The GyroKineticDataBase (GKDB) project: a community tool for code benchmarks and building fast turbulent transport models

Y. Camenen, K. van de Plassche,
C. Bourdelle, J. Citrin, F. Imbeaux, D. van Vugt

EUROfusion Enabling Research support

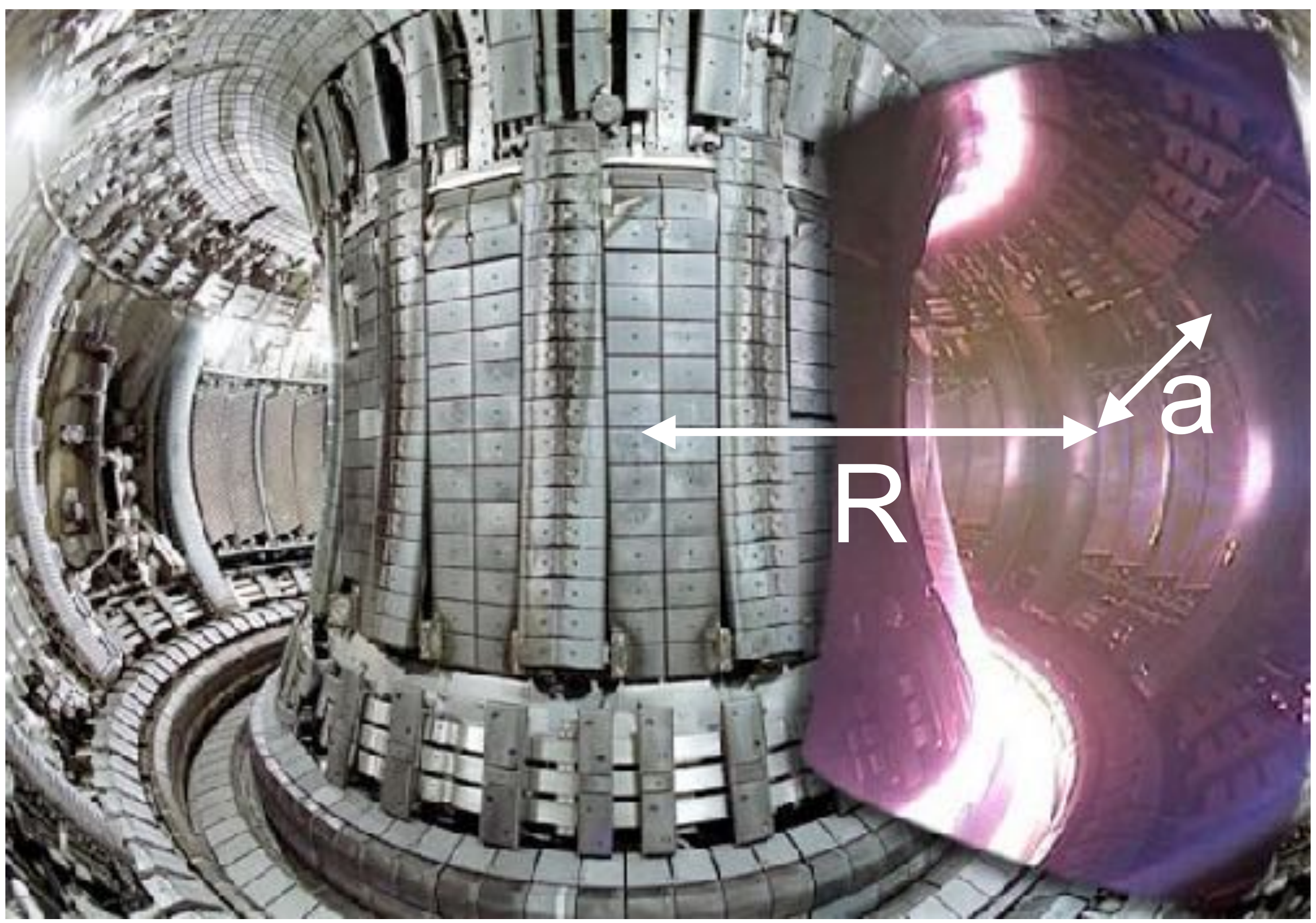
Context

Magnetic fusion research

Aim: energy production from fusion reactions (D+T→He+n) in magnetically confined plasmas with auxiliary heating

