

# Analysis and interpretation of the first CROCUS reactor neutron noise experiments using an improved point-kinetics model

A. Brighenti<sup>1</sup>, S. Santandrea<sup>1</sup>, I. Zmijarevic<sup>1\*</sup>

<sup>1</sup>DES/ISAS/DM2S/SERMA/LTSD

Université Paris-Saclay, CEA, Service d'études des réacteurs et de mathématiques appliquées,  
91191, Gif-sur-Yvette, France

[alberto.brighenti@cea.fr](mailto:alberto.brighenti@cea.fr); [simone.santandrea@cea.fr](mailto:simone.santandrea@cea.fr); [igor.zmijarevic@cea.fr](mailto:igor.zmijarevic@cea.fr);

## ABSTRACT

In the framework of the European project CORTEX, included in the H2020 program, a new Improved Point-Kinetics (IPK) model has been developed and validated on the neutron noise measurements recorded during the experimental campaigns carried out with the CROCUS reactor, at the École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland. In the first part of this paper, the methodology for the experimental data analysis developed by CEA is presented and its outcomes are compared to those obtained by the EPFL team. In the second part, taking as reference the first CROCUS experimental campaign, the present work presents a series of interpretive exercises performed with the IPK noise model aiming at showing its simulation capabilities and at trying to address some of the discrepancies observed during the validation exercise. With a deeper understanding of the phenomena inside CROCUS, the following step foresees the application of the code to full reactor studies.

KEYWORDS: TRANSPORT CODE, NEUTRON NOISE, CROCUS, POINT KINETICS, DATA ANALYSIS

## 1. INTRODUCTION

Despite being known since the beginning of the nuclear era, neutron noise is a topic of increasing interest for the nuclear research and industrial community. The CORTEX H2020 European project, aims at setting up methodologies and tools to develop non-invasive core monitoring techniques based on neutron noise. In this context, a series of noise experiments has been performed in the CROCUS reactor [1] moving the fuel pins with an oscillating device called COLIBRI [2] and recording the variations of neutron flux with various detectors located in the core. Using different imposed amplitudes ( $A$ ) and frequencies ( $\omega_c$ ) of oscillations [3], twenty different tests have been performed recording signal in selected detectors' positions.

Being a participant of the CORTEX project, CEA already performed various analyses of CROCUS experiences, firstly, by using a simple point-kinetics model [4] and then by developing a more complex noise model based on an Improved Point-Kinetic approach (IPK), validated on the experimental data from the first CROCUS experimental campaign.

After the first part of the work, in which the CEA methodology for the treatment of experimental data is presented, the second part shows some parametric and interpretive exercises performed with IPK model to investigate the dependence of noise amplitude on the frequency of oscillation and on a non-monochromatic mechanical displacement signal.

## 2. ANALYSIS OF EXPERIMENTAL DATA

Apart from the noise model [5], CEA developed its own simple and computationally fast methodology to estimate the detector Cross-Power Spectral Densities (CPSD), whose outcomes have been compared to

those available in literature [6]. This exercise does not aim at substituting previous analyses, but rather at developing the necessary know-how for signal analysis and develop a critical overview of the state of art. The MATLAB scripts used for the analysis can be found in Appendix A. Firstly, the normalized detector signals  $x_i$  is obtained as:

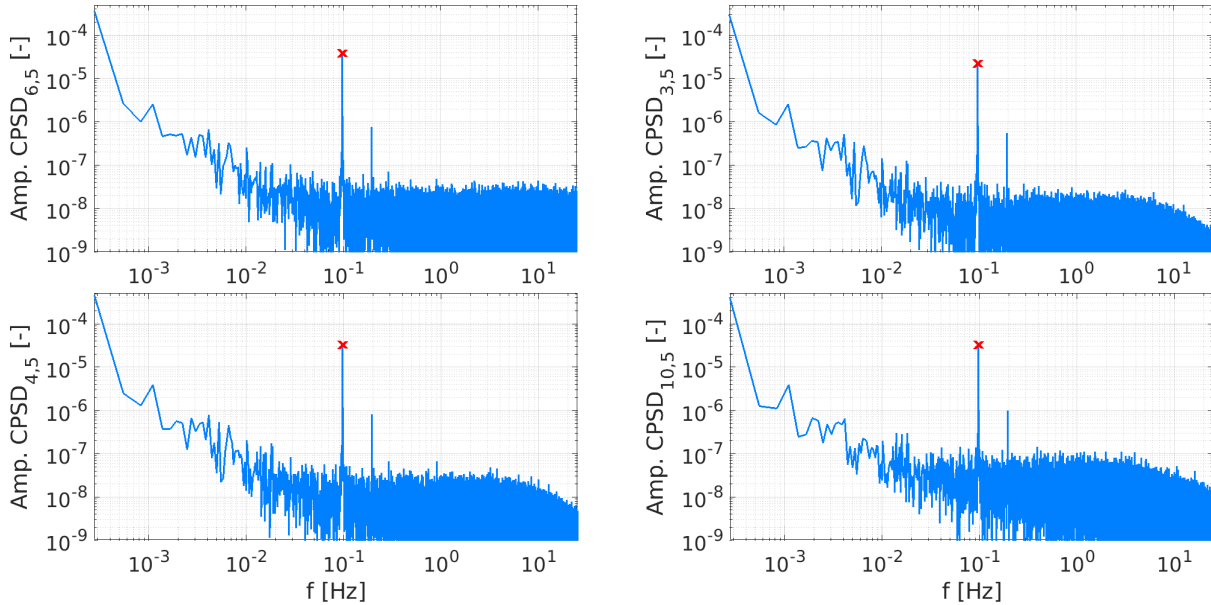
$$x_i(t) = \frac{\tau_{i,raw}(t)}{\bar{\tau}_{i,raw}} \quad (1)$$

where in Eq. (1),  $\tau_{i,raw}$  is the raw temporal signal for detector  $i$ ,  $\bar{\tau}_{i,raw}(t)$  is the arithmetic average of the raw signal. As the detector transfer function is unknown, but assumed to be linear, the chosen normalization allows to eliminate detector dependent quantities (e.g. efficiency) and, therefore, to compare the detectors with each other. Since the mean value chosen for the signal normalization is arbitrary, no major differences are expected, with respect to previous analyses. However, differently from [6], the detector signals are not smoothed using a moving average for two reasons: the first one is to eliminate the additional computational cost required by the averaging procedure, while the second one deals with the preservation of the measurements integrity. The advantage of signal smoothing is that it effectively removes possible impulsive spikes in the signals, however the treated signal results may be altered with the choice of an excessively long averaging window, which by the rule of thumb should be lower than few percent (<5%) of the points available in the oscillation period.

The CPSDs evaluated for all the possible permutation of  $j$ -th normalized detector signal ( $j = 3, \dots, 10$ ) paired with detector #5, see Figure 1, and they are computed as:

$$CPSD_{x,5}(\omega) = Y_j(\omega) \times Y_5^*(\omega) \quad (2)$$

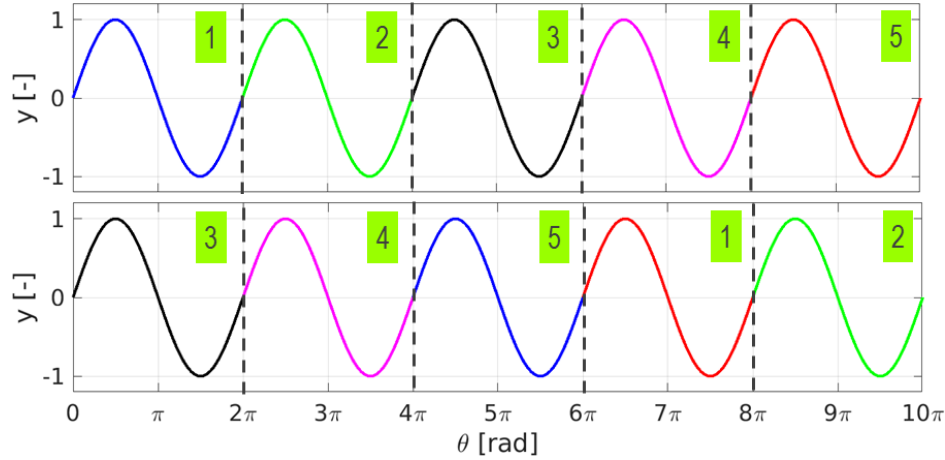
where in Eq. (2),  $Y_j = \mathcal{F}(x_j(t))$  is the Fourier transform  $\mathcal{F}$  of signal  $x_j(t)$  and  $Y_j^*$  is its complex conjugate.



**Figure 1.** *CPSD* amplitude for experiment #12 ( $A = 1.85$  mm,  $\omega_c = 0.097$  Hz) for pairs 6&5 (top left), 3&5 (top right), 4&5 (bottom left) and 10&5 (bottom right). The peak amplitude at  $\omega_c$  is marked with a red cross.

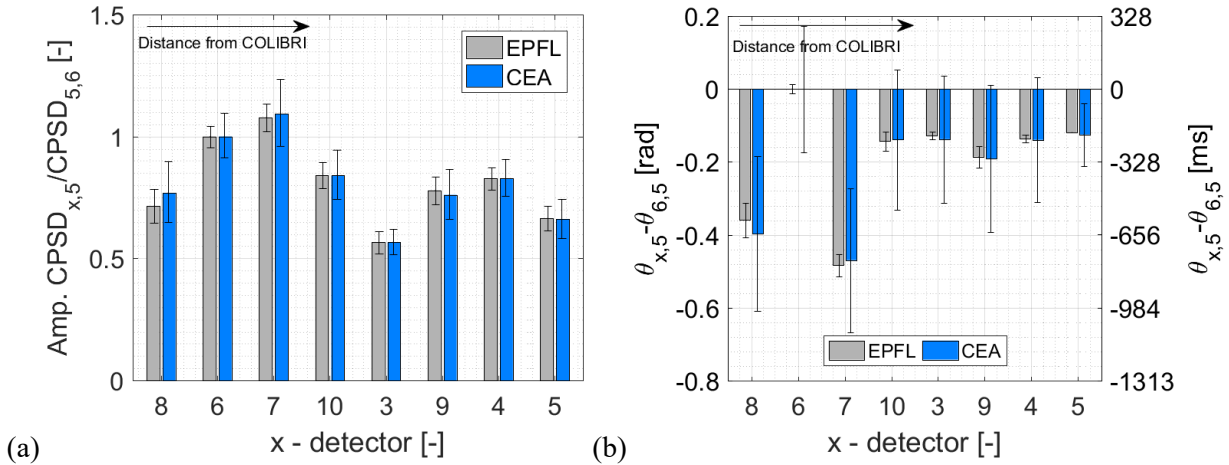
The CPSD uncertainties are quantified using the bootstrap random sampling with replacement method [7] [8] [9]. With this technique, each detector time signal is cut in segments with the length of a period identifying each segment with an index, see Figure 2a. Then a set of random number is generated and the segments, whose indexes corresponds to the previously generated random numbers, are used to build the

so called “resampled” signal on which the CPSD are computed, see Figure 2b. Here, the bootstrapping with replacement performs a total of 1000 resampling, compared to the 100k iterations performed in the reference analysis [6]. The bootstrap technique allows to increase the number of sampled available for the analysis and therefore to have an estimation of the average CPSD amplitudes and phases and their associated standard deviations. The reduction of the number of iterations allows to reduce significantly the computational resources needed to process the data, without affecting the values of the quantities of interest while maintaining an acceptable estimation of the uncertainties associated to the measurements [7]. Moreover, even if some impulsive peaks are present in the signals as these have not been smoothed, the bootstrap method and the following resampling allows to reduce the weight of the peaks in the final outcome.

