Electromagnetic gyrokinetic turbulence simulations in the tokamak edge with discontinuous Galerkin methods

Noah Mandell
DOE CSGF Program Review — July 2019

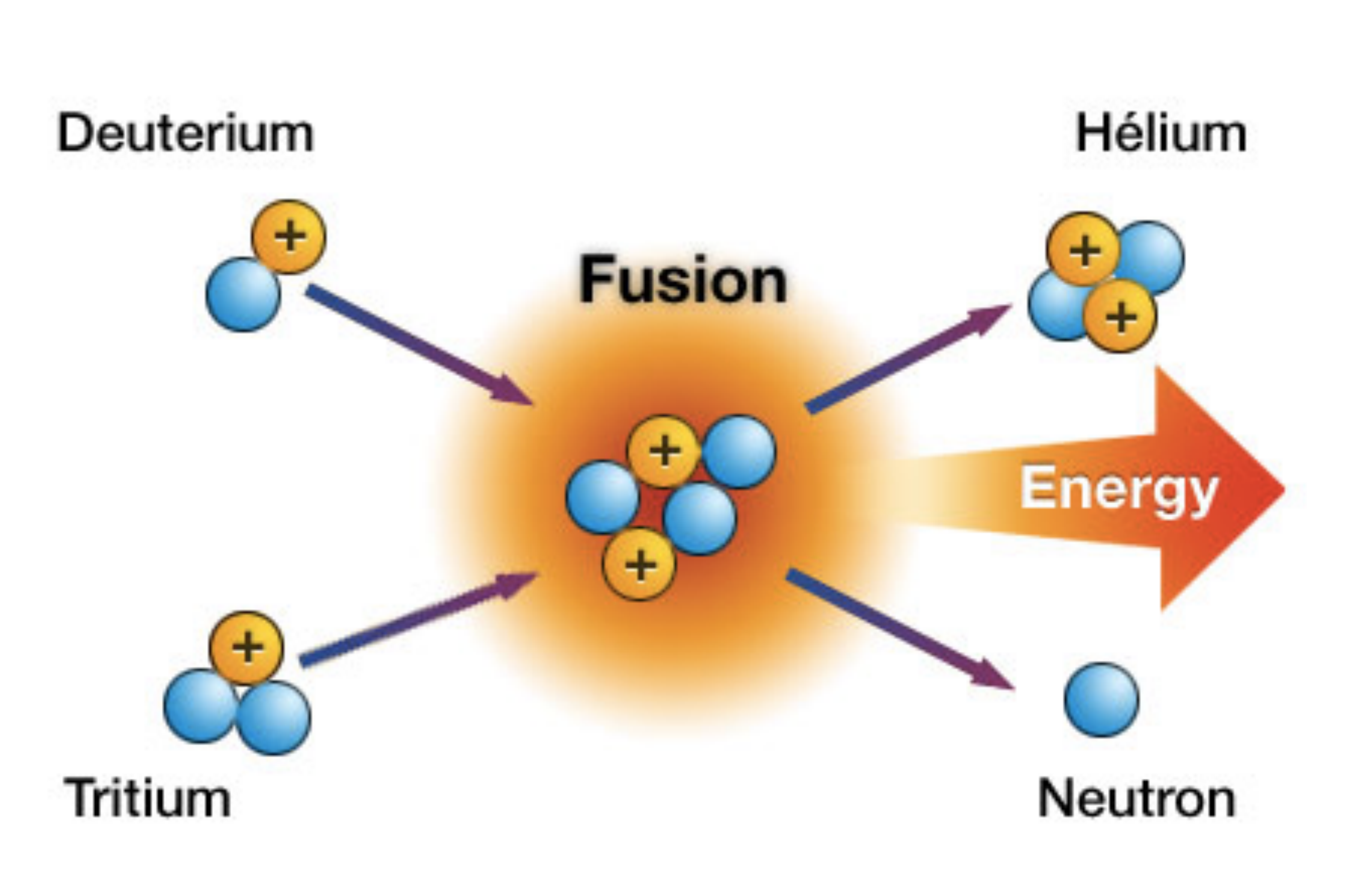
PRINCETON UNIVERSITY

DOE CSGF

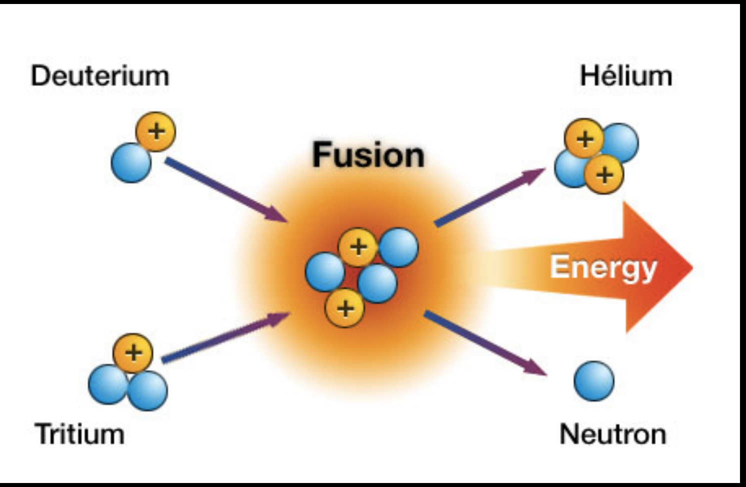
PPPL
PRINCETON PLASMA PHYSICS LABORATORY

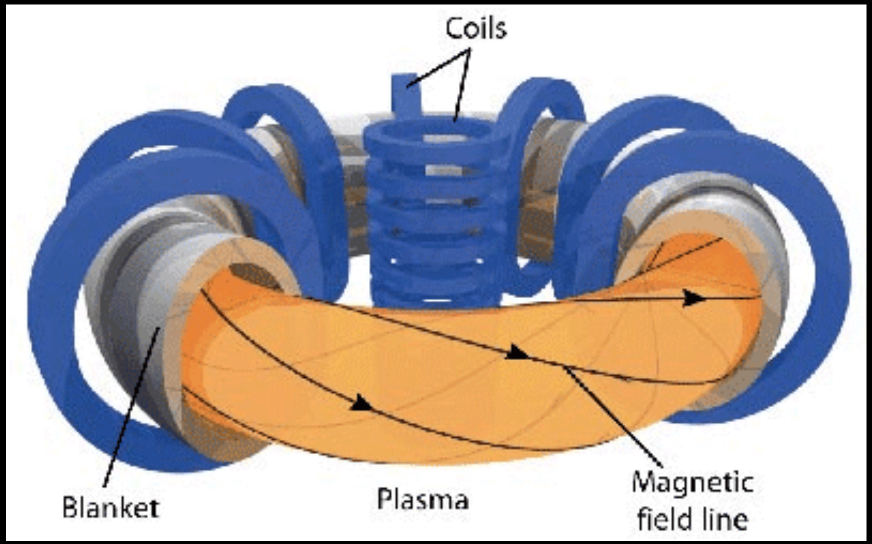
How can we save the world?

How can we make fusion energy?

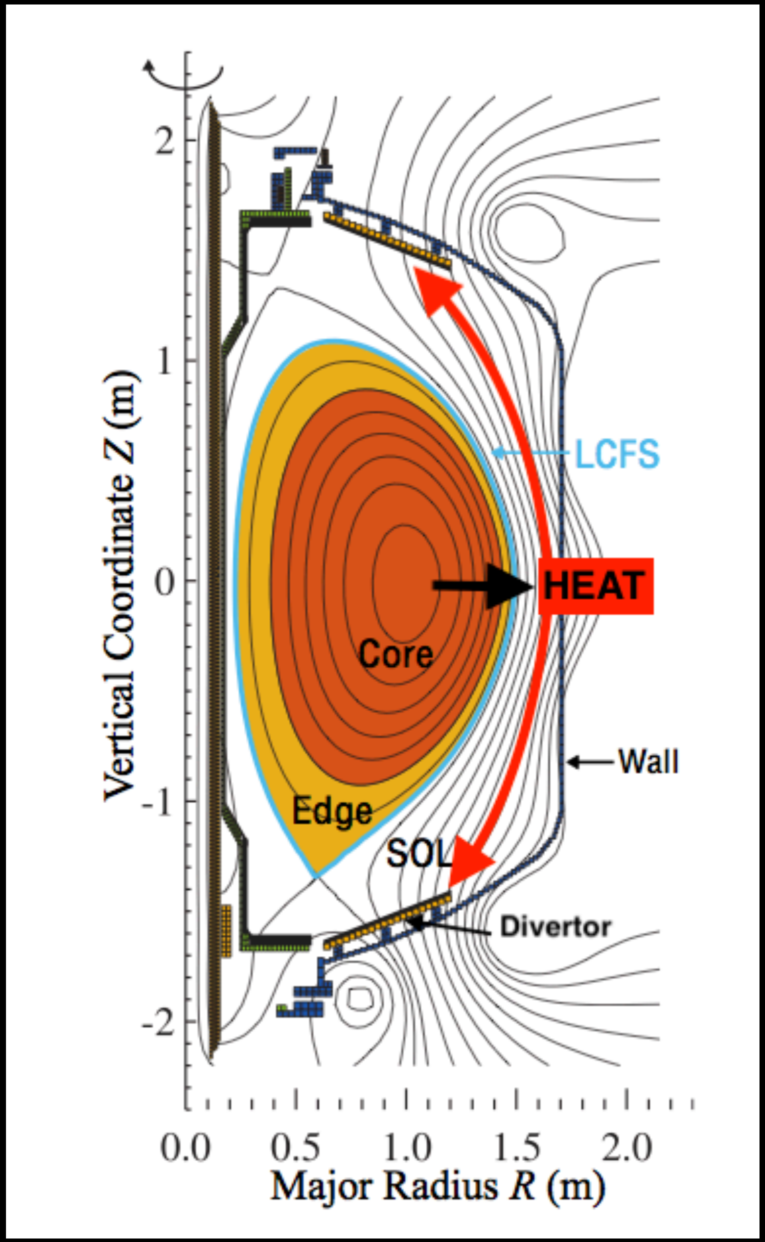


How can we make fusion energy?

