INSTITUTE OF PLASMA PHYSICS OF THE CZECH ACADEMY OF SCIENCES

Fluid transport modelling of the plasma core and edge

P. MACHA[1,2]

List of affiliations:
1) Institute of the plasma physics of the CAS
2) Faculty of nuclear science and physical engineering

EUROfusion

GAČR CZECH SCIENCE FOUNDATION

Tento projekt je spolufinancován se státní podporou Technologické agentury ČR v rámci Programu THÉTA.
www.tacr.cz
Výzkum užitečný pro společnost.

Outline

- Basics of fluid turbulent models.
- Simple 2D model.
  - Equations
  - Implementation
  - Results and limitations
- From kinetic to fluid equations to Braginskii equations.
- 3D GBS modell.
  - Inputs / outputs.
  - Implementation of the GBS code.
  - Results and limitations.
- Fluid codes on GPU using Python and JAX.

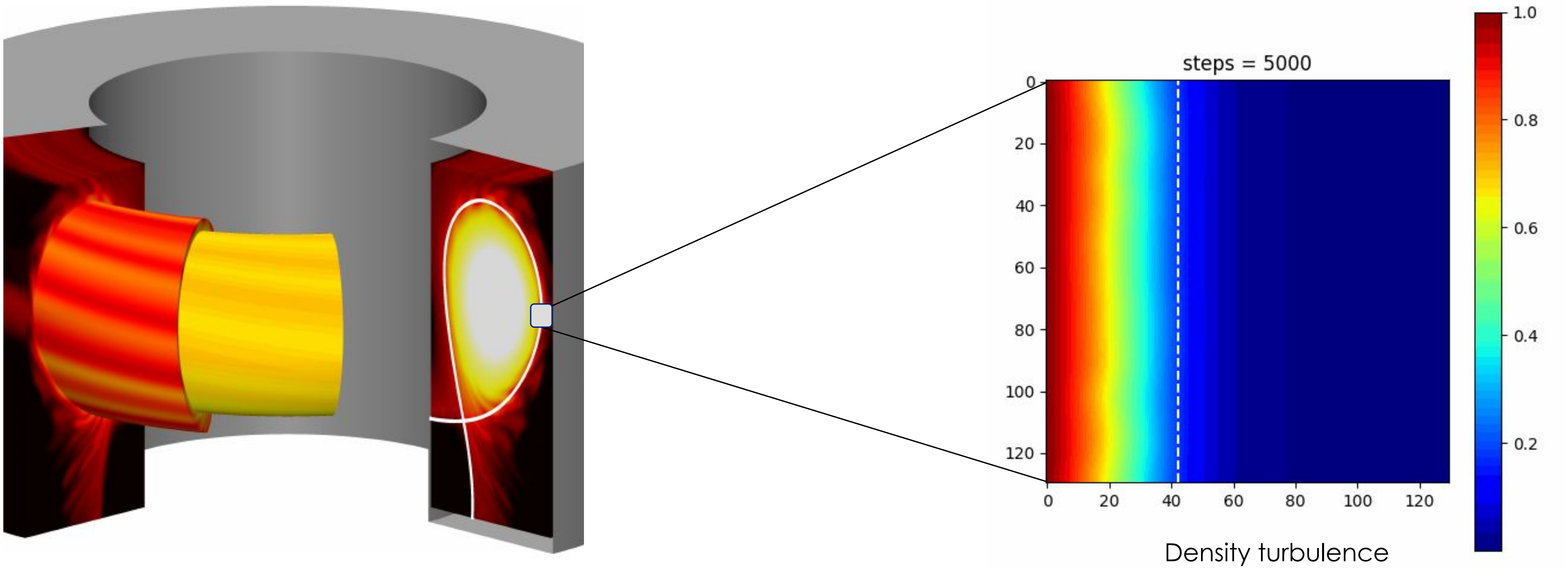
Fluid models basics

- From kinetic equation to fluid equations.
- **Maxwellian distribution is assumed (high collisionality)!**
- Several first moments (up to temperature equations) + closure.
- Much **faster** compared to kinetic simulations, several 3D models exists (**GBS**, TOKAM3X, GRILLIX).
- **Full-size simulations** of medium size machines (COMPASS, TCV, etc).
- Kinetic effects and gyromotions are neglected.
- Describes **edge plasma only** => unable to simulate core plasma (ITGs, ETGs, TEM neglected).

Turbulence v okrajovém plazmatu

Complex 3D model

Simple 2D model



Density turbulence

Towards simple 2D fluid model

Drift reduced approximation

- The momentum equation for each charged particle species reduced to an algebraic expression for the fluid drifts in terms of scalar fields.
- To **separate** the **parallel** and **perpendicular** motion.
- To remove fast temporal scales.
- Can be used because the **turbulence** is much **slower** compared to gyro-frequency and much **larger** compared to the gyro-radius.
- Perpendicular motion given by **ExB** drift, **diamagnetic** drift, and **polarization** drift.

$$\boldsymbol{v}_\perp = \overbrace{\frac{1}{B}\boldsymbol{b}\times\boldsymbol{\nabla}\phi}^{\text{ExB drift}} + \overbrace{\frac{1}{qnB}\boldsymbol{b}\times\boldsymbol{\nabla}p}^{\text{diamagnetic drift}} + \overbrace{\frac{m}{qB}\boldsymbol{b}\times\left(\frac{\partial}{\partial t}+\boldsymbol{v}\cdot\boldsymbol{\nabla}\right)\boldsymbol{v}}^{\text{polarization drift}},$$

IPP

Simple 2D model - equations

Perpendicular transport:

$$\boldsymbol{v}_\perp = \overbrace{\frac{1}{B}\boldsymbol{b}\times\boldsymbol{\nabla}\phi}^{\text{ExB drift}} + \overbrace{\frac{1}{qnB}\boldsymbol{b}\times\boldsymbol{\nabla}p}^{\text{diamagnetic drift}} + \overbrace{\frac{m}{qB}\boldsymbol{b}\times\left(\frac{\partial}{\partial t}+\boldsymbol{v}\cdot\boldsymbol{\nabla}\right)\boldsymbol{v}}^{\text{polarization drift}},$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{v}_E = \overbrace{\boldsymbol{\nabla}\left(\frac{1}{B}\right)\cdot\boldsymbol{b}\times\boldsymbol{\nabla}\phi}^{(1)} + \overbrace{\frac{1}{B}\boldsymbol{\nabla}\times\boldsymbol{b}\cdot\boldsymbol{\nabla}\phi}^{(2)} = \mathcal{C}(\phi),$$

Kinetic equation

$$\frac{\partial f}{\partial t} + \nabla\cdot(vf) + \nabla\cdot\left(\frac{F}{m}f\right) = C$$

Density equation

$$\frac{\partial n}{\partial t} + \boldsymbol{\nabla}\cdot(n\boldsymbol{v}) = 0,$$

Temperature equation

$$\frac{3}{2}n\left(\frac{\partial}{\partial t}+\boldsymbol{v}\cdot\boldsymbol{\nabla}\right)T + nT\boldsymbol{\nabla}\cdot\boldsymbol{v} + \boldsymbol{\nabla}\cdot\boldsymbol{q}_\perp = 0,$$

$$\begin{aligned}
\frac{\mathrm{d}n}{\mathrm{d}t} + n\mathcal{C}(\phi) - \mathcal{C}(nT) &= \Lambda(n)\\
\frac{\mathrm{d}T}{\mathrm{d}t} + \frac{2T}{3}\mathcal{C}(\phi) - \frac{7T}{3}\mathcal{C}(T) - \frac{2T^2}{3n}\mathcal{C}(n) &= \Lambda(T)\\
\frac{\mathrm{d}\Omega}{\mathrm{d}t} - \mathcal{C}(nT) &= \Lambda(\Omega)
\end{aligned}$$

$$\Omega = \boldsymbol{\nabla}\times v_E = B^{-2}\boldsymbol{\nabla}\times(\boldsymbol{B}\times\boldsymbol{\nabla}\phi) = \nabla_\perp^2\phi.$$

