Integrated modeling with TRANSP and OMFIT

Orso Meneghini

TRANSP user group

PPPL, March 23rd 2015

Outline

Outline

One Modeling Framework for Integrated Tasks

OMFIT is an integrated modeling framework that:

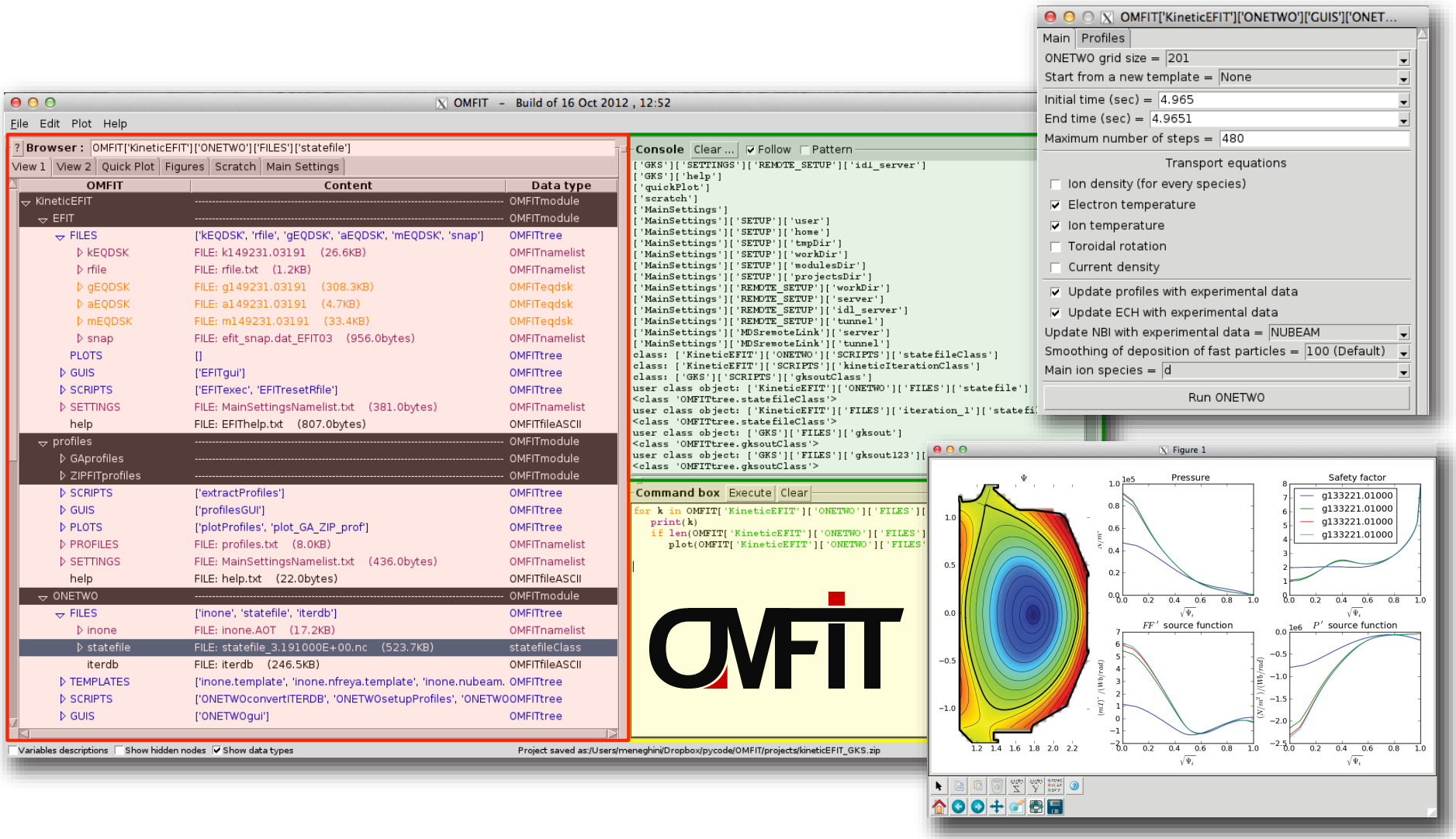
1. **Enables code to interact in complicated workflows**
   Managing complexity of data exchange and codes execution

2. **Facilitates all stages of the modeling cycle**

   Workflow execution, but also: simulation setup, code development, debugging, comparing to other codes or experiments, data management, post-processing, plotting, ...

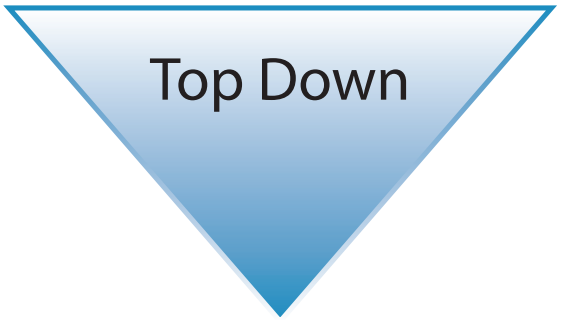
How? OMFIT is a hybrid between a workflow manager and an integrated development environment

O. Meneghini and L. Lao, Plasma and Fusion Research, **8** 2403009 (2013)



OMFIT adopts a 'bottom-up' development approach

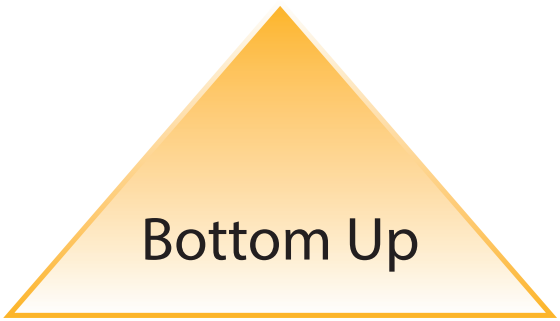
Assume design requirements are well known and immutable



Structured development process

- *Waterfall* methodology
  - Big design up-front
  - Delivered in full
  - Users test at the end
- Hard to change
- High investment risk

Adaptive development process



Assume design requirements are not well known or evolving

- *Agile* methodology
  - Start simple
  - Incremental deliveries
  - Embed users feedback
- Easy to change
- Lower investment risk

The centerpiece of OMFIT is its flexible data structure

The **OMFIT-tree** is a hierarchical, self-descriptive data structure that enables data exchange among different codes

- Collect data independently of its origin and type
- Objects' content appear in their subtree
- No a-priori decision of what is stored and how
- Codes exchange data by referring to quantities in the tree

Same functionality as the "statefile" structures of other frameworks...

...but **free-form !**

Like *MDS+* or *file-system* on your own laptop: the data is stored however it is deemed more logical to accomplish a certain task

With N codes, it's an N^2 problem! How is it possible to make all these codes talk to one another?

- By reading/writing a few (10+) standard scientific data formats OMFIT can interact with many different codes
- Often codes need to exchange only small amount of data
- Exploit existing integration efforts:
  - Many codes already accept each others' files
  - Conversion utilities that are already available
- Real-world applications do not require coupling N^2 codes!

Several advantages:

- No need to modify codes and their I/O
  - No burden on developers of individual codes
  - Effort done by users interested in integrating
  - Compatible with distributed community developments
- Does not exclude use of data structures from other frameworks
  - Skips all-together arguments about which data structure to use
  - Survival of the framework does not depend on widespread adoption of its own data structure by the community

Other important characteristics of the OMFIT framework

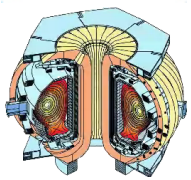
Lightweight, pure-Python framework is easy to install, maintain, and expand

Code execution is natively **remote**, and can be **parallel**

Python scripting and **component based approach** allow building of powerful and complex workflows

Graphical user interfaces ease execution of each component and their interaction

Power users retain full control of their codes I/O and execution



Integrated with experimental databases for data analysis, generation of code inputs, and validation

Collaborative environment encourages users contributions, and community revision

OMFIT is an integral element of the *Advanced Tokamak Modeling* (AToM) SciDAC project

Not a new physics code, but rather a concept:

"To enhance and extend present modeling capabilities, by supporting, leveraging, and integrating existing research"

GENERAL ATOMICS

OAK RIDGE National Laboratory

UCSD

Lawrence Livermore National Laboratory

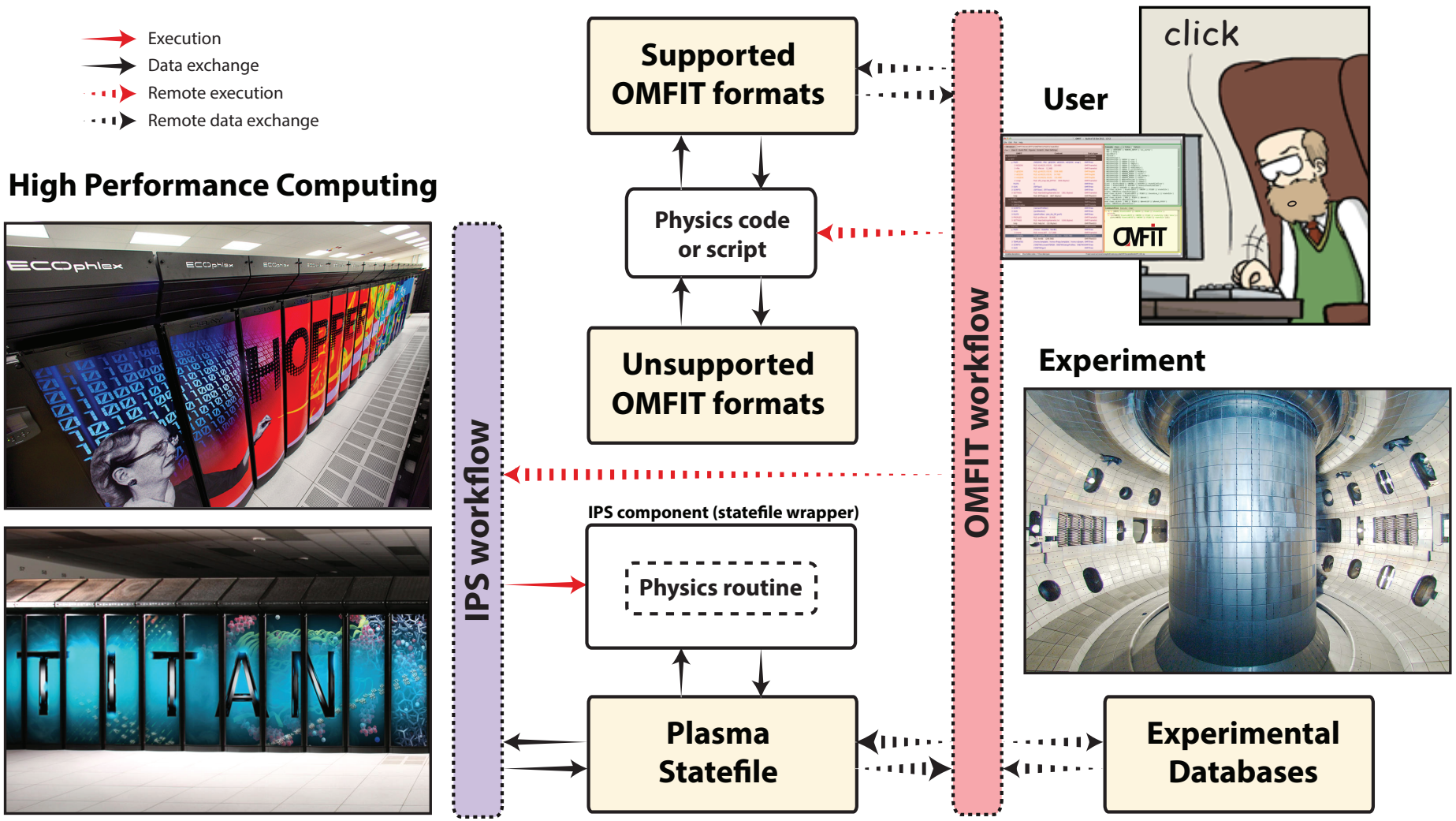
FASTMATH

SUPER

1. Bridge gap between *experimental data analysis* and *high performance simulation* communities
2. Performance engineering of critical HPC components
3. Study coupled core, pedestal, and scrape-off layer physics eg.: TGYRO $\leftrightarrow$ EPED $\leftrightarrow$ COGENT

As part of AToM, couple *Integrated Plasma Simulator* (IPS) and OMFIT and exploit their synergy

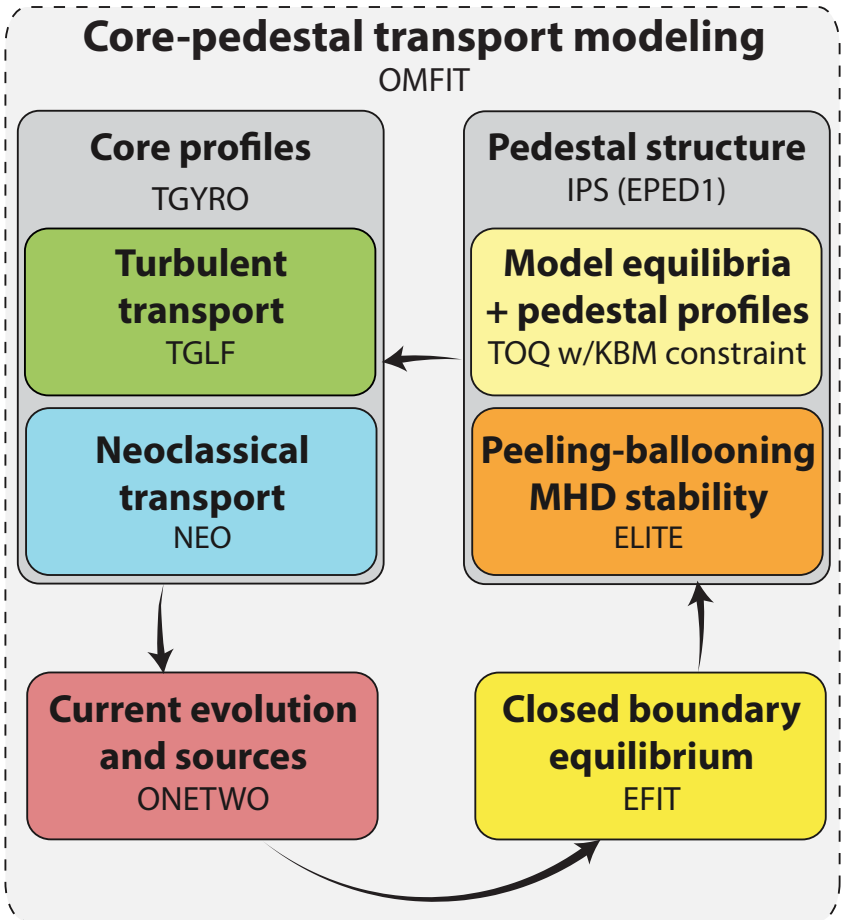
- IPS provides powerful HPC scheduling capabilities
- OMFIT provides interface to users, experiments and codes

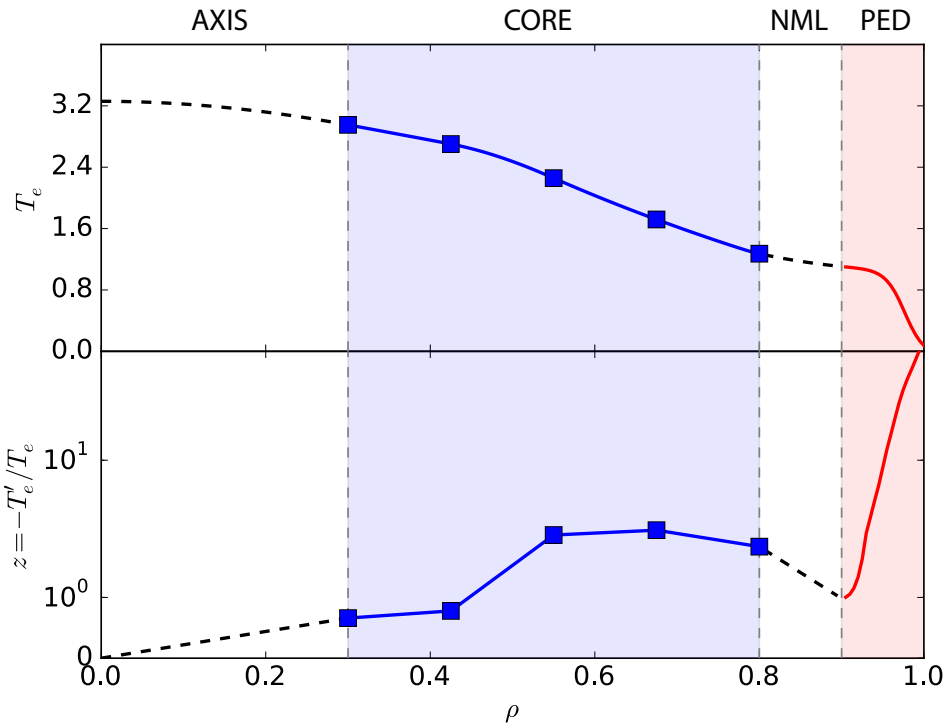


Outline

AToM project enabled first-principles self-consistent equilibrium, core transport, and edge stability studies

