limit ($k_x \ll k_y$), $\hat{\omega}_s = 0.05/(k_\perp^2 \rho_s^2)$





Diamagnetic stabilization (ω_*) at large $k_y\rho_s$

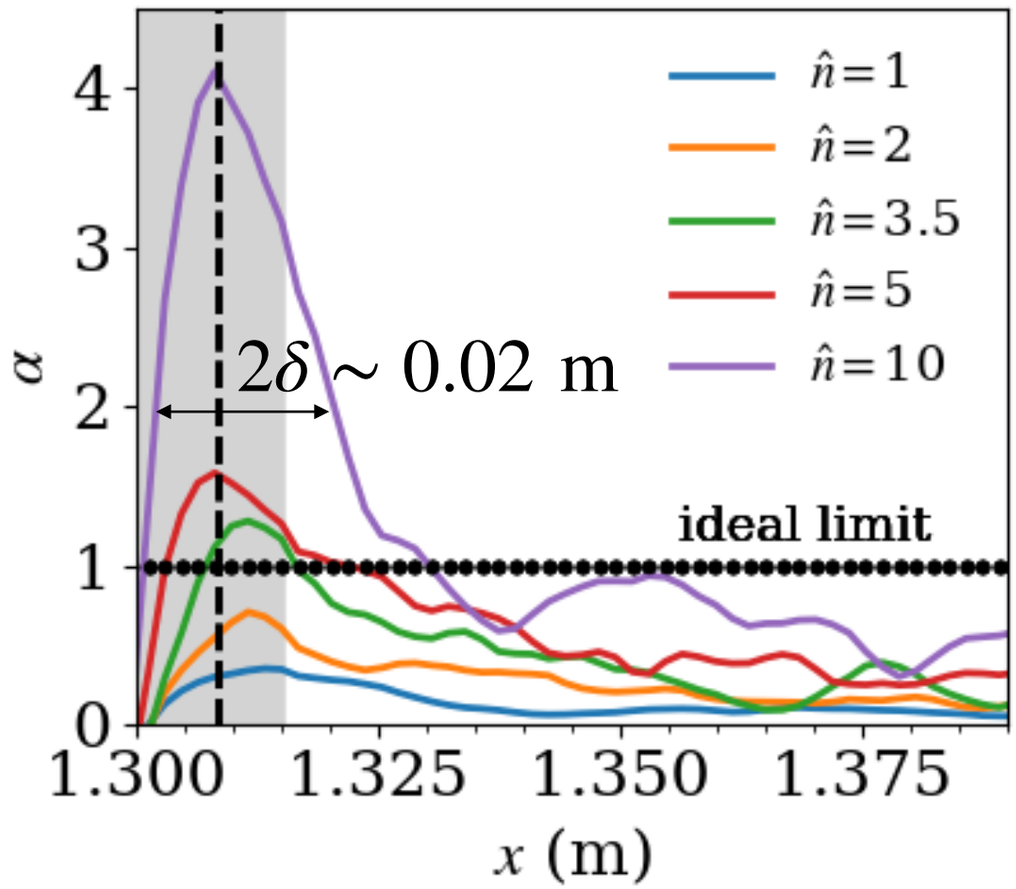
Including non-local effects

- Standard ballooning mode theory is in the radially local limit, where the gradients varying over a width $\sim \delta$ can be assumed constant on the scale of the mode, i.e. $k_y \delta \gg 1$

- The steep pressure gradient region in our simulations is relatively narrow, so non-local effects accounting for the radial variation of $\alpha(x)$ may be important

  - The eigenfunction needs to localize in the narrow steep-gradient region

  - Rogers & Drake (1999)† showed that outside the steep-gradient region, the eigenfunction has the form $\varphi(x) = \exp(-|k_y|x)$, which can stabilize the ballooning mode

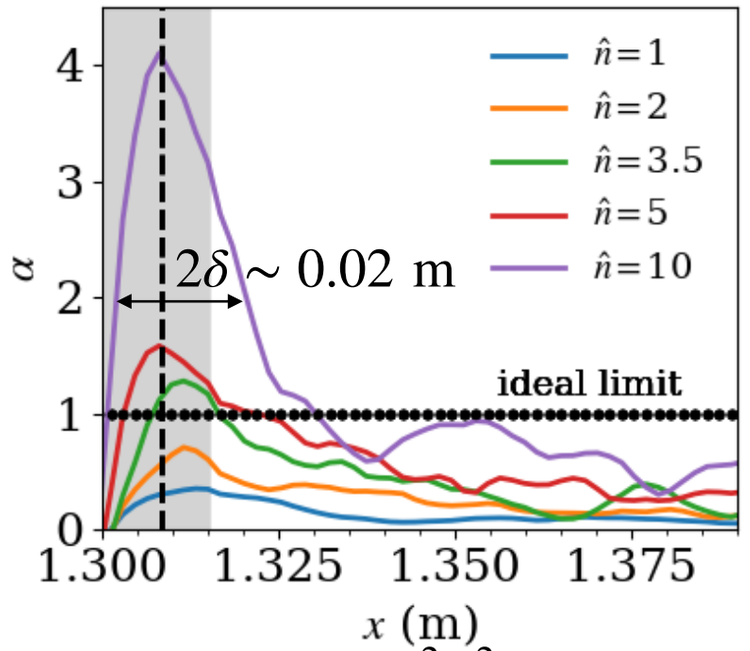


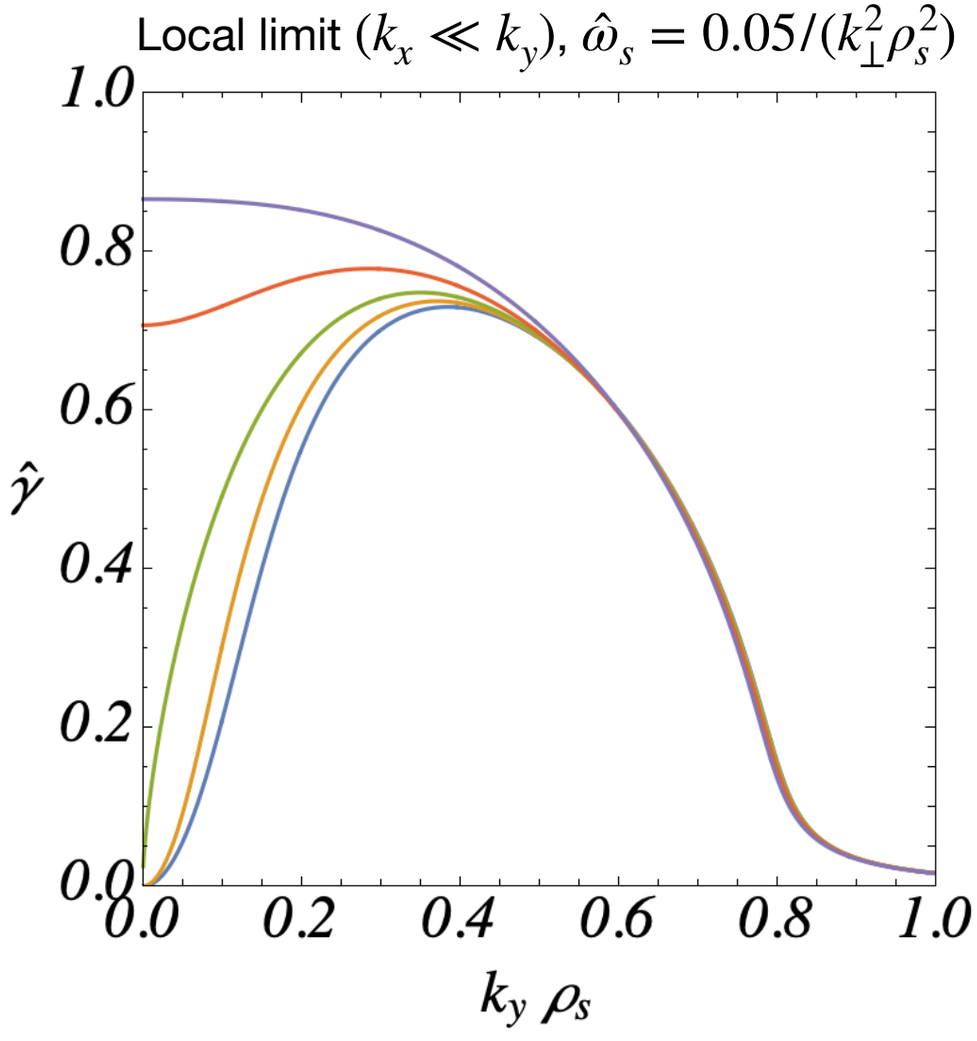
† Rogers & Drake, Phys. Plasmas (1999)

