Karlsruhe Institute of Technology

Coupling of PARCS with the porus-media two-phase flow code TwoPorFlow for the improved analysis of SMR-cores using the ICoCo approach

A. Campos-Muñoz, V. Sanchez-Espinoza, U. Imke

Institute for Neutron Physics and Reactor Technology

Content

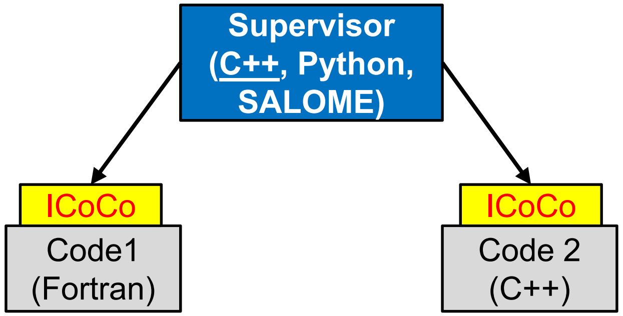
- Introduction to ICoCo
- PARCS ICoCo implementation
- TwoPorFlow code
- TwoPorFlow ICoCo implementation
- PARCS/TPF coupling
- SMART SMR core description
- Transient definition
- Results
- Conclusions and Outlook

Introduction to ICoCo

- Interface for Code Coupling (ICoCo): provides a standard **frame** for code coupling.

Define methods for:

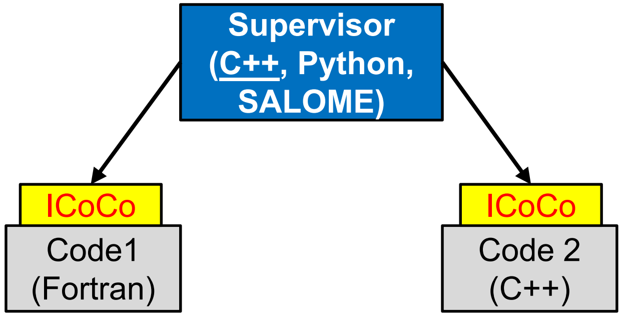
- Initialization and termination
- Time advance
- Save and restore state
- Getting and setting fields



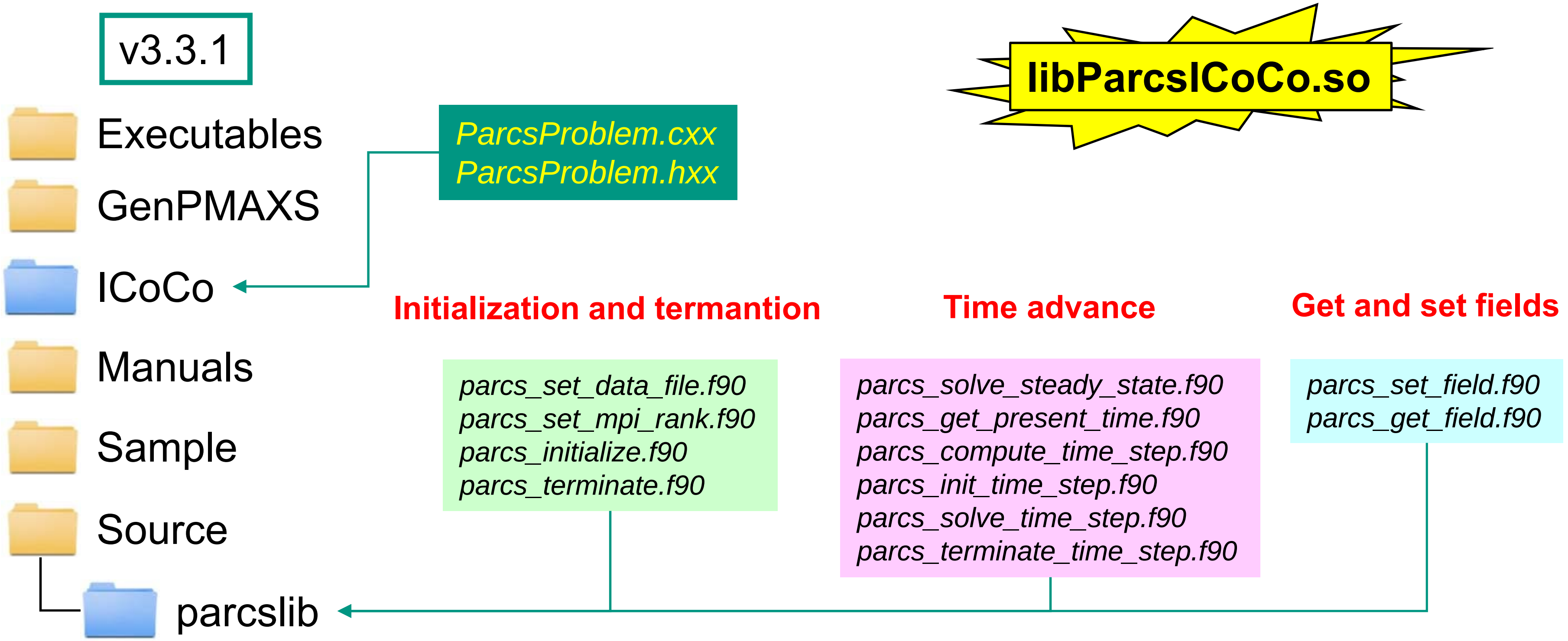
- **Code split** in funtional pieces.
- ICoCo framework MED format **mesh** is compulsory.
- Inherently bound with **MEDCoupling** library.

