MAX-PLANCK-INSTITUT
FÜR PLASMAPHYSIK

Using the IMAS Persistent Actor Framework for implementing the European Transport Simulator

David Coster

EUROfusion

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Outline

- Why IMAS?
- What can the Data Management Plan provide?
- What is the Persistent Actor Framework?
- What is the European Transport Simulator?
- What can we do with the ETS-PAF?

What is IMAS

IMAS is the Integrated Modelling and Analysis Suite being developed at ITER and uses the Interface Data Structure (IDS) as the basic unit of data interchange or storage.

Versions exist for various physics “objects”:

- equilibrium
- core_profiles
- summary
- ec_launchers
- camera_visible

This provides a standard way of storing and accessing tokamak and stellarator data with a well defined ontology

Ref: https://sharepoint.iter.org/departments/POP/CM/IMDesign/Data%20Model/sphinx/4.0/index.html

What can the Data Management Plan provide?

The Data Management Plan is a project within EUROfusion [https://wiki.euro-fusion.org/wiki/DMP]

The project aims at **establishing the team of computer and data experts for implementing the Data Management Plan (DMP)** as it was agreed by the 40th EUROfusion General Assembly.

The design and main elements of the proposed infrastructure are described in the **DMP blueprint**.

The data management plan defines 4 scenarios of increasing ambition in terms of data access and F.A.I.R.ness (Findable, Accessible, Interoperable and Reusable):

- **Scenario A**: making metadata only available and searchable using IMAS data subsets for interoperable definitions of quantities [F,(I)];
- **Scenario B**: adds to Scenario A by allowing a subset of the data to be accessed using common tools (for example UDA). Facilities are responsible for the access level and qualification of data through the data mappings [F,A,I,(R)];
- **Scenario C**: builds on the previous stages and allows for enhanced data provenance and referencing through PID's [F,A,I,R].
- **Scenario D**: adds a lightweight layer for open access to non-embargoed metadata and where allowed by the facilities also data access for export in human readable formats (CSV files) [F,A,I,R] and open.

It is decided to implement **in 2023-2025 the scenario A fully** and to **prepare for the implementation of scenario B (prototype level)**. The infrastructure should be built on the Fair for Fusion provided software tools and developed practices and it is expected that the portal interface and development activities will be hosted on the Gateway where additional virtual machines will be made available for the effort. The documentation, software and ticketing tools should be integrated with the existing PSNC based gitlab and Jira services.

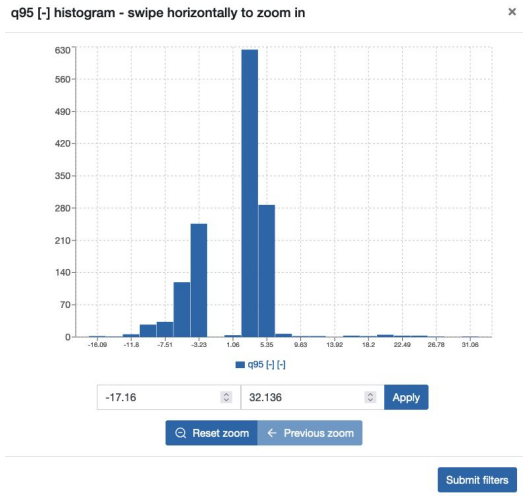
Further development will depend on the outcome of these initially implemented activities and will be decided at a later stage.

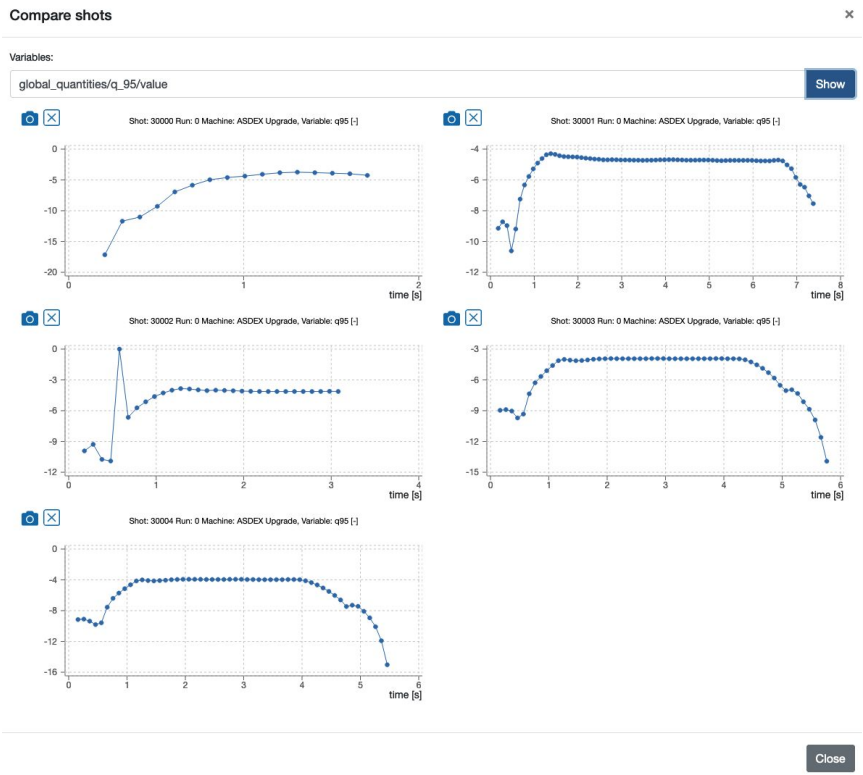
DMP: Scenario A

https://dmp.eufus.psnc.pl/dashboard/

Prototype – full system with “summary” data from AUG, TCV, WEST, JET, MAST(-U) on the new Gateway (whenever it arrives)

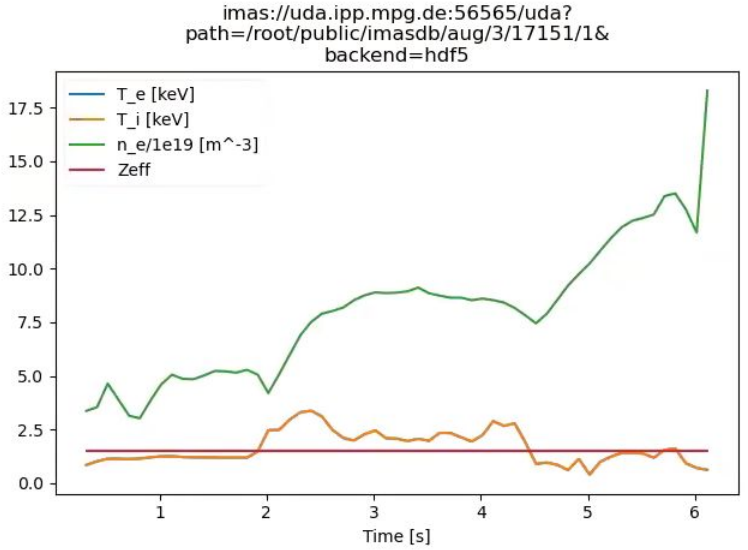
Can be used to search for data with desired properties …

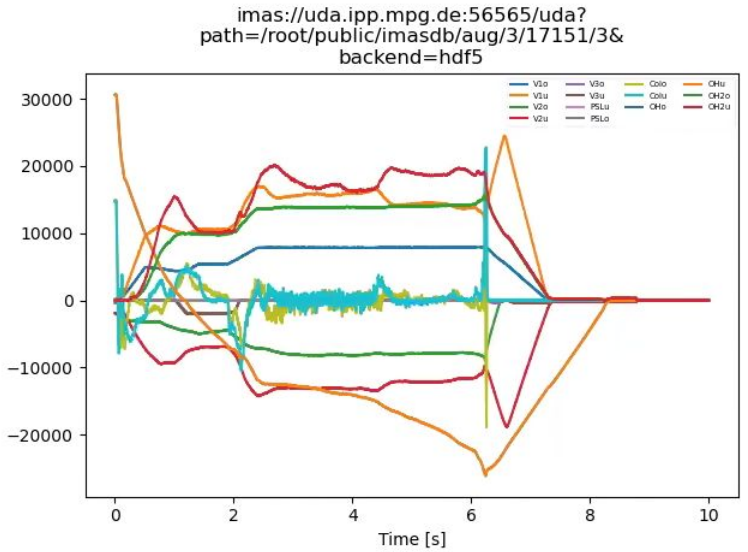




DMP: Scenario B

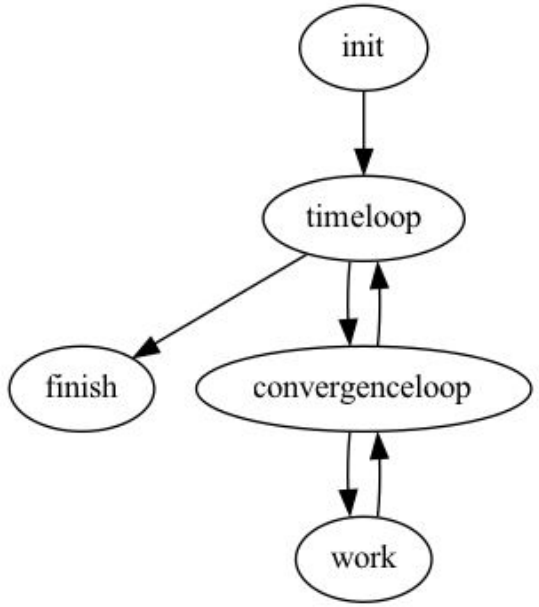
- Provide access to data from the experiments
  - **Processed data: core_profiles, equilibrium, …**
  - **Diagnostic data**
- For AUG data
  - **Processed data**
    - **Run “trview” from Giovanni Tardini**
    - **I transfer the produced IDS HDF5 files to “uda.ipp.mpg.de”**
    - **Data can then be accessed from outside**
      - Implementation of “Authentication and Authorization” under development
    - **'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/17151/1&backend=hdf5' provides ['core_profiles', 'dataset_description', 'equilibrium', 'ic_antennas', 'nbi', 'pulse_schedule', 'summary', 'tf', 'wall']**
  - **Diagnostic data**
    - **Produce IDSes using various codes**
    - **I transfer the produced IDS HDF5 files to “uda.ipp.mpg.de”**
    - **Will move to dynamic UDA later**
    - **'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/17151/3&backend=hdf5' provides ['equilibrium', 'magnetics', 'pf_active', 'tf', 'wall']**





What is the Persistent Actor Framework?

- A concept from ITER based on the MUSCLE3 framework
  - **“The third major version of the MUltiScale Coupling Library and Environment”**
  - **https://github.com/multiscale/muscle3 (open source)**
- While MUSCLE3 allows for a range of data to be exchanged between “actors”, the Persistent Actor Framework restricts the data to be exchanged to be IDSes
- Allows one to easily build a workflow based on communicating independent programs (“actors”)
  - **Supports actors in fortran, c++, python**
- The iWrap tool is available to prepare existing fortran or c++ codes for use in PAF workflows (though this can also be done manually)
- The actors can remain alive through the whole workflow
  - **Can remember previous state**



What is the ETS?

- Started with a fortran only workflow using CPOs
  - **Implemented MMS**
- Implemented the full workflow in Kepler (using CPOs)
  - **Still used ETS-5**
- Re-implemented the full Kepler workflow to use IDSs
  - **Production ETS-6**
- Two, in development, versions using Python
  - **Object oriented ETS workflow (ETSpy) (Rui Coelho)**
  - **ETS-PAF, using MUSCLE3 / Persistent Actor Framework (David Coster)**

Goals of the Python developments

- Need to move away from Kepler
  - **Lack of support**
  - **Wasn’t really compatible with HPC (no or old versions of Java, …)**
- To re-use the physics components of the Kepler ETS
  - **Moved from FC2K wrapped actors to iWrap wrapped actors**
    - **Some logic changes were necessary where non-IDSs had been used**
- Two complementary approaches
  - **Pure python (all logic in the python code, direct call of the wrapped physics routines)**
  - **Persistent Actor Framework**
    - **Workflow consists of communicating programs**
      - Logic in the interconnections and some glue python actors

Development

- Also a pure python approach is being pursued by Rui Coelho
  - **Object oriented ETS workflow (ETSpy)**
- ETS-PAF
  - **Uses MUSCLE3 (**https://muscle3.readthedocs.io/en/latest/, https://github.com/multiscale/muscle3**)**
- **Not** a direct re-implementation of the Kepler workflow
  - **Started by building physics-less workflows to test ideas and see what was possible**
  - **Then built up more data centric workflows**
    - **To test timings**
    - **To test physics actors**
  - **Then added more physics**

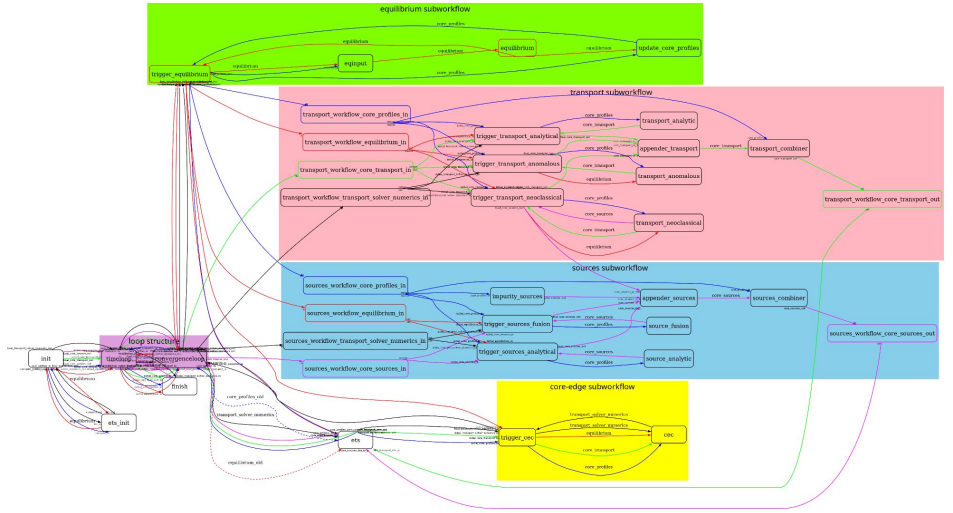
Running an AUG case: grab the data using UDA

```
costerd@sdcc-login02 uda-test]$ time idscp -s
'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/32408/1&backend=hd
f5' -d "imas:hdf5?user=costerd;pulse=32408;run=1;database=aug;version=3" -a
ERROR:root:b'al_begin_dataentry_action: [ALBackendException = HDF5 master file not
found: /home/ITER/costerd/public/imasdb/aug/3/32408/1/master.h5]'
ERROR:root:b'al_begin_dataentry_action: [ALBackendException = HDF5 master file not
found: /home/ITER/costerd/public/imasdb/aug/3/32408/1/master.h5]'
[11/18/24 15:39:35] WARNING  Destination pulse does not exist. Creating.
idscp:99
WARNING:module:Destination pulse does not exist. Creating.
Copying core_profiles
Copying dataset_description
Copying ec_launchers
Copying ece
Copying equilibrium
Copying ic_antennas
Copying nbi
Copying pulse_schedule
Copying summary
Copying tf
Copying wall
5.390u 6.406s 9:56.97 1.9%   0+0k 0+0io 0pf+0w
```

