Garching, march 2012

EFDA

EUROPEAN FUSION DEVELOPMENT AGREEMENT

Task Force
INTEGRATED TOKAMAK MODELLING

European Transport Solver Training

ETS_C WORKFLOW

T.ANIEL, V.BASIUK, P.HUYNH

The ETS is a core transport solver which solves a set of diffusion equations (like JETTO, ASTRA and CRONOS)

It is a modular workflow combining different actors/workflows such as :

- Transport Solvers
- Workflows/actors of other IMP's (IMP12, IMP4 and IMP5)

Workflow includes nested loop

- Time loop
- Transport Convergence loop

The ETS includes 2 workflows linked with 2 transport solvers with different technical choices but the **workflows are compatible with any ITM actors and the 2 transport solvers will be interchangeable in most of the cases.**

They are called ETS_A and ETS_C.

The differences concern the treatment of the ion species, radial grid, equilibrium loop and will be detailed during the trainings

Outline

I. ETS, from the physics point of view

- Description
- V&V

II. ETS, KEPLER workflow

- Description
- How to configure a run from Kepler (as a developer)
- How to add your actor

III. Visualization and Post-treatment

- Matlab
- Python

IV. Practice

I. ETS from physics point of view

ETS, SOLVER (1/4)

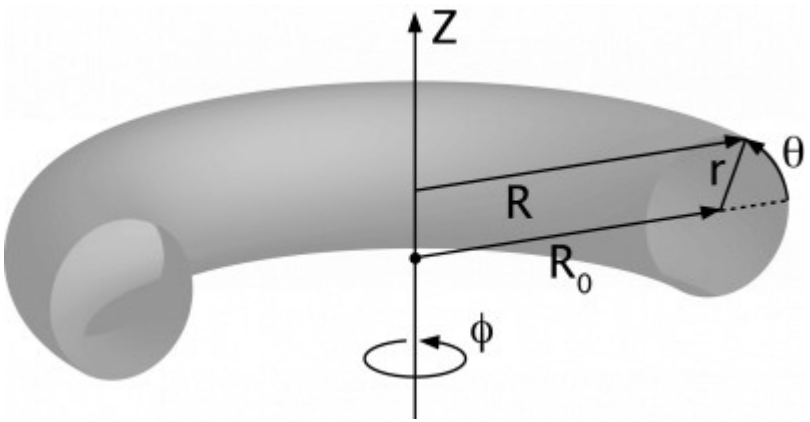
ITM convention

$$\psi = -\int B dS$$

ψ and $\partial\psi$ *quantities have the sign of* $-I_p$

q *has the sign of* $-I_p * +B_0$

B_ϕ *quantities have the sign of* $+B_0$



ETS, SOLVER (2/4)

All the ETS transport equations are developed in the following form

$$\left.\frac{\partial F}{\partial t}\right|_x = A\ddot{F} + B\dot{F} + CF + D$$

where $F = \begin{Bmatrix} \psi \\ Pe \\ Pi \\ ne \\ vtor \end{Bmatrix}$ and $x = \frac{\rho}{\rho_{max}}$

A, B, C, D are matrices

A(nbrho,nbeq,nbeq)
B(nbrho,nbeq,nbeq)
C(nbrho,nbeq,nbeq)
D(nbrho,nbeq)

The radial grid is uniformed and normalized (nbrho = 101)

The solver takes into account non diagonal terms coupling Pe, Pi and ne
Adding a new equation is easy (nbeq=nbeq+1, needs to fill the matrices)

ETS, SOLVER (3/4)

Ion density computation

5 ion species

n_{main}, n_{min1}, n_{min2}, n_{imp1}, n_{imp2} (index 1 to 5 in the UAL)

fully ionized (used in NCLASS)

deduced from :

- the electro neutrality
- the effective charge (Z_{eff})
- and 3 ratios

$$Z_{eff} = \frac{\sum_{i=1,5} Z_i^2 \, n_i}{n_e}$$

$$n_e = \sum_{i=1,5} Z_i \, n_i$$

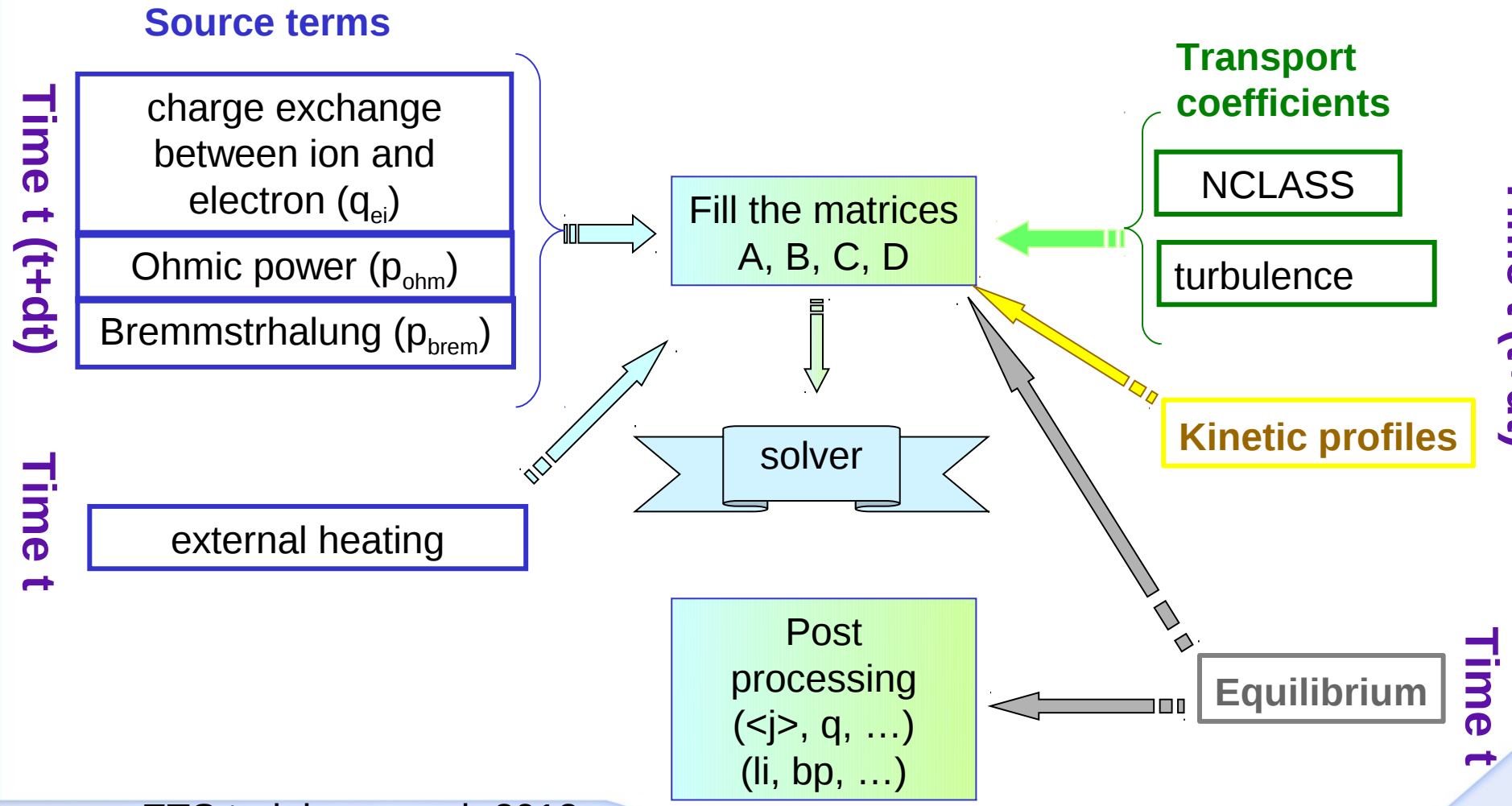
$$r_1 = \frac{n_{\min 1}}{n_{main}}$$

$$r_2 = \frac{n_{\min 2}}{n_{main}}$$

$$r_3 = \frac{n_{imp\,1}}{n_{imp\,2}}$$

ETS, SOLVER (4/4)