Introduction to ICoCo

- Methods for initialization and termination:
  - *setDataFile( )*
  - *initialize( )*
  - *presentTime( )*
  - *terminate ( )*
- Methods for time advance:
  - *solveSteadyState( )*
  - *computeTimeStep( )*
  - *initTimeStep( )*
  - *solveTimeStep( )*
  - *validateTimeStep( )*
- Methods for getting and setting fields to the code:
  - *getInputFieldNames( )*
  - *getOutputFieldNames( )*
  - *setInputMEDField( )*
  - *getInputMEDFieldTemplate( )*
  - *getOutputMEDField( )*



PARCS ICoCo implementation



v3.3.1

Executables

GenPMAXS

ICoCo

Manuals

Sample

Source

parcslib

ParcsProblem.cxx
ParcsProblem.hxx

libParcsICoCo.so

Initialization and termantion

parcs_set_data_file.f90
parcs_set_mpi_rank.f90
parcs_initialize.f90
parcs_terminate.f90

Time advance

parcs_solve_steady_state.f90
parcs_get_present_time.f90
parcs_compute_time_step.f90
parcs_init_time_step.f90
parcs_solve_time_step.f90
parcs_terminate_time_step.f90

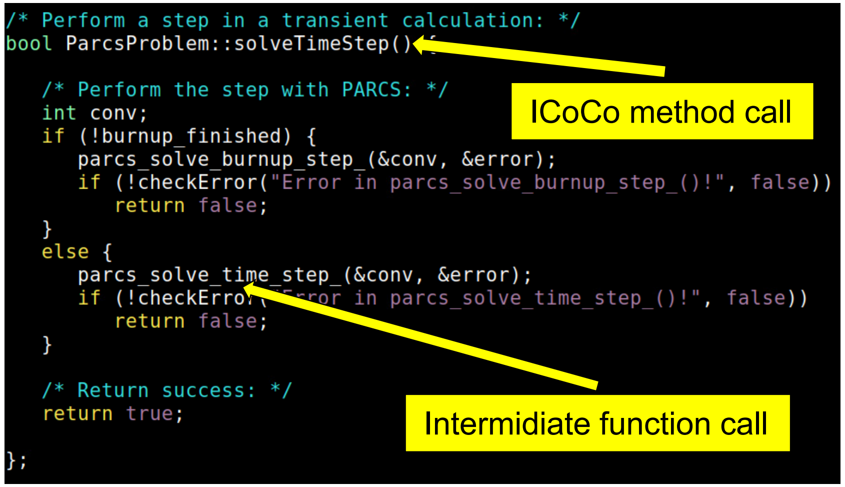
Get and set fields

parcs_set_field.f90
parcs_get_field.f90

PARCS ICoCo implementation

ParcsProblem.cxx

ICoCo method example



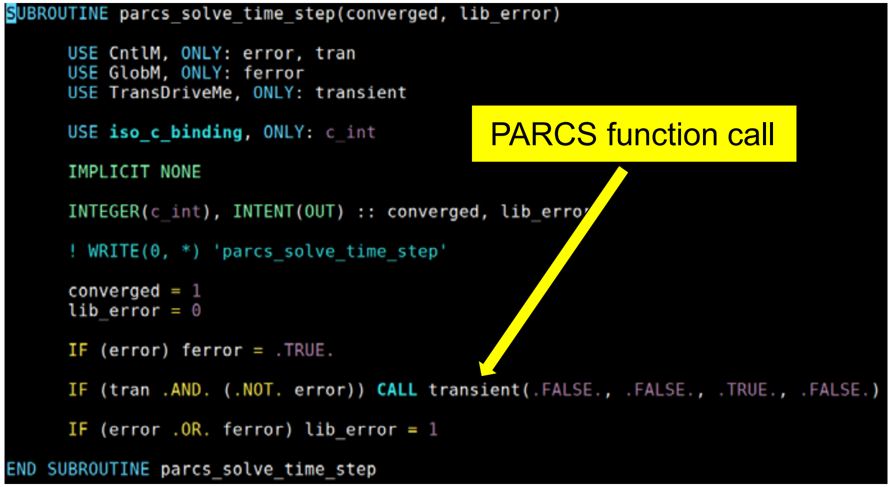
```