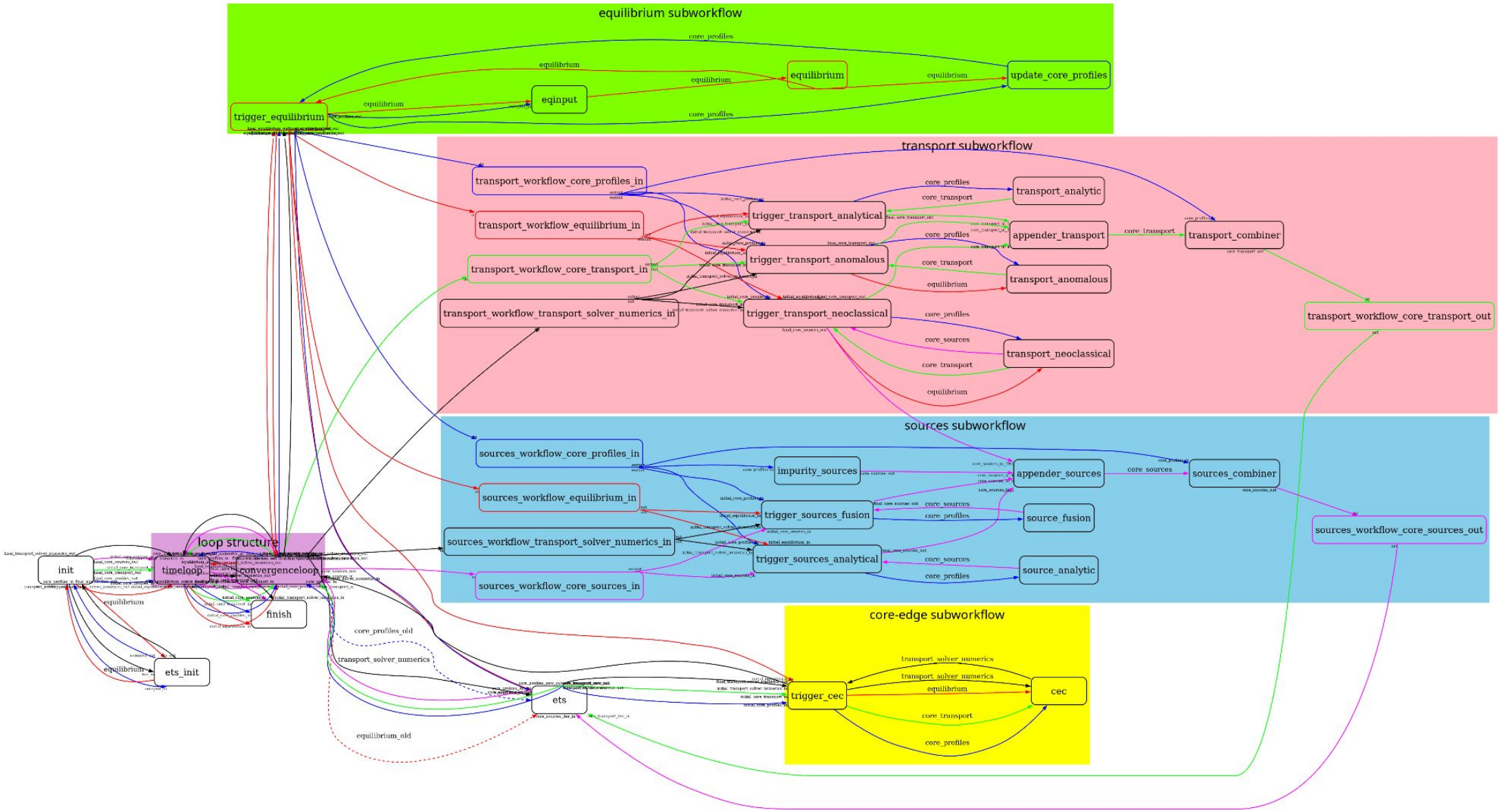
Using rsync to copy the HDF5 files (18 MB) from ITER to Garching took 5.54 seconds!

Running an AUG case

- Grab the data using UDA (similar to previous slide except that we will use a previously copied case for AUG 17151 – I did not save the timing information)
- Run muscle
  - muscle_manager --start-all
    - workflows/transport_subworkflow_encapsulated.ymmsl
    - workflows/sources_subworkflow_encapsulated.ymmsl
    - workflows/ets_main_encapsulated_with_ets_init.ymmsl
    - implementations/transport_anomalous_tcibgb.ymmsl
    - settings/aug_BGB.ymmsl



Running an AUG case



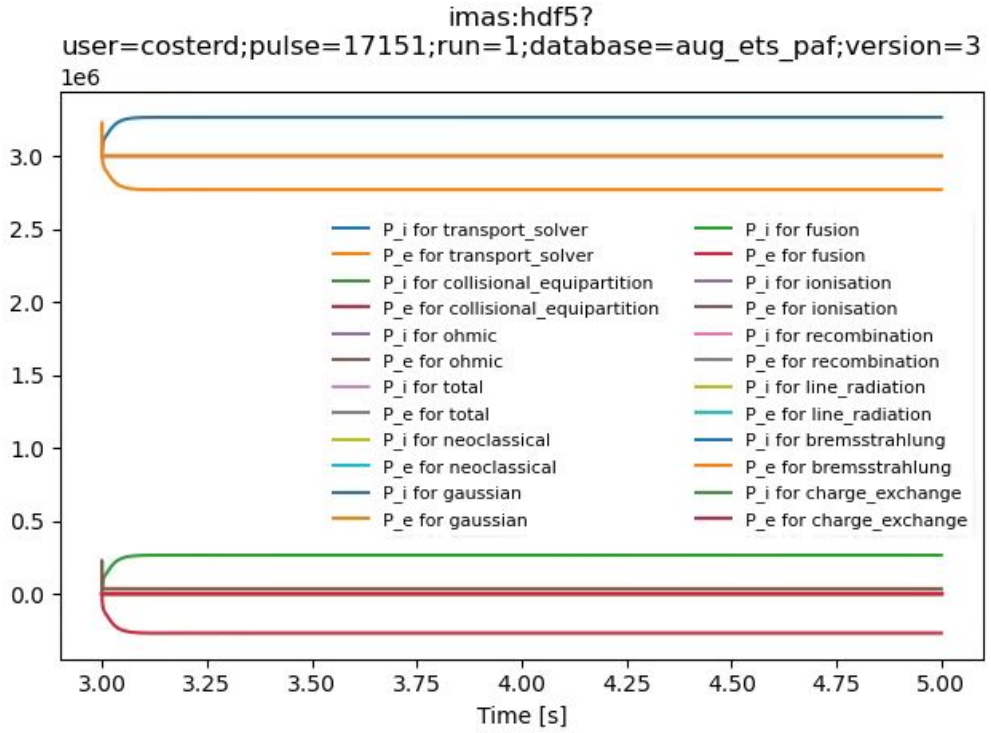
First attempt at starting from UDA (AUG/17151)

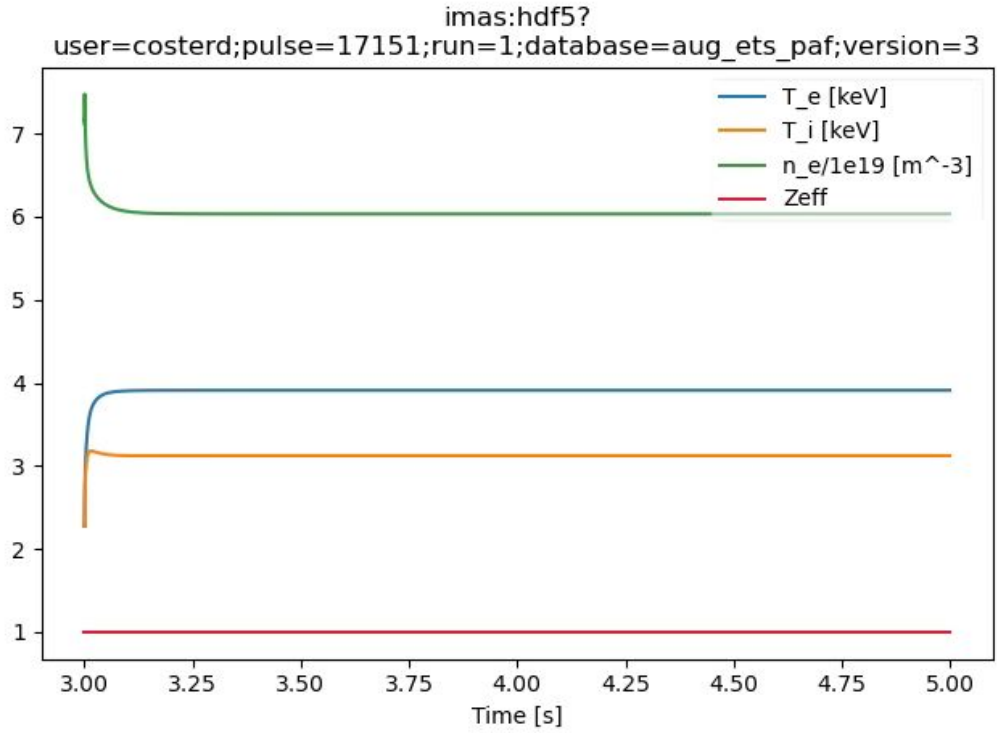
- idscp -s 'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/17151/1&backend=hdf5' -d 'imas:hdf5?path=db.tmp/aug/3/17151/1' -a
- rsync -avP db.tmp/aug/3/17151 ~/public/imasdb/aug/3/
- muscle_manager --start-all workflows/transport_subworkflow_encapsulated.ymmsl workflows/sources_subworkflow_encapsulated.ymmsl workflows/ets_main_encapsulated_with_ets_init.ymmsl implementations/transport_anomalous_tcibgb.ymmsl settings/aug_BGB.ymmsl

- HDF5_BACKEND/aug_ets_paf/17151/1 run_ETS_MAIN_ENCAPSULATED_WITH_ETS_INIT_20240912_165718
- ErrTol=0.001 amix_transport=
- 3.000001 3.33467 5.0 1e-06 0.0294118 0.1
- ResourceUsage = 32.692 3.605 415.695 581652
- 135 iterations
- 68 time-steps

First attempt at starting from UDA (AUG/17151)

- Used dummy sources rather than the real ones
  - **Gaussian**
    - **Particles: 1.000000e+21 / second**
    - **Electron energy: 3.000000e+06 W**
    - **Ions energy: 3.000000e+06 W**





Some other parts of ETS-PAF

- Source available at https://gitlab.eufus.psnc.pl/ets/ets_paf
- Documentation generated by pipeline on a push
  - **https://ets.pages.eufus.psnc.pl/ets_paf/**
- A single (currently) non-regression test is also run on a push
  - Comparing Data Entries... ━━ 100% 0:00:00
  - Comparisen between
  - imas:hdf5?path=test_data/ets.ymmsl
  - imas:hdf5?path=run_ETS_20241205_171527/instances/finish/workdir
  - ┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┓
  - ┃ ids field name ┃ Comment ┃
  - ┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━┩
  - └──────────────────┴─────────┘
  - [12/05/24 17:16:51] INFO Writing to html file idsdiff:52
  - :idsdiff_reports/20241205_171651_compare
  - _report.html
  - Segmentation fault (core dumped)
  - The test tests/01_ets passed
- Have also run cases with
  - **W (all charge states)**
  - **Stiff transport models (mixing of transport coefficients)**

Experience gained during this effort

- MUSCLE3 provides a very convenient framework for developing workflows
  - **Separate stdout/stderr in files for each component**
  - **Easy to set up test workflows before adding to the main workflow**
    - **Important to do this**
  - **Pretty clean way of building up the workflow on the command line**
    - **muscle_manager --start-all workflows/transport_subworkflow_encapsulated.ymmsl workflows/sources_subworkflow_encapsulated.ymmsl workflows/ets_main_encapsulated.ymmsl implementations/transport_anomalous_tcibgb.ymmsl settings/jet.ymmsl**
  - **There is a tool still in development for visualizing workflows**
    - **I'm not sure how helpful a GUI would be to develop workflows**
- What would be nice to have
  - **A better model for encapsulation**
    - **The so far implemented "encapsulation" is *ad hoc* and implemented by the workflow designer**
    - **Will need something better when we incorporate the ETS workflow into an UQ workflow**

Plans

- Continue development of the two versions
  - **Benchmark them against ETS-6 and other transport codes using IDSes**
    - **ETS-6 has a non-regression set of test of increasing complexity**
  - **Add more physics**
    - **HCD**
  - **Look to setting up desired cases using the autoGUI or its successor**
  - **Build the user-base**
  - **Look at making doing UQ easier**
- Look to choose (at some point) whether we want to maintain both of the new ETSes
- ETS-PAF might morph into ??? at ITER

Acknowledgements

- Thanks to the ETS working group
  - **Pär Strand, Rui Coelho, Dimitry Yadikin, Thomas Jonsson, Francesca Poli**
    - **Some funded as part of an ACH activity**
  - **A very big thank you to Chalmers University who have paid for face-to-face working sessions**

Backup slides

Timing tests (serialization and deserialization)

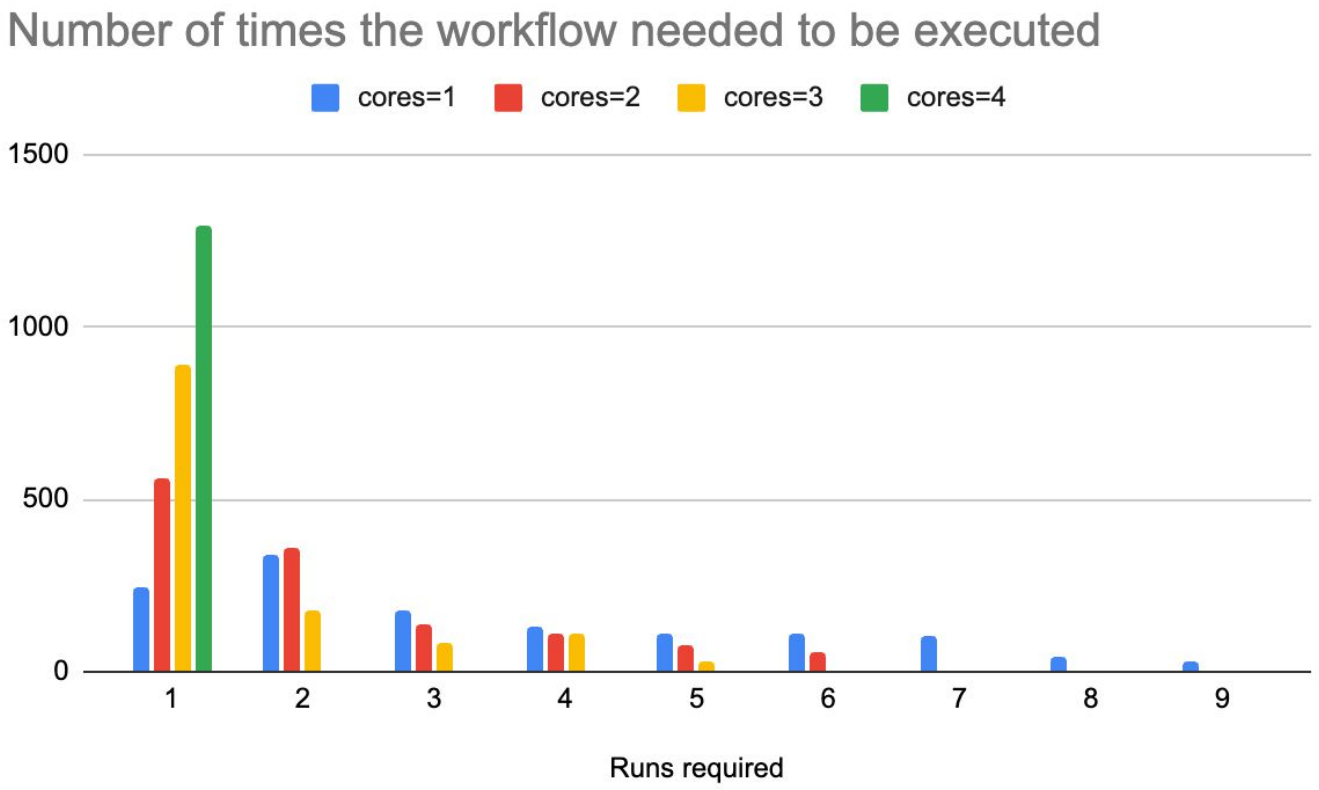
	ASCII_SERIALIZER_PROTOCOL			FLEXBUFFERS_SERIALIZER_PROTOCOL	
	run_TIMER_SERIALIZATION_20240828_104208			run_TIMER_SERIALIZATION_20240828_105408	
IDS	**Serialization / Deserialization time (s)**		**Ratio**	**Serialization / Deserialization time (s)**	
	average	**stddev**	**FLEXBUFFERS / ASCII**	**average**	**stddev**
equilibrium	0.988	0.014	2.1%	0.020	0.001
core_profiles	0.022	0.001	34.8%	0.008	0.000
core_transport	0.051	0.001	39.6%	0.020	0.001
core_sources	0.025	0.001	39.9%	0.010	0.000
transport_solver_numerics	0.028	0.002	33.7%	0.010	0.000

- PAF relies on serialization / deserialization to transfer IDSes
- Implementation of a new serializer resulted in a substantial speed-up for larger IDSes and a smaller but still significant speed-up for smaller IDSes

Stress testing MUSCLE3: running ETS-PAF on a few cores

	Number of cases			
Runs required	cores=1	cores=2	cores=3	cores=4
1	247	558	891	1296
2	340	357	176	
3	179	137	87	
4	133	109	110	
5	110	75	32	
6	108	55		
7	105	5		
8	41			
9	33			
Total required	4554	2859	2104	1296
Ratio required	3.51	2.21	1.62	1.00