Schematic view of the solver



V&V

In collaboration with ISM, comparison with existing codes (ASTRA, CRONOS, JETTO, …) on JET Shot #77922

Prescribed quantities

- **Heating source (gaussian profile)**
- **electron density**
- **equilibrium**

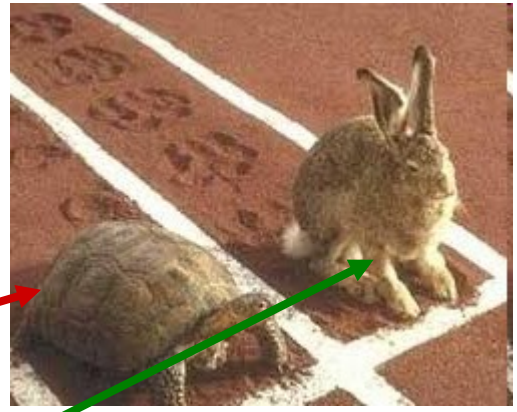
Computation

- **Ohmic Power, Equipartition, Bremsstrahlung**
- **Transport Model (NCLASS + B/gB)**
- **Ψ, Te, Ti**

Good agreement between ETS and existing codes

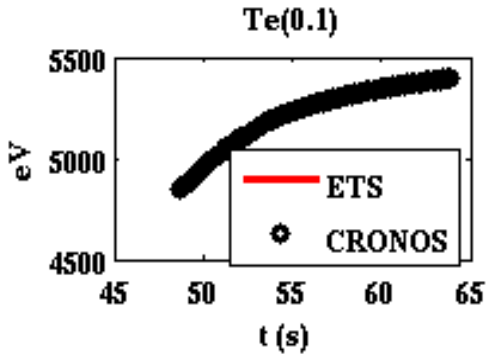
Acceleration

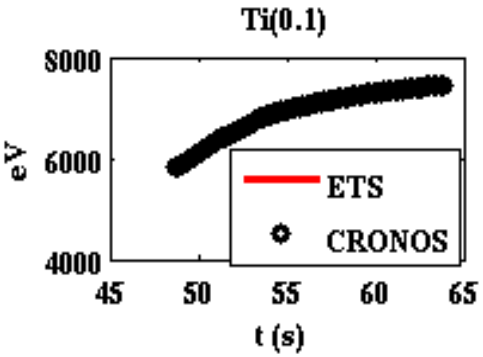
Test of the workflow with cache

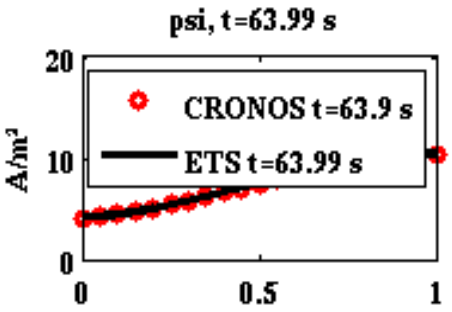


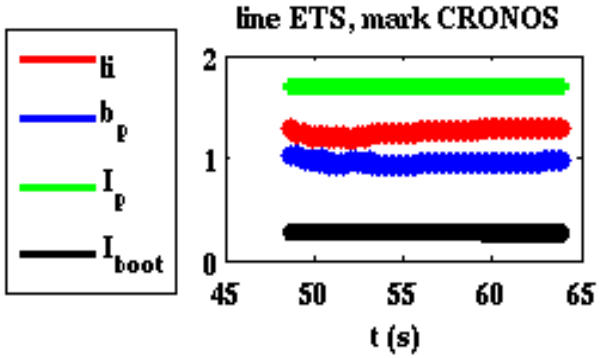
Ets_c w/o cache	6h29	2,9 Go
Ets_c with cache	29min	68M
Gain	13.4	42.6