Simple 2D model - without temperature

2D model without temperature:

- **Turbulence** can be evolved even **without temperature**.
- Considering **constant temperature** - simplification.
- Poisson equation is unchanged.
- Numerical solution:
  - Poisson equation- Poisson solver
  - Operator d/dt
  - Curvature operator C(.)
  - Diffusion operator Λ

$$\frac{dn}{dt} + nC(\phi) - C(n) = \Lambda(n)$$

$$\frac{d\Omega}{dt} - C(n) = \Lambda(\Omega)$$

$$\Delta\phi = \Omega$$

Simple 2D model - implementation

$$\frac{dn}{dt} + nC(\phi) - C(n) = \Lambda(n)$$

Total time derivative -> time change + convection

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{\partial \phi}{\partial y} - \frac{\partial f}{\partial y}\frac{\partial \phi}{\partial x}$$

convection

Curvature operator - derivative in y direction

$$C(f) = -\frac{\partial f}{\partial y}$$

Diffusion term -> diffusion in x and y + parallel decay

$$\Lambda(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} - \frac{1}{\tau_{||}} f$$

1. derivative in x: $$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+\Delta h, y) - f(x-\Delta h, y)}{\Delta h}$$

2. derivative in y: $$\frac{\partial^2 f(x,y)}{\partial x^2} \approx \frac{f(x+\Delta h, y) - 2f(x,y) + f(x-\Delta h, y)}{\Delta h^2}$$

Simple 2D model - Arakawa scheme

- **Convection term** using Arakawa scheme for numerical stability.
- Arakawa conserves:
  - Mean vorticity
  - Mean-square vorticity
  - Kinetic energy

$$\partial\zeta/\partial t = (\partial\zeta/\partial x)(\partial\psi/\partial y) - (\partial\zeta/\partial y)(\partial\psi/\partial x) \equiv J(\zeta, \psi)$$

$$\begin{aligned}
J_{i,j}(\zeta, \psi) = -\frac{1}{12d^2} [&(\psi_{i,j-1} + \psi_{i+1,j-1} - \psi_{i,j+1} - \psi_{i+1,j+1})(\zeta_{i+1,j} - \zeta_{i,j}) \\
&+ (\psi_{i-1,j-1} + \psi_{i,j-1} - \psi_{i-1,j+1} - \psi_{i,j+1})(\zeta_{i,j} - \zeta_{i-1,j}) \\
&+ (\psi_{i+1,j} + \psi_{i+1,j+1} - \psi_{i-1,j} - \psi_{i-1,j+1})(\zeta_{i,j+1} - \zeta_{i,j}) \\
&+ (\psi_{i+1,j-1} + \psi_{i+1,j} - \psi_{i-1,j-1} - \psi_{i-1,j})(\zeta_{i,j} - \zeta_{i,j-1}) \\
&+ (\psi_{i+1,j} - \psi_{i,j+1})(\zeta_{i+1,j+1} - \zeta_{i,j}) \\
&+ (\psi_{i,j-1} - \psi_{i-1,j})(\zeta_{i,j} - \zeta_{i-1,j-1}) \\
&+ (\psi_{i,j+1} - \psi_{i-1,j})(\zeta_{i-1,j+1} - \zeta_{i,j}) \\
&+ (\psi_{i+1,j} - \psi_{i,j-1})(\zeta_{i,j} - \zeta_{i+1,j-1})], \qquad (45)
\end{aligned}$$

Simple 2D model - Poisson Solver

Solve Poisson equation in 2D using **periodical BCs in y**:

- The Poisson equation in 2D:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \Omega$$

- **Fourier transform** in y direction:

$$\frac{\partial^2 \hat{\phi}}{\partial x^2} - k^2 \hat{\phi} = \hat{\Omega}$$

- Discretization - solving for phi:

$$\frac{\hat{\phi}_{i+1,j} - 2\hat{\phi}_{i,j} + \hat{\phi}_{i-1,j}}{\Delta x^2} - k^2 \hat{\phi}_{i,j} = \hat{\Omega}_{i,j}$$

Simple 2D model - Poisson Solver

Solve Poisson equation in 2D using **general BCs**:

- The Poisson equation in 2D:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \Omega$$

- Fourier transform cannot be used (no periodic BCs) -> **finite difference** matrix solver [1]:

$$\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2} = \Omega_{i,j}$$

Simple 2D model - Boundary conditions

Left:

- Temperature and density set to 1 (normalization).

Right:

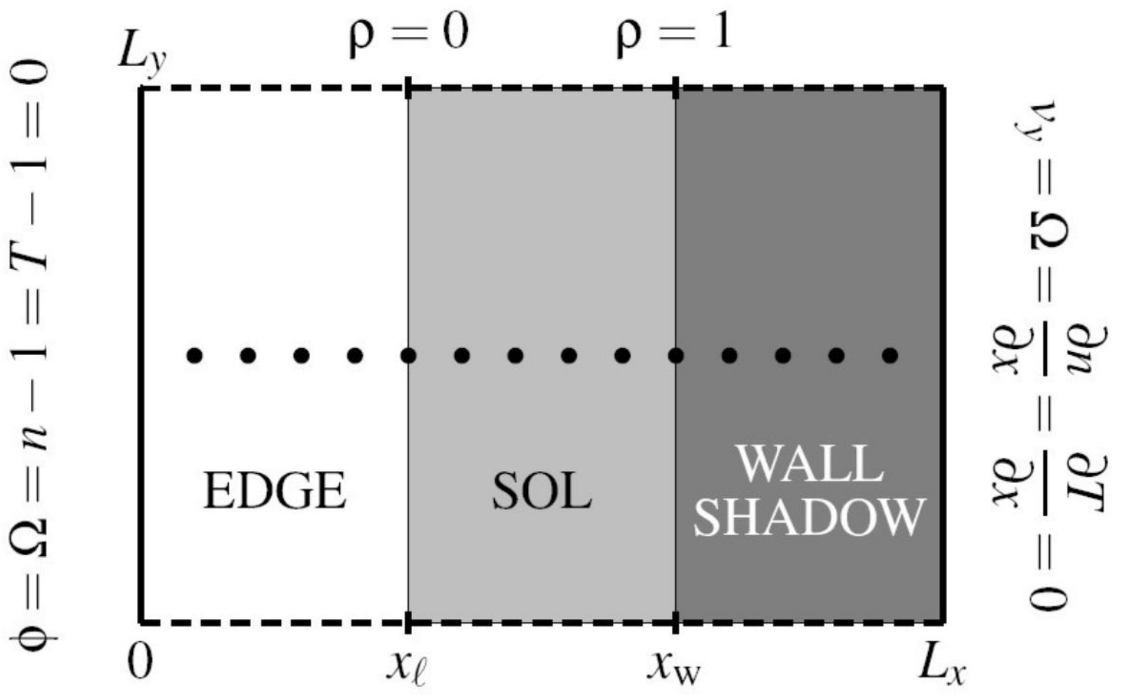
- Neumann for temperature, density and potential.
- Dirichlet for vorticity.