Difficulty: minimize transport and radiation losses

Tokamak: device with the best performance (so far)



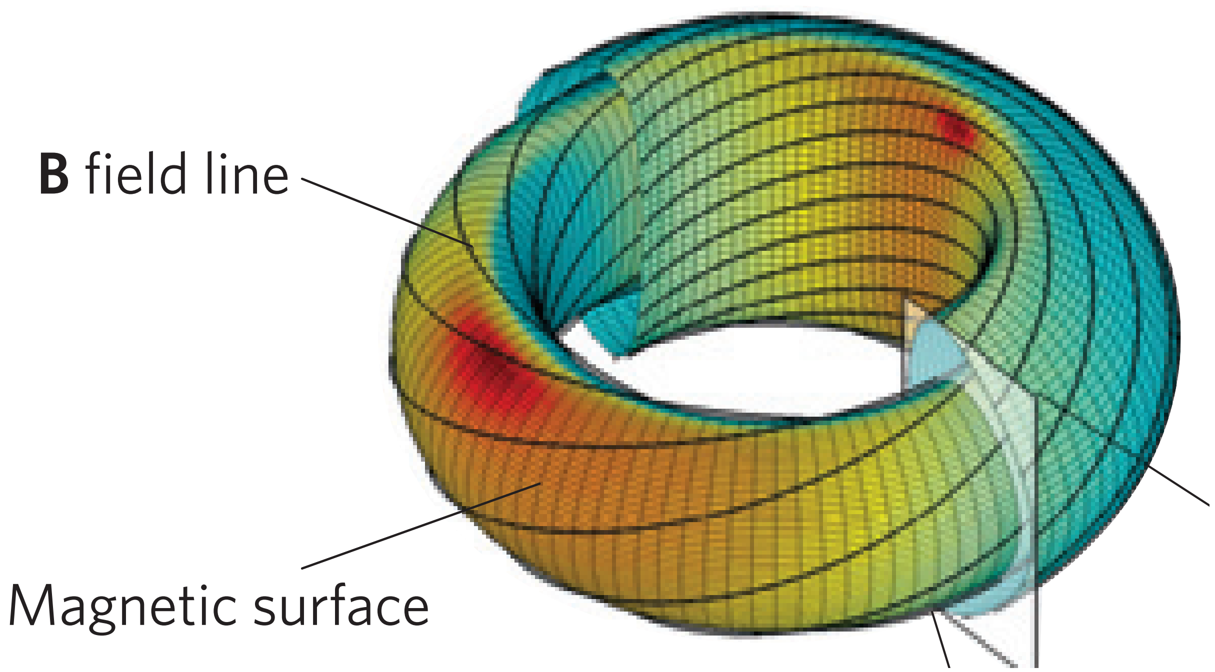
Joint European Torus (JET)