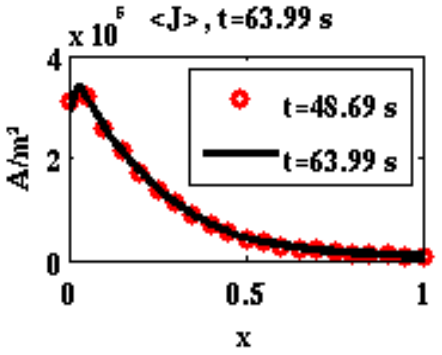
Comparison with CRONOS

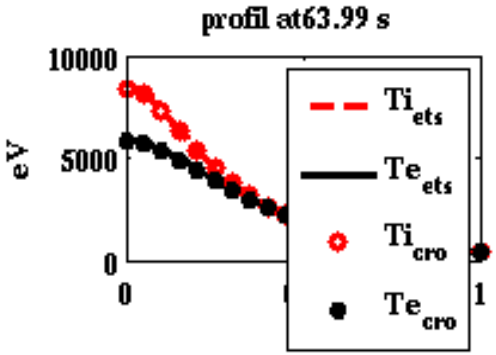


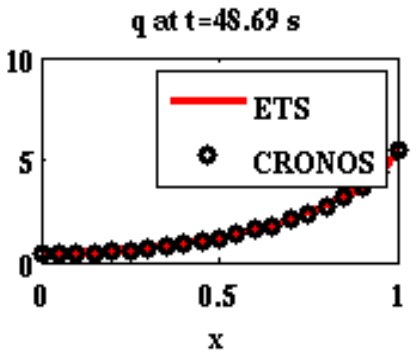


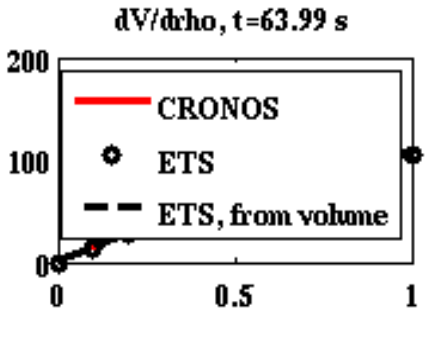


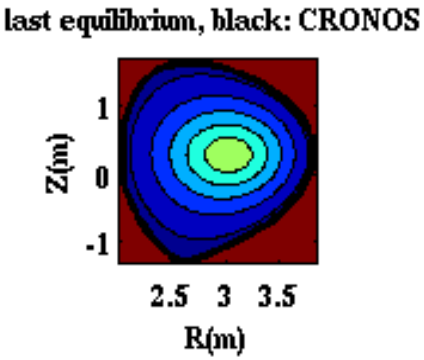




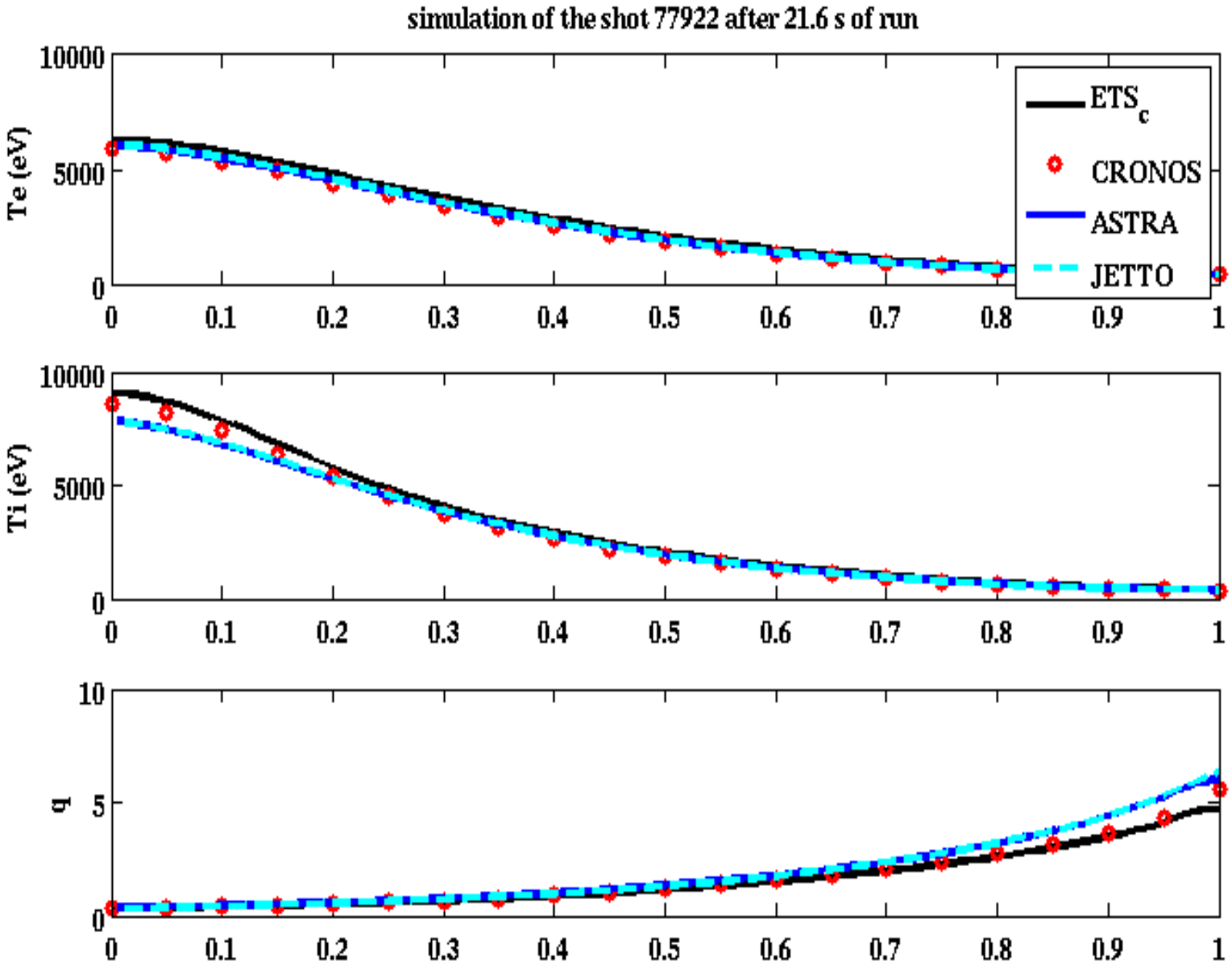








ISM benchmark



II. KEPLER WORKFLOW

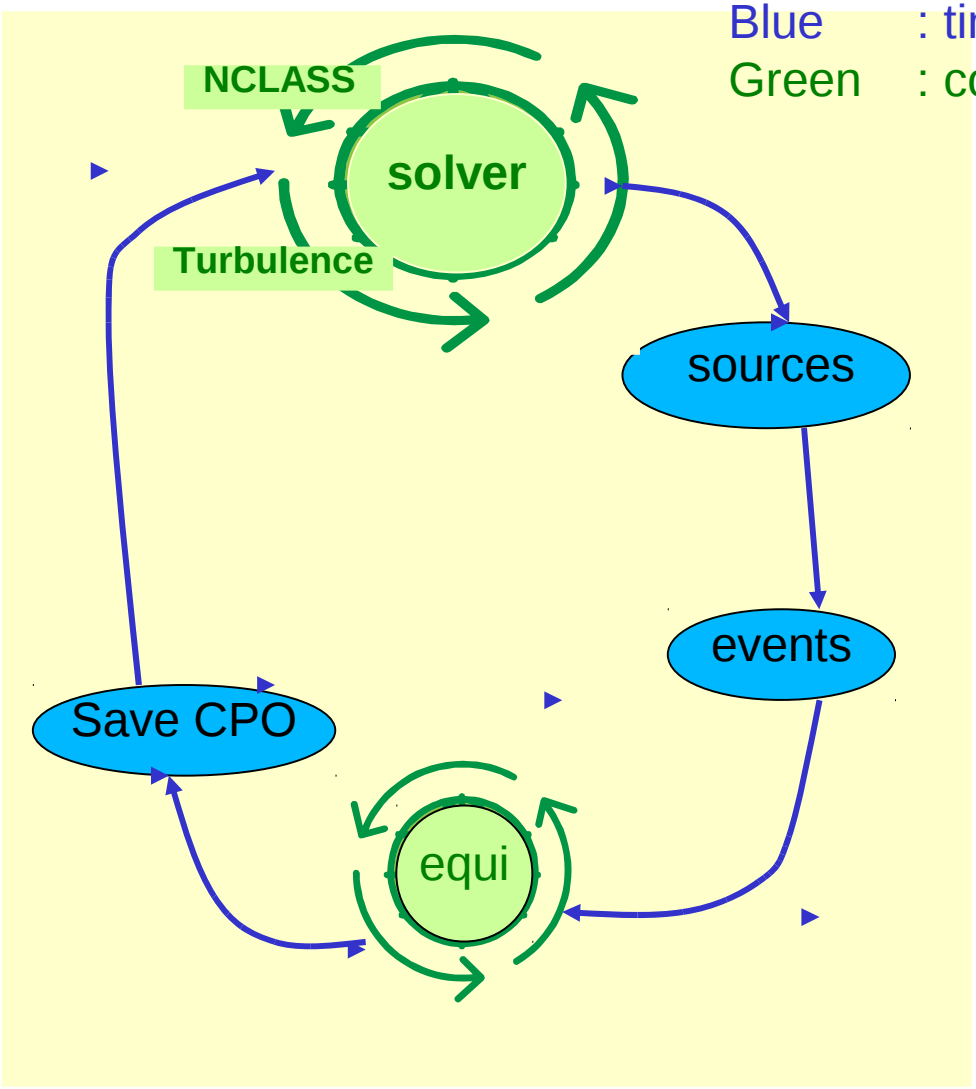
ETS WORKFLOW

- The KEPLER workflow is adapted for the development
  - Reading input data and prescribed profiles from a Cronos input file
  - Debug mode allowed: actors are compiled in debug mode, and can be executed under Totalview
  - Configure the workflow from Kepler: access to workflow/actors parameters, choice of actors (equilibrium, transport terms, …)
- Configure a run can be done using ISE but we still are in development phase that's why we don't show the ISE interface
- We tried to limit the workflow complexity by limiting the number of workflow level to 4
- All the configurable workflow parameters are at the toplevel workflow