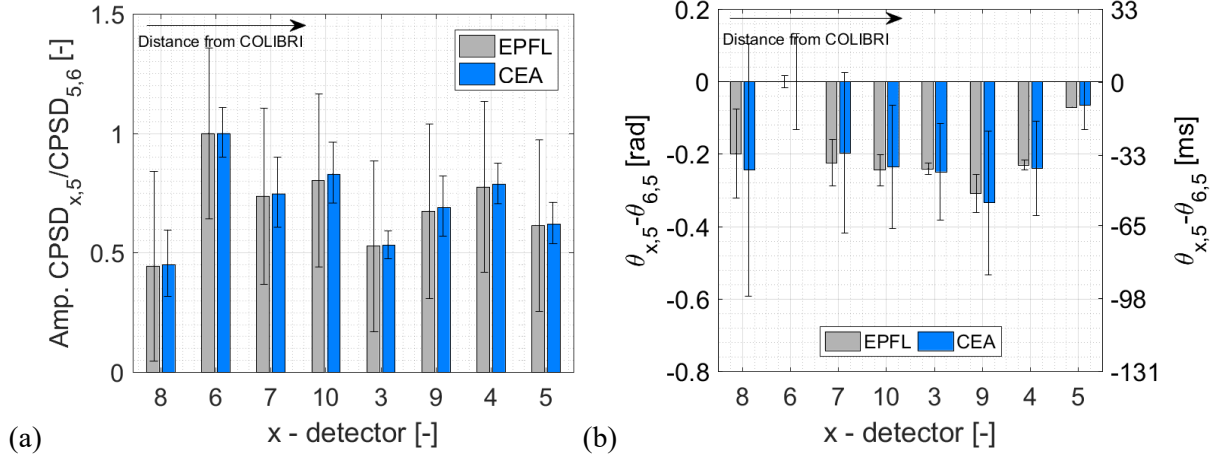


**Figure 2. Visual example (a) before and (b) after the use of the bootstrap method on an ideal sine curve.**

According to general CORTEX guidelines [3], the CPSDs amplitude and phases are normalized to pair 6&5, see Figure 3 and Figure 4, respectively, and the results show that concerning the outcomes of the two procedures give consistent results in for the average quantities proving the accuracy and reliability, while the quantified uncertainties are relatively different as a consequence of the different assumptions adopted (smoothing, normalization, number of bootstrap iterations, ...).



**Figure 3. Measured relative CPSD (a) amplitudes and (b) phases for experiment #12 as obtained from EPFL (grey bars) and CEA (blue bars). Error bars indicates the 95% confidence interval.**



**Figure 4. Measured relative CPSD (a) amplitudes and (b) phases for experiment #13 as obtained from EPFL (grey bars) and CEA (blue bars). Error bars indicates the 95% confidence interval.**

### 3. INTERPRETIVE AND PARAMETRIC STUDIES OF COLIBRI EXPERIENCES

#### 3.1. Simulation setup

The IPK model used for the simulation of neutron noise is developed starting from the flux and precursors concentration equations:

$$\left( \frac{1}{v} \partial_t + \Omega \cdot \nabla + \tilde{\Sigma}_t(\vec{r}, E, t) \right) \psi = H\psi + \frac{F_p}{k_{dyn}} \phi + \frac{F_d \vec{C}}{k_{dyn}} \quad (3)$$

$$\partial_t C_i(\vec{r}, t) = -\bar{\lambda}_i C_i(\vec{r}, t) + \beta_i \int_E dE' \nu \Sigma_f(\vec{r}, E', t) \phi(\vec{r}, E', t), \quad (4)$$

