CORTEX
Core monitoring techniques and
experimental validation and demonstration

Simulating neutron noise with the CORE SIMplus diffusion solver

CORTEX validation workshop, 12-13 March 2020, GRS, Munich

Antonios Mylonakis (Chalmers)

antmyl@chalmers.se

This project has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 754316.

Outline

- Motivation for developing CORE SIMplus
- Code overview
- Noise source modelling
- Green's function method
- Summary
- Practical session

Motivation for developing CORE SIMplus

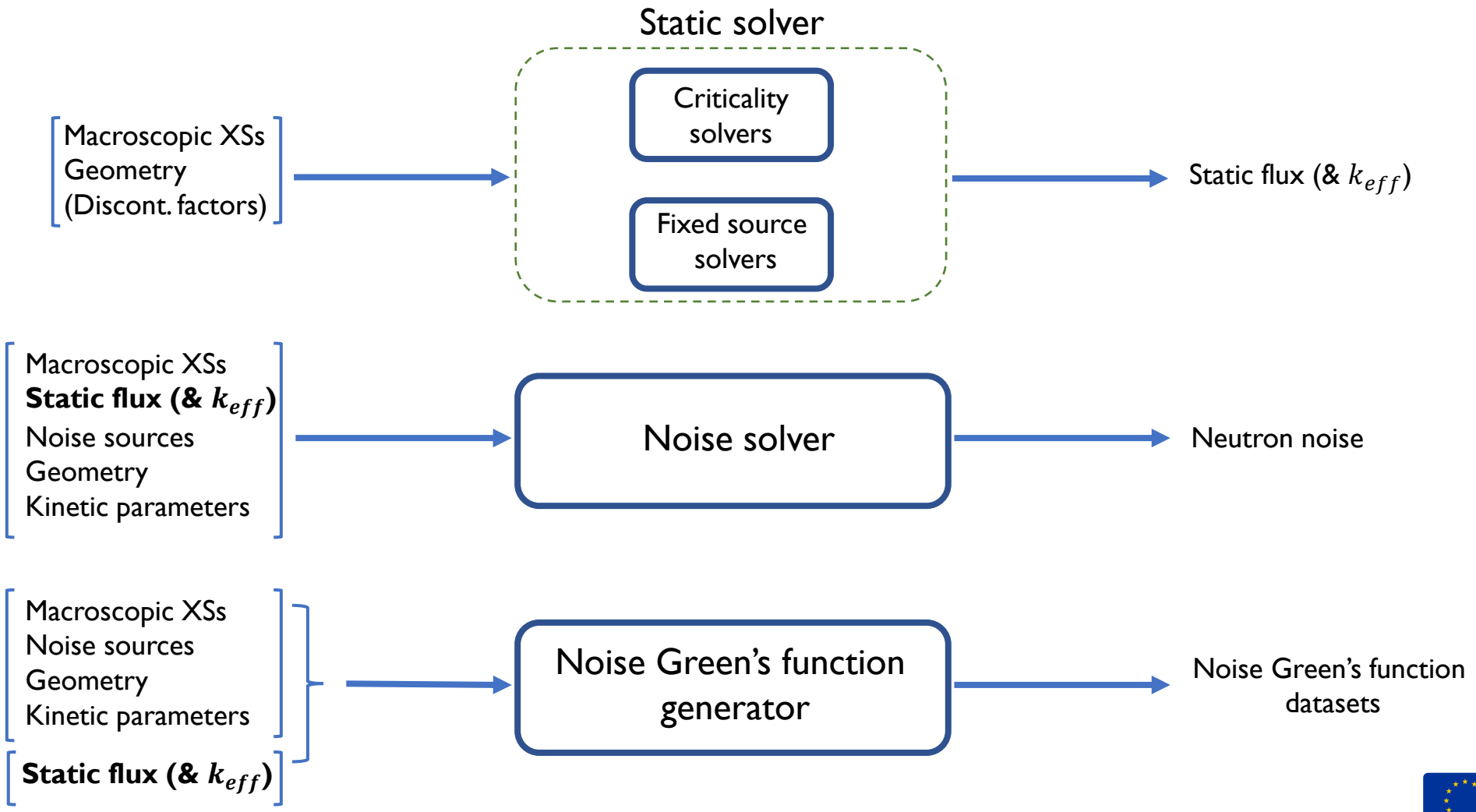
- Need for a solver for neutron noise applications
  - Core monitoring
  - Reactor diagnostics
  - Experiment design

- The solver needs to be fast and flexible
  - Modelling of all relevant noise sources
  - Large datasets generation for Machine Learning training

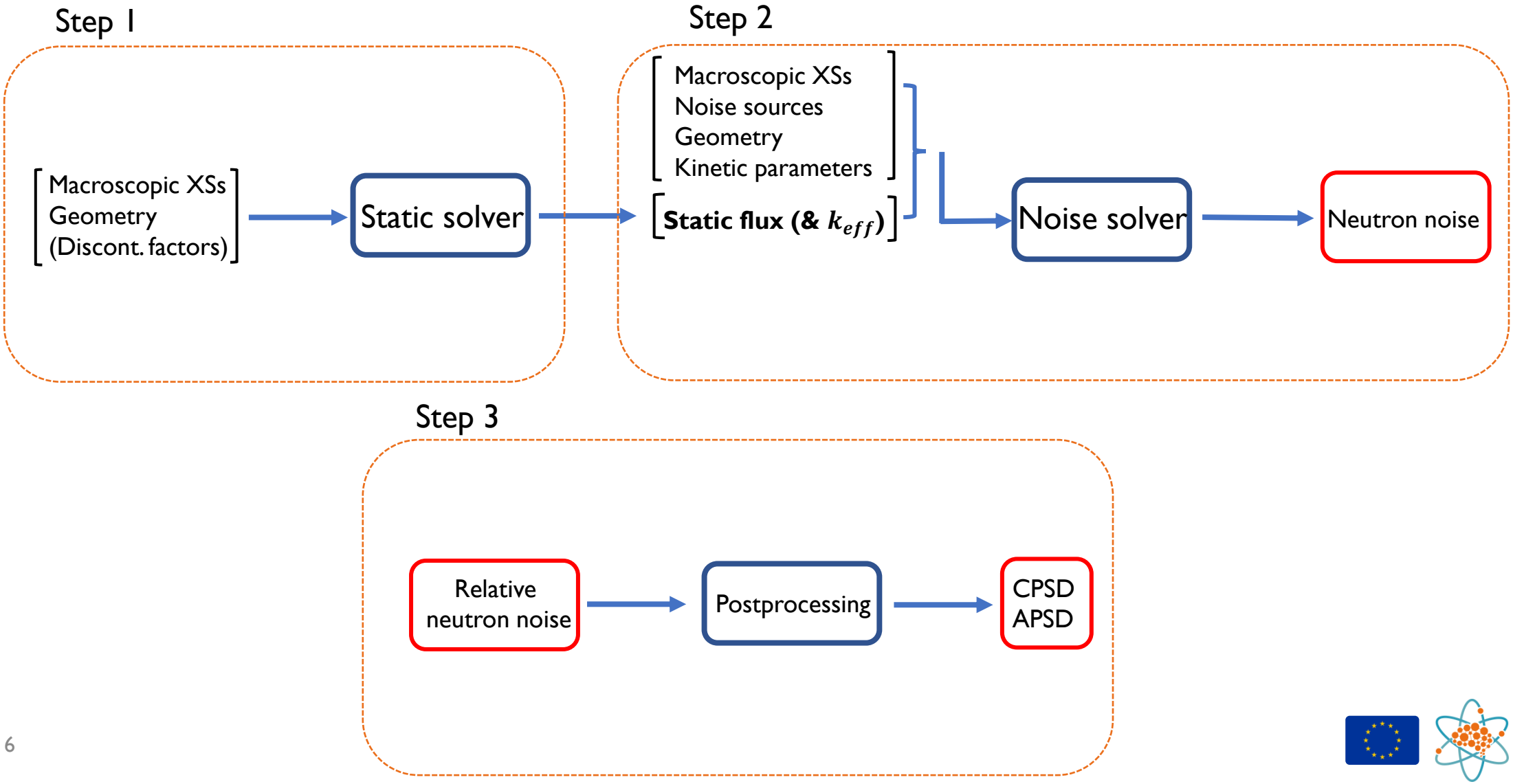
Useful features of CORE SIMplus

- The code capitalizes on CORE SIM
- Frequency domain solver
- Non uniform mesh
- Various numerical methods
- Noise Green's function generator
- Matlab-based code