Schematic view of the workflow

Blue : time loop
Green : convergence loop

Initialization
Reading file_ETS
from
a cronos input file



External sources and equilibrium are outside the transport convergence loop

TOPLEVEL WORKFLOW

EUROPEAN TRANSPORT SOLVER

Workflow parameters

Time parameters
- tbegin_in: 48.1
- tend_in: 50.0
- dtmin_in: 1.0e-05
- dtmax_in: 0.01

output shot
- runwork_in: 2
- runout_in: runwork_in

saving parameters
- sourcecountLH_in: 50
- savenumber_in: 50
- equicount_in: 1

DDF Director

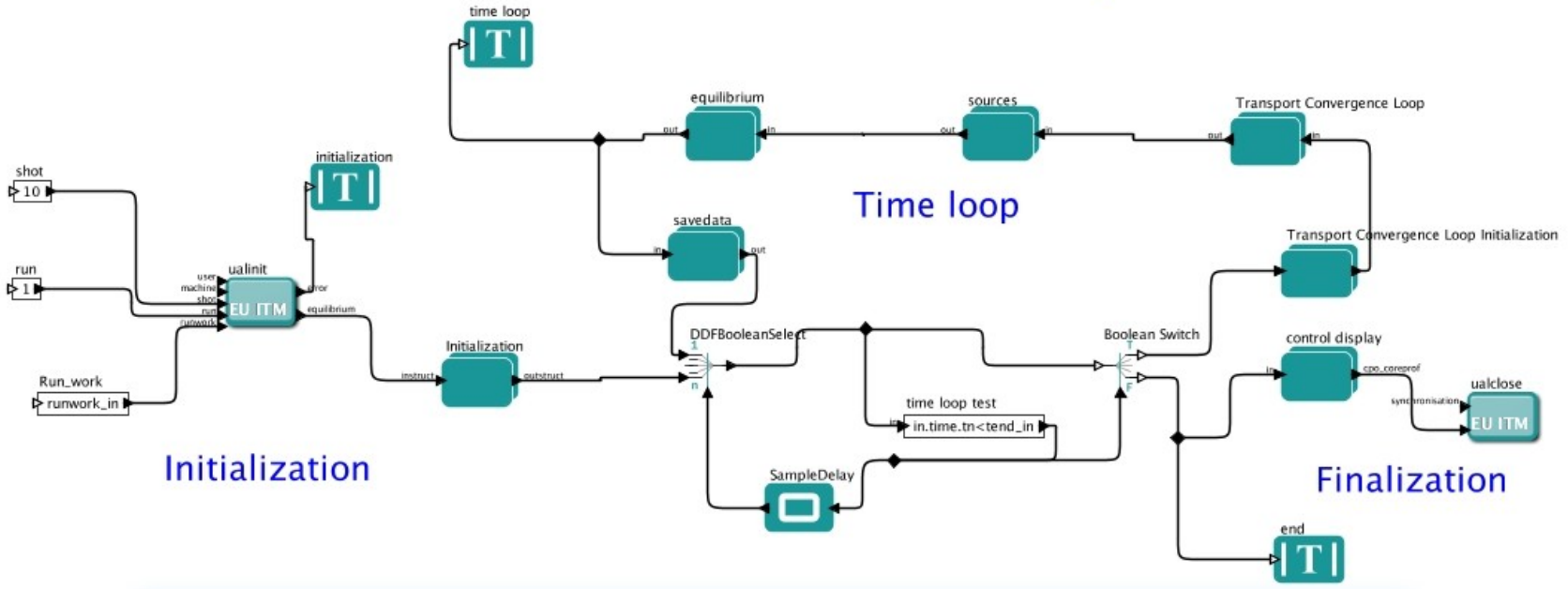
Provenance Recorder

Convergence parameters
- iterationmax_in: 15
- tolerance_in: 1.0e-6

Equation parameters
- ElectronHeatEquation_in: 1
- IonHeatEquation_in: 1
- ElectronDensityEquation_in: 0

Sources parameters
- sourceLHwithfeedback_in: 0
- PrescribedSourceElectronDensity_in: 1
- PrescribedElectronHeatProfile_in: 1
- PrescribedIonHeatProfile_in: 1

Equilibrium parameters
- PrescribeEquilibrium_in: 0
- EquilibriumConvergenceTolerance_in: 5e-2



ETS WORKFLOW : time management

- The time step is **variable**
- It is managed by a unique composite actor called time manager placed in the transport convergence loop
- Changing rules based on the convergence iterations number:
  - fast convergence → increase the time step
  - medium convergence → keep the same time step
  - slow convergence → decrease the time step

- Implementation in CPOs terms
  - 2 temporary cpos: one for the current time step call tn, the other for the next time step tn +1. Temporary CPOS are stored in different occurrencies.
  - Once a time step is computed after some convergence, it can be stored if the time follow some rules. Final results are stored in occurrence 0.

ACTORS

Actors and workflows tested, Actors and workflows under validation

- Equilibrium (HELENA21, HELENA, CHEASE, EMEQ)
- Neoclassic (NCLASS, NEOWES)
- Transport (B/GB, GLF23, COPPITANG, ETAIGB)
- IMP5 workflow (NEMO, RISK)
- NTM
- Sawteeth

The workflow can deal with the 2 ETS solvers (OK in 4.08b but not yet in 4.09a)

HOW TO CONFIGURE A RUN 1/3

All the configurable workflow parameters are at the toplevel

Time parameters

- tbegin_in: 48.1
- tend_in: 50.0
- dtmin_in: 1.0e-05
- dtmax_in: 0.01

output shot

- runwork_in: 2
- runout_in: runwork_in

saving parameters

- sourcecountLH_in: 50
- savenumber_in: 50
- equicount_in: 1

Convergence parameters

- iterationmax_in: 15
- tolerance_in: 1.0e-6

Equation parameters

- ElectronHeatEquation_in: 1
- IonHeatEquation_in: 1
- ElectronDensityEquation_in: 0

Sources parameters

- sourceLHwithfeedback_in: 0
- PrescribedSourceElectronDensity_in: 1
- PrescribedElectronHeatProfile_in: 1
- PrescribedIonHeatProfile_in: 1

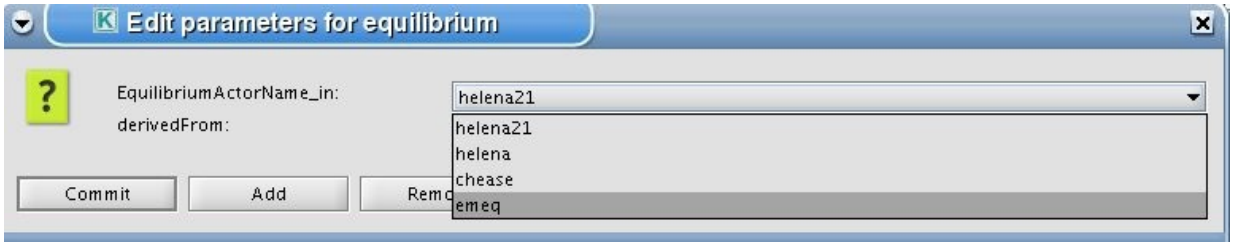
Equilibrium parameters

- PrescribeEquilibrium_in: 0
- EquilibriumConvergenceTolerance_in: 5e-2

More sophisticated rules for triggering actors will be implemented and graphical interface will be developed in ISE for configuring these rules.

