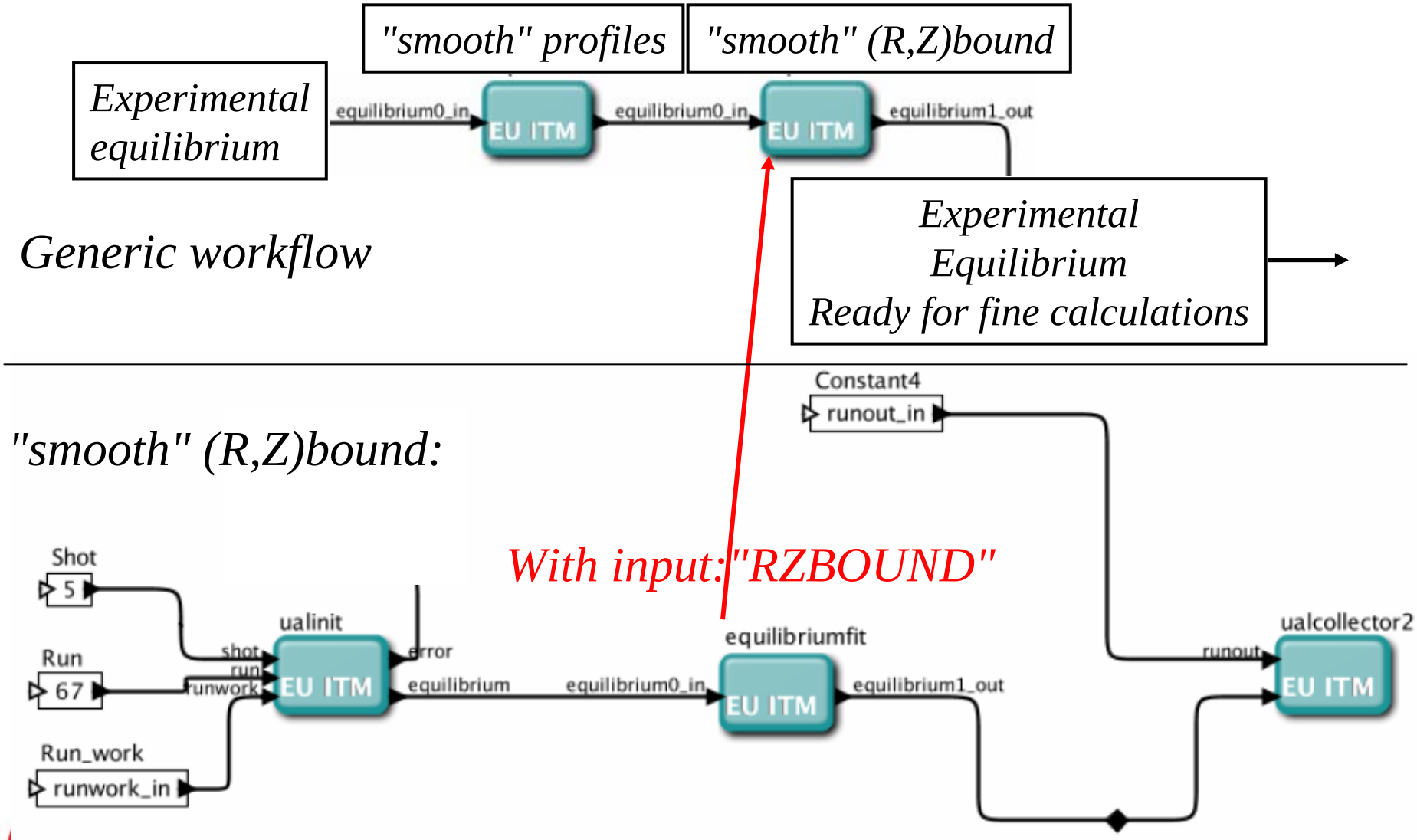
Interpos
generic code params
"numerical fit"