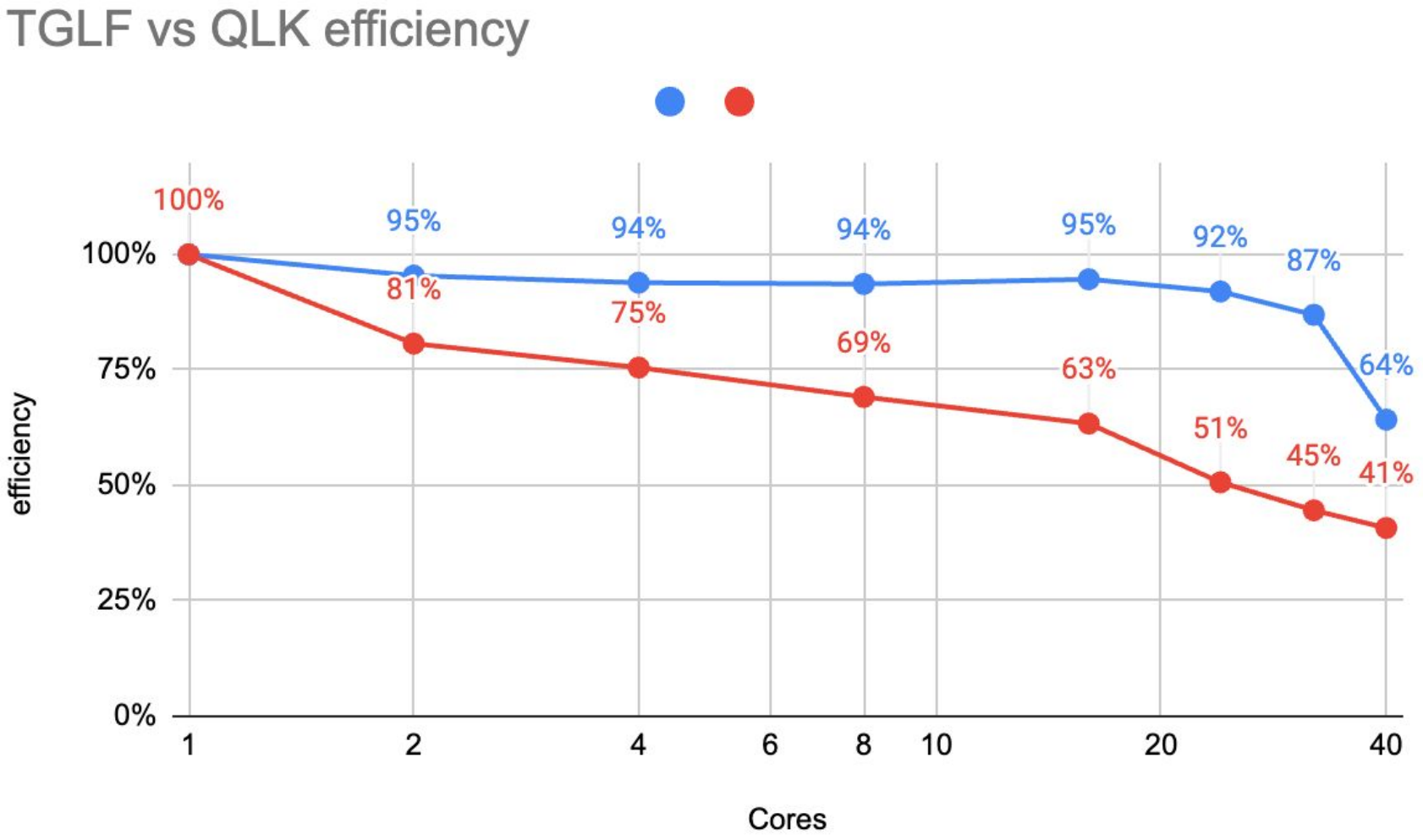
This was done with MUSCLE3 0.7.1
PCE order 5



TCI TGLF vs QLK scaling

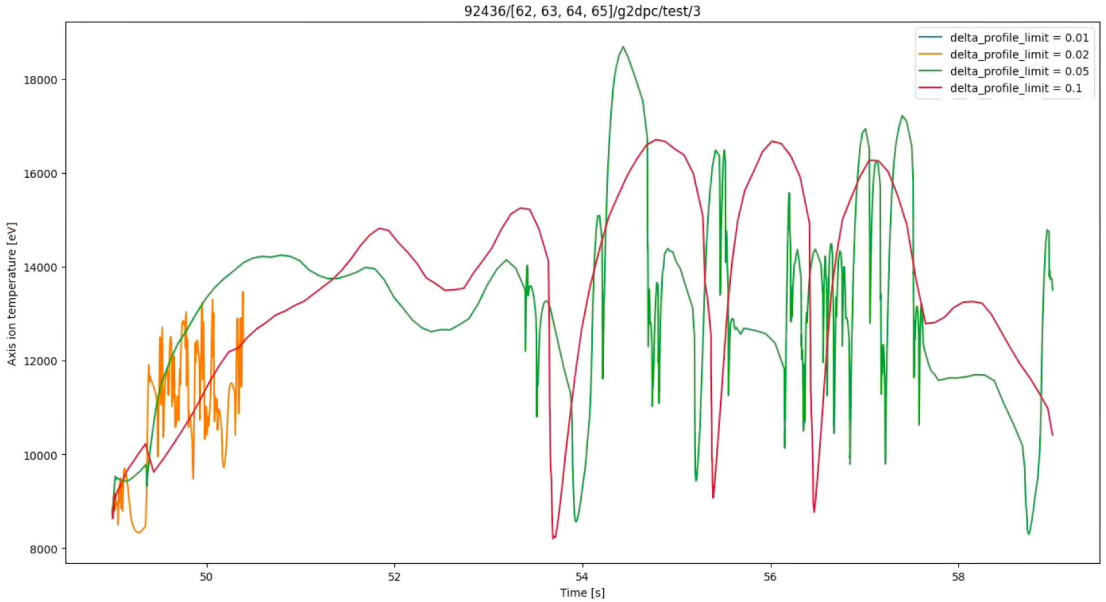
The scaling for TGLF seems to be much better for TGLF than for QLK

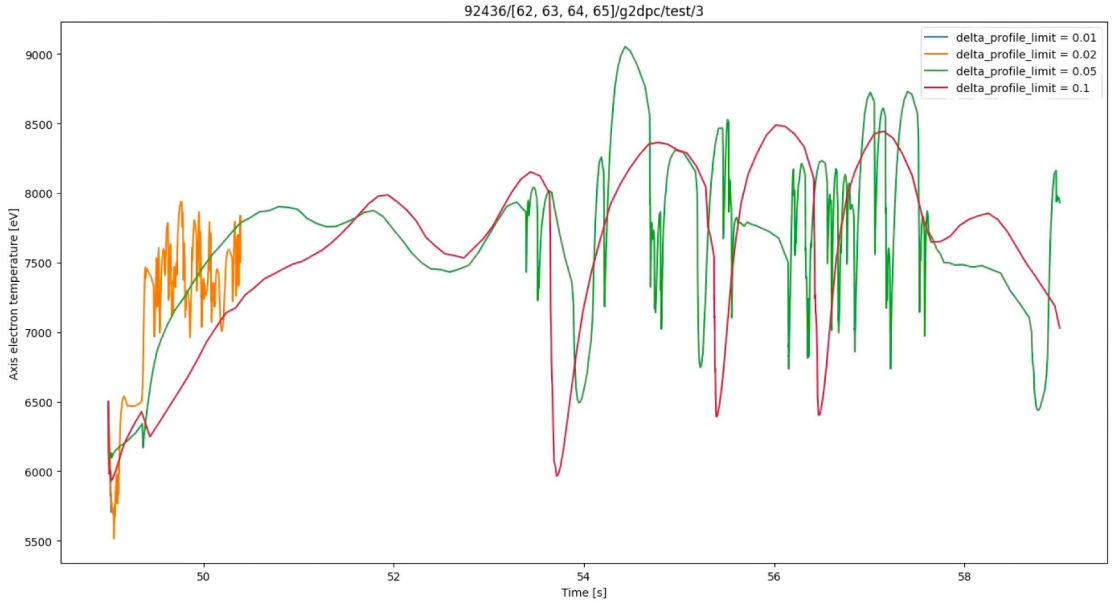
- For 32 cores
  - TGLF : 87%
  - QLK : 45%
- Different strategy?



Status of 2023-09-13

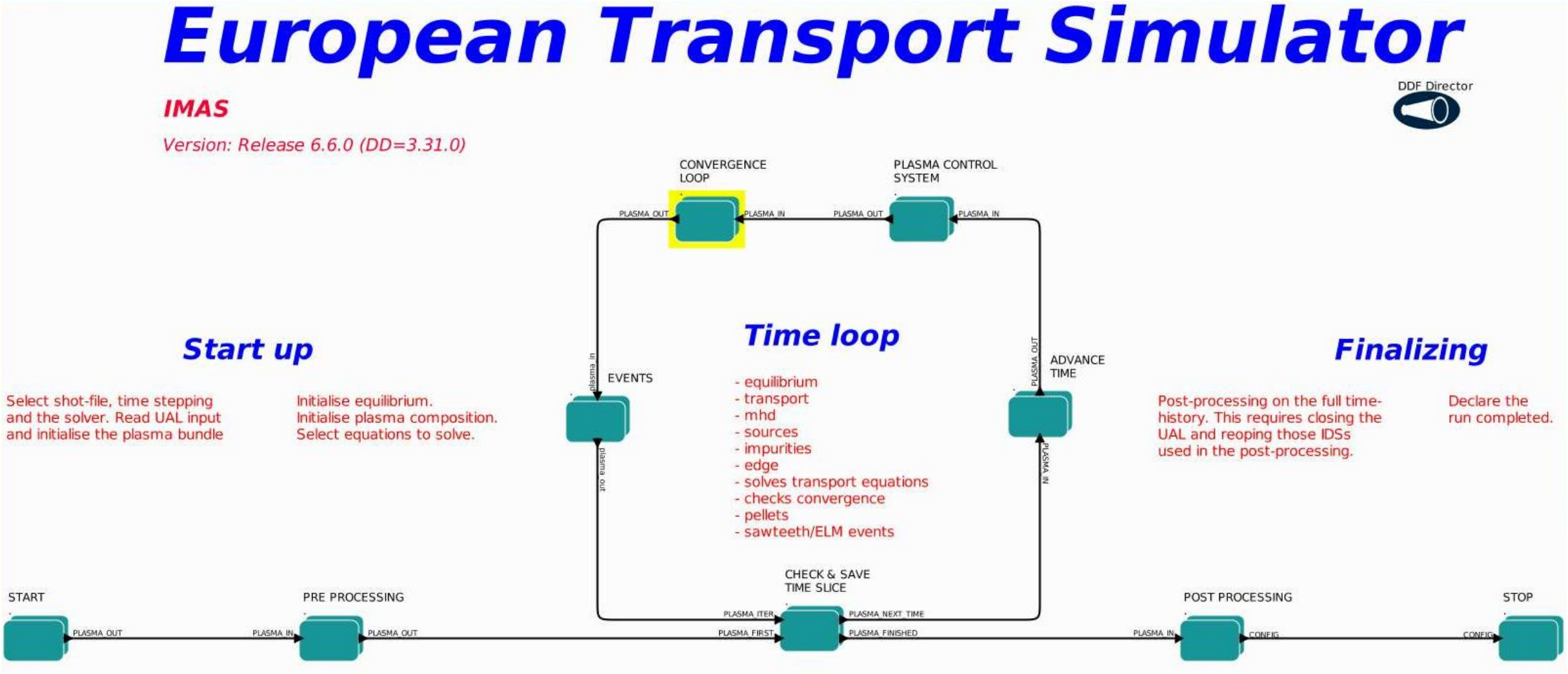
More physics, or fighting TGLF …



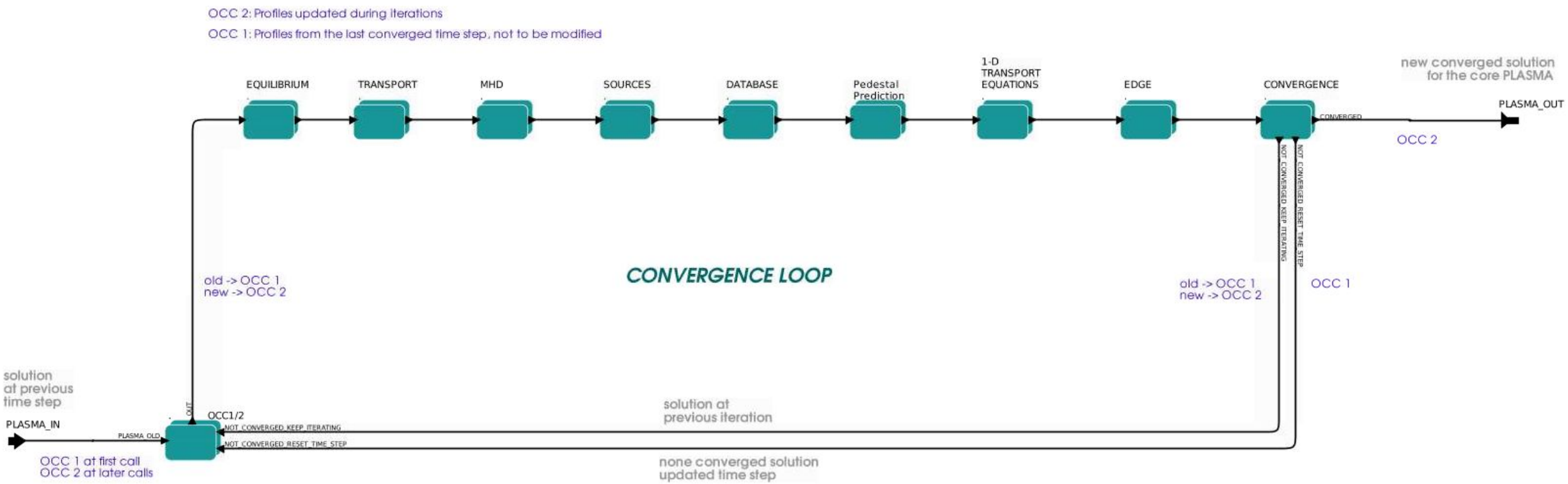


- Learning experience: running TGLF with internal boundary condition at rho_tor_norm=0.95
  - **Have not yet found a way of stabilizing the process**
    - **Using stride=4**
    - **Limitation on the (relative) amount a profile can change on a time-step**
- Applying the IBC at 0.75 was much more stable

Main panel of ETS6 (Kepler)