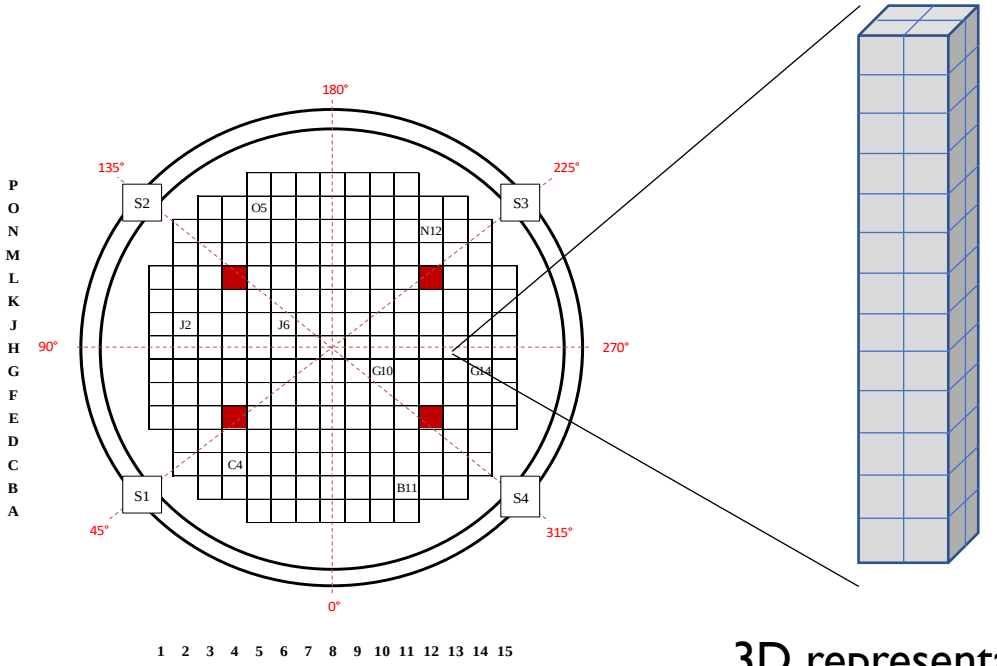
Overview of the code



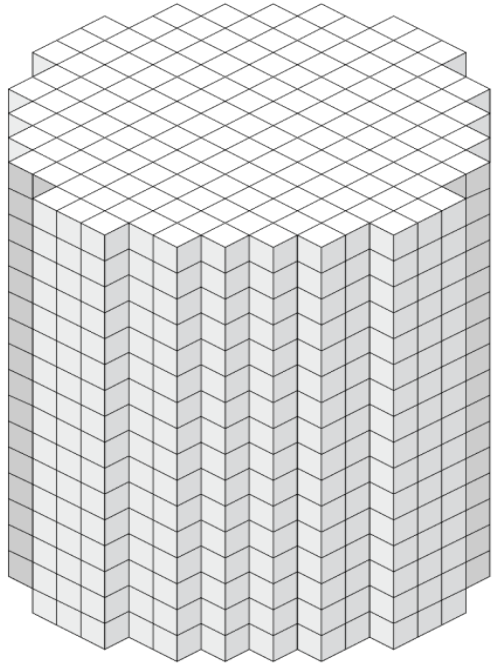
Solving a noise problem



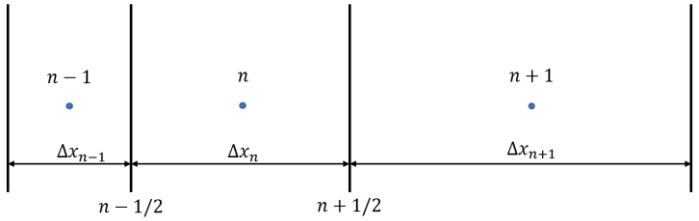
Solving a noise problem: spatial discretization



3D representation of each fuel assembly



3D representation of the core



Non uniform mesh

Solving a noise problem: noise source models

$$\boldsymbol{A}_{noise}\boldsymbol{\Phi}_{noise} = \boldsymbol{S}_{noise}$$

- All the models are built based on the XS fluctuations or arbitrary source terms

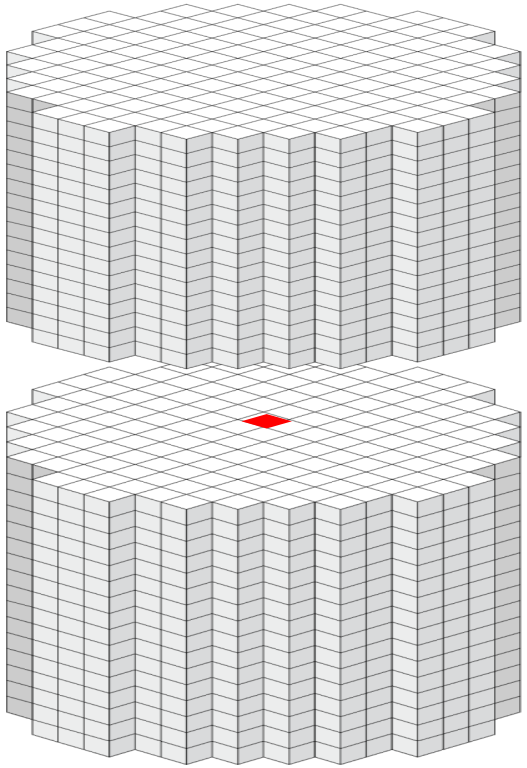
$$\begin{bmatrix} S_1(\boldsymbol{r},\omega) \\ S_2(\boldsymbol{r},\omega) \end{bmatrix} = \boldsymbol{\phi}_r(\boldsymbol{r})\delta\Sigma_r(\boldsymbol{r},\omega) + \boldsymbol{\phi}_a(\boldsymbol{r})\begin{bmatrix} \delta\Sigma_{a,1}(\boldsymbol{r},\omega) \\ \delta\Sigma_{a,2}(\boldsymbol{r},\omega) \end{bmatrix} + \boldsymbol{\phi}_f^{crit}(\boldsymbol{r},\omega)\begin{bmatrix} \delta\upsilon\Sigma_{f,1}(\boldsymbol{r},\omega) \\ \delta\upsilon\Sigma_{f,2}(\boldsymbol{r},\omega) \end{bmatrix} + \begin{bmatrix} s_1(\boldsymbol{r},\omega) \\ s_2(\boldsymbol{r},\omega) \end{bmatrix}$$

Input Noise model

Noise models in CORTEX:

- ❑ Absorber of variable strength
- ❑ Travelling perturbation
- ❑ Control rod vibration
- ❑ Fuel assembly vibration
- ❑ Core barrel vibration

Solving a noise problem: thermal absorber of variable strength

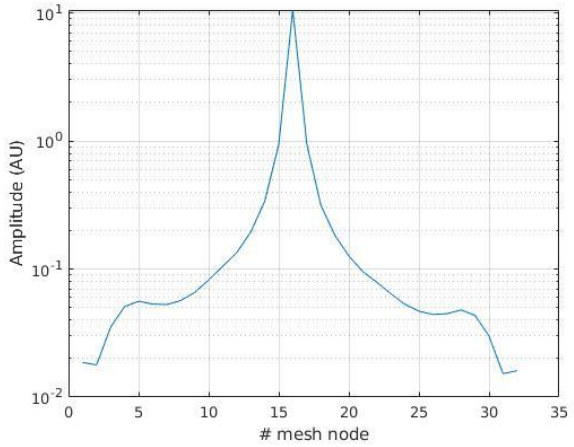


$$\Sigma_{a,2}(r,t) = \Sigma_{a,2,0}(r) + \delta\Sigma_{a,2}(r,t)$$

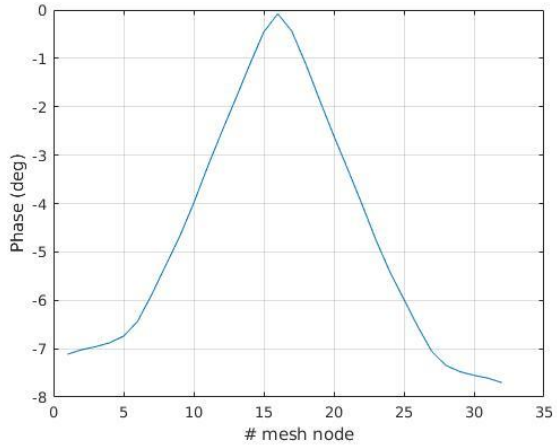
$$\delta\Sigma_{a,2}(r,t) = Asin(\omega t + \phi) = \left(\mathrm{c} \times \Sigma_{\mathrm{a},2,0}(r)\right) sin(\omega t + \phi)$$

$$with\ c \ll 1.\ \text{For the sake of simplicity } \phi = 0$$

$$\delta\Sigma_{\mathrm{a},2}(\mathrm{r},\omega) = {}^{c}/_{2} \times \Sigma_{\mathrm{a},2,0}(r)\delta(\omega - \omega_0)$$



