



CORTEX

Core monitoring techniques and
experimental validation and demonstration

Simulating neutron noise with the **CORE SIM**plus 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

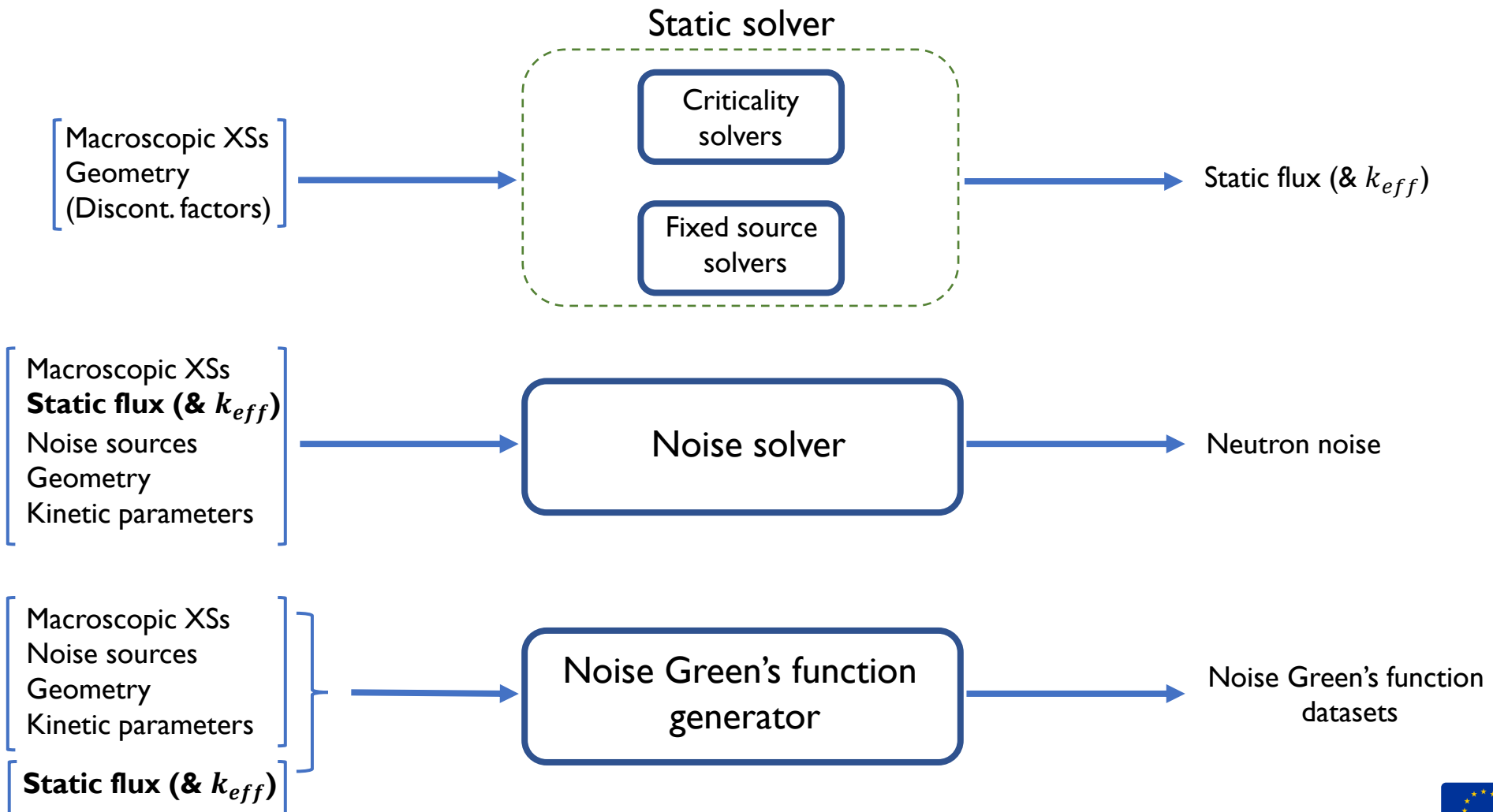
Motivation for developing CORE SIMplus

- 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

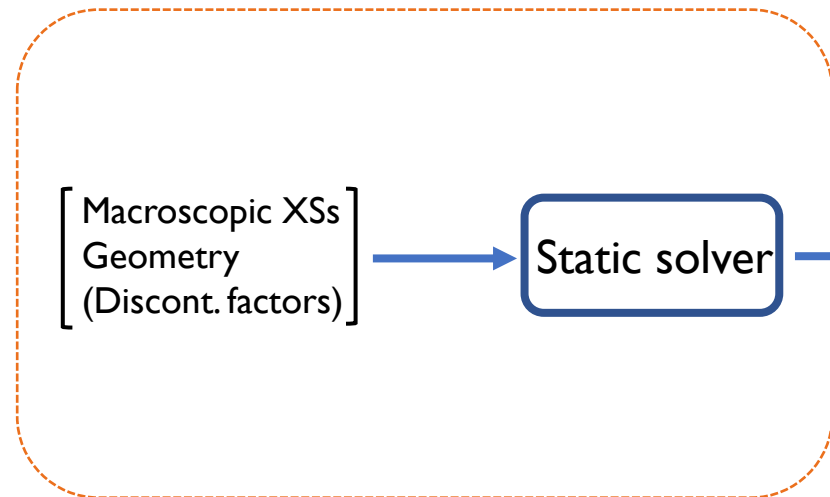
- The code capitalizes on CORE SIM
- Frequency domain solver
- Non uniform mesh
- Various numerical methods
- Noise Green's function generator
- Matlab-based code