Zoom into the convergence loop



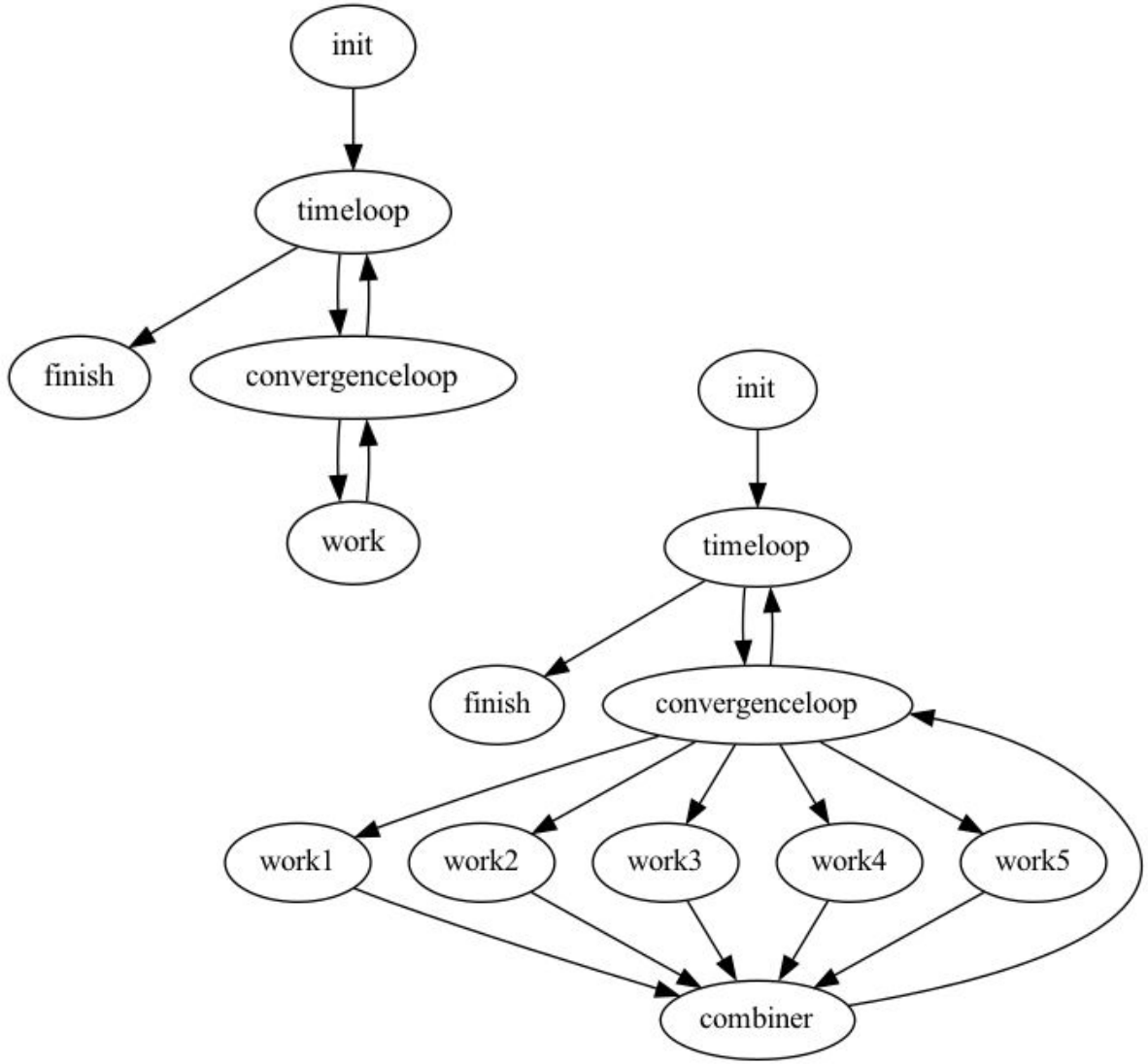
Started by implementing the logical pieces that would be needed

Started small:

- Implemented a prototype version of the ETS using MUSCLE3 (workflows/ets.ymmsl)
- Send a small equilibrium IDS around the workflow

  Have a
  - “timeloop” actor which advances time from 0 to 1 seconds
  - “convergenceloop” actor which performs 10 iterations
  - “work” actor representing the main physics loop of the ETS

Then extended the prototype to have 5 work actors in parallel and a combiner (workflows/ets_parallel.ymmsl)

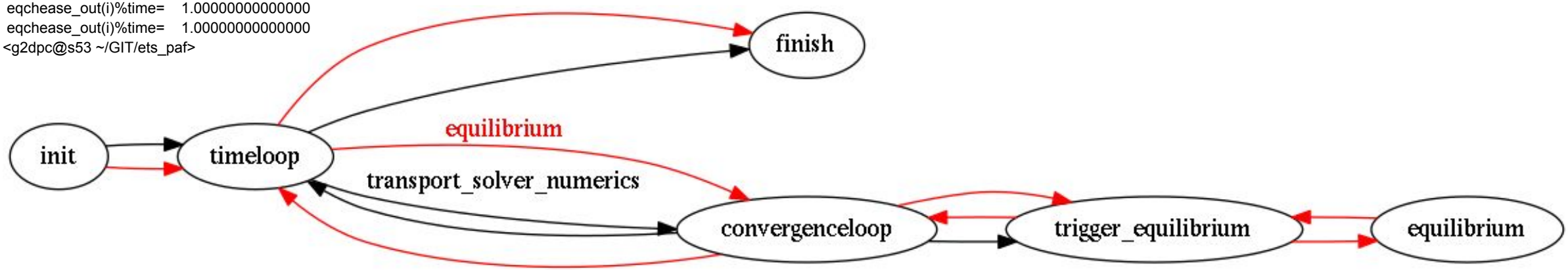


Test of physics actor trigger: trigger_equilibrium

iterations: 2
EquilibriumFrequency: 0: every_iteration; 1: every_time_step; 2: with_fixed_time_intervals
EquilibriumTimeIntervals: 0.25

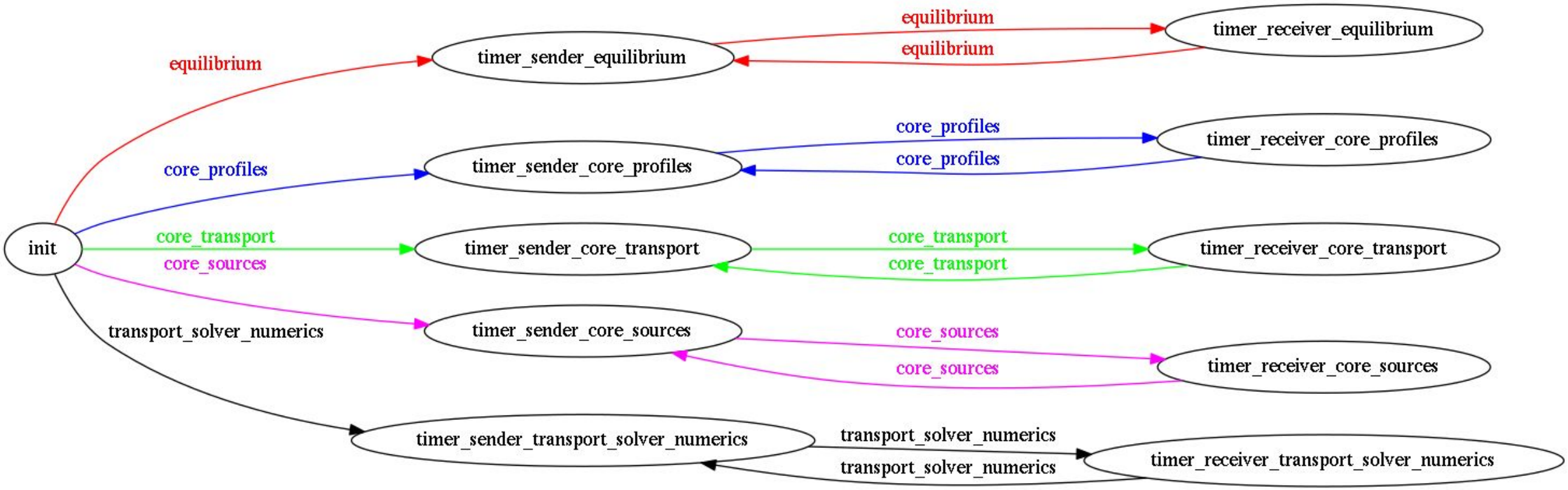
```
<g2dpc@s53 ~/GIT/ets_paf>grep 'eqchease_out(i)'
run_ETS_TEST_TRIGGER_EQUILIBRIUM_20230530_131004/instances/equilibrium/stdout.txt
 eqchease_out(i)%time=   0.100000000000000
 eqchease_out(i)%time=   0.100000000000000
 eqchease_out(i)%time=   0.200000000000000
 eqchease_out(i)%time=   0.200000000000000
 eqchease_out(i)%time=   0.300000000000000
 eqchease_out(i)%time=   0.300000000000000
 eqchease_out(i)%time=   0.400000000000000
 eqchease_out(i)%time=   0.400000000000000
 eqchease_out(i)%time=   0.500000000000000
 eqchease_out(i)%time=   0.500000000000000
 eqchease_out(i)%time=   0.600000000000000
 eqchease_out(i)%time=   0.600000000000000
 eqchease_out(i)%time=   0.700000000000000
 eqchease_out(i)%time=   0.700000000000000
 eqchease_out(i)%time=   0.800000000000000
 eqchease_out(i)%time=   0.800000000000000
 eqchease_out(i)%time=   0.900000000000000
 eqchease_out(i)%time=   0.900000000000000
 eqchease_out(i)%time=    1.00000000000000
 eqchease_out(i)%time=    1.00000000000000
<g2dpc@s53 ~/GIT/ets_paf>
```

```
<<g2dpc@s53 ~/GIT/ets_paf>grep 'eqchease_out(i)'
run_ETS_TEST_TRIGGER_EQUILIBRIUM_20230530_131507/instances/equilibrium/stdout.txt
 eqchease_out(i)%time=   0.100000000000000
 eqchease_out(i)%time=   0.200000000000000
 eqchease_out(i)%time=   0.300000000000000
 eqchease_out(i)%time=   0.400000000000000
 eqchease_out(i)%time=   0.500000000000000
 eqchease_out(i)%time=   0.600000000000000
 eqchease_out(i)%time=   0.700000000000000
 eqchease_out(i)%time=   0.800000000000000
 eqchease_out(i)%time=   0.900000000000000
 eqchease_out(i)%time=    1.00000000000000
<g2dpc@s53 ~/GIT/ets_paf>grep 'eqchease_out(i)'
run_ETS_TEST_TRIGGER_EQUILIBRIUM_20230530_132253/instances/equilibrium/stdout.txt
 eqchease_out(i)%time=   0.100000000000000
 eqchease_out(i)%time=   0.400000000000000
 eqchease_out(i)%time=   0.700000000000000
 eqchease_out(i)%time=    1.00000000000000
<g2dpc@s53 ~/GIT/ets_paf>
```



Timing tests (send and receive / receive and send)

Workflow implemented to time ping-pong flows of IDSes between a sender and receiver



Timing tests (send and receive / receive and send)

Timing results:

- **Less than 0.1 seconds on average to send and receive back again the biggest IDS (24.75 MB)**

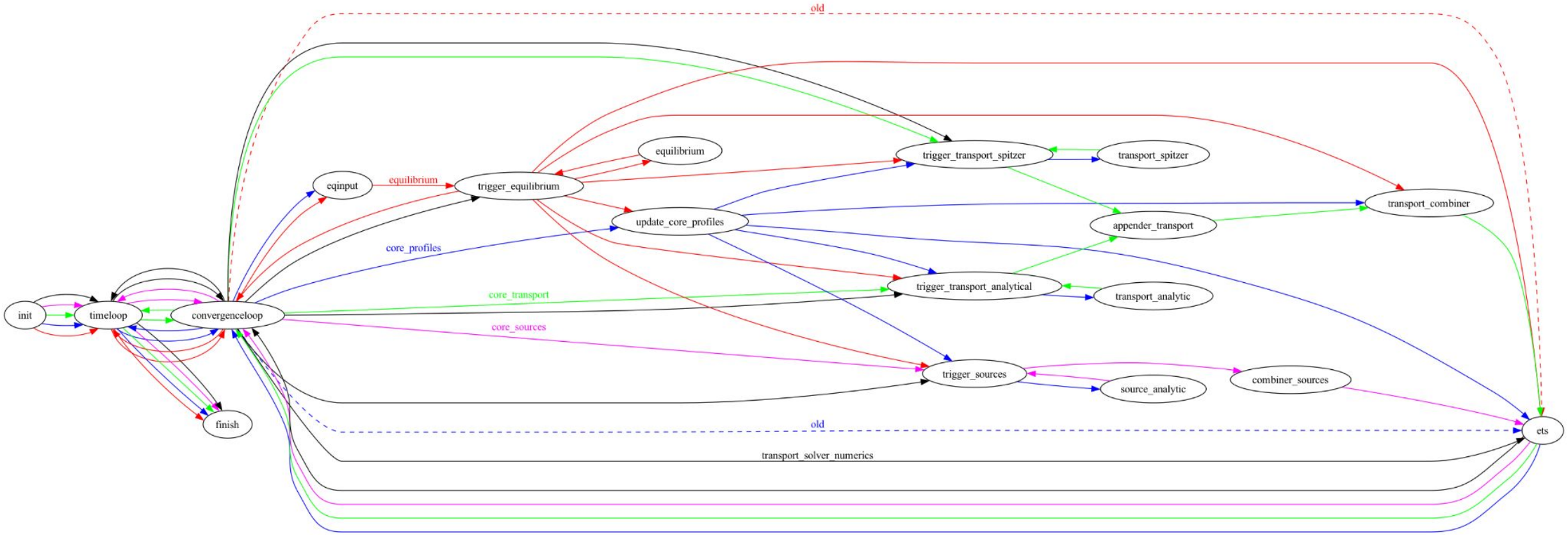
		Elapsed time (seconds)				
Cycles	**IDS**	**Size**	**sender**		**receiver**	
			average	**stddev**	**average**	**stddev**
10	equilibrium	25952797	0.1035	0.0244	0.0290	0.0052
	core_profiles	289325	0.0270	0.0652	0.0011	0.0004
	core_transport	766757	0.0203	0.0399	0.0016	0.0006
	core_sources	350596	0.0162	0.0317	0.0012	0.0003
	transport_solver_numerics	420188	0.0064	0.0027	0.0013	0.0007
100	equilibrium	25952797	0.0890	0.0102	0.0268	0.0019
	core_profiles	289325	0.0068	0.0023	0.0012	0.0004
	core_transport	766757	0.0082	0.0014	0.0016	0.0003
	core_sources	350596	0.0076	0.0014	0.0013	0.0004
	transport_solver_numerics	420188	0.0072	0.0023	0.0013	0.0005

Incorporated the pieces into a prototype ETS workflow

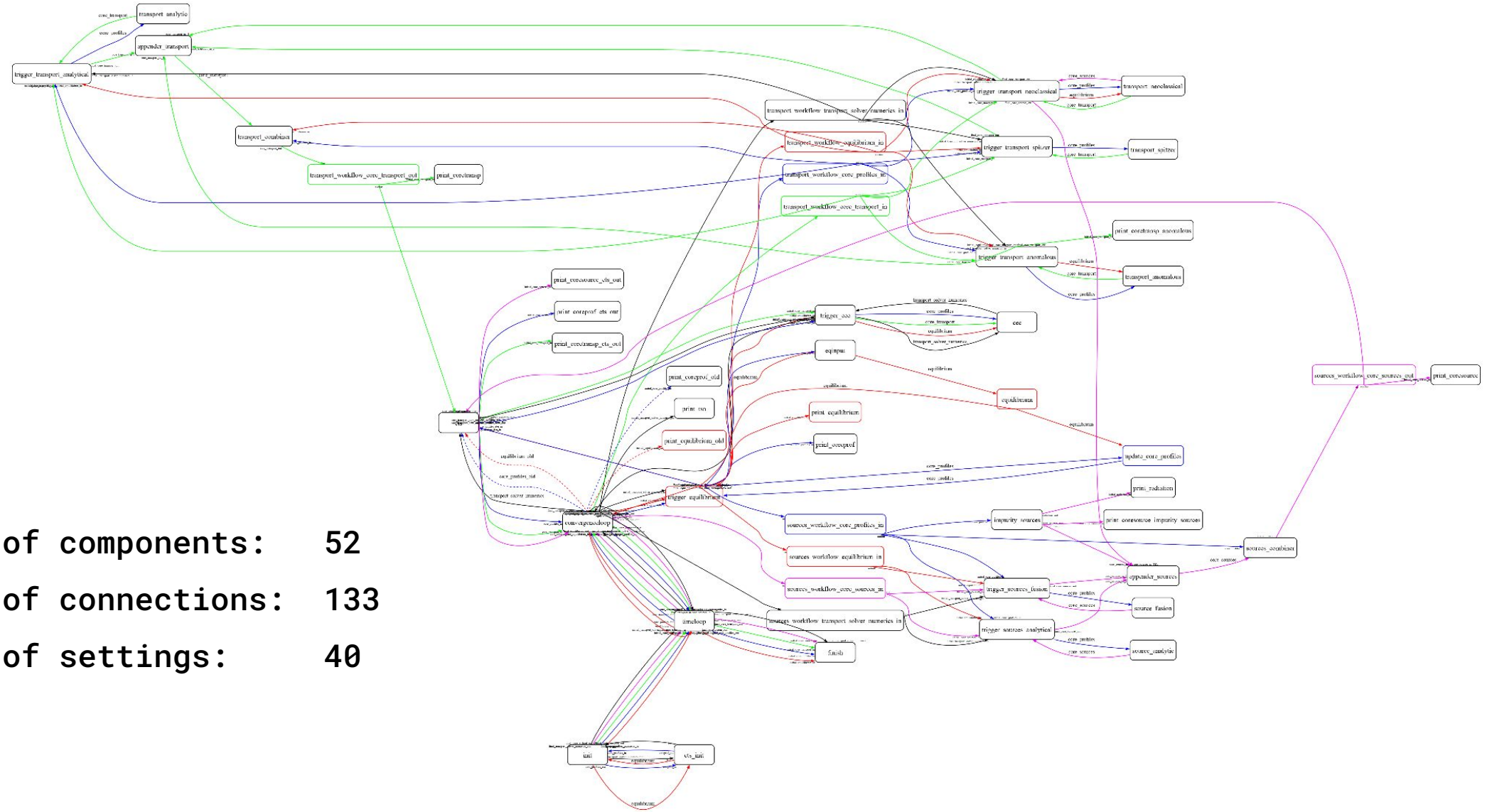
Have options to run “expensive” models every iteration, every time-step or every X seconds