O. Sauter
CRPP-EPFL, Switzerland
ITM annual meeting, Lisbon, 2010

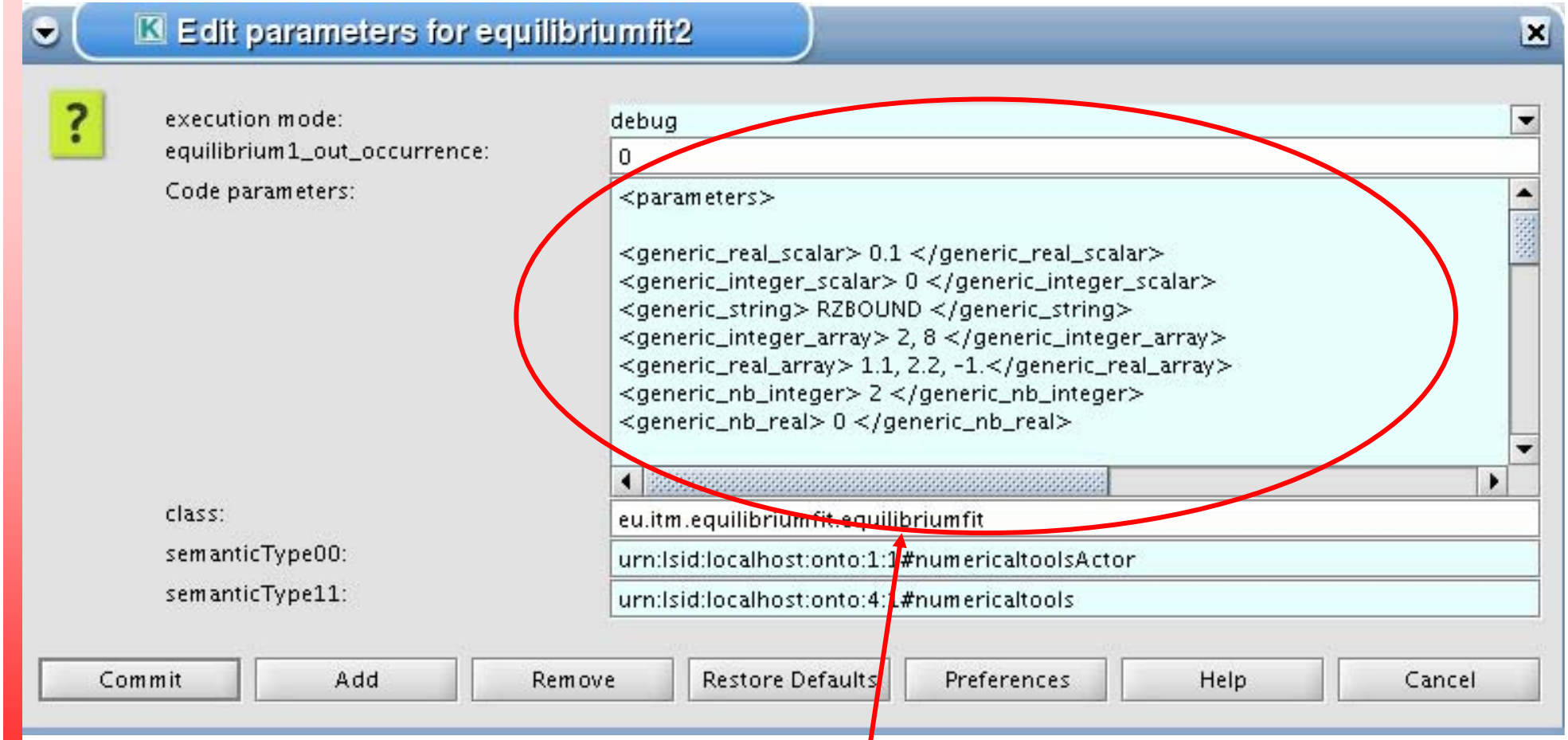
Goal of this technical presentation

- Many codes need "code_param" which is official ITM way to have inputs modifiable at run time
- Some equilibrium CPO do not have all the variables filled in or need some plasma boundary smoothing or need a different metric in ouput
- Interpos is a powerful cubic spline interpolation/extrapolation, mainly because it has versatile boundary conditions, it is fast and accurate
- We present here a simple program which uses these features to give an example of:
  - Generic library callable code_params
  - Interpos with periodic boundary conditions
- In addition it gives a tool to manipulate equilibria (will be essential in a workflow using experimental data)

A typical module: equilibriumfit



Configure Actor allows changing the codeparams values:



Equilibriumfit module uses the generic_codeparams which should be sufficient for many applications