HOW TO CONFIGURE A RUN 2/3

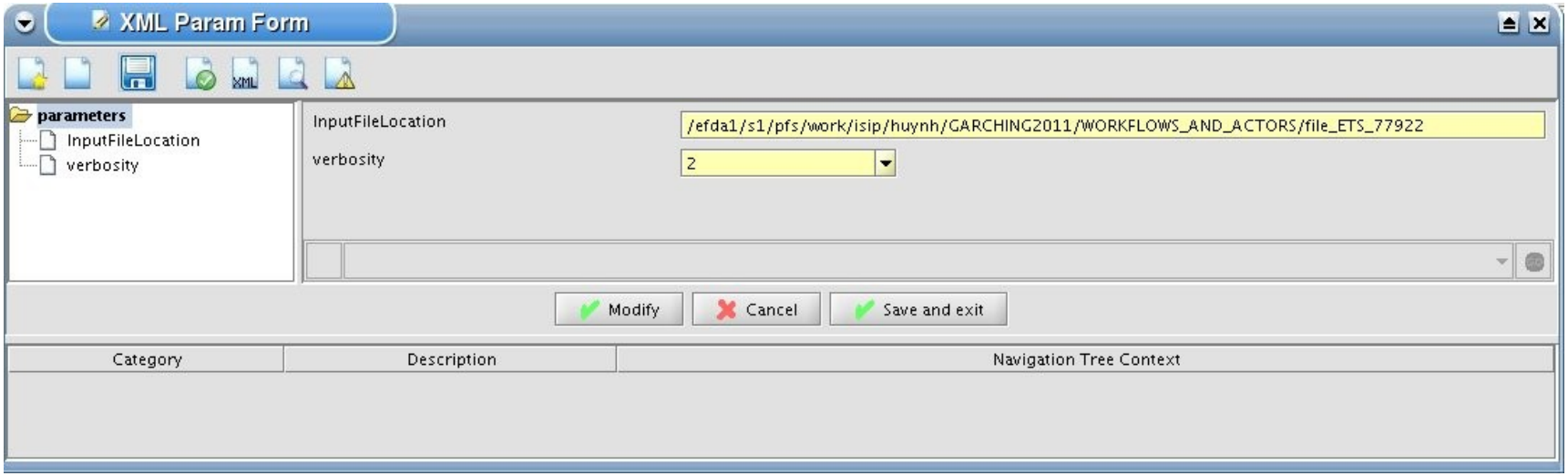
Change the equilibrium code by double clicking on the equilibrium composite actor



Change the transport coefficients code and neoclassical terms code by double clicking on transport convergence loop

HOW TO CONFIGURE A RUN

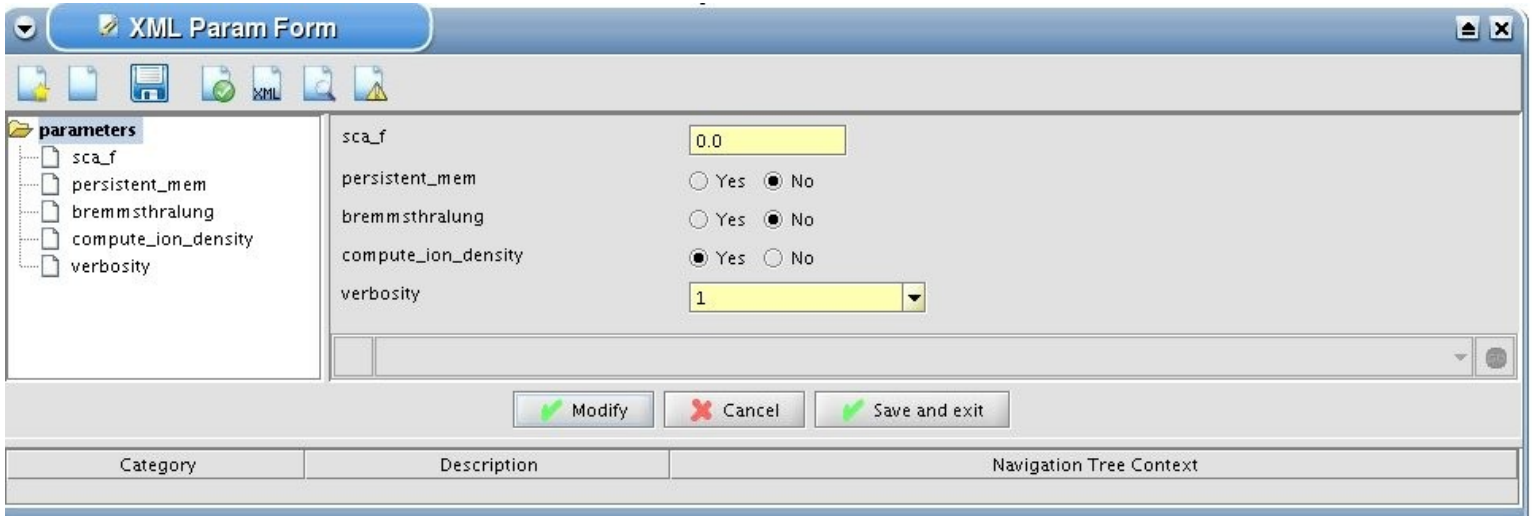
Change the specific code parameter of an actor by double clicking on the actor and press Edit Code Parameters, for instance the initdata actor (in Initialization subworkflow)



InputFileLocation is the location of the input data file from Cronos. You will see in the pratice part different files examples.

HOW TO CONFIGURE A RUN 3/3

Specific code parameters for the ETS actor (Transport Convergence Loop/TransportSolver/etssolver)



- sca_f is a real scalar between 0.0 (fully implicit) and 0.5 (Crank-Nicholson schema)
- Persistent memory : compute again the tn matrices or use the persistent memory
- Bremmsthralung : compute or not the internal Bremmsthralung source
- compute_ion_density : compute or not the ion densities

HOW TO ADD YOUR ACTOR 1/4

Adding an actor which is not an equilibrium code nor a transport coefficient nor a neoclassical code is relatively complex and it is better to discuss first with the ETS developpement team

The Kepler bundle is a record that contains the references to the plasma state that are needed for your code. This bundle is transmitted from one composite actor to another. It is currently composed by the following but will evolve.

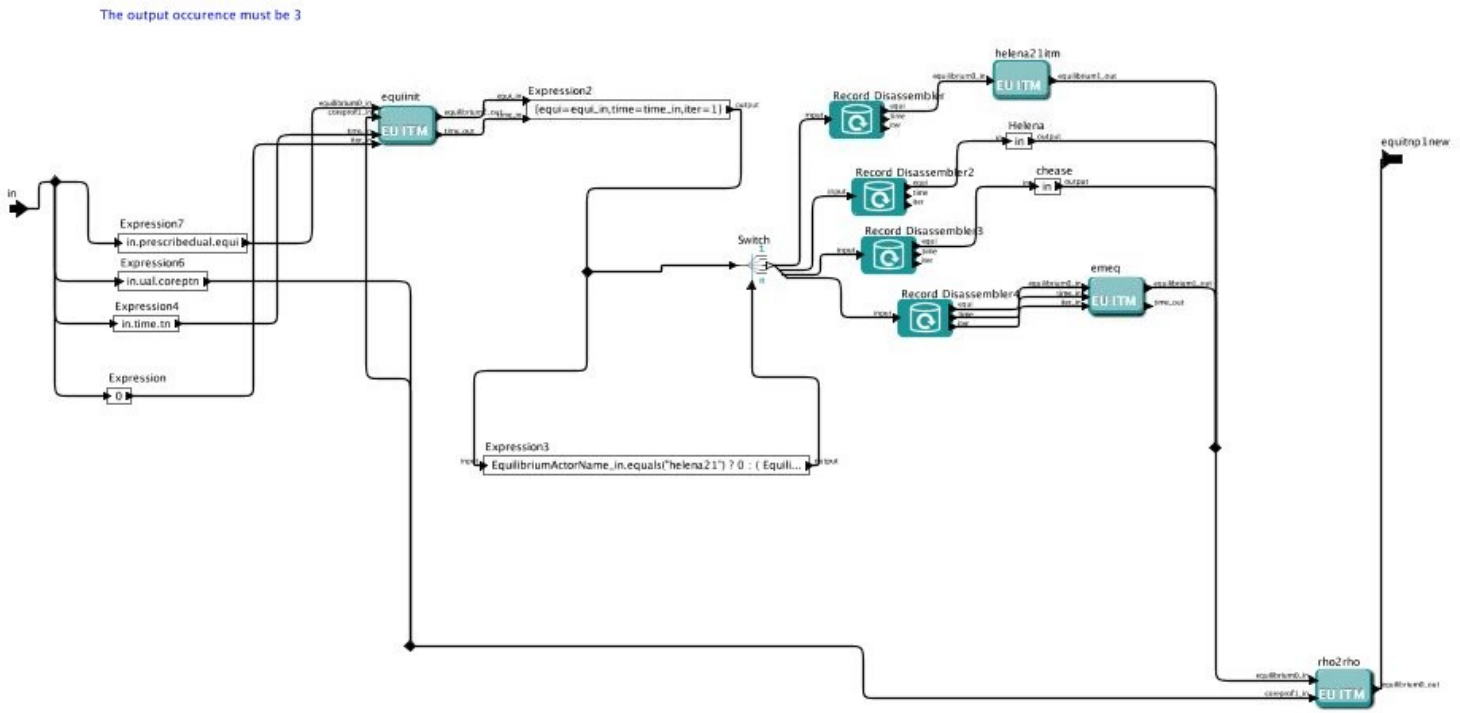
time composed by : tn,tnp1,dt,iterloop
cvg composed by : eps,iter,count,bestvalue
ual composed by : coreptn,coreptnp1,neotn,neotnp1,
corettn,corettnp1,equitn,corestn,antentn
prescribedual composed by : equi,corep,coret,anten

