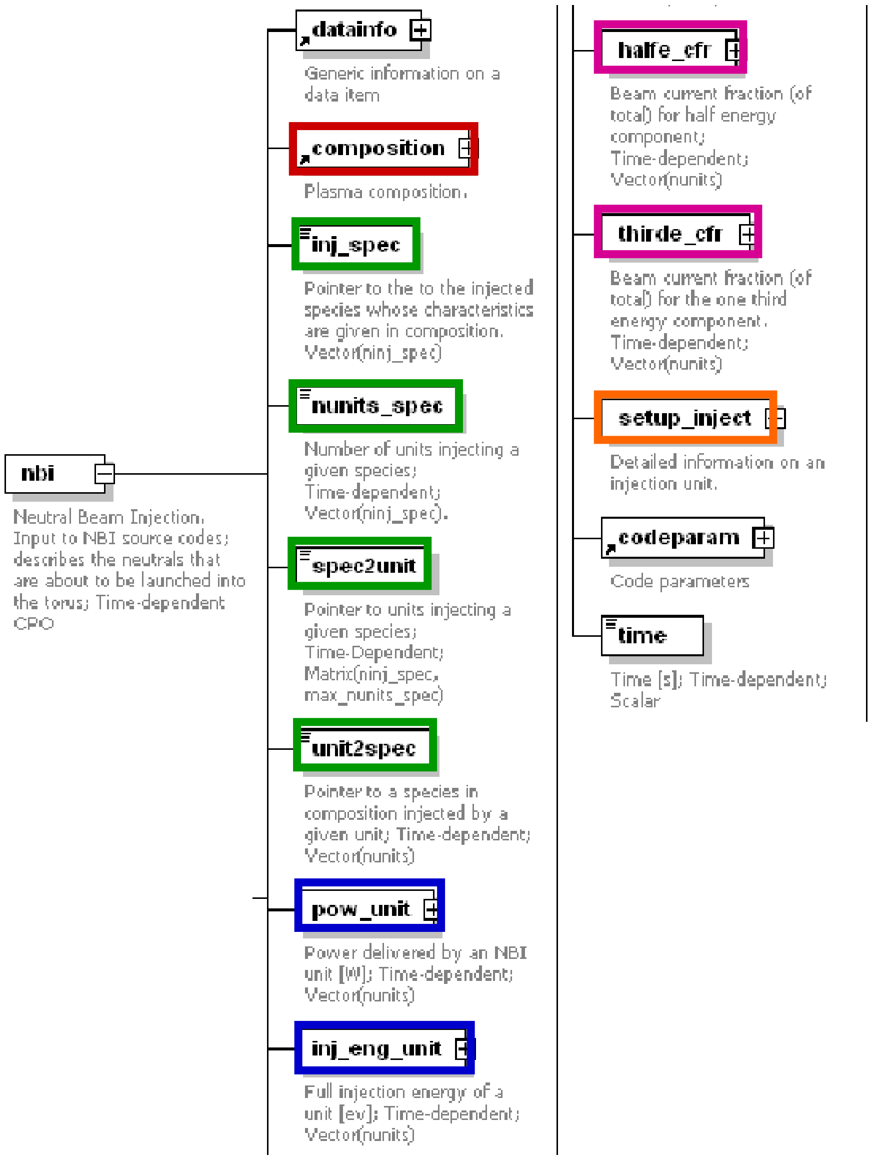
Neutral Beam Injection in ITM

Mireille Schneider, L.-G. Eriksson

OUTLINE:

- Neutral beam CPOs ***nbi*** and ***distsource***
- Description of the NEMO NBI source model and input/output
- NBI setup routine ⇨ fills input CPO ***nbi***
- Standalone Test Bed for NEMO ⇨ produces CPO ***distsource***
- The Kepler NBI test workflow
- Summary and prospects

The NBI input CPO ***nbi***



nbi
Neutral Beam Injection. Input to NBI source codes; describes the neutrals that are about to be launched into the torus; Time-dependent CPO

datainfo
Generic information on a data item

composition
Plasma composition.

inj_spec
Pointer to the to the injected species whose characteristics are given in composition. Vector(ninj_spec)

nunits_spec
Number of units injecting a given species; Time-dependent; Vector(ninj_spec).

spec2unit
Pointer to units injecting a given species; Time-Dependent; Matrix(ninj_spec, max_nunits_spec)

unit2spec
Pointer to a species in composition injected by a given unit; Time-dependent; Vector(nunits)

pow_unit
Power delivered by an NBI unit [W]; Time-dependent; Vector(nunits)

inj_eng_unit
Full injection energy of a unit [ev]; Time-dependent; Vector(nunits)

halfe_cfr
Beam current fraction (of total) for half energy component; Time-dependent; Vector(nunits)

thirde_cfr
Beam current fraction (of total) for the one third energy component. Time-dependent; Vector(nunits)

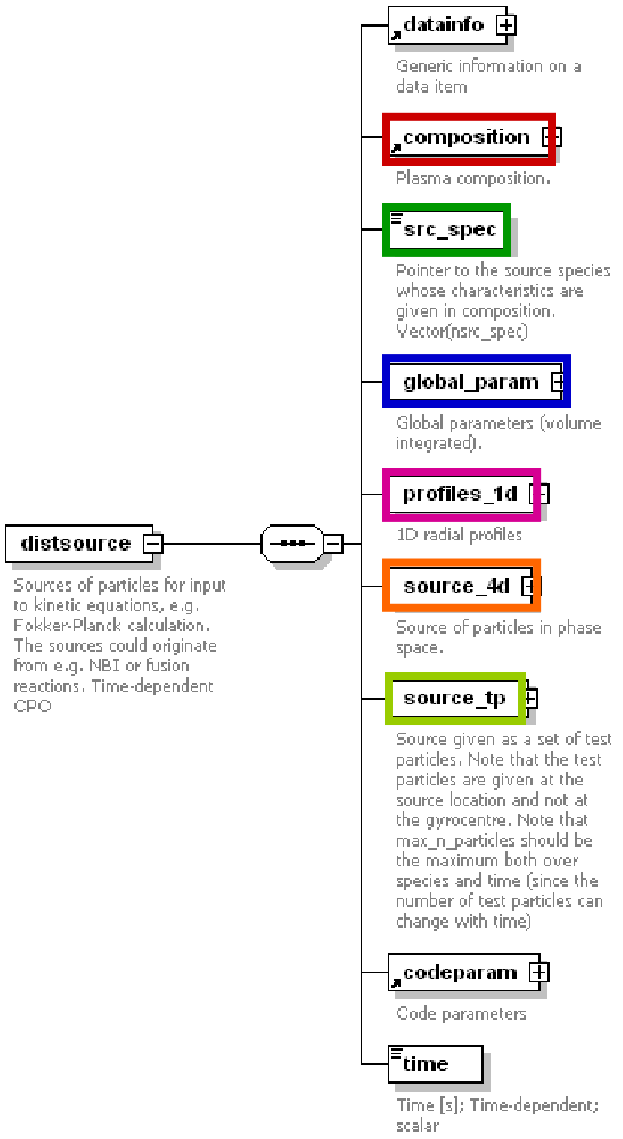
setup_inject
Detailed information on an injection unit.