Overview of the code

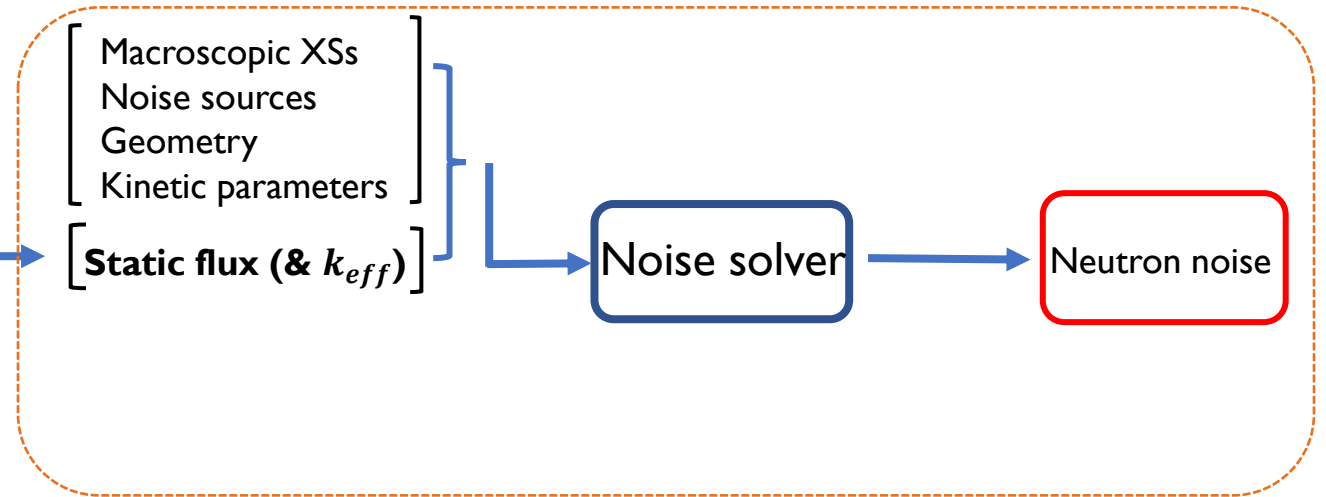


Solving a noise problem

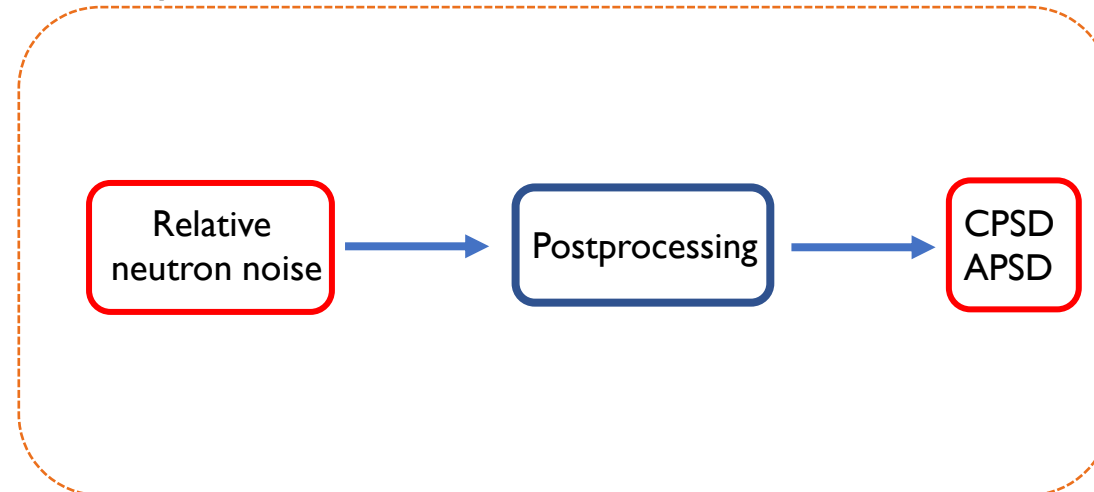
Step 1



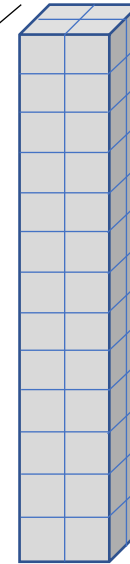
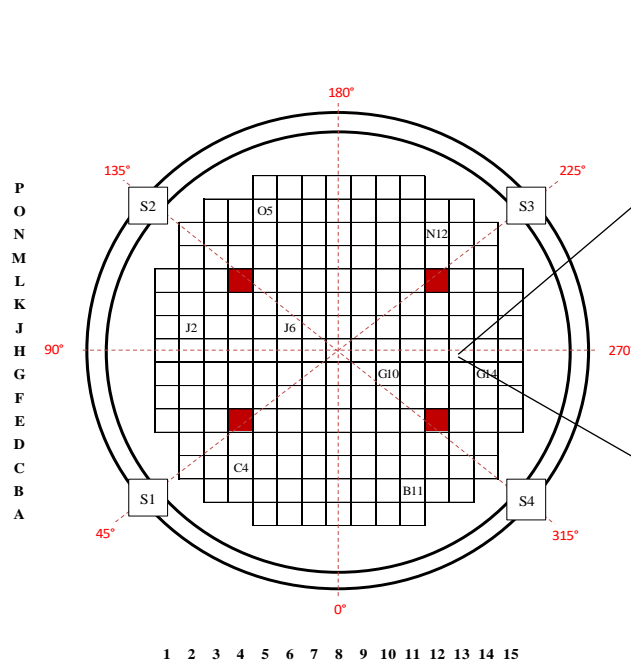
Step 2



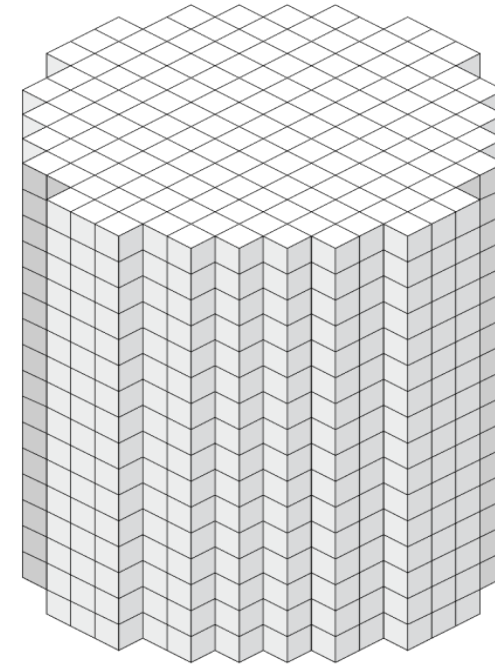
Step 3



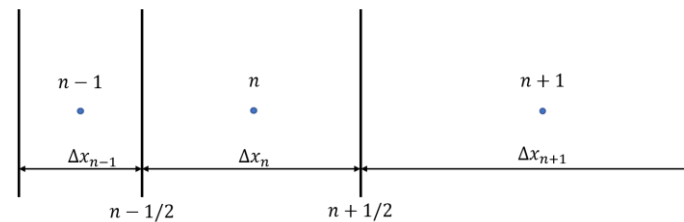
Solving a noise problem: spatial discretization



3D representation
of each fuel assembly



3D representation of the core



Non uniform mesh

Solving a noise problem: noise source models

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

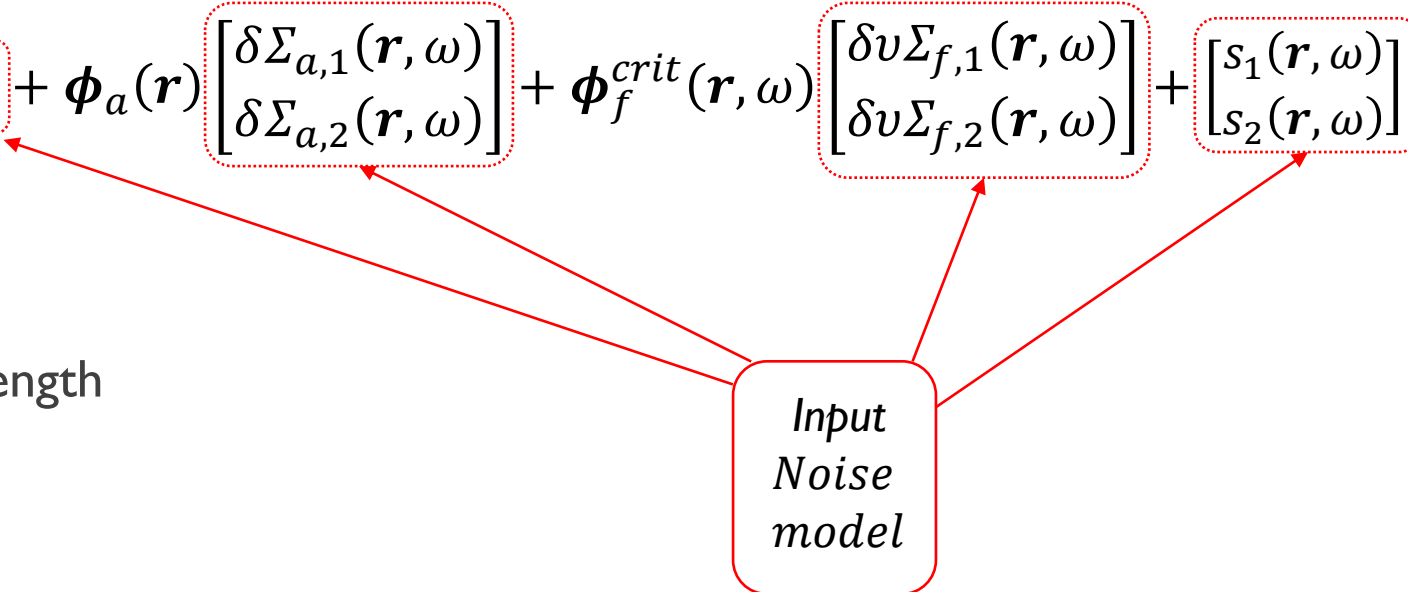
- All the models are built based on the XS fluctuations or arbitrary source terms

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

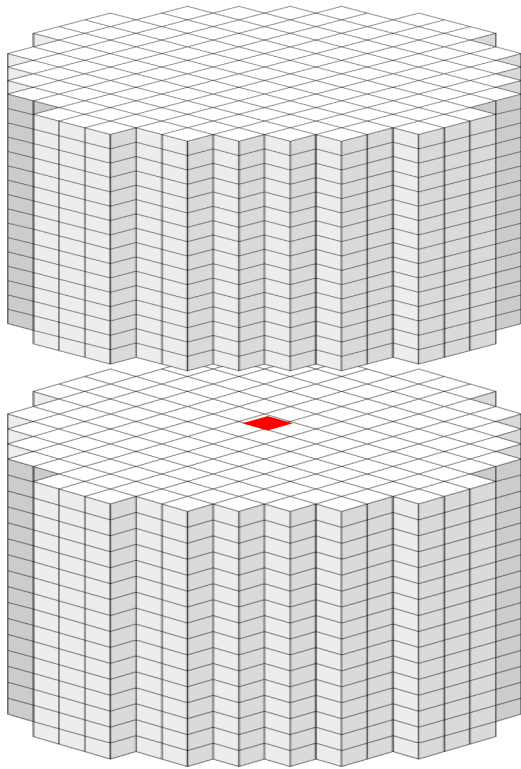
Noise models in CORTEX:

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

Input
Noise
model



Solving a noise problem: thermal absorber of variable strength

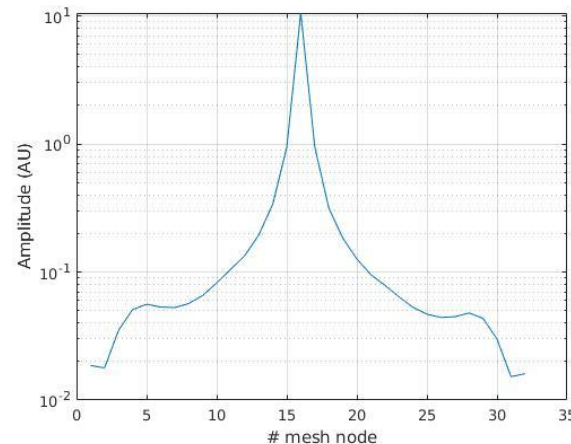


$$\Sigma_{a,2}(r, t) = \Sigma_{a,2,0}(r) + \delta\Sigma_{a,2}(r, t)$$

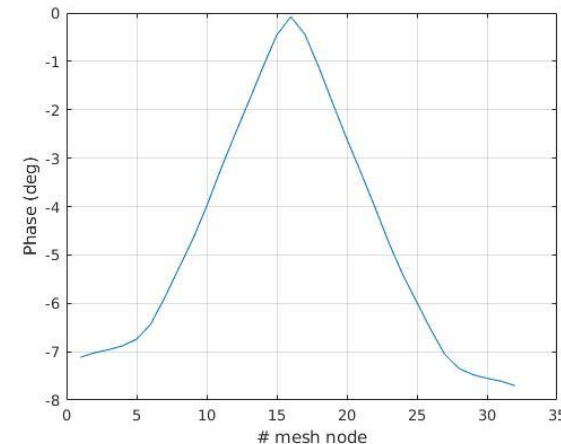
$$\delta\Sigma_{a,2}(r, t) = A \sin(\omega t + \phi) = (c \times \Sigma_{a,2,0}(r)) \sin(\omega t + \phi)$$

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

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



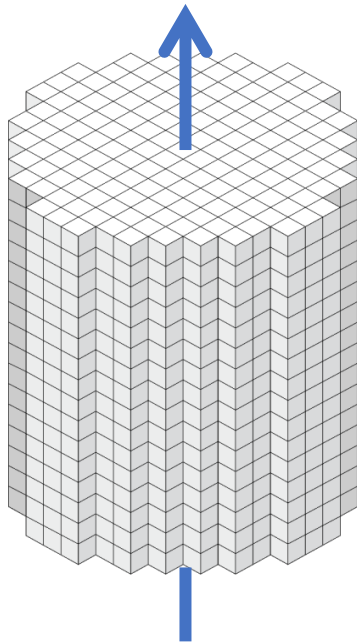
Amplitude of the fast noise



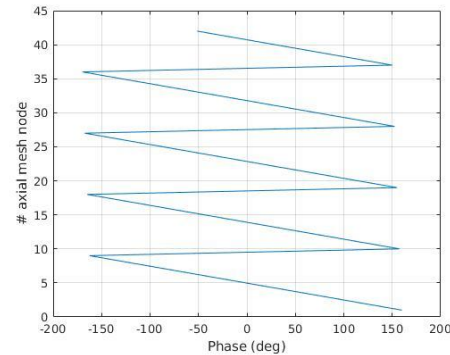
Phase of the fast noise

Solving a noise problem: travelling perturbation

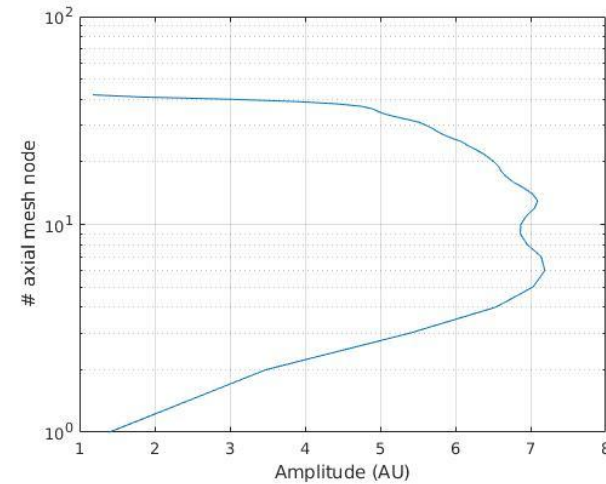
Perturbation leaving the core



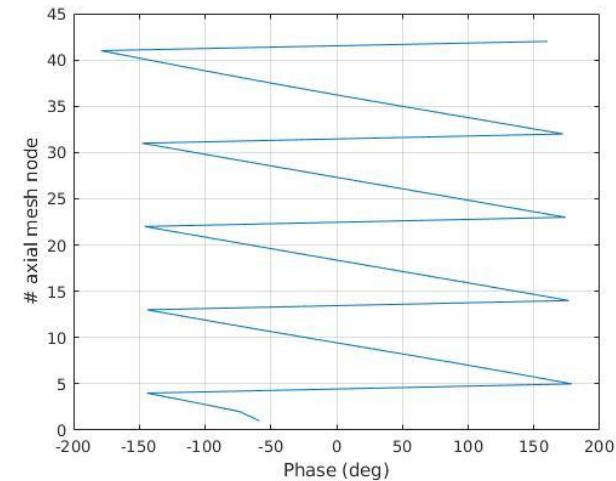
Perturbation entering the core



Phase of the fast source



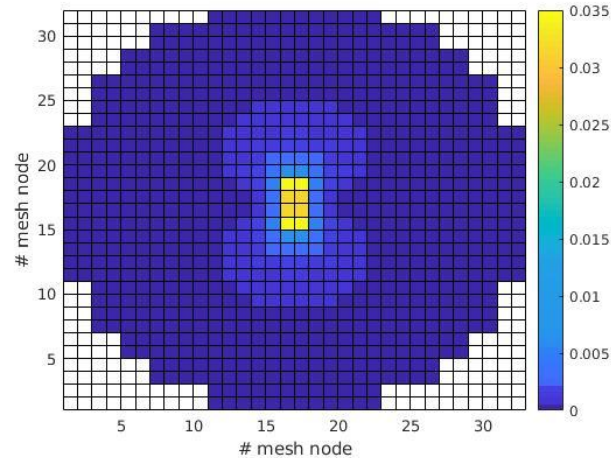
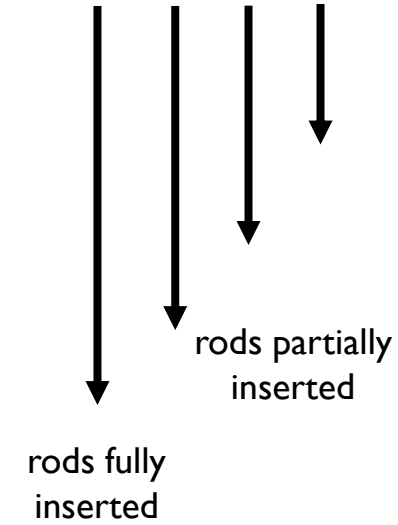
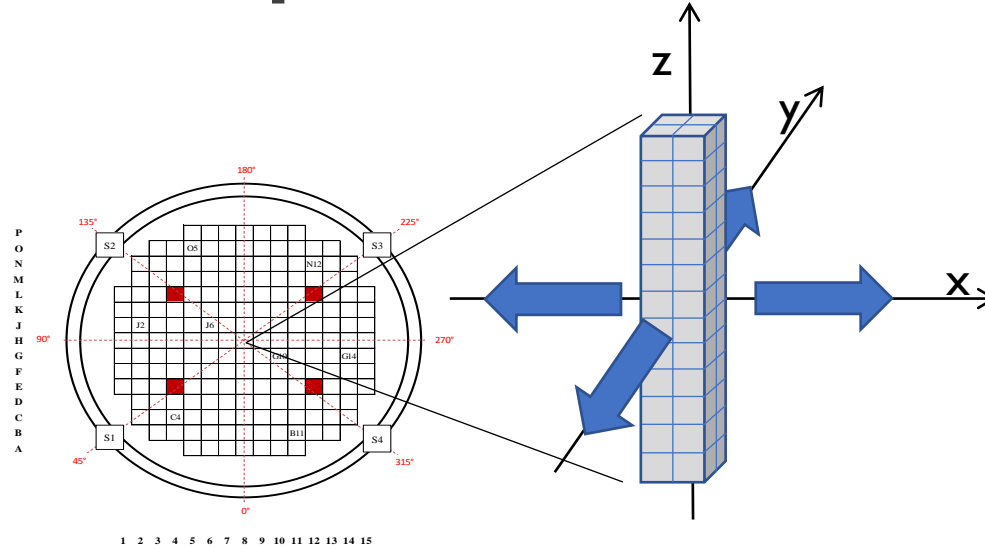
Amplitude of the fast noise