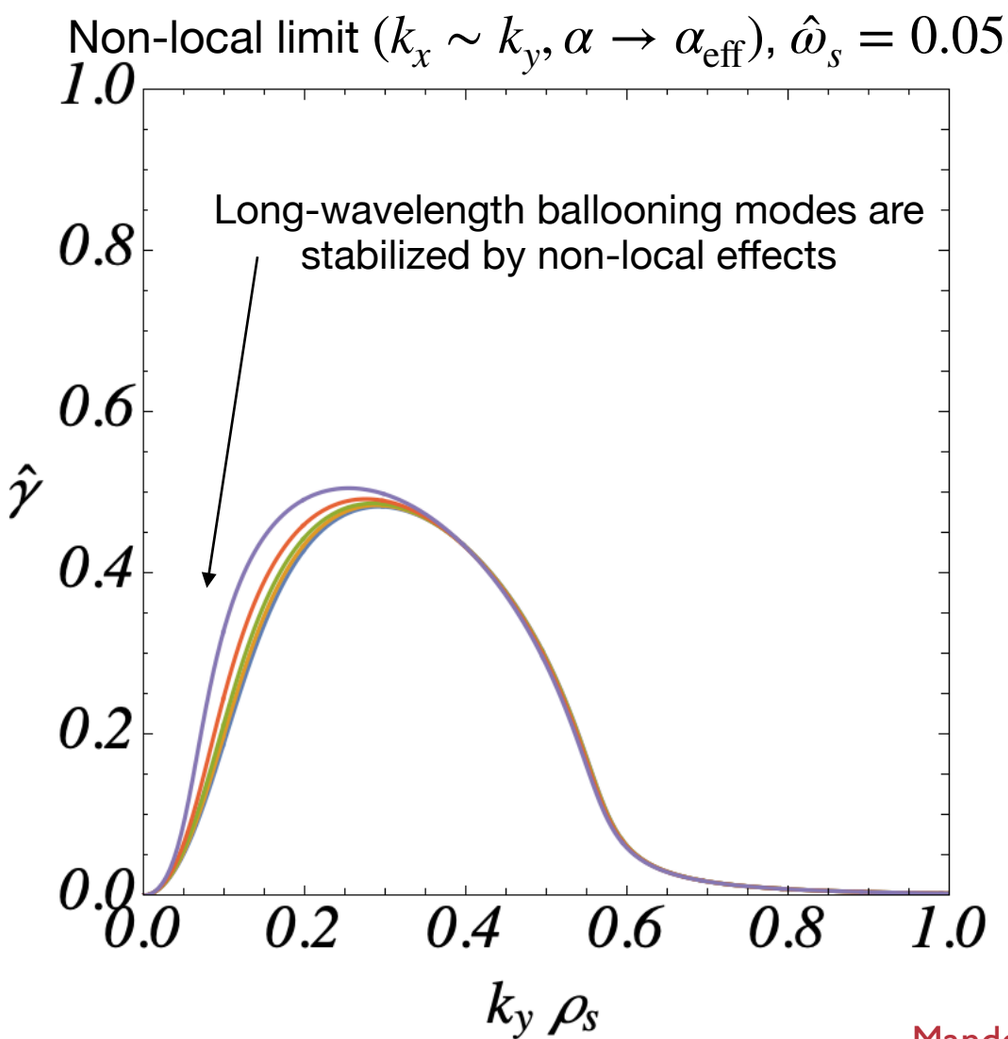
Including non-local effects

- Using this to compute an eigenfunction-averaged α_{eff}, and taking $k_x \sim k_y$, gives strong stabilization of long-wavelength ballooning modes (with finite sheath current)