Models for equilibrium update, sources (analytical expressions), transport (Spitzer and analytical expressions), appenders and combiners (fully for transport, in progress for sources)

Can solve current, density and temperature equations



Then developed the ETS workflow with all of the key pieces

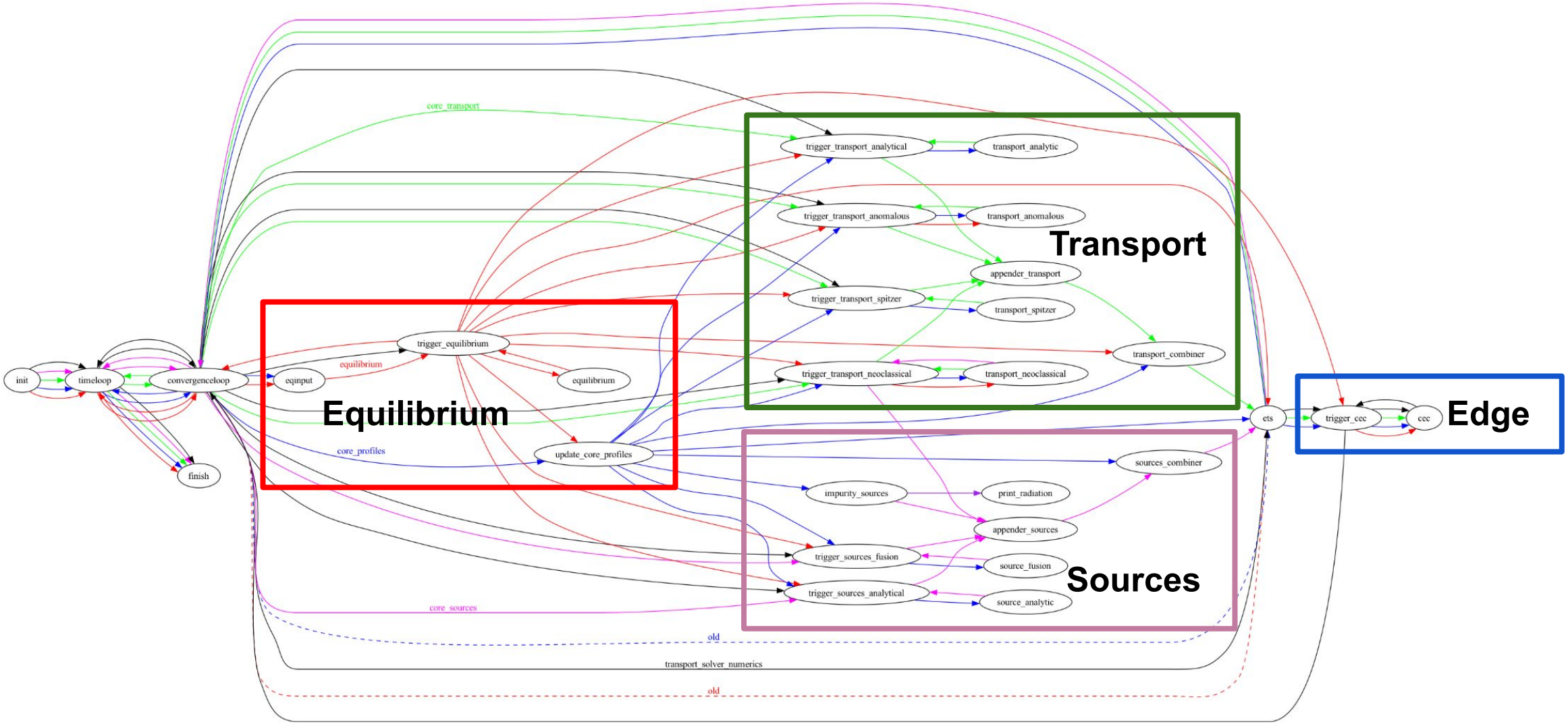


Number of components: 52

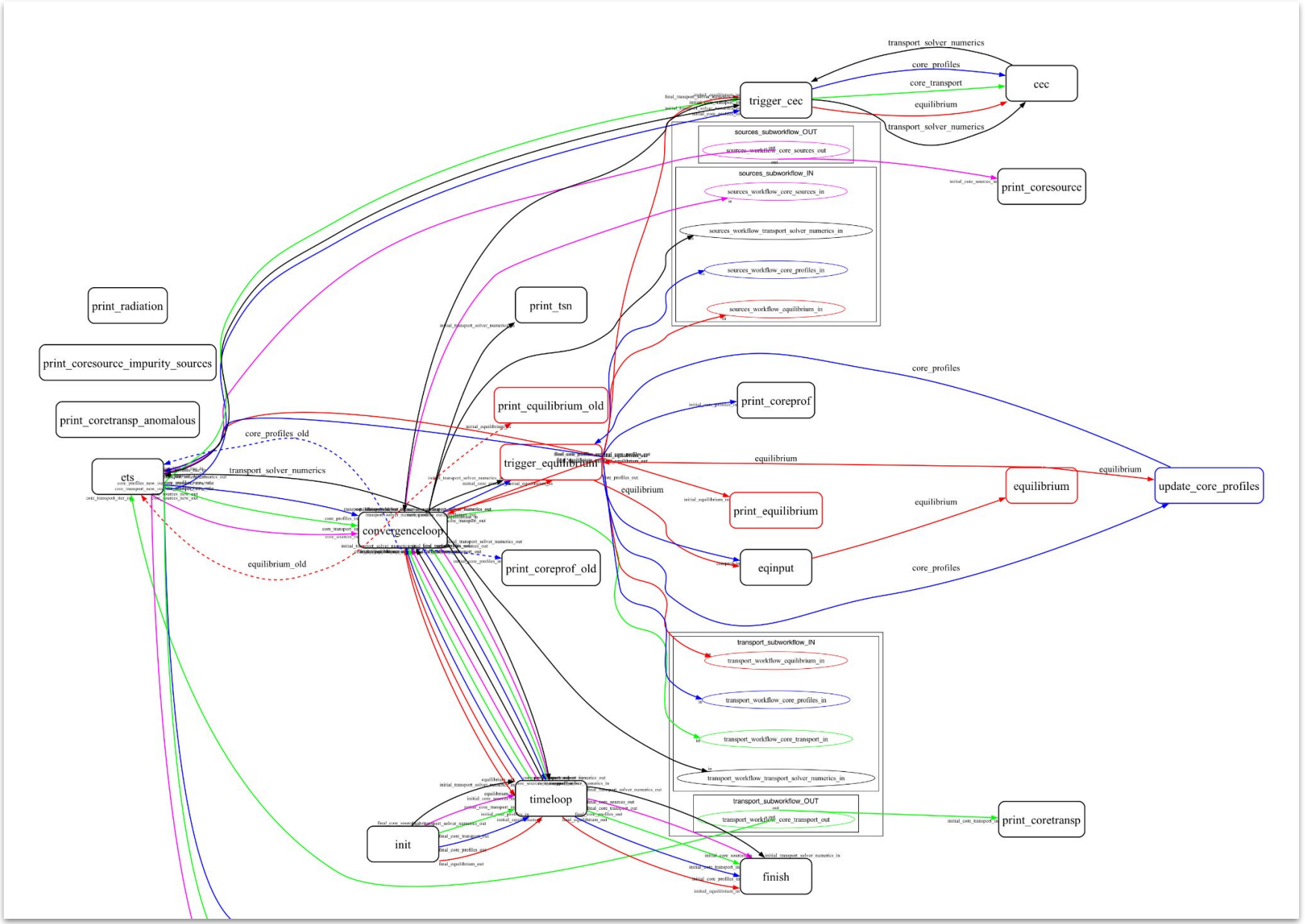
Number of connections: 133

Number of settings: 40

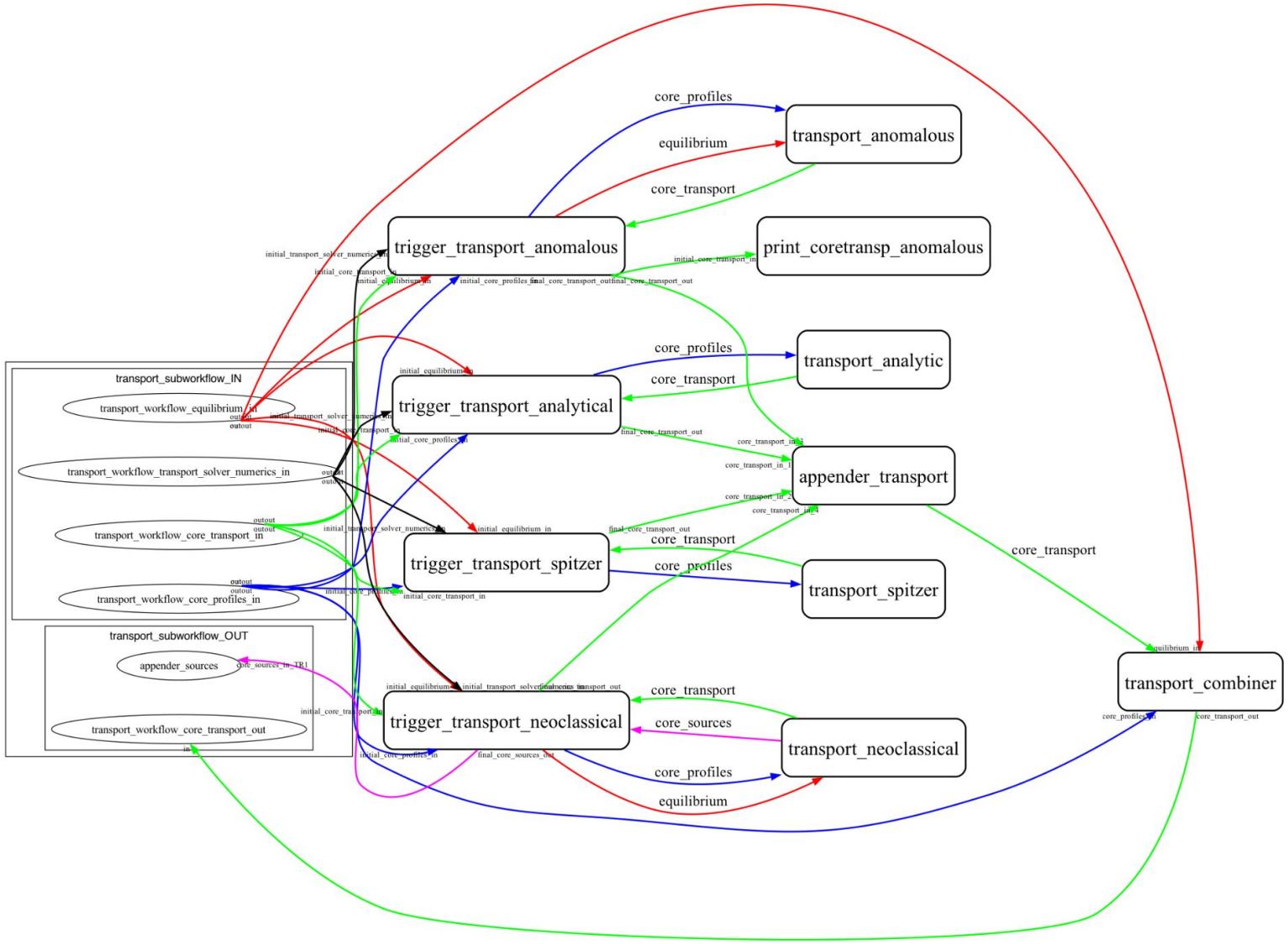
Decided to implement subworkflows



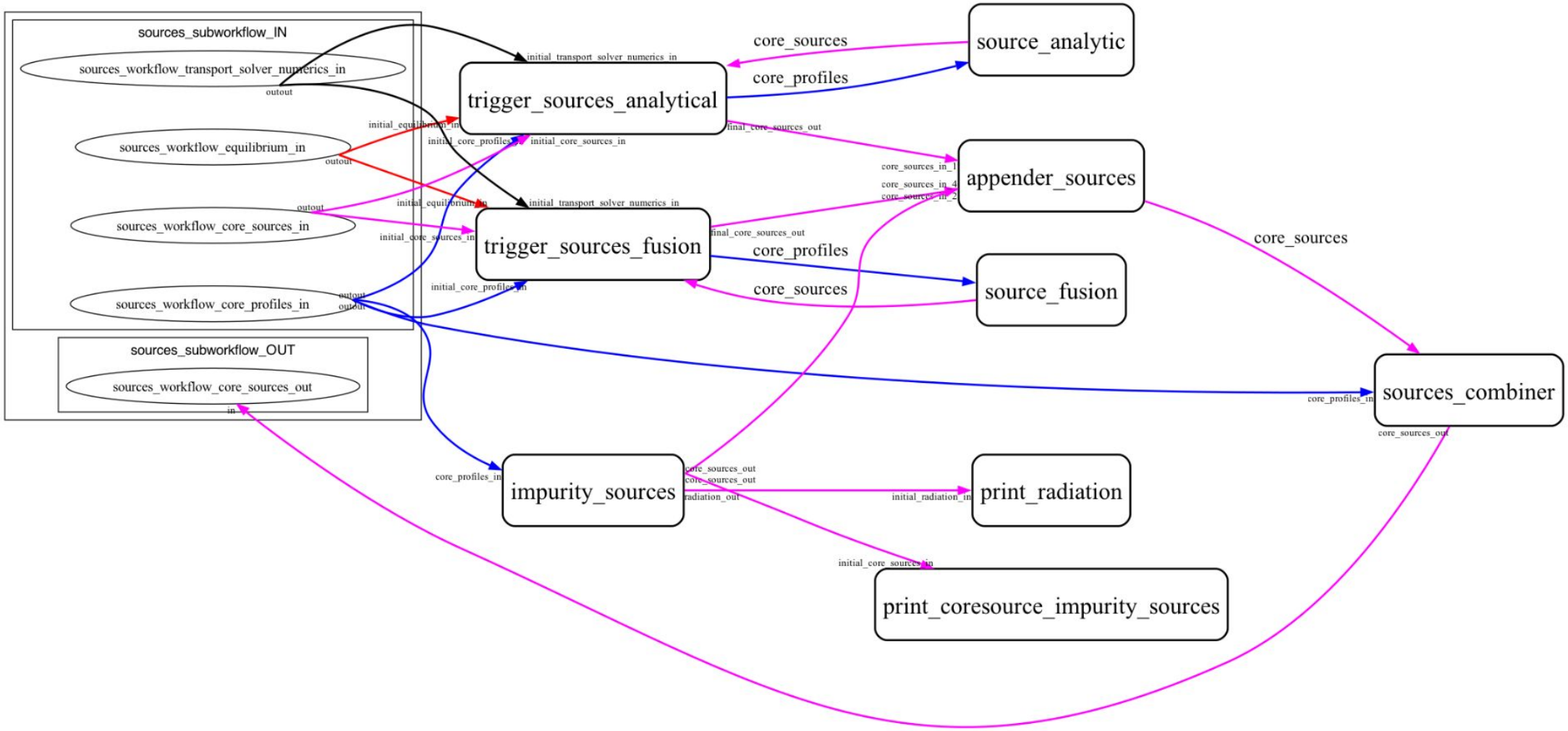
ets_main_encapsulated



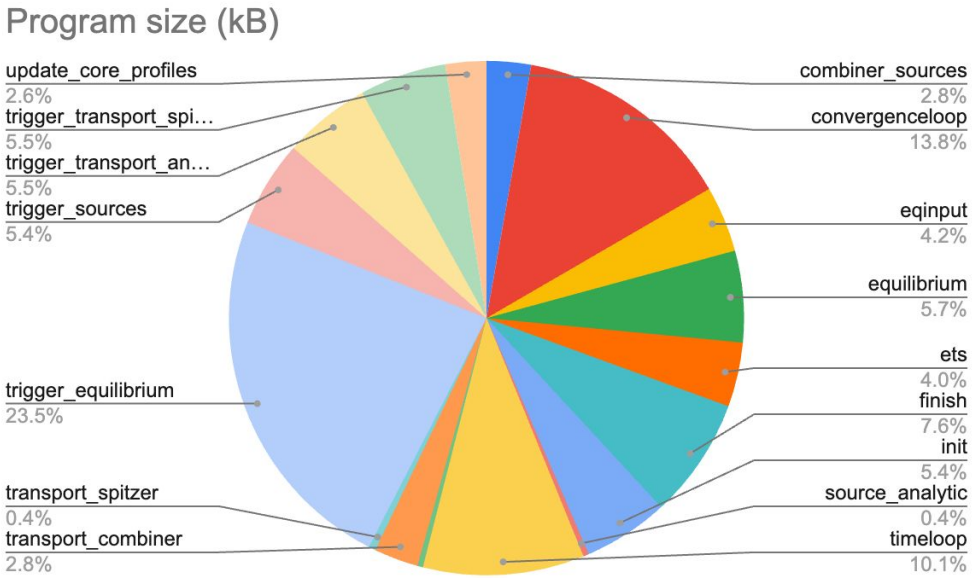
transport_subworkflow_encapsulated: redrawn