/* Perform a step in a transient calculation: */
bool ParcsProblem::solveTimeStep() {

   /* Perform the step with PARCS: */
   int conv;
   if (!burnup_finished) {
      parcs_solve_burnup_step_(&conv, &error);
      if (!checkError("Error in parcs_solve_burnup_step_()!", false))
         return false;
   }
   else {
      parcs_solve_time_step_(&conv, &error);
      if (!checkError("Error in parcs_solve_time_step_()!", false))
         return false;
   }

   /* Return success: */
   return true;

};
```

ICoCo method call

Intermidiate function call

parcs_solve_time_step.f90

Fortran function example

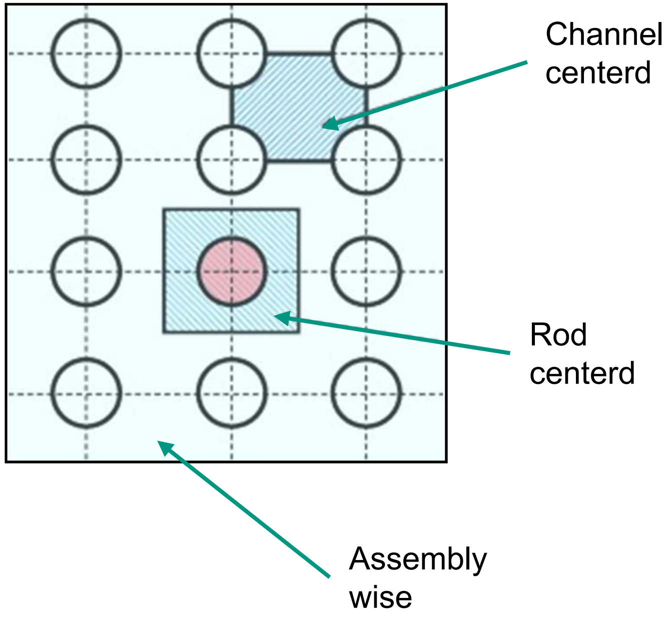


```
SUBROUTINE parcs_solve_time_step(converged, lib_error)

     USE CntlM, ONLY: error, tran
     USE GlobM, ONLY: ferror
     USE TransDriveMe, ONLY: transient

     USE iso_c_binding, ONLY: c_int

     IMPLICIT NONE

     INTEGER(c_int), INTENT(OUT) :: converged, lib_error

     ! WRITE(0, *) 'parcs_solve_time_step'

     converged = 1
     lib_error = 0

     IF (error) ferror = .TRUE.

     IF (tran .AND. (.NOT. error)) CALL transient(.FALSE., .FALSE., .TRUE., .FALSE.)

     IF (error .OR. ferror) lib_error = 1

END SUBROUTINE parcs_solve_time_step
```