Top and bottom:

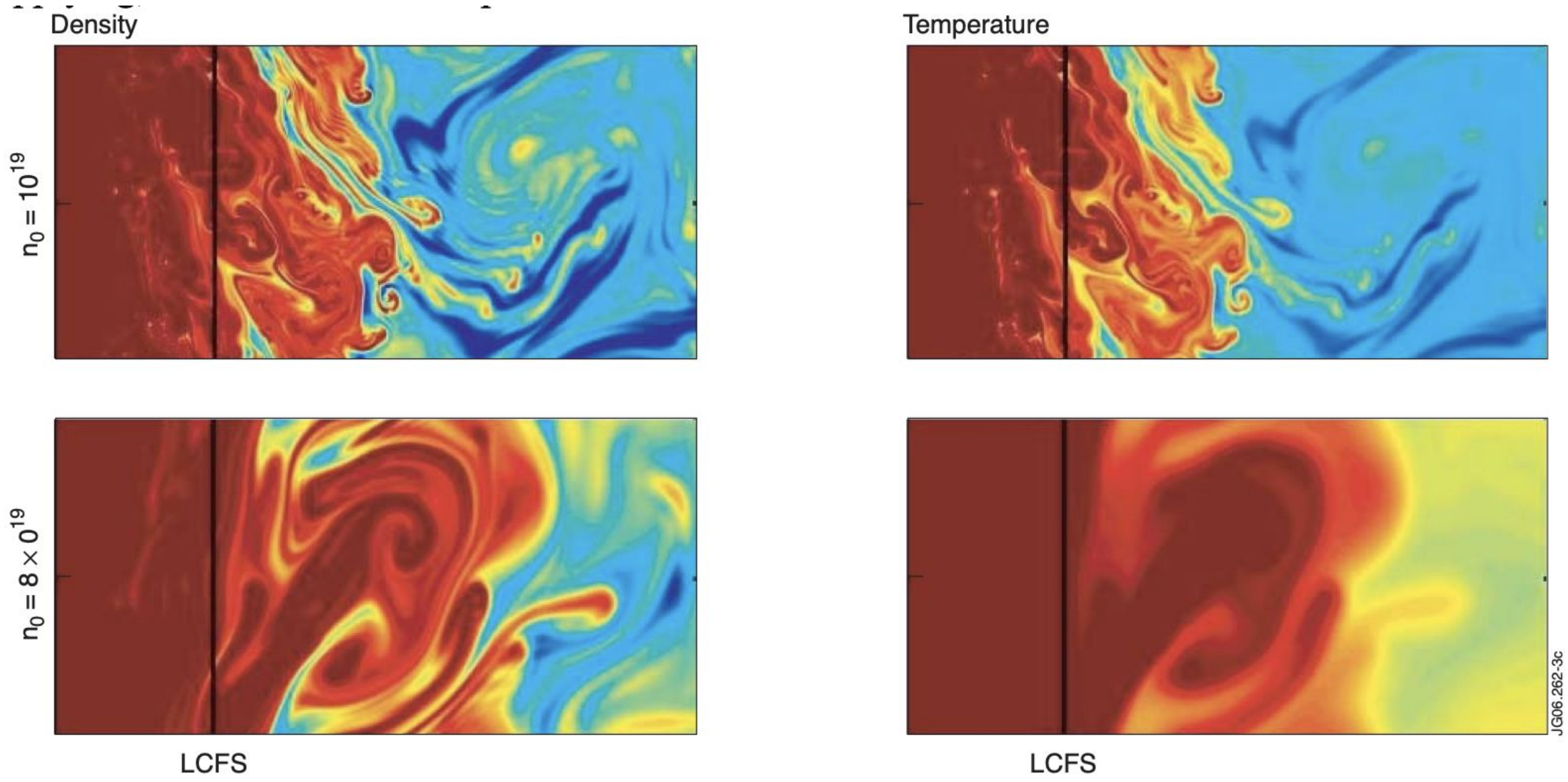
- Periodic boundary conditions for all the fields.

Parallel transport:

- Exponential decay in SOL and WALL SHADOW.
- Represents region of open / closed mg. field lines.



Simple 2D model - results



21st IAEA Fusion Energy Conference

Simple 2D model - limits

Advantages

- Reduced computational resources.
- Faster simulation time
- Easier interpretation of the results.
- Easier implementation (simple equations).
- Some processes can be reasonably approximated by 2D model.
- Validation of more complex 3D codes.

Disadvantages

- Limited accuracy (neglects 3rd dimension).
- Oversimplification (some processes cannot be described in 2D).
- Not possible to perform full-size simulation.
- Cannot describe the complex tokamak geometry.

IPP

Towards complex 3D fluid model

Towards Braginskii equations I

$$\frac{\partial f}{\partial t} + \nabla \cdot (vf) + \nabla \cdot (\frac{F}{m} f) = C$$

$$\frac{\partial n_e}{\partial t} = -\nabla \cdot (n_e \mathbf{v}_e) + n_n \nu_{iz} - n_i \nu_{rec} + S_n$$

$$\frac{\partial n_i}{\partial t} = -\nabla \cdot (n_i \mathbf{v}_i) + n_n \nu_{iz} - n_i \nu_{rec} + S_n$$

$$m_e n_e \frac{d_e \mathbf{v}_e}{dt} = -\nabla p_e - \nabla \cdot \Pi_e - e n_e [\mathbf{E} + \mathbf{v}_e \mathbf{B}] + \mathbf{R}_{ei} + m_e n_n (\nu_{en} + 2\nu_{iz})(\mathbf{v}_n - \mathbf{v}_e)$$

$$m_i n_i \frac{d_i \mathbf{v}_i}{dt} = -\nabla p_i - \nabla \cdot \Pi_i - Z_i e n_i [\mathbf{E} + \mathbf{v}_i \mathbf{B}] - \mathbf{R}_{ei} + m_i n_n (\nu_{iz} + \nu_{cx})(\mathbf{v}_n - \mathbf{v}_i)$$

$$\frac{3}{2} n_e \frac{d_e T_e}{dt} = -p_e \nabla \cdot \mathbf{v}_e - \nabla \cdot \mathbf{q}_e - \Pi_e : \nabla \mathbf{v}_e + Q_e + n_n \nu_{iz} [-E_{iz} - \frac{3}{2} m_e \mathbf{v}_e \cdot (\mathbf{v}_e - \frac{4}{3} \mathbf{v}_n)] - n_n \nu_{en} m_e \mathbf{v}_e \cdot (\mathbf{v}_n - \mathbf{v}_e) + \frac{3}{2} n_e S_{Te}$$

$$\frac{3}{2} n_i \frac{d_i T_i}{dt} = -p_i \nabla \cdot \mathbf{v}_i - \nabla \cdot \mathbf{q}_i - \Pi_i : \nabla \mathbf{v}_i + Q_i + n_n (\nu_{iz} + \nu_{cx}) [\frac{3}{2}(T_n - T_i) + \frac{m_i}{2} (\mathbf{v}_n - \mathbf{v}_i)^2] + \frac{3}{2} n_i S_{Ti}$$