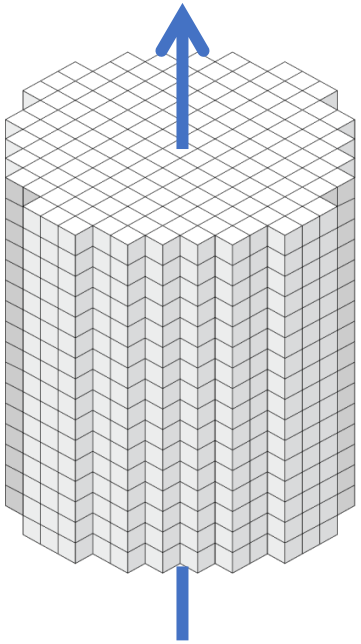
Amplitude of the fast noise



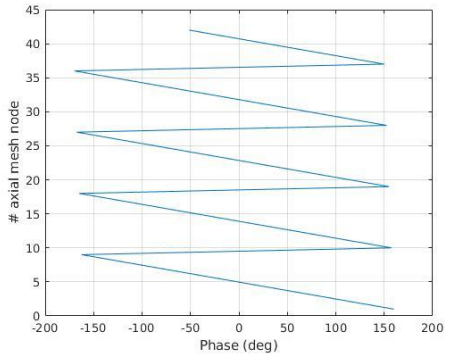
Phase of the fast noise

Solving a noise problem: travelling perturbation

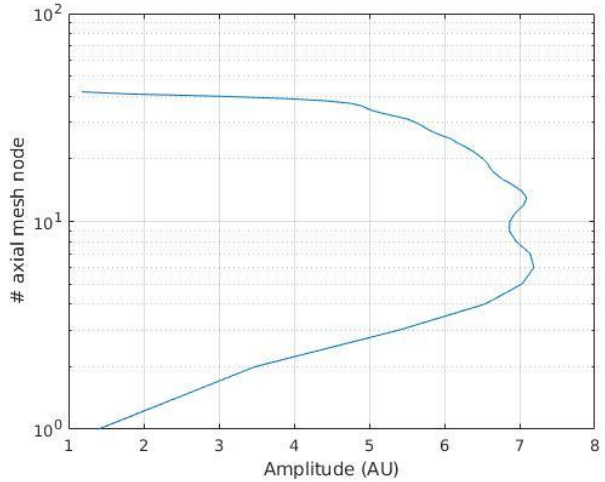
Perturbation leaving the core



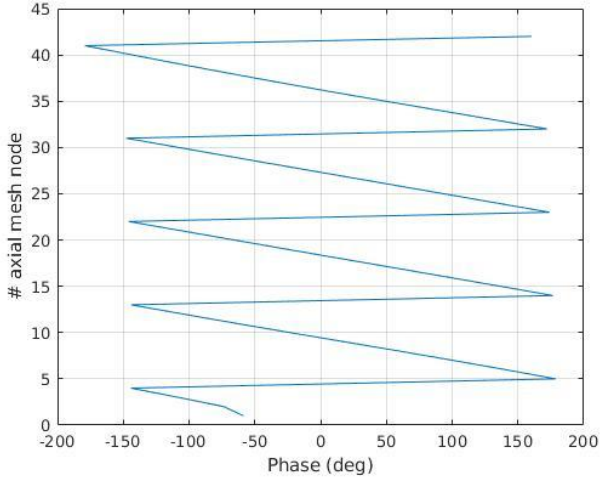
Perturbation entering the core



Phase of the fast source



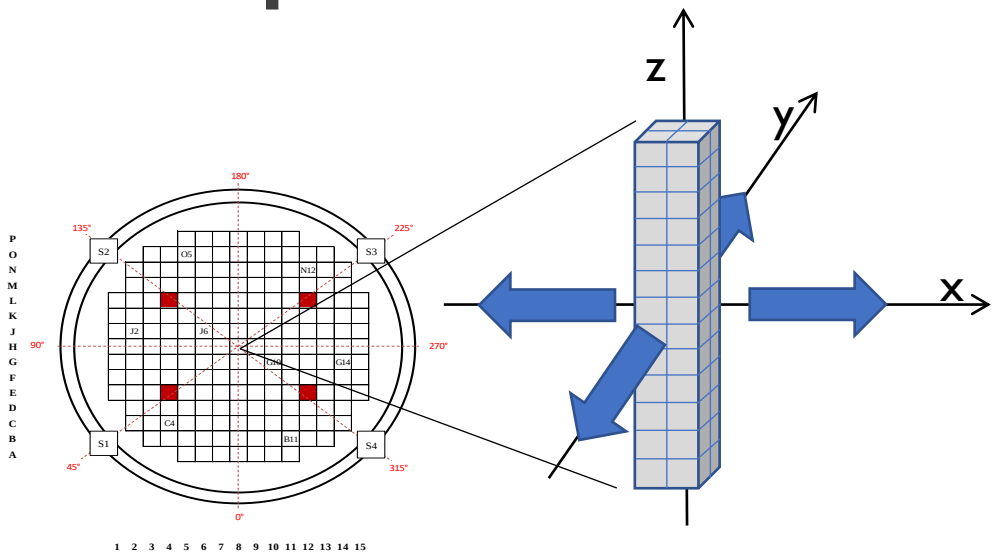
Amplitude of the fast noise

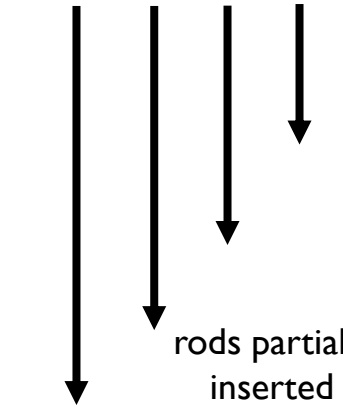


Phase of the fast noise

Solving a noise problem: control-rod vibration

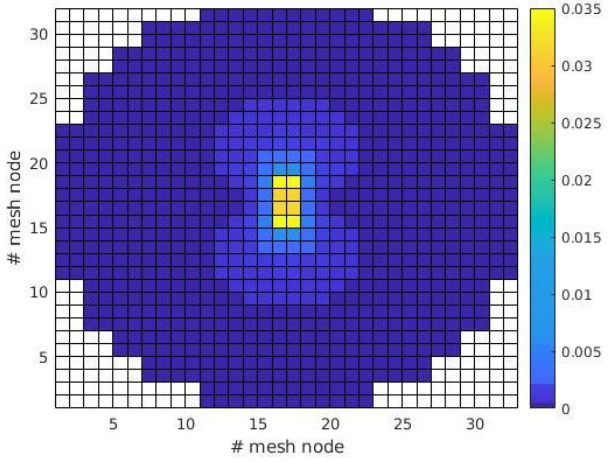
$\varepsilon/_d$ model



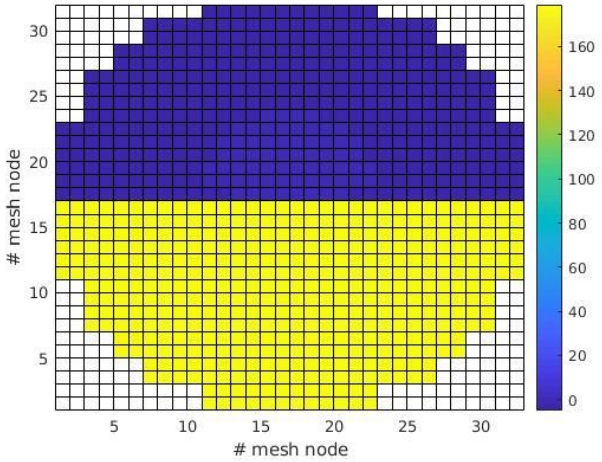


rods partially inserted

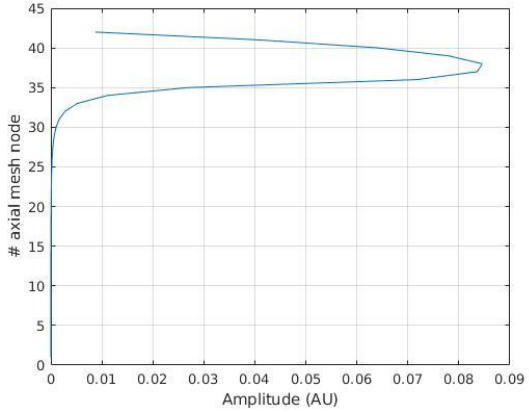