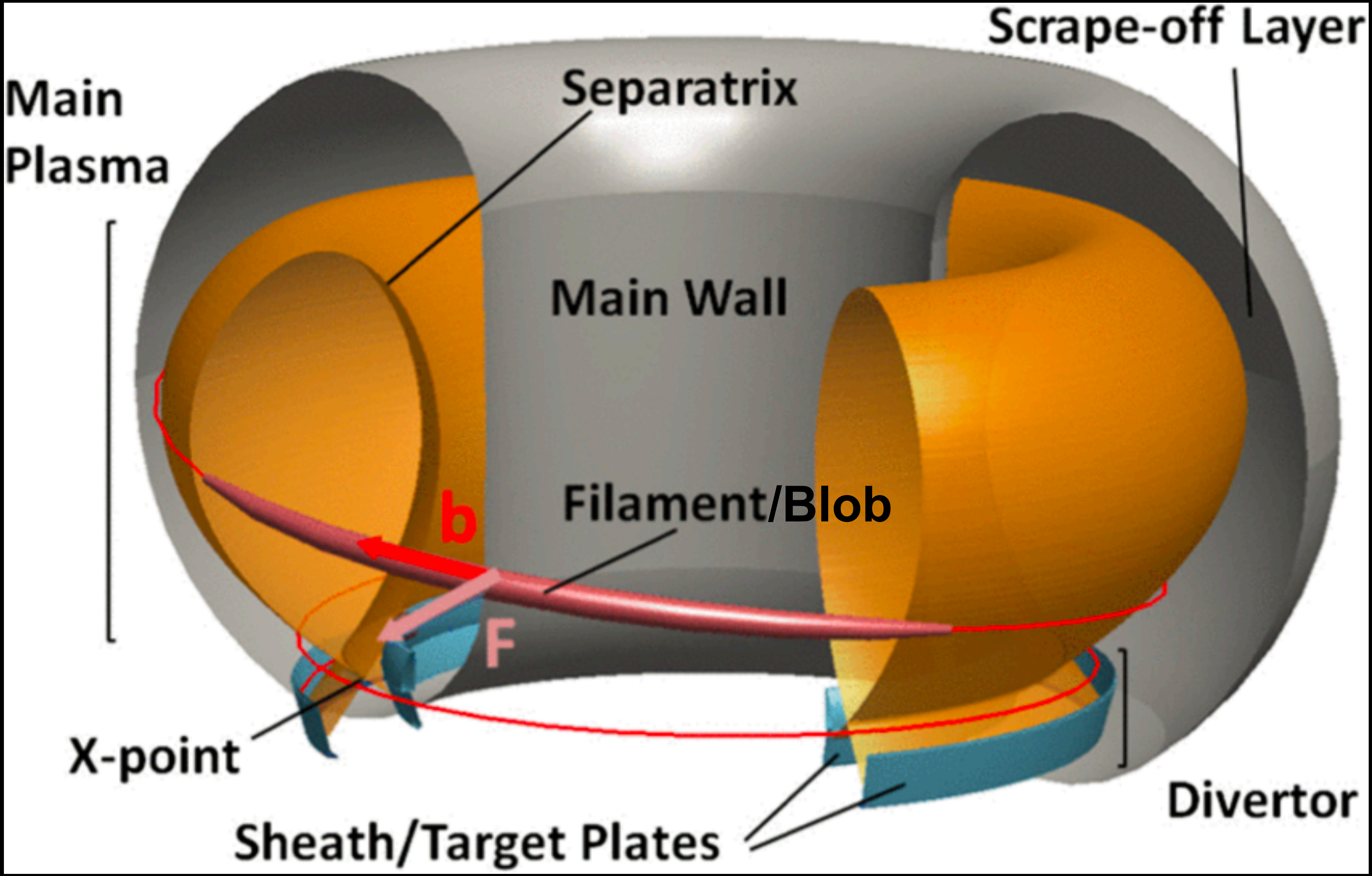




- Confine plasma with magnetic fields in a donut-shaped reactor called a tokamak and heat it to > 100 million °C
- Turbulence is a main source of inefficiency (in core)
- Plasma properties in the edge/SOL constrain performance and component lifetime
  - Heat exhausted in SOL could damage divertor plates
  - Sets boundary condition on core profiles (e.g. H mode)



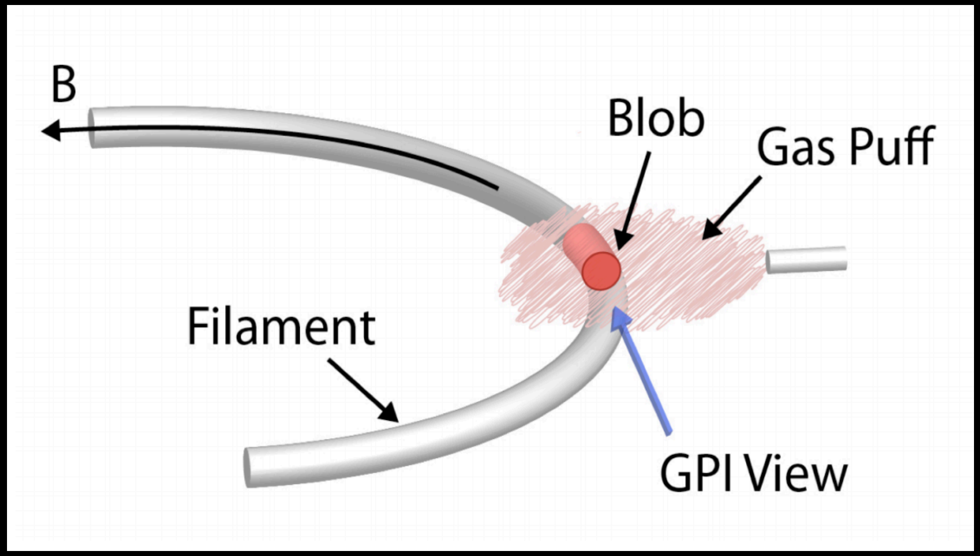
Scrape-Off Layer Dynamics

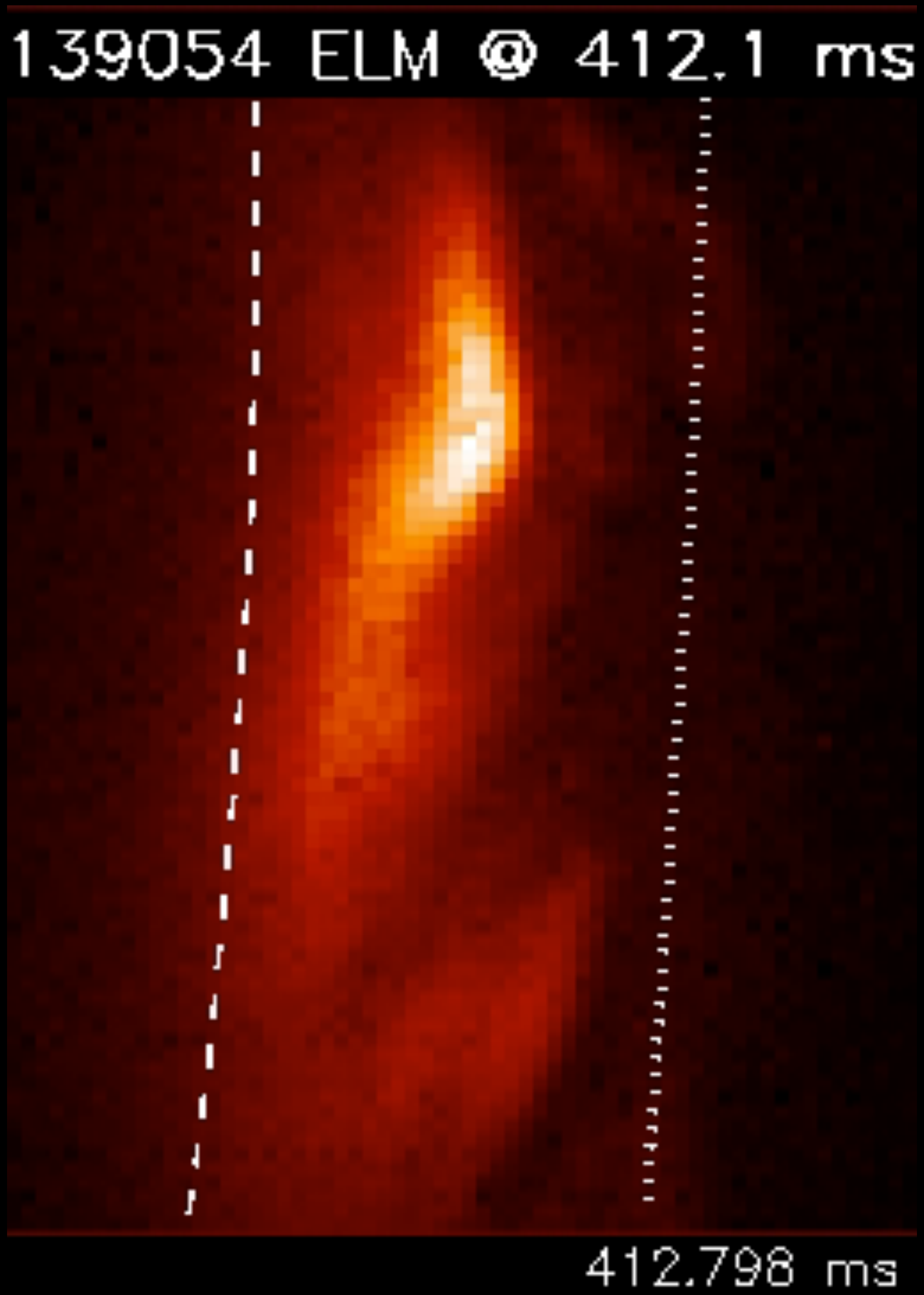


Carralero et al, PRL (2015)

Imaging SOL with GPI

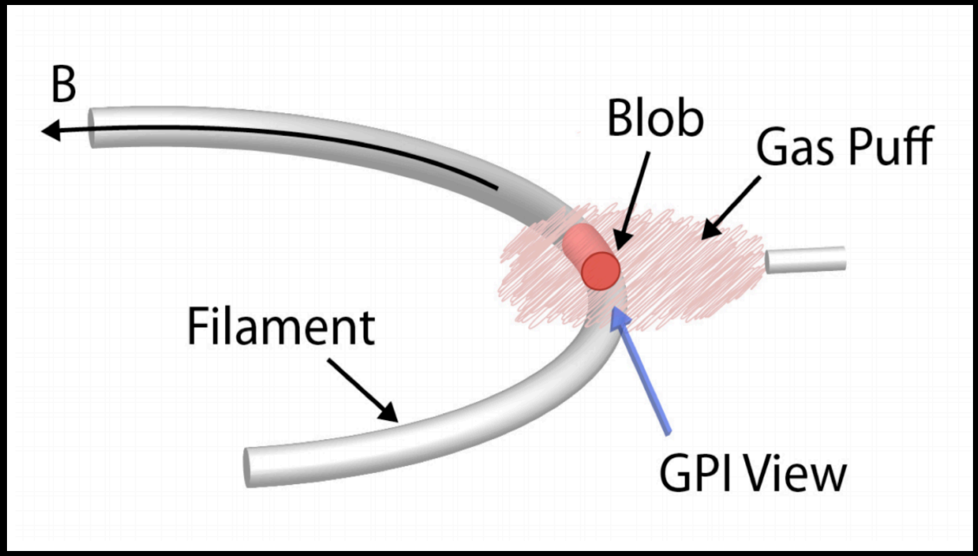
- GPI = Gas-puff imaging diagnostic (S. Zweben)
- Real-time turbulence movies in NSTX SOL
- Data taken using fast camera (400,000 fr/s)