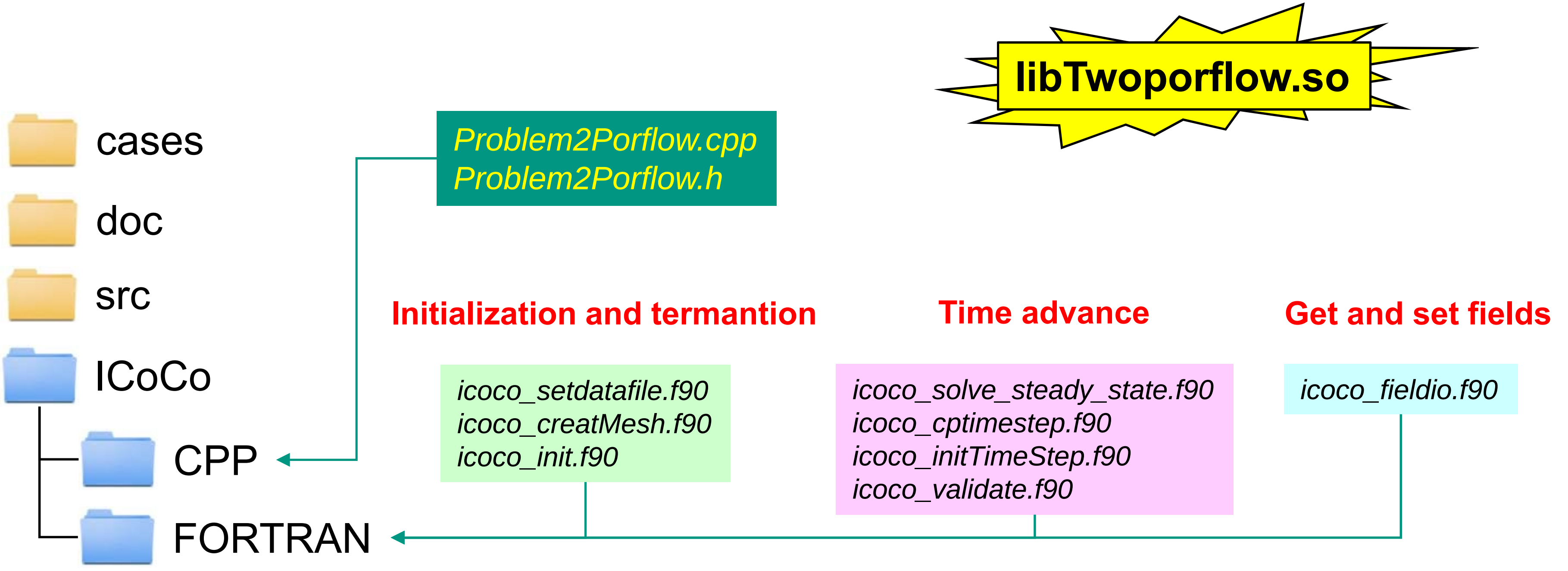
PARCS function call

TwoPorFlow code

- Porus-media forced convection
- Sub-channel fuel assembly simulations
- Steady-state and transient solution
- Two-phase flow (6 equations)
- 3D conservation equations
- 2D heat conduction model for fuel rods
- Coarse Cartesian grids
- Programmed in Fortran95