Just the file params_generic_equilibriumfit.xml is needed (1/3)

```
<parameters>
<generic_real_scalar>      0.1          </generic_real_scalar>
<generic_integer_scalar>   0            </generic_integer_scalar>
<generic_string>           CHEASE       </generic_string>
<generic_integer_array>    2, 8         </generic_integer_array>
<generic_nb_integer>       2            </generic_nb_integer>
<generic_real_array>       1.1, 2.2, -1.</generic_real_array>
<generic_nb_real>          3            </generic_nb_real>
<generic_boolean>          .true.       </generic_boolean>
</parameters>
<! Note the space after commas in arrays>

<!-- DOCUMENTATION  -->
<!--  If generic_nb_real is non-zero then  -->
<!--     generic_nb_real > 0: number of time slices to compute at
time=generic_real_array(1:generic_nb_real) -->
<!--     generic_nb_real= -1: run on all time slices (same as if generic_nb_integer= -1)  --
>
<!--     generic_nb_real= -2: run on all time slices in time interval
                  [generic_real_array(1),generic_real_array((2)]  -->
<!--
```

file params_generic_equilibriumfit.xml (2/3)

```
<!--  If generic_nb_integer is non-zero then  -->
<!--     generic_nb_integer > 0: number of time slices to compute at time
indices=generic_integer_array(1:generic_nb_integer) -->
<!--     generic_nb_integer= -1: run on all time slices (same as if generic_nb_real= -1) -->
<!--     generic_nb_integer= -2: run for all time indices i=generic_integer_array(1) to
                                  generic_integer_array((2)  -->
<!-- It first checks if generic_nb_real is non-zero, so supersedes choices in
                                                          generic_integer_array -->
<!--    -->
<!-- If both generic_nb_real=0 and generic_nb_integer=0, use only last time index
<!--    -->
```

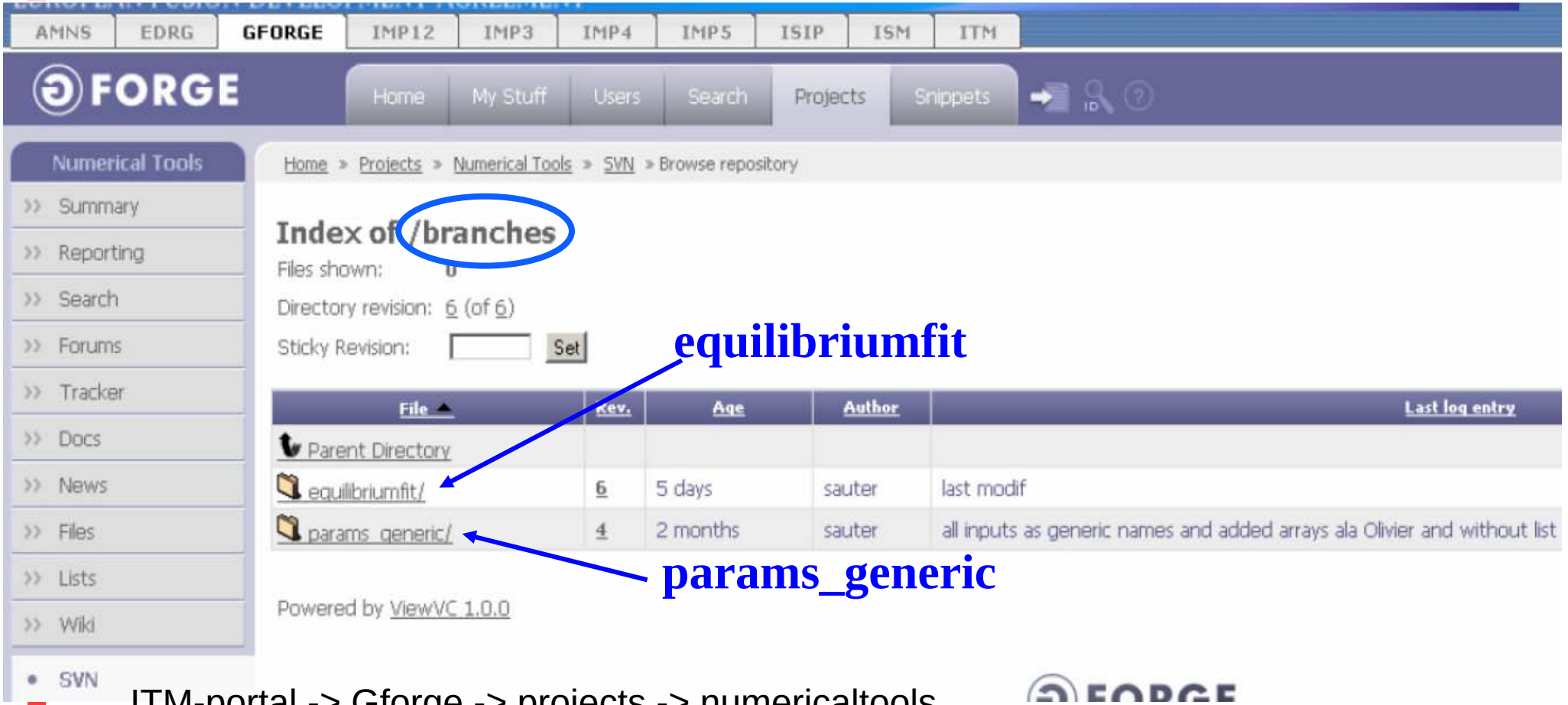
file params_generic_equilibriumfit.xml (3/3)