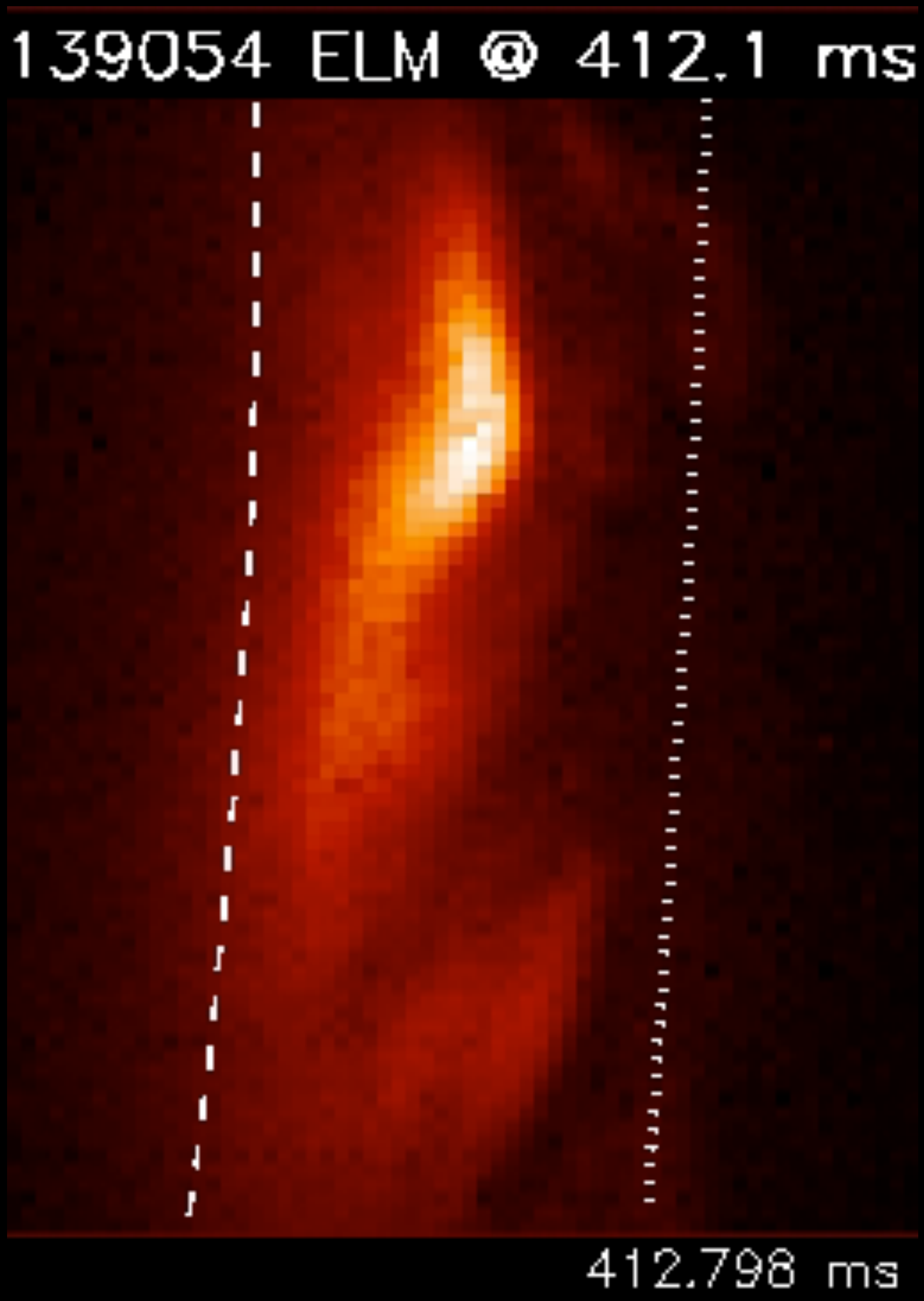




Imaging SOL with GPI

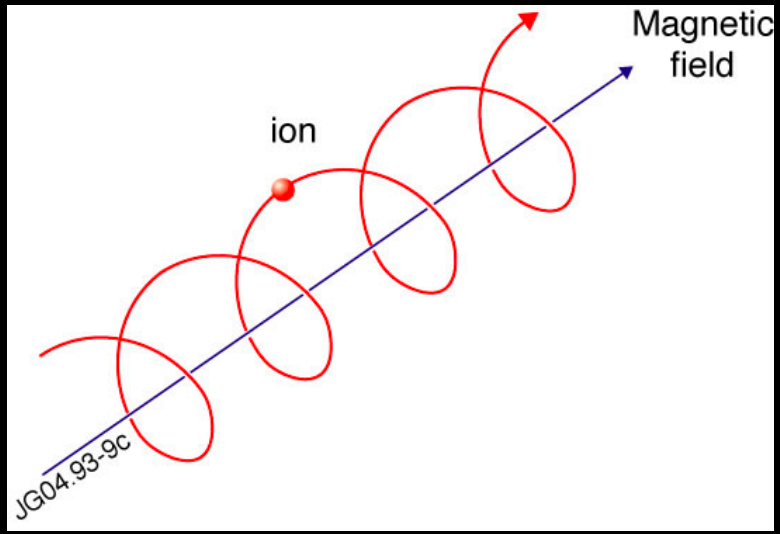
- GPI = Gas-puff imaging diagnostic (S. Zweben)
- Real-time turbulence movies in NSTX SOL
- Data taken using fast camera (400,000 fr/s)

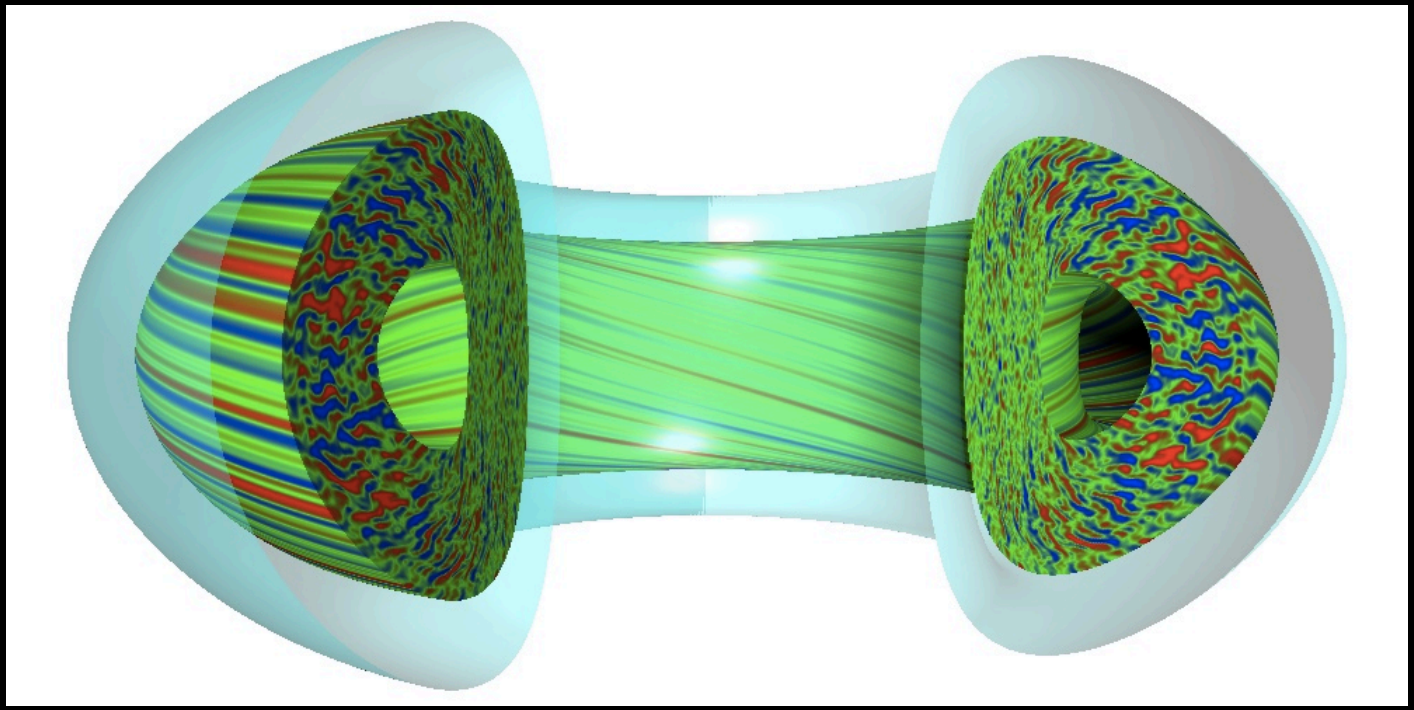




Gyro-… what??

- Gyrokinetics describes turbulence in fusion plasmas
  - "Kinetic": phase space with spatial dimensions AND velocity dimensions
  - "Gyro": reduce 6D→ 5D (3 spatial, 2 velocity) by averaging over high frequency particle gyration in strong background magnetic field





GYRO simulation, Candy

What is the gyrokinetic equation?

- Basically a hyperbolic PDE that describes time evolution of phase-space density of particles $f(x, y, z, v_\parallel, v_\perp) = f(\vec{R}, v_\parallel, v_\perp)$

$$\frac{\partial f}{\partial t} + \nabla \cdot \left(\dot{\vec{R}} f \right) + \frac{\partial}{\partial v_\parallel} \left(\dot{v}_\parallel f \right) = C[f] + S$$

$$\dot{\vec{R}} = \frac{\vec{B}}{B} v_\parallel + \vec{v}_{D\perp}$$

$$\dot{v}_\parallel = \frac{q}{m} E_\parallel + \ldots$$

- Conservation laws are important!
    - GK is a **Hamiltonian** system
    - integrals of GK eq. give conservation laws for particles, energy, etc
    - conservation laws are **implicit** (e.g. no explicit energy conservation equation)

Discretizing the GK equation

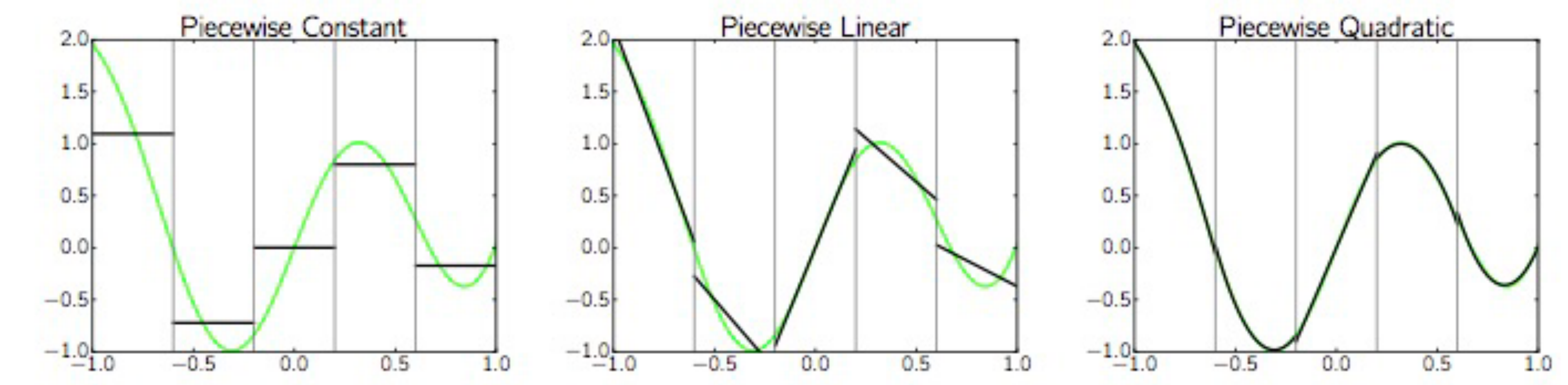


Figure: The best L_2 fit of $x^4 + \sin(5x)$ (green) using piecewise constant (left), linear (center), and quadratic (right) polynomials.