TWOPORFLOW ICoCo implementation



libTwoporflow.so

cases

doc

src

ICoCo

CPP

FORTRAN

Problem2Porflow.cpp
Problem2Porflow.h

Initialization and termantion

icoco_setdatafile.f90
icoco_creatMesh.f90
icoco_init.f90

Time advance

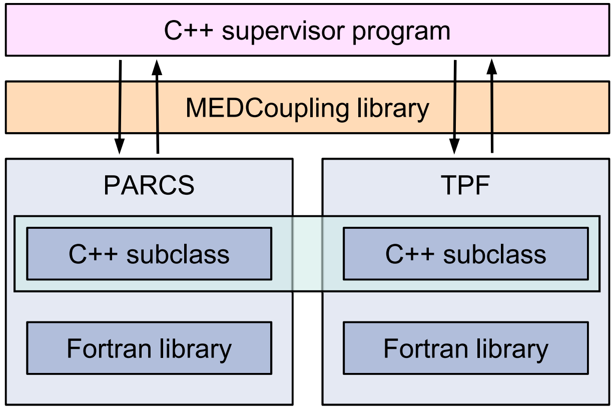
icoco_solve_steady_state.f90
icoco_cptimestep.f90
icoco_initTimeStep.f90
icoco_validate.f90

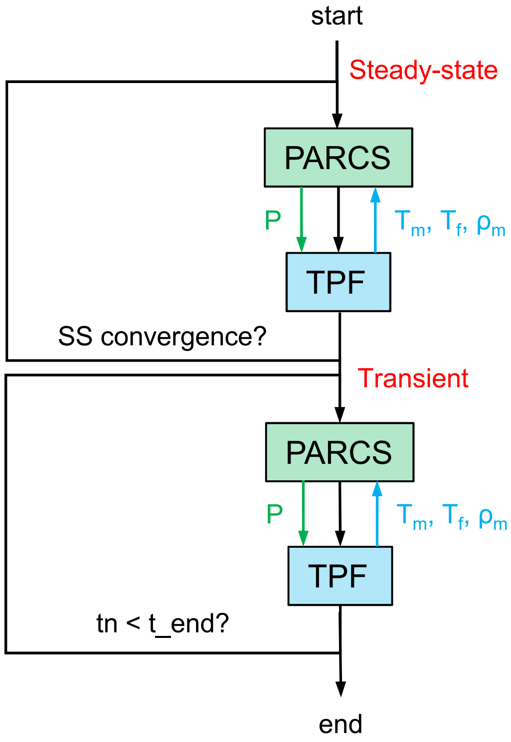
Get and set fields