We use different occurrences : Occurrence 2 = tn
Occurrence 3 = tnp1
Occurrence 1 = prescribed cpo
Occurrence 0 = final results

HOW TO ADD YOUR ACTOR 2/4

For an equilibrium actor

- Go to equilibrium/equilibrium convergence/compute new equilibrium subworkflow

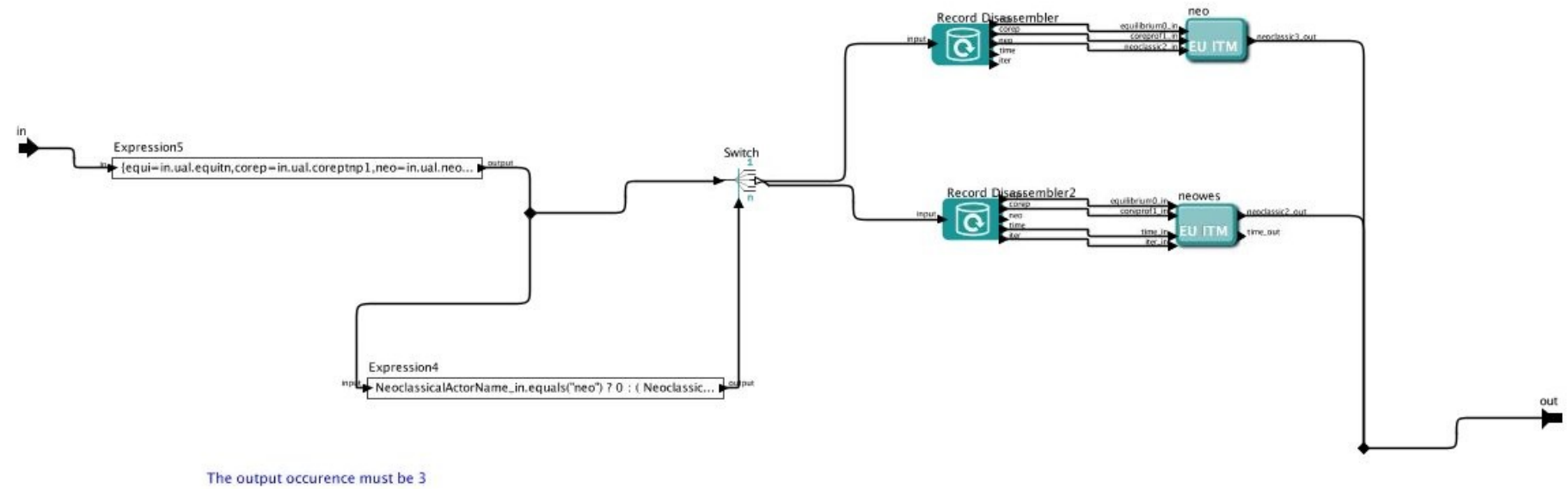


- Replace one equilibrium actor by your actor and don't forget to set the output equilibrium occurrence to 3

HOW TO ADD YOUR ACTOR 3/4

For a neoclassical actor

- Go to Transport Convergence Loop/Neoclassical Terms subworkflow

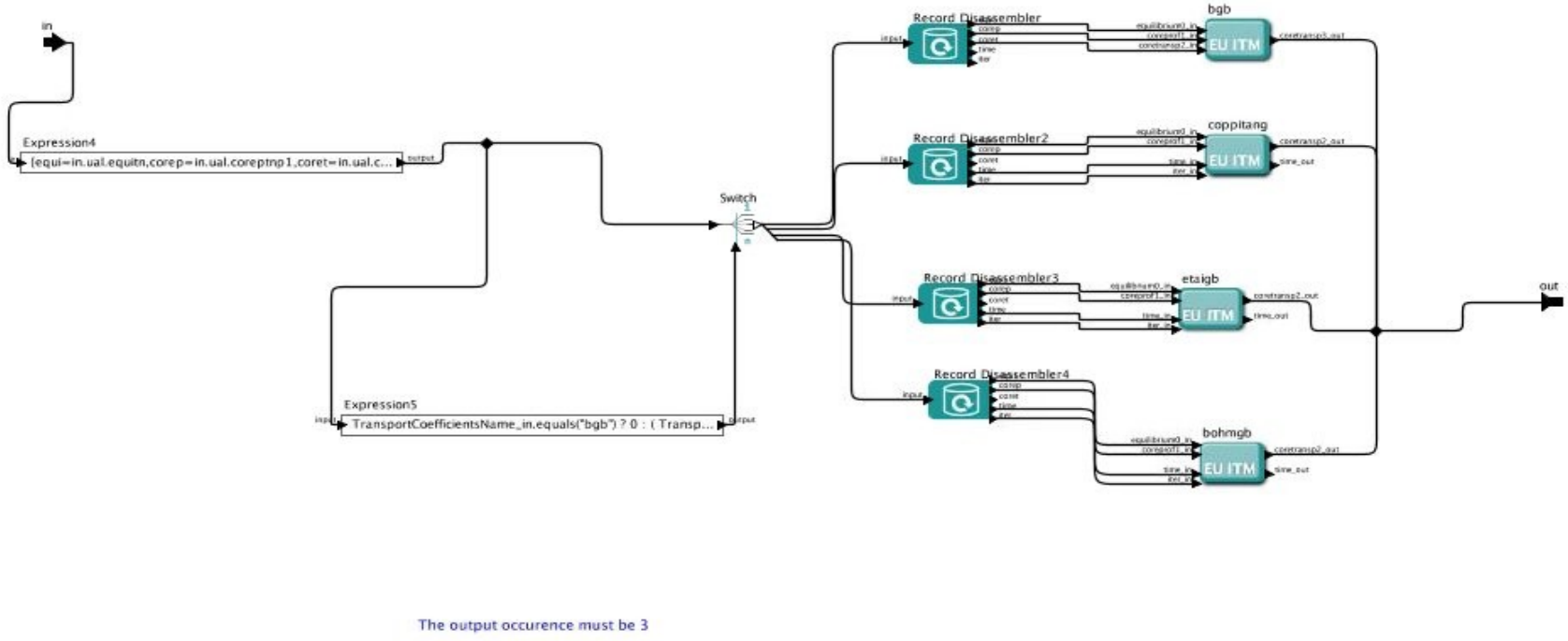


- Replace one neoclassical actor by your actor and don't forget to set the output neoclassical occurrence to 3