- Discontinuous Galerkin (DG) method
  - 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

Discretizing the GK equation

$$\frac{\partial f}{\partial t} + \nabla \cdot \left(\dot{\vec{R}} f\right) + \frac{\partial}{\partial v_\parallel}\left(\dot{v}_\parallel f\right) = C[f] + S$$

Discretizing the GK equation

$$\frac{\partial f}{\partial t} + \nabla \cdot \left(\dot{\vec{R}} f\right) + \frac{\partial}{\partial v_\parallel}\left(\dot{v}_\parallel f\right) = 0$$

Discretizing the GK equation

$$\frac{\partial f}{\partial t} + \nabla_Z \cdot (\vec{\alpha} f) = 0$$

Discretizing the GK equation

$$\frac{\partial f}{\partial t} + \nabla_Z \cdot (\vec{\alpha} f) = 0$$

- 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

$$\int_{C_m} d\overrightarrow{Z}\ w_i \frac{\partial f}{\partial t} + \oint_{\partial C_m} dS\ w_i \hat{f}\, \overrightarrow{\alpha} \cdot \overrightarrow{n} - \int_{C_m} d\overrightarrow{Z}\ f\, \overrightarrow{\alpha} \cdot \nabla_Z w_i = 0$$

- Implicit conservation laws via integrals:
  - particle conservation by taking $w = 1$
  - energy conservation by taking $w = H$, the Hamiltonian
  - 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

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

- Naively, this is no better than quadrature
- But if we choose basis functions to be *orthonormal*, $\vec{T}_{ijk}$ is sparse!
- We use Legendre polynomials as our orthonormal basis functions
- Use a computer algebra system (Maxima) to compute sparse tensor products and generate solver kernels

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

Look ma, no loops!

$$out_i = \sum_{j,k} \vec{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
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
```

Look ma, no loops!

$$out_i = \sum_{j,k} \vec{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
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!

Look ma, no loops!

5D GK, piecewise linear basis = 32 basis functions

```
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]+(alphax[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*(alpha                    )*f[27]+alphax[4]*f[24]+alphay[4]*f[2
[21] += 0.3061862178478971*(alpha                    ax[1]*f[21]+alphax[0]*f[14]+(alphax[7
[22] += 0.3061862178478971*(alpha                    ay[4]*f[25]+alphay[2]*f[22]+alphay[1]
[23] += 0.3061862178478971*(alpha                    lphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alph                  v[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
```

$$out_i = \sum_{j,k} \vec{T}_{ijk} \cdot \vec{\alpha}_j f_k \rightarrow$$

163,840 multiplications, 98,304 additions

540 multiplications, 608 additions

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

Look ma, no loops!

5D GK, piecewise linear basis = 32 basis functions

$$out_i = \sum_{j,k} \vec{T}_{ijk} \cdot \vec{\alpha}_j f_k$$

```
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*(alpha                                )*f[27]+alphax[4]*f[24]+alphay[4]*f[2
out[21] += 0.3061862178478971*(alpha                          ax[1]*f[21]+alphax[0]*f[14]+(alphax[7
out[22] += 0.3061862178478971*(alpha                          ay[4]*f[25]+alphay[2]*f[22]+alphay[1]
out[23] += 0.3061862178478971*(alpha                          lphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alpha                                       v[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                                 hav[10]*f[31]+alphav[17]*f[30]+(alphav[4]+alphax[1])*f[29]+alphav[19]*
out[30                                 phav[9]+alphay[6])*f[31]+(alphav[4]+alphay[2])*f[30]+(alphav[17]+alpha
out[31                                 phav[4]+alphay[2]+alphax[1])*f[31]+(alphav[9]+alphay[6]+alphax[0])*f[3
```

163,840 multiplications,
98,304 additions

540 multiplications,
608 additions

~30x faster!

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

Look ma, no loops!