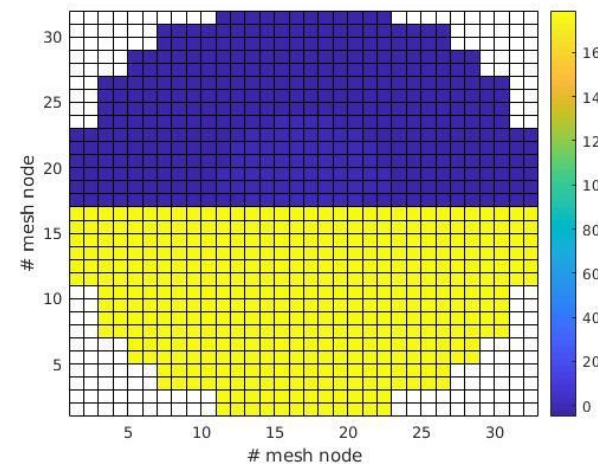
Phase of the fast noise

Solving a noise problem: control-rod vibration

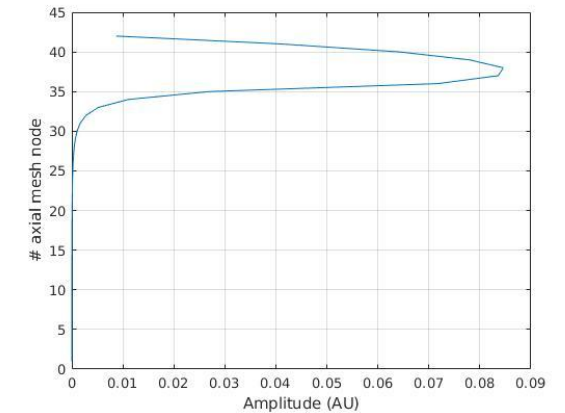
ε/d model



Amplitude of the fast noise



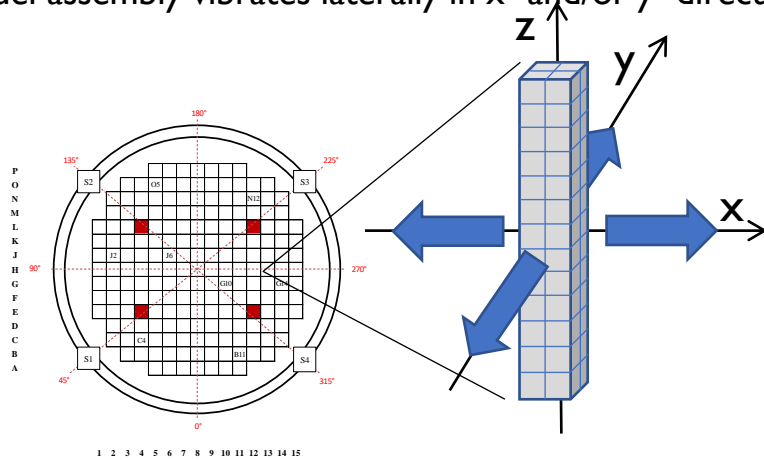
Phase of the fast noise



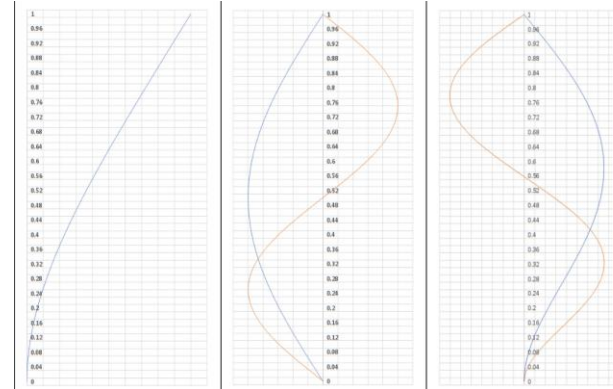
Axial profile of neutron noise amplitude

Solving a noise problem: fuel-assembly vibration

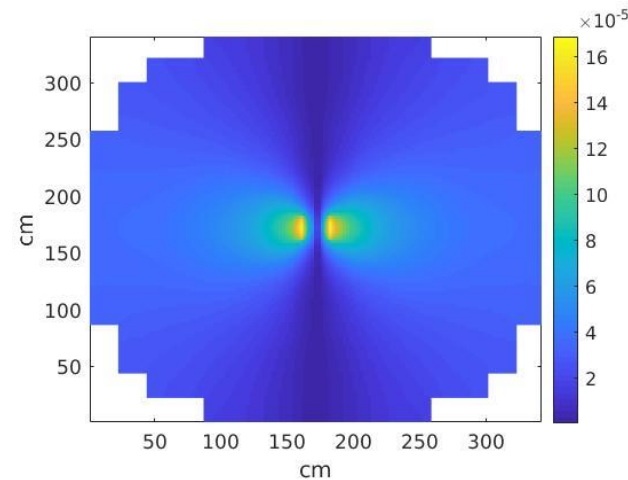
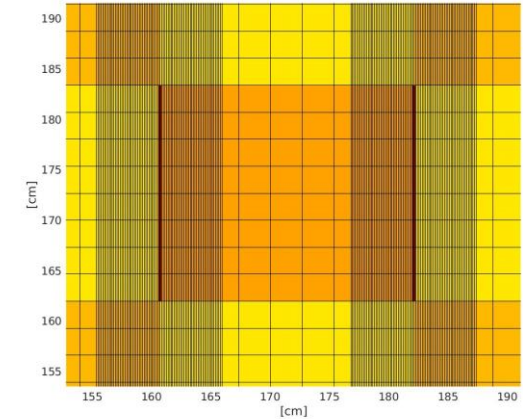
A fuel assembly vibrates laterally in x- and/or y- direction