Towards Braginskii equations II

- Assumptions on drift reduced limit, electrostatic limit, etc…

$$\frac{\partial n}{\partial t} + \nabla \cdot (\mathbf{V}_{\mathbf{E}\times\mathbf{B}} + \mathbf{V}_{\text{dia,e}} + \mathbf{V}_{\|\text{e}}) = 0$$

$$\frac{nc}{B\omega_\text{i}} \frac{d}{dt}\left(-\nabla_\perp^2 \phi - \frac{1}{en}\nabla_\perp^2 p_\text{i}\right) + \frac{1}{3m_\text{i}\omega_\text{i}}\mathbf{b} \times \kappa \cdot \nabla G_\text{i} + \nabla_\| \frac{j_\|}{e} +$$
$$\nabla \cdot n\left(\mathbf{V}_{\text{dia,i}} - \mathbf{V}_{\text{dia,e}}\right) = 0$$

$$m_\text{e}\frac{dV_{\|\text{e}}}{dt} = -\frac{1}{n}\nabla_\| p_\text{e} - \frac{2}{3}\nabla_\| G_\text{e} + e\nabla_\|\phi - \frac{e}{c}\frac{\psi}{\partial t} + e\frac{j_\|}{\sigma_\|} - 0.71\nabla_\| T_\text{e}$$

$$m_\text{i}\frac{dV_{\|\text{i}}}{dt} = -\frac{1}{n}\nabla(p_\text{i} + p_\text{e}) - p_\text{i}\nabla \times \frac{\mathbf{b}}{\omega_\text{i}} \cdot \nabla V_{\|\text{i}} - \frac{2}{3}\nabla_\| G_\text{i}$$

$$\frac{3}{2}n\frac{dT_\text{i}}{dt} + \frac{3}{2}n\mathbf{V}_{\text{dia,e}} \cdot \nabla T_\text{e} + p_\text{e}\nabla \cdot \left(\mathbf{V}_{\perp\text{e}} + \mathbf{V}_{\|\text{e}}\right) - \frac{5}{2}\frac{c}{e}\nabla \cdot p_\text{e}\left(\frac{\mathbf{b}}{B} \times \nabla T_\text{e}\right) -$$

$$0.71 T_\text{e}\nabla_\| j_\| - \nabla \cdot \left(\chi_{\|\text{e}}\nabla_{\|\text{T}_\text{e}}\right) = 0$$

$$\frac{3}{2}n\frac{dT_\text{i}}{dt} + T_\text{i}\left[m \cdot \left(\mathbf{V}_{\mathbf{EB}} + \mathbf{V}_{\|\text{e}}\right) + \nabla \cdot \left(n\mathbf{V}_{\text{dia,e}}\right)\right] + \frac{5}{2}\frac{c}{e}p_\text{i}\left(\nabla\frac{\mathbf{b}}{B}\right) \cdot \nabla T_\text{i} = 0$$

Towards Braginskii equations III

$$\frac{\partial n}{\partial t} = -\frac{1}{B}[\phi, n] + \frac{2}{eB}\Big[C(p_e) - nC(\phi)\Big] - \nabla_\parallel(n v_{\parallel e}) + D_n \nabla_\perp^2 n + s_n + \nu_{\mathrm{iz}} n_{\mathrm{n}} - \nu_{\mathrm{rec}} n, \quad (1)$$

$$\frac{\partial \Omega}{\partial t} = -\frac{1}{B}\nabla\cdot[\phi, \omega] - \nabla\cdot\left(v_{\parallel i}\nabla_\parallel \omega\right) + \frac{B\Omega_{ci}}{e}\nabla_\parallel j_\parallel + \frac{2\Omega_{ci}}{e} C(p_e + p_i) + \frac{\Omega_{ci}}{3e} C(G_i) + D_\Omega \nabla_\perp^2 \Omega - \frac{n_{\mathrm{n}}}{n}\nu_{\mathrm{cx}}\Omega, \quad (2)$$

$$\frac{\partial U_{\parallel e}}{\partial t} = -\frac{1}{B}[\phi, v_{\parallel e}] - v_{\parallel e}\nabla_\parallel v_{\parallel e} + \frac{e}{m_e}\left(\frac{j_\parallel}{\sigma_\parallel} + \nabla_\parallel \phi - \frac{1}{en}\nabla_\parallel p_e - \frac{0.71}{e}\nabla_\parallel T_e - \frac{2}{3en}\nabla_\parallel G_e\right) + D_{v_{\parallel e}} \nabla_\perp^2 v_{\parallel e} + \frac{n_{\mathrm{n}}}{n}(\nu_{\mathrm{en}} + 2\nu_{\mathrm{iz}})(v_{\parallel n} - v_{\parallel e}), \quad (3)$$

$$\frac{\partial v_{\parallel i}}{\partial t} = -\frac{1}{B}[\phi, v_{\parallel i}] - v_{\parallel i}\nabla_\parallel v_{\parallel i} - \frac{1}{m_i n}\nabla_\parallel(p_e + p_i) - \frac{2}{3 m_i n}\nabla_\parallel G_i + D_{v_{\parallel i}} \nabla_\perp^2 v_{\parallel i} + \frac{n_{\mathrm{n}}}{n}(\nu_{\mathrm{iz}} + \nu_{\mathrm{cx}})(v_{\parallel n} - v_{\parallel i}), \quad (4)$$

$$\frac{\partial T_e}{\partial t} = -\frac{1}{B}[\phi, T_e] - v_{\parallel e}\nabla_\parallel T_e + \frac{2}{3}T_e\left[0.71\frac{\nabla_\parallel j_\parallel}{en} - \nabla_\parallel v_{\parallel e}\right] + \frac{4}{3}\frac{T_e}{eB}\left[\frac{7}{2}C(T_e) + \frac{T_e}{n}C(n) - eC(\phi)\right] + \nabla_\parallel(\chi_{\parallel e}\nabla_\parallel T_e) + D_{T_e}\nabla_\perp^2 T_e + s_{T_e} - \frac{n_{\mathrm{n}}}{n}\nu_{\mathrm{en}} m_e \frac{2}{3} v_{\parallel e}(v_{\parallel n} - v_{\parallel e}) - 2\frac{m_e}{m_i}\frac{1}{\tau_e}(T_e - T_i) + \frac{n_{\mathrm{n}}}{n}\nu_{\mathrm{iz}}\left[-\frac{2}{3}E_{\mathrm{iz}} - T_e + m_e v_{\parallel e}\left(v_{\parallel e} - \frac{4}{3}v_{\parallel n}\right)\right], \quad (5)$$

$$\frac{\partial T_i}{\partial t} = -\frac{1}{B}[\phi, T_i] - v_{\parallel i}\nabla_\parallel T_i + \frac{4}{3}\frac{T_i}{eB}\left[C(T_e) + \frac{T_e}{n}C(n) - eC(\phi)\right] - \frac{10}{3}\frac{T_i}{eB}C(T_i) + \frac{2}{3}T_i\left[(v_{\parallel i} - v_{\parallel e})\frac{\nabla_\parallel n}{n} - \nabla_\parallel v_{\parallel e}\right] + \nabla_\parallel(\chi_{\parallel i}\nabla_\parallel T_i) + D_{T_i}\nabla_\perp^2 T_i + s_{T_i} + 2\frac{m_e}{m_i}\frac{1}{\tau_e}(T_e - T_i) + \frac{n_{\mathrm{n}}}{n}(\nu_{\mathrm{iz}} + \nu_{\mathrm{cx}})\left[T_n - T_i + \frac{1}{3}(v_{\parallel n} - v_{\parallel i})^2\right], \quad (6)$$

convection $$[\phi, f] = \mathbf{b}\cdot(\nabla\phi \times \nabla f)$$

curvature $$C(f) = \frac{B}{2}\left(\nabla \times \frac{\mathbf{b}}{B}\right)\cdot\nabla f$$

parallel gadient $$\nabla_\parallel f = \mathbf{b}\cdot\nabla f + \frac{1}{B}[\psi, f]$$

perp. Laplace $$\nabla_\perp^2 f = \nabla\cdot\left[(\mathbf{b}\times\nabla f)\times\mathbf{b}\right]$$