<!-- **generic_string** determines the action (select case) intended for the module with the following options: -->
<!-- generic_string = **RZBOUND** : smooth the (R,Z) plasma boundary and replace within the equilibrium_out CPO on specified times (see above) -->
<!-- generic_string = **CHEASE** : run CHEASE on specified times (see above) -->
<!-- generic_string = no other cases yet defined -->

<!-- HELP for specific cases: -->
<!-- for **RZBOUND** -->
<!-- **generic_real_scalar** determines the factor multiplying the default tension for the interpos spline smoothing (default=0.1) -->

<!-- END OF DOCUMENTATION -->

Gforge project: numerical_tools/branches



ITM-portal -> Gforge -> projects -> numericaltools

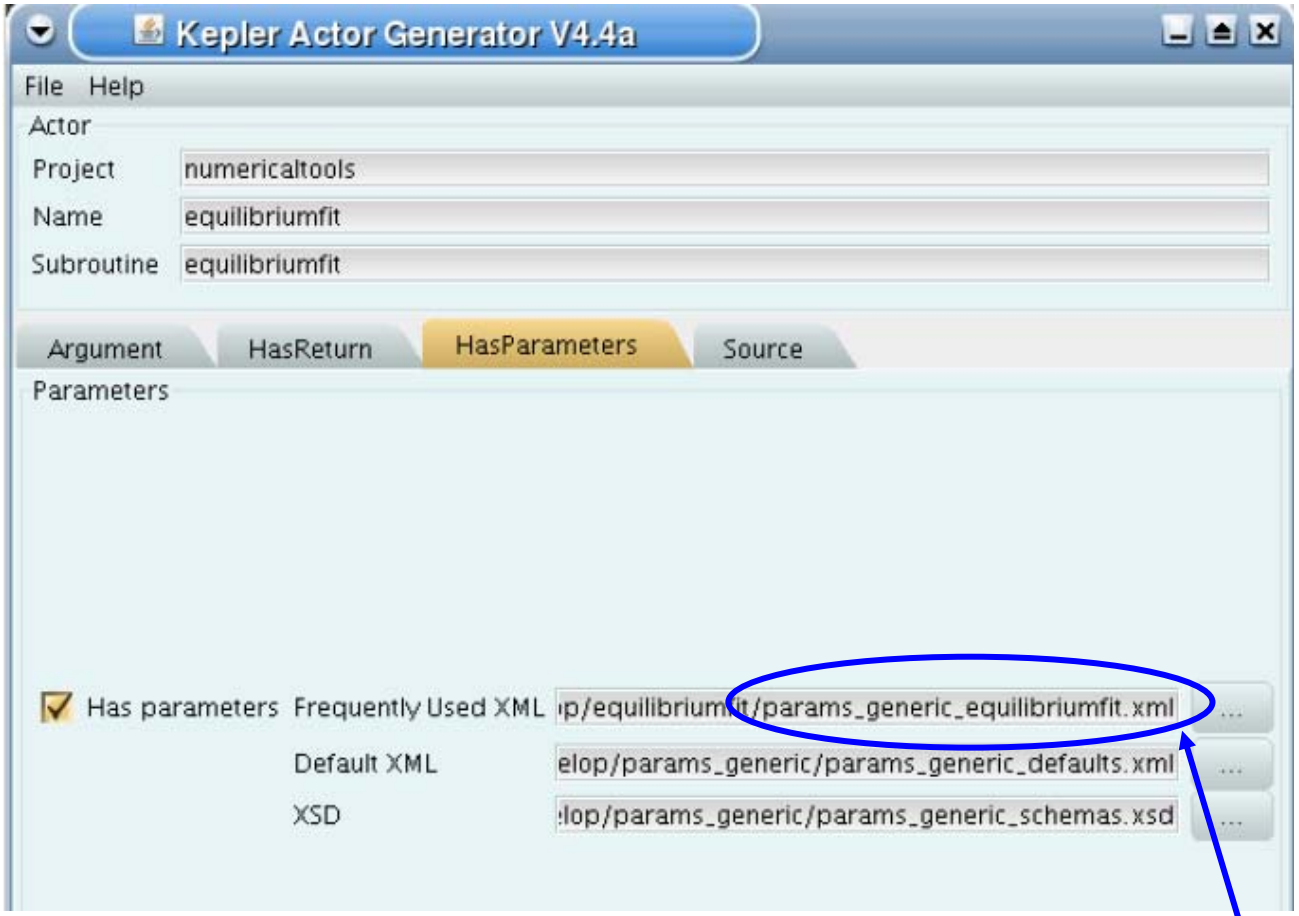
svn co http://gforge.efda-itm.eu/svn/numerical_tools/branches numerical_tools