```
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]+(alphax[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*(alpha                    )*f[27]+alphax[4]*f[24]+alphay[4]*f[2
[21] += 0.3061862178478971*(alphax                    ax[1]*f[21]+alphax[0]*f[14]+(alphax[7
[22] += 0.3061862178478971*(alphay                    ay[4]*f[25]+alphay[2]*f[22]+alphay[1]
[23] += 0.3061862178478971*(alpha                     lphav[11]+alphax[7])*f[29]+alphav[10]
out[24] += 0.3061862178478971*((alph                  v[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                    hav[10]*f[31]+alphav[17]*f[30]+(alphav[4]+alphax[1])*f[29]+alphav[19]*
out[30                    phav[9]+alphay[6])*f[31]+(alphav[4]+alphay[2])*f[30]+(alphav[17]+alpha
out[31                    phav[4]+alphay[2]+alphax[1])*f[31]+(alphav[9]+alphay[6]+alphax[0])*f[3
```

5D GK, piecewise linear basis = 32 basis functions

$$out_i = \sum_{j,k} \vec{T}_{ijk} \cdot \vec{\alpha}_j f_k \Rightarrow$$

163,840 multiplications,
98,304 additions

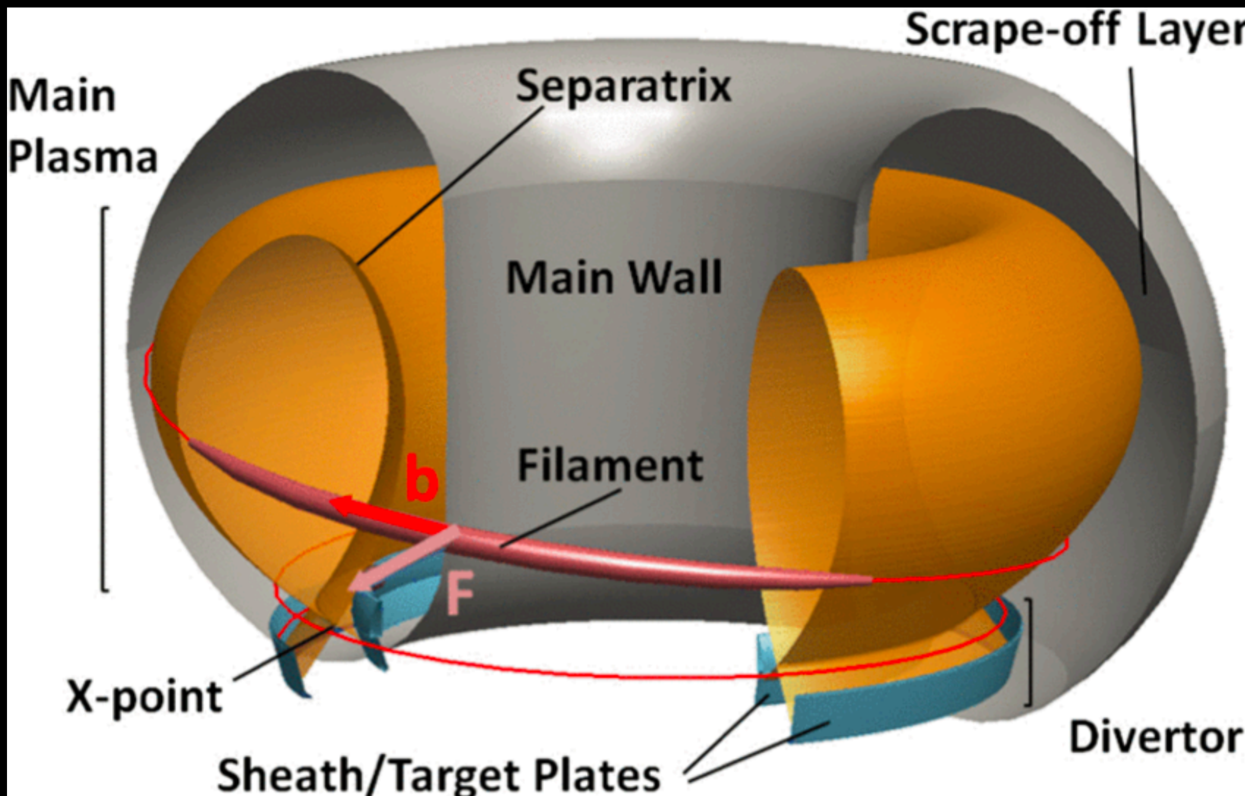
540 multiplications,
608 additions

~30x faster!

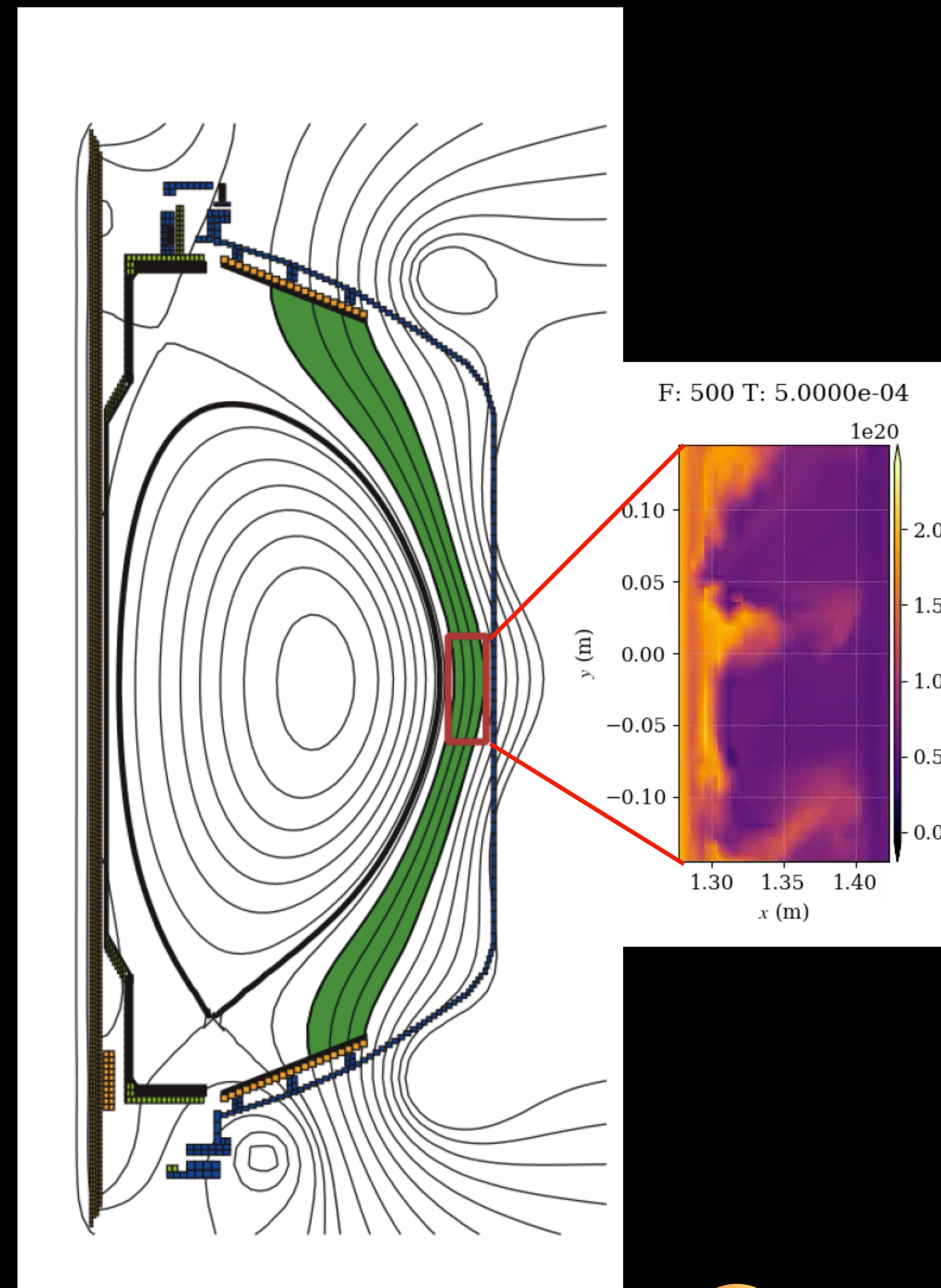
- Maxima generates thousands of lines of machine-written C code… no loops!
- Easier to generalize to different dimensionality/polynomial order, add new terms, debug/test, etc.

PPPL

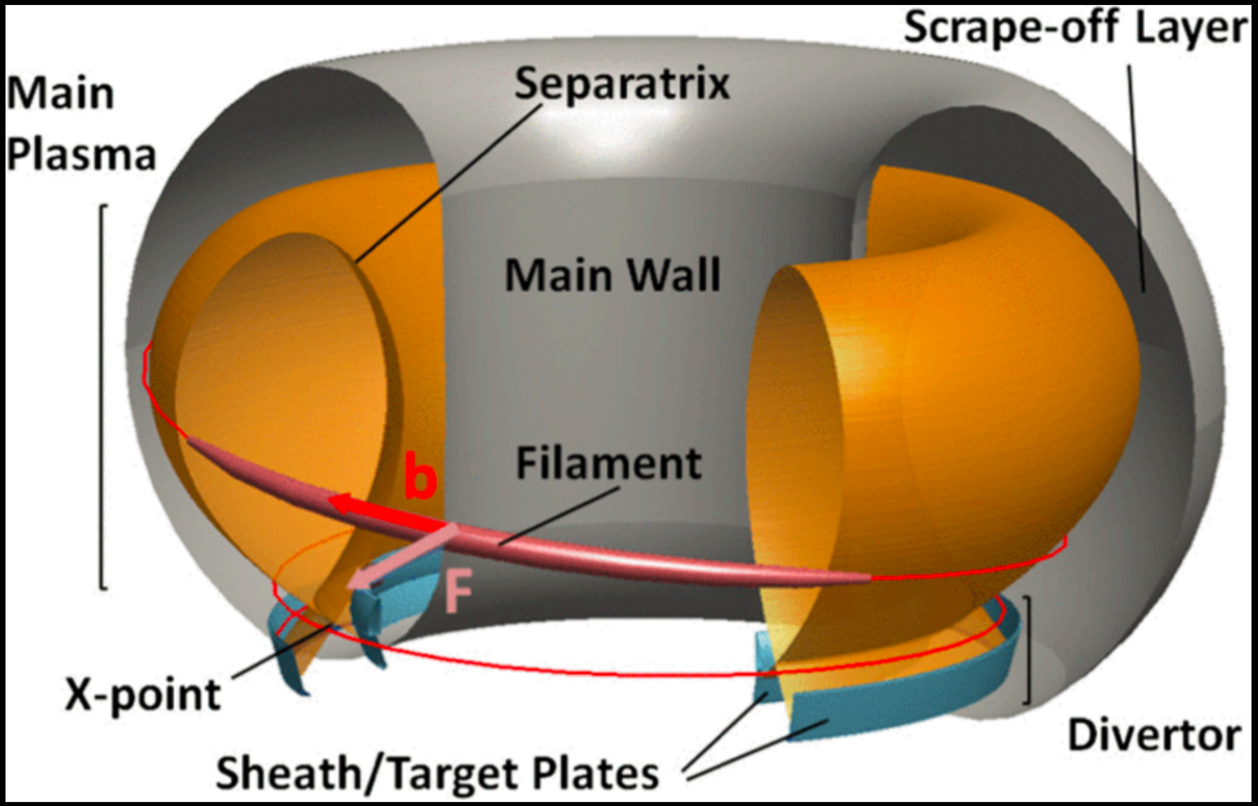
Modeling the NSTX SOL with `Gkeyll`

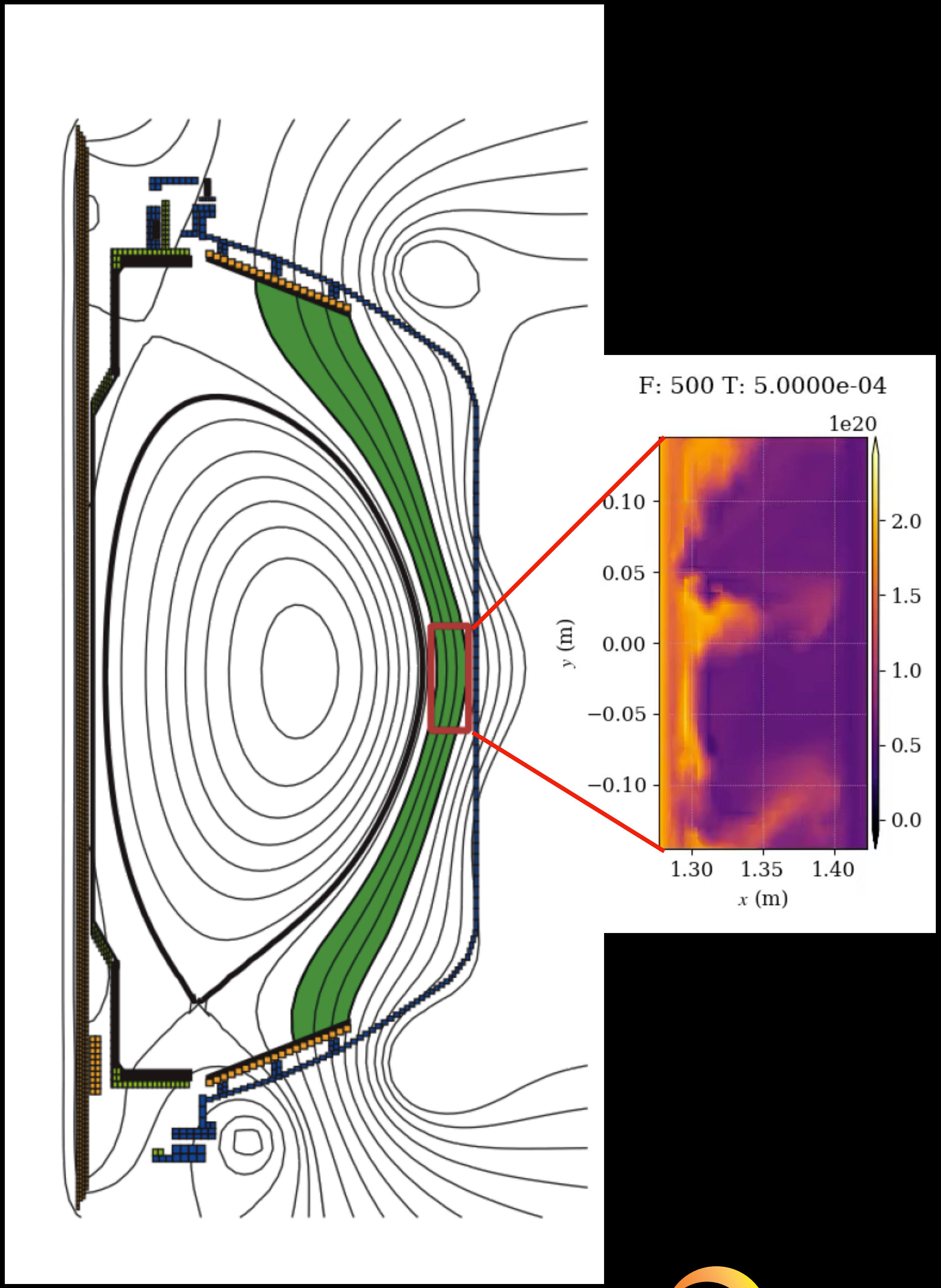


- Simple helical model of tokamak SOL
  - Field-aligned simulation domain that follows field lines from bottom divertor plate, around the torus, to the top divertor plate
  - Like the green region, but straightened out to vertical flux surfaces
  - Curvature drives interchange instability (like Rayleigh-Taylor, but with centrifugal force from curvature as “effective gravity”)



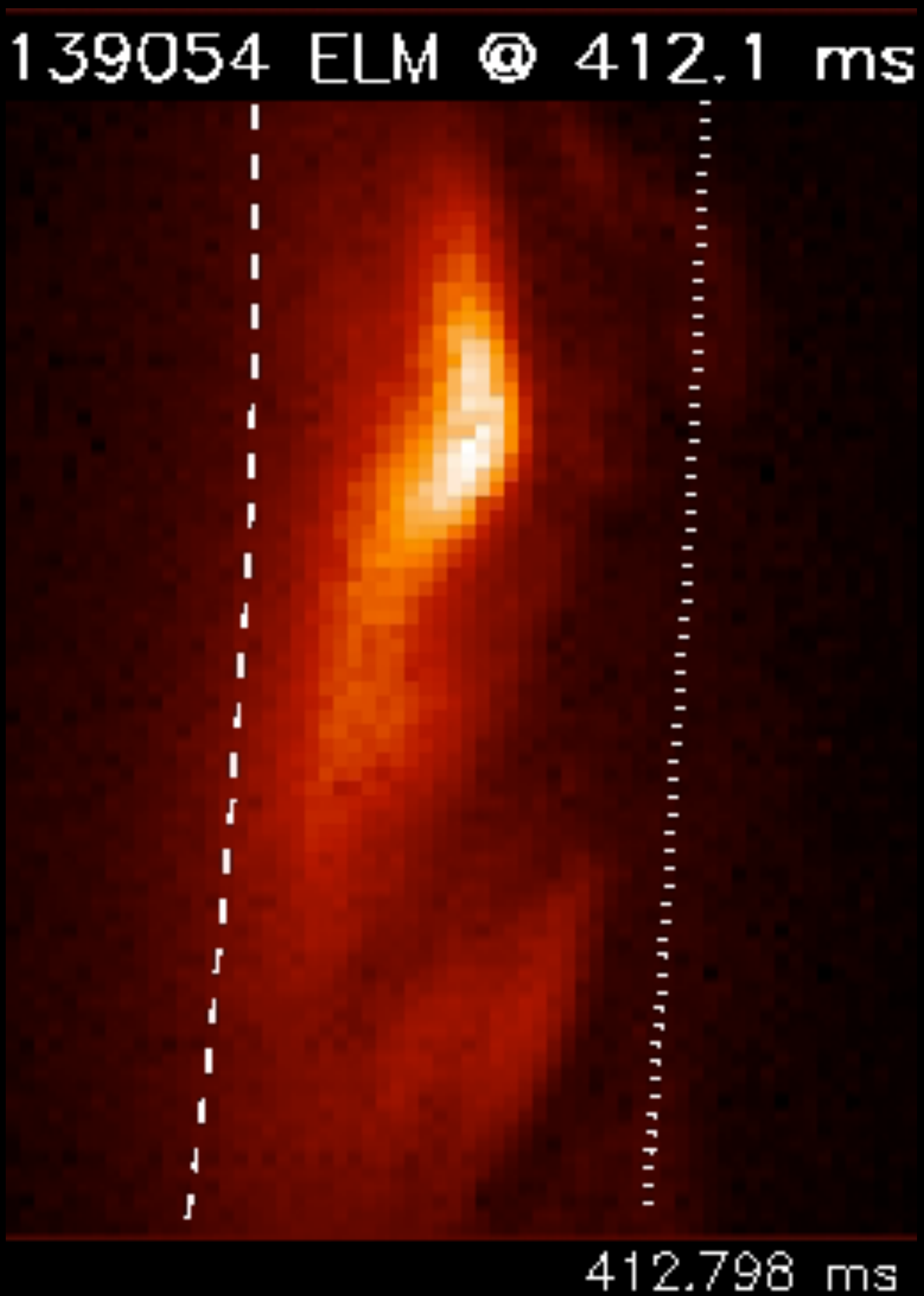