Gyroviscous terms
$$G_i = -\eta_{0i}\left[2\nabla_\parallel v_{\parallel i} + \frac{1}{B}C(\phi) + \frac{1}{enB}C(p_i)\right]$$
$$G_e = -\eta_{0e}\left[2\nabla_\parallel v_{\parallel e} + \frac{1}{B}C(\phi) - \frac{1}{enB}C(p_e)\right]$$

Operators used in fluid codes

Poisson bracket

$$[\phi, f] = \frac{B_\varphi}{B}(\partial_Z\phi\,\partial_R f - \partial_R\phi\,\partial_Z f)$$

Curvature operator

$$C(f) = \frac{B_\varphi}{B_0}\partial_Z f$$

Perpendicular laplacian

$$\nabla_\perp^2 f = \partial_{RR}^2 f + \partial_{ZZ}^2 f$$

Parallel laplacian

$$\nabla_{||} f = \partial_Z\Psi\,\partial_R f - \partial_R\Psi\,\partial_Z f + \frac{B_\varphi}{B_0}\partial_\varphi f$$

IPP

Derivatives in 3D model

$$(\partial_Z f)_{i,j,k} = \frac{1}{\Delta Z}\left(\frac{1}{12} f_{i,j,k-2} - \frac{2}{3} f_{i,j,k-1} + \frac{2}{3} f_{i,j,k+1} - \frac{1}{12} f_{i,j,k+2}\right)$$

$$(\partial_{ZZ} f)_{i,j,k} = \frac{1}{\Delta Z^2}\left(-\frac{1}{12} f_{i,j,k-2} + \frac{4}{3} f_{i,j,k-1} - \frac{5}{2} f_{i,j,k} + \frac{4}{3} f_{i,j,k+1} - \frac{1}{12} f_{i,j,k+2}\right)$$

- Poisson equation:

$$\nabla \cdot (n \nabla_\perp \phi) = \Omega - \frac{\nabla_\perp^2 p_i}{e}$$

- Ampere equation:

$$\left(\nabla_\perp^2 - \frac{e^2 \mu_0}{m_e} n\right) v_{\parallel e} = \nabla_\perp^2 U_{\parallel e} - \frac{e^2 \mu_0}{m_e} n v_{\parallel i} + \frac{e^2 \mu_0}{m_e} \bar{j}_\parallel$$

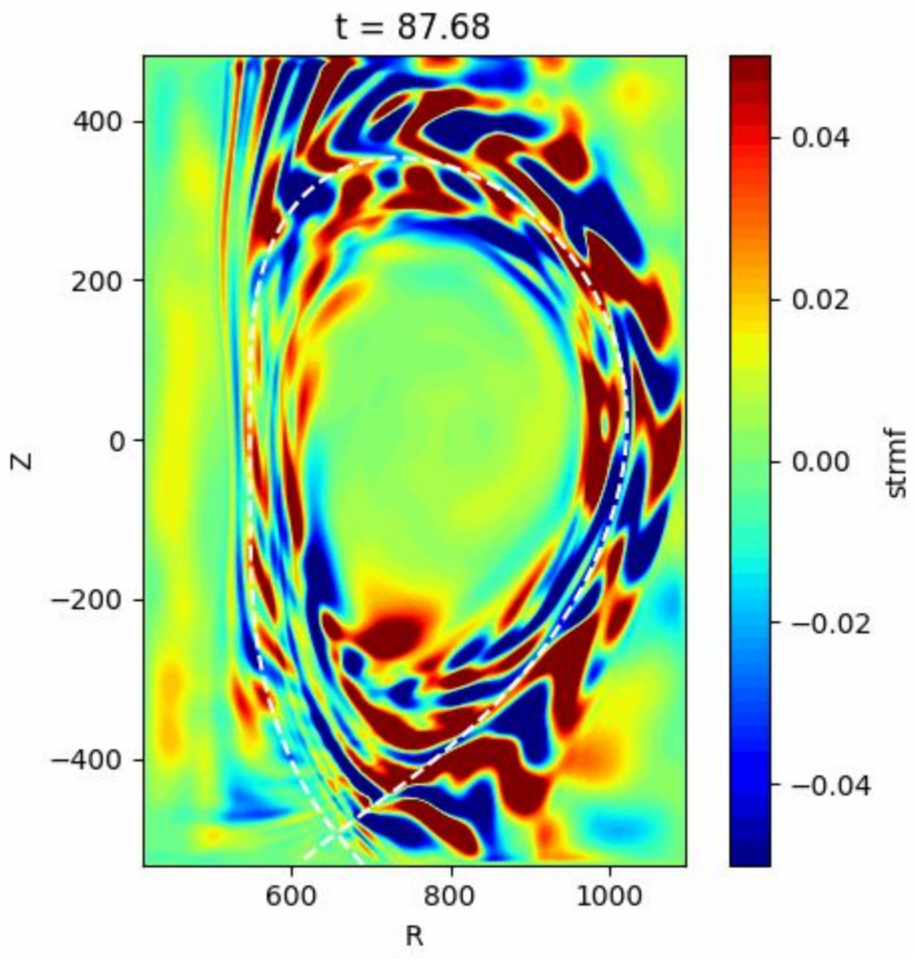
Complex 3D codes

3D, flux-driven turbulence codes, based on drift-reduced Braginskii model:

- **GBS**
  - Non-aligned grid, includes plasma core, kinetic neutrals, electromagnetic effects, ion dynamics.
- GRILLIX
  - Cylindrical grid, includes plasma core, electron-ion heat exchange, drift corrections at the magnetic presheath.
  - Evolves parallel component of the electromagnetic vector potential $A_{\parallel}$.
- TOKAM3X
  - Electron-ion heat exchange, drift corrections at the magnetic presheath.
- BOUT++
  - Framework for writing plasma simulations.
  - Any set of equations can be inserted and solved.
  - Can perform fluid or kinetic simulations.

GBS - FLUID TURBULENCE SIMULATION

- **G**lobal **B**raginskii **S**olver - first principle, 3D, flux-driven, global, **turbulence code** for plasma edge simulations based on Braginskii equations.
- Full plasma volume, Divertor geometry, electromagnetic effects, kinetic neutrals, ion temperature dynamics, **self-consistent turbulence evolution**.
- High computational requirements (~2000 cores, ~5-10 M CPU hours / simulation), however still lower compared to full kinetic models.