- Weak dependence of instability on α is consistent with simulation results



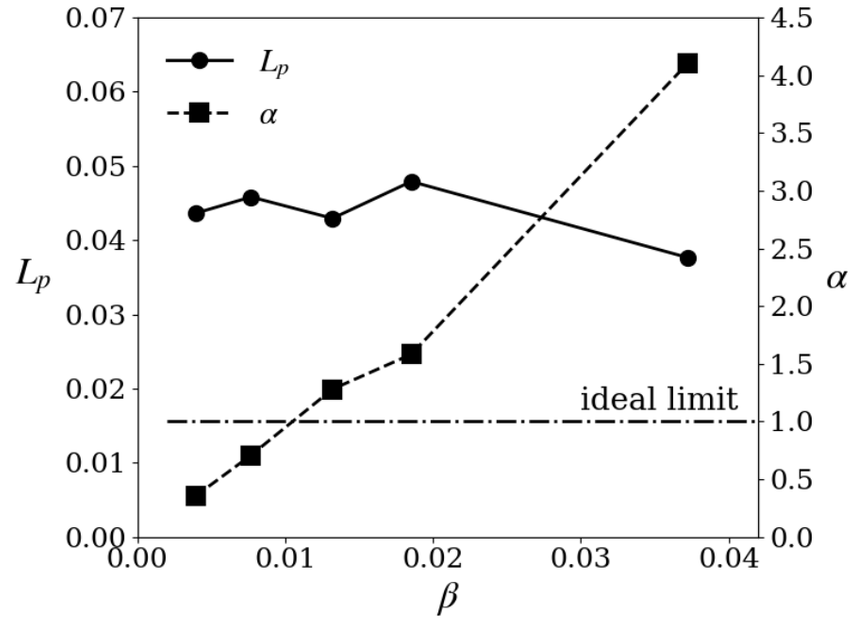


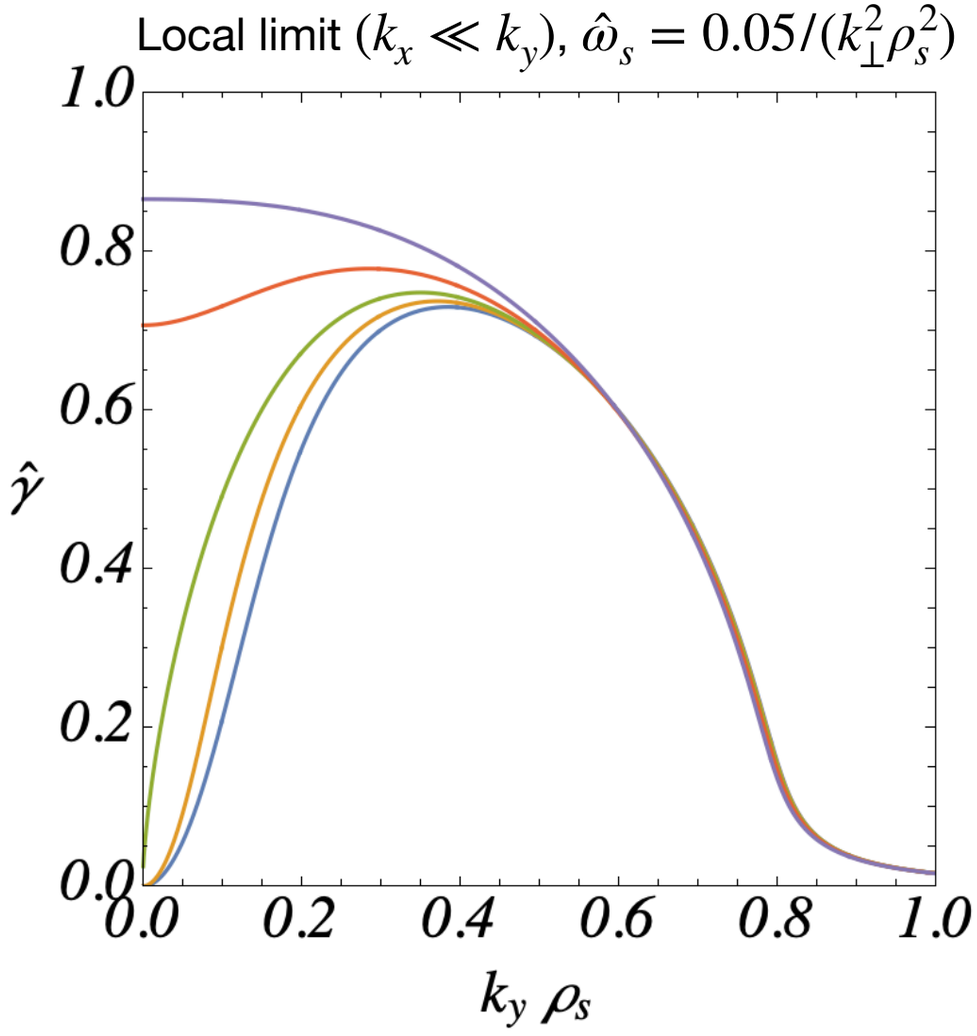


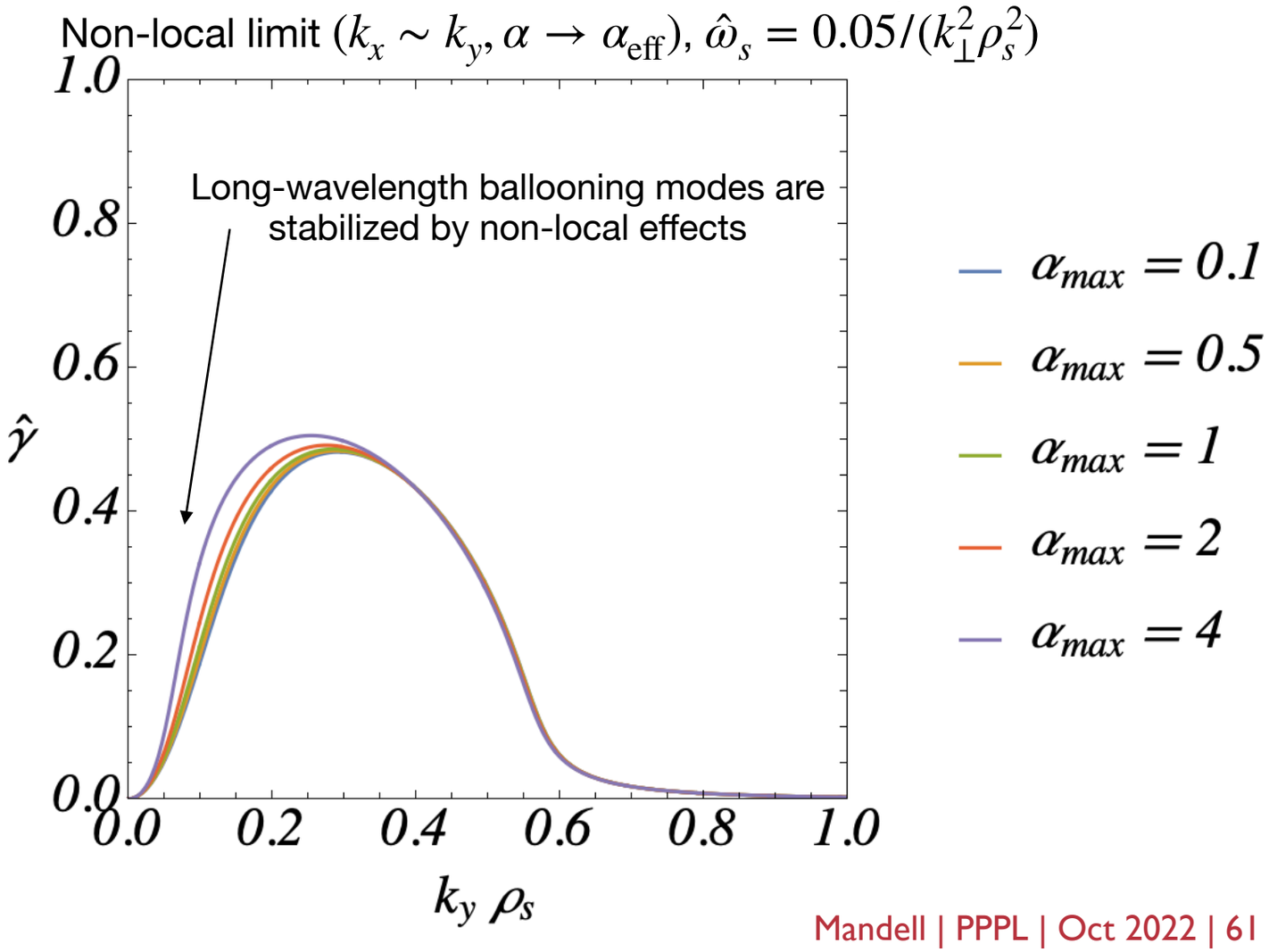
Including non-local effects

- Using this to compute an eigenfunction-averaged α_{eff}, and taking $k_x \sim k_y$, gives strong stabilization of long-wavelength ballooning modes (with finite sheath current)

- Weak dependence of instability on α is consistent with simulation results







The discontinuous Galerkin method

- We use the **discontinuous Galerkin** (DG) method to discretize the full-f gyrokinetic equation in Gkeyll
  - Class of finite-element methods with discontinuous basis functions to represent solution in each cell
  - Highly local, highly parallelizable, allows high-order accuracy, enforces local conservation laws
  - Can use limiters for stability (as in FV)