rods fully inserted



Amplitude of the fast noise



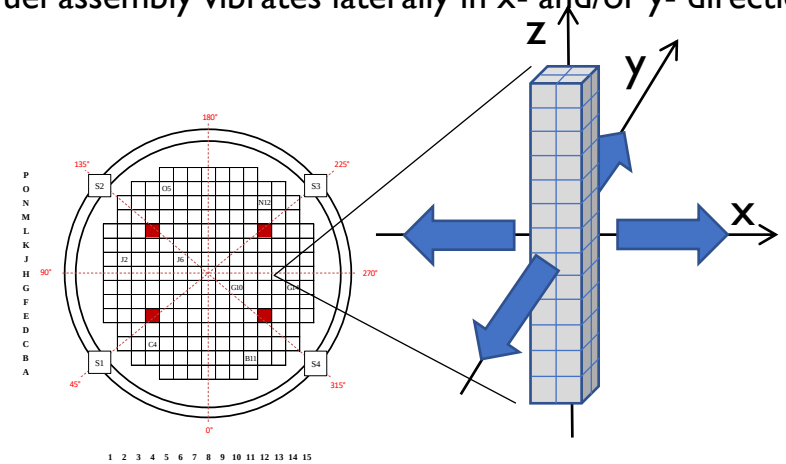
Phase of the fast noise



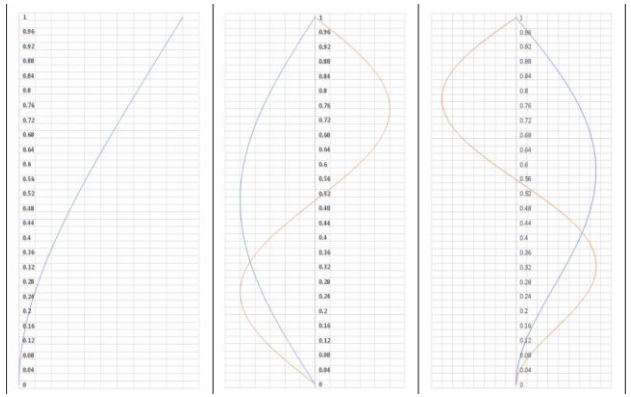
Axial profile of neutron noise amplitude

Solving a noise problem: fuel-assembly vibration

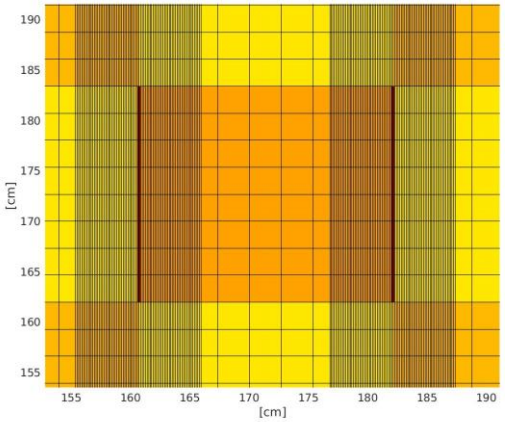
A fuel assembly vibrates laterally in x- and/or y- direction

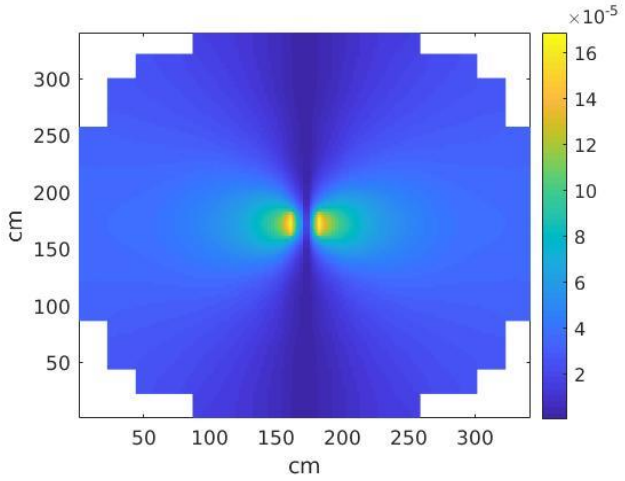


Several axial profiles can be used

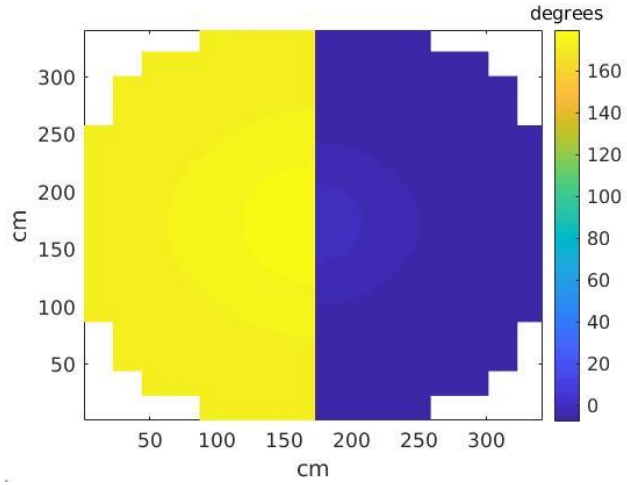


$\varepsilon/_d$ model





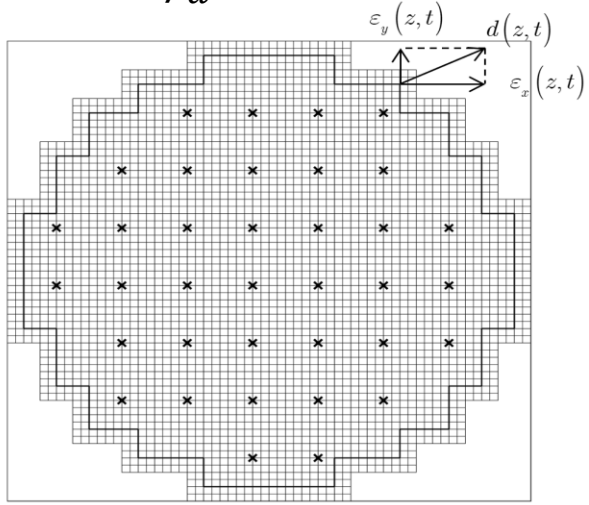
Amplitude of the fast noise

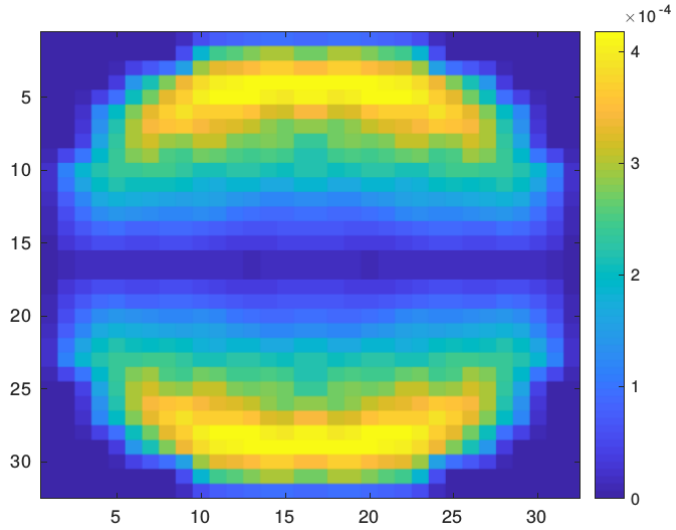


Phase of the fast noise

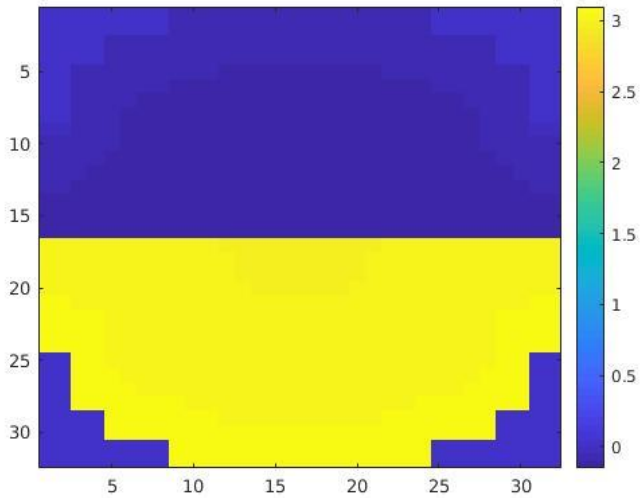
Solving a noise problem: core-barrel vibration