GBS - SIMULATION DOMAIN

Set of **Mag**netic boundary conditions
(Bohm Chodura boundary conditions)

$$v_{\|i} = \pm c_s \sqrt{1 + \frac{T_i}{T_e}},$$

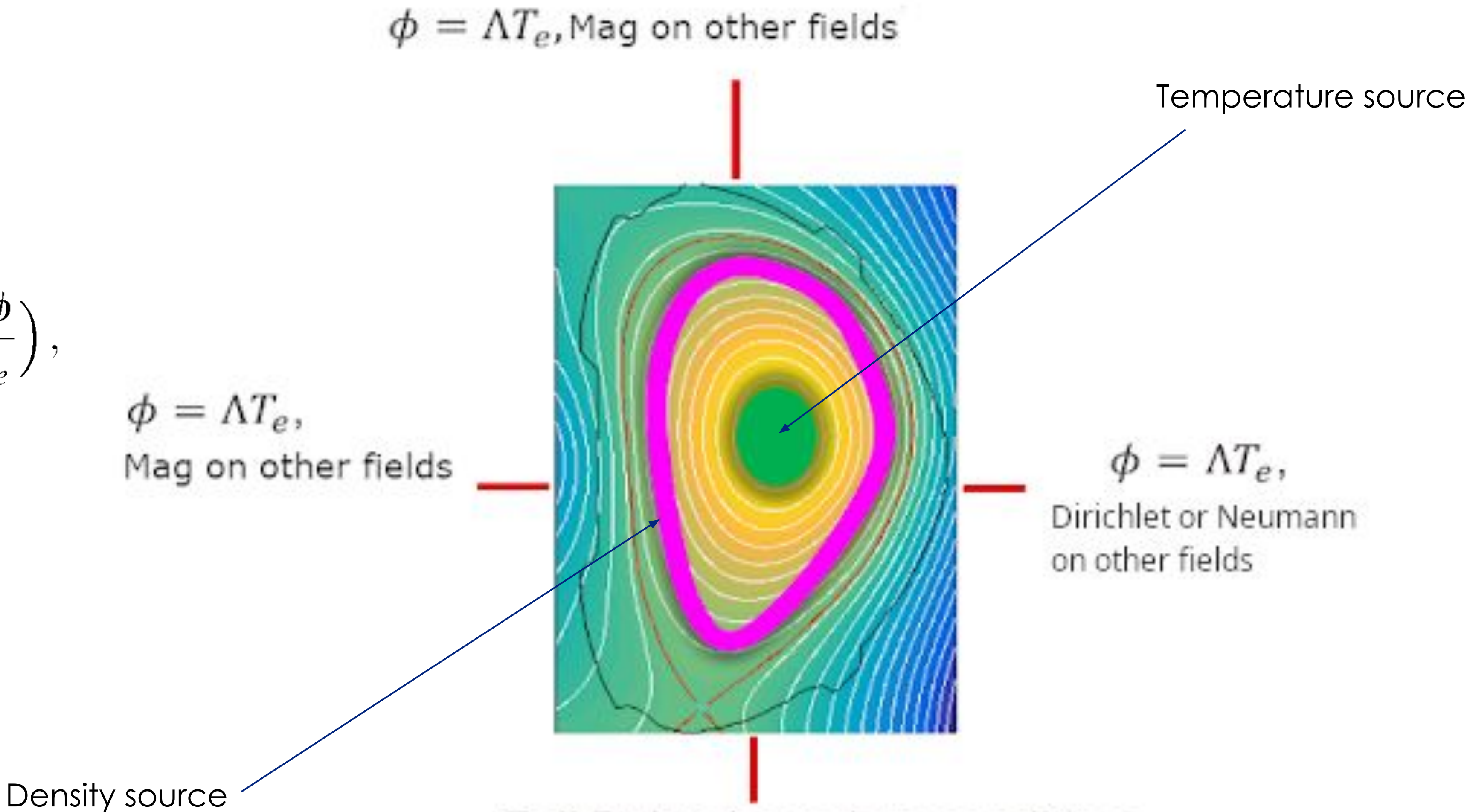
$$v_{\|e} = \pm c_s \sqrt{1 + \frac{T_i}{T_e}} \exp\left(\Lambda - \frac{e\phi}{T_e}\right),$$

$$\partial_s n = \mp \frac{n}{c_s \sqrt{1 + \frac{T_i}{T_e}}} \partial_s v_{\|i},$$

$$\partial_s T_e = \partial_s T_i = 0,$$

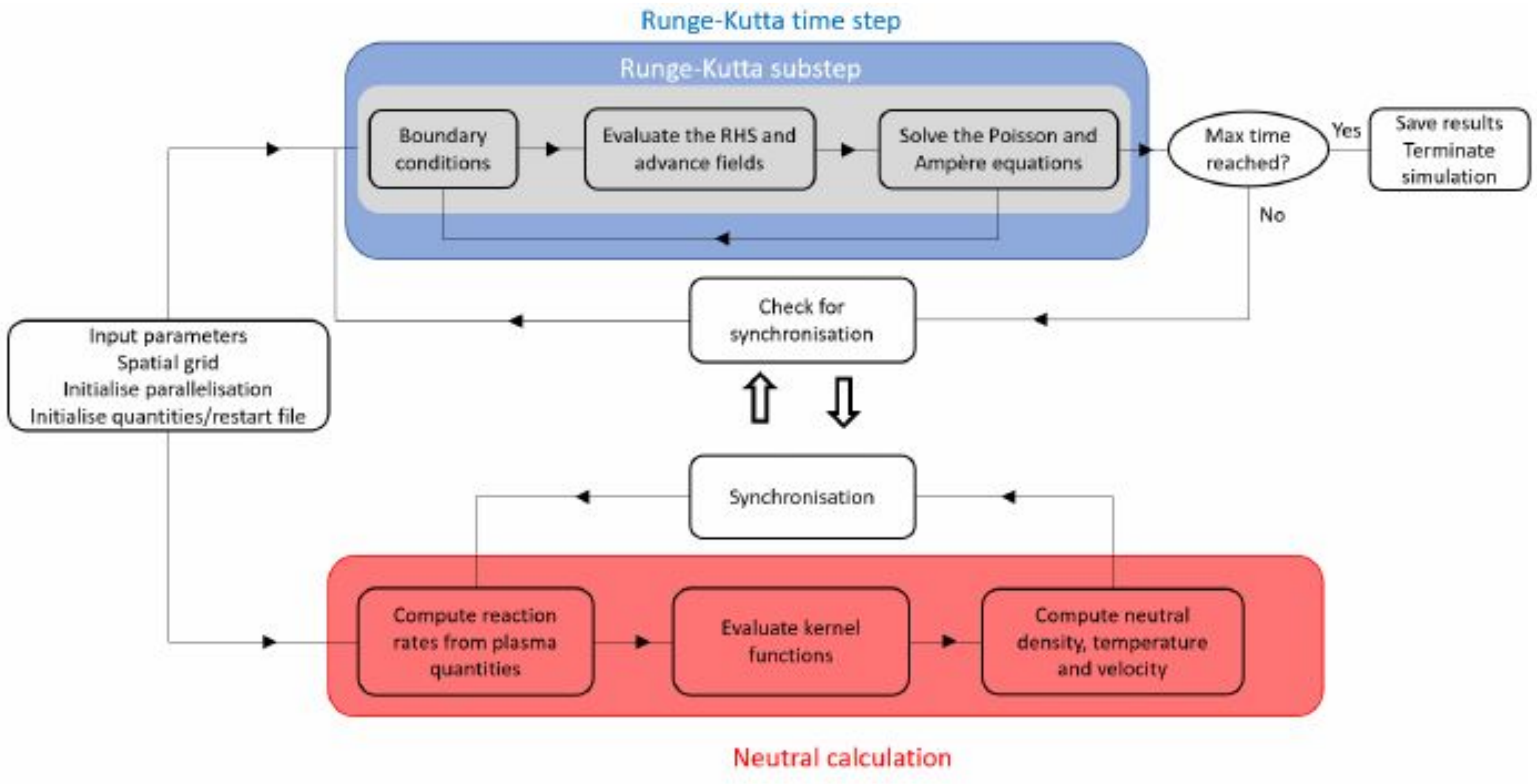
$$\Omega = \mp \frac{m_i n}{e} c_s \sqrt{1 + \frac{T_i}{T_e}} \partial^2_{ss} v_{\|i},$$

$$\partial_s \phi = \mp \frac{m_i c_s}{e \sqrt{1 + \frac{T_i}{T_e}}} \partial_s v_{\|i},$$