Actually we have actor repository, where do we have library repository…?

The coding in the subroutine:

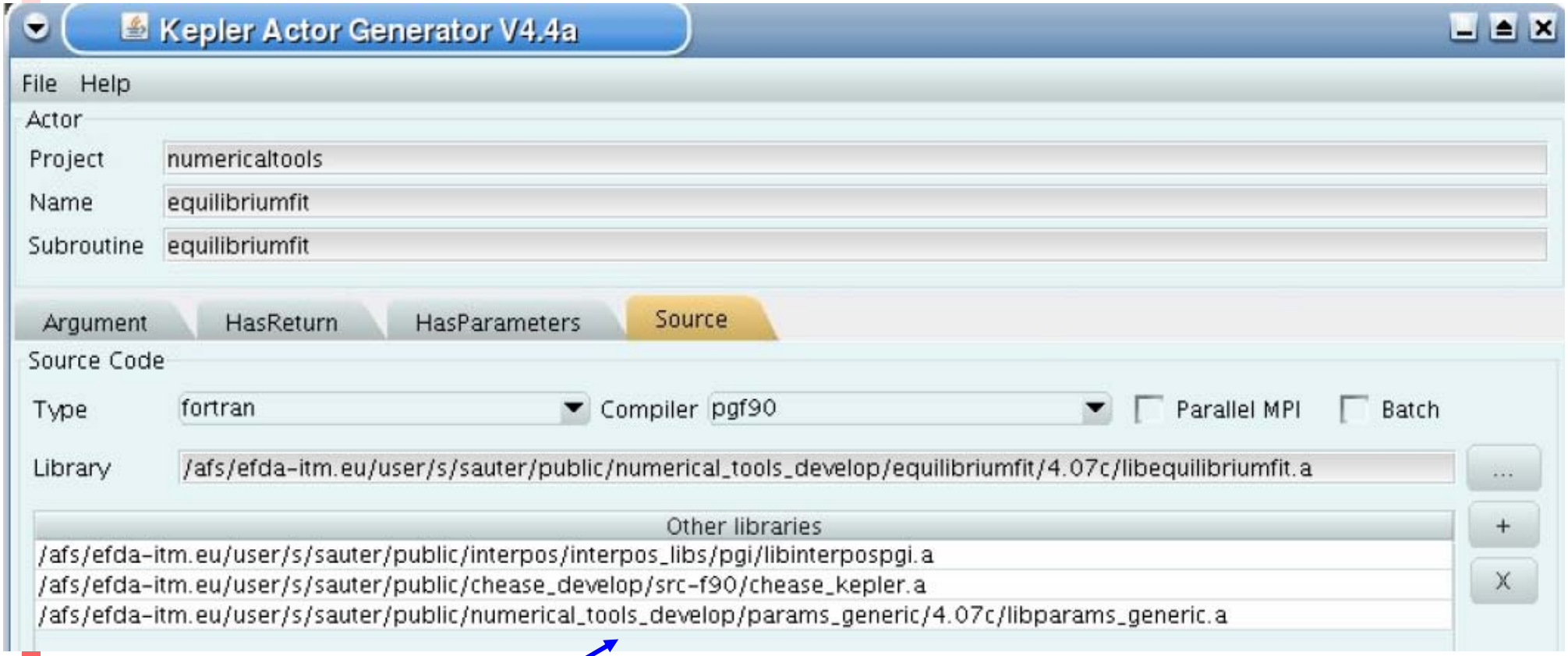
```
subroutine equilibriumfit(equilibrium_in,equilibrium_out,params_in)
  use itm_types
  use params_generic_module
  use euITM_schemas
  use euITM_routines
  use interpos_module

  type (type_param) :: params_in
  …
  call assign_params_generic(params_in,istatus) ! subroutine in library
  ! Now you can use all the variables set in "parameters" xml part:
  if (generic_nb_integer .eq. 2) then …
  if (generic_string .eq. 'CHEASE') then …
```