HOW TO ADD YOUR ACTOR 4/4

For transport coefficients actor

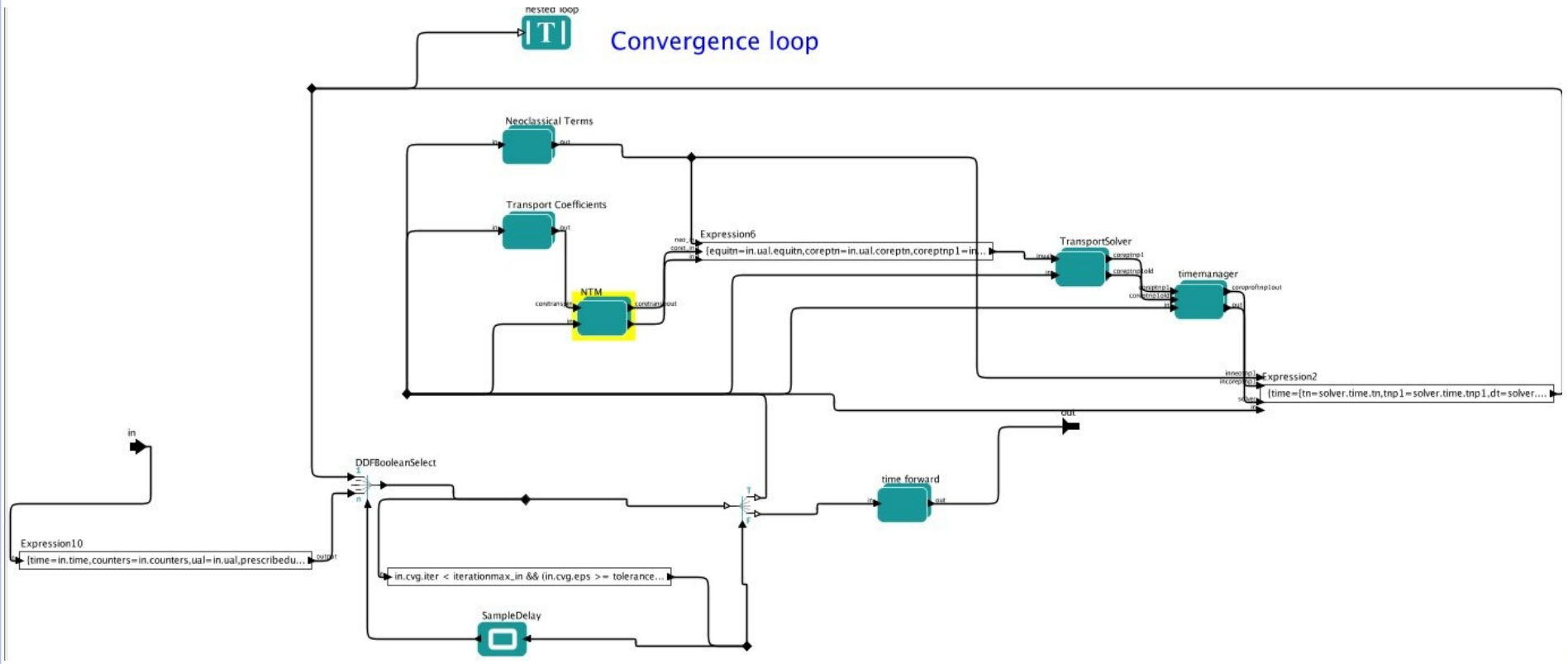
- Go to Transport Convergence Loop/Transport Coefficients subworkflow



- Replace one transport coefficents actor by your actor and don't forget to set the output neoclassical occurrence to 3

An example of inserting NTM actor in the ETS workflow

Work in progress done with S.NOWAK and O.SAUTER



III. Visualization and Post-treatement

You will use in the practice different methods to visualise results stored in the UAL database :

- freely :
  - using matlab interpretor
  - using python interpretor
- among predefined choices :
  - using VCOS [C-program]

Part 1 : UAL-access

Python syntax (script VISU/vcpo.py)

```
from vmod import *

shot = 10
run  = 4

ob   = ual.itm(shot,run,shot,0)
ob.open()
ob.coreprofArray.get()
ob.close()

nt   = len(ob.coreprofArray.array)
nrho = len(ob.coreprofArray.array[0].te.value)

# comment line must begin with a « # »  character
# array indices start from 0 and are inclosed in square brakets

time = np.zeros([nt])
ip   = np.zeros([nt])
li   = np.zeros([nt])

psi  = np.zeros([nt,nrho])
rho  = np.zeros([nt,nrho])
te   = np.zeros([nt,nrho])
jtot = np.zeros([nt,nrho])

legs = []

for kt in range(nt) :
  coreprof   = ob.coreprofArray.array[kt]
  time[kt]   = coreprof.time
  psi[kt,:]  = coreprof.psi.value[:]
  rho[kt,:]  = coreprof.rho_tor_norm[:]
  ip[kt]     = coreprof.globalparam.current_tot
  li[kt]     = coreprof.globalparam.li
  te[kt,:]   = coreprof.te.value[:]
  jtot[kt,:] = coreprof.profiles1d.jtot.value[:]
  legs.append("t = %(#).3f s" % {'#': time[kt]})

# you must indent (using tab) loop body
```

Matlab syntax (script VISU/vcpo.m)

```
shot     = 10;
run      = 4;

ob       = euitm_open('euitm',shot,run);
acoreprof = euitm_get(ob,'coreprof');
euitm_close(ob)

nt       = length(acoreprof);
nrho     = length(acoreprof(1).te.value);

% comment line must begin with a « % »  character
% array indices start from 1 and are inclosed in parenthesis


% array initialisation are not mandatory


legs     = {};
for kt=[1:nt]

  coreprof   = acoreprof(kt);
  time(kt)   = coreprof.time;
  psi(kt,:)  = coreprof.psi.value;
  rho(kt,:)  = coreprof.rho_tor_norm;
  ip(kt)     = coreprof.globalparam.current_tot;
  li(kt)     = coreprof.globalparam.li;
  te(kt,:)   = coreprof.te.value;
  jtot(kt,:) = coreprof.profiles1d.jtot.value;
  legs       = {legs{:} ['t = ' num2str(time(kt)) ' s']};

end
```

Part 2 : plotting

Python syntax (script VISU/vcpo.py)

```
fig0 = plt.figure(1)
fig0.clf()
plt.plot(time,ip,'-or')
plt.axis([time[0],time[nt-1],0,2.0e6])
plt.title(("shot %(A)d,run %(B)d : plasma current" %{"A":shot,"B":run}))
plt.ylabel('ip')
plt.xlabel('time')
fig1 = plt.figure(2)
fig1.clf()
plt.plot(np.transpose(psi),np.transpose(jtot))
plt.legend(legs)
plt.title(("shot %(shot)d,run %(run)d : total current" % {"shot":shot,"run":run}))
plt.ylabel('jtot')
plt.xlabel('psi')
plt.show()
```