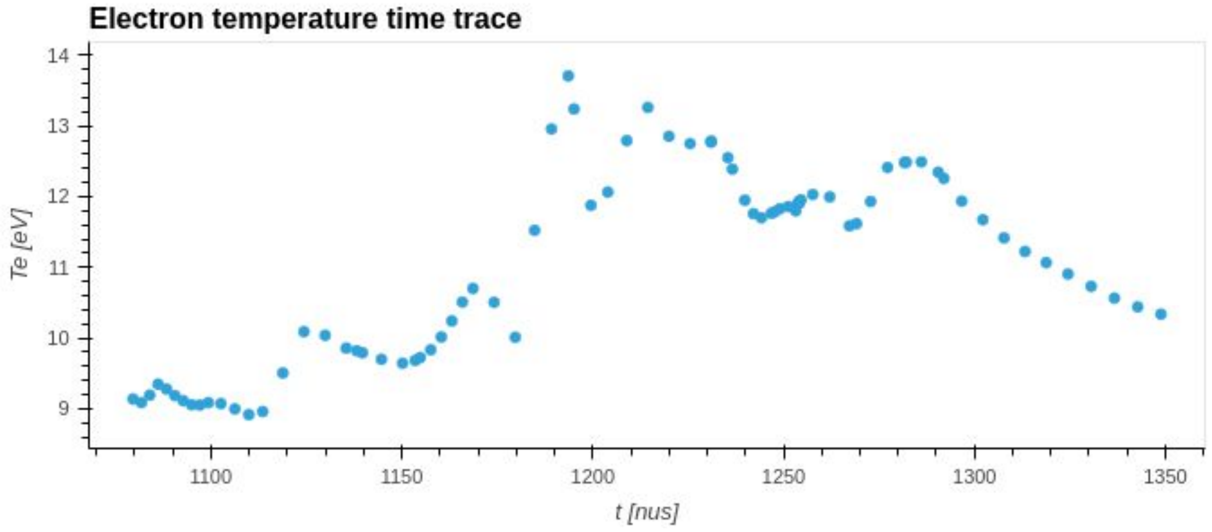


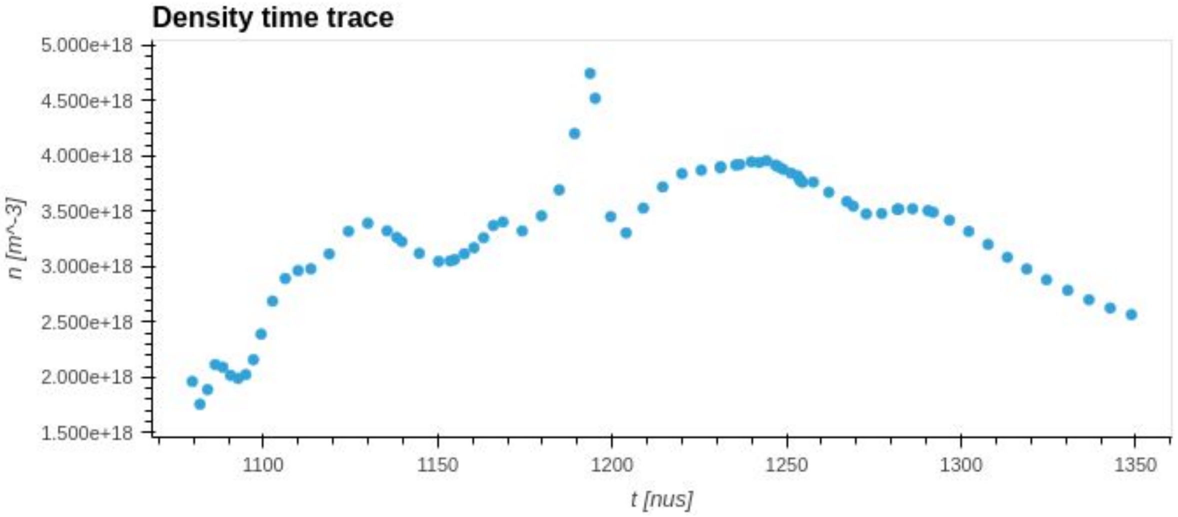
Full Bohm boundary conditions
(J. Loizu PoP 2012)