in fc2k (1/2):



- **Use the generic files for the parameters,**
- **Only the "frequently used xml" file needs to be copied locally and modified for the module purposes**

in fc2k (2/2):



You add the libraries, here:

libinterpos chease and libparams_generic

Makefile and libraries

In Makefile:

```
DIRparams_generic=…numerical_tools_develop/params_generic/$(UAL_version)
NAMEparams_generic=params_generic

F90=pgf90
F90FLAGS = -Mbounds -Mchkfpstk -Mchkptr -g -r8 -fPIC
           -Mnosecond_underscore
           -I/afs/efda-itm.eu/project/switm/ual/$(UAL_version)/include/amd64_pgi
           -I$(DIRinterpos) -I$(DIRparams_generic) -I$(DIRchease)
```

In FC2K: add libraries:

/afs/efda-itm.eu/user/s/sauter/public/params_generic/4.07c/libparams_generic.a

/afs/efda-itm.eu/user/s/sauter/public/interpos/interpos_libs/pgi/libinterpospgi.a

A typical module: equilibriumfit

Fitting the plasma boundary to avoid wiggles in the input data:

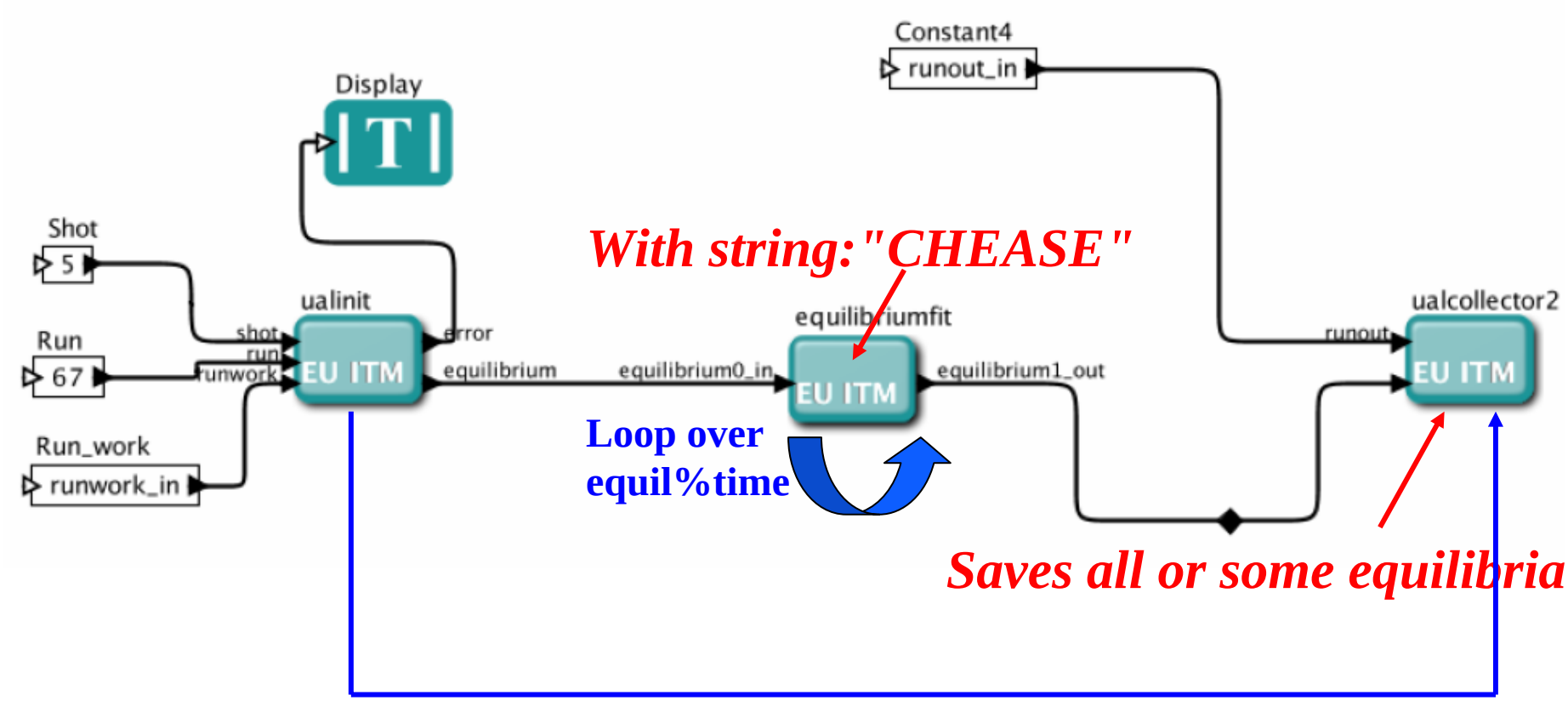
call bndfit(RIN,ZIN,NIN,ROUT,ZOUT,NOUT,tension):

```
call bndfit(equilibrium_in(itime_eff)%eqgeometry%boundary%r, &
        & equilibrium_in(itime_eff)%eqgeometry%boundary%z, &
        & size(equilibrium_in(itime_eff)%eqgeometry%boundary%r), &
        & equilibrium_out(itime)%eqgeometry%boundary%r, &
        & equilibrium_out(itime)%eqgeometry%boundary%z, &
        & size(equilibrium_out(itime)%eqgeometry%boundary%r), &
        & generic_real_scalar)
```

Within bndfit, just call interpos after sorting rho(theta) points:

```
call interpos(theta_sorted,rho_sorted,NIN_eff,nout=NBFIT,tension=TENSION,&
    & xout=thetaout,yout=rhoout,nbc=-1,ybc=twopi)
```

Equilibriumfit:
re-ran with CHEASE all equilibria from given



Would like to copy "all other cpos" as well? (ISIP question)
(not yet implemented)

Interpos is robust