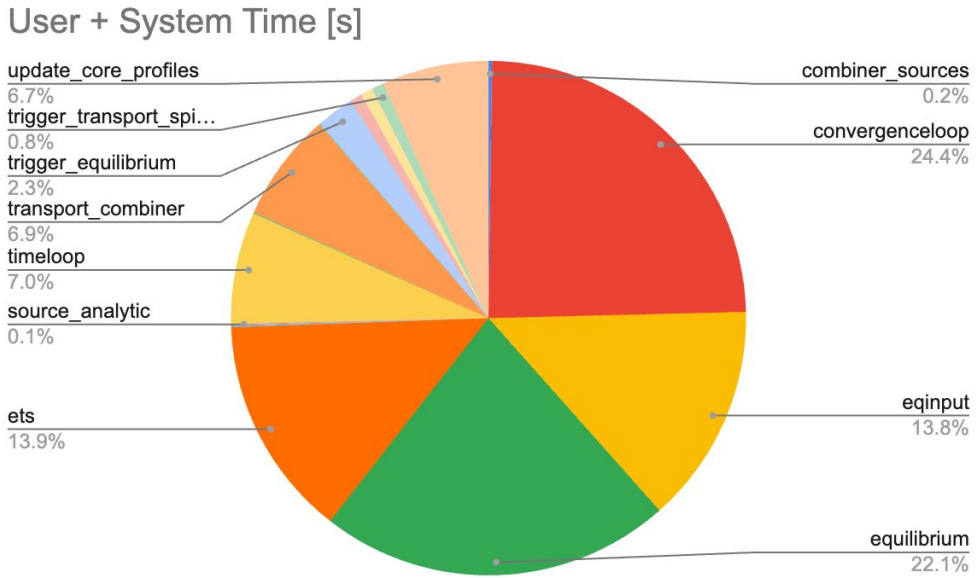


sources_subworkflow_encapsulated: redrawn



Memory and time usage of ETS_PAF

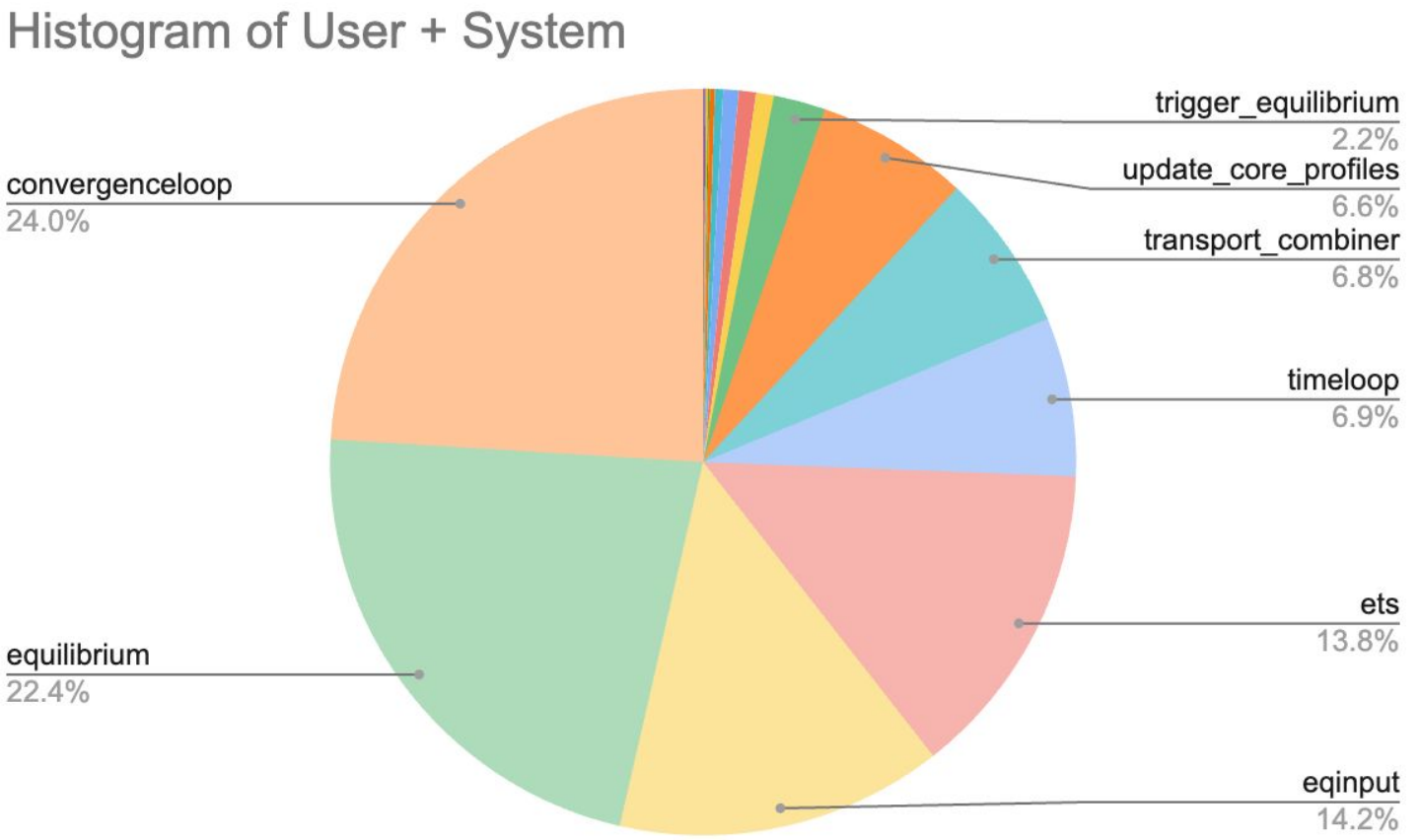




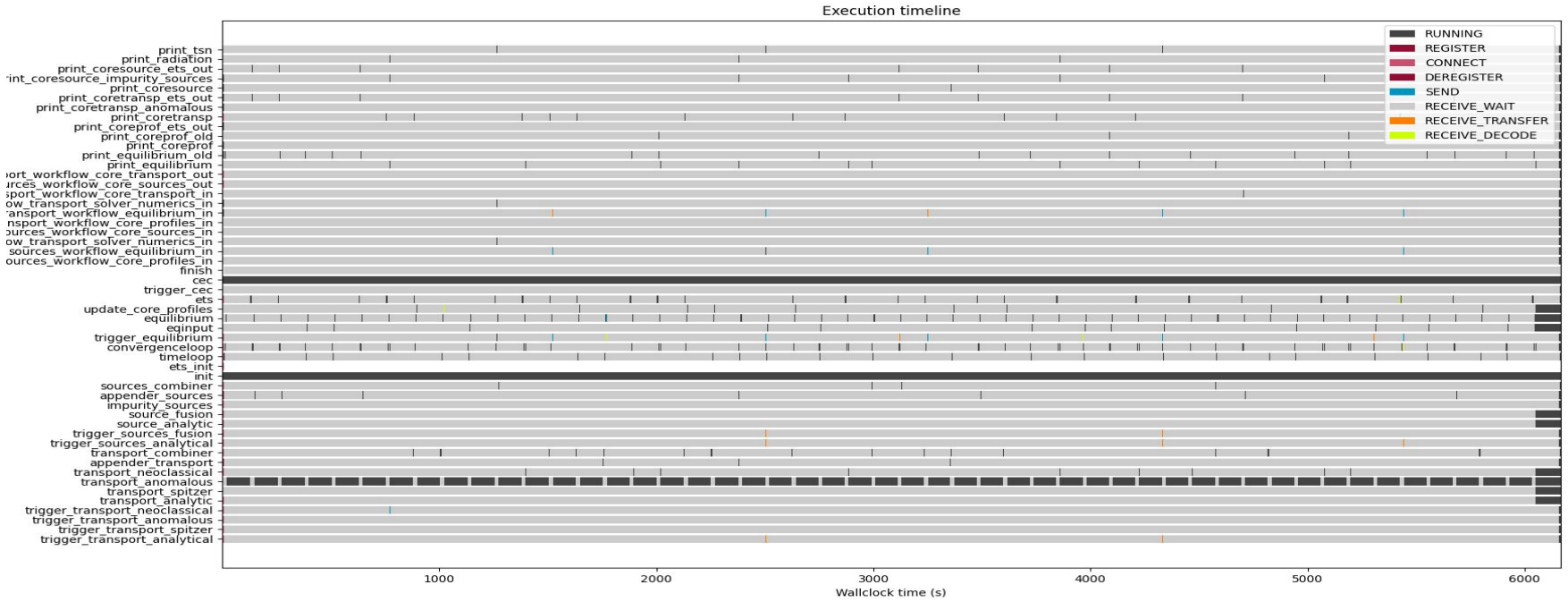
- Biggest user of memory is trigger_equilibrium
- Biggest users of time
  - **convergenceloop [multiple serialize/deserialize operations]**
  - **equilibrium (chease)**
  - **ets [multiple serialize/deserialize operations, including two equilibrium IDSes]**
  - **eqinput [multiple serialize/deserialize operations, particularly of the equilibrium IDS]**

Digging deeper into two python routines

- convergenceloop
  - **992.49 elapsed serialization time**
  - **857.03 elapsed deserialization time**
  - **97% of user+system time**
- timeloop
  - **258.24 elapsed serialization time**
  - **226.39 elapsed deserialization time**
  - **88% of user+system time**
- Speed-up 12%
- Efficiency 6%



Timeline for a JET simulation (50 timesteps)

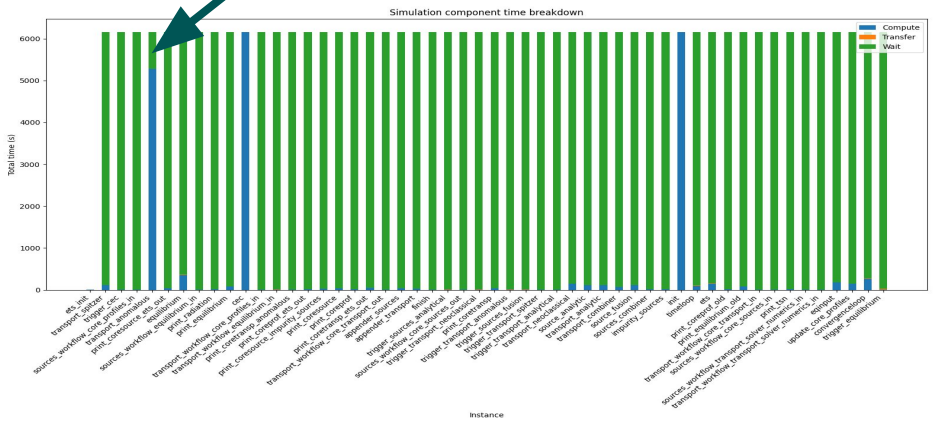


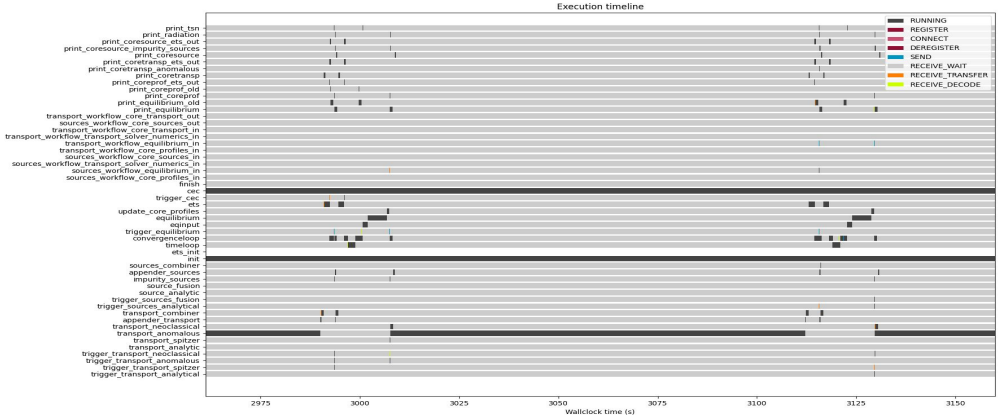
Unexpected pattern

Unexpected pattern

MPI code on 32 cores

MPI code on 32 cores





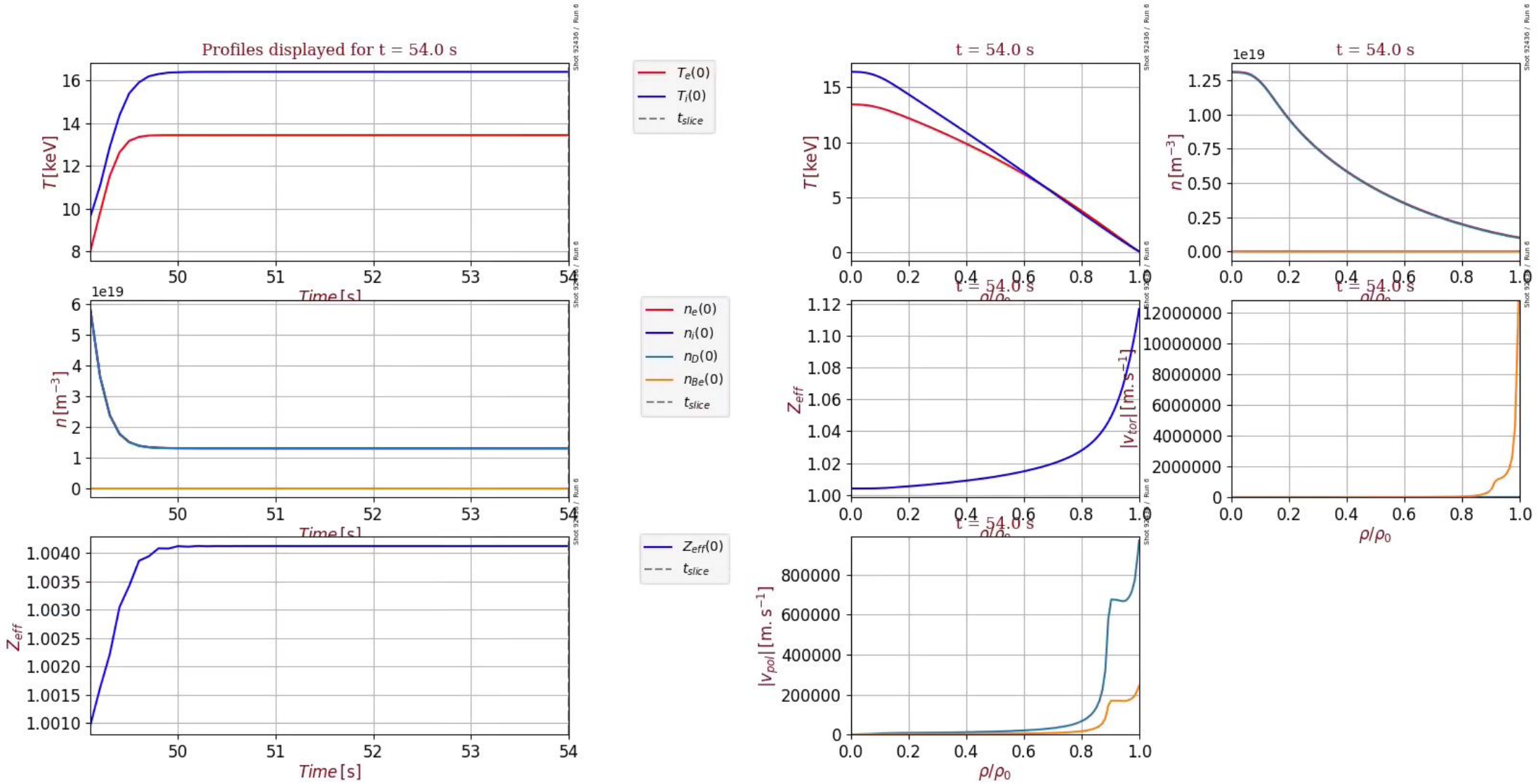
Status

- Have a workflow running with all (?) of the key pieces
  - **Time loop**
  - **Convergence loop**
    - **Equilibrium / update core_profiles**
    - **Transport codes**
      - With appender and combiner
      - Pretty complete
        - TCI based codes + analytical background
    - **Sources**
      - Not yet complete, but have
        - Analytical models
        - Thermonuclear fusion (simplified)
        - Impurities
    - **Core-edge prototype**
    - **European Transport Solver (transport solver)**

Status, II

- Much of the logic of the workflow is implemented in one or more YAML files that provide
  - **The names of the actors**
  - **The connections between the actors**
  - **The resources needed by the actors**
  - **The actual implementation for each actor**
  - **The settings needed by the actors**
- Then have a collection of python actors that implement I/O, logic (data readers and writers, time loop, convergence loop, triggers, …)
- Multiple sources of the iWrap'ped actors
  - **Waiting for the implementation of the Actor Release Procedure**

Example: JET D+Be simulation, BGB + analytic sources



UQ: Two options for doing Uncertainty Quantification

1. Run EasyVVUQ* in the usual fashion, launching many copies of the MUSCLE workflow using one of the supported technologies (QCG-PJ, SLURM, DASK, etc.)