Tokamak

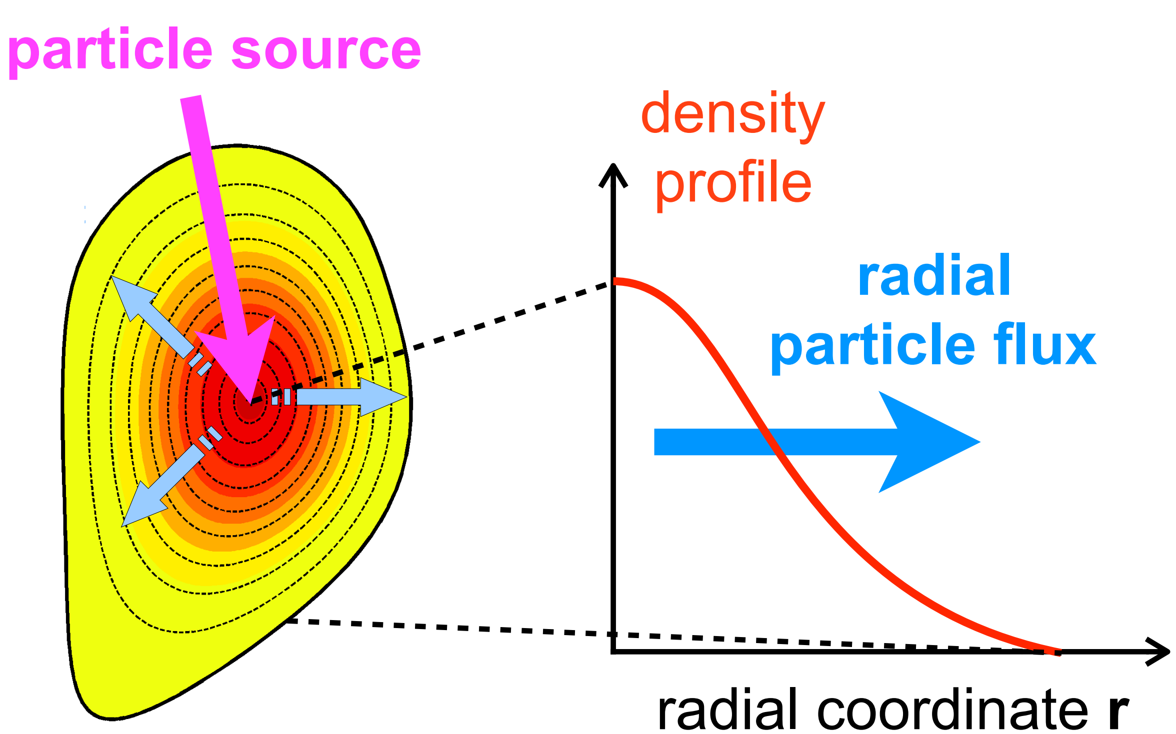
JET parameters

Torus	$R = 3\,\mathrm{m} \quad a = 1.25\,\mathrm{m}$
Magnetised	$B_T < 3.4\,\mathrm{T}$
	$I_p < 4\,\mathrm{MA}$
Injected power	$P_{\mathrm{inj}} < 38\,\mathrm{MW}$
Hot	$T = 1 - 20\,\mathrm{keV}$
Thin	$n = 1 - 10 \times 10^{19}\,\mathrm{m}^{-3}$

The issue of transport in tokamaks



- Nested magnetic flux surfaces
- Radial transport << parallel transport
- But radial transport is still finite...

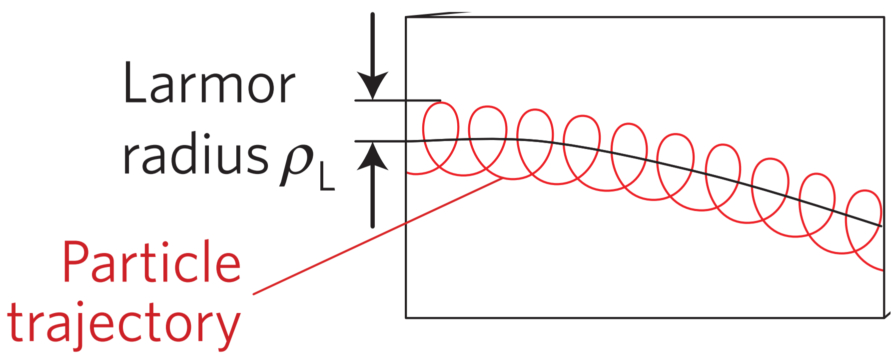


- Radial transport
  → link between sources and profiles
- Lower transport
  → better confinement
- What drives transport?
- How to predict and control transport?