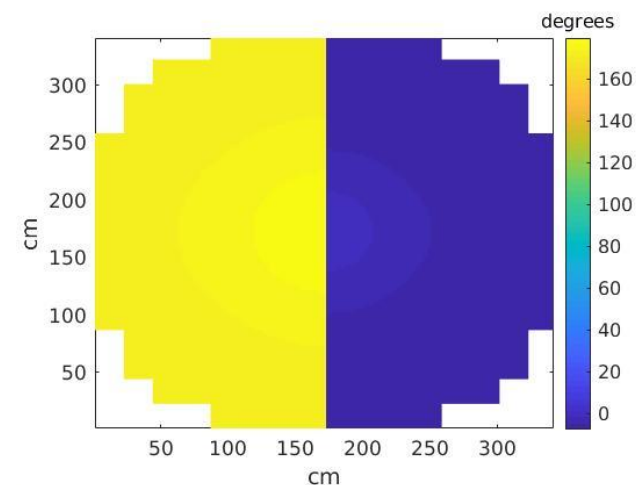
Several axial profiles can be used



ε/d model

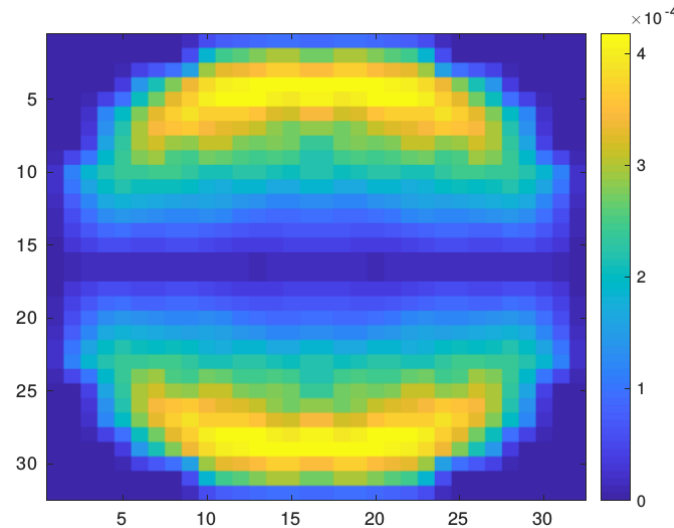
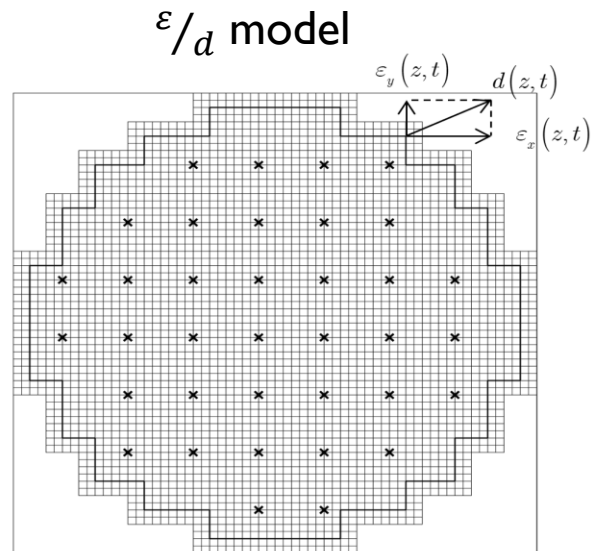


Amplitude of the fast noise

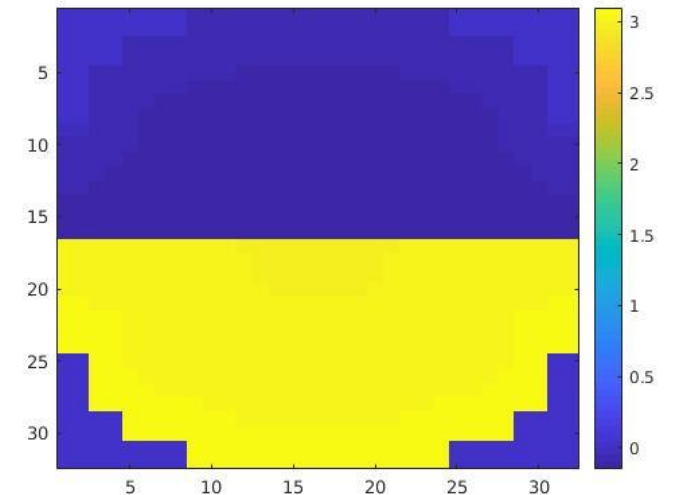


Phase of the fast noise

Solving a noise problem: core-barrel vibration



Amplitude of the fast noise



Phase of the fast noise

Noise Green's function generator

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

- ❑ 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

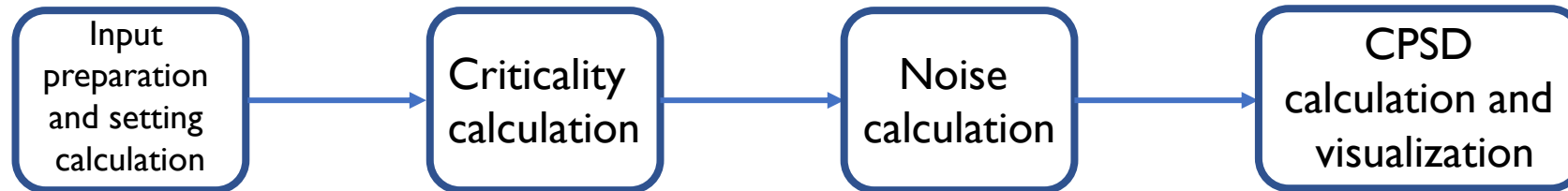
- CORE SIMplus is fast and flexible diffusion solver
- It can be used for reactor monitoring, diagnostics, experiment design
- Extensive use within CORTEX