   EASYVVUQ(MUSCLE3)

2. Embed EasyVVUQ as a component within a MUSCLE3 workflow

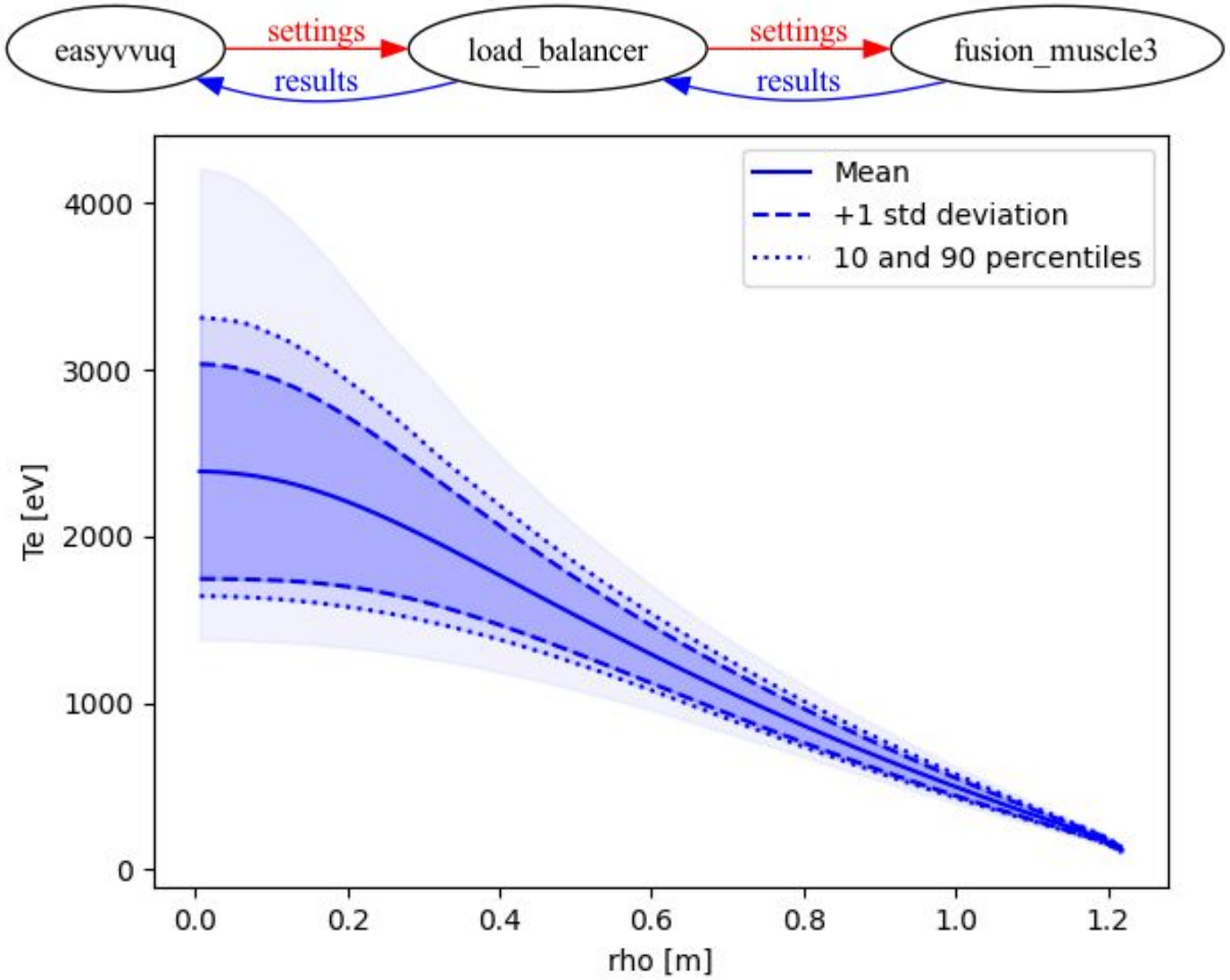
   MUSCLE3(EASYVVUQ)

* https://easyvvuq.readthedocs.io/en/dev/ "EasyVVUQ: Uncertainty intervals for everyone!"

UQ inside MUSCLE3 [MUSCLE3(EASYVVUQ)]

Have an

- **easyvvuq** actor that creates an UQ "campaign"
  - PCE order 5
  - 5 varying parameters
- Passes settings to the **load_balancer**
- Which runs multiple copies (10 in this case) of the **fusion_muscle3** code, each with different settings
- Use the fusion proxy app instead of the full workflow at this stage
  - Intend to use the full workflow in the future

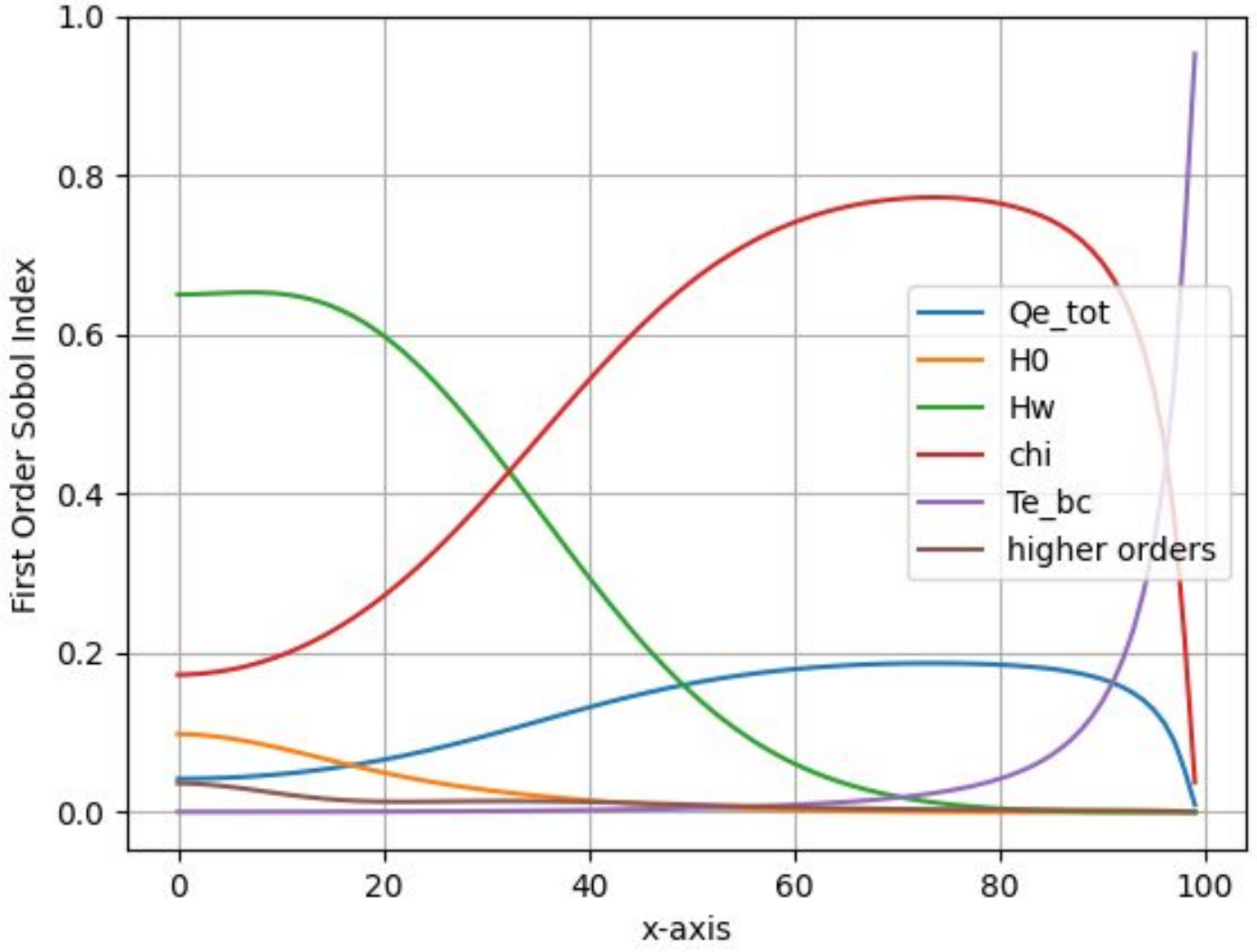


UQ inside MUSCLE3

UQ workflow using EasyVVUQ with muscle3 on the simple fusion transport solver from the EasyVVUQ tutorials

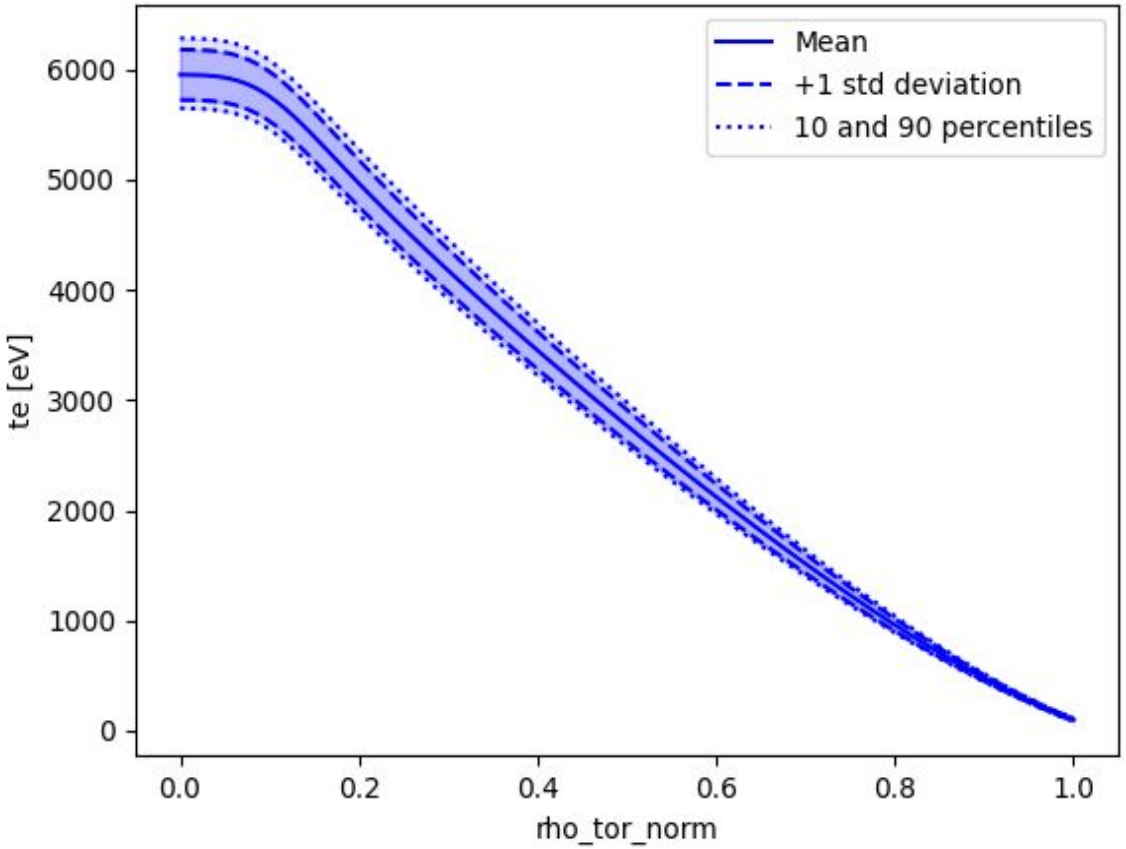
- PCE order 5
- 5 varying parameters
- 10 copies of the fusion code

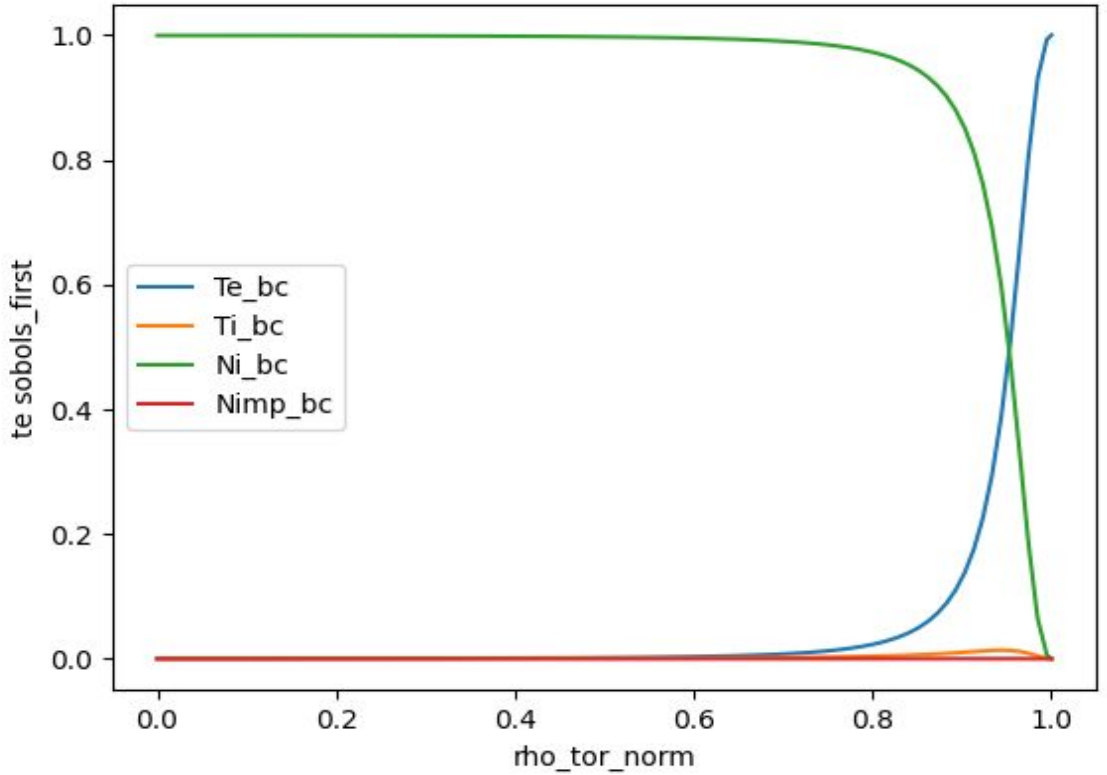
Hope to implement this functionality with the ETS-PAF



UQ applied to ETS-PAF [EASYVVUQ(MUSCLE3)]