icoco_fieldio.f90

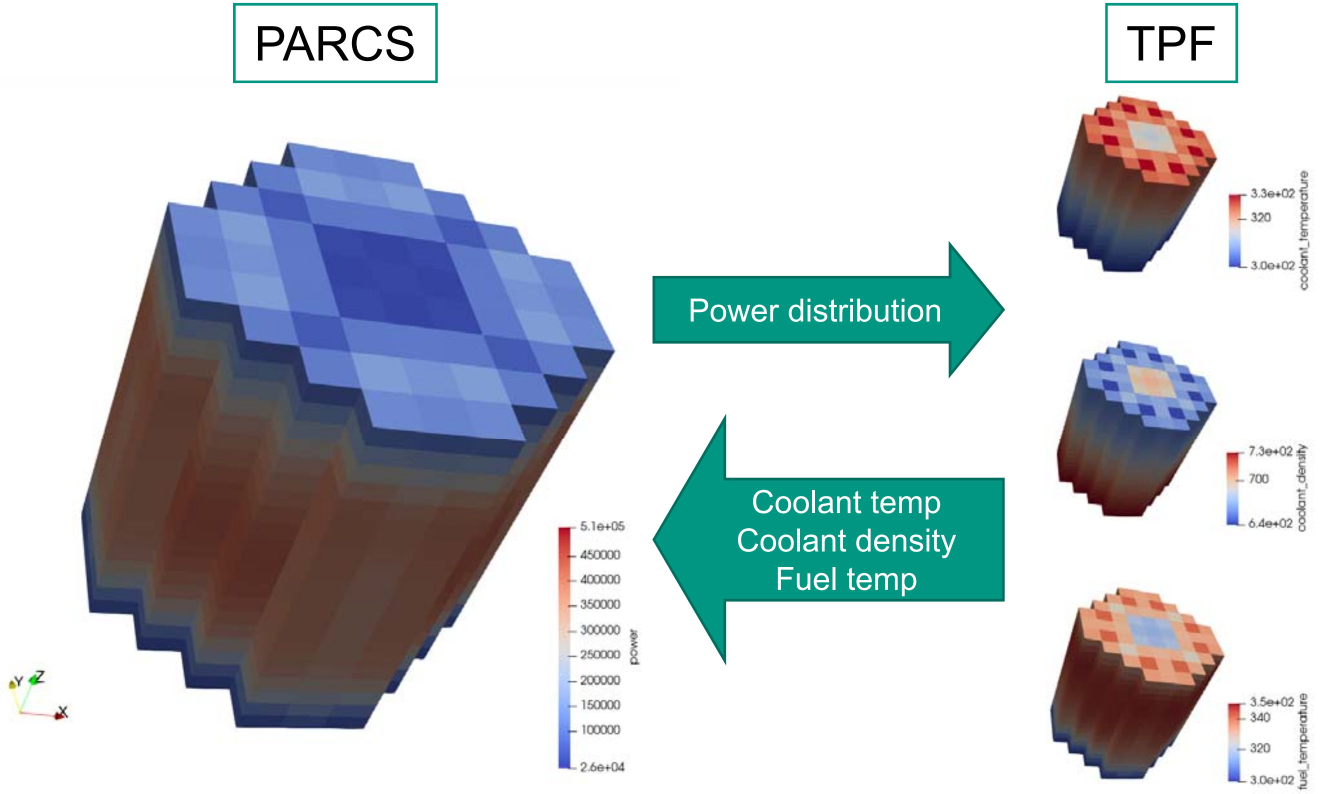
PARCS/TPF coupling

- External coupling.
- Serial execution.
- Domain overlapping.
- Fields mapping via MEDCoupling library.
- Explicit iterative scheme.
- Node-wise feedback.





PARCS/TPF coupling



PARCS

TPF

Power distribution

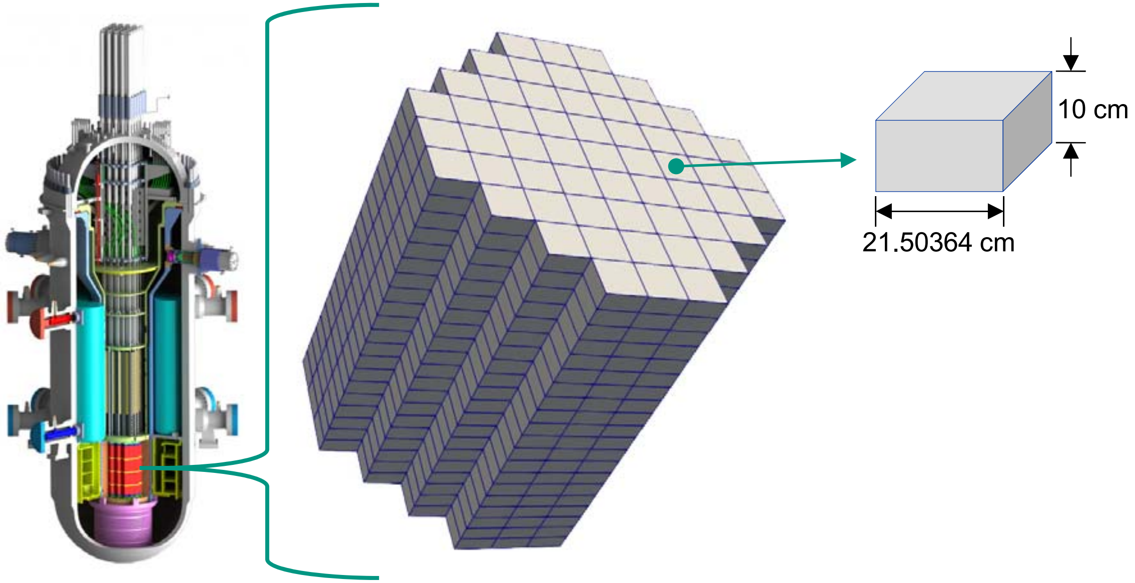
Coolant temp
Coolant density
Fuel temp

SMART SMR core description

- Integrated PWR (iPWR)