codeparam
Code parameters

time
Time [s]; Time-dependent; Scalar

- Plasma composition
- Pointers to injected and plasma species
- Power and energy for each injection unit
- Particle fraction for each energy
- Geometry of the injector:
  ⇨position, tangency radius, angle,direction, divergence, focal lengths, beamlet positions.

The NBI output CPO ***distsource***



distsource

Sources of particles for input to kinetic equations, e.g. Fokker-Planck calculation. The sources could originate from e.g. NBI or fusion reactions. Time-dependent CPO

datainfo

Generic information on a data item

composition

Plasma composition.

src_spec

Pointer to the source species whose characteristics are given in composition. Vector(nsrc_spec)

global_param

Global parameters (volume integrated).

profiles_1d

1D radial profiles

source_4d

Source of particles in phase space.

source_tp

Source given as a set of test particles. Note that the test particles are given at the source location and not at the gyrocentre. Note that max_n_particles should be the maximum both over species and time (since the number of test particles can change with time)

codeparam

Code parameters

time

Time [s]; Time-dependent; scalar

- Plasma composition
- Pointers to injected species
- Scalar quantities: source power and source rate
- 1D profiles: toroidal flux coord., power density, source rate
- 4D source matrix: source of particles (ndim1,ndim2,ndim3,ndim4) and its associated vectors
- Source as a set of test particles and its associated vectors

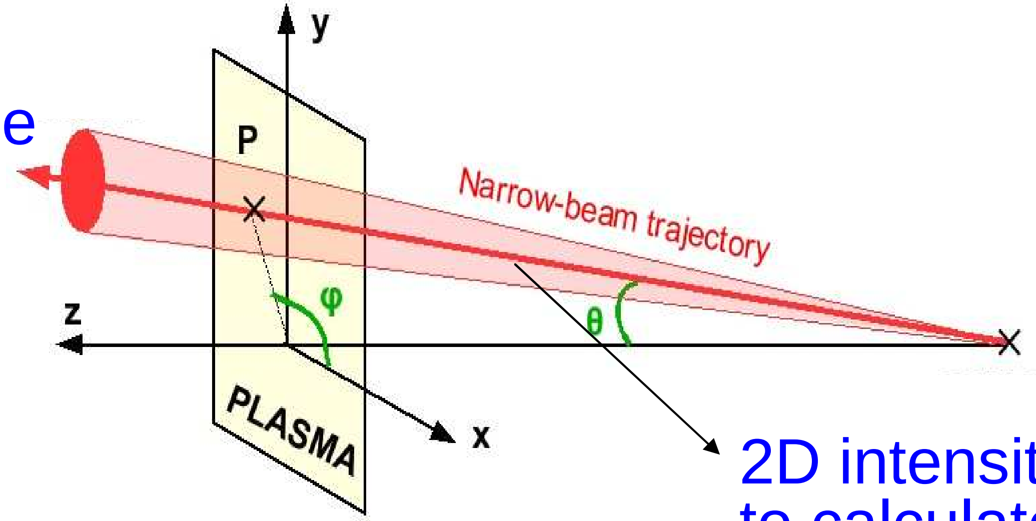
The NBI source model in NEMO

NEMO is based on the narrow-beam model first seen in [Y. Feng et al, Comp. Phys. Comm. 88 (1995) 161-172].

Neutral beam = punctual source
+ angular divergence
+ 2D intensity distribution.

⇨Simplified geometry:

Angular divergence



Punctual source

2D intensity distribution to calculate

NEMO input

Plasma geometry:

- Plasma major & minor radius
- SOL radius or (R,Z) wall coordinates
- 2D toroidal flux coordinate in (R,Z) grid
- B_R, B_Z, B_φ of the magnetic field
- Direction of B_T and I_P

Beam geometry:

- Number of beams in the injector
- Tangency radius of each beam
- X, Y, Z, R coordinates of each beam
- Angle between beam and Z-midplane
- Horizontal & vertical focal distances
- Beam divergence (rad)
- Width & height of each beam source
- Directivity

Plasma kinetics:

- Radial coordinate vector
- Plasma volume profile
- n_e, n_i, T_e, T_i profiles
- Mass & charge of plasma ion species

Beam parameters:

- Power on each beam
- Energy of injected neutrals
- Particle fraction per energy
- Mass & charge of injected neutrals

NEMO output

Profiles:

- Radius coordinate vector
- Heating profile per energy and beam
- Power profile per energy and beam
- Pitch profile per energy
- Torque profile