Separation of MHD, transport, and current diffusion timescales





Four radial zones:

PED Pedestal structure model (PB+KBM)

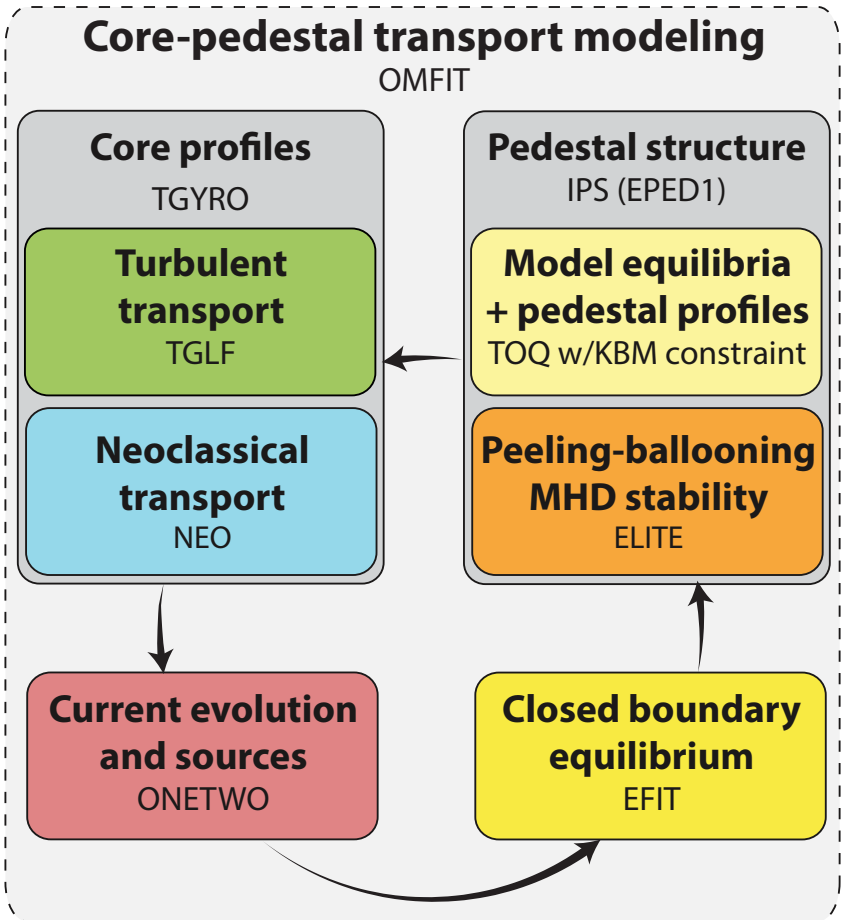
NML Linear scale-length interpolation

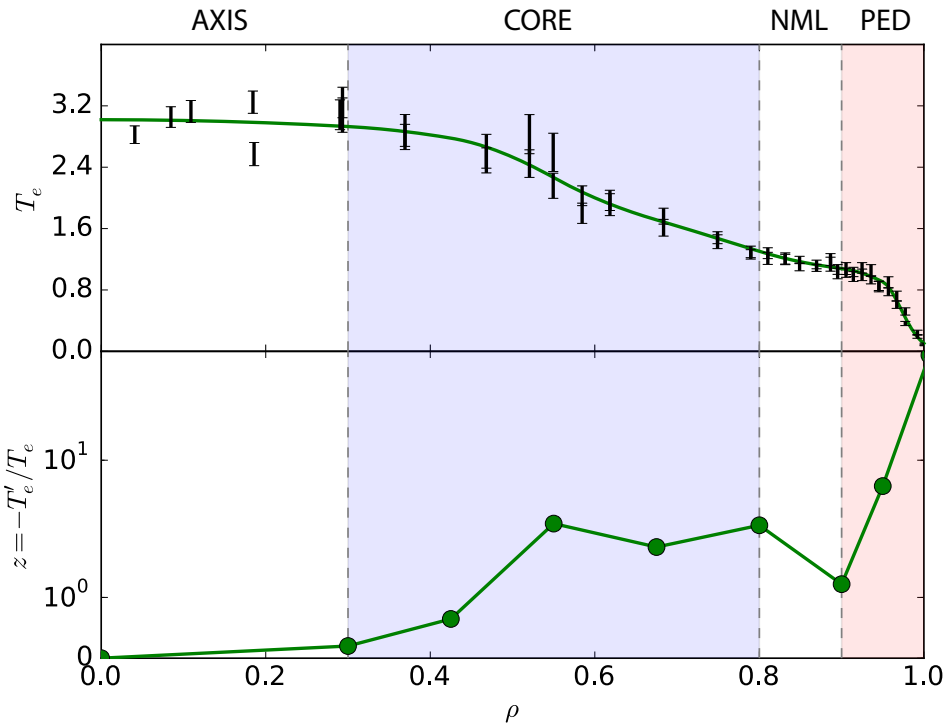
CORE Reduced gyrokinetic model

AXIS Linear scale-length to zero

AToM project enabled first-principles self-consistent equilibrium, core transport, and edge stability studies

Separation of MHD, transport, and current diffusion timescales



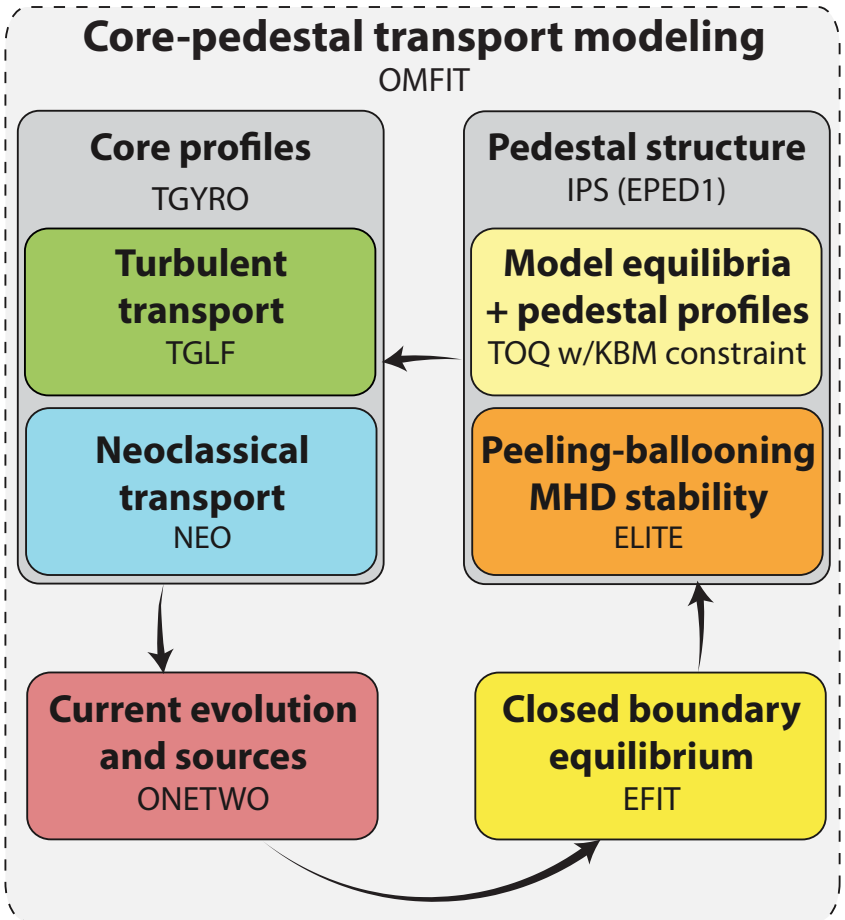


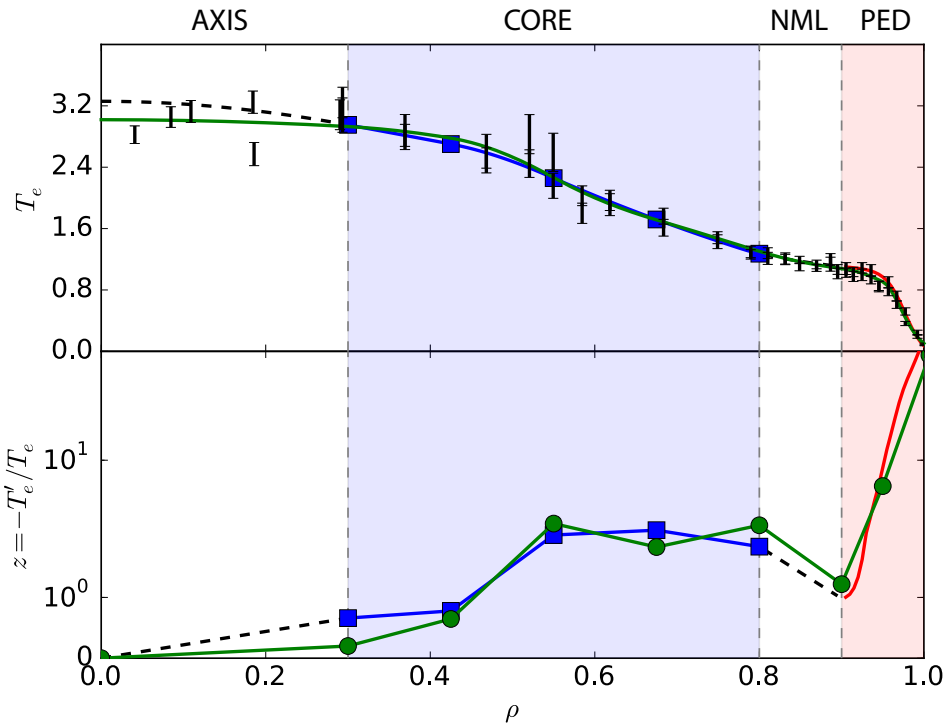
Linear scale-length fitting of experimental data supports choices of:

- Radial zone locations
- Using few TGLF calculation points

AToM project enabled first-principles self-consistent equilibrium, core transport, and edge stability studies

Separation of MHD, transport, and current diffusion timescales



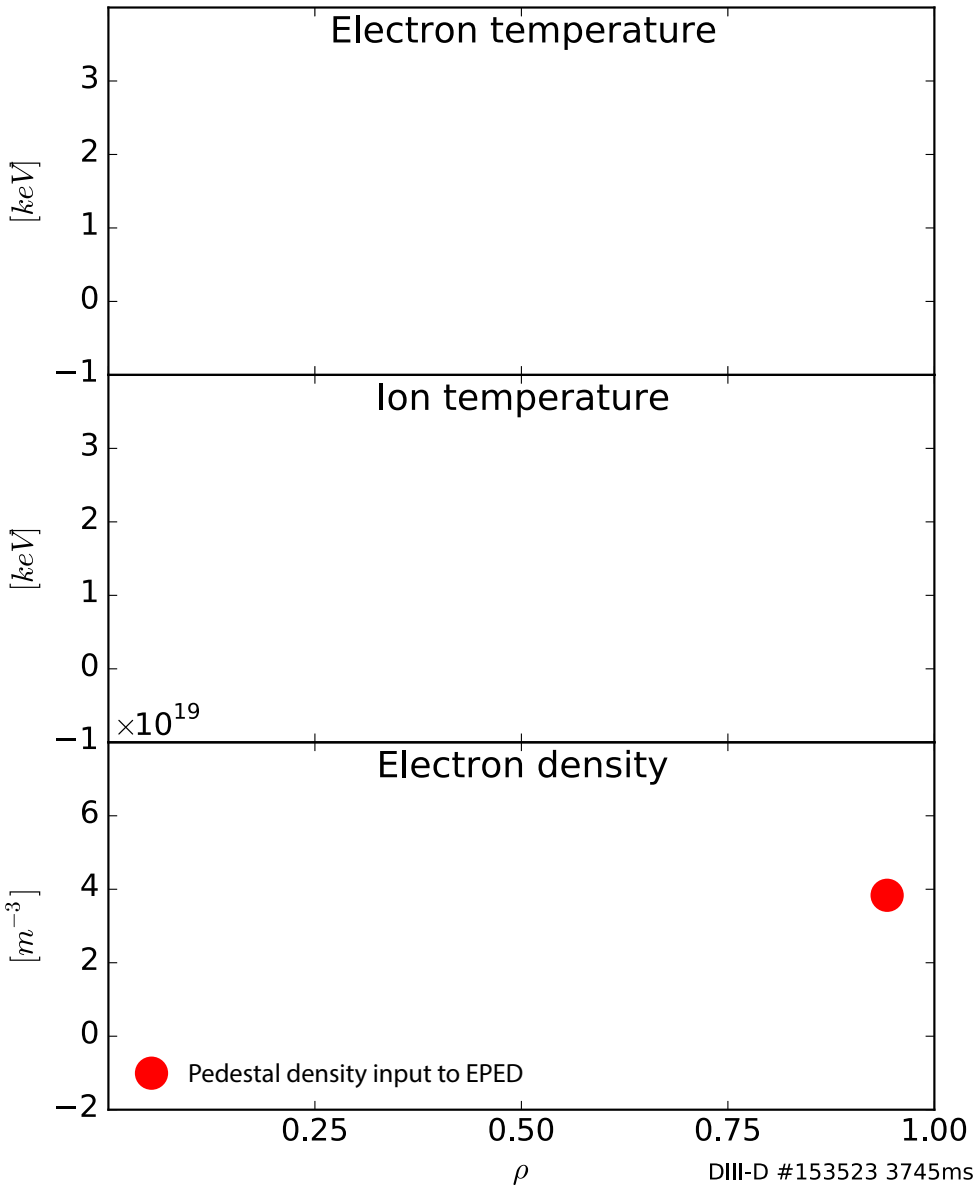


Linear scale-length fitting of experimental data supports choices of:

- Radial zone locations
- Using few TGLF calculation points

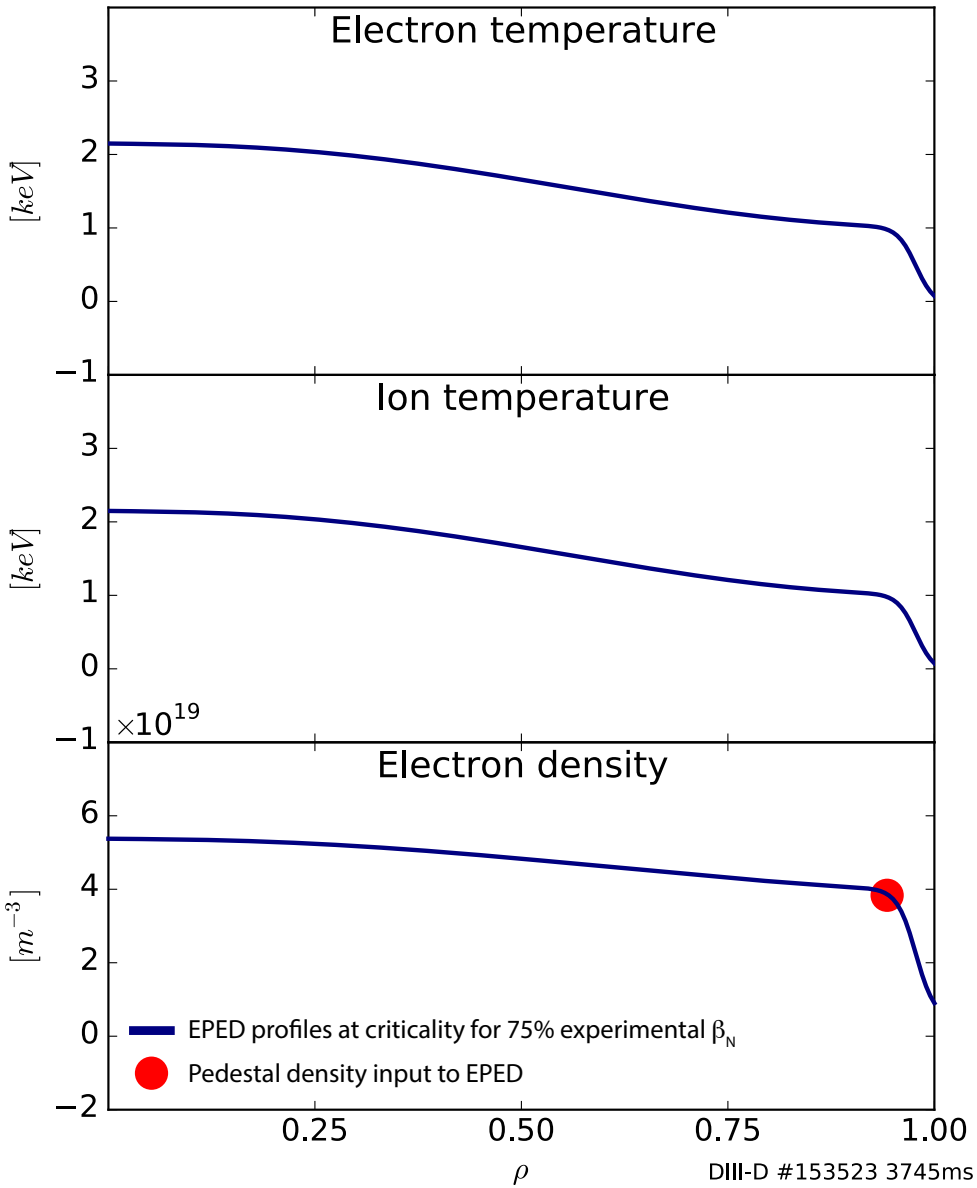
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N