Parameter	Value
Total power	330 MW_{th}
System pressure	15 MPa
Inlet temperature	296 C
Core flow	2006.4 kg/s

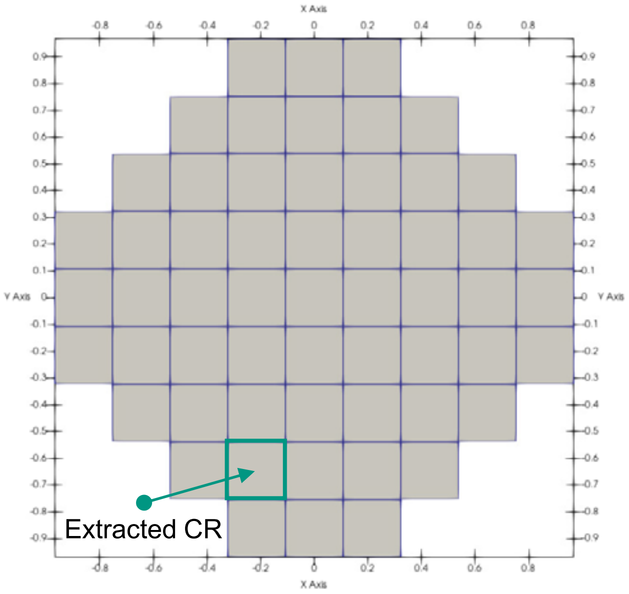
- 9x9 radial nodalization
- 20 axial levels



Transient definition

- Half highest CR worth extraction (0.725 $) at the hot zero power (HZP) condition.

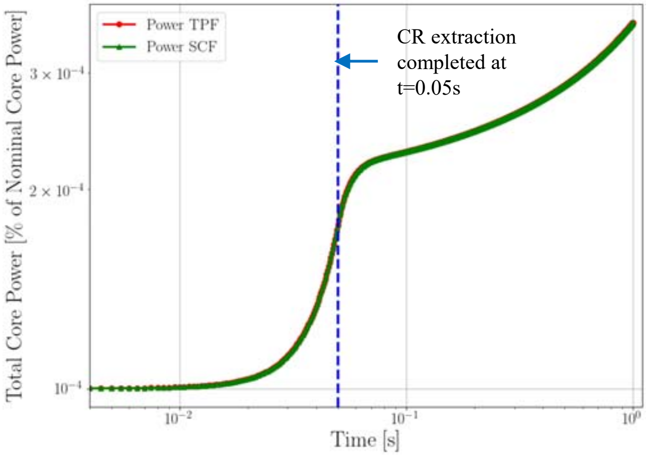
Parameter	Value
Initial core power	1.0E-4 %
Highest CR worth	1.45 $
Ejection duration	0.05 s
End of transient simulation	1.0 s
Time step	0.0005 s