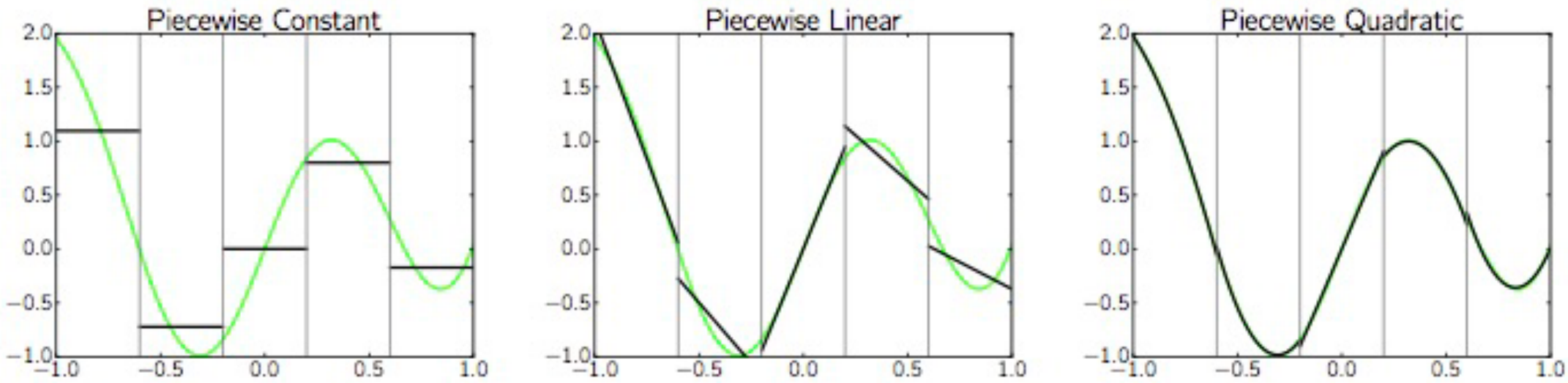


Figure: The best L_2 fit of $x^4 + \sin(5x)$ (green) using piecewise constant (left), linear (center), and quadratic (right) polynomials.

DG for general Hamiltonian system

$$\frac{\partial f}{\partial t} = \{H, f\}$$

Define phase-space velocity $\vec{\alpha} = \{\vec{Z}, H\}$, write in conservative form as

$$\frac{\partial f}{\partial t} + \nabla_{\vec{Z}} \cdot \left(\vec{\alpha} f\right) = 0$$

DG weak form:

- divide global phase-space domain into cells
- multiply GK eq. by a test function w_i and integrate (by parts) over cell C_m

$$\int_{C_m} d\vec{Z}\; w_i \frac{\partial f}{\partial t} + \oint_{\partial C_m} dS\; w_i^- \widehat{f\vec{\alpha}\cdot\vec{n}} - \int_{C_m} d\vec{Z}\; f\vec{\alpha}\cdot\nabla_{\vec{Z}} w_i = 0$$

- Particle conservation by taking $w = 1$
- Energy conservation by taking $w = H$, requires H to be continuous!

DG for full-f gyrokinetics

- GK is a Hamiltonian system, with $H = \frac{1}{2} m v_\parallel^2 + \frac{1}{2} m v_\perp^2 + q\phi$, and a non-canonical Poisson bracket:

$$\{F, G\} = \frac{\vec{B}^*}{m B_\parallel^*} \cdot \left(\nabla F \frac{\partial G}{\partial v_\parallel} - \frac{\partial F}{\partial v_\parallel} \nabla G \right) - \frac{\hat{b}}{q B_\parallel^*} \times \nabla F \cdot \nabla G$$

- Same DG weak form, recall $\overrightarrow{\alpha} = \{\overrightarrow{Z}, H\}$:

$$\int_{C_m} d\vec{Z}\ w_i \frac{\partial f}{\partial t} + \oint_{\partial C_m} dS\ w_i^- \widehat{f \vec{\alpha} \cdot \vec{n}} - \int_{C_m} d\vec{Z}\ f \vec{\alpha} \cdot \nabla_{\vec{Z}} w_i = 0$$

- Implicit conservation laws via integrals:
  - particle conservation $\int d\vec{Z}\ \frac{\partial f}{\partial t} = 0$ by taking $w = 1$
  - energy conservation $\int d\vec{Z}\ H \frac{\partial f}{\partial t} = 0$ by taking $w = H$, requires H continuous
  - since H must be continuous, ϕ must be continuous — use standard FEM for Poisson eq.
  - conservation laws require integrals to be computed exactly! (i.e. no aliasing errors)
  - exact integration with numerical quadrature $\sim \mathcal{O}(N_q N_b) \sim \mathcal{O}(N_b^3)$

Orthonormal bases to the rescue

- *Modal* expansion in each cell: $$f(\vec{Z},t) = \sum_k^{N_b} f_k(t) w_k(\vec{Z})$$
- Fundamental operations are tensor products, e.g. volume term:

$$\int_{C_m} d\vec{Z}\ f\vec{\alpha}\cdot\nabla w_i = \sum_{j,k} \underbrace{\left(\int_{C_m} d\vec{Z}\ w_j w_k \nabla w_i\right)}_{\overleftrightarrow{T}_{ijk}} \cdot \vec{\alpha}_j f_k$$

- Naively, this requires $\mathcal{O}(N_b^3)$ operations, same as quadrature (without aliasing)
- But if we choose basis functions to be *orthonormal*, $\overleftrightarrow{T}_{ijk}$ is sparse!
- We use “Serendipity” Legendre polynomials as our orthonormal basis functions
- Use a computer algebra system (Maxima) to compute sparse tensor products *analytically* and generate solver kernels

```
gkVolTerm_i : innerProd([x,y,z,vpar,vperp], 1, f_expd, alphaDotGradBasis_expd)
```