Modeling the NSTX SOL with `Gkeyll`

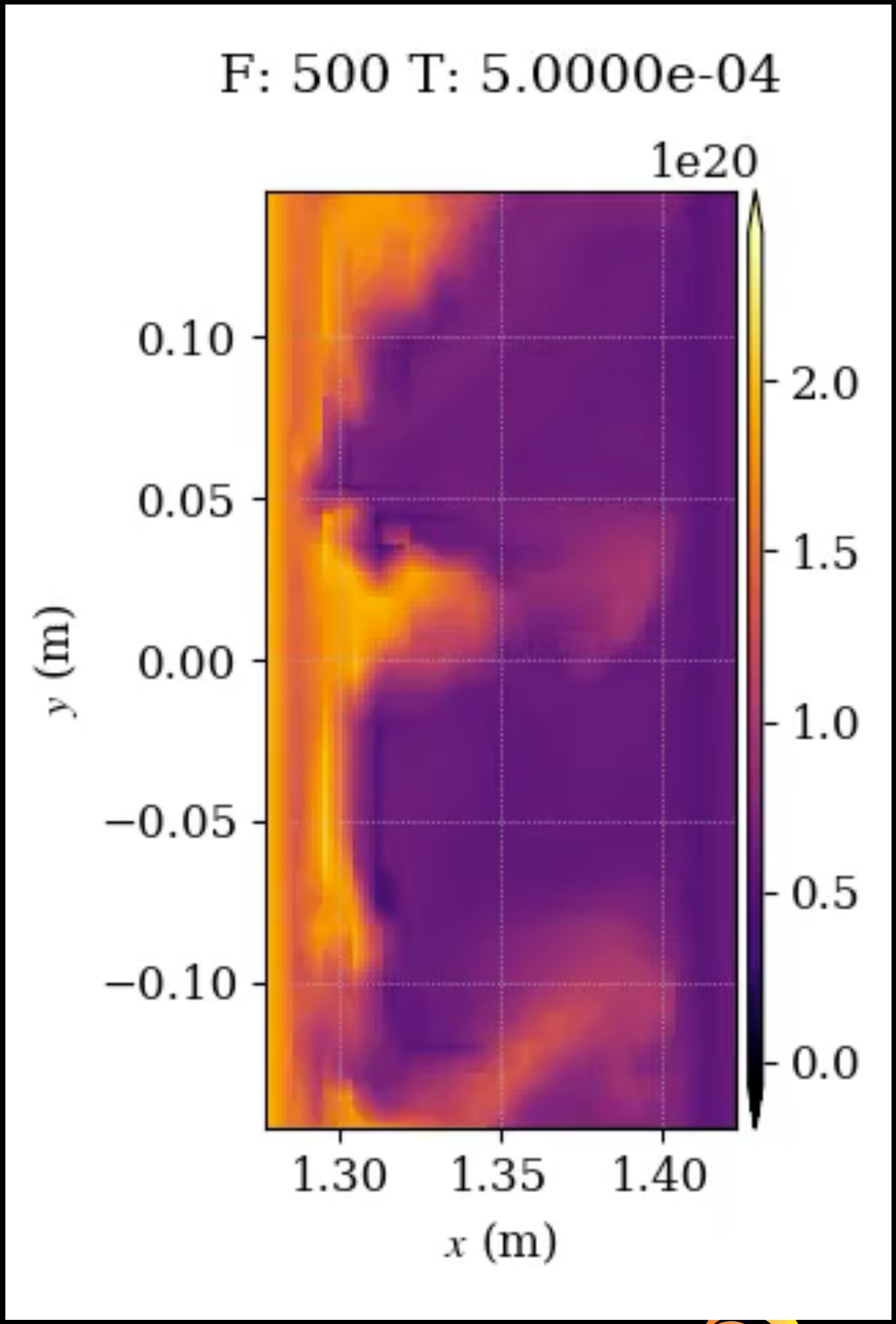




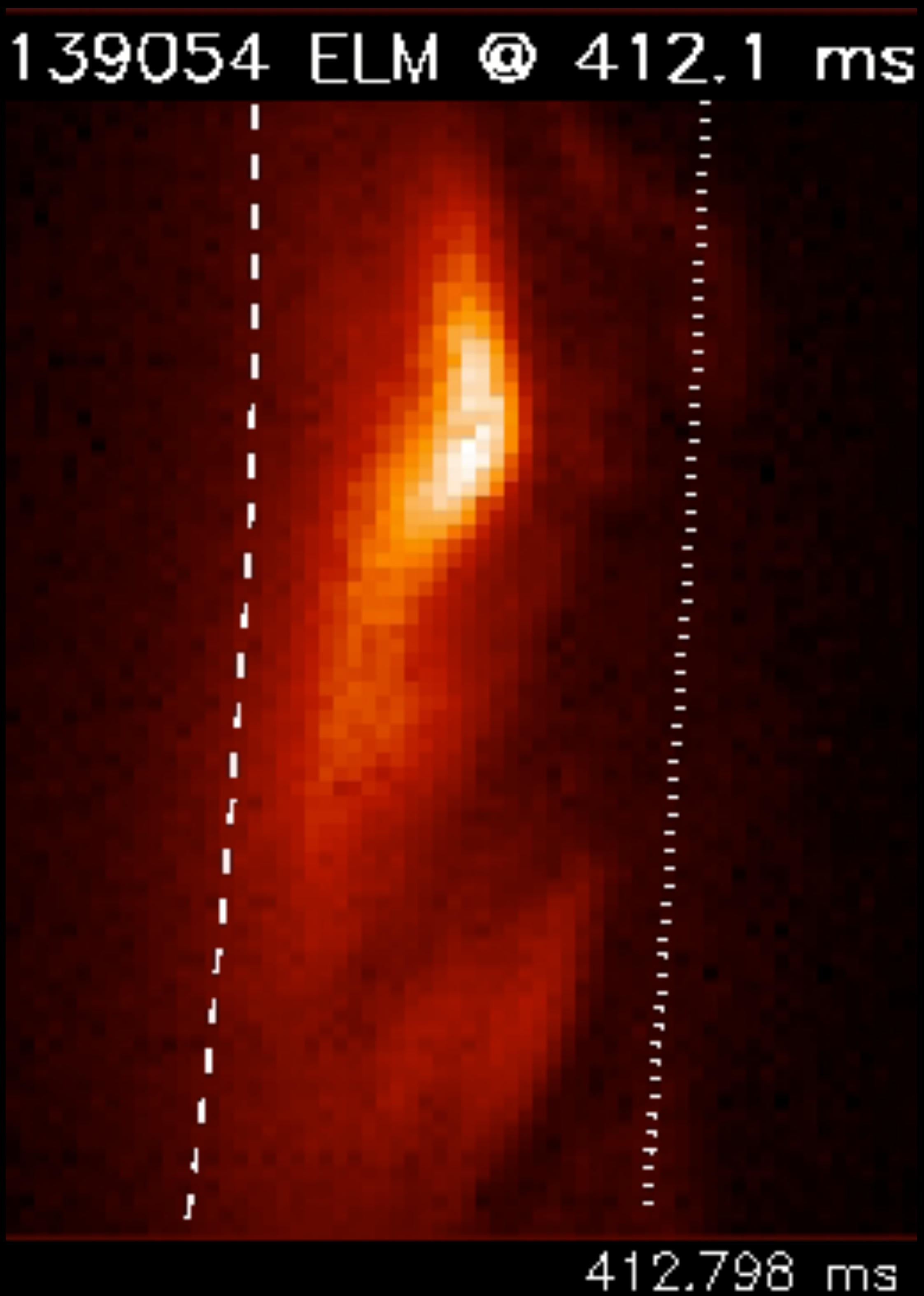
- Simple helical model of tokamak SOL
  - Field-aligned simulation domain that follows field lines from bottom divertor plate, around the torus, to the top divertor plate
  - Like the green region, but straightened out to vertical flux surfaces
  - Curvature drives interchange instability (like Rayleigh-Taylor, but with centrifugal force from curvature as “effective gravity”)

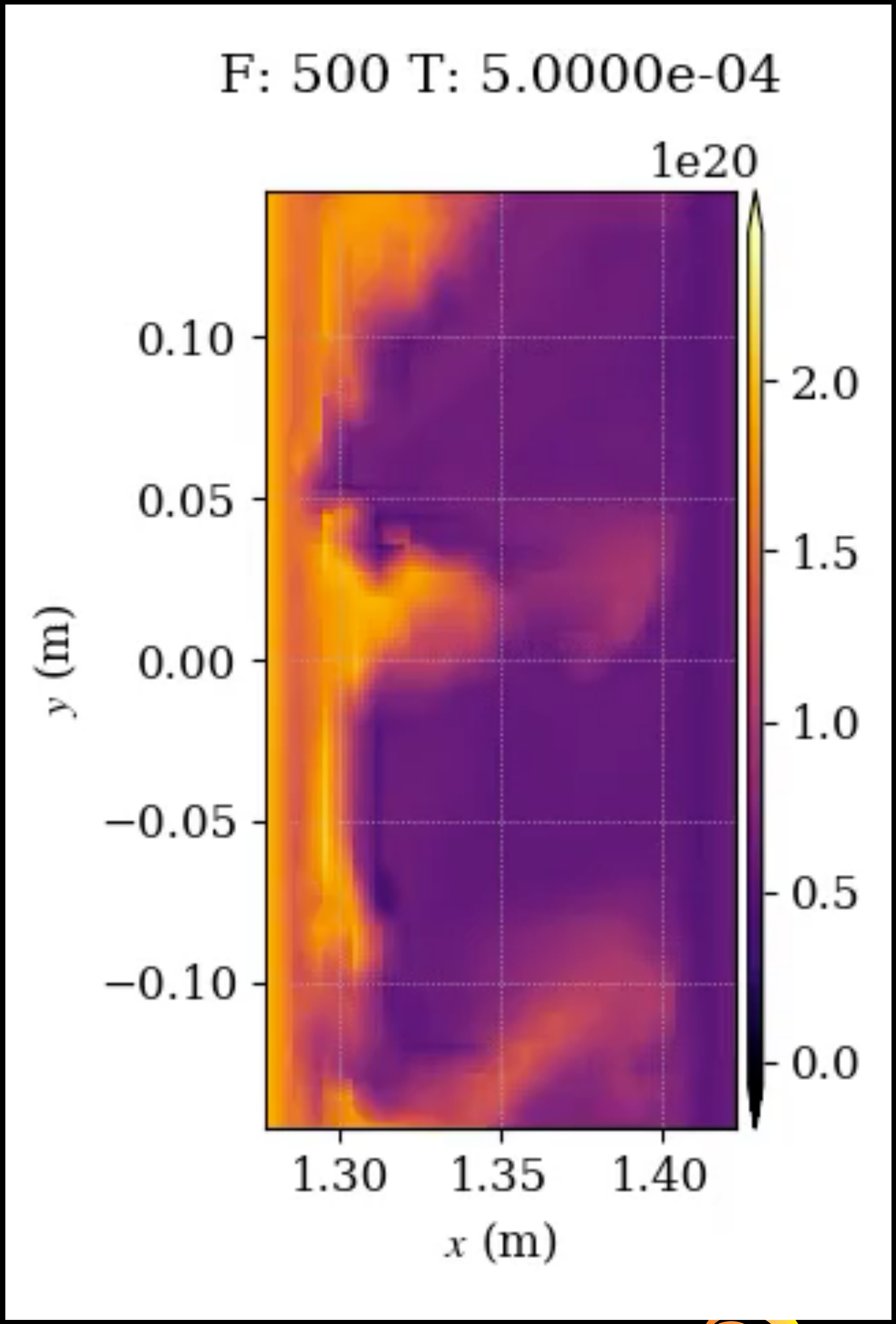
Modeling the NSTX SOL with `Gkeyll`





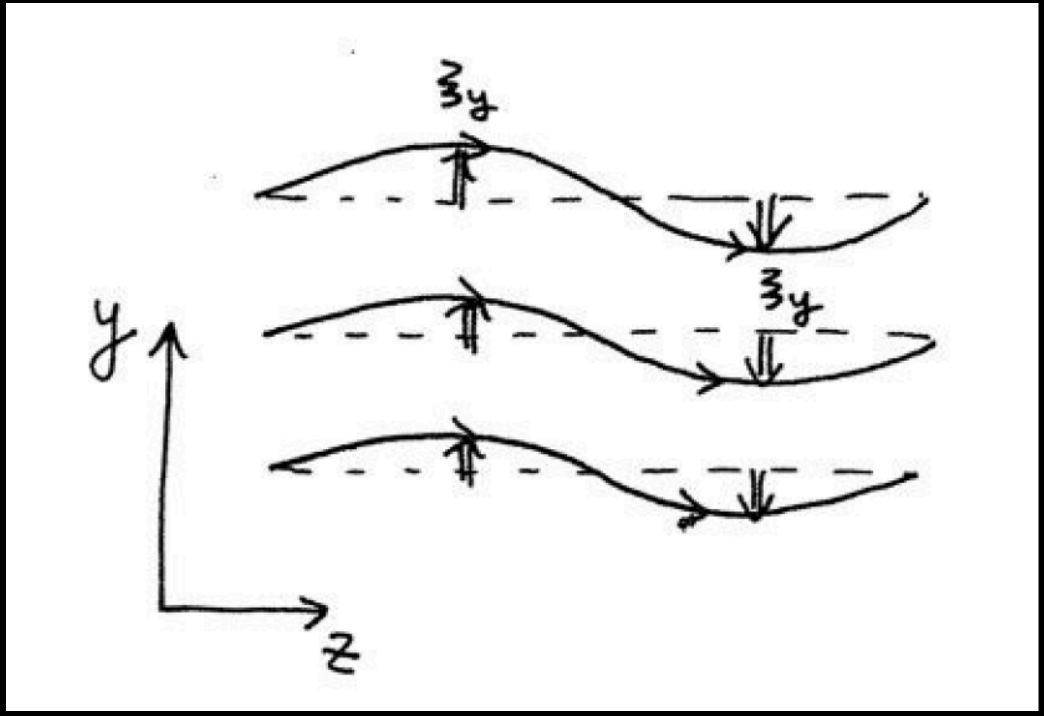
Modeling the NSTX SOL with `Gkeyll`



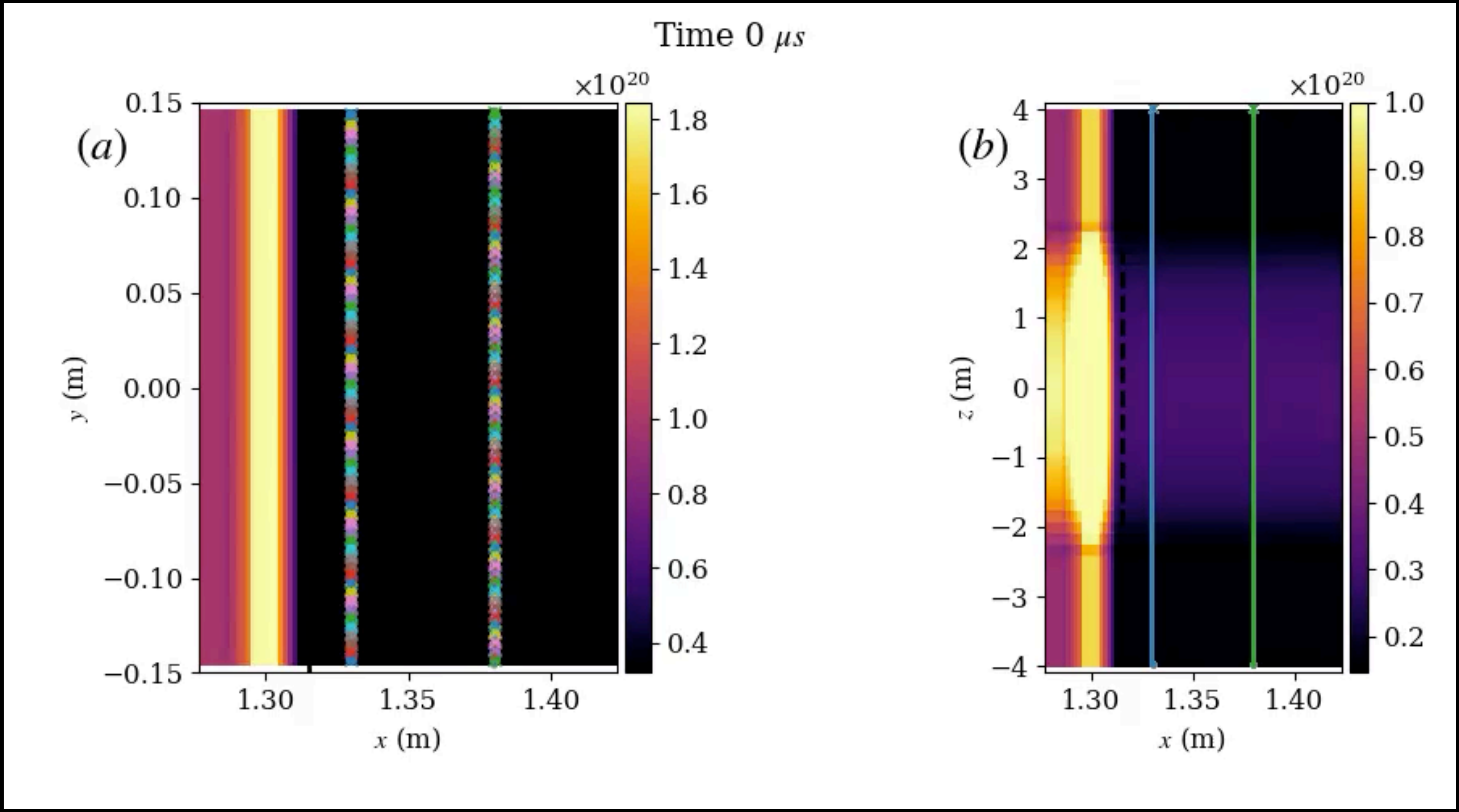


Including electromagnetic effects

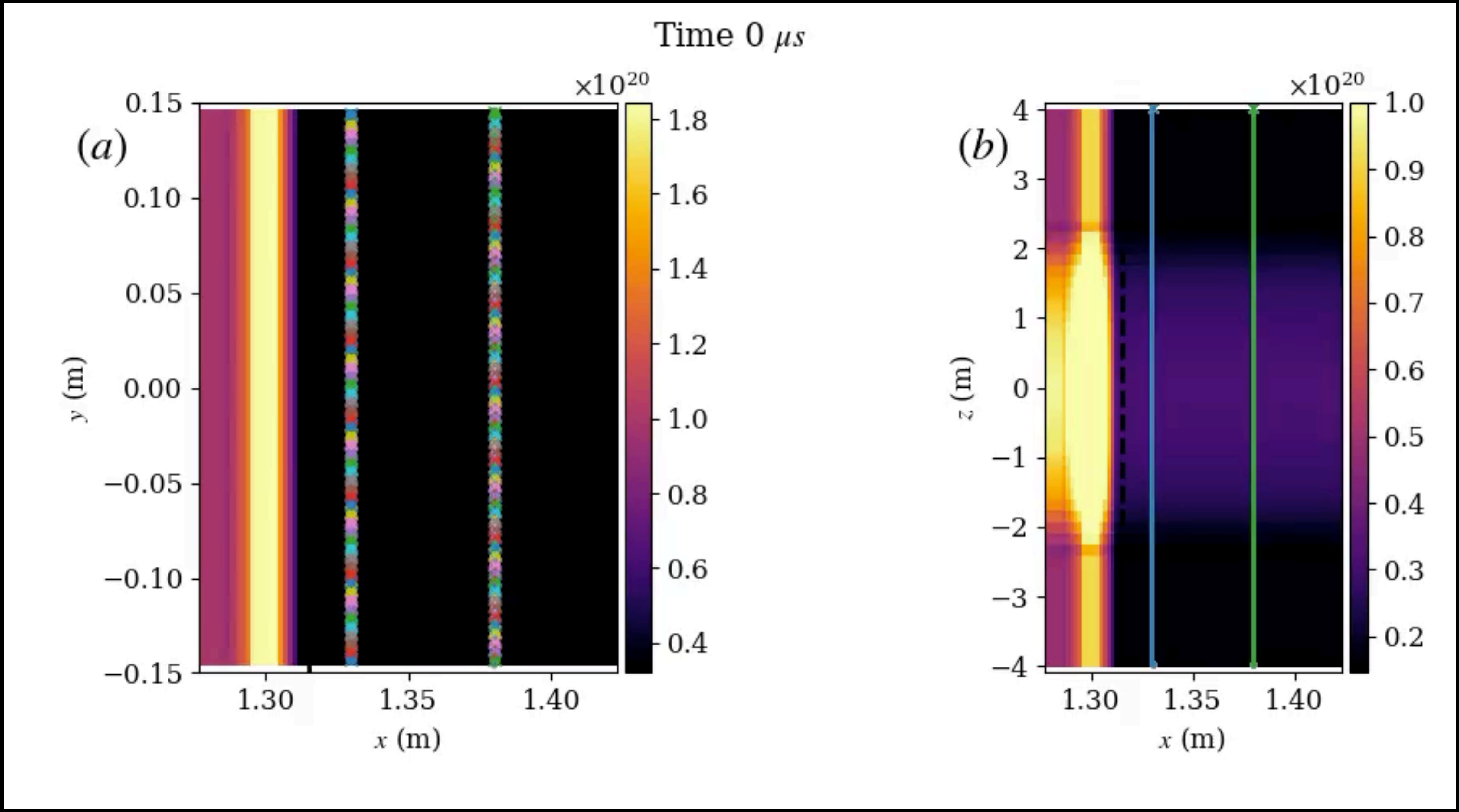
- SOL modeling usually uses an electrostatic approximation, which neglects magnetic perturbations
- In reality, magnetic field lines can bend
- Example: Alfven waves