Extracted CR

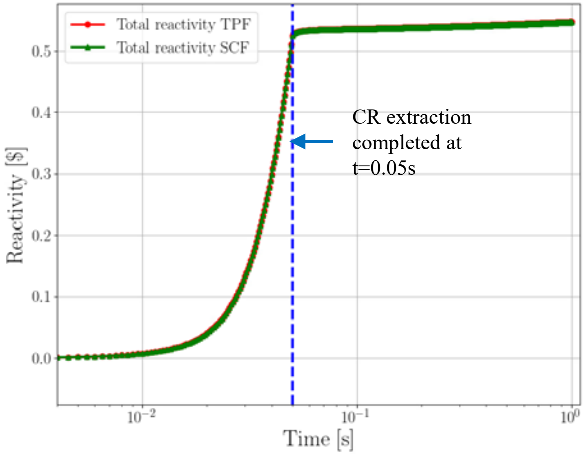
Results

PARCS Total Core Power



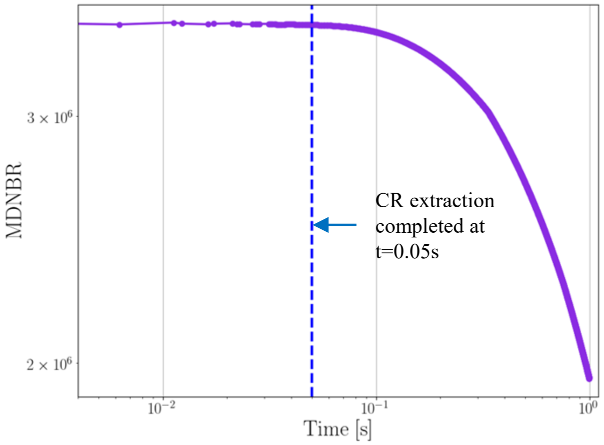
CR extraction completed at t=0.05s

PARCS Total Reactivity



CR extraction completed at t=0.05s

TPF MDNBR



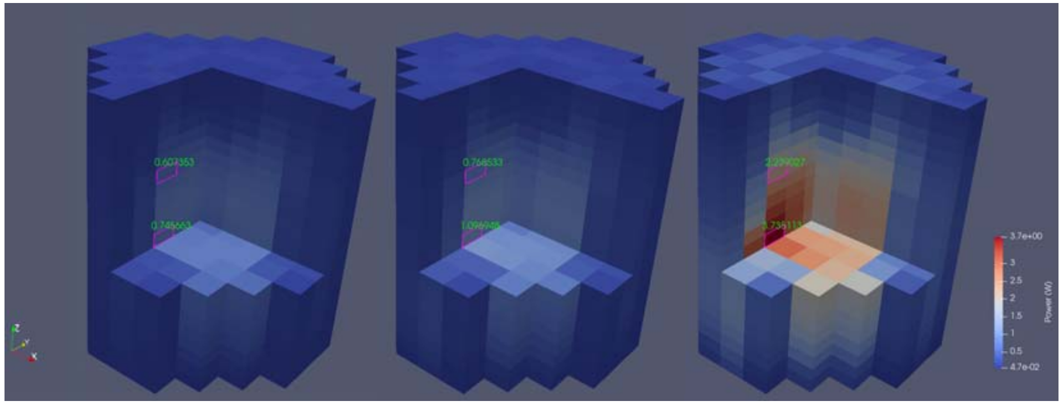
CR extraction completed at t=0.05s

Run time = 18 min

Results

Karlsruhe Institute of Technology

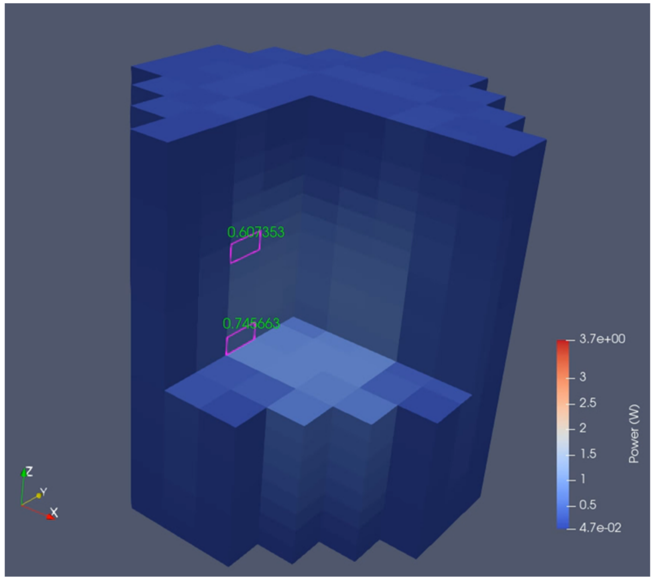
Cell power evolution



0.0 s

0.05 s

1.0 s



Conclusions and Outlook

- Successful coupling of PARCS and TwoPorFlow based on ICoCo.
- PARCS/TPF results were verified with PARCS/SCF.
- Since PARCS post-processing capabilities are not user-friendly:
  - Thanks to the inherent MED mesh format where all the variables are stored, post-processing is easier to do.

- Next steps:
  - Perform different transients where strong feedback and/or non-symetrical flow appears.
  - Analyse a Rod Ejection Accident of a SMR.

Karlsruhe Institute of Technology

Thank you for your attention