ε/d model





Amplitude of the fast noise



Phase of the fast noise

Noise Green's function generator

$$\begin{bmatrix}\delta\phi_1(r,\omega)\\ \delta\phi_2(r,\omega)\end{bmatrix} = \begin{bmatrix}\int [G_{1\to1}(r,r',\omega)S_1(r',\omega) + G_{2\to1}(r,r',\omega)S_2(r',\omega)]d^3r'\\ \int [G_{1\to2}(r,r',\omega)S_1(r',\omega) + G_{2\to2}(r,r',\omega)S_2(r',\omega)]d^3r'\end{bmatrix}$$

- ❑ The red terms are computed once with the solver
- ❑ The blue terms are specified in a postprocessing stage
- ❑ The noise induced by different scenarios can be computed in the postprocessing stage
- ➢ Using this method we have generated a **~2 TB dataset** for WP3 and a **~5 TB dataset** for WP4

CORE SIMplus in CORTEX

- Contributions to
  - Generation of training data for Machine Learning (task 3.1.3, task 4.2.2)
  - Simulation of COLIBRI experiments (task 2.2.2)
  - Simulation of AKR experiments (task 2.2.1)
  - Comparison with a Sn solver (task 1.3.3)
  - Code-to-code benchmark (task 1.3.3)
  - Uncertainty quantification (task 2.2.3)

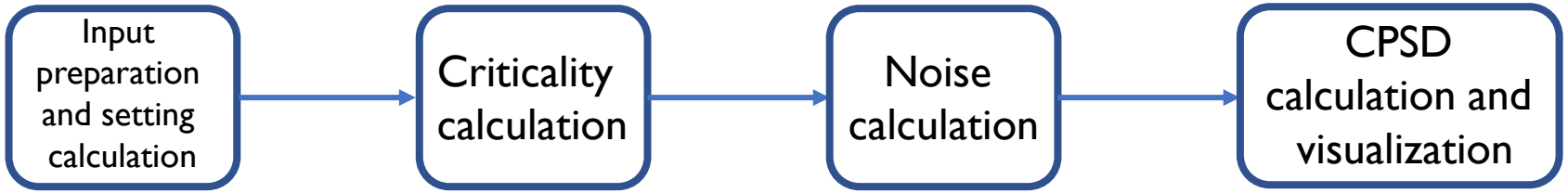
Summary

- CORE SIMplus is fast and flexible diffusion solver
- It can be used for reactor monitoring, diagnostics, experiment design
- Extensive use within CORTEX