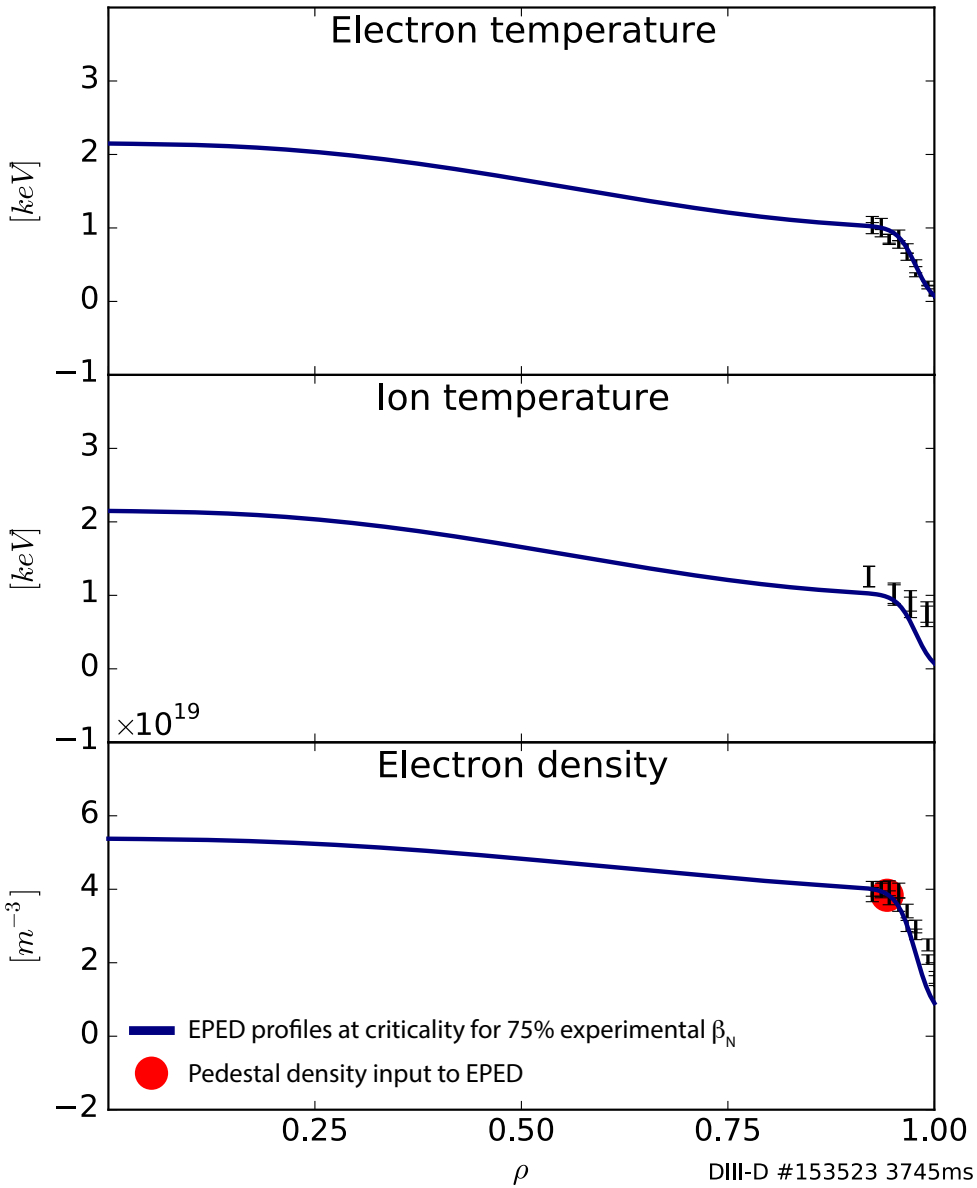
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N



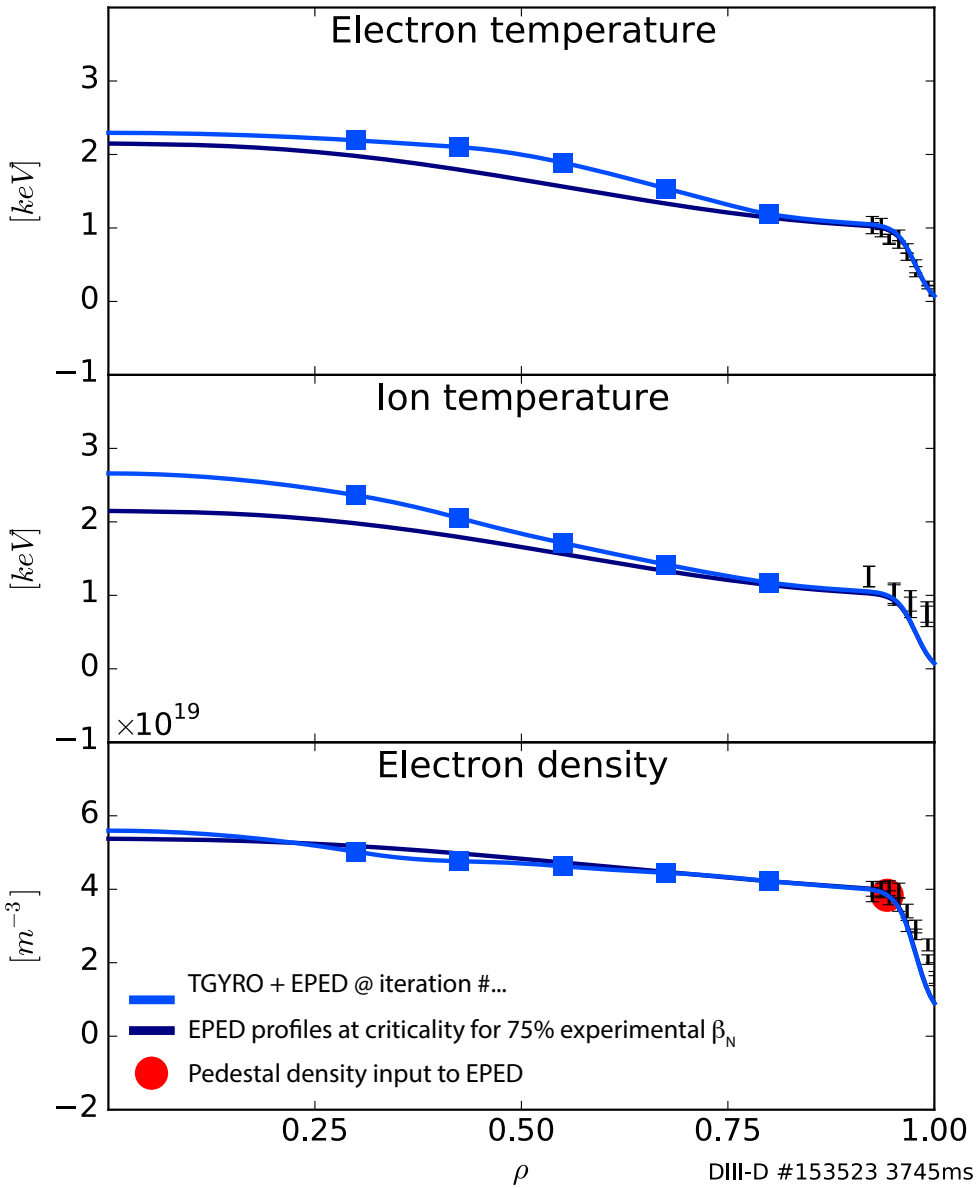
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N



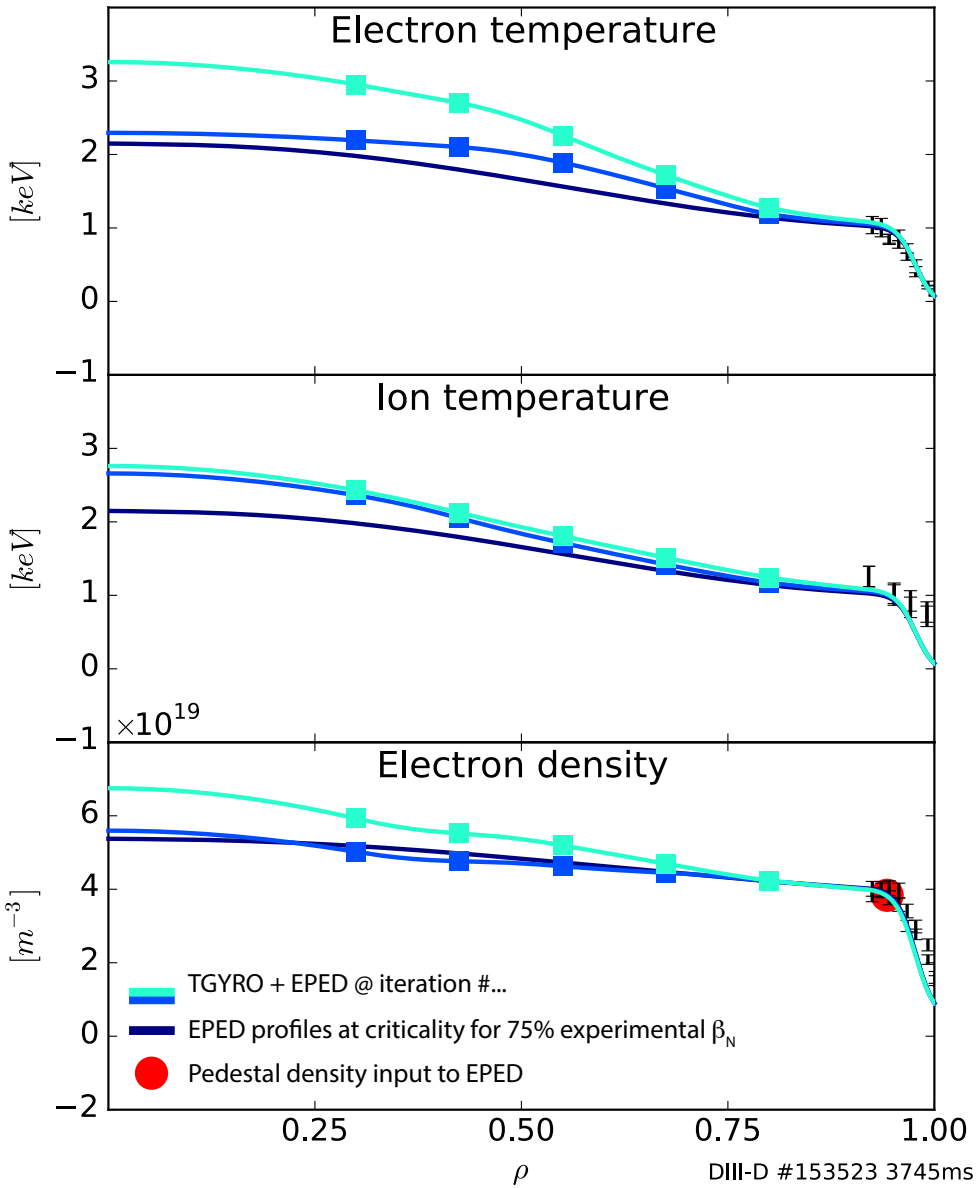
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N
- TGYRO to predict profiles and β_N. Then iterate...



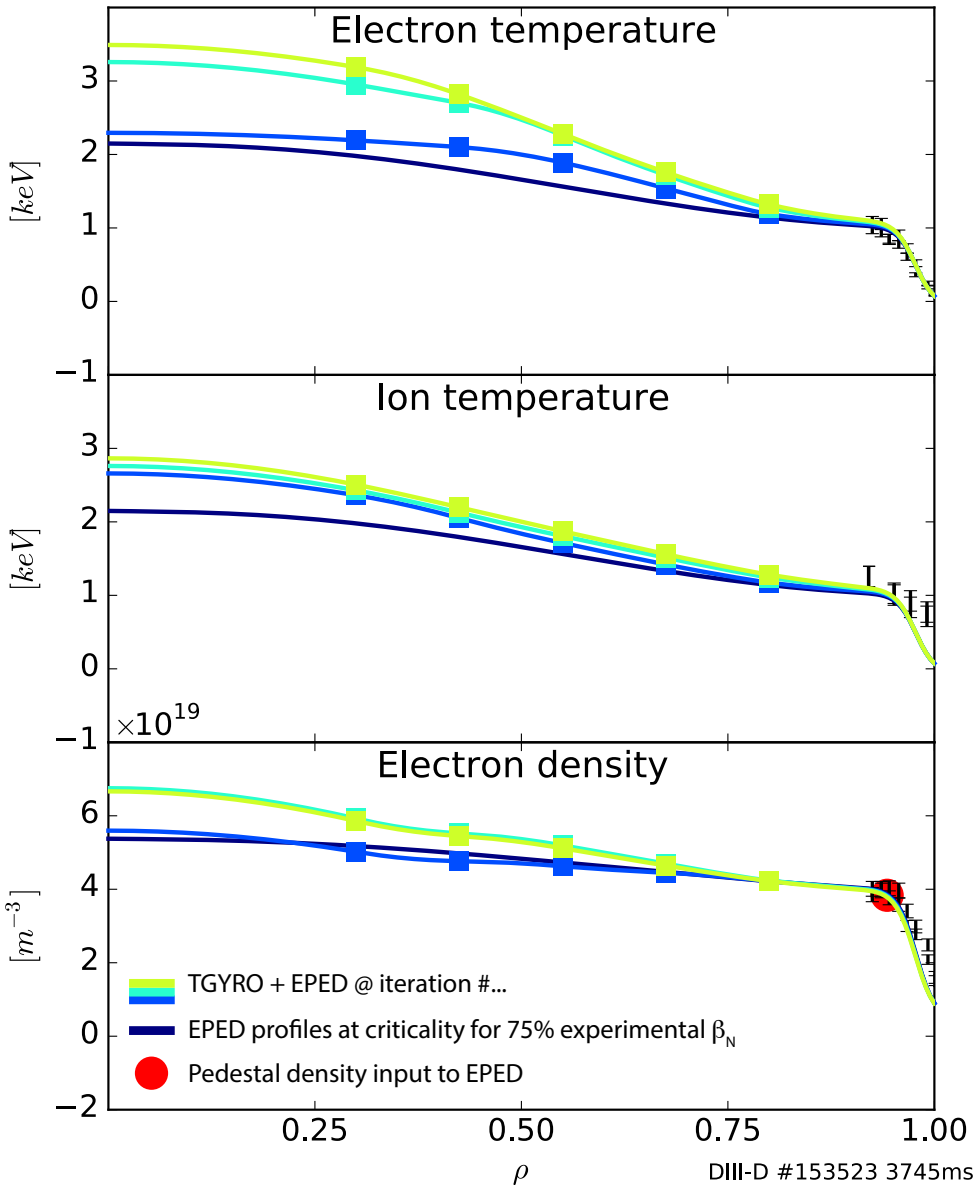
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N
- TGYRO to predict profiles and β_N. Then iterate...



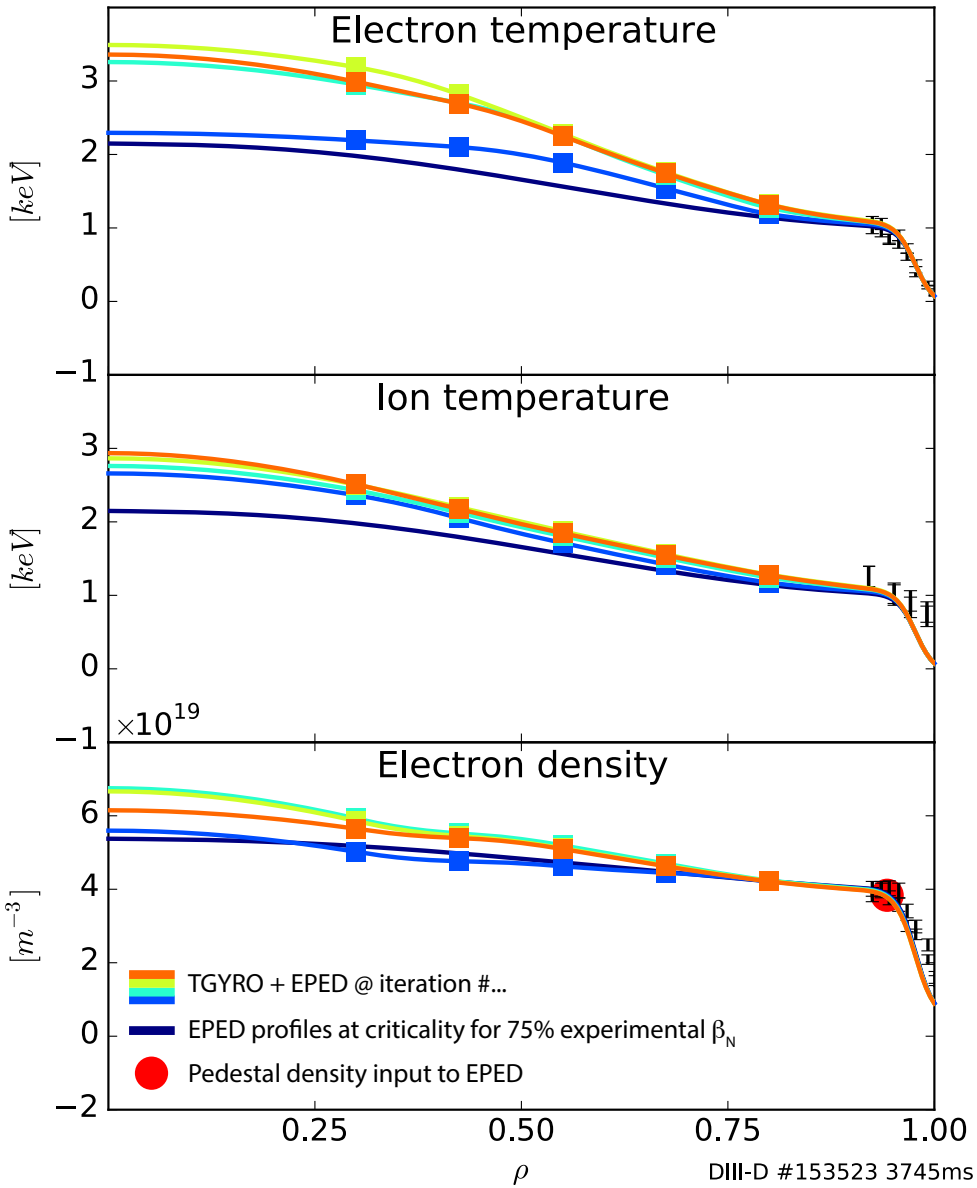
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N
- TGYRO to predict profiles and β_N. Then iterate...



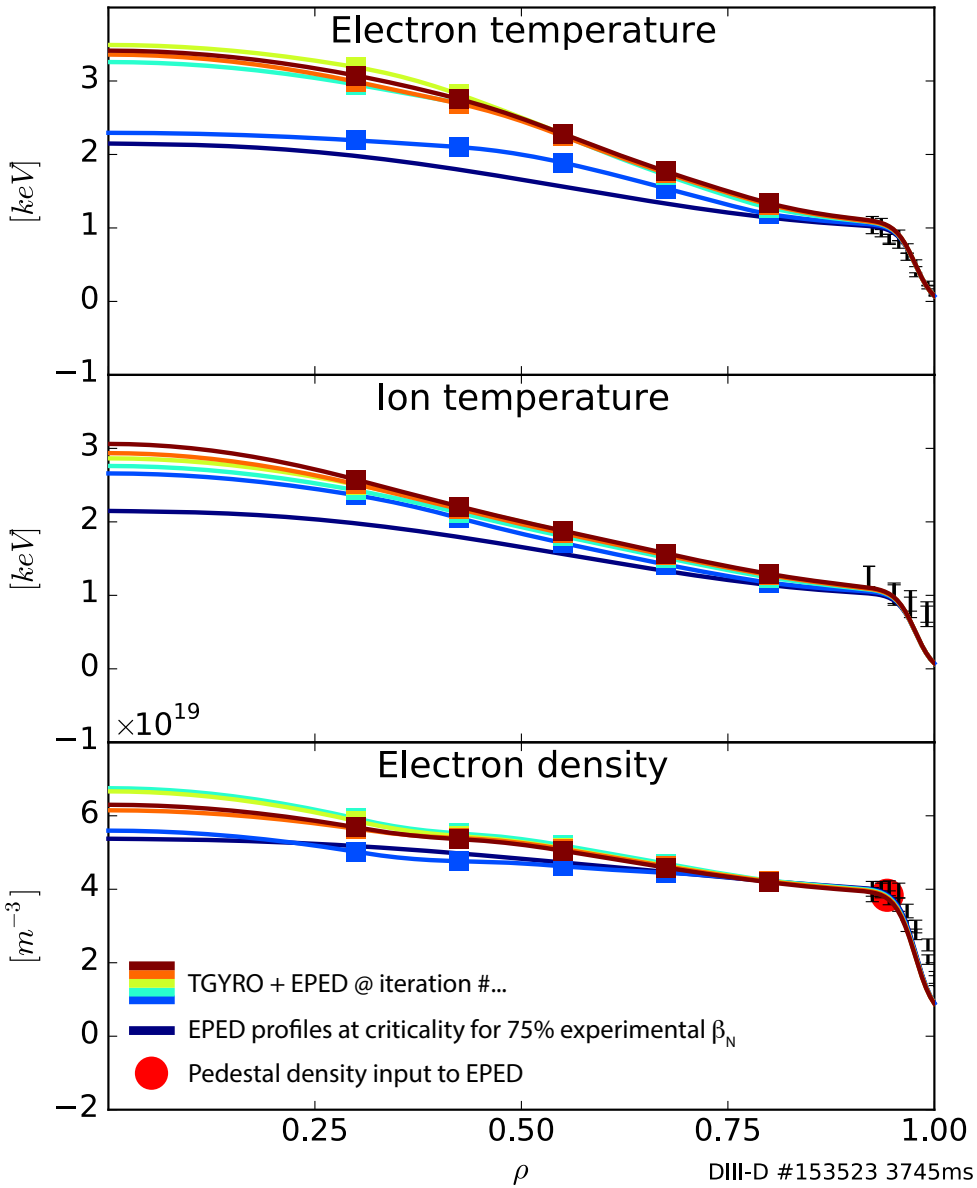
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N
- TGYRO to predict profiles and β_N. Then iterate...



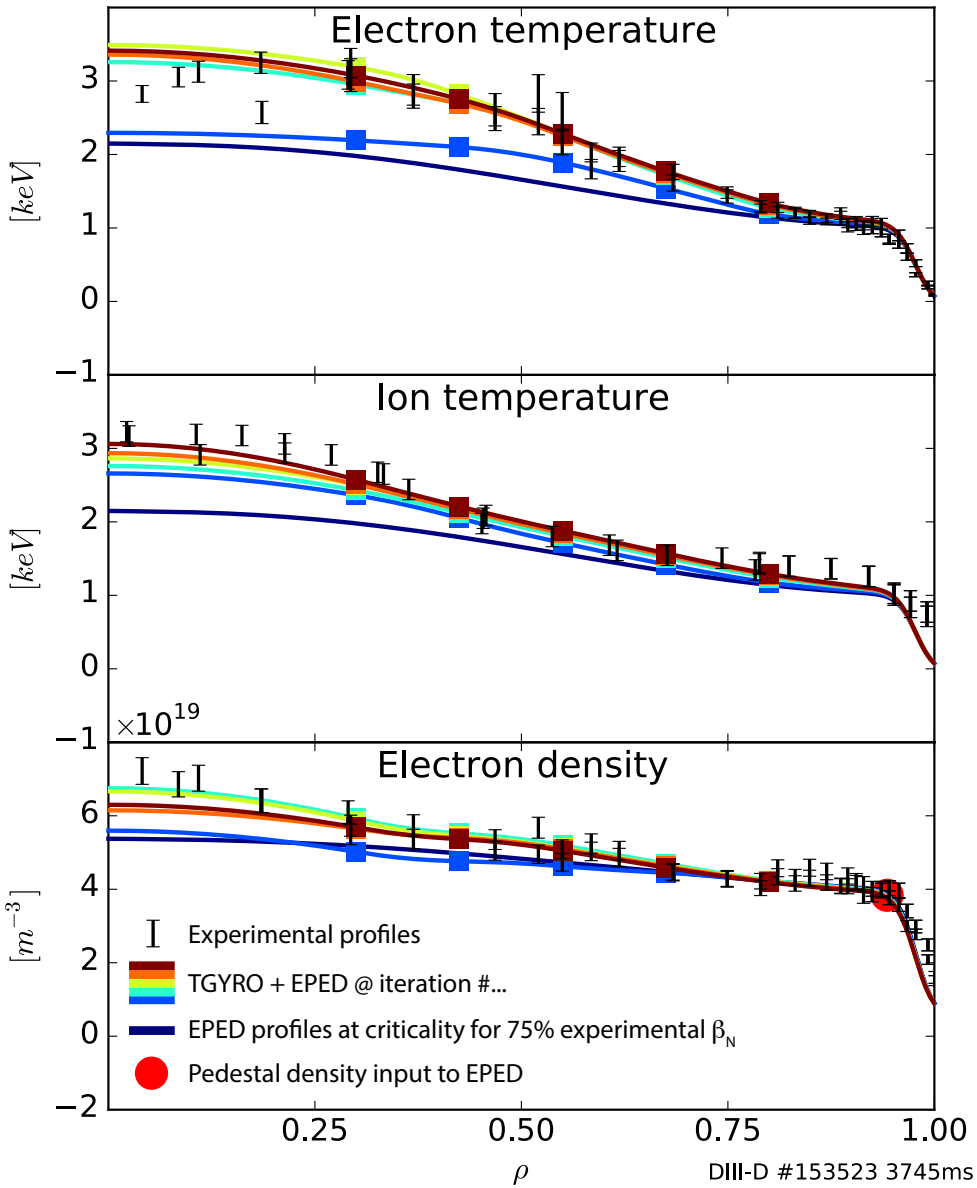
First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N
- TGYRO to predict profiles and β_N. Then iterate...



First applied this workflow to DIII-D ITER baseline scenario with low torque and electron heating

- Inputs are *shape*, B_t, $J(r)$, $\omega(r)$, $n_{e,\mathrm{ped}}$, guess for β_N
- Start with EPED. First run uses (poor) initial guess for β_N
- TGYRO to predict profiles and β_N. Then iterate...
- Measurements recovered very well across the plasma
- Pedestal prediction is weakly sensitive to β_N Shafranov shift stabilization



Outline

Running TRANSP from OMFIT is in line with AToM philosophy

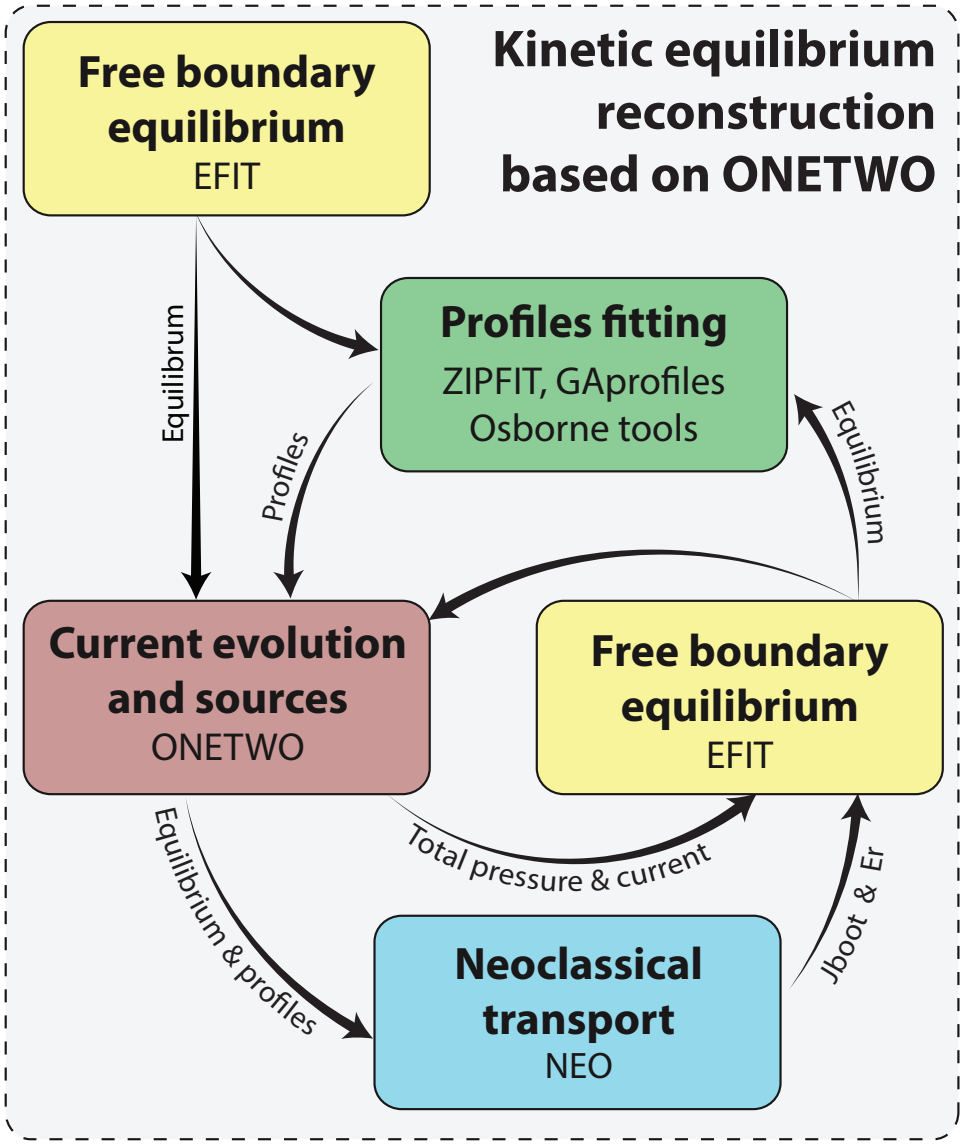
TRANSP is a comprehensive transport code

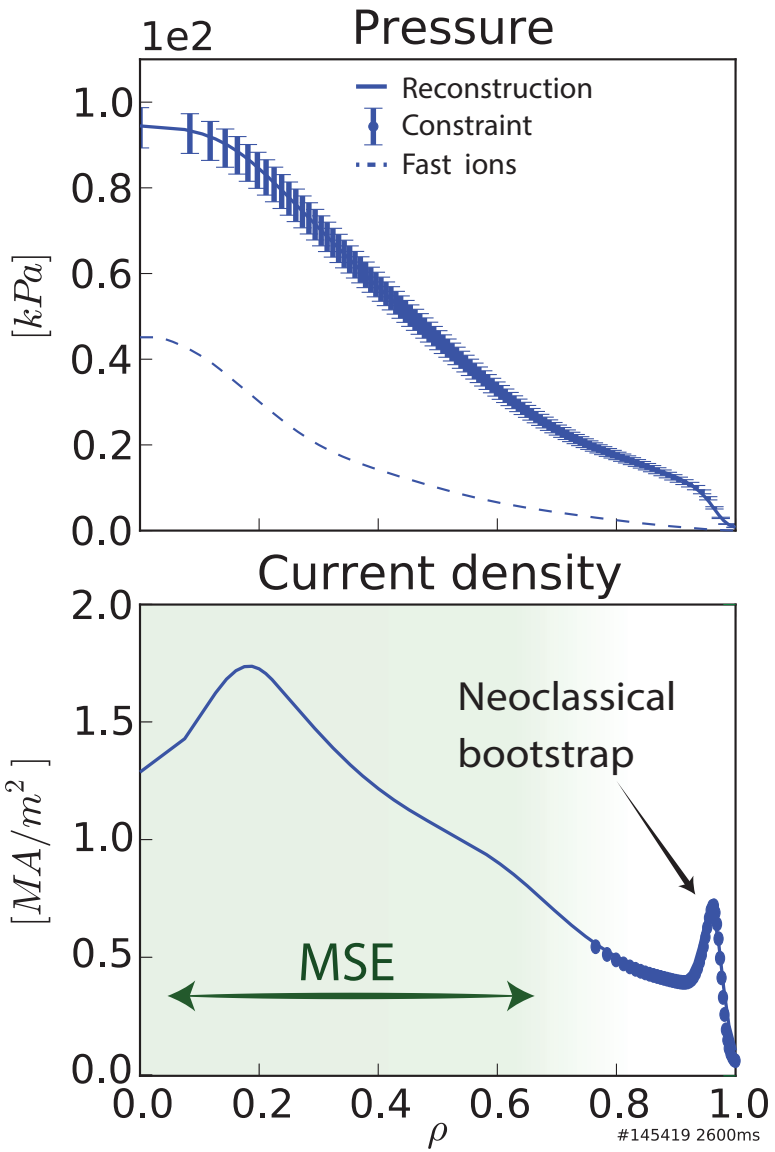
- Widely adopted
- Validated on numerous devices
- Solid infrastructure $\rightarrow$ TRANSP grid
- Responsive support

OMFIT can provide a convenient interface to user, experimental data and codes outside of TRANSP

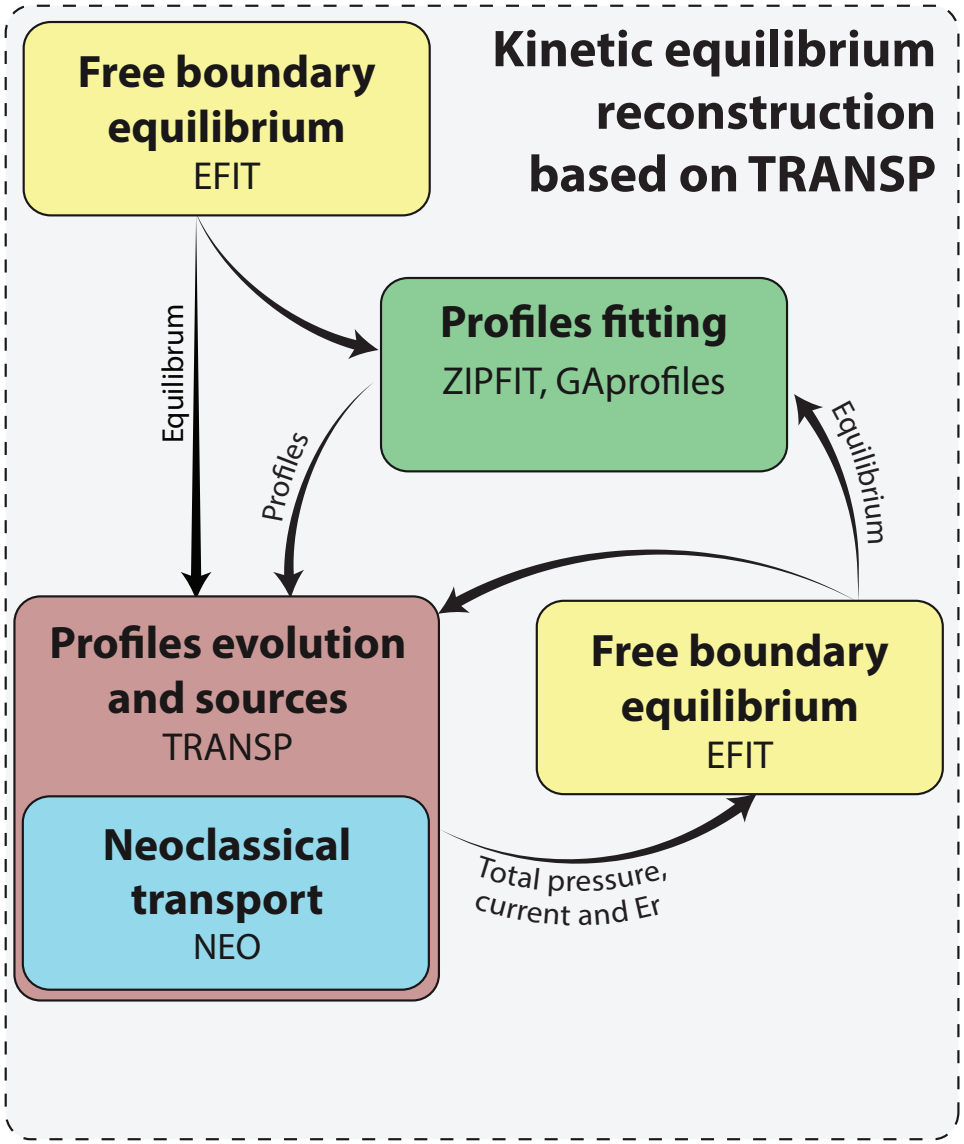
Initial OMFIT-TRANSP integration sparked by need to generate time-dependent DIII-D kinetic equilibrium reconstructions based on TRANSP runs (B. Grierson)

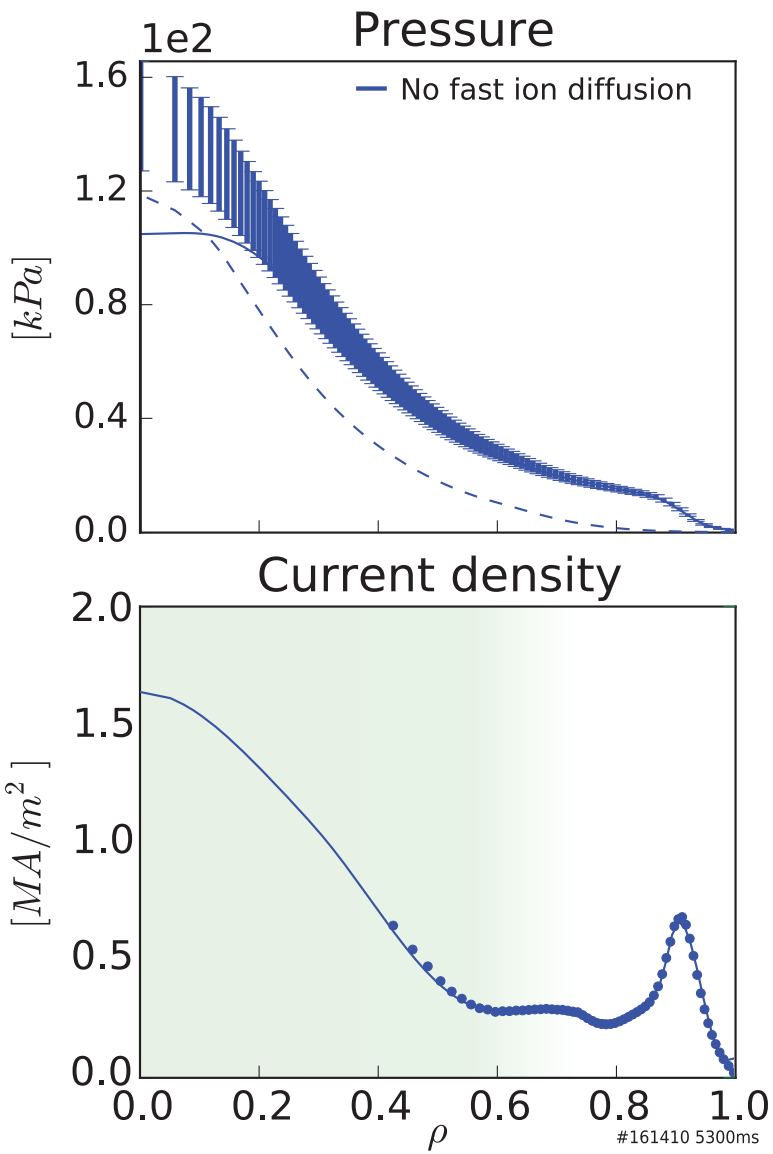
DIII-D kinetic EFIT analysis originally based upon ONETWO



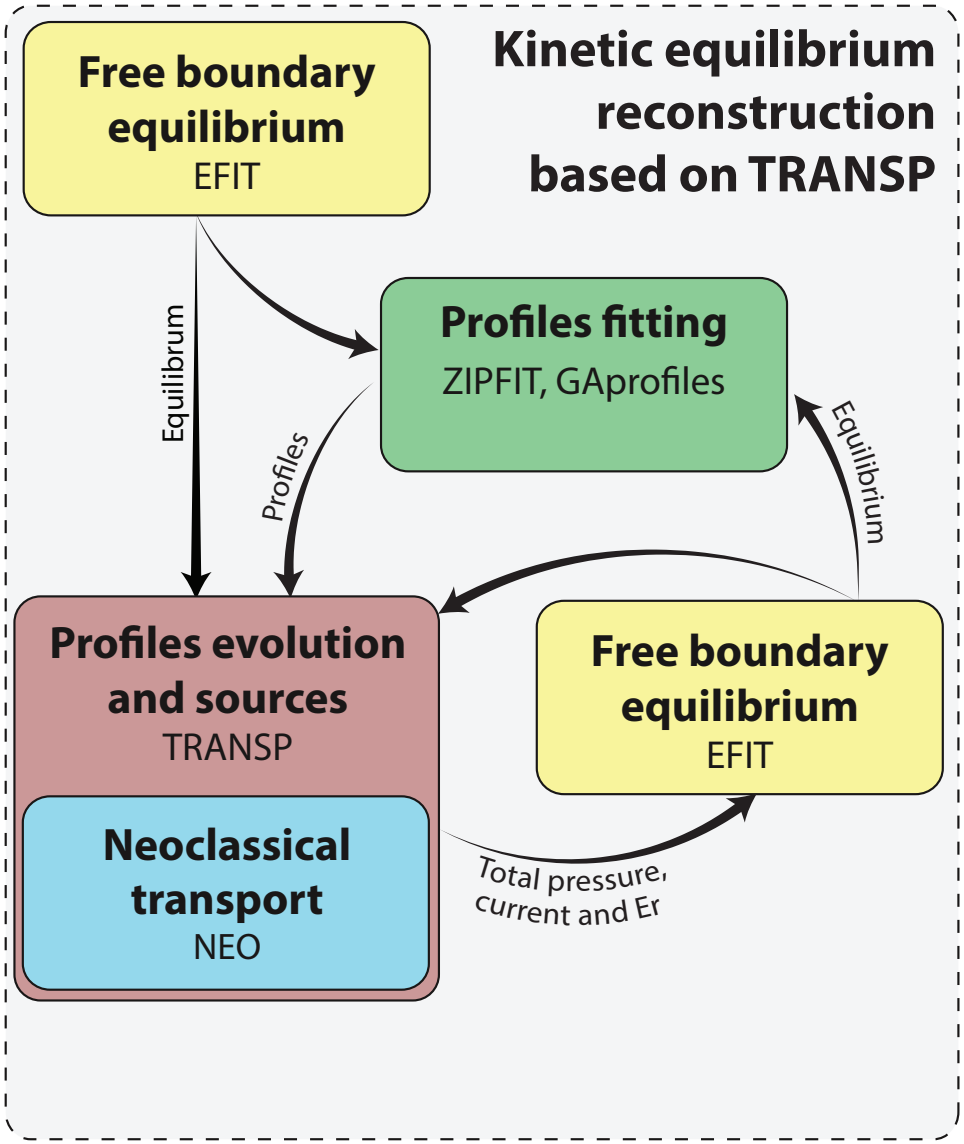


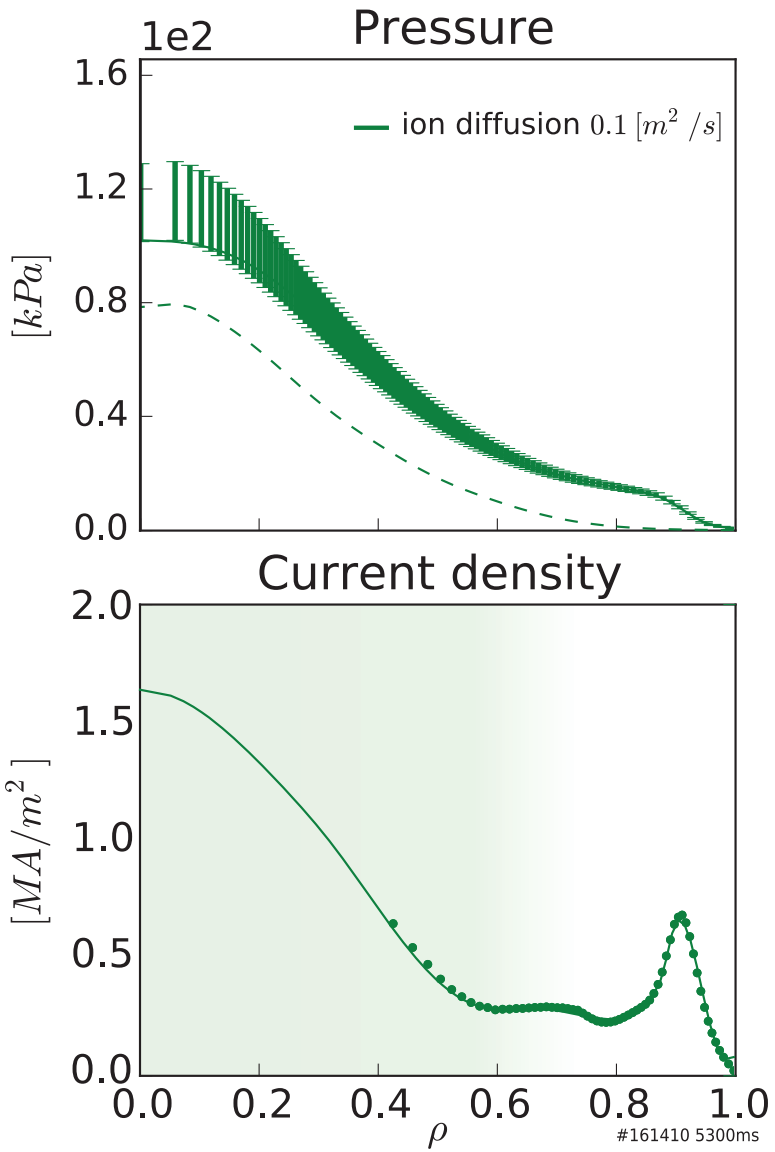
Developed new TRANSP module and integrated it as part of the OMFIT kineticEFIT module



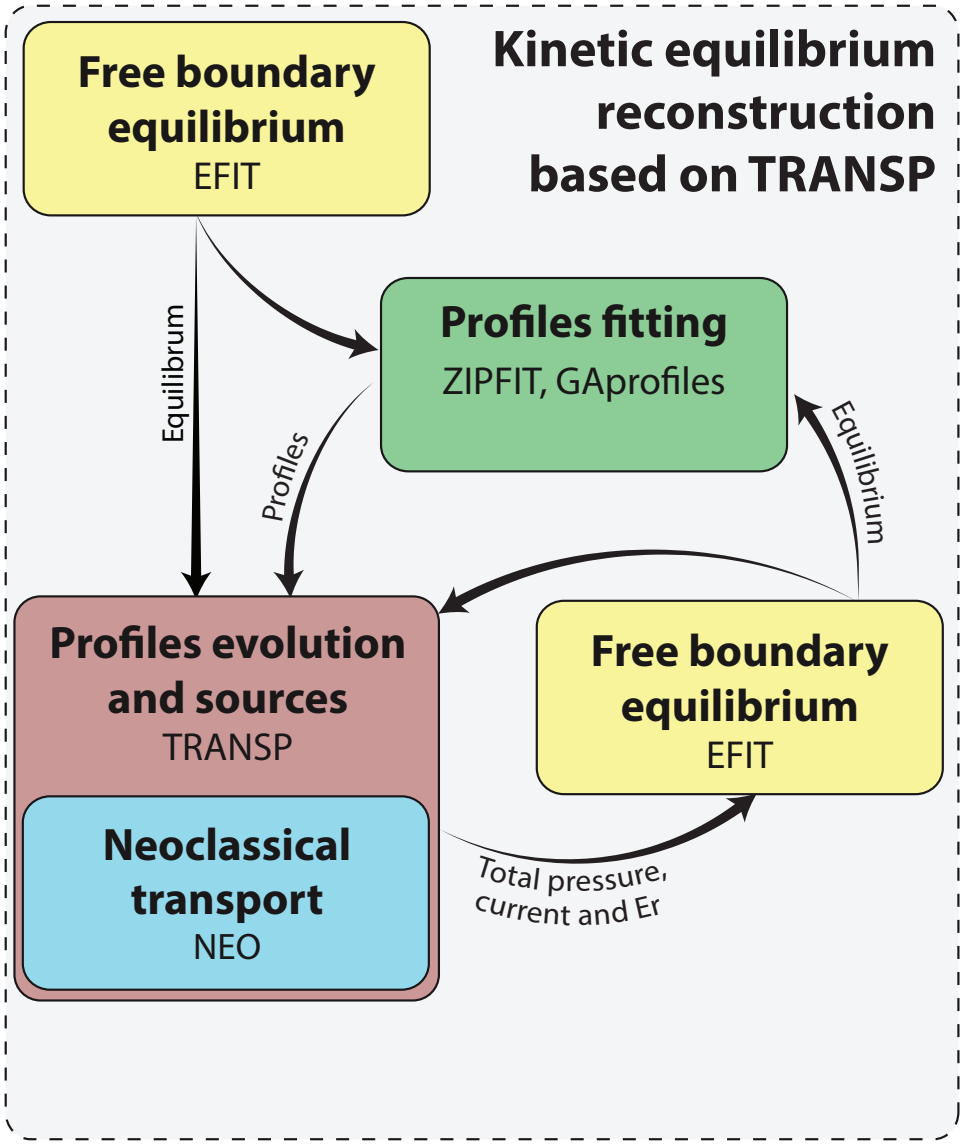


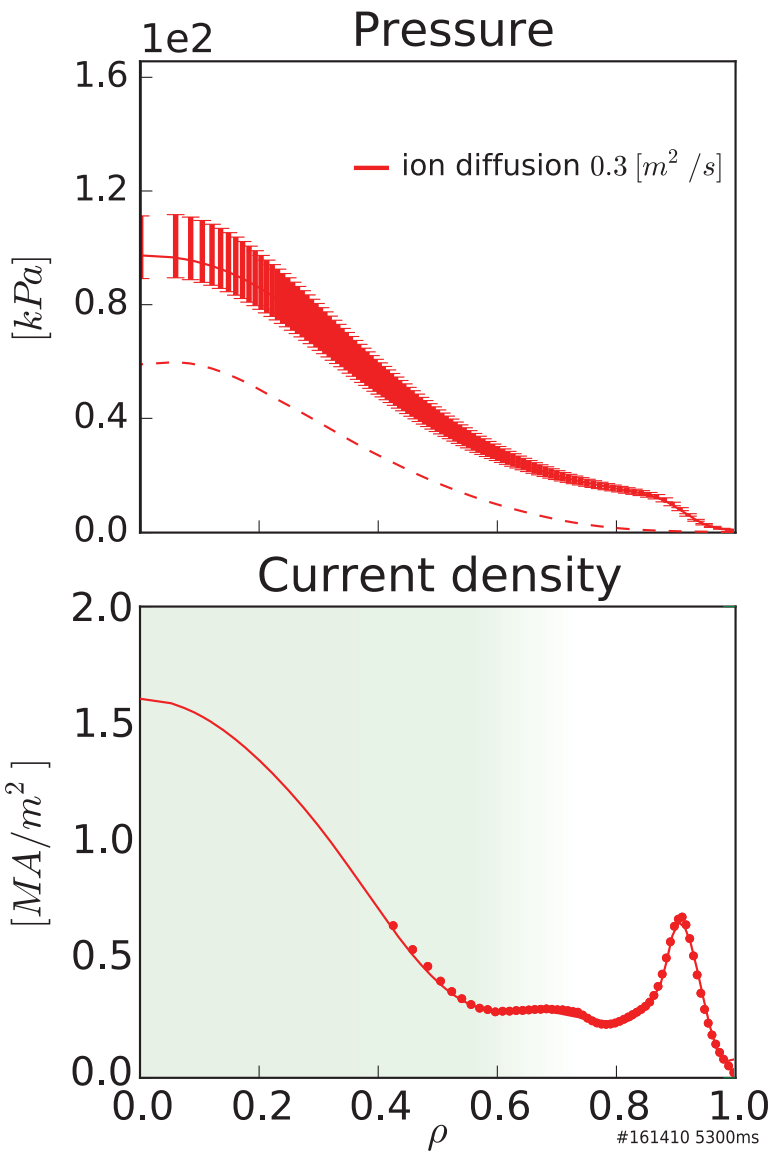
Developed new TRANSP module and integrated it as part of the OMFIT kineticEFIT module



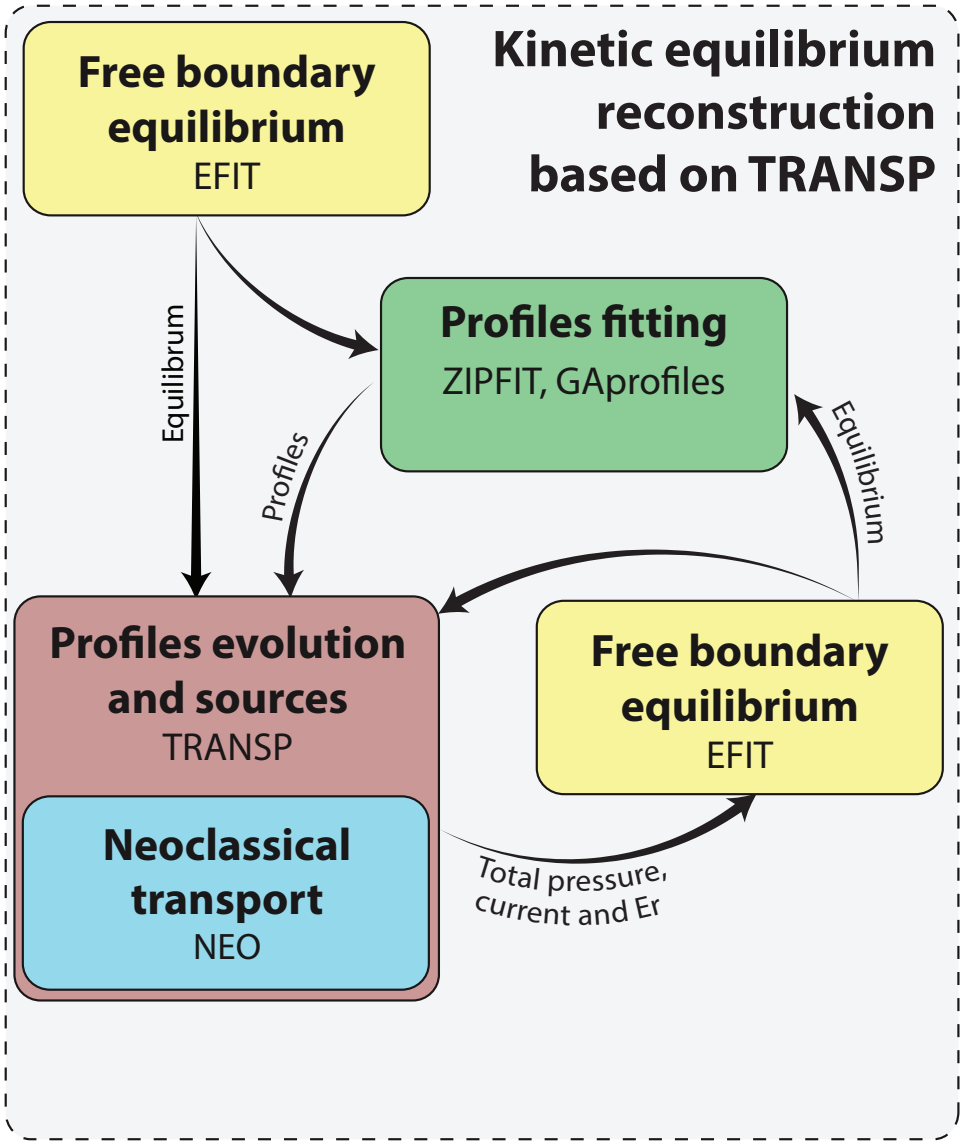


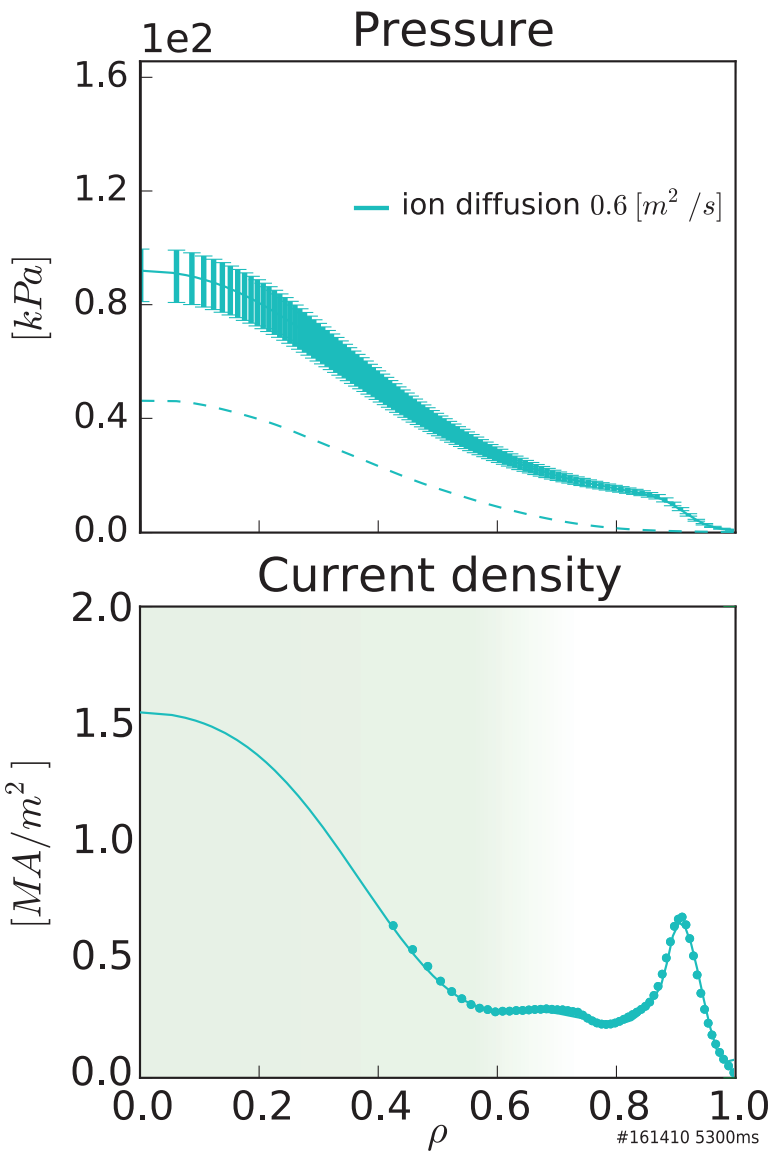
Developed new TRANSP module and integrated it as part of the OMFIT kineticEFIT module



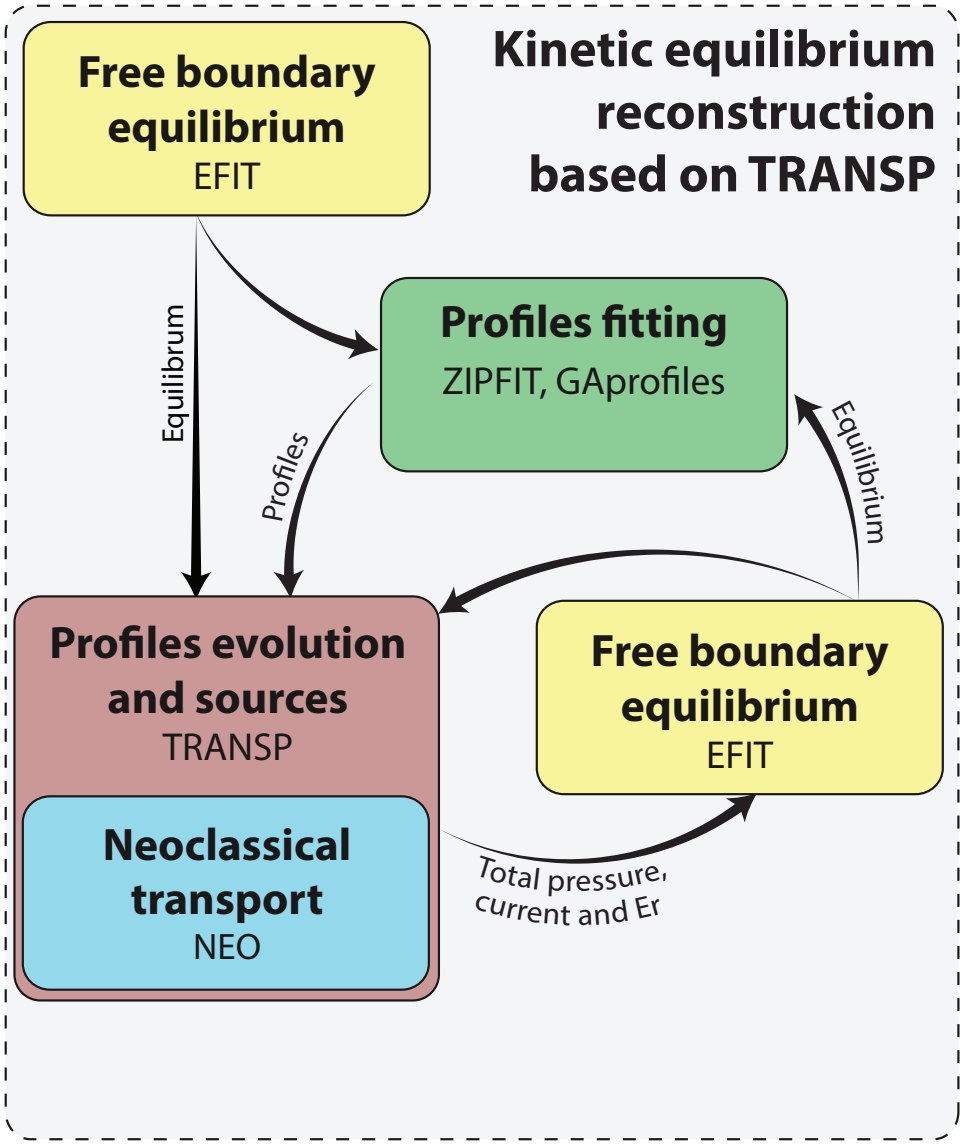


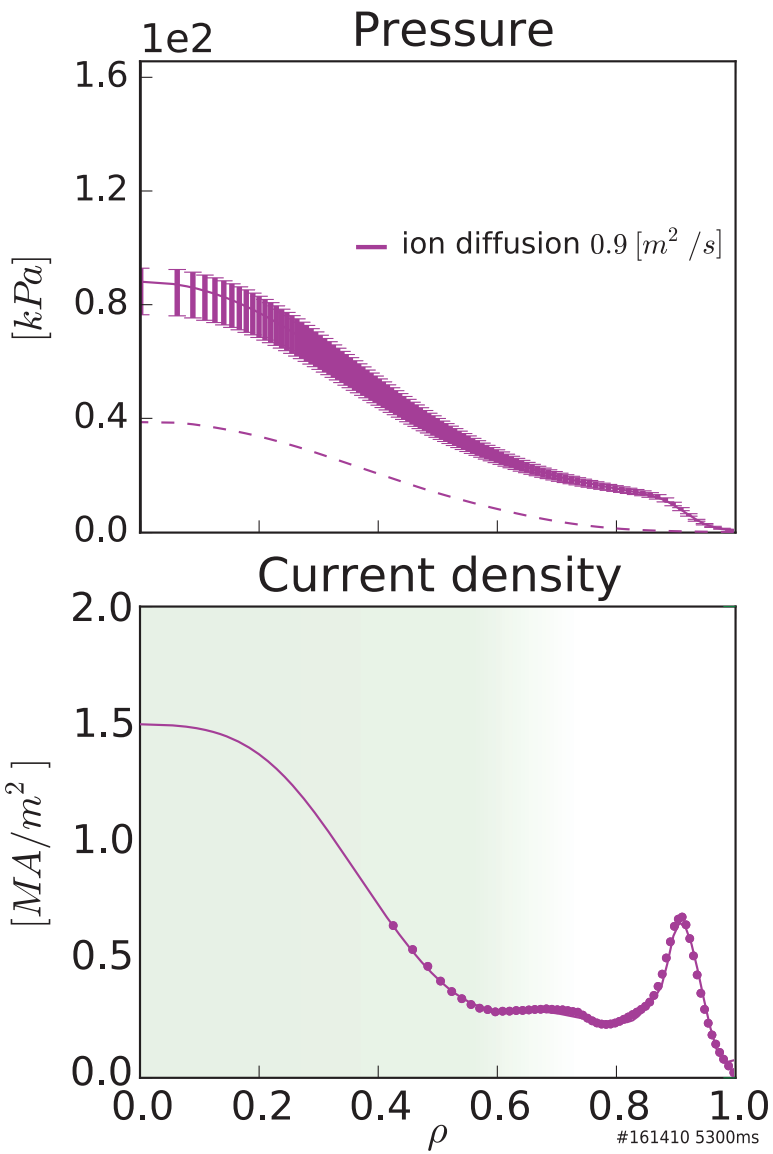
Developed new TRANSP module and integrated it as part of the OMFIT kineticEFIT module



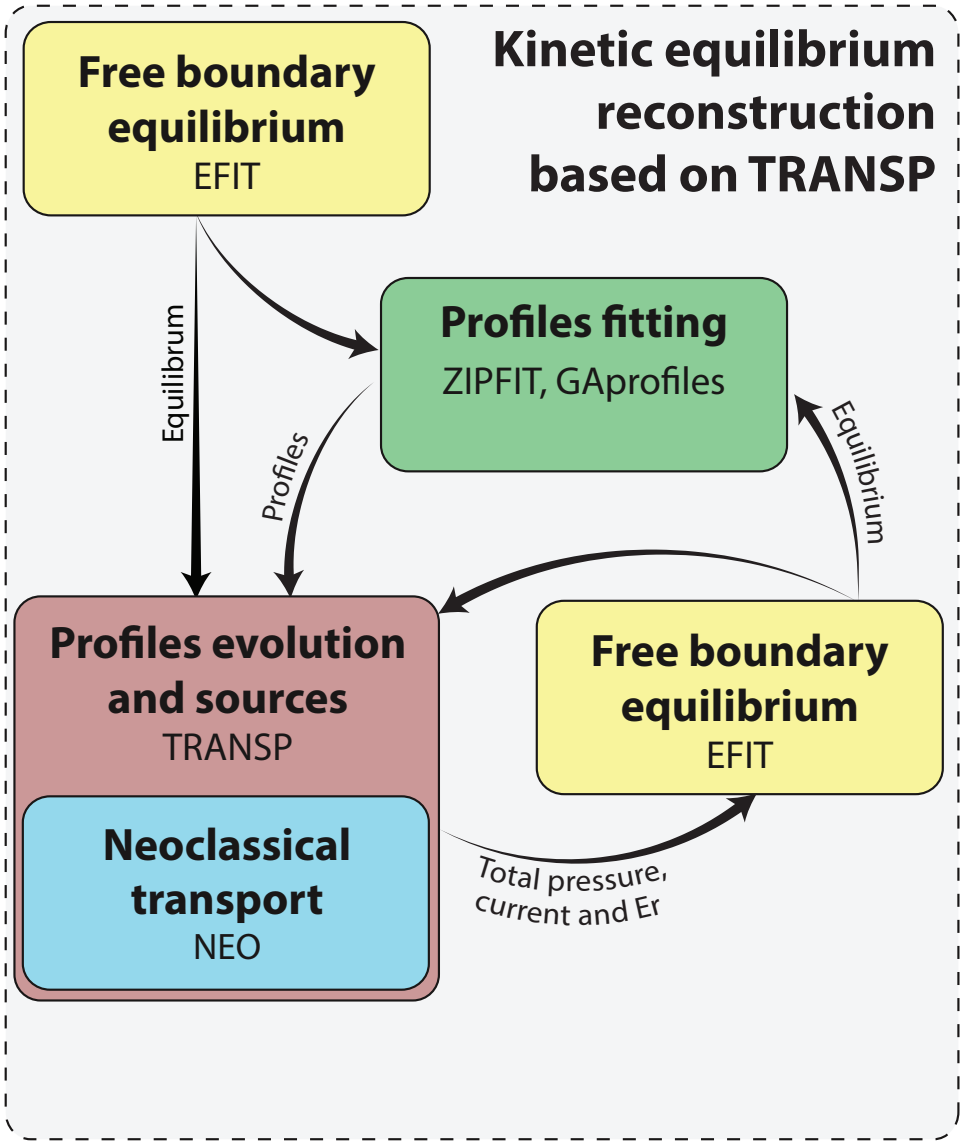


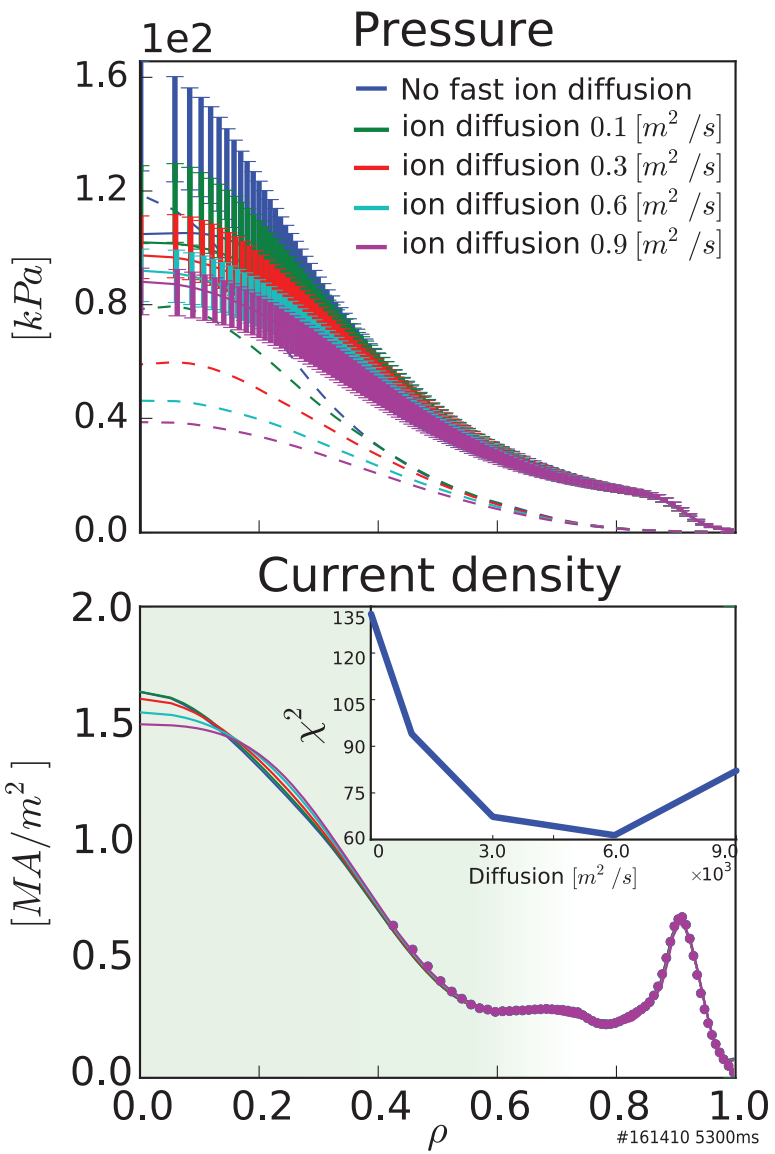
Developed new TRANSP module and integrated it as part of the OMFIT kineticEFIT module



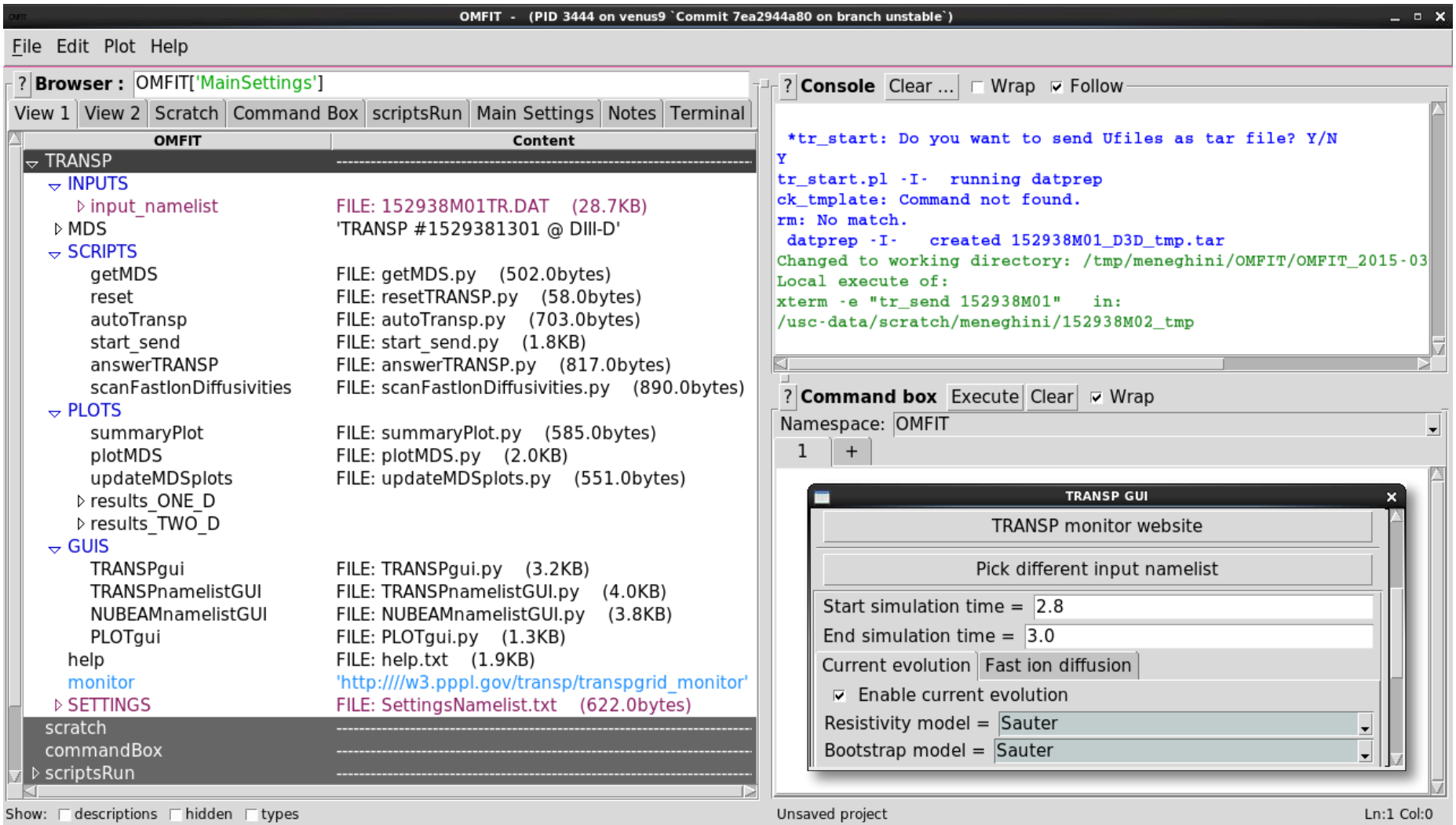


Developed new TRANSP module and integrated it as part of the OMFIT kineticEFIT module





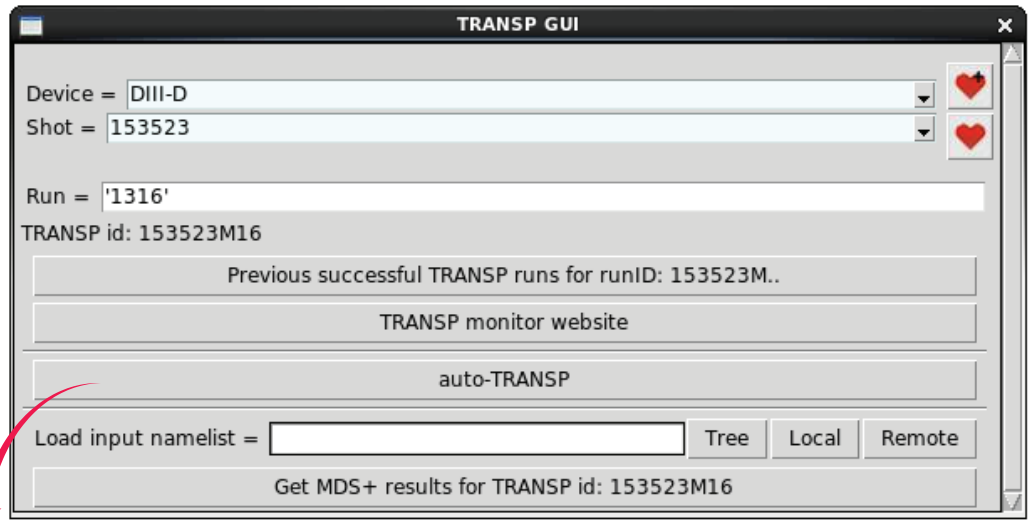
Overview of the TRANSP module in OMFIT



TRANSP module uses *auto*-TRANSP tool to setup input U-files and initial namelist

Preparing TRANSP runs is a complex problem, that is machine dependent

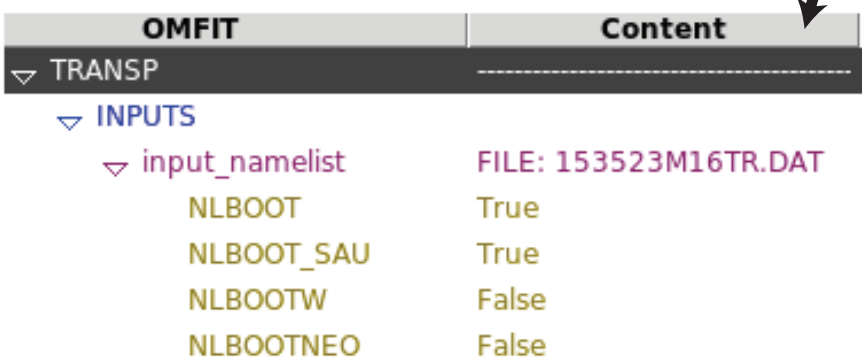
Example of how OMFIT can *"stand on the shoulders of giants"* by (remotely) executing tools that were developed over the years

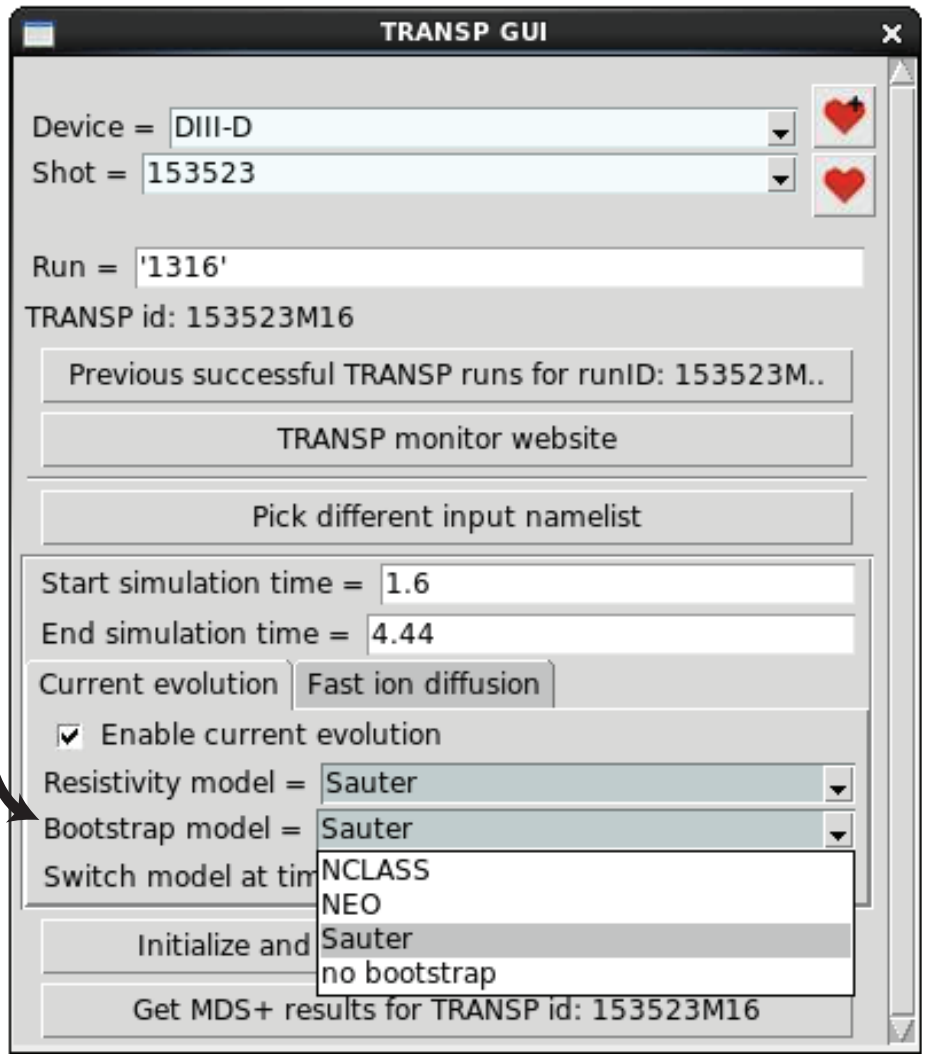


OMFIT API allows to easily build GUIs that manage the complex inter-dependencies within the TRANSP namelist

```
ComboBox(["root['INPUTS']['input_namelist']['%s']"%k for k in ['NLBOOT','NLBOOT_SAU','NLBOOTW','NLBOOTNEO']],
         {'no bootstrap':[False,False,False,False],
          'Sauter'      :[True ,True ,False,False],
          'NCLASS'      :[True ,False,True ,False],
          'NEO'         :[True ,False,False,True ]},
         'Bootstrap model')
```