- It is used in many codes
- It provides fitted function, 1st, 2nd derivatives and integral
- It is used for automatic profile fitting on TCV for many years
- On TCV we have:
  - Automatically ne and Te fitted profiles
  - Use them for neoclassical resistivity and bootstrap current
  - Get bootstrap fraction
  - Get Zeff from Ip – Iohm – IBS =0 (when there is no ECCD)
  - Run automatically TORAY-GA on all time-steps
  - Get chie (rho,t) on all time steps (from power balance)
  - ….
- **http://crpp.epfl.ch/~sauter/interpos**
- **http://crpp.epfl.ch/~sauter/neoclassical**
- **http://crpp.epfl.ch/~sauter/chease**

…

Generic codeparams

- It works within f90 alone as well of course. It can replace a "generic namelist"
- Could be a standard simple codeparams since with a scalar real, integer, string and array of real, integers and strings, you cover many options…
- You just need to link with the library and copy/paste the file ….xml, add the comments/documentation in the file and relate to it within fc2k
- Note: the wrapper created by fc2k can be used as the "program" part to call the module in a f90 program outside kepler. It creates the file (good to look at):

  …kepler/src/cpp/itm/equilibriumfit/standalone.f90

equilibriumfit

- Is an easy to follow example using the generic code_params
- Is an easy example using interpos and chease libraries
- Is a useful tool which will be required when using experimental data. In TCV there is an automatic smoothing of plasma boundary of the experimental data before running CHEASE and then other codes like KINX, TORAY-GA, etc
- Is a useful tool to rerun an existing equilibrium CPO and get the metric you need for your code (will need some work for standard choices of inputs for CHEASE for example)
- Is extendable with: string='other_case'