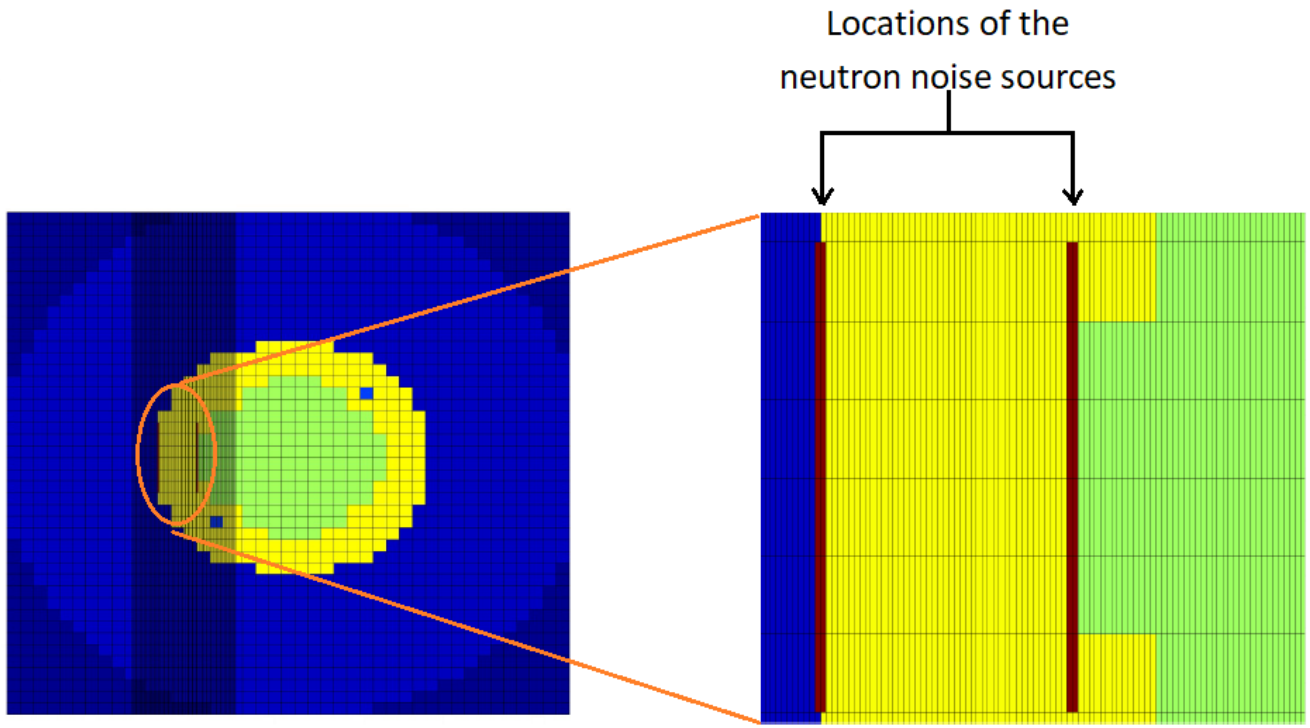
Practical session

Workflow

- Simulation of CROCUS reactor and COLIBRI experiment

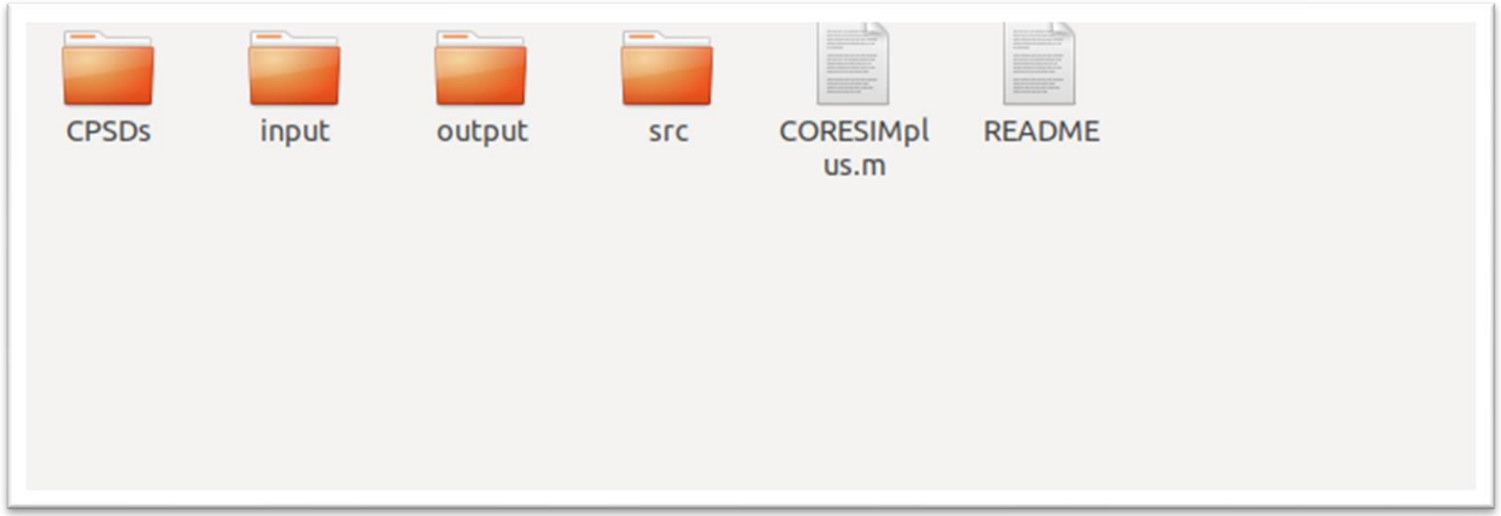


CROCUS and COLIBRI model

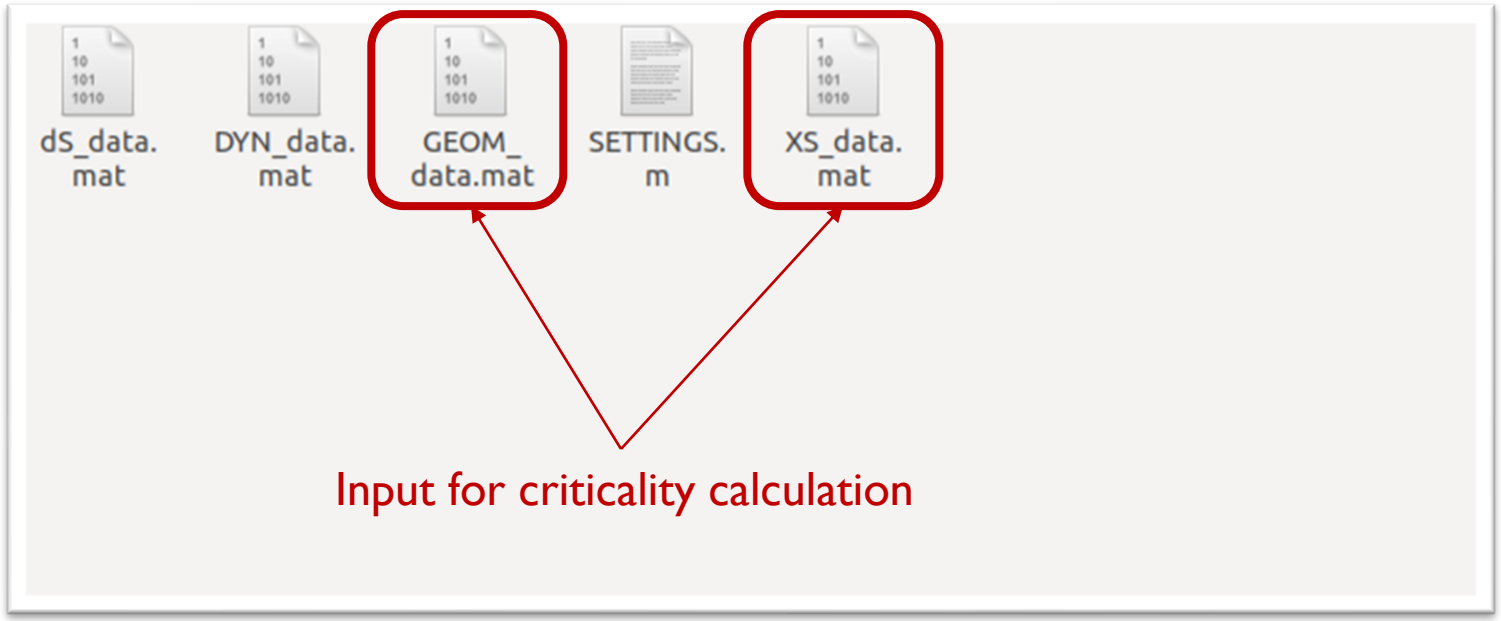


ε/d model

Working directory



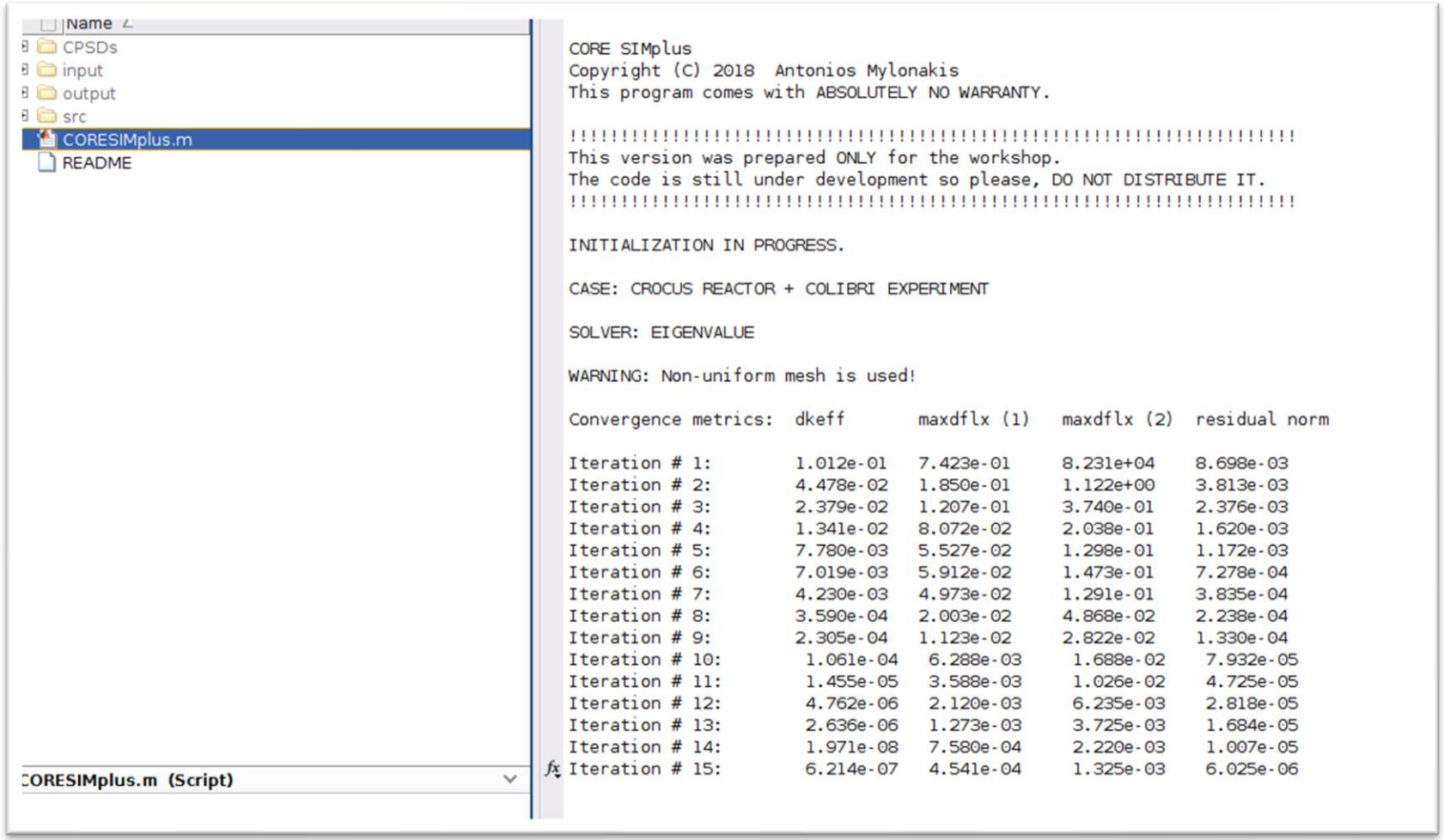
Input directory



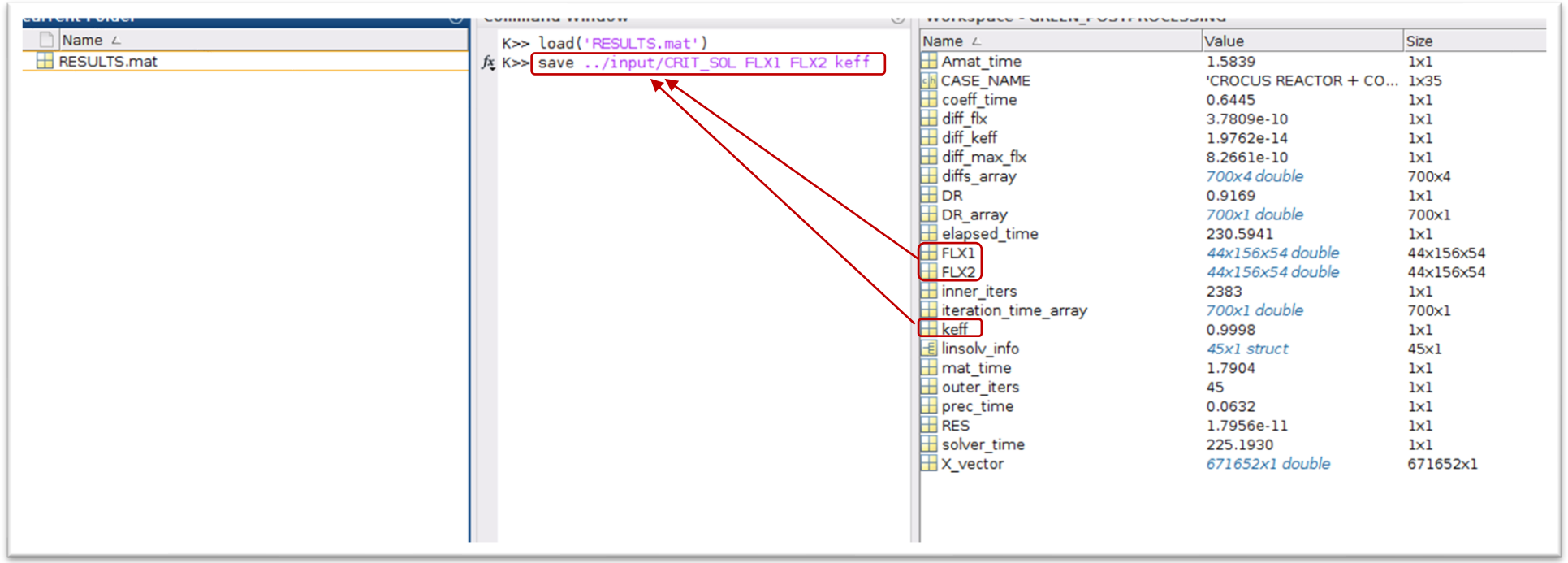
SETTINGS.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SETTINGS file
%
%       DESCRIPTION:
%       Here the user can define various parameters selecting the solver,
%       the acceleration method of the criticality solver,
%       the preconditioner, reflective boundary conditions and numerical
%       tolerances.
%
%       AUTHOR: Antonios Mylonakis (antmyl@chalmers.se)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CASE_NAME='CROCUS REACTOR + COLIBRI EXPERIMENT';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%`
%%Solvers
EIGEN_LARGE_SOLVER_on=1;
FS_LARGE_SOLVER_on=0;
DYN_CRIT_LARGE_SOLVER_on=0;
DYN_SUBCRIT_LARGE_SOLVER_on=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%special parameters
%Acceleration of the criticality solver
CHEBYSEV_ACC_on=1;
JFNK_ACC_on=0;

%Preconditioning of the linear solver
SGS_on=0;
PRECONDITIONING_off=0;