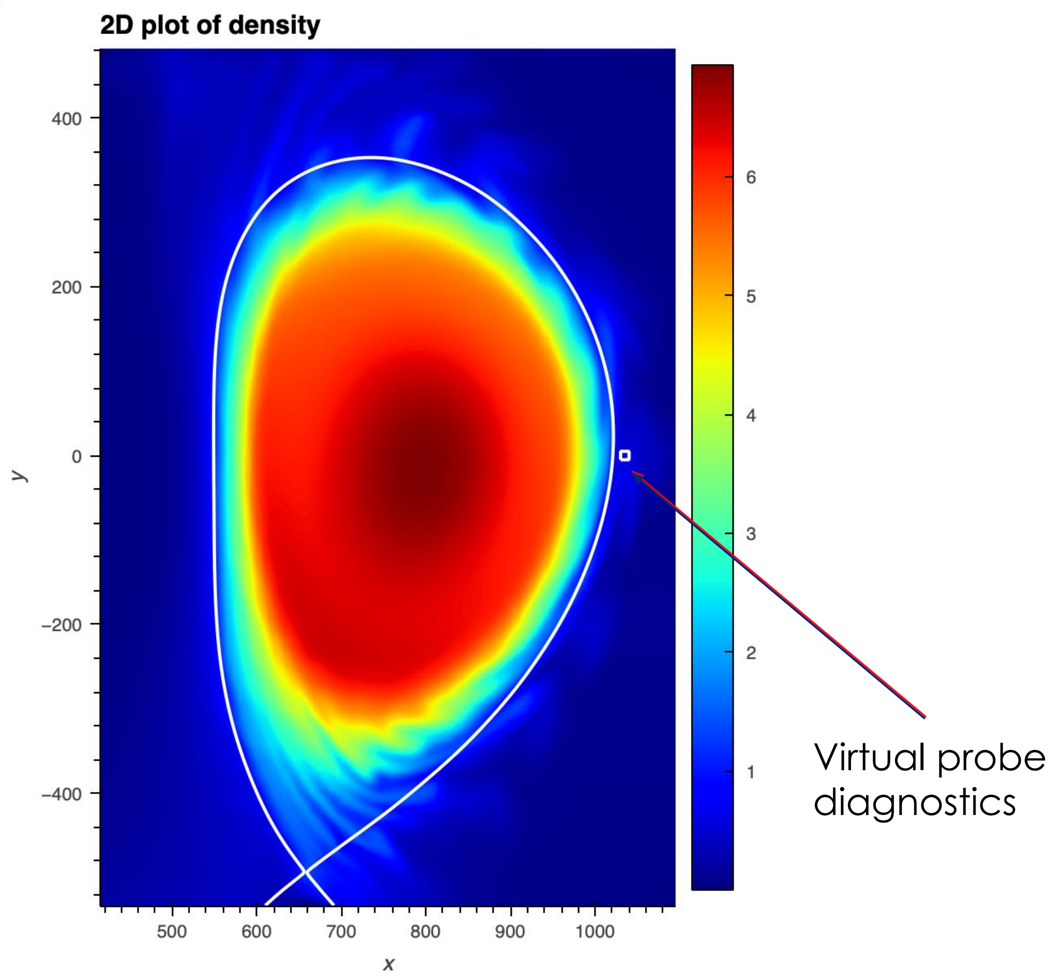
GBS - SIMULATION DOMAIN



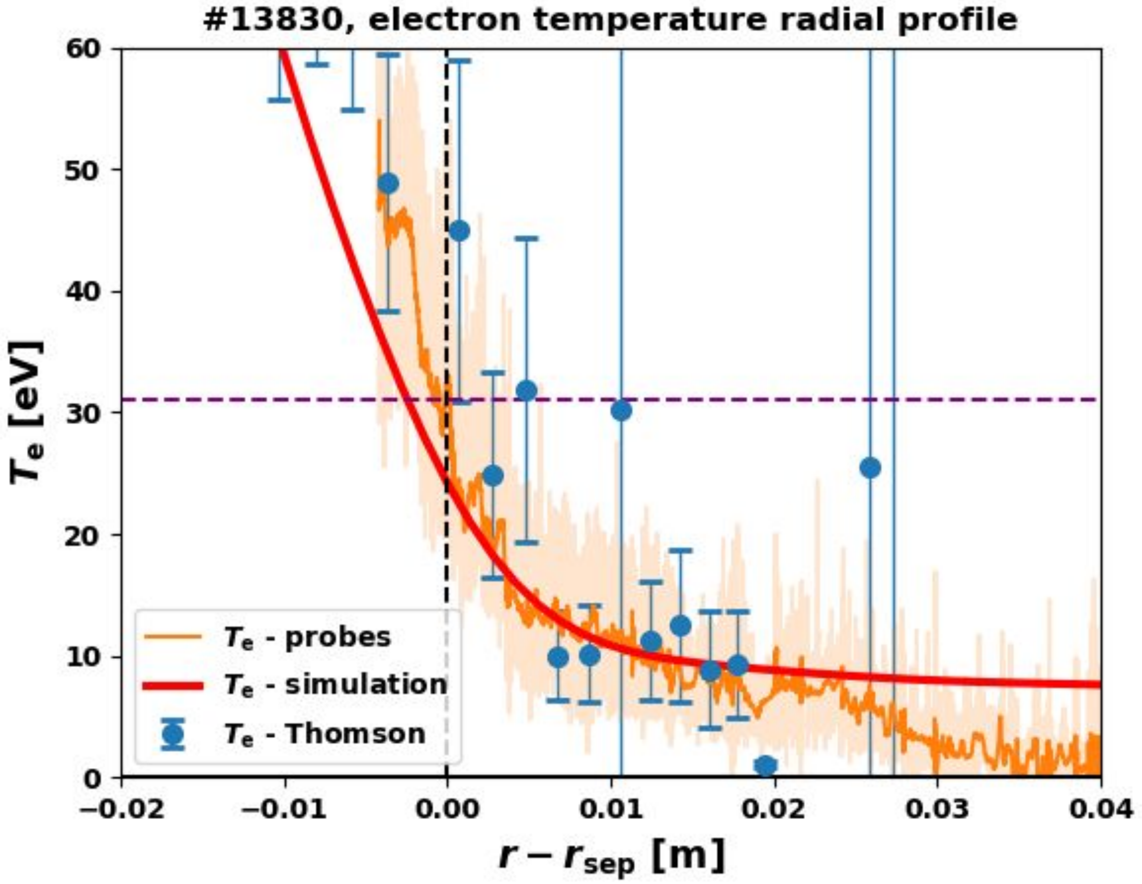
SIMULATION RESULTS EXAMPLES





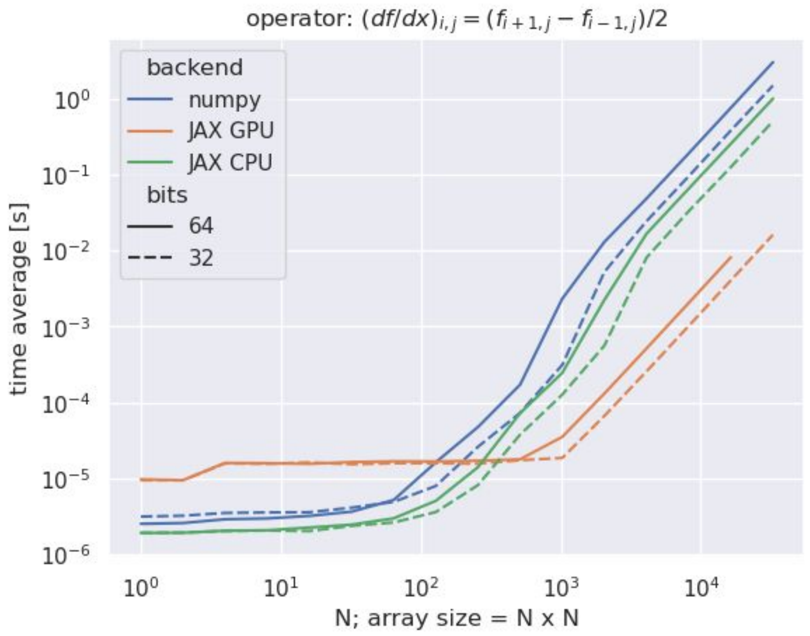


SIMULATION RESULTS EXAMPLES



Boosting simulation using GPU

- Python allows simple transformations of the code to GPU.
- All the fluid terms can be transformed.
- Central difference: $\frac{\partial f(t,x,y)}{\partial x} \approx \frac{f(t,x+\Delta h,y) - f(t,x-\Delta h,y)}{2\Delta h}$
- Compare of speeds using **numpy** vs **NUMBA** vs **JAX CPU** vs **JAX GPU**.
- Huge boost on GPU.
- Problem - limited memory on GPUs.



GPUs using Python

- **JAX** - module for calculations on GPU.
- Very easy to use and to convert code into JAX.
- Numpy-like syntax - **very easy to be used.**
- The code is parallelized along GPU automatically.
- JAX insluces **JIT** (just-in-time) compiler boosting the code.
- Uses machine learning for boosting the code even more.

many simple cores
parallel operations
fast computing
fast memory access

several fast cores
universal
sequential operations
a lot of memory

Conclusions

- Turbulence plays a key role in particle and heat transport.
- Understanding and controlling turbulence in plasma edge can lead to better confinement.
- Simulations can provide **interpretation** or prediction on turbulent transport.
- Fluid models offer **higher speeds** while still **encompassing important physics.**
  - Much faster compared to kinetic / gyrokinetic codes.
  - Does not include kinetic effects (Maxwellian distribution is assumed).
- Simple 2D model can be written very easily, providing still good results.
- Complex 3D models are more complicated, but almost the only way to perform full-size simulations.
- Consider using GPUs in your future works.

REFERENCES

1. *M.A. Zaman Electronics* **2022**, *11*(15), 2365
2. M. Giacomin et al J. Comput. Phys. 463 (2022) 111294 (The GBS code for the self-consistent simulation of plasma turbulence and kinetic neutral dynamics in the tokamak boundary)
3. M. Giacomin *et al* 2021 *Nucl. Fusion* **61** 076002 (Theory-based scaling laws of near and far scrape-off layer widths in single-null L-mode discharges)

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS