Large scale validation simulations with python jetto-tools wrappers: application to 8 WEST pulses.

Enzo VERGNAUD

enzo.vergnaud@cea.fr

TSVV-H Workshop 16-06-2026

Outline

1. Large-scale validation challenges: the WEST example
2. The jetto-tools framework
3. Python wrappers for jetto-tools to ease the large-scale job submission
4. TGLFNN as a turbulent transport surrogate
5. Proof of principle of the large-scale submission pipeline

Outline

1. **Large-scale validation challenges: the WEST example**
2. The jetto-tools framework
3. Python wrappers for jetto-tools to ease the large-scale job submission
4. TGLFNN as a turbulent transport surrogate
5. Proof of principle of the large-scale submission pipeline

Large-scale validation challenges

Since 2020:

- Integrated modelling (IM) applied on a few shots on different machines
- Successful predictions with TGLF-sat2 with validation metrics to attest the « successful » meaning

But:

- Only a few shots have been simulated and not enough statistics to have strong validation of the IM
- Tools to do multiple runs have been developped (Duqtools, jetto-tools,….) but not routinely used

Paper	Machine	Integrated Modelling Framework	J Diffu -sion	Heat Transp.	Particle Transp.	Particle feed -back	Heat Sources	Impurity Transp.
[Angioni 2022 NF] [Fajardo 2024 NF]	AUG	ASTRA+ TGLFsat2	✓	✓	✓	✓	NBI/ ECRH	He B, W & Ar
[Marin 2025 NF]	TCV	HFPS+ TGLFsat2	✓	✓	✓	✓	✗	C
[Fonghetti 2025 NF]	WEST	HFPS+ TGLFsat2	✓	✓	✓	✓	LHCD	✗
[Vergnaud 2026 PPCF(submitted)]	WEST	HFPS+ TGLFsat2	✓	✓	✓	✓	LHCD	N & W

Large-scale validation challenges

Since 2020:

- Integrated modelling (IM) applied on a few shots on different machines
- Successful predictions with TGLF-sat2 with validation metrics to attest the « successful » meaning

But:

- Only a few shots have been simulated and not enough statistics to have strong validation of the IM
- Tools to do multiple runs have been developped (Duqtools, jetto-tools,….) but not routinely used

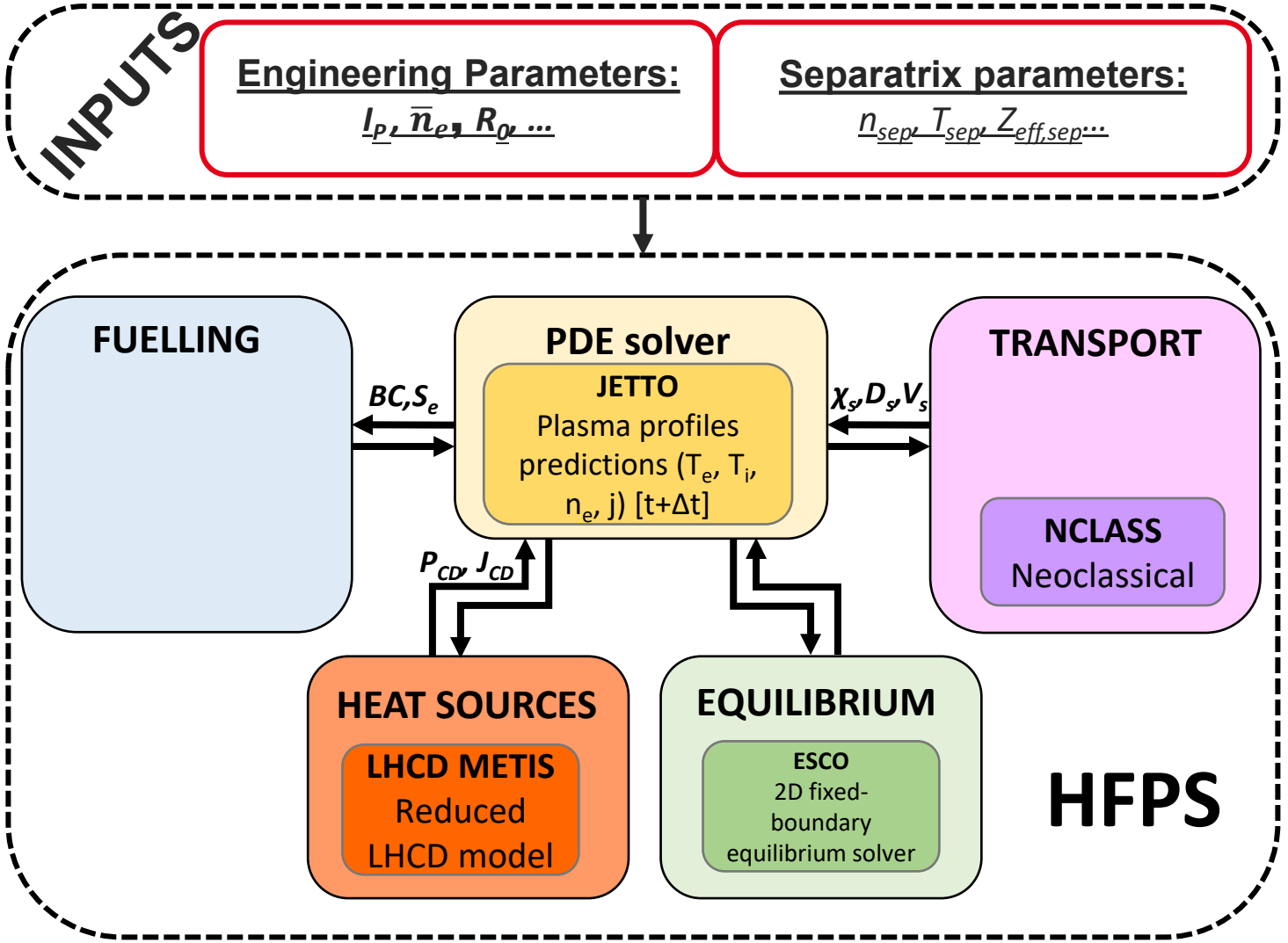
Paper	Machine	Integrated Modelling Framework	J Diffu -sion	Heat Transp.	Particle Transp.	Particle feed -back	Heat Sources	Impurity Transp.
[Angioni 2022 NF] [Fajardo 2024 NF]	AUG	ASTRA+ TGLFsat2	✓	✓	✓	✓	NBI/ ECRH	He B, W & Ar
[Marin 2025 NF]	TCV	HFPS+ TGLFsat2	✓	✓	✓	✓	✗	C
[Fonghetti 2025 NF]	WEST	HFPS+ TGLFsat2	✓	✓	✓	✓	LHCD	✗
[Vergnaud 2026 submit.]	WEST	HFPS+ TGLFsat2	✓	✓	✓	✓	LHCD	N & W

→ 8 WEST L-mode pulses with HFPS on the whole radius (no SOL simulated)

Stepwise approach by adding physics layers

Simulation Name	J Diffusion	Heat Transp.	Particle Transp.	Impurity Transp.
Pred J	✓	✗	✗	✗

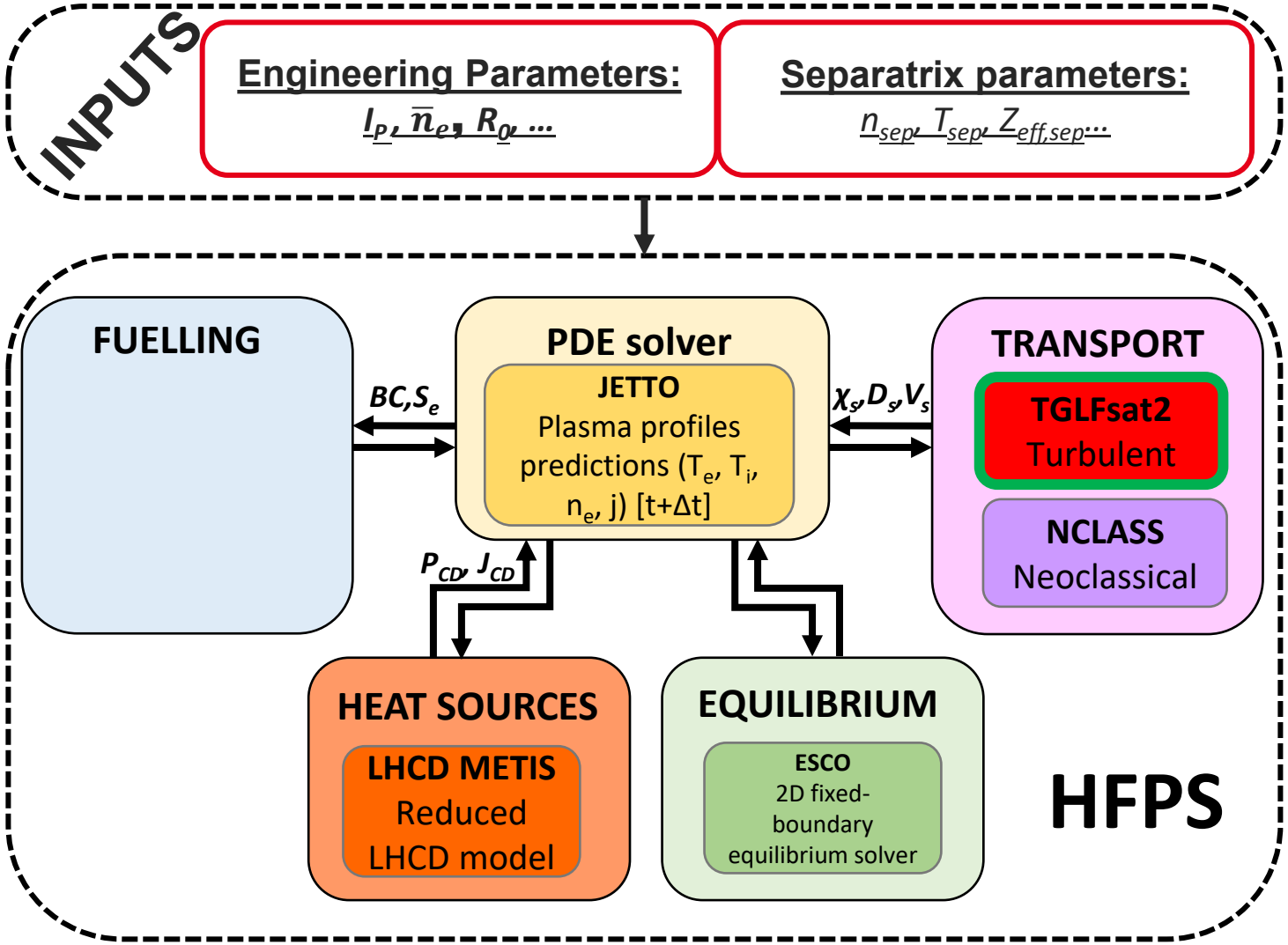
Only current diffusion considered



Stepwise approach by adding physics layers

Simulation Name	J Diffusion	Heat Transp.	Particle Transp.	Impurity Transp.
Pred J	✓	✗	✗	✗
Pred JT	✓	✓	✗	✗

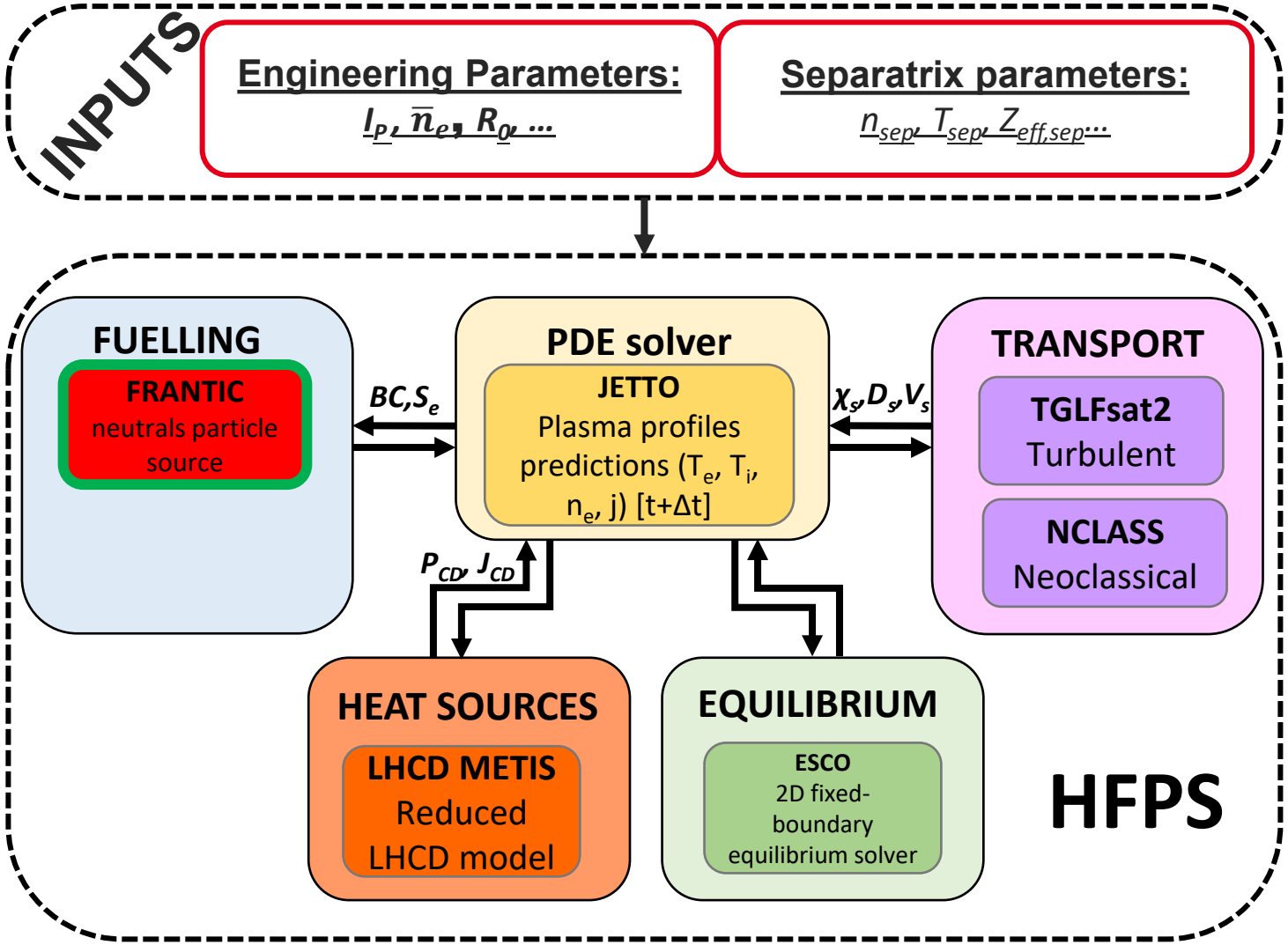
+ heat transport



Stepwise approach by adding physics layers

Simulation Name	J Diffusion	Heat Transp.	Particle Transp.	Impurity Transp.
Pred J	✓	✗	✗	✗
Pred JT	✓	✓	✗	✗
Pred JTn	✓	✓	✓	✗

+ **particle transport** : particle source adjustment by feedback on electron line averaged density

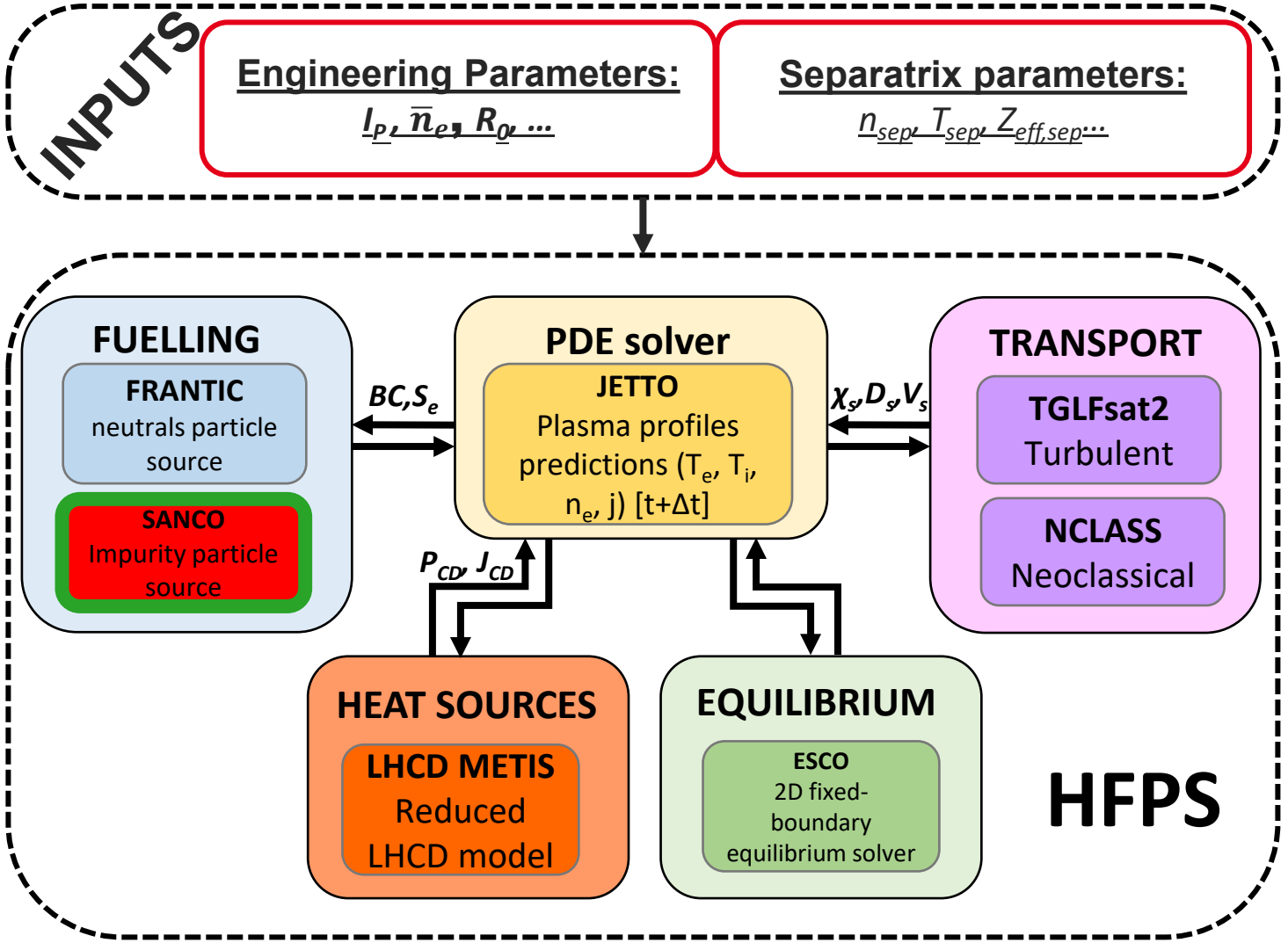


Stepwise approach by adding physics layers

Simulation Name	J Diffusion	Heat Transp.	Particle Transp.	Impurity Transp.
Pred J	✓	✗	✗	✗
Pred JT	✓	✓	✗	✗
Pred JTn	✓	✓	✓	✗
Pred JTnW	✓	✓	✓	✓

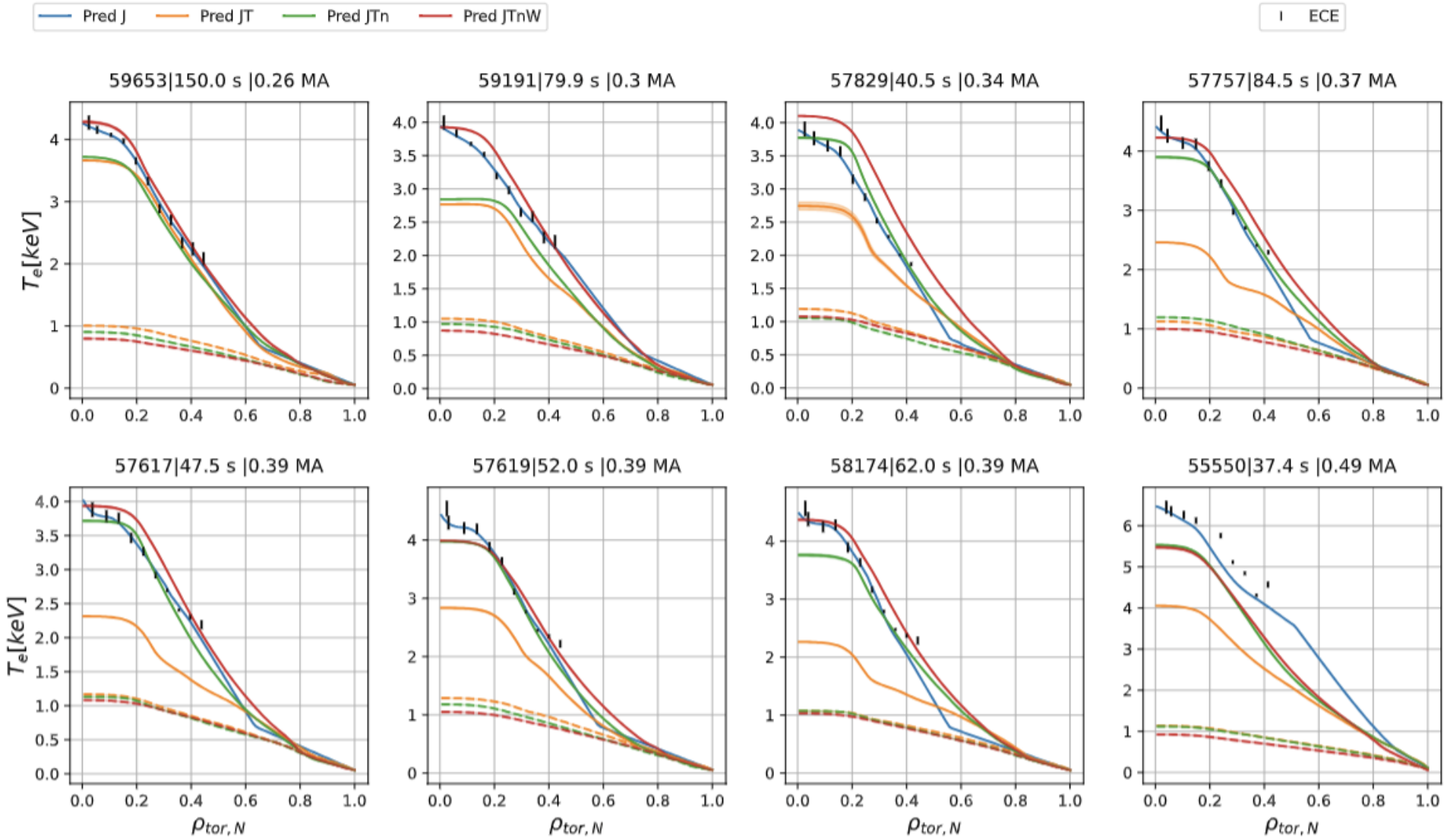
+ N and W impurities transport

Feedback on $n_N(1)$ & $n_W(1)$such that $P_{rad}(t_0)=P_{rad,exp}$, $Z_{eff}(t_0)=Z_{eff,exp}$



Application on WEST

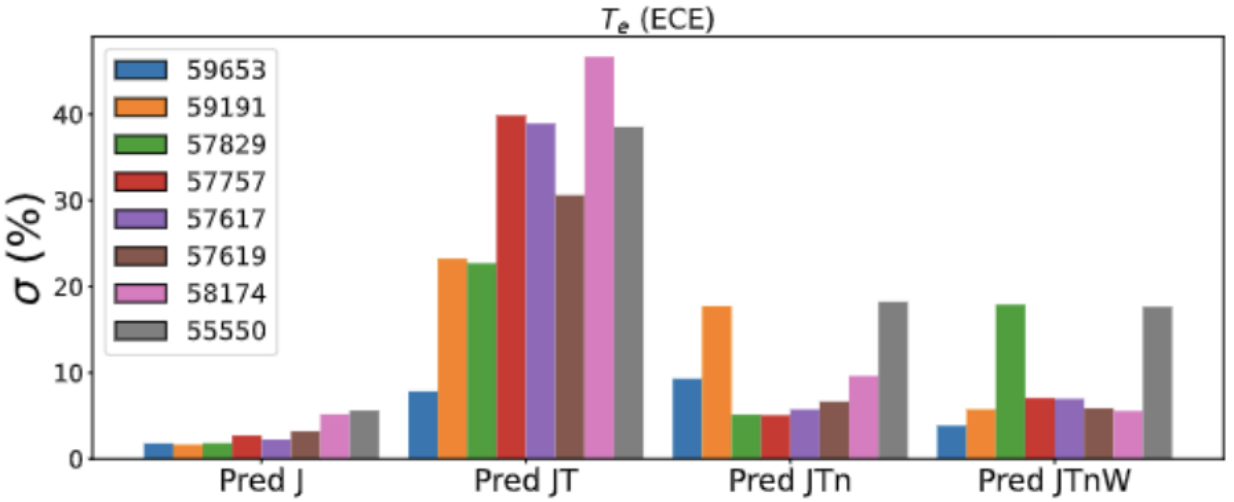
- 4 CPU for Pred J simulations
- 48 CPU when TGLFsat2 is switched on
- 6s of simulation (until that the profiles are converged (q(rho=0); Te(r=0)…))

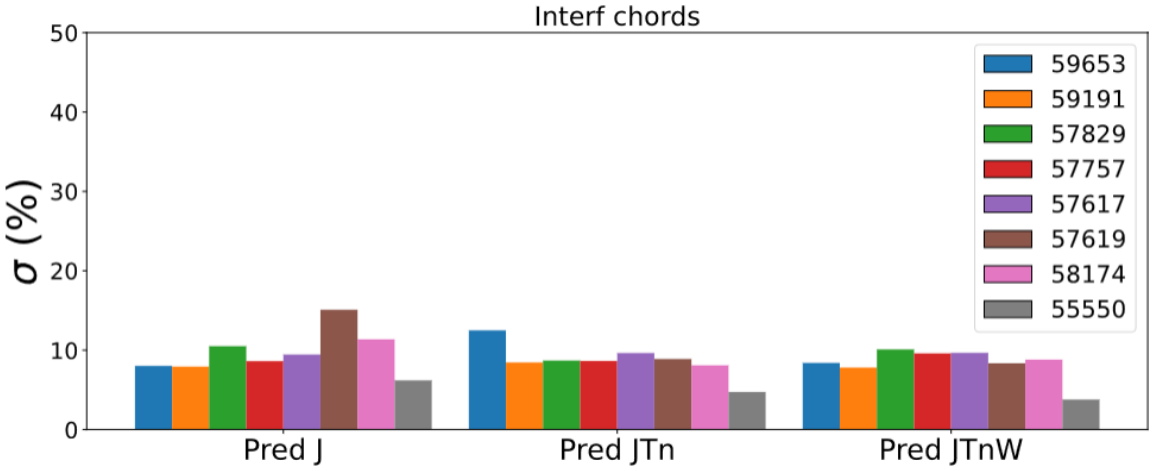


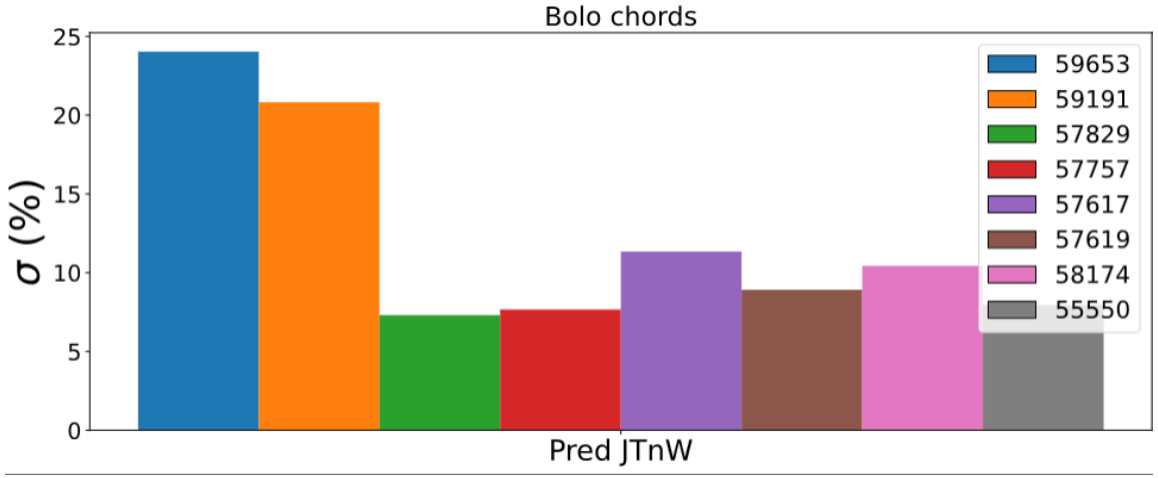
Application on WEST

Diagnostics presented here:
- ECE
- Bolometry LoS
- Inteferometry LoS

$$\sigma = MAPE = \frac{100}{N}\sum\left|\frac{y_{exp} - y_{pred}}{y_{exp}}\right|$$







Outline

1. Large-scale validation challenges: the WEST example
2. **The jetto-tools framework**
3. Python wrappers for jetto-tools to ease the large-scale job submission
4. TGLFNN as a turbulent transport surrogate
5. Proof of principle of the large-scale submission pipeline

Jetto-tools purposes

What is jetto-tools?

- An open-source Python package (also called jetto-pythontools), developed as a community collaboration within the JINTRAC project
- A toolkit to read, manipulate and plot JETTO runs, providing Python classes that parse the standard JETTO output files (jsp, jst, ssp...) and EDGE2D outputs
- A programmatic way to set up runs: starting from a template run, it modifies the JETTO input files (jetto.jset, exfile, configuration) to generate new, ready-to-run input sets
- Used with a plotting GUI (jpyplot) for quick inspection of results

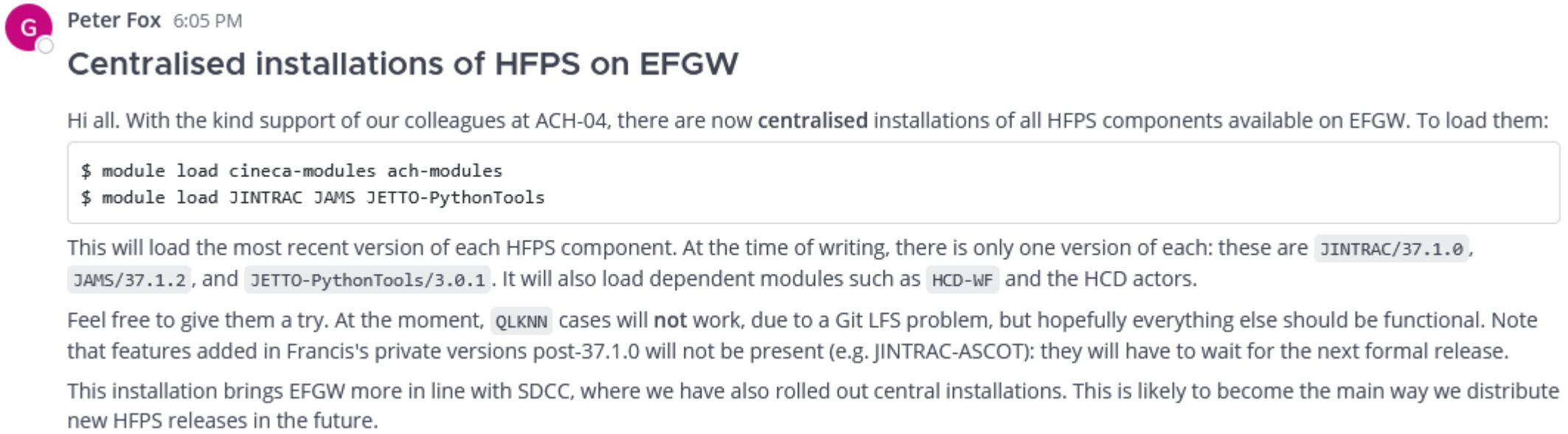
Purpose

- Replace manual, GUI-based run setup (jams) with a scriptable Python interface, making run preparation reproducible and automatable
- Act as the building block for large-scale workflows: because runs can be created and configured from code, the same logic can be looped over many cases (parameter scans, database-wide simulations)
- Provide a common, modular foundation reused by higher-level tools such as duqtools for uncertainty quantification [Azizi 2024 arXiv]

https://pypi.org/project/jetto-tools/
https://jintrac.gitlab.io/jetto-pythontools/

WARNING

- ❑ So far, the jetto-pythontools used in this presentation is located in my own Python virtual environment
- ❑ This version of jetto-pythontools differs from the latest one 3.0.1 → some modifications need to be merged ?
- ❑ Peter has since presented a new centralised installation for the HFPS components (including jetto-pythontools)
- ❑ The scripts shown later sit in a private GitHub repo, and we are in discussion with Slava to share the work
- ❑ We need to unify everything, and agree on a clear plan to implement the features useful for everyone



Peter Fox 6:05 PM

Centralised installations of HFPS on EFGW

Hi all. With the kind support of our colleagues at ACH-04, there are now **centralised** installations of all HFPS components available on EFGW. To load them:

```
$ module load cineca-modules ach-modules
$ module load JINTRAC JAMS JETTO-PythonTools
```

This will load the most recent version of each HFPS component. At the time of writing, there is only one version of each: these are `JINTRAC/37.1.0`, `JAMS/37.1.2`, and `JETTO-PythonTools/3.0.1`. It will also load dependent modules such as `HCD-WF` and the HCD actors.

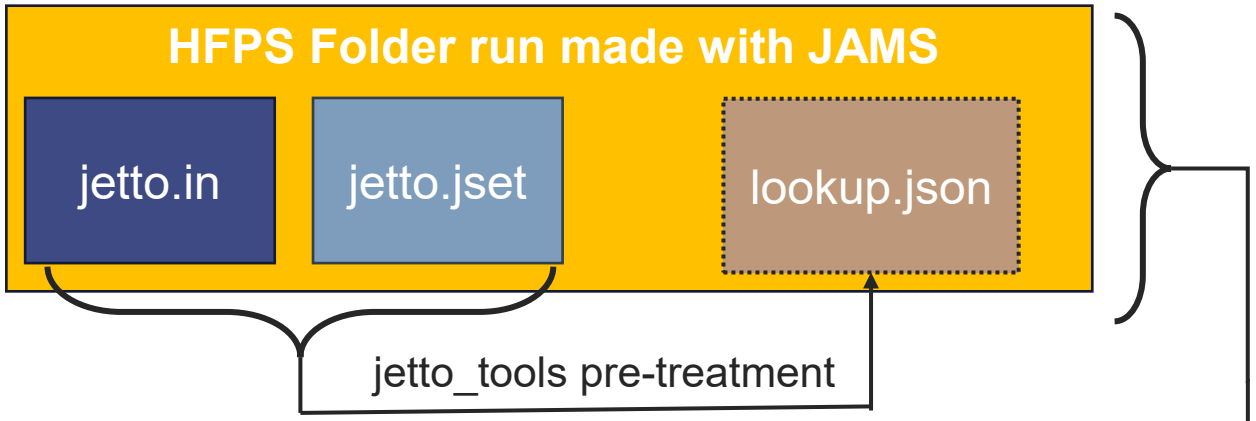
Feel free to give them a try. At the moment, `QLKNN` cases will **not** work, due to a Git LFS problem, but hopefully everything else should be functional. Note that features added in Francis's private versions post-37.1.0 will not be present (e.g. JINTRAC-ASCOT): they will have to wait for the next formal release.

This installation brings EFGW more in line with SDCC, where we have also rolled out central installations. This is likely to become the main way we distribute new HFPS releases in the future.

From a template to a job submission via jetto-tools

```
pip install jetto-tools

import jetto_tools.config
import jetto_tools.job
import jetto_tools.jset
import jetto_tools.template
import jetto_tools.lookup
```




```
map_lookup["nD_sep"] = {"jset_id":
"BoundCondPanel.ionDens.ConstValue",  "nml_id": {"namelist": "NLIST1",
"field": "DNHB1"}, "type": "real", "dimension": "vector"}

map_lookup["DTNEFLFB"] = {"jset_id": None,  "nml_id": {"namelist":
"NLIST4", "field": "DTNEFLFB"}, "type": "real", "dimension": "vector"}
```

Template for jetto_tools run

```
template = jetto_tools.template.from_directory(ref_run_path)
config = jetto_tools.config.RunConfig(template)
config['nD_sep’] = 1e13 # cm^-3
config['DTNEFLFB'] = np.asarray([50, 55]) #[t0, t1]
manager = jetto_tools.job.JobManager()
_ = manager.submit_job_to_batch(config, run_path, run=True)
```

1. *Setup HFPS simu with IDSs pulse*

2. *Setup to perform scan on some parameters*

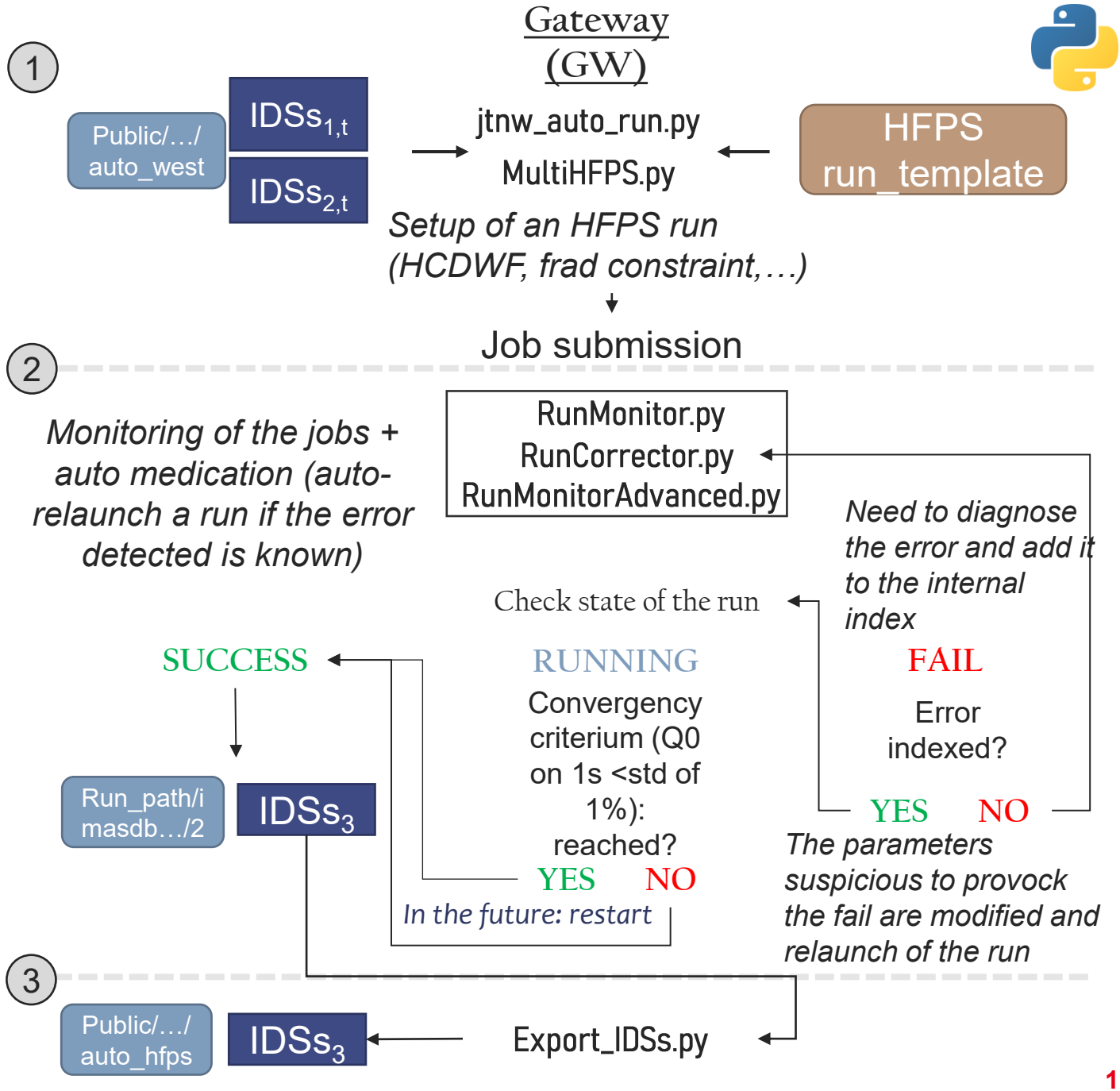
Outline

1. Large-scale validation challenges: the WEST example
2. The jetto-tools framework
3. **Python wrappers for jetto-tools to ease the large-scale job submission**
4. TGLFNN as a turbulent transport surrogate
5. Proof of principle of the large-scale submission pipeline

From an experimental database to a simulated database: python wrappers of jetto-tools

→ The main idea is to restrict the different configuration to specific lookup correponding to different simulation configurations (J, JT, JTn, JTnW, with or w/o HCDWF…). The setup is done in the methods `config_jtnw_setup()` and `lookup_JTnW_creation()` In MultiHFPS.py

→ Scripts on the EFGW: */afs/eufus.eu/user/g/g2evergn/Documents/Python/jetto_tools_extension*



Wrapper 1: MultiHFPS.py

```
multi                       = MultiHFPS(ref_run_path=ref_run_path)
# STEP 1: creation of the config object
config                      = multi.lookup_to_config(force_lookup_creation=True,
lookup_method="JTnW")
# STEP 2: auto-filling of the config object with the IDS + CPU, jintrac version and other
HFPS parameters for the run submission
multi.config_jtnw_setup(config, shot=shot, machine=machine, run=run,
                        user_name=user_name, data_version=data_version,
                        tsimu=tsimu, nsubdi_esco=int(nsubdi_esco*tsimu),
                        walltime=walltime, CPU=CPU,
                        jintrac_version=jintrac_version,
                        frad_constraint=False,
                        jintrac_userid=jintrac_userid)
# STEP 3: run folder creation
multi.run_folder_setup(config, shot=shot, machine=machine, run=run, user_name=user_name,
data_version=data_version, run_path=run_path)
# STEP 4: if needed, modules such as the HCD workflow can be added
multi.hcdwf_setup(config, runs_path=run_path)
# STEP 5: job submission
multi.run(config, runs_path=run_path)
```

Wrapper 2: RunMonitor.py

Was a first attempt to do a tqdm Monitor :

```
mon = RunMonitor()
run = "run_TGLF_3s"
# --- avancement / convergence ---
mon.monitor_bars(run, poll=30.0) # tqdm
bars
mon.diagnose(run) # gives the error
detected if indexed in the catalogue
mon.plot_profiles_binary(run) #
profiles.pdf 2 pages (p1 profils jsp, p2
traces jst)
# --- action ---
mon.stop_run(run) # job scancel
```

Now remplaced by **RunMonitorAdvanced.py** but still important for the ERROR_CATALOGUE indexing the different errors that can be encountered in a failed run folder.

```
ERROR_CATALOGUE: list[dict] = [
{
            "code":       "ESCO_BAD_CONVERGENC
E",
            "family":     "numerical",
            "patterns":   [r"ESCO : BAD
CONVERGENCE FOR CURRENT SCALING ==>
EXIT",      # stable signature
                           r"\(TRY INCREASING
BAVT ->0.98 AND INCREASING NO. OF
ITERATIONS\)"],     # safety net

            "priority":   70,
            "message":    "Early crash because
bad numerical scheme for ESCO convergency.",
            "suggestion": "Modify either the
BAVT to 0.98 or above/ increase the number of
iterations of ESCO call "
        },
    ]
```

Wrapper 3: RunCorrector.py

```
mon    = RunMonitor()
multi =
MultiHFPS(ref_run_path=ref_run_path)
doc    = RunCorrector(mon,
launcher=multi)     # launcher used to
relaunch a run



doc.auto_correct("run_TGLF_3s_a",
shot=57757, machine="west", run=5,
user_name="g2evergn", data_version="3",
tsimu=6.0, nsubdi_esco=nsubdi_esco,
walltime=24, CPU=4,
jintrac_version="37.0.0-rc",
frad_constraint=False,
lookup_method="JTnW", escalate=True) #
relaunch the run with the correction
```

```
CORRECTION_CATALOGUE: dict[str, dict] =
{
        "BAVT_modification": {
            "applies_to":
["ESCO_BAD_CONVERGENCE"],
            "label":      "BAVT",
            "param_key":  "BAVT",
            "access":     "item",
            # -1 -> 0.98 ; values <
0.98 -> 0.98 ; values > 0.98 -> -1
            "corrector": lambda z: (
                0.98 if z == -1
                else 0.999 if z >= 0.98
                else 0.98 if z > 0
                else -1
            ),
        },
}
```

Wrapper 4: RunMonitorAdvanced.py

```
# STEP 1: adding the runs to the reference runs to have a save on the runs created from
this reference run
RunMonitorAdvanced.add_run(ref_run_path, run_name)   # -> ref_run_path/watchlist.txt
# STEP 2: monitor creation
mon = RunMonitorAdvanced()
# STEP 3: doctor creation
doc = RunCorrector(mon, launcher=multi)
# STEP 4: "htop" monitor (co ctrl+c to exit)
mon.monitor_dashboard(
    ref_run_path, poll=10.0, title="Campagne X",
    auto_correct=True, corrector=doc, nsubdi_esco=nsubdi_esco,   # auto-cure
    conv_field="Q0", conv_dt=1.0, conv_target=0.01,              # convergence criterium
    auto_stop=True,                                              # allow automatic stop
    stall_after_s=4*3600, stall_min_sim_s=0.1,                   # STOP if <0.1s simu in 4h
    patterns=["JTnW_test"],                    # STOP if <0.1s simu in 4h
)
```

Wrapper 4: RunMonitorAdvanced.py

TGLFNN MONITOR | Ref path = /pfs/work/g2evergn/jetto/runs/run_Article1/JTnW/57757_ids7_LHCD_6splasma_XiXe_zeff1P1

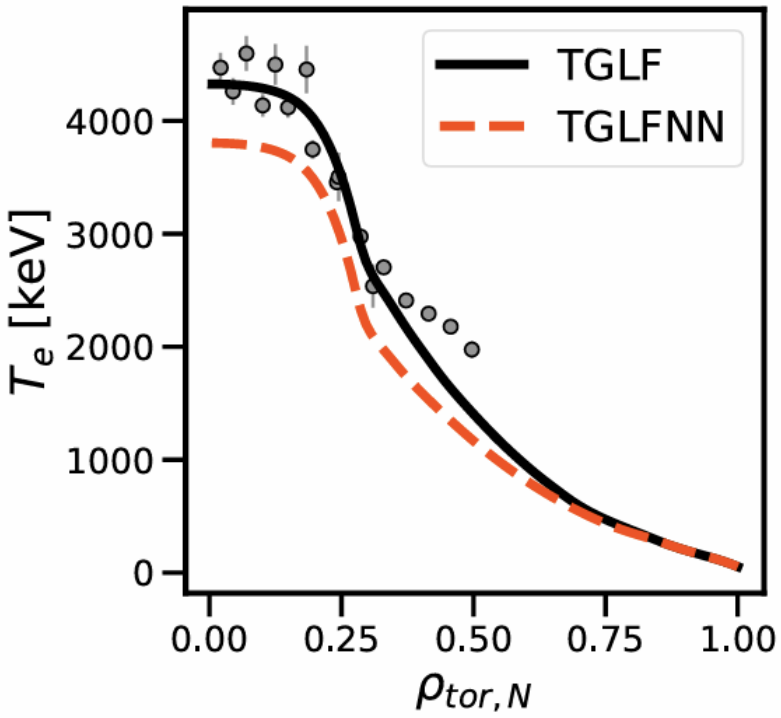
Run	ID	Status	Corr	Sim	Wall	ETA	Conv	Node	CPU
run59653_JTnW_HCDWF	28507	FAILED	0	0% 0.000/6.000s	0% 10:40/48:00:00	--	no jst	r325n002	48
run59653_JTnW_HCDWF_zeff_init_1.32_corr1	30702	RUNNING	1	0% 0.000/6.000s	0% 8:44/48:00:00	6927086:49:04	no jst	r326n005	48
run59191_JTnW_HCDWF	28508	FAILED	0	0% 0.000/6.000s	0% 8:00/48:00:00	--	no jst	r325n002	48
run59191_JTnW_HCDWF_zeff_init_1.21_corr1	30545	RUNNING	1	19% 1.144/6.000s	86% 41:38:50/48:00:00	176:41:27	1.65%	r326n001	48
run57829_JTnW_HCDWF	28509	DONE	0	100% 6.000/6.000s	71% 34:16:33/48:00:00	--	0.66%	r326n006	48
run57757_JTnW_HCDWF	28510	FAILED	0	0% 0.041/6.000s	0% 20:55/48:00:00	--	no jst	r326n006	48
run57757_JTnW_HCDWF_zeff_init_1.05_corr1	30544	CONVERGED	1	39% 2.368/6.000s	83% 39:56:45/48:00:00	--	0.31%	r326n001	48
run57617_JTnW_HCDWF	28511	DONE	0	100% 6.000/6.000s	70% 33:39:33/48:00:00	--	0.61%	r326n006	48
run57619_JTnW_HCDWF	28512	FAILED	0	1% 0.101/6.000s	50% 24:22:27/48:00:00	--	wait	r326n007	48
run57619_JTnW_HCDWF_zeff_init_1.01_corr1	30696	FAILED	1	0% 0.025/6.000s	0% 0:44/48:00:00	--	no jst	r326n005	48
run57619_JTnW_HCDWF_zeff_init_1.01_BAVT_0.98_corr2	30697	RUNNING	2	1% 0.061/6.000s	2% 1:00:48/48:00:00	99:08:22	wait	r326n005	48
run58174_JTnW_HCDWF	28513	WALLTIME	0	55% 3.331/6.000s	100% 48:02:53/48:00:00	--	0.60%	r326n007	48
run55550_JTnW_HCDWF	28514	DONE	0	100% 6.000/6.000s	67% 32:31:13/48:00:00	--	0.94%	r326n007	48

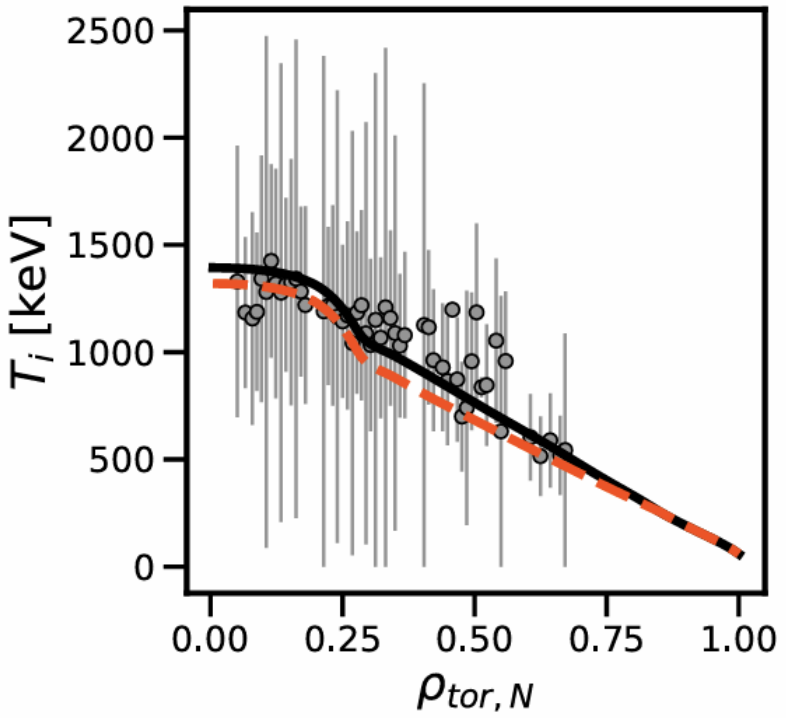
Outline

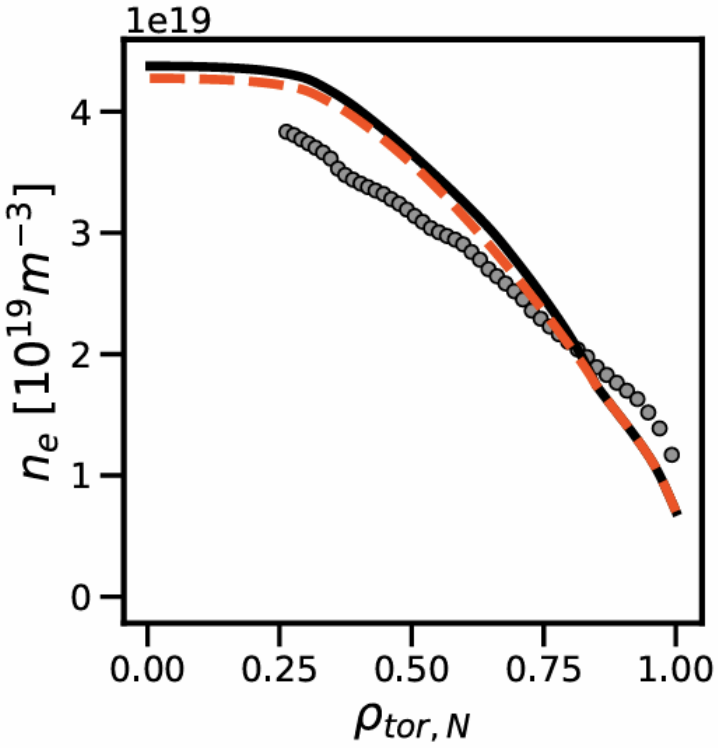
1. Large-scale validation challenges: the WEST example
2. The jetto-tools framework
3. Python wrappers for jetto-tools to ease the large-scale job submission
4. **TGLFNN as a turbulent transport surrogate**
5. Proof of principle of the large-scale submission pipeline

TGLFNN: application on WEST

- ITRDOWNMULT = 30
- BCRINTRHON = 0.85
- RHOMIN = 0.3
- RHOMAX = 0.85
- X_i/e_on_axis = 5E3 cm²/s
- D _on_axis = 5E3 cm²/s
- ITRTGLFFLSP=2
- ZEFF=1



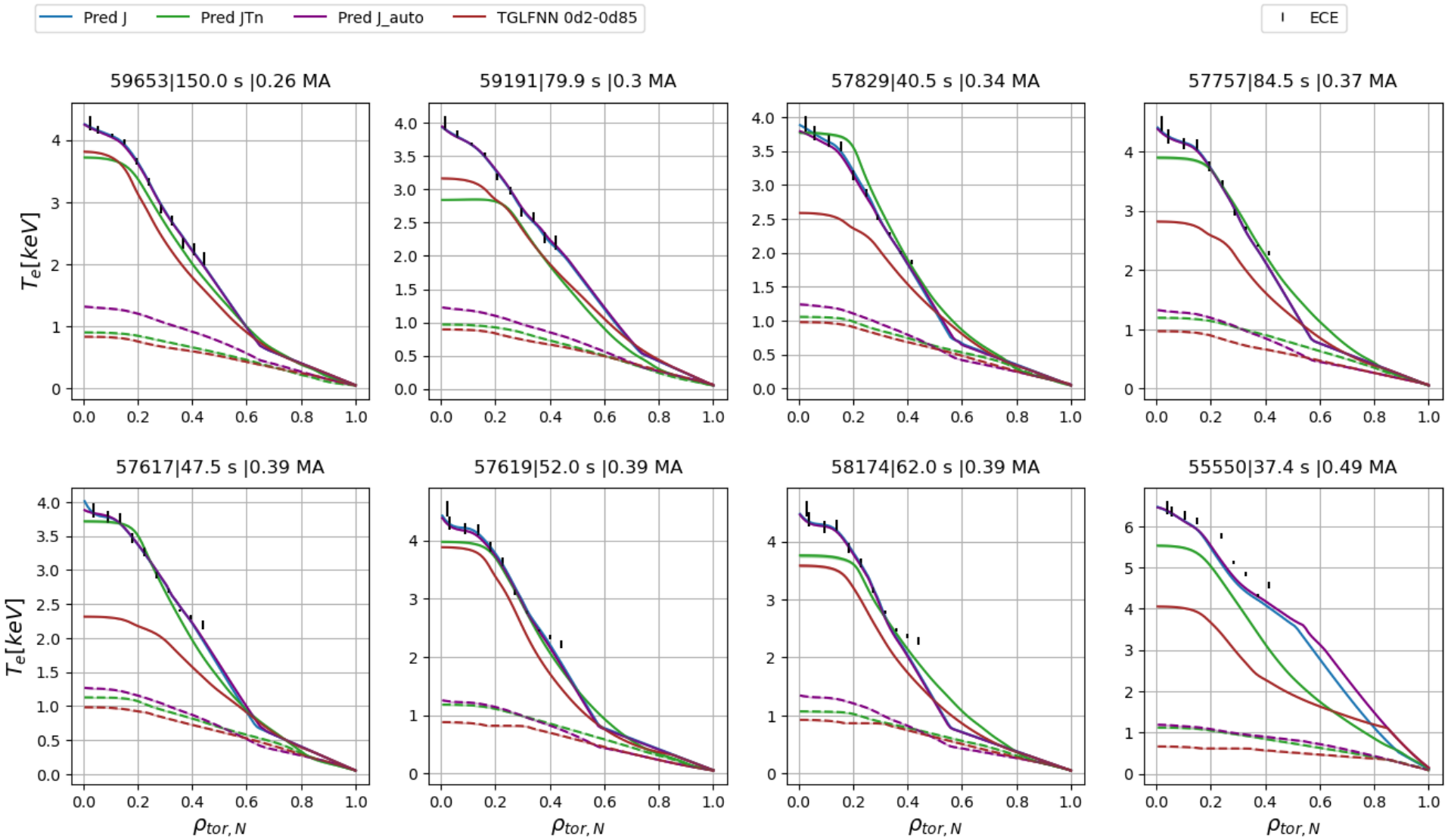




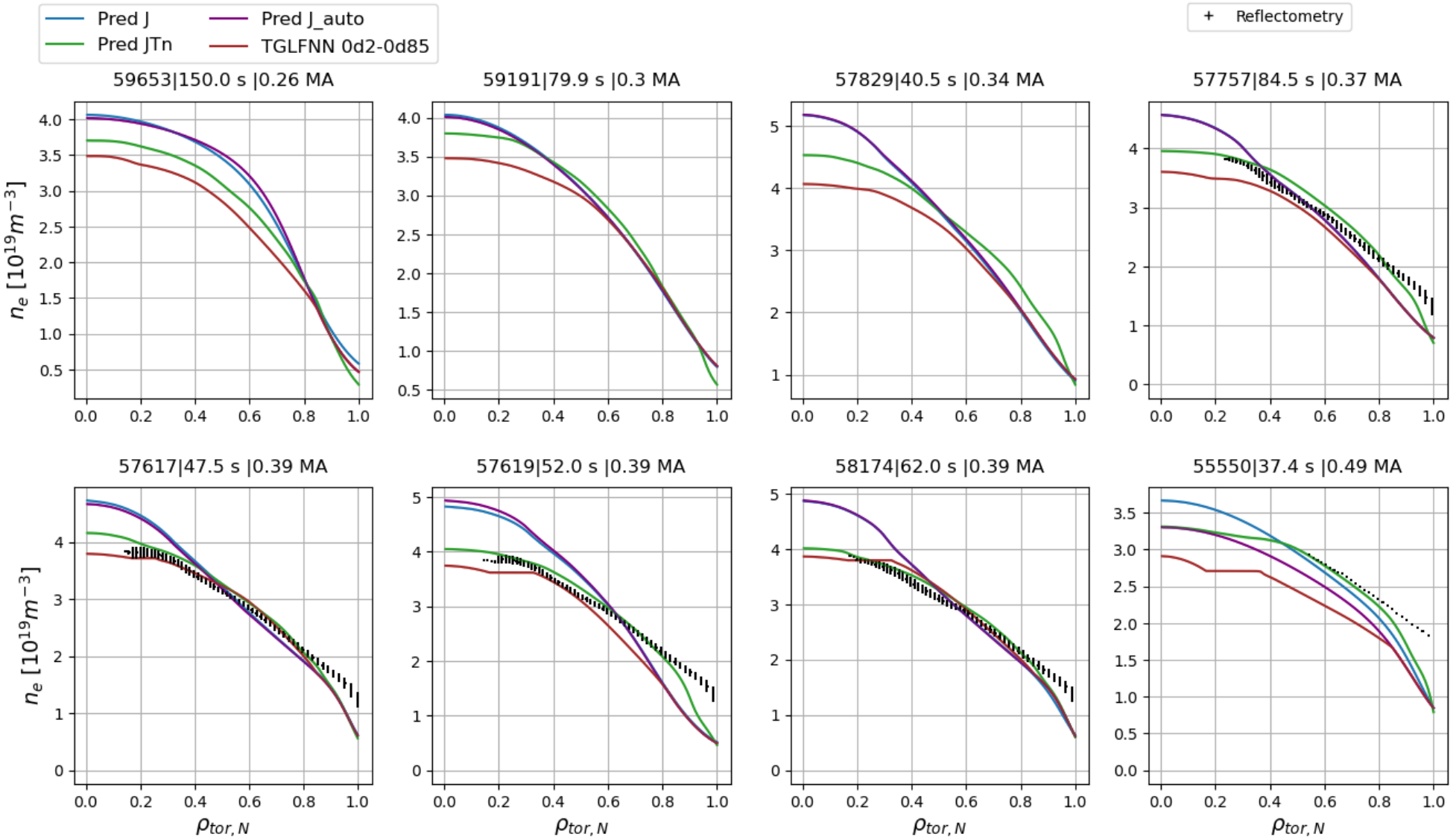
Outline

1. Large-scale validation challenges: the WEST example
2. The jetto-tools framework
3. Python wrappers for jetto-tools to ease the large-scale job submission
4. TGLFNN as a turbulent transport surrogate
5. **Proof of principle of the large-scale submission pipeline**

Application of the jetto-tools python wrappers on WEST: verification+TGLFNN

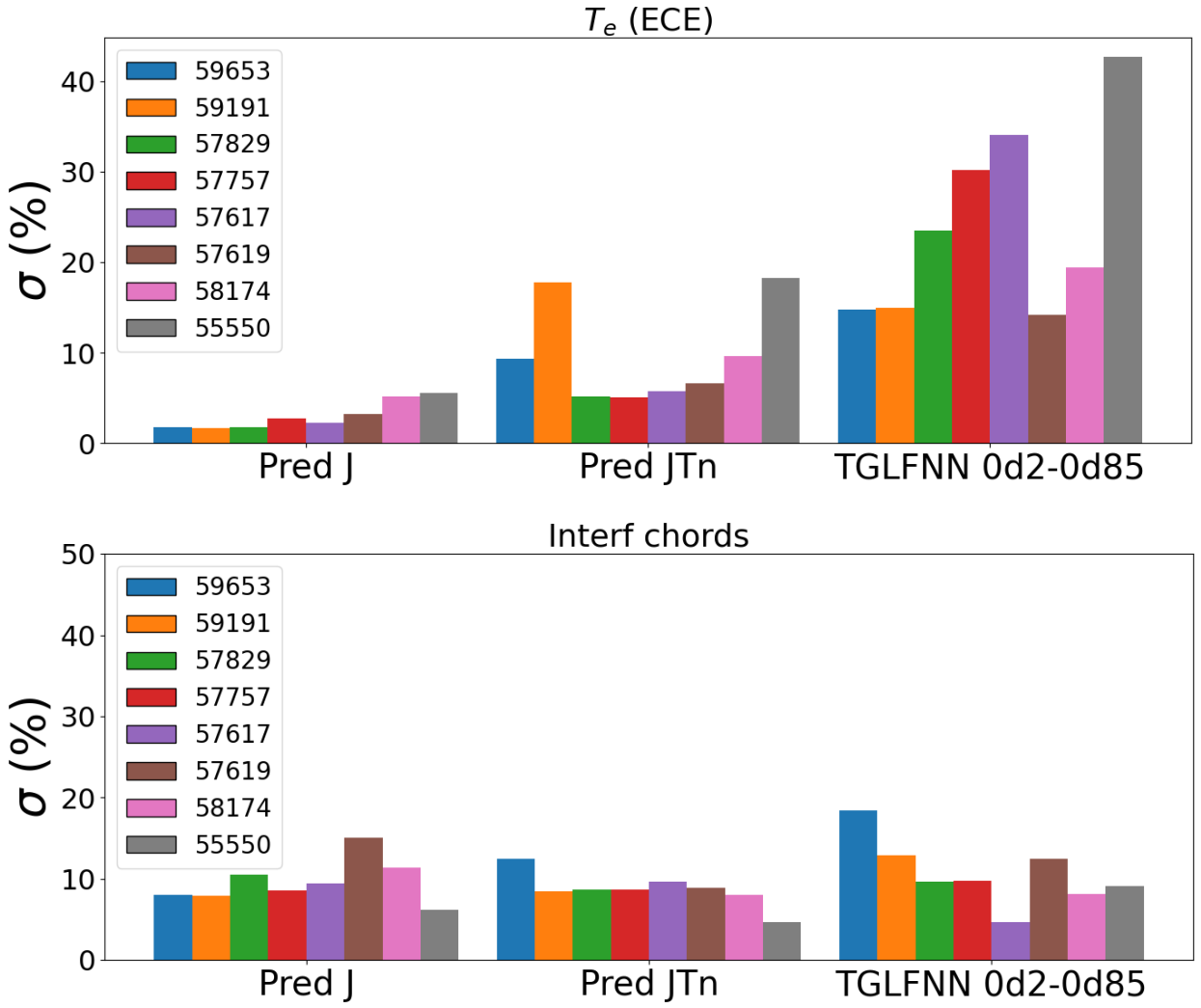


Application of the jetto-tools python wrappers on WEST: verification+TGLFNN

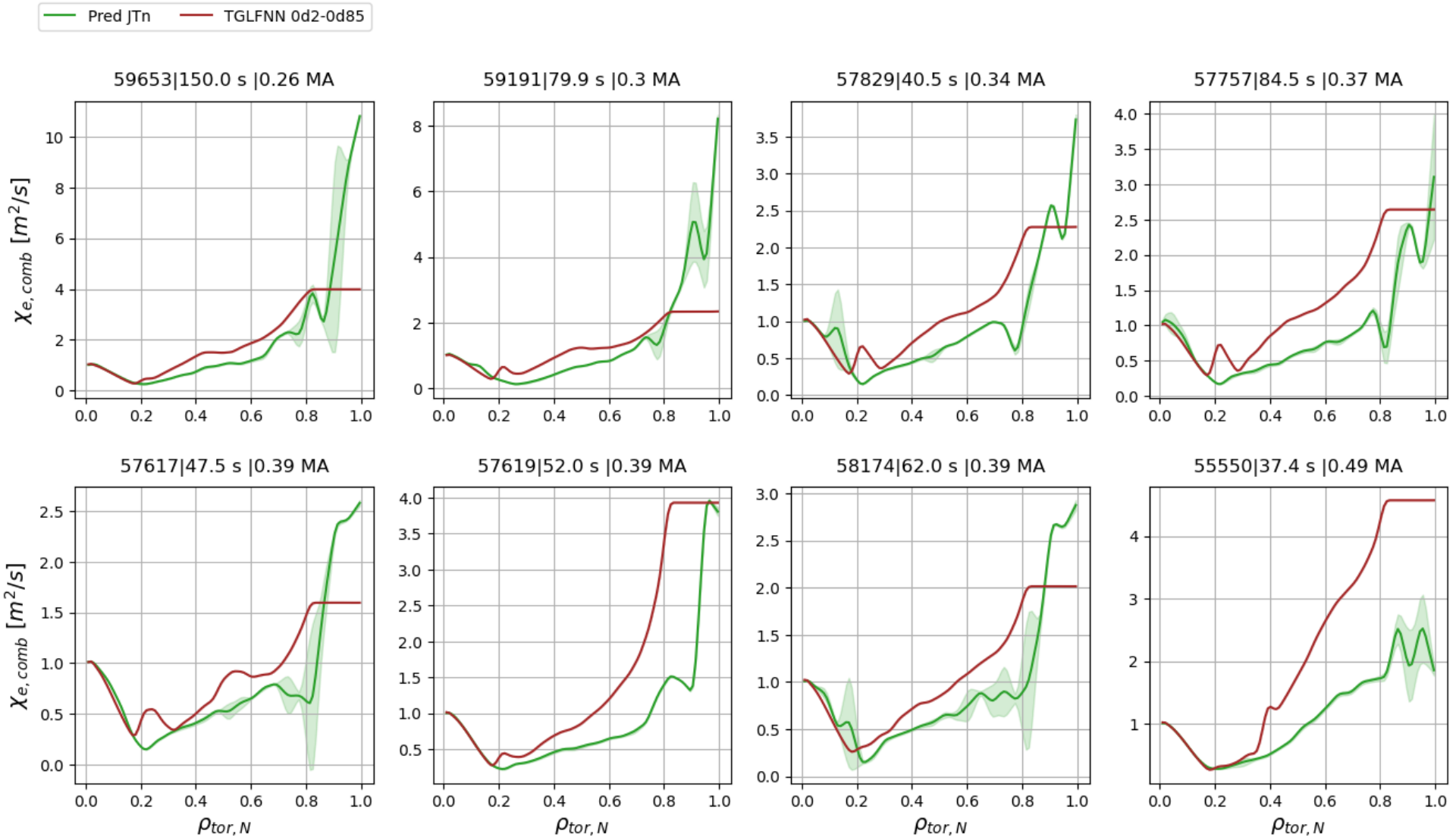


Application of the jetto-tools python wrappers on WEST: verification+TGLFNN

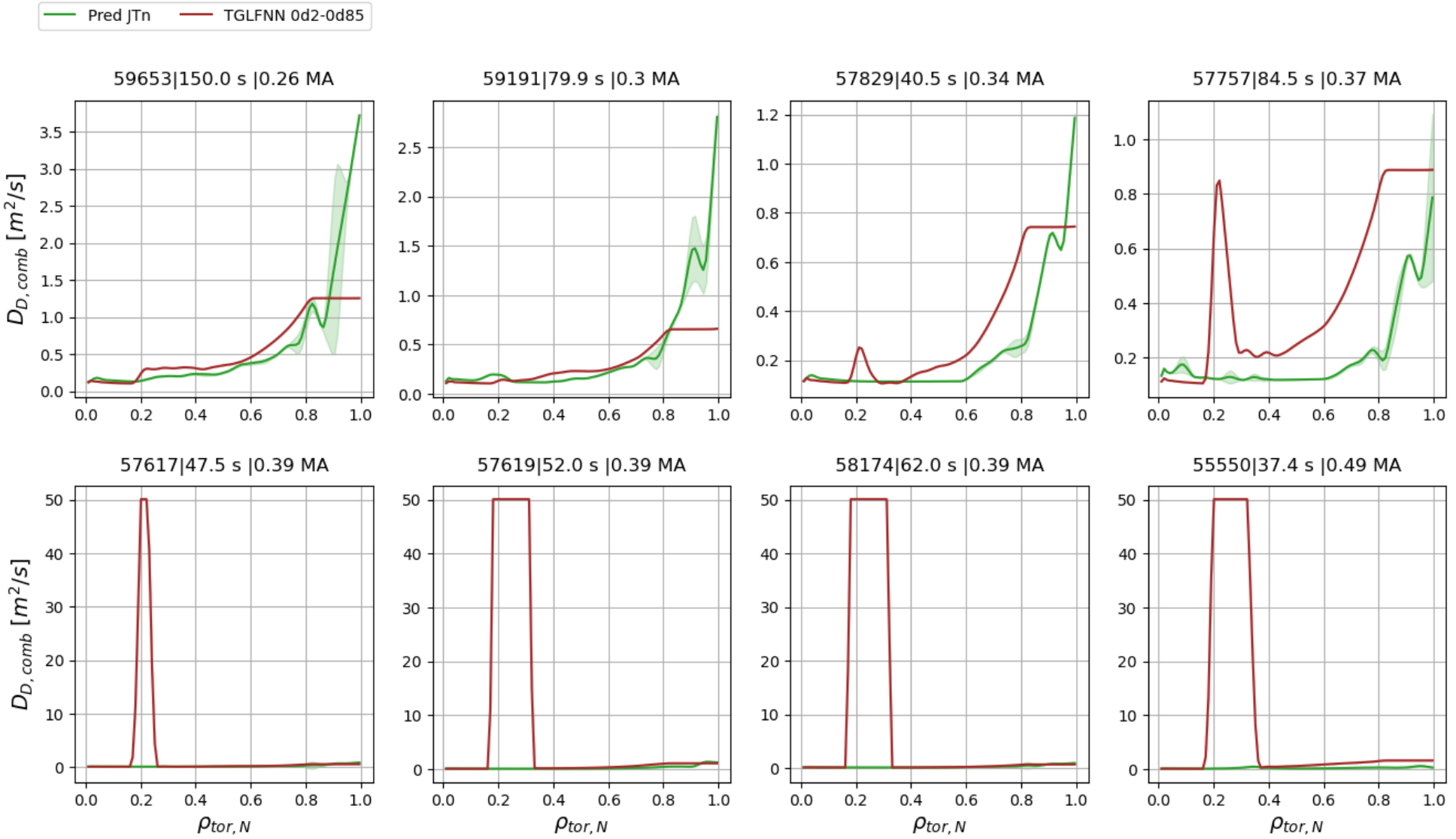
$$\sigma = MAPE = \frac{100}{N}\sum\left|\frac{y_{exp} - y_{pred}}{y_{exp}}\right|$$



Application of the jetto-tools python wrappers on WEST: TGLFNN



Application of the jetto-tools python wrappers on WEST: TGLFNN



Summary & Perspectives

1.) On the jetto-tools wrappers side:

- ❑ Couple these codes with Slava's work
- ❑ Add a restart option
- ❑ Create a "universal lookup" to expose more modifiable options + details such as type checking and blank-field handling still need to be simplified within jetto-tools

2.) On the WEST large-scale validation side:

- ❑ Select hundreds of pulses spanning a range of line-averaged electron densities and total injected powers
- ❑ Run with TGLFNN, restricted to rho_tor_norm = 0.2-0.85 and without impurities
- ❑ Run with TGLF-SAT2 using the JTn configuration
- ❑ Perform statistical analysis to validate (or not) the HFPS with these two configurations
- ❑ TGLF-SAT2 + SANCO? (more challenging)