%reflective boundary conditions
REFLECTIVE_on=0; %!!!!!!!!!!!!!!ONLY FOR 2D and 3D
x_right_off=0;
x_left_off=0;
y_right_off=0;
y_left_off=0;

%Convergence parameters
%-------------------------------------
%parameters of the GMRES linear solver
GMRES_tol=10^-9;
GMRES_restart=30;
GMRES_maxits=100;
%parameter of the power iteration
PI_tol=10^-9;

%output
PARAVIEW_on=0;
```

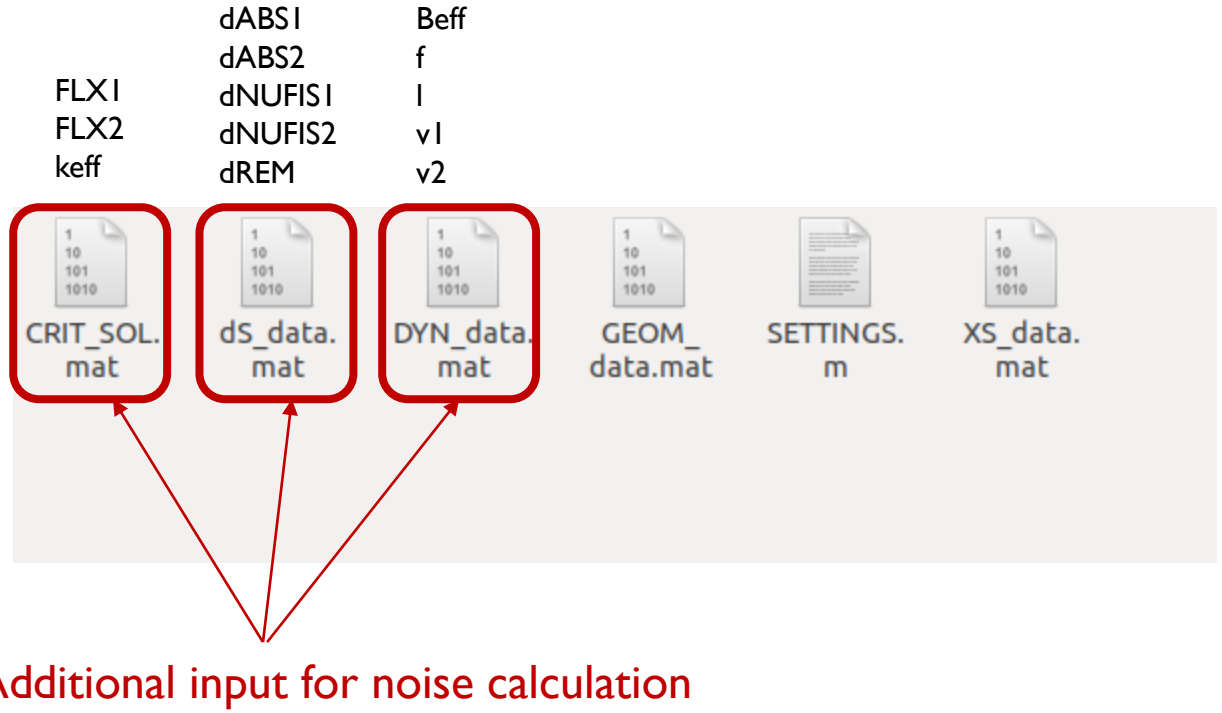
Run criticality calculation



Results


Name
RESULTS.mat
K>> load('RESULTS.mat')
K>> save ../input/CRIT_SOL FLX1 FLX2 keff
Name
Value
Size
Amat_time 1.5839 1x1
CASE_NAME 'CROCUS REACTOR + CO... 1x35
coeff_time 0.6445 1x1
diff_flx 3.7809e-10 1x1
diff_keff 1.9762e-14 1x1
diff_max_flx 8.2661e-10 1x1
diffs_array 700x4 double 700x4
DR 0.9169 1x1
DR_array 700x1 double 700x1
elapsed_time 230.5941 1x1
FLX1 44x156x54 double 44x156x54
FLX2 44x156x54 double 44x156x54
inner_iters 2383 1x1
iteration_time_array 700x1 double 700x1
keff 0.9998 1x1
linsolv_info 45x1 struct 45x1
mat_time 1.7904 1x1
outer_iters 45 1x1
prec_time 0.0632 1x1
RES 1.7956e-11 1x1
solver_time 225.1930 1x1
X_vector 671652x1 double 671652x1

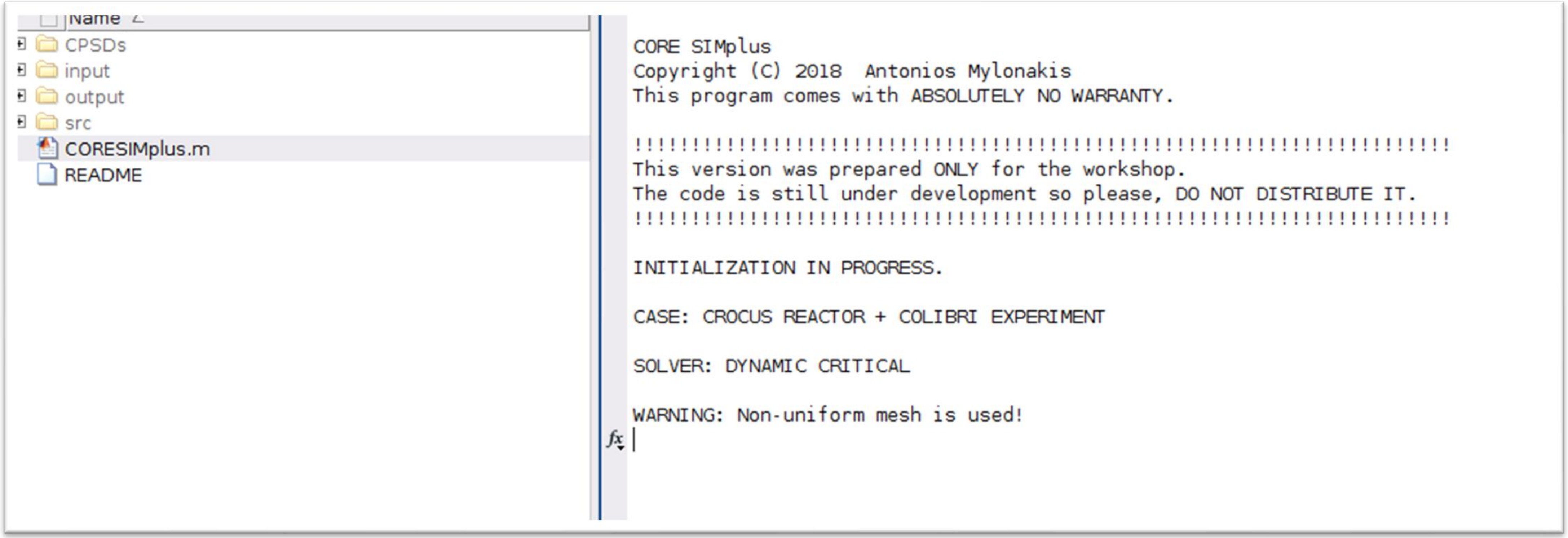
Input directory



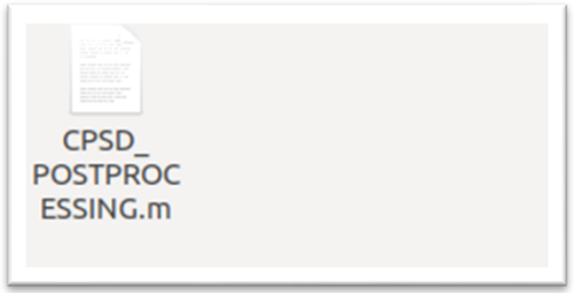
SETTINGS.m