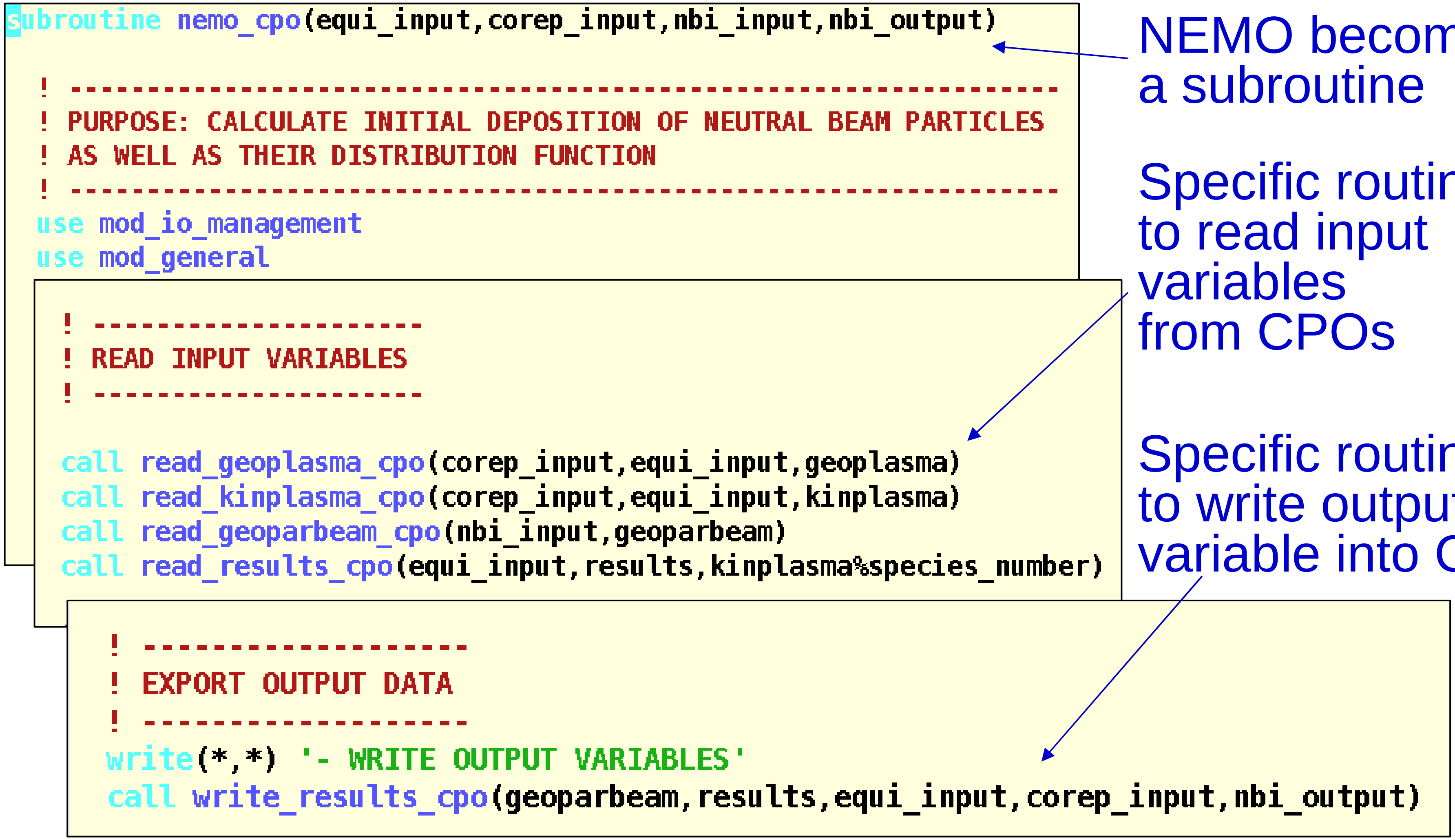
Scalar quantities:

- Particle shinethrough per beam
- Power shinethrough per beam

Deposition matrix and associated vectors:

- Deposition 5D matrix (beam,E,R,Z,pitch)
- X, Y, Z, R, pitch vectors associated to birth matrix

CPO implementation in NEMO



```
subroutine nemo_cpo(equi_input,corep_input,nbi_input,nbi_output)

  ! ------------------------------------------------------------------
  ! PURPOSE: CALCULATE INITIAL DEPOSITION OF NEUTRAL BEAM PARTICLES
  ! AS WELL AS THEIR DISTRIBUTION FUNCTION
  ! ------------------------------------------------------------------
  use mod_io_management
  use mod_general
```

```
  ! ---------------------
  ! READ INPUT VARIABLES
  ! ---------------------

  call read_geoplasma_cpo(corep_input,equi_input,geoplasma)
  call read_kinplasma_cpo(corep_input,equi_input,kinplasma)
  call read_geoparbeam_cpo(nbi_input,geoparbeam)
  call read_results_cpo(equi_input,results,kinplasma%species_number)
```

```
  ! ------------------
  ! EXPORT OUTPUT DATA
  ! ------------------
  write(*,*) '- WRITE OUTPUT VARIABLES'
  call write_results_cpo(geoparbeam,results,equi_input,corep_input,nbi_output)
```