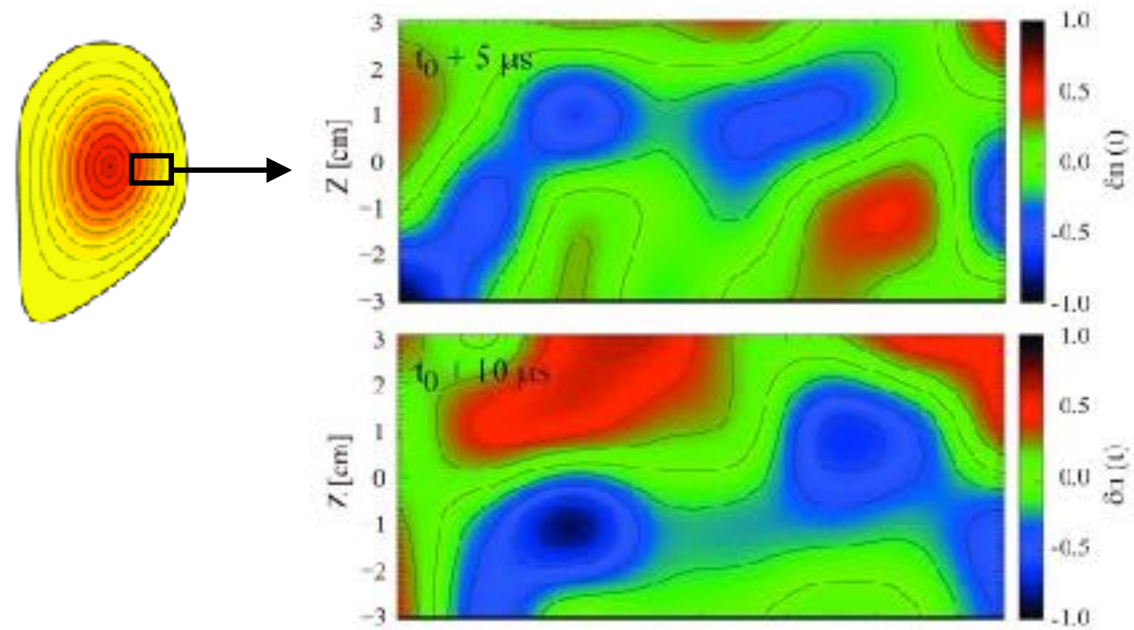
The gyrokinetic description of turbulent transport

- Kinetic description desirable (wave/particles interaction)
- 6D + fast and short scales → very costly!!
- Exploit scale separation between gyro-motion and plasma fluctuations

Gyro-motion



Density fluctuations



Larmor radius	Cyclotron frequency
$\rho_e = 0.02 - 0.1\,\mathrm{mm}$	$f_{c,e} = 10^9\ \mathrm{Hz}$
$\rho_i = 1 - 6\,\mathrm{mm}$	$f_{c,i} = 10^7\ \mathrm{Hz}$

$$L_\perp \sim 5 - 20\rho_i$$
$$f \sim 10^4 - 10^5\ \mathrm{Hz}$$

- **Drop the gyro-phase: 6D → 5D**
- **Gyrokinetic Vlasov-Maxwell system**

Model hierarchy

- Good but numerically too expensive, simplifying further:
  - Frozen magnetic equilibrium $\partial \mathbf{B}_0/\partial t = 0$ ~10^7-10^8 CPU hours
    Global full-f codes (e.g. GYSELA, GT5D, ORB5)
  - Frozen background (δf approximation) $\partial F_0/\partial t = 0$ with $F = F_0 + \delta f$
    Global delta-f codes (e.g. GYRO, GENE, GKW)
  - Local approximation $F_0(r) = F_0(r_0)$ and $\nabla F_0(r) = \nabla F_0(r_0)$
    Local delta-f codes (e.g. GYRO, GENE, GKW) ~10^4-10^5 CPU hours
  - Quasi-linear approximation
    Local delta-f linear codes ~10^2-10^3 CPU hours
    - Cross-phase assumed to be given by the linear response
    - Saturation amplitude is modelled
  - Quasi-linear + additional simplifications ~0.01 CPU hours
    Gyro-fluid (e.g. TGLF)
    Gyro-kinetic with fluid eigenfunctions (QuaLiKiz)

A gyrokinetic database, what for?

- Wishlist:
  - Store inputs/outputs of linear gyrokinetic runs
  - Possibility to store/access millions of entries (SQL requests)
  - Open access
- Purposes of the database (non exhaustive):
  - Repository for data presented in publications and conferences
  - References for benchmarks
  - Instantaneous linear stability calculations (from fits of the database content)
- Opens the route to ultrafast 1st principle QL transport models
  - Proof of principle demonstrated for a 5D database using neural network fits *[Citrin NF2015]*
  - Applications: real-time control, fast integrated modeling, uncertainty propagation, etc..