  - Field lines behave like taut strings, “plucked” by plasma motion
  - Magnetic “tension” restoring force ~ B^2; string mass from plasma $\sim \rho$
  - Higher $\beta \sim \rho/B^2 \rightarrow$ larger magnetic perturbations



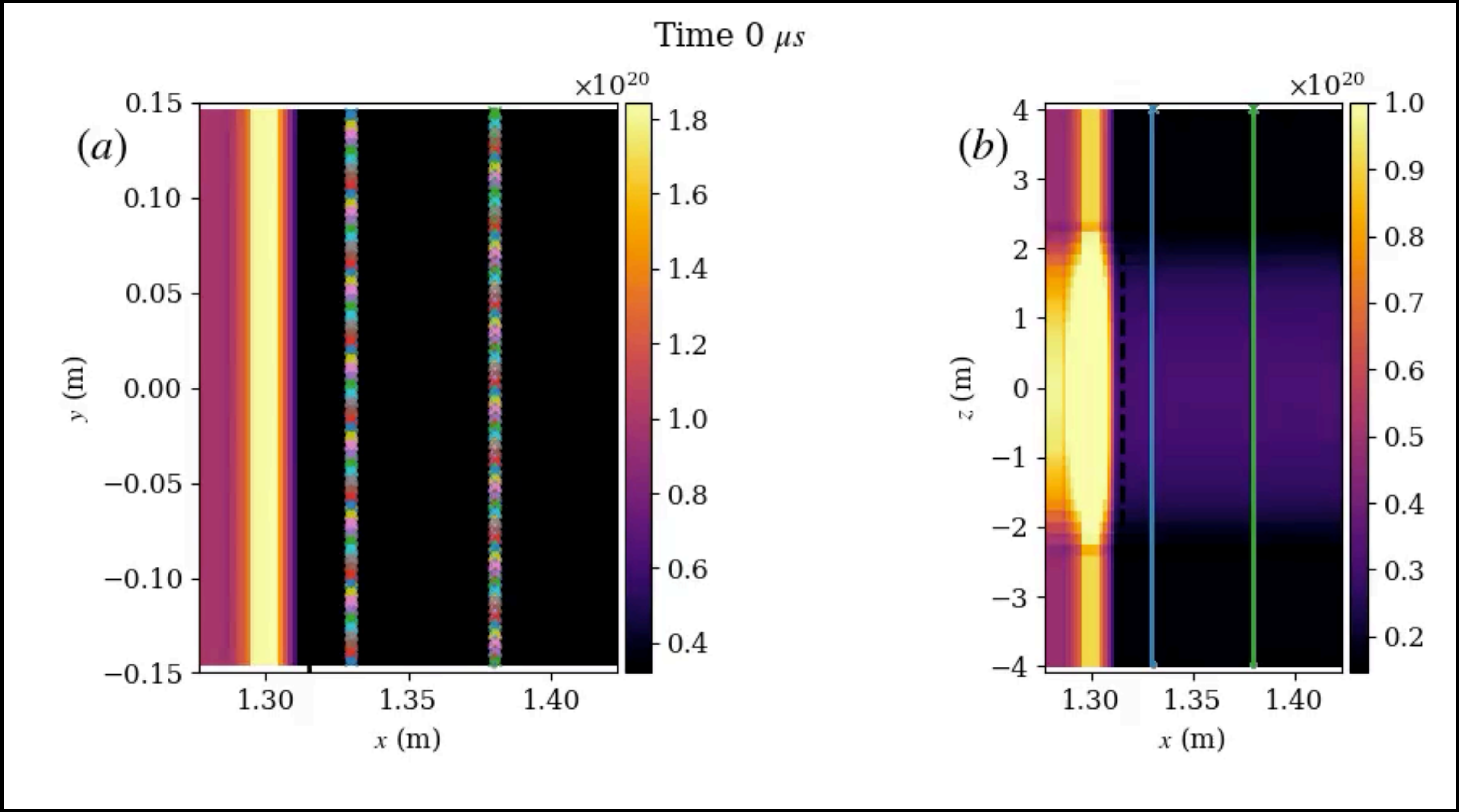
Electromagnetic GK in SOL



Electromagnetic GK in SOL

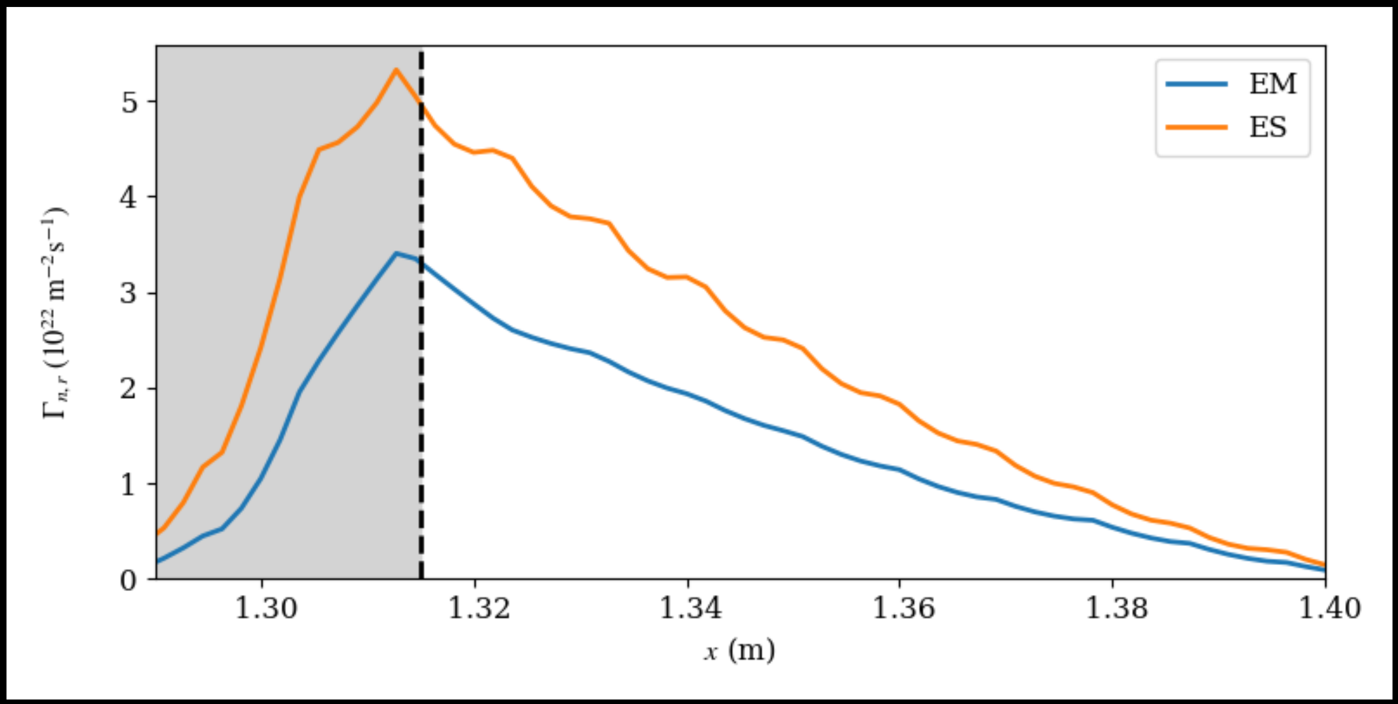


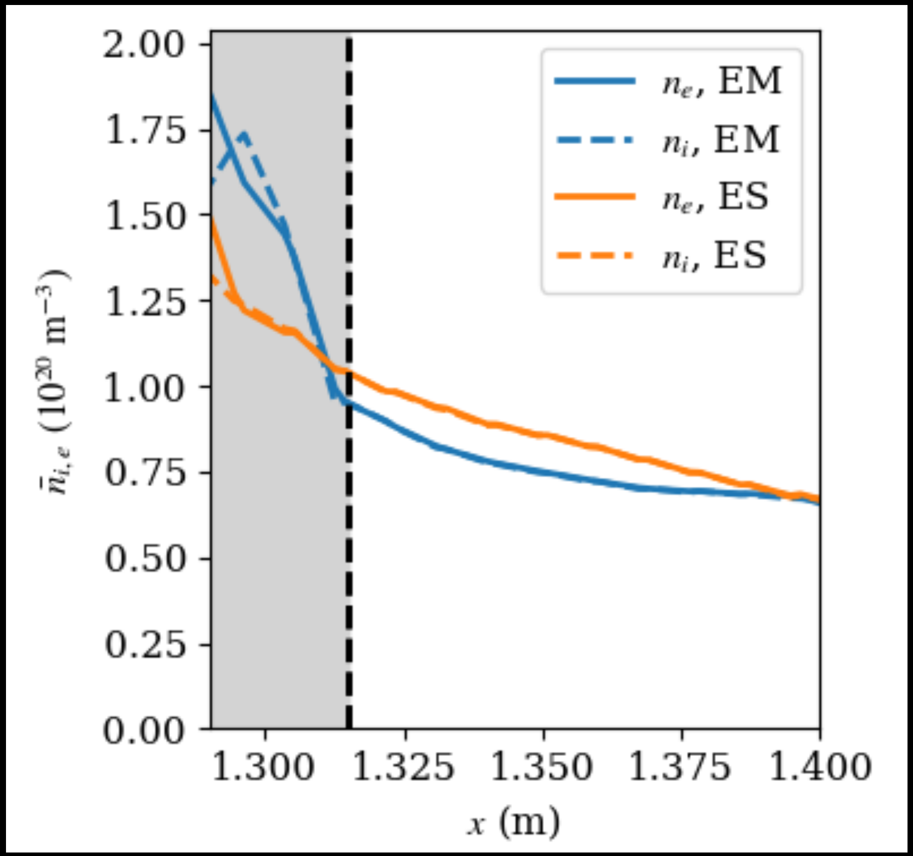
Electromagnetic GK in SOL



First ever electromagnetic GK simulations of SOL!

Does EM affect transport?





- Particle transport is reduced when EM effects included
- Results in flatter density profiles in SOL

PPPL

Thank you CSGF!!

DOE CSGF

PRINCETON PLASMA PHYSICS LABORATORY

PRINCETON UNIVERSITY

Gkeyll team:
- **Greg Hammett**
- **Ammar Hakim**
- **Jimmy Juno**
- **Mana Francisquez**
- **Tess Bernard**
- **Petr Cagas**
- **Eric Shi**

https://bitbucket.org/ammarhakim/gkyl/src/default/

https://gkeyll.readthedocs.io/en/latest/index.html