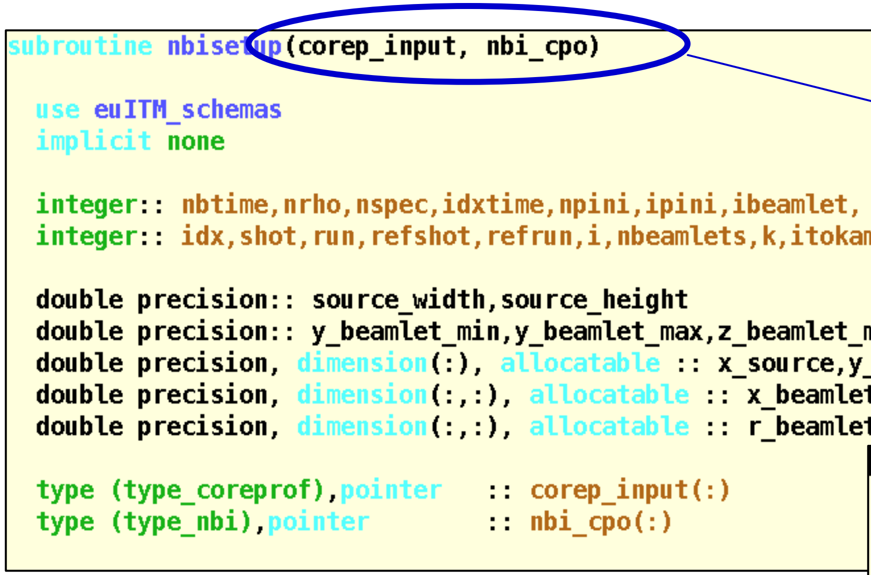
NEMO becomes a subroutine

Specific routines to read input variables from CPOs

Specific routine to write output variable into CPO

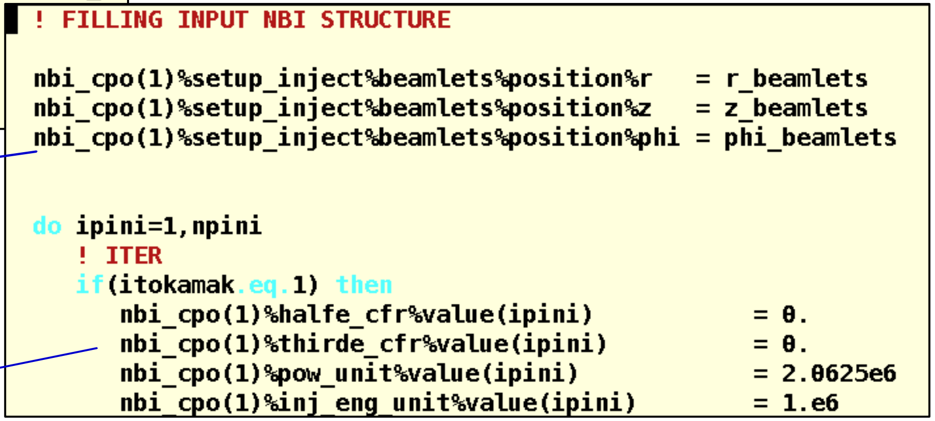
The NBI setup routine

Purpose: to fill in the ***nbi*** CPO structure needed as input for an NBI source code (note: official machine descriptions for NBI injectors are not yet available; for now the relevant data are therefore set in nbisteup)



```
subroutine nbisetup(corep_input, nbi_cpo)

  use euITM_schemas
  implicit none

  integer:: nbtime,nrho,nspec,idxtime,npini,ipini,ibeamlet, itest
  integer:: idx,shot,run,refshot,refrun,i,nbeamlets,k,itokamak

  double precision:: source_width,source_height
  double precision:: y_beamlet_min,y_beamlet_max,z_beamlet_min,z_bea
  double precision, dimension(:), allocatable :: x_source,y_source,z
  double precision, dimension(:,:), allocatable :: x_beamlets,y_beam
  double precision, dimension(:,:), allocatable :: r_beamlets,phi_be

  type (type_coreprof),pointer   :: corep_input(:)
  type (type_nbi),pointer        :: nbi_cpo(:)
```

- requires ***coreprof*** CPO
- writes ***nbi*** CPO



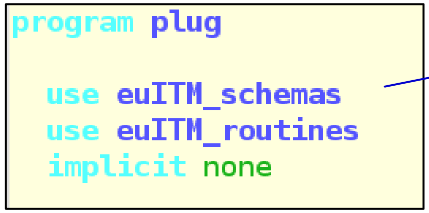
```
! FILLING INPUT NBI STRUCTURE

nbi_cpo(1)%setup_inject%beamlets%position%r   = r_beamlets
nbi_cpo(1)%setup_inject%beamlets%position%z   = z_beamlets
nbi_cpo(1)%setup_inject%beamlets%position%phi = phi_beamlets


do ipini=1,npini
   ! ITER
   if(itokamak.eq.1) then
      nbi_cpo(1)%halfe_cfr%value(ipini)          = 0.
      nbi_cpo(1)%thirde_cfr%value(ipini)         = 0.
      nbi_cpo(1)%pow_unit%value(ipini)           = 2.0625e6
      nbi_cpo(1)%inj_eng_unit%value(ipini)       = 1.e6
```

- Calculates width and height of beam source from beamlets' coordinates.

- Fill the ***nbi*** CPO structure

The NEMO standalone Test Bed

Purpose: to read required CPOs from database and to call NEMO in order to fill in the ***distsource*** NBI output CPO.



```
program plug

  use euITM_schemas
  use euITM_routines
  implicit none
```

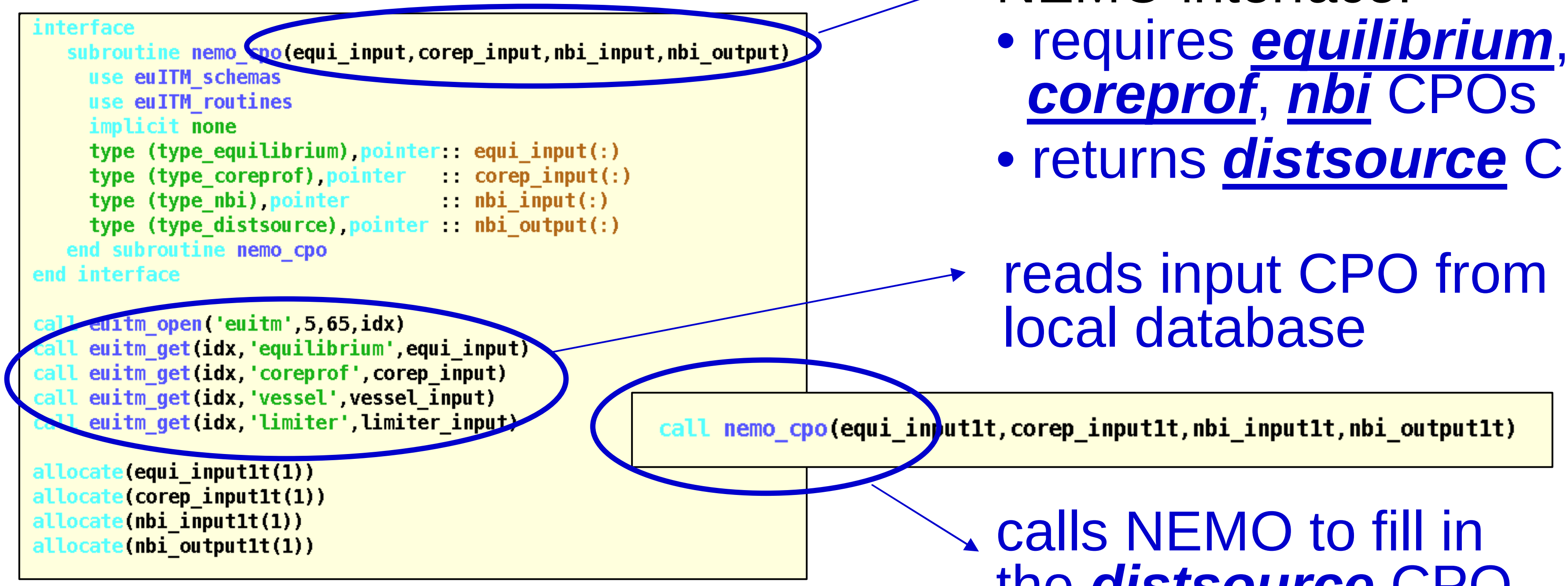
The Test Bed is a main program which calls NEMO as a subroutine.