```
%#########################################################################
%       SETTINGS file
%
%       DESCRIPTION:
%       Here the user can define various parameters selecting the solver,
%       the acceleration method of the criticality solver,
%       the preconditioner, reflective boundary conditions and numerical
%       tolerances.
%
%       AUTHOR: Antonios Mylonakis (antmyl@chalmers.se)
%#########################################################################
CASE_NAME='CROCUS REACTOR + COLIBRI EXPERIMENT';

%***********************************`
%%Solvers
EIGEN_LARGE_SOLVER_on=0;
FS_LARGE_SOLVER_on=0;
DYN_CRIT_LARGE_SOLVER_on=1;
DYN_SUBCRIT_LARGE_SOLVER_on=0;
%***********************************

%special parameters
%Acceleration of the criticality solver
CHEBYSEV_ACC_on=1;
JFNK_ACC_on=0;

%Preconditioning of the linear solver
SGS_on=0;
PRECONDITIONING_off=0;

%reflective boundary conditions
REFLECTIVE_on=0; %!!!!!!!!!!!!!!ONLY FOR 2D and 3D
x_right_off=0;
x_left_off=0;
y_right_off=0;
y_left_off=0;

%Convergence parameters
%-----------------------------------
%parameters of the GMRES linear solver
GMRES_tol=10^-9;
GMRES_restart=30;
GMRES_maxits=100;
%parameter of the power iteration
PI_tol=10^-9;

%output
PARAVIEW_on=0;
```

Run noise simulation



CPSDs



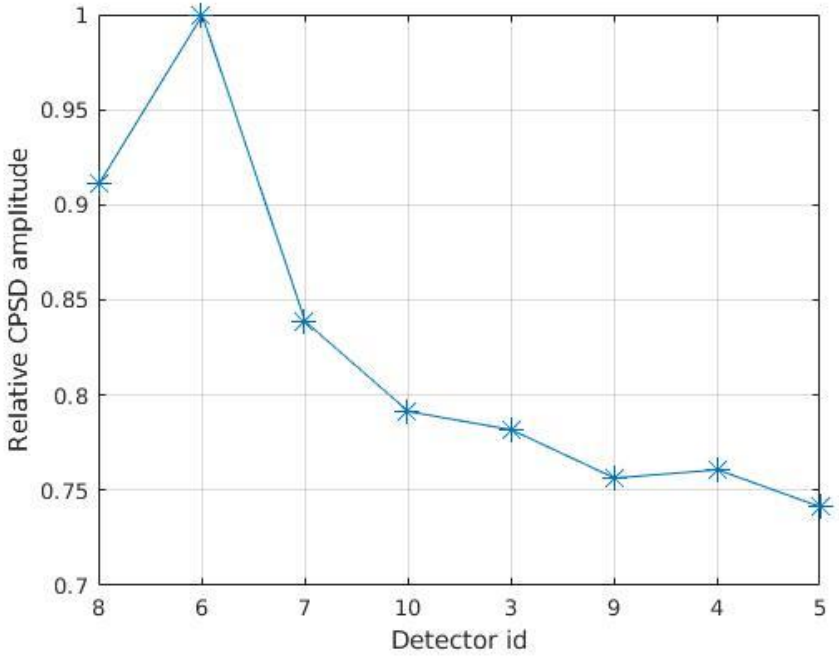
Directory: CPSDs

The script:

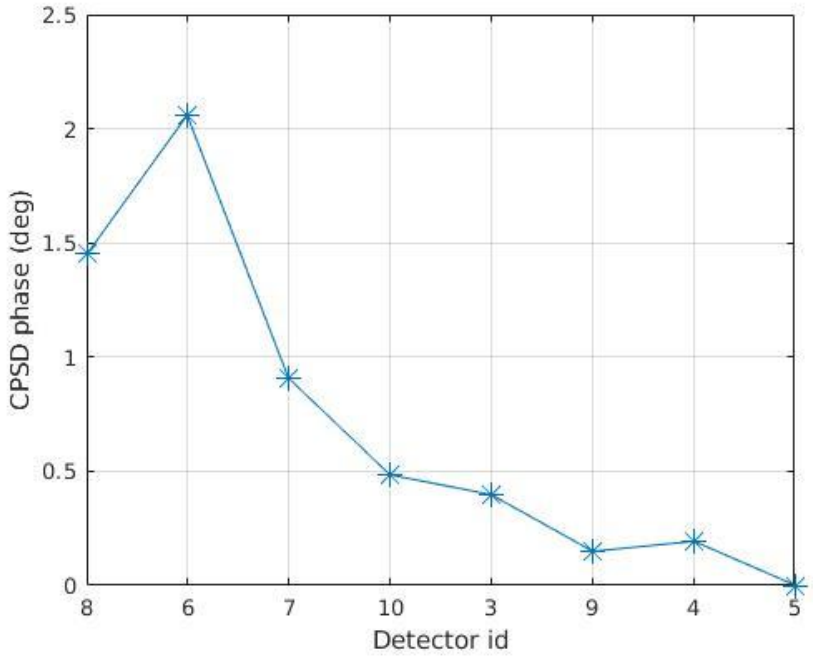
1. loads the computed neutron noise
2. computes the relative neutron noise
3. computes the CPSDs between the detector locations using the formula:

$$CPSD_{2,i,j} = \delta\phi_{2,i}^{rel} \times (\delta\phi_{2,j}^{rel})^{\dagger}$$

CPSD results

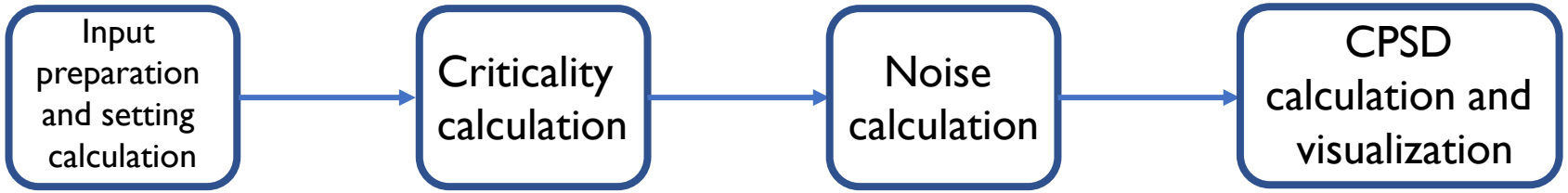


CPSD amplitude based on detector 5
as function of distance from noise source



CPSD phase based on detector 5
as function of distance from noise source

Summary


Input preparation and setting calculation
Criticality calculation
Noise calculation
CPSD calculation and visualization

Thank you