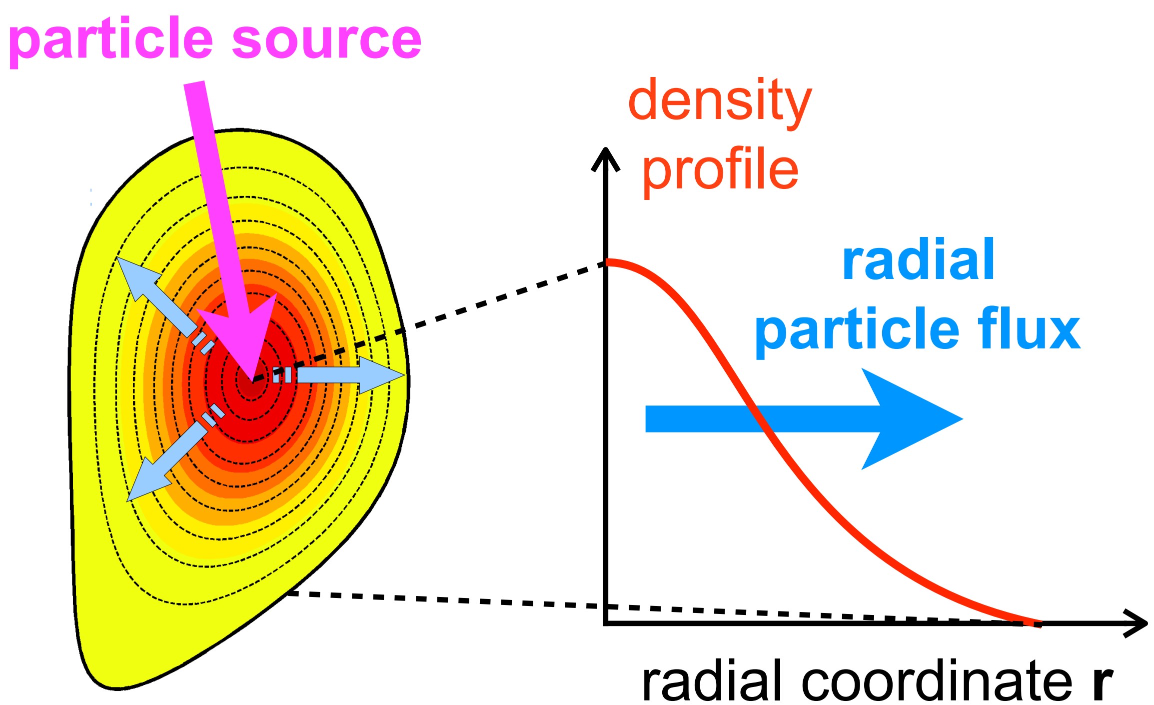
The GKDB project

- **Project** hosted on Gitlab: https://gitlab.com/gkdb/gkdb
  - Source for the SQL/Python interface
  - Routines (matlab, python,…) to convert data from various GK codes to the GKDB format (unified normalisations)
  - Documentation (wiki): GKDB format, coupling to GK codes, how to download/upload data, etc…