NEMO interface:

- requires ***equilibrium***, ***coreprof***, ***nbi*** CPOs
- returns ***distsource*** CPO

reads input CPO from local database

calls NEMO to fill in the ***distsource*** CPO.

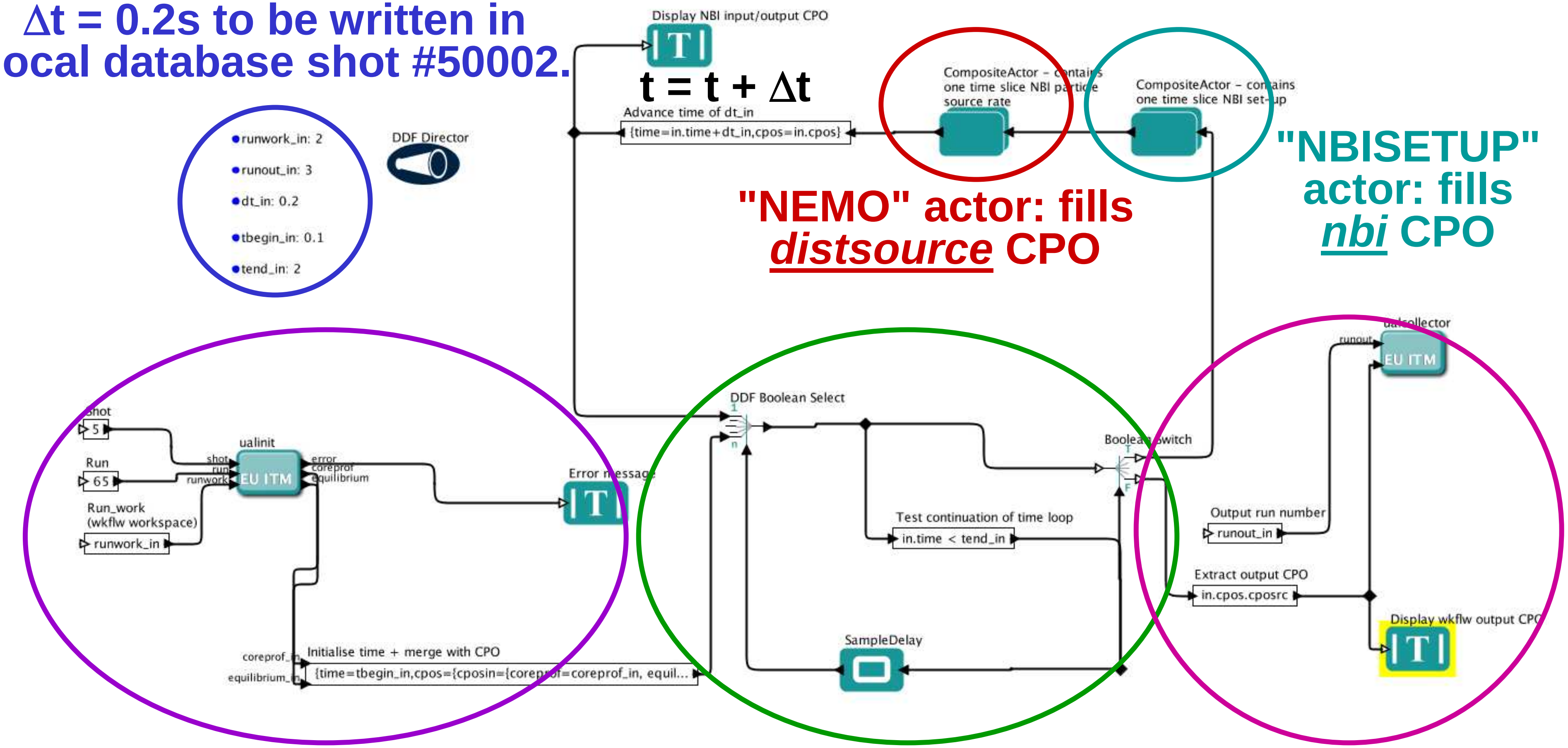


```
interface
   subroutine nemo_cpo(equi_input,corep_input,nbi_input,nbi_output)
     use euITM_schemas
     use euITM_routines
     implicit none
     type (type_equilibrium),pointer:: equi_input(:)
     type (type_coreprof),pointer   :: corep_input(:)
     type (type_nbi),pointer        :: nbi_input(:)
     type (type_distsource),pointer :: nbi_output(:)
   end subroutine nemo_cpo
end interface

call euitm_open('euitm',5,65,idx)
call euitm_get(idx,'equilibrium',equi_input)
call euitm_get(idx,'coreprof',corep_input)
call euitm_get(idx,'vessel',vessel_input)
call euitm_get(idx,'limiter',limiter_input)

allocate(equi_input1t(1))
allocate(corep_input1t(1))
allocate(nbi_input1t(1))
allocate(nbi_output1t(1))
```

```
call nemo_cpo(equi_input1t,corep_input1t,nbi_input1t,nbi_output1t)
```