Computer-generated DG kernels

$$\mathrm{out}_i = \sum_{j,k} \overleftrightarrow{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
out[1] += 0.3061862178478971*(alphax[9]*f[9]+alphax[7]*f[7]+alphax[4]*f[4]+alphax[3]*f[3]+alphax[1]*f[1]
out[2] += 0.3061862178478971*(alphay[16]*f[16]+alphay[12]*f[12]+alphay[9]*f[9]+alphay[8]*f[8]+alphay[7]*
out[3] += 0.3061862178478971*(alphaz[1]*f[1]+alphaz[0]*f[0]);
out[4] += 0.3061862178478971*(alphav[26]*f[26]+alphav[23]*f[23]+alphav[19]*f[19]+alphav[18]*f[18]+alphav
out[6] += 0.3061862178478971*(alphax[9]*f[17]+(alphay[8]+alphax[7])*f[16]+f[8]*alphay[16]+alphay[5]*f[12
out[7] += 0.3061862178478971*(alphax[9]*f[18]+alphax[4]*f[11]+alphax[1]*f[7]+f[1]*alphax[7]+alphax[0]*f[
out[8] += 0.3061862178478971*(alphay[12]*f[21]+alphay[9]*f[18]+alphay[6]*f[16]+f[6]*alphay[16]+alphay[5]
out[9] += 0.3061862178478971*(alphav[19]*f[26]+f[19]*alphav[26]+alphav[15]*f[23]+f[15]*alphav[23]+(alpha
out[10] += 0.3061862178478971*(alphav[23]*f[28]+(alphav[18]+alphay[16])*f[26]+f[18]*alphav[26]+alphav[15
out[11] += 0.3061862178478971*(alphav[23]*f[29]+alphav[17]*f[26]+f[17]*alphav[26]+alphav[15]*f[25]+alpha
out[12] += 0.3061862178478971*(alphax[9]*f[23]+alphax[7]*f[21]+alphax[4]*f[15]+alphax[3]*f[14]+alphax[1]
out[13] += 0.3061862178478971*(alphay[16]*f[27]+alphay[9]*f[23]+alphay[8]*f[22]+alphay[7]*f[21]+alphay[6
out[14] += 0.3061862178478971*(alphaz[1]*f[12]+alphaz[0]*f[5]);
out[15] += 0.3061862178478971*(alphav[26]*f[31]+alphav[19]*f[30]+alphav[18]*f[29]+alphav[17]*f[28]+alpha
out[16] += 0.3061862178478971*(alphax[9]*f[26]+alphay[5]*f[21]+alphax[4]*f[19]+alphay[4]*f[18]+(alphay[2
out[17] += 0.3061862178478971*(alphav[15]*f[28]+(alphav[11]+alphay[8]+alphax[7])*f[26]+f[11]*alphav[26]+
out[18] += 0.3061862178478971*(alphav[15]*f[29]+alphav[10]*f[26]+f[10]*alphav[26]+alphav[23]*f[25]+alpha
out[19] += 0.3061862178478971*(alphav[23]*f[31]+alphav[15]*f[30]+alphay[12]*f[29]+alphav[12]*f[27]+(alph
out[20] += 0.3061862178478971*(alphax[9]*f[28]+(alphay[8]+alphax[7])*f[27]+alphax[4]*f[24]+alphay[4]*f[2
out[21] += 0.3061862178478971*(alphax[9]*f[29]+alphax[4]*f[25]+alphax[1]*f[21]+alphax[0]*f[14]+(alphax[7
out[22] += 0.3061862178478971*(alphay[9]*f[29]+alphay[6]*f[27]+alphay[4]*f[25]+alphay[2]*f[22]+alphay[1]
out[23] += 0.3061862178478971*(alphav[19]*f[31]+alphav[26]*f[30]+(alphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alphav[18]+alphay[16])*f[31]+(alphav[11]+alphay[8])*f[30]+(alphav[26]+al
out[25] += 0.3061862178478971*(alphav[17]*f[31]+alphav[10]*f[30]+alphav[9]*f[29]+alphav[26]*f[28]+alphav
out[26] += 0.3061862178478971*(alphav[15]*f[31]+alphav[23]*f[30]+alphay[5]*f[29]+alphav[5]*f[27]+(alphav
out[27] += 0.3061862178478971*(alphax[9]*f[31]+alphax[4]*f[30]+alphay[4]*f[29]+(alphay[2]+alphax[1])*f[2
out[28] += 0.3061862178478971*((alphav[11]+alphay[8]+alphax[7])*f[31]+(alphav[18]+alphay[16]+alphax[3])*
out[29] += 0.3061862178478971*(alphav[10]*f[31]+alphav[17]*f[30]+(alphav[4]+alphax[1])*f[29]+alphav[19]*
out[30] += 0.3061862178478971*((alphav[9]+alphay[6])*f[31]+(alphav[4]+alphay[2])*f[30]+(alphav[17]+alpha
out[31] += 0.3061862178478971*((alphav[4]+alphay[2]+alphax[1])*f[31]+(alphav[9]+alphay[6]+alphax[0])*f[3
```

Computer-generated DG kernels

$$\text{out}_i = \sum_{j,k} \overleftrightarrow{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
out[1] += 0.3061862178478971*(alphax[9]*f[9]+alphax[7]*f[7]+alphax[4]*f[4]+alphax[3]*f[3]+alphax[1]*f[1]
out[2] += 0.3061862178478971*(alphay[16]*f[16]+alphay[12]*f[12]+alphay[9]*f[9]+alphay[8]*f[8]+alphay[7]*
out[3] += 0.3061862178478971*(alphaz[1]*f[1]+alphaz[0]*f[0]);
out[4] += 0.3061862178478971*(alphav[26]*f[26]+alphav[23]*f[23]+alphav[19]*f[19]+alphav[18]*f[18]+alphav
out[6] += 0.3061862178478971*(alphax[9]*f[17]+(alphay[8]+alphax[7])*f[16]+f[8]*alphay[16]+alphay[5]*f[12
out[7] += 0.3061862178478971*(alphax[9]*f[18]+alphax[4]*f[11]+alphax[1]*f[7]+f[1]*alphax[7]+alphax[0]*f[
out[8] += 0.3061862178478971*(alphay[12]*f[21]+alphay[9]*f[18]+alphay[6]*f[16]+f[6]*alphay[16]+alphay[5]
out[9] += 0.3061862178478971*(alphav[19]*f[26]+f[19]*alphav[26]+alphav[15]*f[23]+f[15]*alphav[23]+(alpha
out[10] += 0.3061862178478971*(alphav[23]*f[28]+(alphav[18]+alphay[16])*f[26]+f[18]*alphav[26]+alphav[15
out[11] += 0.3061862178478971*(alphav[23]*f[29]+alphav[17]*f[26]+f[17]*alphav[26]+alphav[15]*f[25]+alpha
out[12] += 0.3061862178478971*(alphax[9]*f[23]+alphax[7]*f[21]+alphax[4]*f[15]+alphax[3]*f[14]+alphax[1]
out[13] += 0.3061862178478971*(alphay[16]*f[27]+alphay[9]*f[23]+alphay[8]*f[22]+alphay[7]*f[21]+alphay[6
out[14] += 0.3061862178478971*(alphaz[1]*f[12]+alphaz[0]*f[5]);
out[15] += 0.3061862178478971*(alphav[26]*f[31]+alphav[19]*f[30]+alphav[18]*f[29]+alphav[17]*f[28]+alpha
out[16] += 0.3061862178478971*(alphax[9]*f[26]+alphay[5]*f[21]+alphax[4]*f[19]+alphay[4]*f[18]+(alphay[2
out[17] += 0.3061862178478971*(alphav[15]*f[28]+(alphav[11]+alphay[8]+alphax[7])*f[26]+f[11]*alphav[26]+
out[18] += 0.3061862178478971*(alphav[15]*f[29]+alphav[10]*f[26]+f[10]*alphav[26]+alphav[23]*f[25]+alpha
out[19] += 0.3061862178478971*(alphav[23]*f[31]+alphav[15]*f[30]+alphay[12]*f[29]+alphav[12]*f[27]+(alph
out[20] += 0.3061862178478971*(alphax[9]*f[28]+(alphay[8]+alphax[7])*f[27]+alphax[4]*f[24]+alphay[4]*f[2
out[21] += 0.3061862178478971*(alphax[9]*f[29]+alphax[4]*f[25]+alphax[1]*f[21]+alphax[0]*f[14]+(alphax[7
out[22] += 0.3061862178478971*(alphay[9]*f[29]+alphay[6]*f[27]+alphay[4]*f[25]+alphay[2]*f[22]+alphay[1]
out[23] += 0.3061862178478971*(alphav[19]*f[31]+alphav[26]*f[30]+(alphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alphav[18]+alphay[16])*f[31]+(alphav[11]+alphay[8])*f[30]+(alphav[26]+al
out[25] += 0.3061862178478971*(alphav[17]*f[31]+alphav[10]*f[30]+alphav[9]*f[29]+alphav[26]*f[28]+alphav
out[26] += 0.3061862178478971*(alphav[15]*f[31]+alphav[23]*f[30]+alphay[5]*f[29]+alphav[5]*f[27]+(alphav
out[27] += 0.3061862178478971*(alphax[9]*f[31]+alphax[4]*f[30]+alphay[4]*f[29]+(alphay[2]+alphax[1])*f[2
out[28] += 0.3061862178478971*((alphav[11]+alphay[8]+alphax[7])*f[31]+(alphav[18]+alphay[16]+alphax[3])*
out[29] += 0.3061862178478971*(alphav[10]*f[31]+alphav[17]*f[30]+(alphav[4]+alphax[1])*f[29]+alphav[19]*
out[30] += 0.3061862178478971*((alphav[9]+alphay[6])*f[31]+(alphav[4]+alphay[2])*f[30]+(alphav[17]+alpha
out[31] += 0.3061862178478971*((alphav[4]+alphay[2]+alphax[1])*f[31]+(alphav[9]+alphay[6]+alphax[0])*f[3
```