e.g. set bootstrap current model

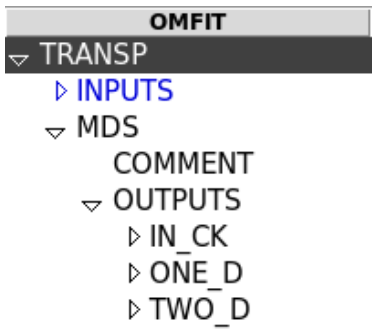


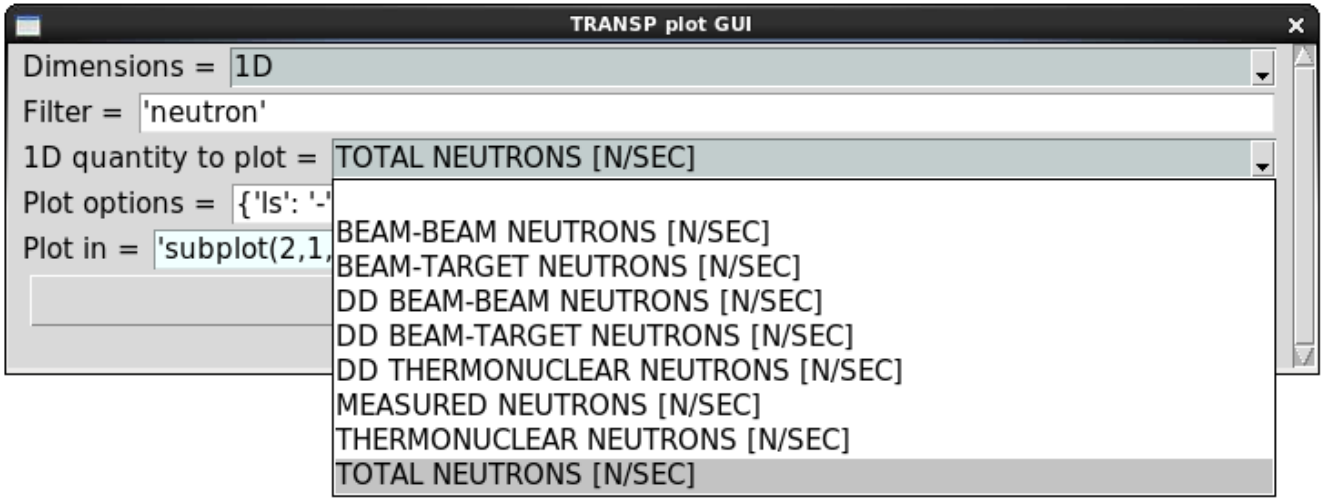


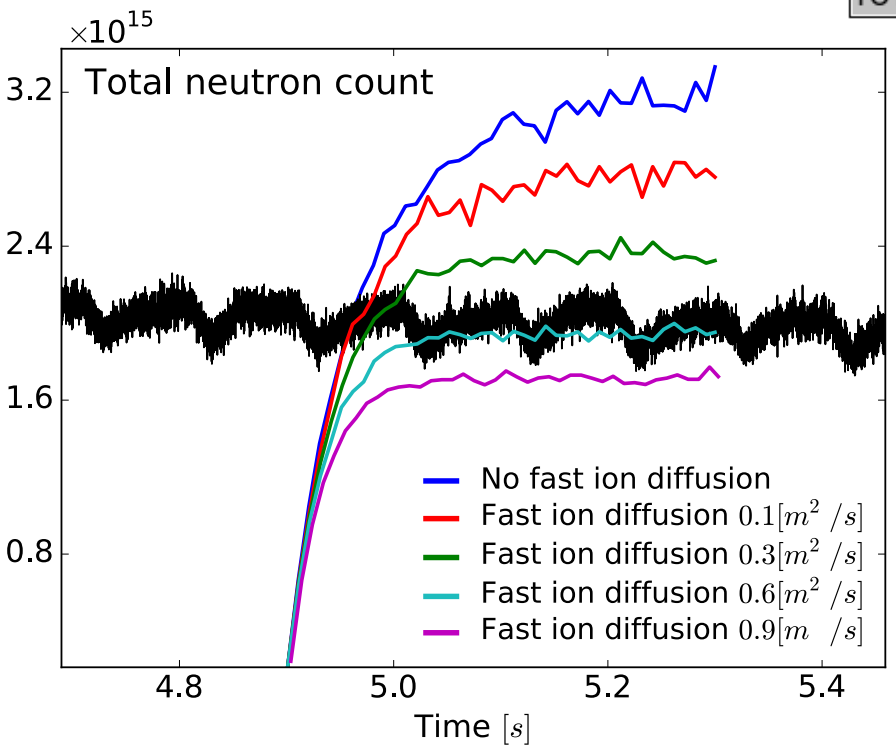
Scripting within OMFIT allows many possibilities e.g. trivial to scan fast ion diffusion

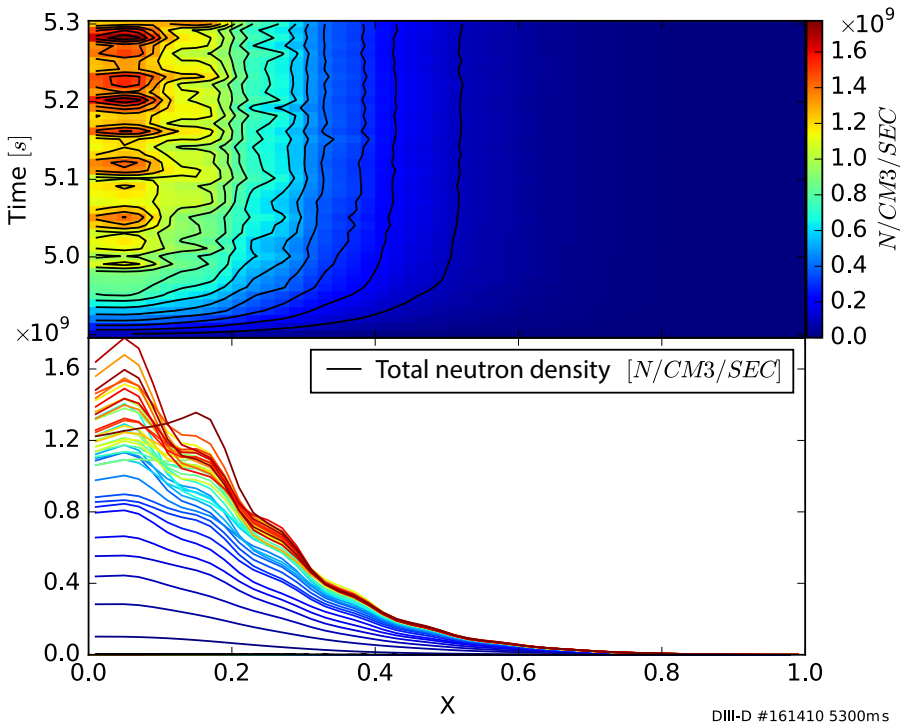
```
starting_runID=2
diffusivities=[1000,3000,6000,9000]
for k,diffusivity in enumerate(diffusivities):
    comment='fast ion diffusion %e cm**2/s'%diffusivity

    root['SETTINGS']['PHYSICS']['runID']='13'+'%02d'%(starting_runID+k)

    root['INPUTS']['input_namelist']['NMDIFB']=1
    root['INPUTS']['input_namelist']['BDIFB']=diffusivity
    root['INPUTS']['input_namelist']['CDIFB']=diffusivity

    root['SCRIPTS']['start_send'].run(send=True,
                                      interactive=False,
                                      comment=comment)
```

OMFIT can post-process results from TRANSP jobs both for plotting or use as part of a workflow



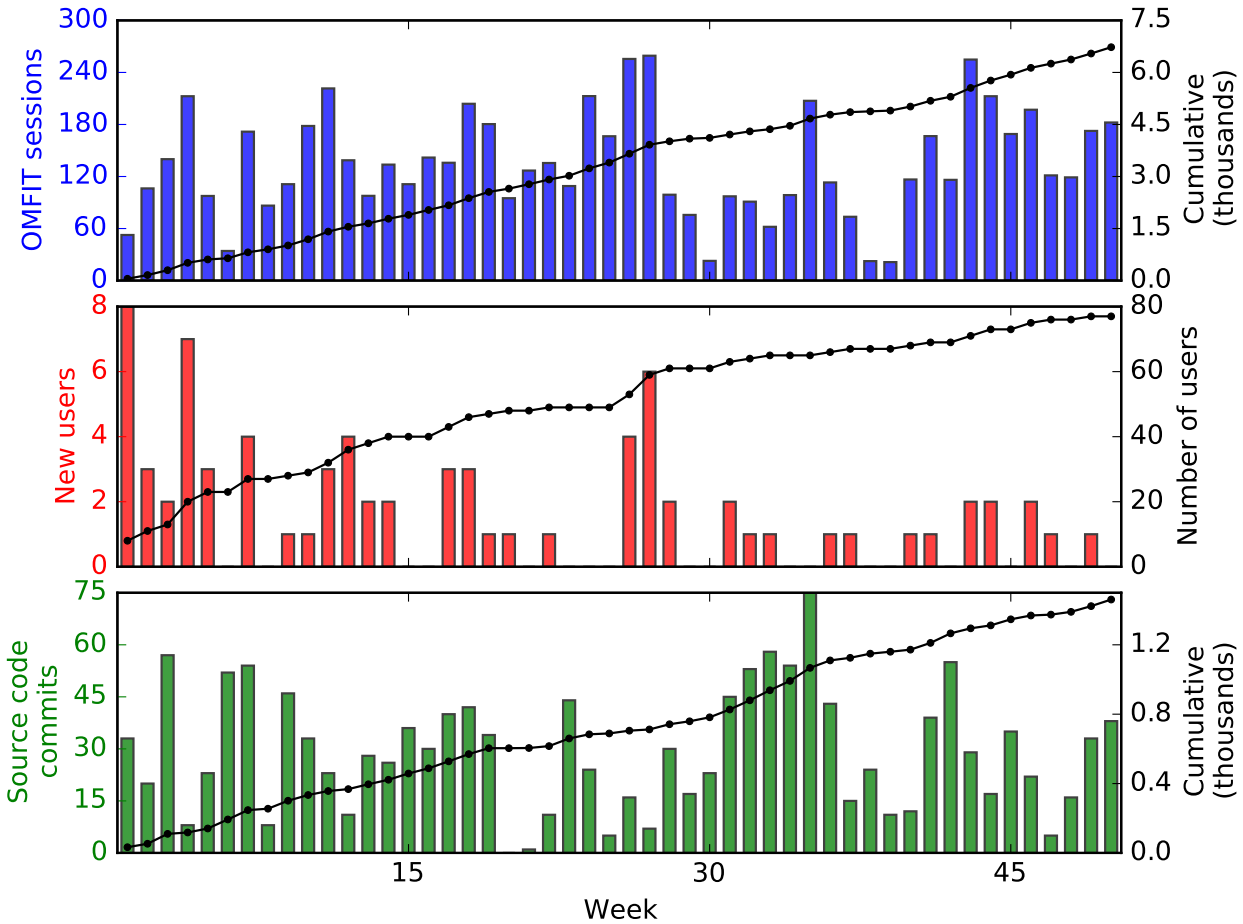






Outline

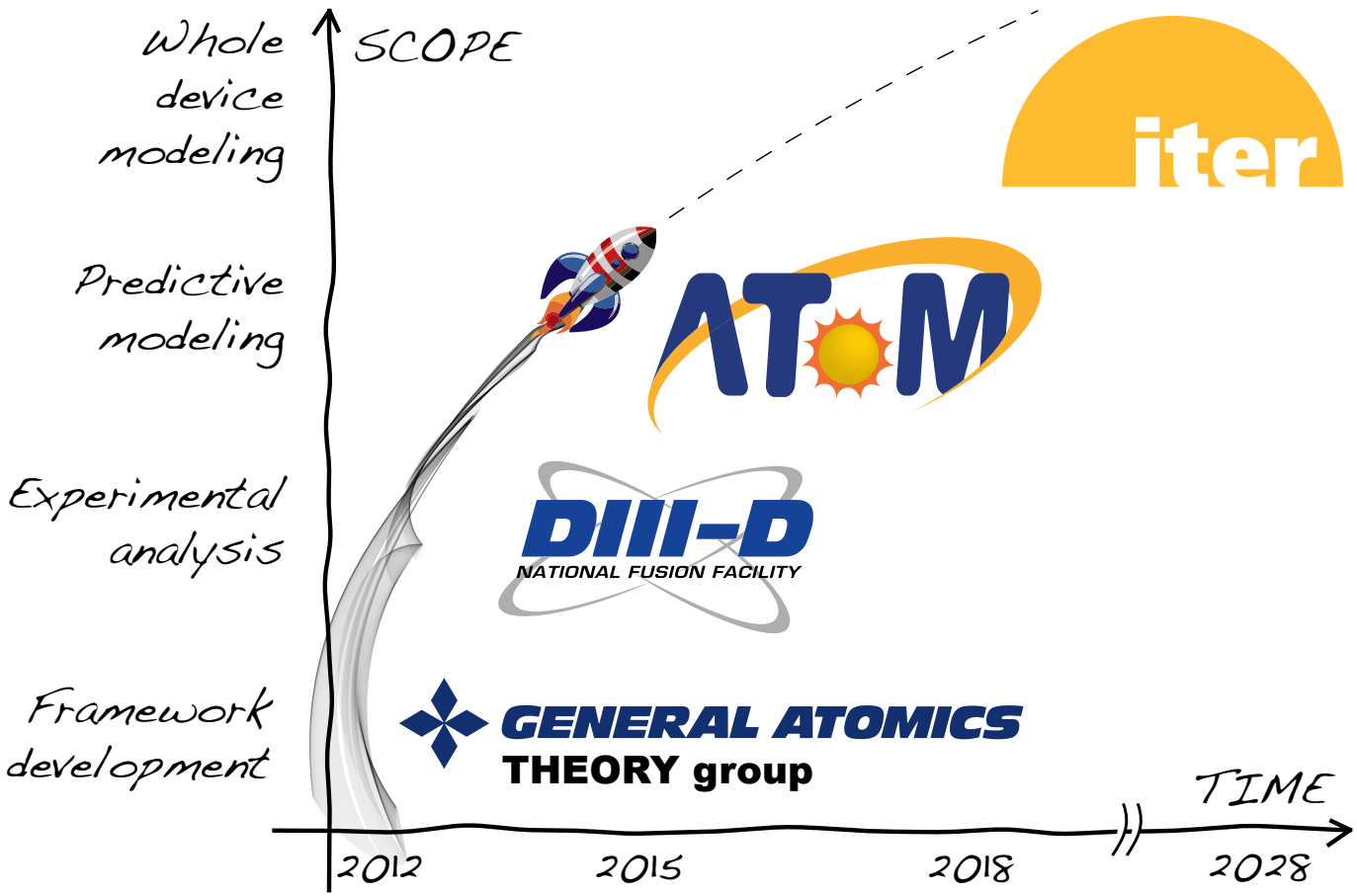
OMFIT is a vibrant project, actively used by many researchers, for a broad range of applications



- Kinetic equilibrium reconstruction
- Core stability calculations
- Edge stability diagrams
- Self-consistent steady state transport simulations
- Neoclassical theory validation
- Validation of Sauter Vs kinetic neoclassical
- Divertor design
- Helicon wave system design
- Magnetic flutter theory validation
- Neoclassical toroidal viscosity theory validation
- Building of neural network transport models
- Self-consistent study of interaction between NTM and ITG
- Experimental data analysis for transport and 3D fields
- ...

Future perspectives

- Continue *"to enhance and extend present modeling capabilities, by supporting, leveraging, and integrating existing research"*, with the aim of boosting a **bottom-up** integrated modeling vision



Future perspectives

- Continue *"to enhance and extend present modeling capabilities, by supporting, leveraging, and integrating existing research"*, with the aim of boosting a **bottom-up** integrated modeling vision

- Self-consistent core-edge modeling
  - Validate TGYRO-EPED workflow for more DIII-D discharges
  - For ITER, optimize fusion performance as a function of pedestal density and other key parameters
  - AToM plans to extend model beyond the separatrix with COGENT

- TRANSP-OMFIT integration
  - OMFIT provides a convenient interface to user, experimental data, and codes outside of TRANSP
  - No changes required to TRANSP or user workflows
  - Experiments using TRANSP could use OMFIT to generate their kinetic equilibrium reconstructions
  - Additional interesting applications: e.g. OMFIT driving TRANSP runs to minimize a cost function (optimize current profile, fusion power, ...)