The NBI test workflow in Kepler

Run from 0.1s to 2s with
Δt = 0.2s to be written in
local database shot #50002.



t = t + Δt

"NEMO" actor: fills *distsource* CPO

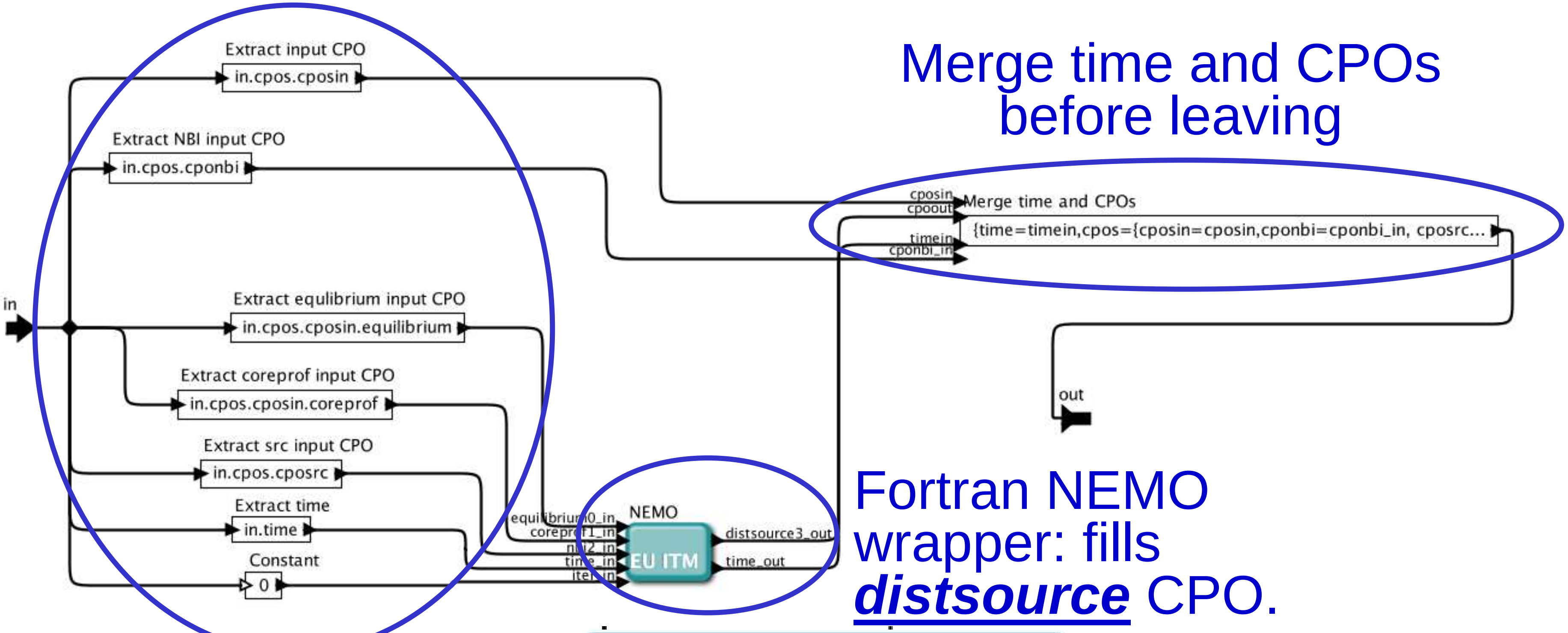
"NBISETUP" actor: fills *nbi* CPO

CPO readout from
local database shot #50065:
time, *coreprof*, *equilibrium*

Time loop
management

Output CPO *distsource*
written in local database
shot #50002

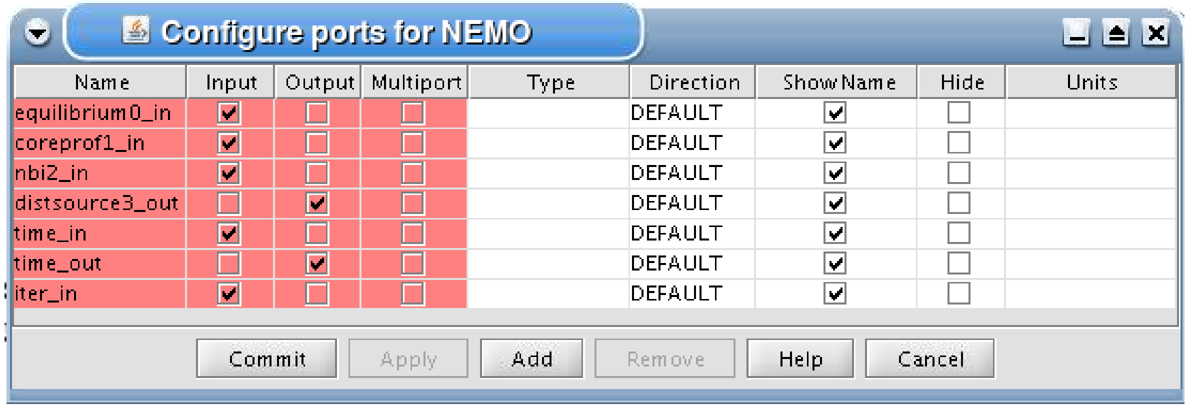
Composite actor containing NEMO



Merge time and CPOs before leaving

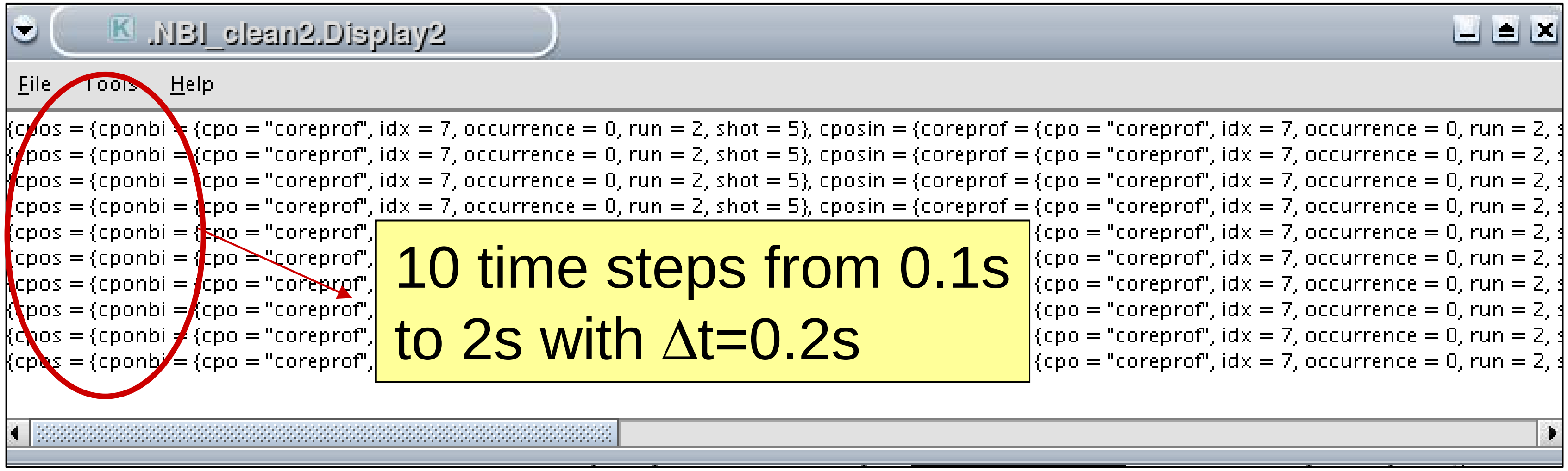
Fortran NEMO wrapper: fills ***distsource*** CPO.

Extract time and input/output CPOs: ***equilibrium, coreprof*, *nbi distsource***



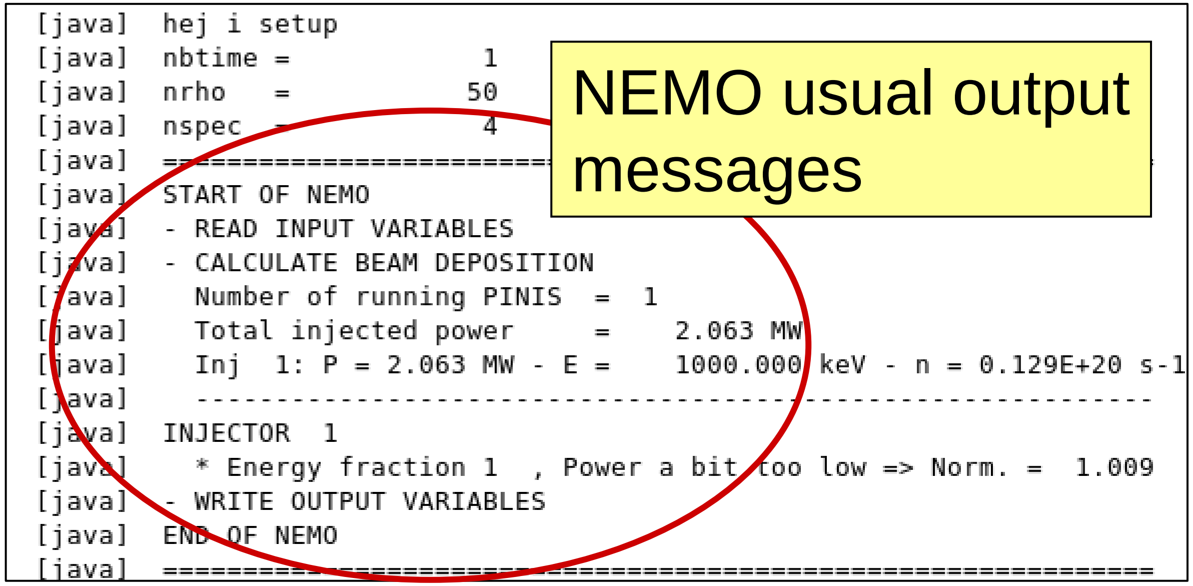
Name	Input	Output	Multiport	Type	Direction	Show Name	Hide	Units
equilibrium0_in	☑	□	□		DEFAULT	☑	□	
coreprof1_in	☑	□	□		DEFAULT	☑	□	
nbi2_in	☑	□	□		DEFAULT	☑	□	
distsource3_out	□	☑	□		DEFAULT	☑	□	
time_in	☑	□	□		DEFAULT	☑	□	
time_out	□	☑	□		DEFAULT	☑	□	
iter_in	☑	□	□		DEFAULT	☑	□	

Run of the NBI test workflow



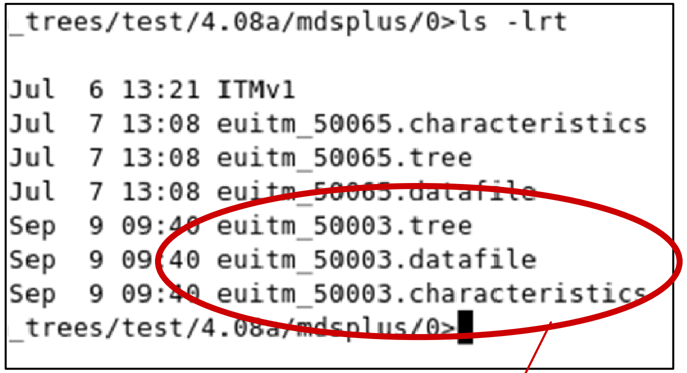
```
{cpos = {cponbi = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, shot = 5}, cposin = {coreprof = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, shot = 5}, cposin = {coreprof = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, shot = 5}, cposin = {coreprof = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, shot = 5}, cposin = {coreprof = {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof",                                                  {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof",                                                  {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof",                                                  {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof",                                                  {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof",                                                  {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
{cpos = {cponbi = {cpo = "coreprof",                                                  {cpo = "coreprof", idx = 7, occurrence = 0, run = 2, s
```