- Maxima generates thousands of lines of machine-written C code… no loops!

Computer-generated DG kernels

$$\text{out}_i = \sum_{j,k} \overleftrightarrow{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
out[1] += 0.3061862178478971*(alphax[9]*f[9]+alphax[7]*f[7]+alphax[4]*f[4]+alphax[3]*f[3]+alphax[1]*f[1]
out[2] += 0.3061862178478971*(alphay[16]*f[16]+alphay[12]*f[12]+alphay[9]*f[9]+alphay[8]*f[8]+alphay[7]*
out[3] += 0.3061862178478971*(alphaz[1]*f[1]+alphaz[0]*f[0]);
out[4] += 0.3061862178478971*(alphav[26]*f[26]+alphav[23]*f[23]+alphav[19]*f[19]+alphav[18]*f[18]+alphav
out[6] += 0.3061862178478971*(alphax[9]*f[17]+(alphay[8]+alphax[7])*f[16]+f[8]*alphay[16]+alphay[5]*f[12
                                                  f[11]+alphax[1]*f[7]+f[1]*alphax[7]+alphax[0]*f[
                                                  *f[18]+alphay[6]*f[16]+f[6]*alphay[16]+alphay[5]
out[9] += 0.3061862178478971*(alphav[19]*f[26]+f[19]*alphav[26]+alphav[15]*f[23]+f[15]*alphav[23]+(alpha
out[10] += 0.3061862178478971*(alphav[23]*f[28]+(alphav[18]+alphay[16])*f[26]+f[18]*alphav[26]+alphav[15
out[11] += 0.3061862178478971*(alphav[23]*f[29]+alphav[17]*f[26]+f[17]*alphav[26]+alphav[15]*f[25]+alpha
out[12] += 0.3061862178478971*(alphax[9]*f[23]+alphax[7]*f[21]+alphax[4]*f[15]+alphax[3]*f[14]+alphax[1]
out[13] += 0.3061862178478971*(alphay[16]*f[27]+alphay[9]*f[23]+alphay[8]*f[22]+alphay[7]*f[21]+alphay[6
out[14] += 0.3061862178478971*(alphaz[1]*f[12]+alphaz[0]*f[5]);
out[15] += 0.3061862178478971*(alphav[26]*f[31]+alphav[19]*f[30]+alphav[18]*f[29]+alphav[17]*f[28]+alpha
out[16] += 0.3061862178478971*(alphax[9]*f[26]+alphay[5]*f[21]+alphax[4]*f[19]+alphay[4]*f[18]+(alphay[2
out[17] += 0.3061862178478971*(alphav[15]*f[28]+(alphav[11]+alphay[8]+alphax[7])*f[26]+f[11]*alphav[26]+
out[18] += 0.3061862178478971*(alphav[15]*f[29]+alphav[10]*f[26]+f[10]*alphav[26]+alphav[23]*f[25]+alpha
out[19] += 0.3061862178478971*(alphav[23]*f[31]+alphav[15]*f[30]+alphay[12]*f[29]+alphav[12]*f[27]+(alph
out[20] += 0.3061862178478971*(alphax[9]*f[28]+(alphay[8]+alphax[7])*f[27]+alphax[4]*f[24]+alphay[4]*f[2
out[21] += 0.3061862178478971*(alphax                                  ax[1]*f[21]+alphax[0]*f[14]+(alphax[7
out[22] += 0.3061862178478971*(alphay                                  ay[4]*f[25]+alphay[2]*f[22]+alphay[1]
out[23] += 0.3061862178478971*(alphav                                  lphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alphav[18]+alphay[16])*f[31]+(alphav[11]+alphay[8])*f[30]+(alphav[26]+al
out[25] += 0.3061862178478971*(alphav[17]*f[31]+alphav[10]*f[30]+alphav[9]*f[29]+alphav[26]*f[28]+alphav
out[26] += 0.3061862178478971*(alphav[15]*f[31]+alphav[23]*f[30]+alphay[5]*f[29]+alphav[5]*f[27]+(alphav
out[27] += 0.3061862178478971*(alphax[9]*f[31]+alphax[4]*f[30]+alphay[4]*f[29]+(alphay[2]+alphax[1])*f[2
out[28] += 0.3061862178478971*((alphav[11]+alphay[8]+alphax[7])*f[31]+(alphav[18]+alphay[16]+alphax[3])*
out[29] += 0.3061862178478971*(alphav[10]*f[31]+alphav[17]*f[30]+(alphav[4]+alphax[1])*f[29]+alphav[19]*
out[30] += 0.3061862178478971*((alphav[9]+alphay[6])*f[31]+(alphav[4]+alphay[2])*f[30]+(alphav[17]+alpha
out[31] += 0.3061862178478971*((alphav[4]+alphay[2]+alphax[1])*f[31]+(alphav[9]+alphay[6]+alphax[0])*f[3
```

5D GK, piecewise linear basis = 32 basis functions

163,840 multiplications,
98,304 additions

540 multiplications,
608 additions

- Maxima generates thousands of lines of machine-written C code… no loops!

Computer-generated DG kernels

$$\text{out}_i = \sum_{j,k} \overleftrightarrow{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
out[1] += 0.3061862178478971*(alphax[9]*f[9]+alphax[7]*f[7]+alphax[
out[2] += 0.3061862178478971*(alphay[16]*f[16]+alphay[12]*f[12]+alph
out[3] += 0.3061862178478971*(alphaz[1]*f[1]+alphaz[0]*f[0]);
out[4] += 0.3061862178478971*(alphav[26]*f[26]+alphav[23]*f[23]+alph
out[6] += 0.3061862178478971*(alphax[9]*f[17]+(alphay[8]+alphax[7])*
                                                  f[11]+alphax
                                                  *f[18]+alph
out[9] += 0.3061862178478971*(alphav[19]*f[26]+f[19]*alphav[26]+alph
out[10] += 0.3061862178478971*(alphav[23]*f[28]+(alphav[18]+alphay[
out[11] += 0.3061862178478971*(alphav[23]*f[29]+alphav[17]*f[26]+f[
out[12] += 0.3061862178478971*(alphax[9]*f[23]+alphax[7]*f[21]+alph
out[13] += 0.3061862178478971*(alphay[16]*f[27]+alphay[9]*f[23]+alph
out[14] += 0.3061862178478971*(alphaz[1]*f[12]+alphaz[0]*f[5]);
out[15] += 0.3061862178478971*(alphav[26]*f[31]+alphav[19]*f[30]+alp
out[16] += 0.3061862178478971*(alphax[9]*f[26]+alphay[5]*f[21]+alph
out[17] += 0.3061862178478971*(alphav[15]*f[28]+(alphav[11]+alphay[
out[18] += 0.3061862178478971*(alphav[15]*f[29]+alphav[10]*f[26]+f[
out[19] += 0.3061862178478971*(alphav[23]*f[31]+alphav[15]*f[30]+alphay[12]*f[29]+alphav[12]*f[27]+(alph
out[20] += 0.3061862178478971*(alphax[9]*f[28]+(alphay[8]+alphax[7])*f[27]+alphax[4]*f[24]+alphay[4]*f[2
out[21] += 0.3061862178478971*(alphax                ax[1]*f[21]+alphax[0]*f[14]+(alphax[7
out[22] += 0.3061862178478971*(alphay                ay[4]*f[25]+alphay[2]*f[22]+alphay[1]
out[23] += 0.3061862178478971*(alphav                lphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alphav[18]+alphay[16])*f[31]+(alphav[11]+alphay[8])*f[30]+(alphav[26]+al
out[25] += 0.3061862178478971*(alphav[17]*f[31]+alphav[10]*f[30]+alphav[9]*f[29]+alphav[26]*f[28]+alphav
out[26] += 0.3061862178478971*(alphav[15]*f[31]+alphav[23]*f[30]+alphay[5]*f[29]+alphav[5]*f[27]+(alphav
out[27] += 0.3061862178478971*(alphax[9]*f[31]+alphax[4]*f[30]+alphay[4]*f[29]+(alphay[2]+alphax[1])*f[2
out[28] += 0.3061862178478971*((alphav[11]+alphay[8]+alphax[7])*f[31]+(alphav[18]+alphay[16]+alphax[3])*
out[29] += 0.3061862178478971*(alphav[10]*f[31]+alphav[17]*f[30]+(alphav[4]+alphax[1])*f[29]+alphav[19]*
                                                  hav[4]+alphay[2])*f[30]+(alphav[17]+alpha
                                                  f[31]+(alphav[9]+alphay[6]+alphax[0])*f[3
```