Practical session

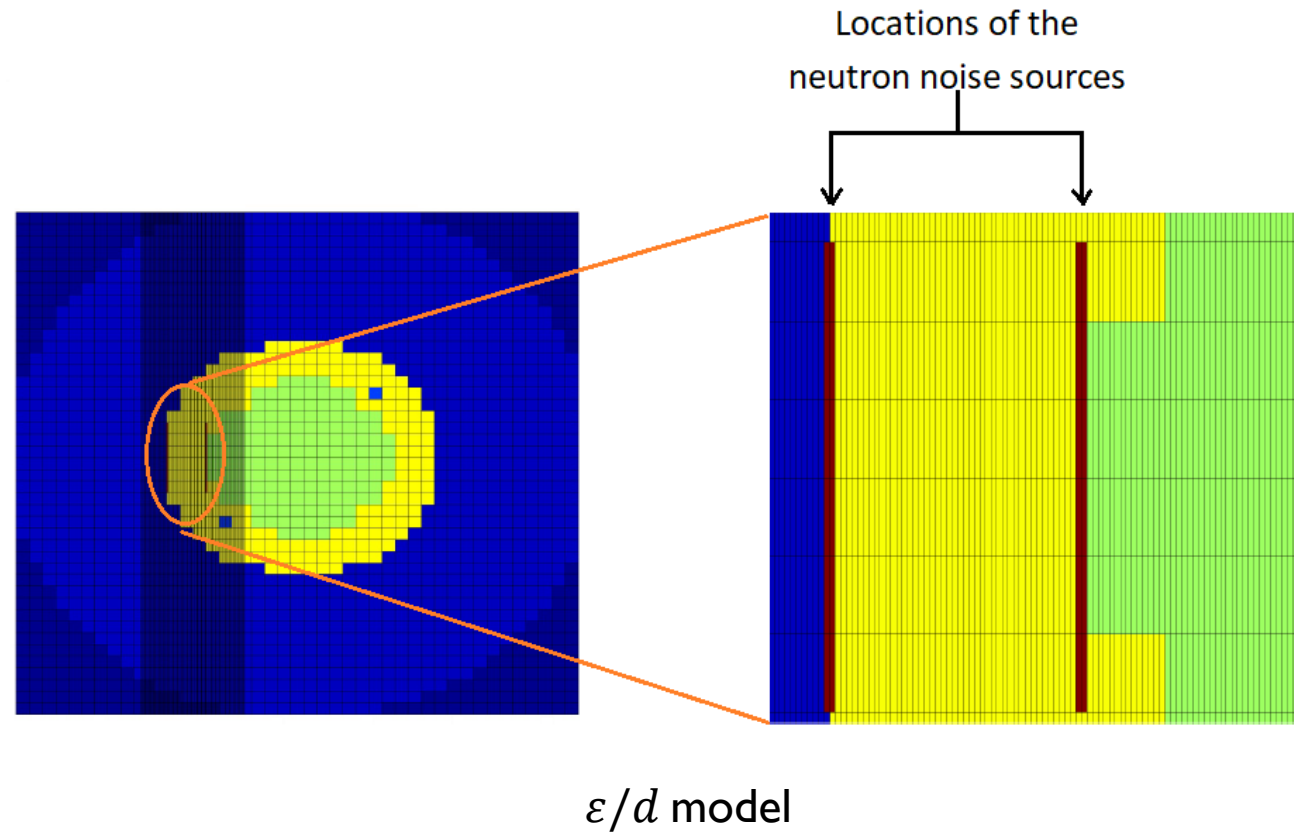


Workflow

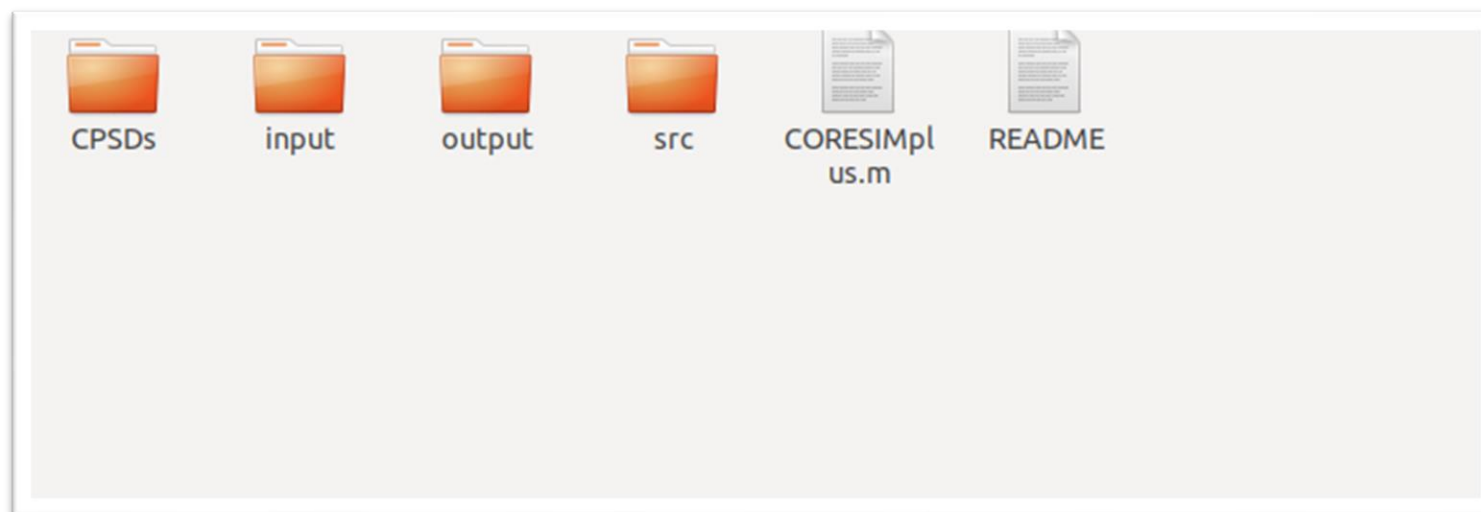
- Simulation of CROCUS reactor and COLIBRI experiment