10 time steps from 0.1s to 2s with Δt=0.2s



```
[java]  hej i setup
[java]  nbtime =           1
[java]  nrho   =          50
[java]  nspec  =           4
[java]  ========================
[java]  START OF NEMO
[java]  - READ INPUT VARIABLES
[java]  - CALCULATE BEAM DEPOSITION
[java]    Number of running PINIS  =   1
[java]    Total injected power     =    2.063 MW
[java]    Inj  1: P = 2.063 MW - E =    1000.000 keV - n = 0.129E+20 s-1
[java]    ------------------------------------------------------------
[java]  INJECTOR  1
[java]    * Energy fraction 1  , Power a bit too low => Norm. =  1.009
[java]  - WRITE OUTPUT VARIABLES
[java]  END OF NEMO
[java]  ==============================================================
```

NEMO usual output messages



```
_trees/test/4.08a/mdsplus/0>ls -lrt

Jul  6 13:21 ITMv1
Jul  7 13:08 euitm_50065.characteristics
Jul  7 13:08 euitm_50065.tree
Jul  7 13:08 euitm_50065.datafile
Sep  9 09:40 euitm_50003.tree
Sep  9 09:40 euitm_50003.datafile
Sep  9 09:40 euitm_50003.characteristics
_trees/test/4.08a/mdsplus/0>
```

Write to local database

Summary and prospects

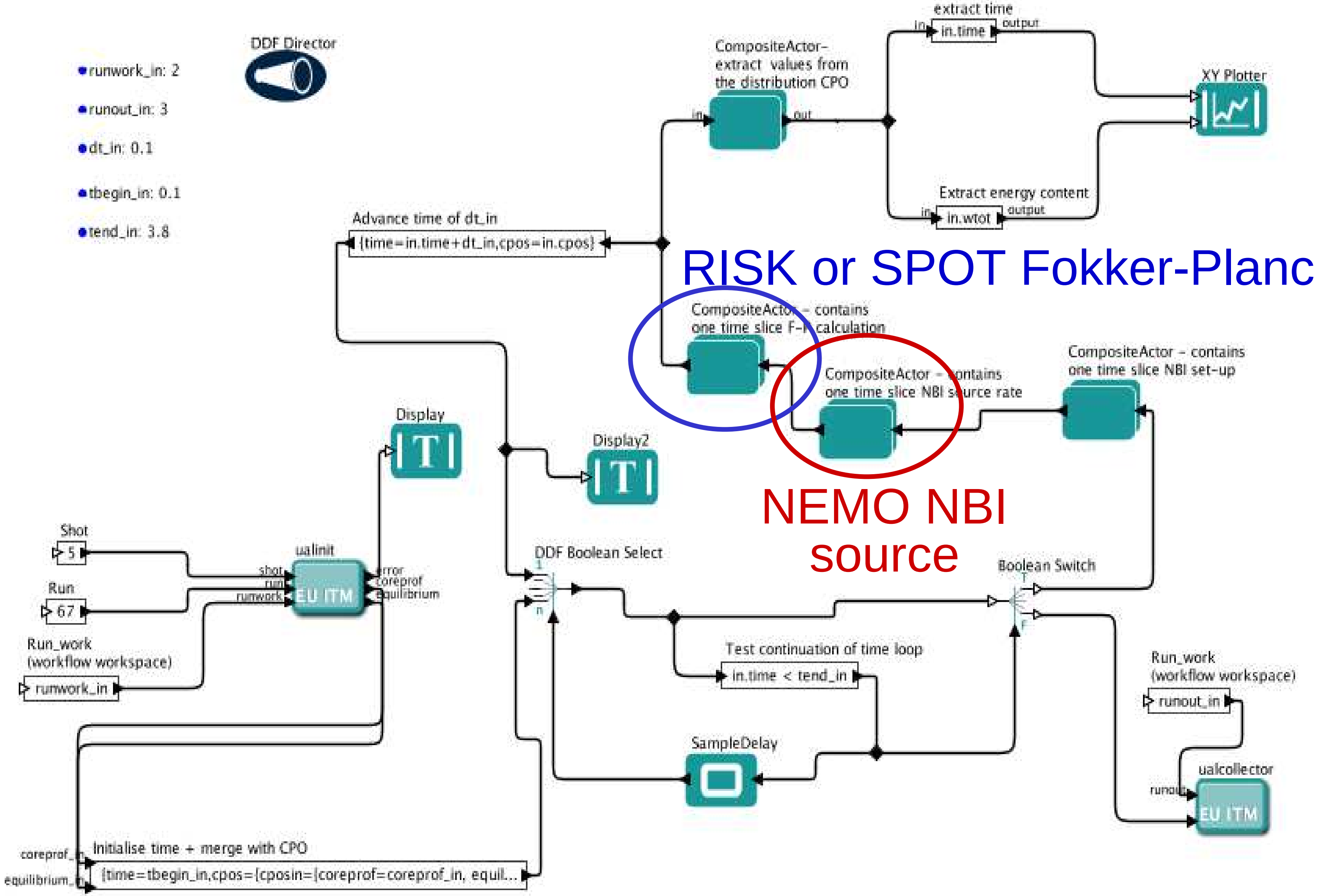
- NEMO (NEutral beam MOdelling) NBI source code ported to the ITM gateway (gforge project = nemo), CPOs implemented, standalone Test Bed created, kepler actor created, NBI test workflow created and running.

⇨The same will be done for following Fokker-Planck codes

- SPOT (Simulation of Particle Orbits in a Tokamak): orbit following Monte Carlo code for fast ion trajectory (gforge project = spot)
- RISK (Rapid Ion Solver for tokamaKs): bounce-averaged Fokker-Planck solver for fast ions (no gforge project yet)

⇨ This will create a complete NBI modelling capability for the ITM!

Workflow coming to a Kepler near you soon!



- runwork_in: 2
- runout_in: 3
- dt_in: 0.1
- tbegin_in: 0.1
- tend_in: 3.8

RISK or SPOT Fokker-Planck

NEMO NBI source