5D GK, piecewise linear basis = 32 basis functions

163,840 multiplications, 98,304 additions

540 multiplications, 608 additions

Code is ~30x faster than old nodal version w/ quadrature!

- Maxima generates thousands of lines of machine-written C code… no loops!

Computer-generated DG kernels

$$\text{out}_i = \sum_{j,k} \overleftrightarrow{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
out[1] += 0.3061862178478971*(alphax[9]*f[9]+alphax[7]*f[7]+alphax[
out[2] += 0.3061862178478971*(alphay[16]*f[16]+alphay[12]*f[12]+alph
out[3] += 0.3061862178478971*(alphaz[1]*f[1]+alphaz[0]*f[0]);
out[4] += 0.3061862178478971*(alphav[26]*f[26]+alphav[23]*f[23]+alph
out[6] += 0.3061862178478971*(alphax[9]*f[17]+(alphay[8]+alphax[7])*
                                                  f[11]+alphax
                                                  *f[18]+alph
out[9] += 0.3061862178478971*(alphav[19]*f[26]+f[19]*alphav[26]+alph
out[10] += 0.3061862178478971*(alphav[23]*f[28]+(alphav[18]+alphay[
out[11] += 0.3061862178478971*(alphav[23]*f[29]+alphav[17]*f[26]+f[
out[12] += 0.3061862178478971*(alphax[9]*f[23]+alphax[7]*f[21]+alph
out[13] += 0.3061862178478971*(alphay[16]*f[27]+alphay[9]*f[23]+alph
out[14] += 0.3061862178478971*(alphaz[1]*f[12]+alphaz[0]*f[5]);
out[15] += 0.3061862178478971*(alphav[26]*f[31]+alphav[19]*f[30]+alp
out[16] += 0.3061862178478971*(alphax[9]*f[26]+alphay[5]*f[21]+alph
out[17] += 0.3061862178478971*(alphav[15]*f[28]+(alphav[11]+alphay[
out[18] += 0.3061862178478971*(alphav[15]*f[29]+alphav[10]*f[26]+f[
out[19] += 0.3061862178478971*(alphav[23]*f[31]+alphav[15]*f[30]+alphay[12]*f[29]+alphav[12]*f[27]+(alph