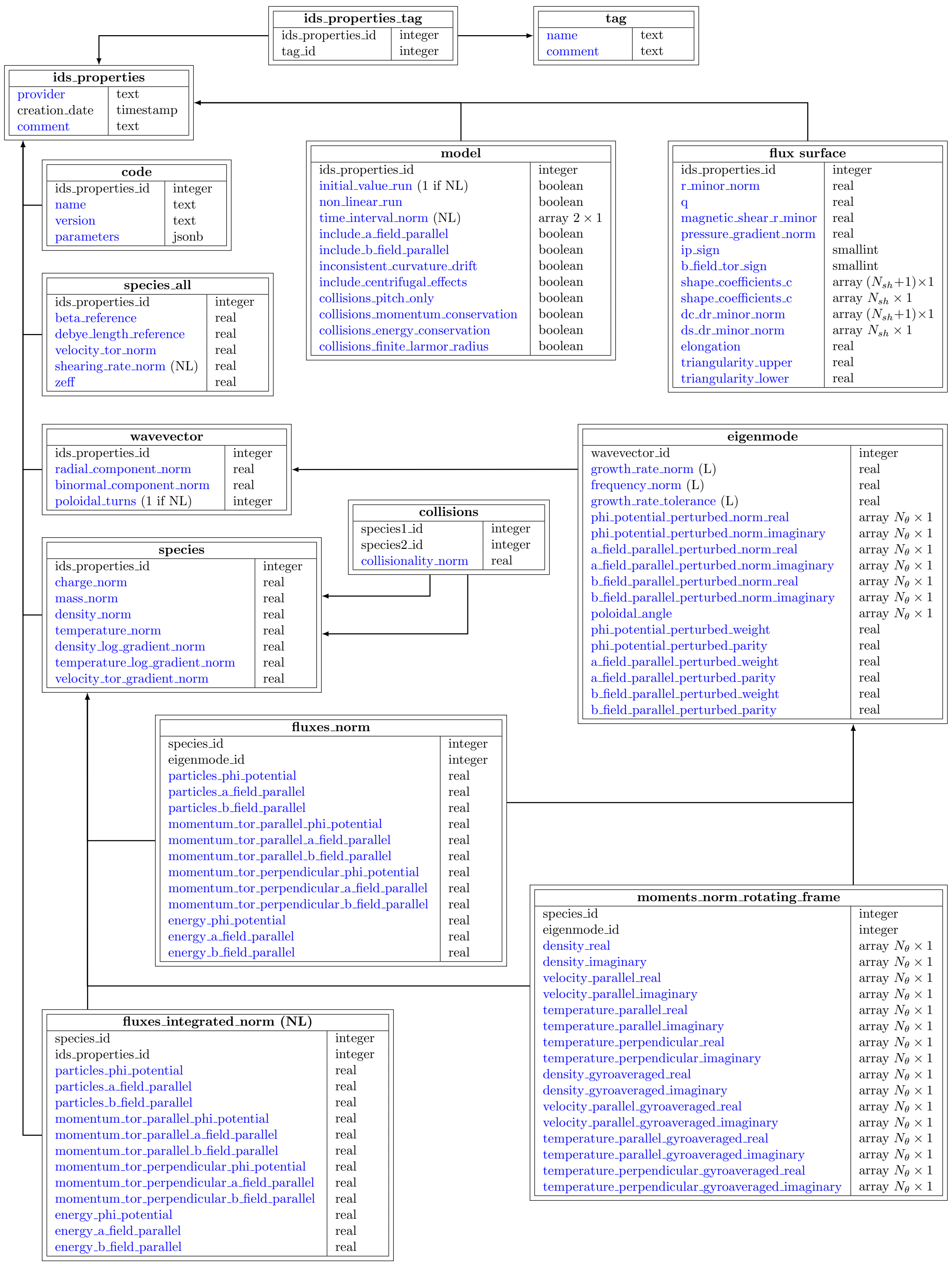
- **Database** on gkdb.org (hosted by DigitalOcean at the moment)
- User access managed by LDAP
  - Account created on request
  - Read access for everybody
  - Write access once validated reference cases are provided

Database content

- Flux-tube δf simulations (linear and non-linear runs)
  - Inputs
    - Mag. equilibrium
    - Species
    - Wavevectors
    - Model (collisions, EM effects,…)
  - Outputs
    - Eigenvalues
    - Eigenfunctions
    - Fluxes
  - Metadata
    - Code name and version
    - Code specific parameters
    - Date
    - Contributor
    - Comments