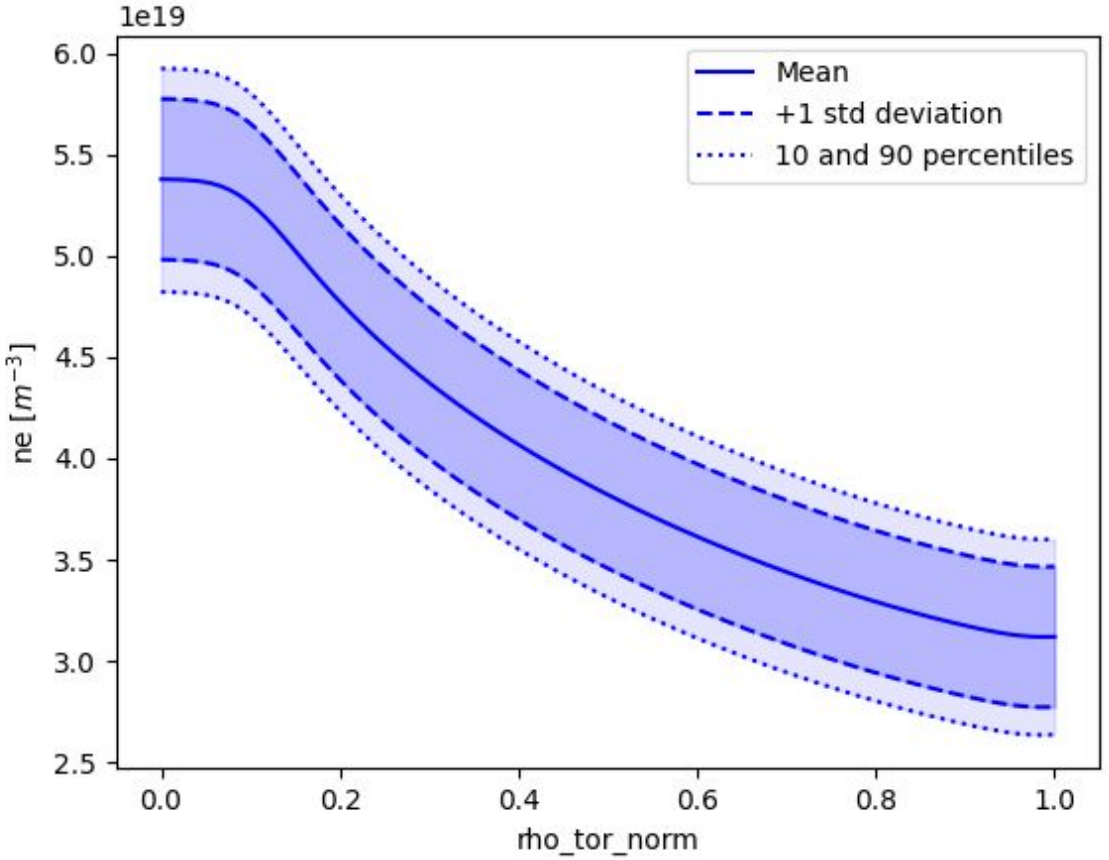
- 5 s JET BGB
- 4 varying parameters
- Electron temperature (T_e) profiles after 5 seconds of simulation
- Sobol' first for T_e

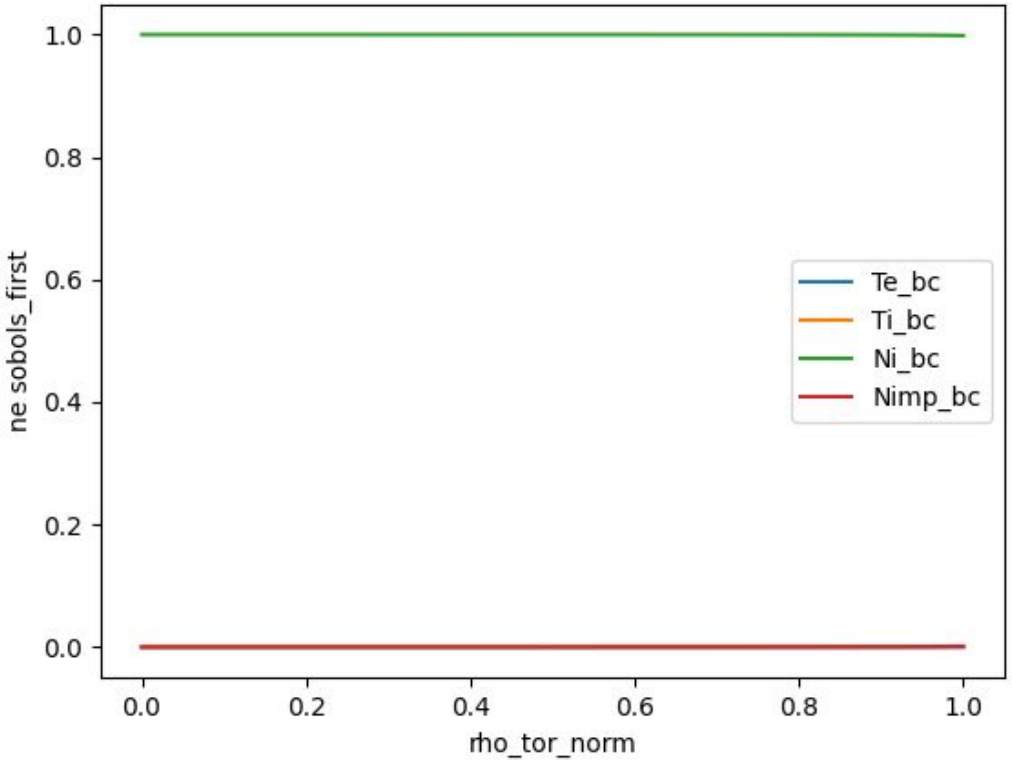




UQ applied to ETS-PAF

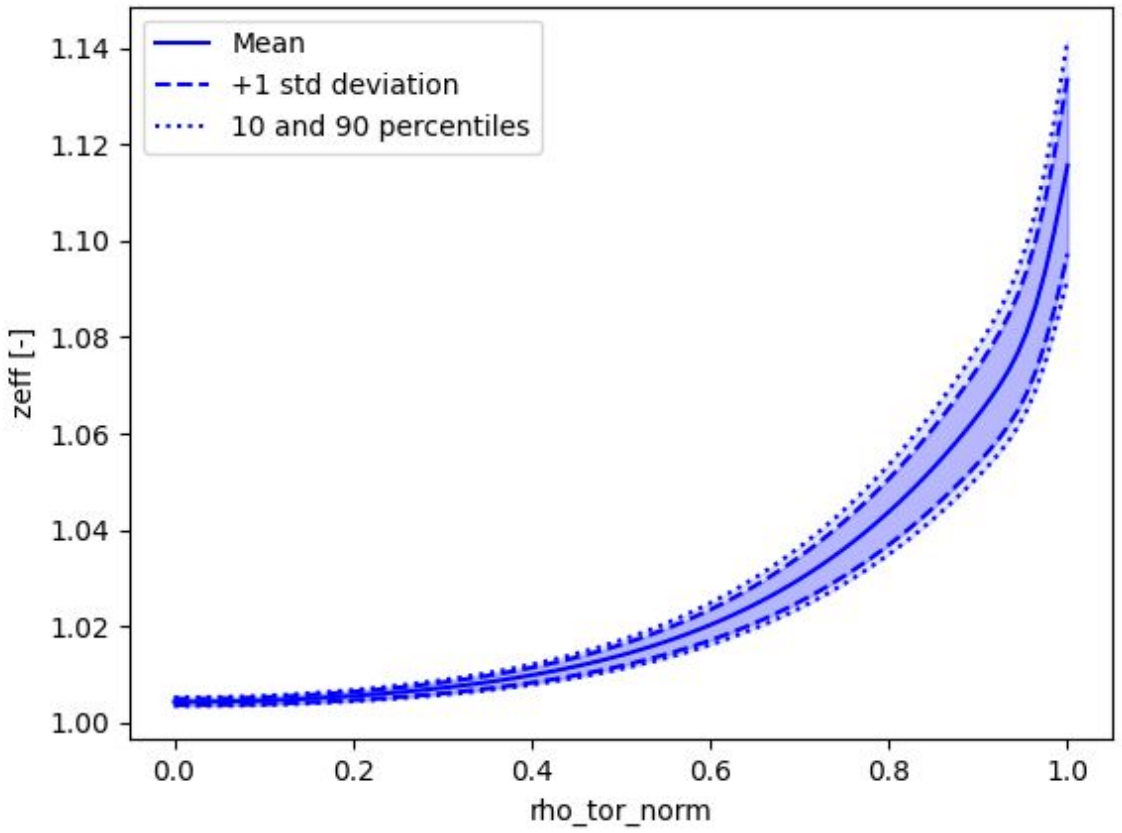
- Electron density (n_e) profiles after 5 seconds of simulation
- Sobol' first for n_e

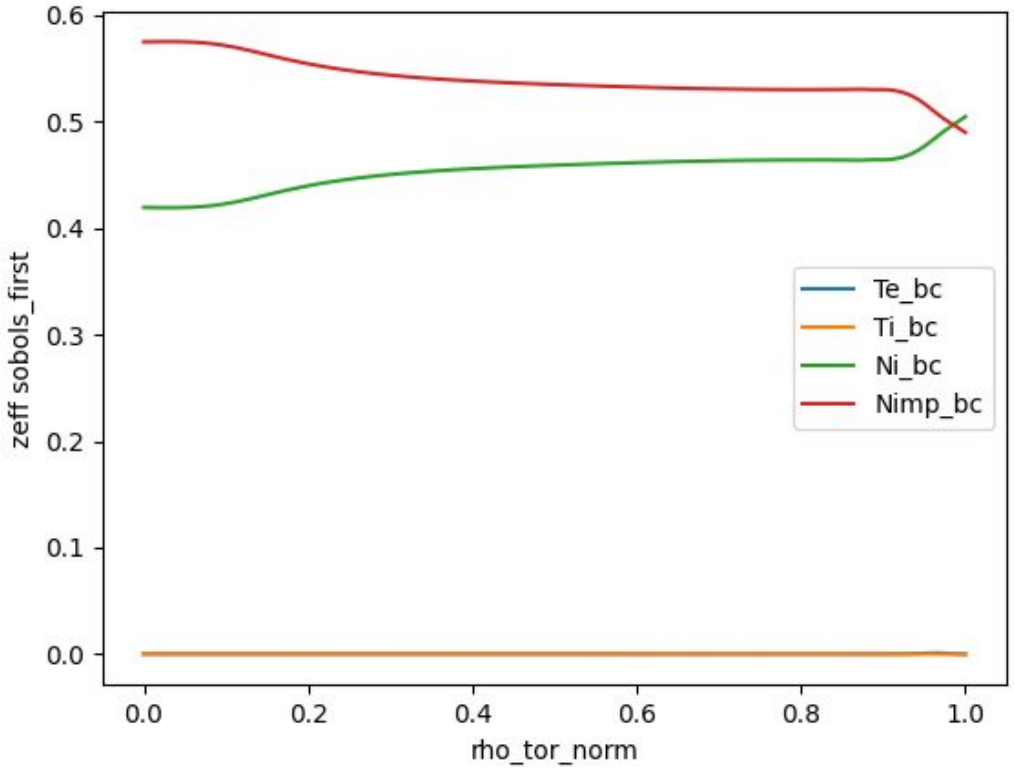




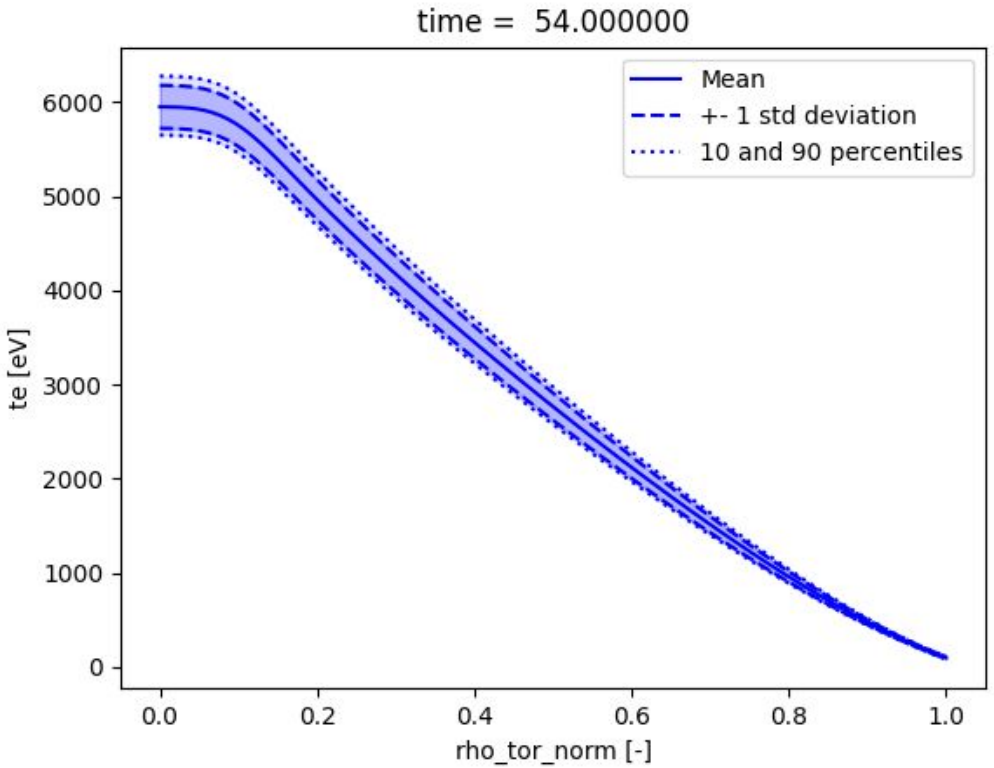
UQ applied to ETS-PAF

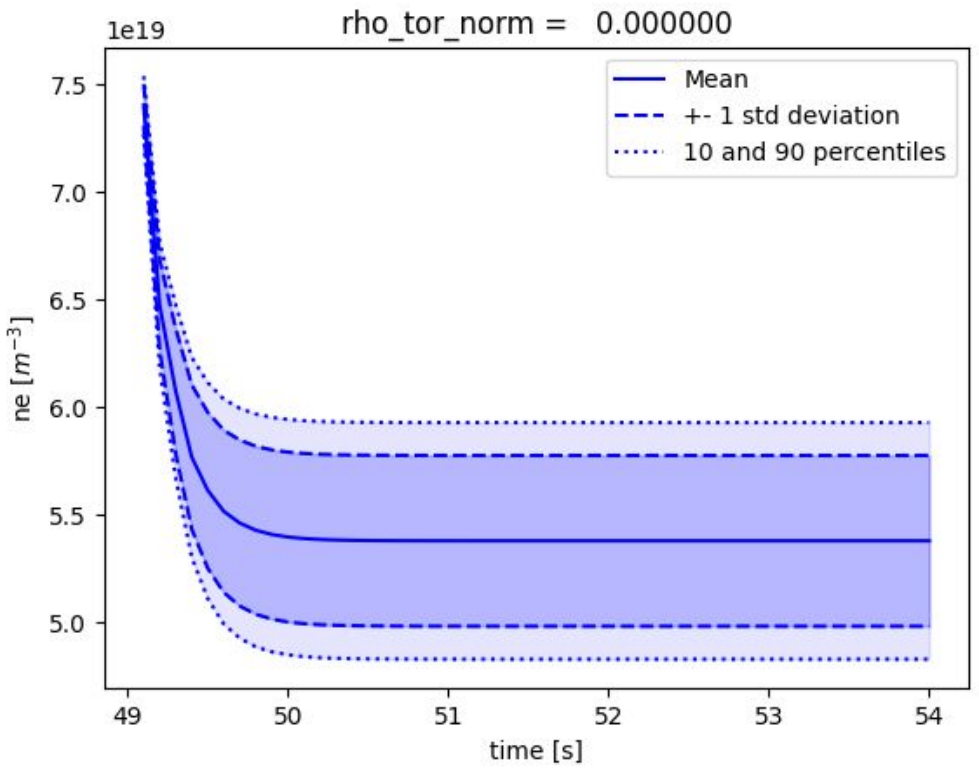
- Z_{eff} profiles after 5 seconds of simulation
- Sobol' first for Z_{eff}





Have extended the UQ to all time-slices





- This involved extracting all time slices from each run and storing the data in results.csv
- Modifying the processing the data to convert the 1d data back to profiles as a function of time and space
- Modifying the plotting to deal with this, allowing the user to choose cuts
  - **[-1,:] : over rho_tor_norm at the last time point**
  - **[:,0] : over time for the centre of the plasma**
- Plots based on PCE order = 3