out[20] += 0.3061862178478971*(alphax[9]*f[28]+(alphay[8]+alphax[7])*f[27]+alphax[4]*f[24]+alphay[4]*f[2
out[21] += 0.3061862178478971*(alphax                ax[1]*f[21]+alphax[0]*f[14]+(alphax[7
out[22] += 0.3061862178478971*(alphay                ay[4]*f[25]+alphay[2]*f[22]+alphay[1]
out[23] += 0.3061862178478971*(alphav                lphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alphav[18]+alphay[16])*f[31]+(alphav[11]+alphay[8])*f[30]+(alphav[26]+al
out[25] += 0.3061862178478971*(alphav[17]*f[31]+alphav[10]*f[30]+alphav[9]*f[29]+alphav[26]*f[28]+alphav
out[26] += 0.3061862178478971*(alphav[15]*f[31]+alphav[23]*f[30]+alphay[5]*f[29]+alphav[5]*f[27]+(alphav
out[27] += 0.3061862178478971*(alphax[9]*f[31]+alphax[4]*f[30]+alphay[4]*f[29]+(alphay[2]+alphax[1])*f[2
out[28] += 0.3061862178478971*((alphav[11]+alphay[8]+alphax[7])*f[31]+(alphav[18]+alphay[16]+alphax[3])*
out[29] += 0.3061862178478971*(alphav[10]*f[31]+alphav[17]*f[30]+(alphav[4]+alphax[1])*f[29]+alphav[19]*
                                                  hav[4]+alphay[2])*f[30]+(alphav[17]+alpha
                                                  f[31]+(alphav[9]+alphay[6]+alphax[0])*f[3
```

5D GK, piecewise linear basis = 32 basis functions

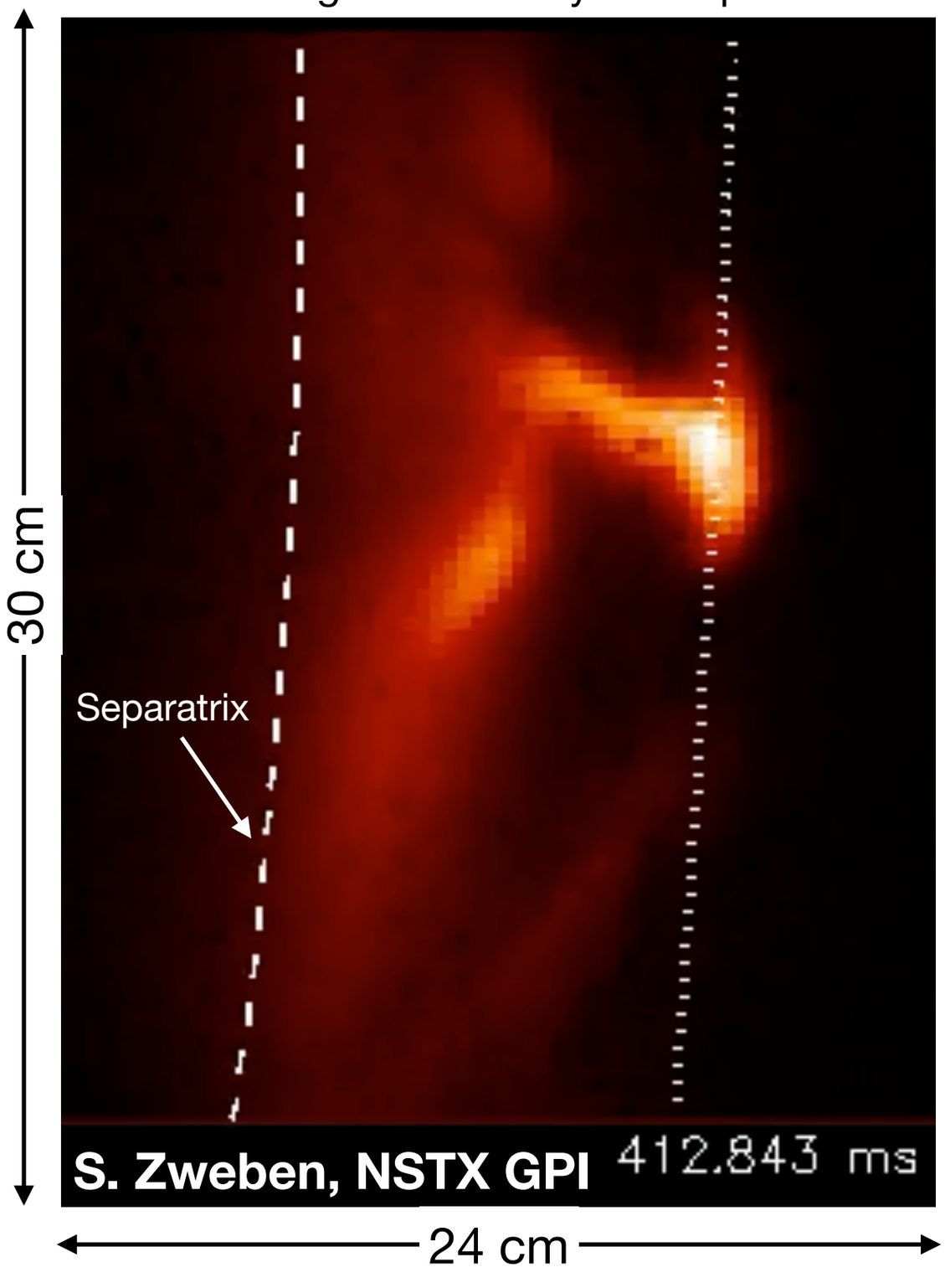
163,840 multiplications, 98,304 additions

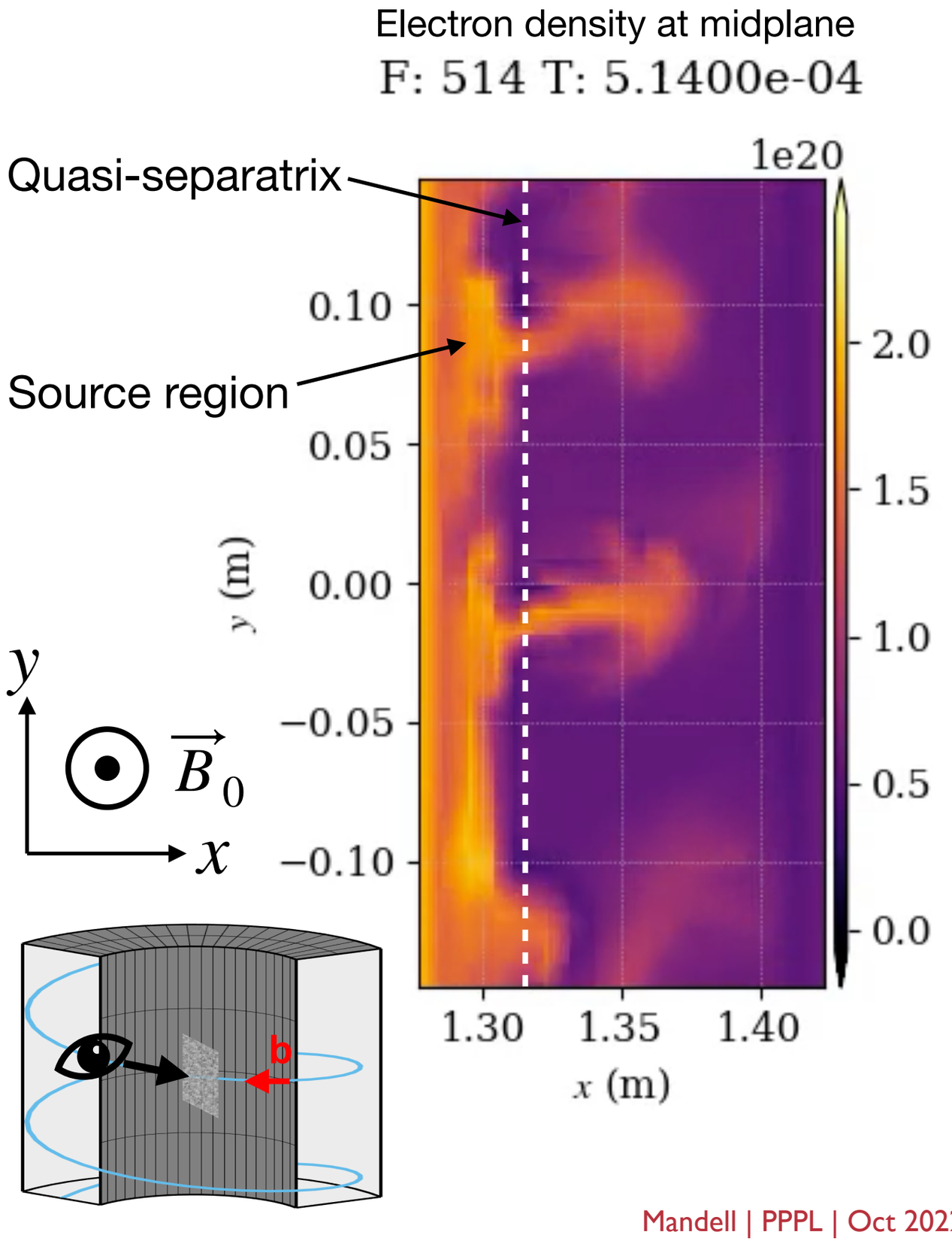
540 multiplications, 608 additions

Code is ~30x faster than old nodal version w/ quadrature!

- Maxima generates thousands of lines of machine-written C code… no loops!
- Easier to generalize to different dimensionality/polynomial order, add new terms, debug/test, etc.

NSTX gas-puff-imaging (GPI) diagnostic vs Gkeyll





NSTX gas-puff-imaging (GPI) diagnostic vs Gkeyll