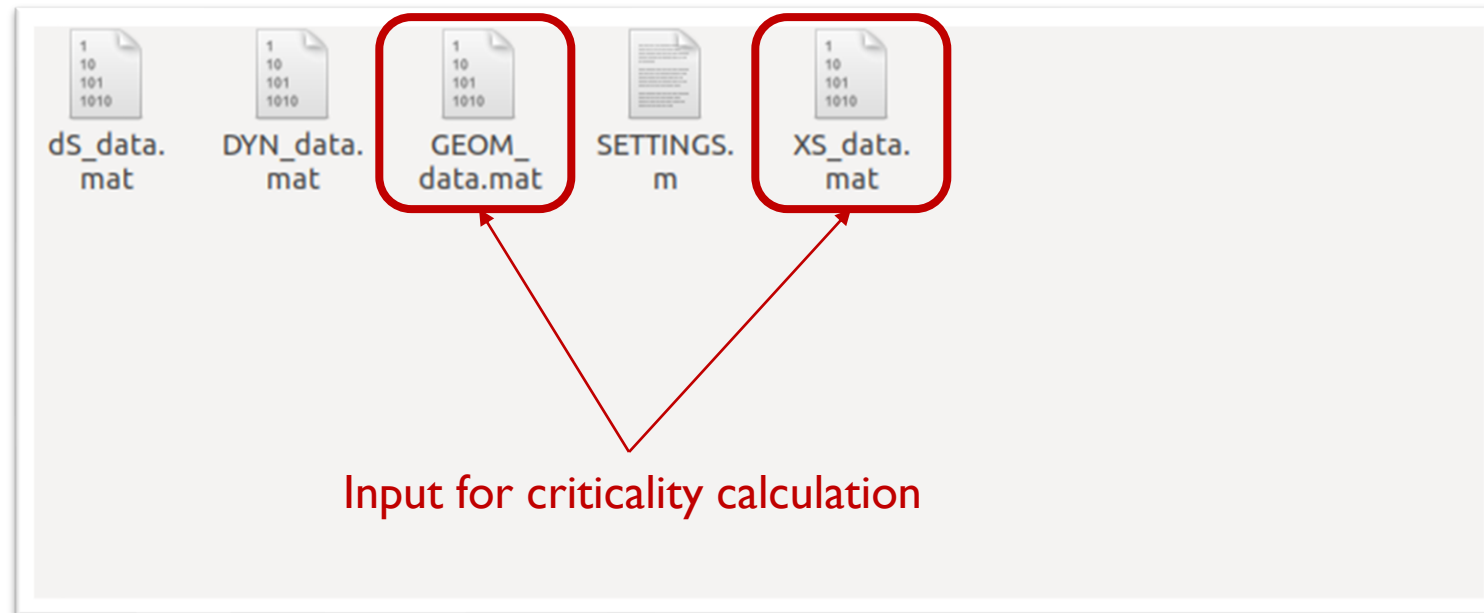
CROCUS and COLIBRI model



Working directory



Input directory

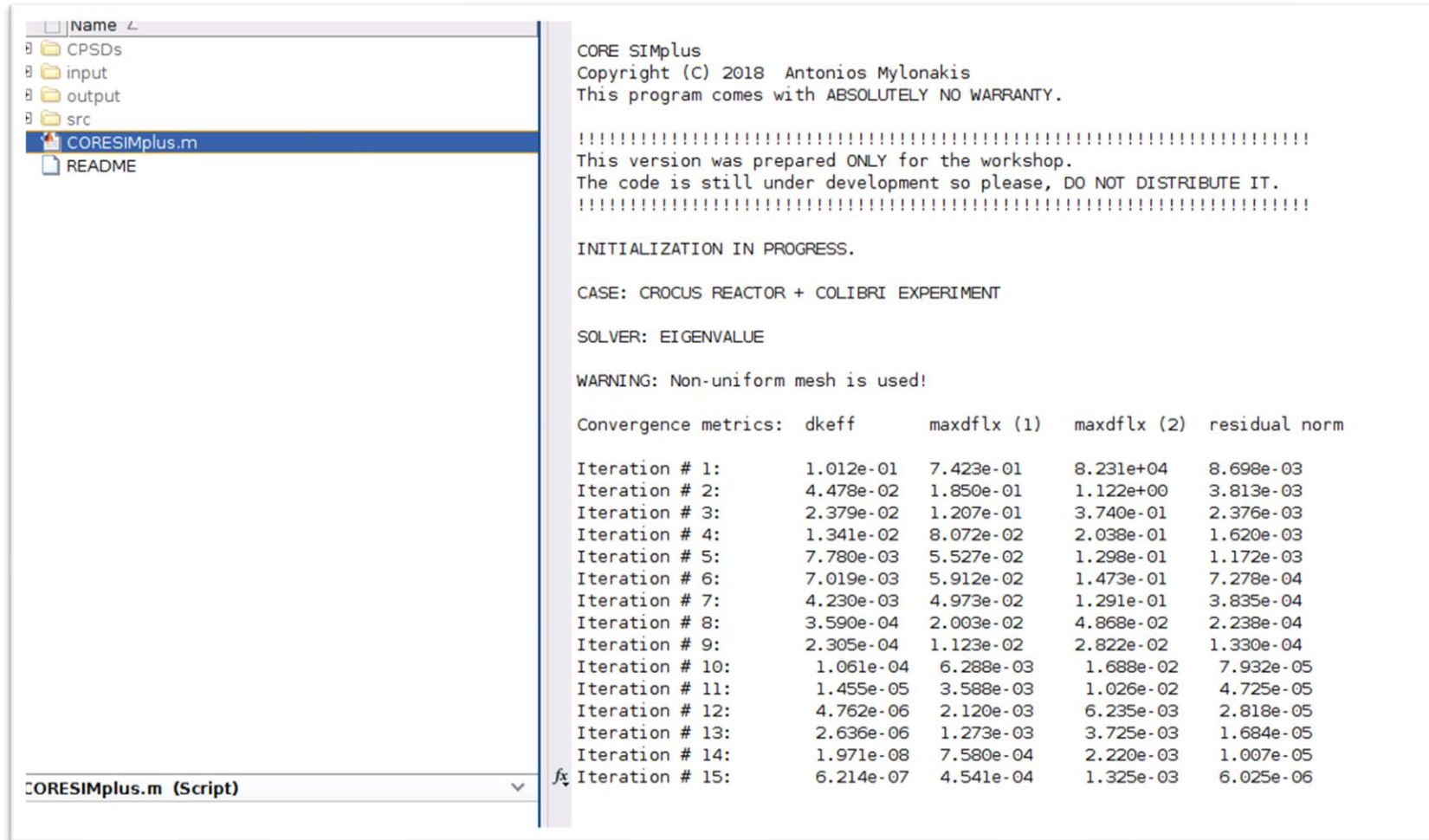


SETTINGS.m

```
1 #####
2 % SETTINGS file
3 %
4 % DESCRIPTION:
5 % Here the user can define various parameters selecting the solver,
6 % the acceleration method of the criticality solver,
7 % the preconditioner, reflective boundary conditions and numerical
8 % tolerances.
9 %
10 % AUTHOR: Antonios Mylonakis (antmyl@chalmers.se)
11 #####
12 CASE_NAME='CROCUS REACTOR + COLIBRI EXPERIMENT';
13
14 %*****
15 %%Solvers
16 EIGEN_LARGE_SOLVER_on=1;
17 FS_LARGE_SOLVER_on=0;
18 DYN_CRIT_LARGE_SOLVER_on=0;
19 DYN_SUBCRIT_LARGE_SOLVER_on=0;
20 %*****
21
22 %special parameters
23 %Acceleration of the criticality solver
24 CHEBYSEV_ACC_on=1;
25 JFNK_ACC_on=0;
26
27 %Preconditioning of the linear solver
28 SGS_on=0;
29 PRECONDITIONING_off=0;
30
31 %reflective boundary conditions
32 REFLECTIVE_on=0; %!!!!!!!!!!!!!!ONLY FOR 2D and 3D
33 x_right_off=0;
34 x_left_off=0;
35 y_right_off=0;
36 y_left_off=0;
37
38 %Convergence parameters
39 %-----
40 %parameters of the GMRES linear solver
41 GMRES_tol=10^-9;
42 GMRES_restart=30;
43 GMRES_maxits=100;
44 %parameter of the power iteration
45 PI_tol=10^-9;
46
47 %output
48 PARAVIEW_on=0;
```



Run criticality calculation



The screenshot shows a MATLAB script execution window. The left pane displays a file explorer with the following structure:

- name
- CPSDs
- input
- output
- src
- CORESImplus.m (selected)
- README

The right pane shows the output of the script:

```
CORE SIMplus
Copyright (C) 2018 Antonios Mylonakis
This program comes with ABSOLUTELY NO WARRANTY.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This version was prepared ONLY for the workshop.
The code is still under development so please, DO NOT DISTRIBUTE IT.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

INITIALIZATION IN PROGRESS.

CASE: CROCUS REACTOR + COLIBRI EXPERIMENT

SOLVER: EIGENVALUE

WARNING: Non-uniform mesh is used!

Convergence metrics: dkeff      maxdflx (1)  maxdflx (2)  residual norm
Iteration # 1:        1.012e-01  7.423e-01   8.231e+04   8.698e-03
Iteration # 2:        4.478e-02  1.850e-01   1.122e+00   3.813e-03
Iteration # 3:        2.379e-02  1.207e-01   3.740e-01   2.376e-03
Iteration # 4:        1.341e-02  8.072e-02   2.038e-01   1.620e-03
Iteration # 5:        7.780e-03  5.527e-02   1.298e-01   1.172e-03
Iteration # 6:        7.019e-03  5.912e-02   1.473e-01   7.278e-04
Iteration # 7:        4.230e-03  4.973e-02   1.291e-01   3.835e-04
Iteration # 8:        3.590e-04  2.003e-02   4.868e-02   2.238e-04
Iteration # 9:        2.305e-04  1.123e-02   2.822e-02   1.330e-04
Iteration # 10:       1.061e-04  6.288e-03   1.688e-02   7.932e-05
Iteration # 11:       1.455e-05  3.588e-03   1.026e-02   4.725e-05
Iteration # 12:       4.762e-06  2.120e-03   6.235e-03   2.818e-05
Iteration # 13:       2.636e-06  1.273e-03   3.725e-03   1.684e-05
Iteration # 14:       1.971e-08  7.580e-04   2.220e-03   1.007e-05
Iteration # 15:       6.214e-07  4.541e-04   1.325e-03   6.025e-06
```

The bottom status bar indicates the script is running: CORESIMplus.m (Script)

Results

Current Folder: RESULTS.mat

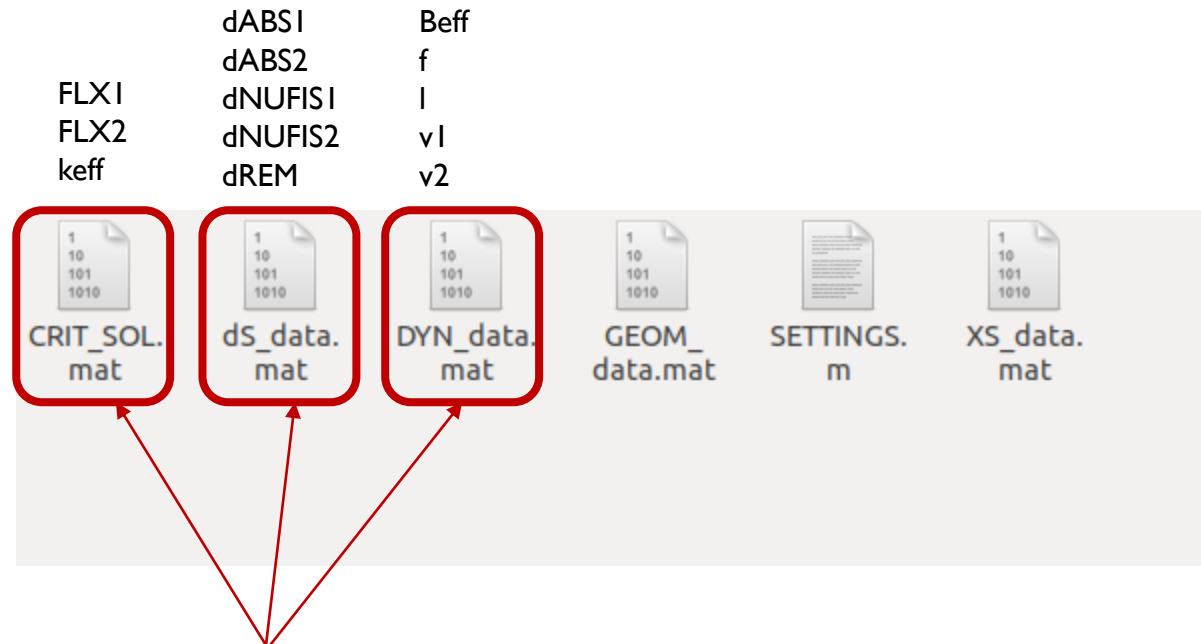
Command Window:

```
K>> load('RESULTS.mat')  
fx K>> save ../input/CRIT_SOL FLX1 FLX2 keff
```

Workspace - CRIT_SOL_PROCESSING

Name	Value	Size
Amat_time	1.5839	1x1
CASE_NAME	'CROCUS REACTOR + CO...	1x35
coeff_time	0.6445	1x1
diff_flux	3.7809e-10	1x1
diff_keff	1.9762e-14	1x1
diff_max_flux	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