Matlab syntax (script VISU/vcpo.m)

```
figure(1)
clf
plot(time,ip,'-or')
axis([time(1) time(end) 0 2.0e6])
title(['shot ' int2str(shot) ',run ' int2str(run) ' : plasma current'])
ylabel('ip')
xlabel('time')
figure(2)
clf
plot(psi.',jtot.')
legend(legs)
title(['shot ' int2str(shot) ',run ' int2str(run) ' : total current'])
ylabel('jtot')
xlabel('psi')
```

vmod.py contain python modules needed to read UAL-database and plot :

```
#!/usr/bin/env python
import matplotlib.pyplot as plt        # plot module
import numpy as np                     # n-dimensional array module
import ual                             # ual database access module
```

Documentation can be found at url :

- python [current installed version] : http://docs.python.org/release/2.5.1/
- matplotlib : http://matplotlib.sourceforge.net/
- numpy : http://numpy.scipy.org/
- ual : https://www.efda-itm.eu/ITM/imports/isip/public/isip_UAL_User_Guide.pdf

VCOS an interactive visualisation compiled program :

- C-language
- UAL access through C++ module
- communicate whith a CGI server (spcgi.cgi need local apache server running) :
  - => XHTML-Forms with embedded SVG on Firefox browser

(XML compliant, W3C specifications)

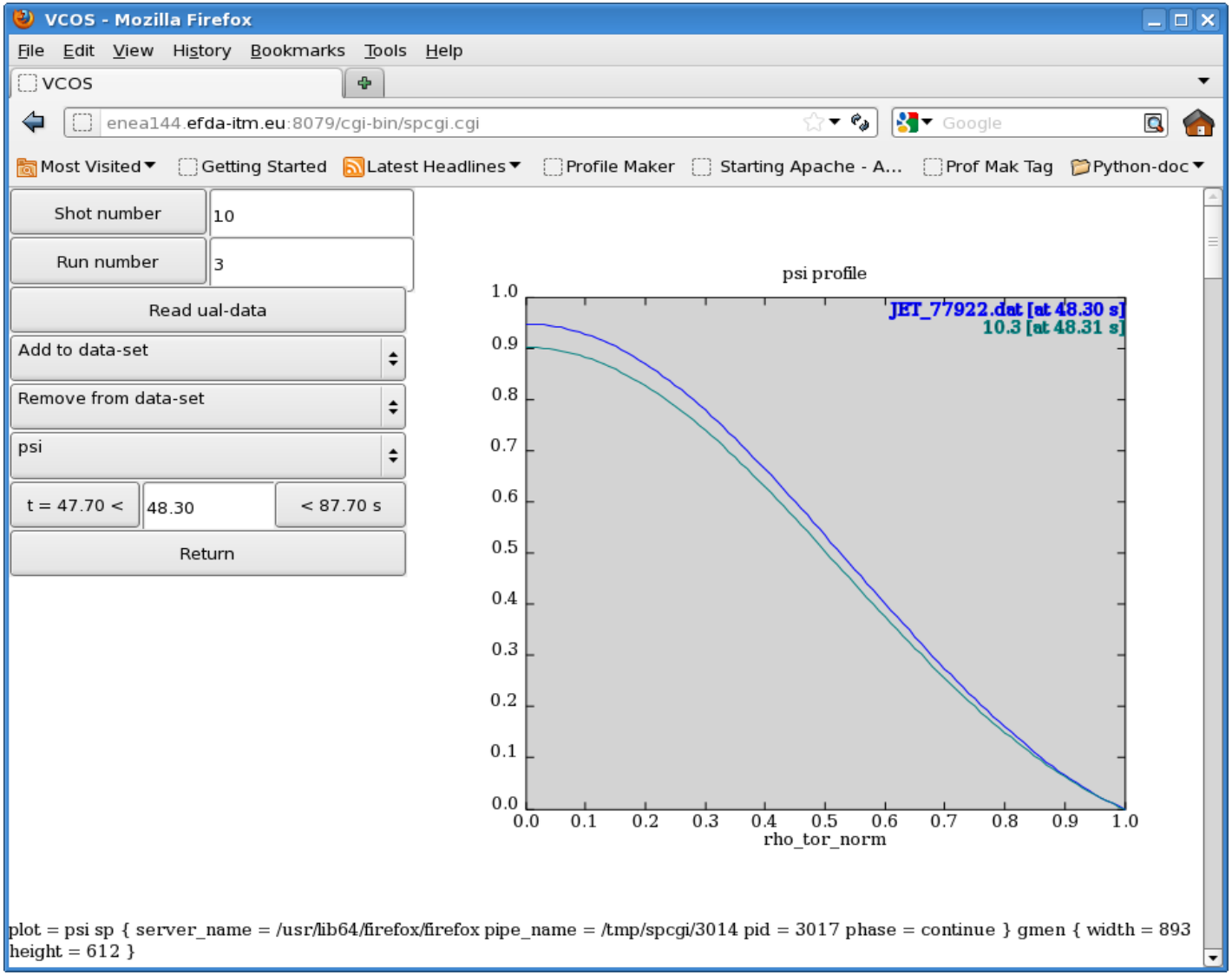
  - <= read edited parameters using POST method

Actual features : **Validation of an ETS simulation by comparison with CRONOS**

- read UAL-data (coreprof-CPO)
- read data from Matlab files after conversion in ASCII-file :

  cronos-output.mat --[zmat2dat.m script]--> cronos-output.dat
- compare data for various shot / run or CRONOS output Matlab files :
  - profiles at fixed time for 1D data
  - time evolution for global data

Launching VCOS : vcos [-file JET_77922.dat]



IV. Practice

GET ETS workflow and actors

Get the material for the ETS workflow by copying GARCHING2011 on your HOME directory or WORK directory

cp -r ~huynh/public/GARCHING2011/ ~

In this directory, you can find :

- the instructions of importing ETS actors are written in the README file.
- the ETS workflow stored in WORKFLOWS_AND_ACTORS
- different input files generated and stored in INPUT_FILES

EXERCISE 1
(current diffusion equation, interpretative mode)

Modify the workflow :

- Update the InputFileLocation parameter of the “initdata” actor with the location of file_ETS_77922
- Change the tend_in value to 48.2s
- Put the interpretative mode (solved only Ψ)
- Verify the mode “Prescribed external source”
- Modify the runwork_in to 3
- save the workflow into EXE1

Execute the workflow

- Don't forget to type itmgo befor launching the execution

Visualization of the results (comparison with CRONOS) :

- vcos -file ISM_77922_new_9_ter_resultat.dat

EXERCISE 2
(more diffusion equations, predictive mode)

Modify the workflow :

- Put the predictive mode (solved Ψ, Te, Ti)
- Verify the transport model (Bohm/gyroBohm)
- modify the runwork_in to 4
- save the workflow into EXE2

Execute the workflow

Visualization of the results (comparison with CRONOS and with the interpretative mode)

EXERCICE 3
(predictive mode, equilibrium convergence)

Modify the workflow :

-Keep the predictive mode (solved Ψ, Te, Ti)
-Decrease the dtmin_in to 0.03
-Computation of the equilibrium (put equilibrium time step to 1)
-Verify prescribed equilibrium
-Modify the runwork_in to 5
-Save the workflow into EXE3

Execute the workflow

Visualization of the results (comparison with CRONOS and with the interpretative mode)

EXERCISE 4
(predictive mode, source term + feedback)

Modify the workflow :

- keep the predictive mode (solved Ψ, Te, Ti)
- prescribed equilibrium
- turn on LH source module, allowed feedback control
- modify the runwork_in to 6
- save the workflow into EXE4

Execute the workflow

Visualization of the results (comparison with CRONOS and with the interpretative mode)