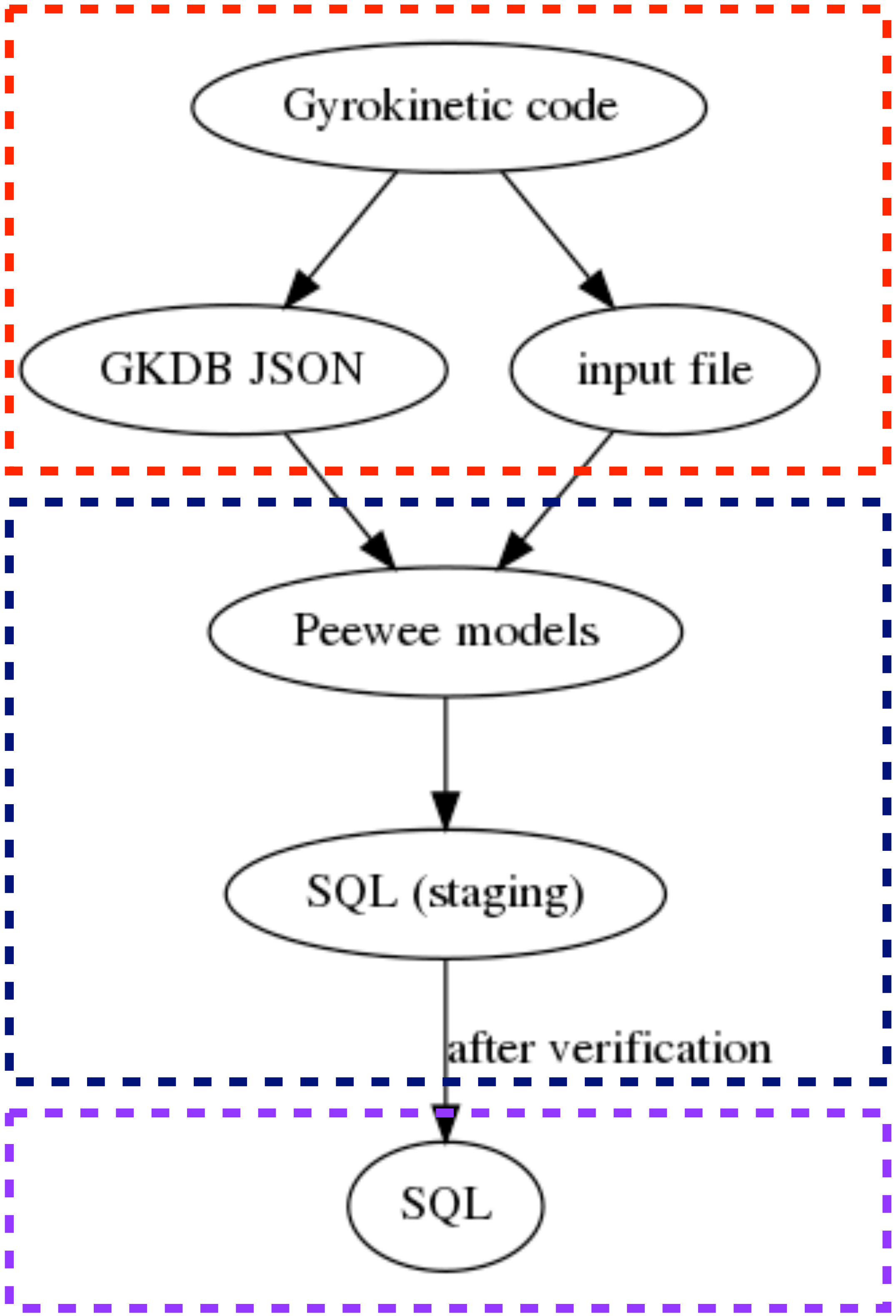


Database structure



- Relational database (SQL)
- Possibility to “tag” entries

Uploading entries to the database



- User to provide:
  - a JSON file with a GKDB entry
  - including code specific inputs (grids, dissipation, etc…)
- Python scripts:
  - Convert the JSON file to SQL
  - Compute derived quantities for queries
  - Check mandatory fields, ranges, dimensions (implemented)
  - Check entry sanity (numerical stability, quasineutrality...)
  - Database server

Querying the database

- GKDB web browser: http://database.gkdb.org
- Direct SQL queries can be performed from Python, Matlab and IDL
- Possible to run Python (and maybe Matlab/Octave) directly on the GKDB server: http://jupyter.gkdb.org
- Small subsets could also be exported via zipped JSON files

Present status and next steps

- Database online at gkdb.org,
- Project repository at https://gitlab.com/gkdb/gkdb
- Database format and conventions documented
- GKDB presently interfaced with gyrokinetic code GKW

- Near future actions:
  - Interface with gyrokinetic code GENE (in progress), and hopefully other codes
  - Use the reference cases to validate the interface with GKDB
  - Finalize Python scripts to check the entries integrity
  - Start populating the database and test the pipeline

Open issues

- **Licensing**
  - Open Database License from Open Data Commons?
- **Storage**
  - Where? Maintenance?
  - Not an issue at present, but needs to be anticipated
  - Database scale:
    - >10 millions of entries
    - 0.1 - 10 Mo per entry → database: ~10To
    - <100 users