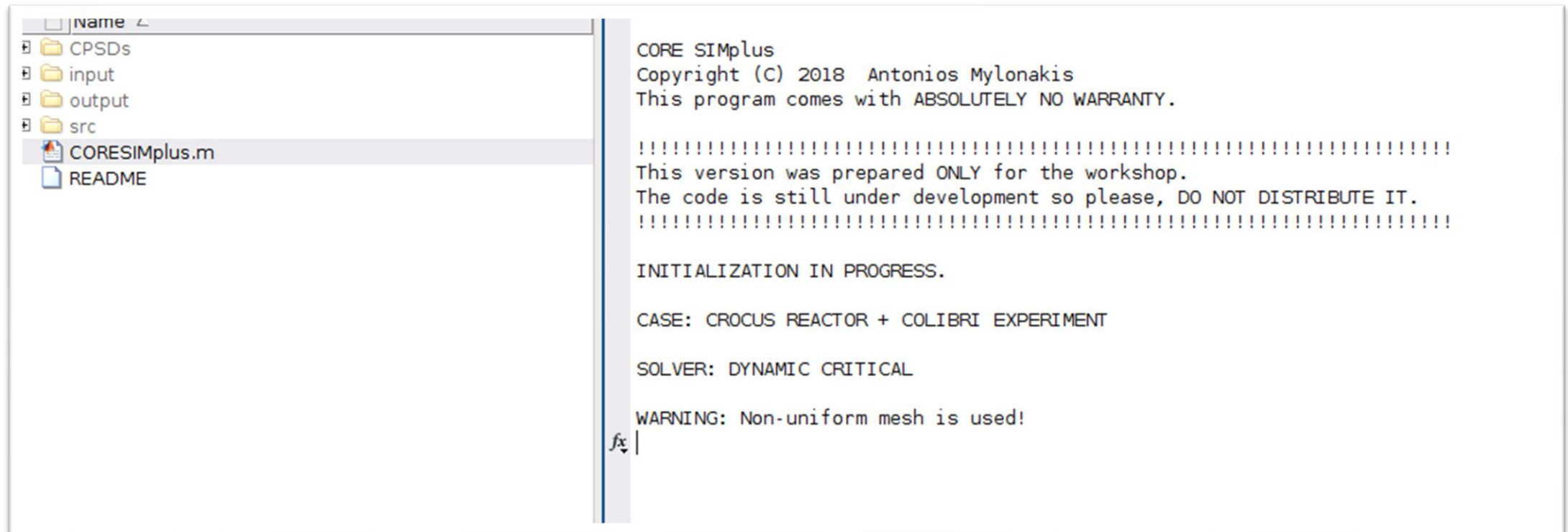
Additional input for noise calculation

SETTINGS.m

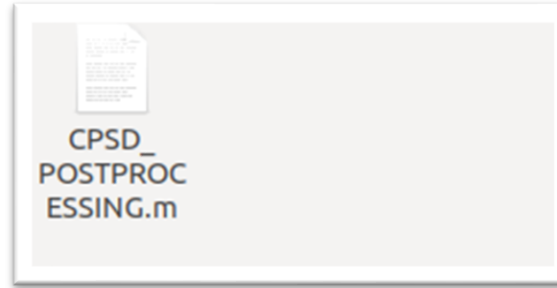
```
1 #####
2 % SETTINGS file
3 %
4 % DESCRIPTION:
5 % Here the user can define various parameters selecting the solver,
6 % the acceleration method of the criticality solver,
7 % the preconditioner, reflective boundary conditions and numerical
8 % tolerances.
9 %
10 % AUTHOR: Antonios Mylonakis (antmyl@chalmers.se)
11 #####
12 CASE_NAME='CROCUS REACTOR + COLIBRI EXPERIMENT';
13
14 %*****
15 %%Solvers
16 EIGEN_LARGE_SOLVER_on=0;
17 FS_LARGE_SOLVER_on=0;
18 DYN_CRIT_LARGE_SOLVER_on=1;
19 DYN_SUBCRIT_LARGE_SOLVER_on=0;
20 %*****
21
22 %special parameters
23 %Acceleration of the criticality solver
24 CHEBYSEV_ACC_on=1;
25 JFNK_ACC_on=0;
26
27 %Preconditioning of the linear solver
28 SGS_on=0;
29 PRECONDITIONING_off=0;
30
31 %reflective boundary conditions
32 REFLECTIVE_on=0; %!!!!!!!!!!!!!!ONLY FOR 2D and 3D
33 x_right_off=0;
34 x_left_off=0;
35 y_right_off=0;
36 y_left_off=0;
37
38 %Convergence parameters
39 %-----
40 %parameters of the GMRES linear solver
41 GMRES_tol=10^-9;
42 GMRES_restart=30;
43 GMRES_maxits=100;
44 %parameter of the power iteration
45 PI_tol=10^-9;
46
47 %output
48 PARAVIEW_on=0;
49
```



Run noise simulation



CPSDs



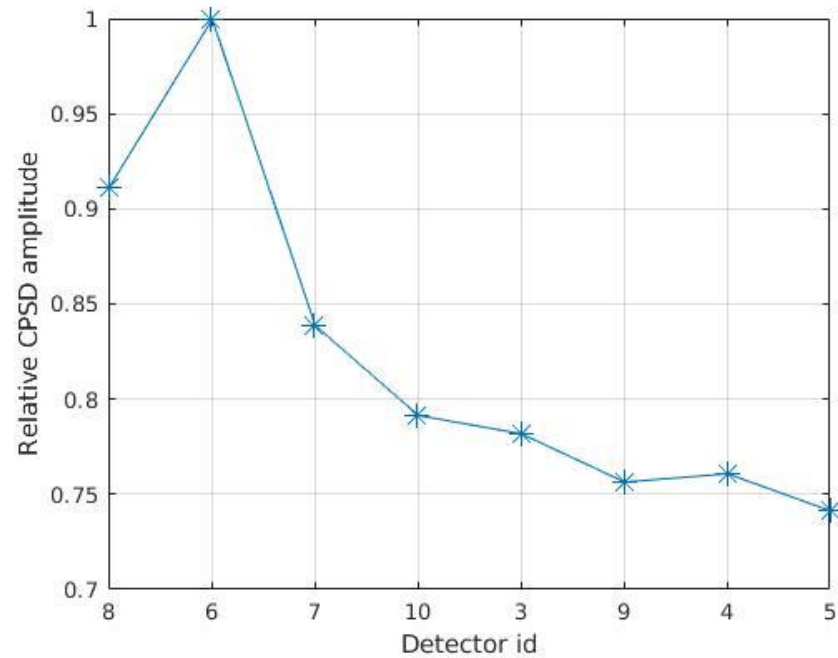
Directory: CPSDs

The script:

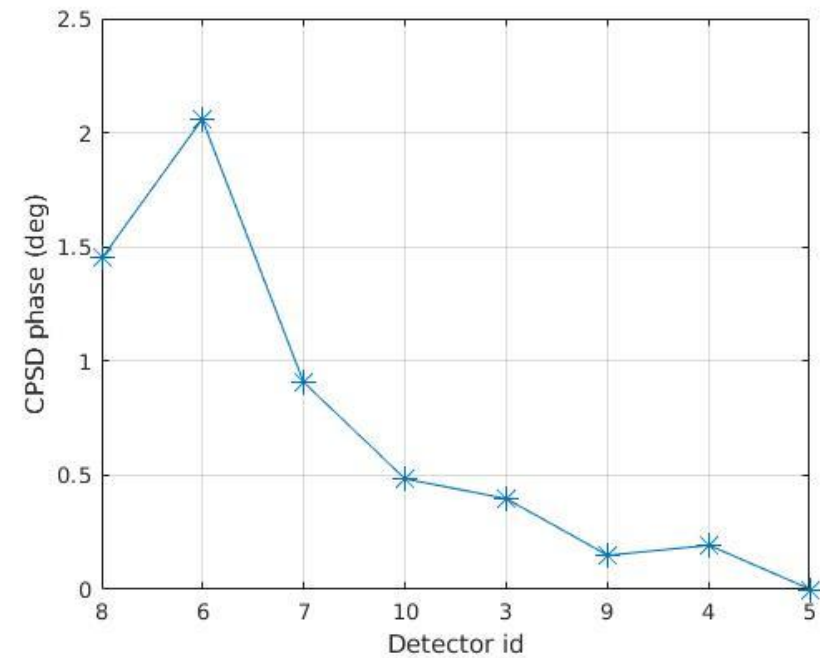
1. loads the computed neutron noise
2. computes the relative neutron noise
3. computes the CPSDs between the detector locations using the formula:

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

CPSD results

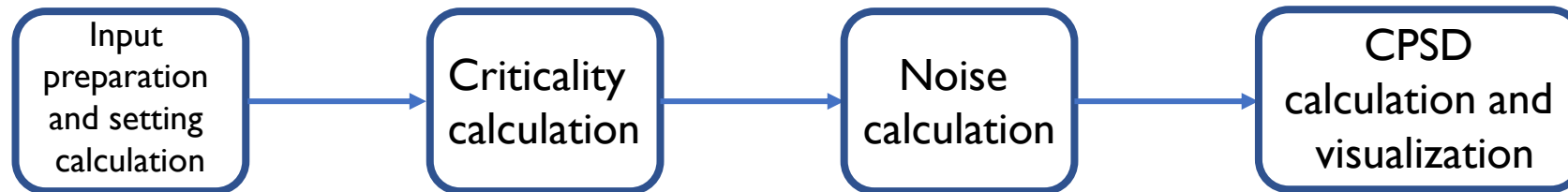


CPSD amplitude based on detector 5
as function of distance from noise source



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

Summary



Thank you