Where  $\vec{r}$  is the spatial coordinate,  $E$  is the energy group,  $\Omega$  is the direction angle,  $t$  the time,  $\tilde{\Sigma}_t(\vec{r}, E, t)$  is the total cross section accounting for the  $DB^2$  coefficient,  $\psi = \psi(\vec{r}, E, \Omega, t)$  is the angular flux,  $\phi = \phi(\vec{r}, E, t)$  is the scalar flux,  $k_{dyn}$  is the dynamic eigenvalue [10],  $\Sigma_f$  is the macroscopic fission cross section,  $\nu$  is the average number of neutrons produced by fission,  $\beta_i$  is the fraction of delayed neutron for family  $i$ . The prompt and delayed fission sources are identified by  $F_p$  and  $F_d \vec{C} = \sum_{i=1}^{N_d} \chi_{d,i} \lambda_i C_i$ , where  $C_i$  contains the convolution integral for the  $i$ -th precursor concentration whose decay constant is  $\lambda_i$ . Differently from the traditional point-kinetics approach [11] [12], the angular flux is factorized with the shape  $S(\vec{r}, E, \Omega, t)$  and power  $P(E, t)$  functions that preserve their energy dependence. By adopting a suitable normalization condition for all energy groups  $N_G$  and after some manipulations, the final form of the point-kinetics equation is:

$$\partial_t P + \frac{1}{v} \langle \partial_t S \rangle P + \frac{J^+ - J^-}{\langle S/v \rangle} P + \langle \tilde{\Sigma}_t S \rangle P - \langle HS \rangle P = \frac{1}{k_{dyn}} \langle F_p S \rangle P + \frac{1}{k_{dyn}} \langle F_d \vec{C} \rangle \quad (5)$$

where  $\langle \dots \rangle$  indicates the scalar product for space and angle. More details on the IPK model formulation and the leakage model used to estimate the  $DB^2$  coefficient can be found in [5] and [13], respectively.

### 3.2. Effects of the higher frequency on noise amplitude

The amplitude of neutron noise depends on the oscillations of the delayed neutron source  $F_d \vec{C}$ , since in the IPK model, this is the only term coupled in time by means of the quadrature formula used [10]. Therefore it is worth investigating the effects of various frequencies of oscillations  $\omega_c$  on this operator. Consider that the period of oscillation is discretized in  $M$  time steps and suppose to sit on one of such given time instants, say  $t_k$ : if one computes explicitly the weights to each time interval  $t_{k'}$ , for a generic precursor family  $i$  with decay constant  $\lambda_i$ , it turns out that with  $1/\omega_c \ll \lambda_i$ , the weights tend to be uniform among all the time steps  $t_{k'}$  with  $k' = 1, \dots, M$ . On the other side, if  $1/\omega_c \gg \lambda_i$ , the weights decrease exponentially as the difference  $t_k - t_{k'}$  increases for  $k = 1, \dots, M$  with  $k \neq k'$ . In fact, looking at the values of the decay constant of delayed neutron precursors, if  $\lambda_i$  is larger than the period of oscillations ( $\lambda_i \gg 1/\omega_c$ ), the precursor concentrations in the system cannot reach their equilibrium during a cycle, which means that they do not decay sufficiently to induce a significant variation of the delayed source. Considering the extreme value for the frequency, i.e.  $\omega_c = \infty$ , one can imagine that the oscillation degenerates on an “intermediate stationary” state with negligible fluctuations of the delayed source. On the opposite, if  $\lambda_i \ll 1/\omega_c$ , during the oscillation period, the concentration for the  $i$ -th family may reaches equilibrium, so it may decay significantly contributing to the fluctuations of the delayed source and therefore of the neutron flux. According to [14] [15], when using the classical point-kinetics transfer function, if the frequency of oscillation increases by one order of magnitude, the amplitude of the transfer function  $G(j\omega)$ , is reduced by a factor (almost) two, see Figure 5. In addition, it is clear that the largest amplitude reduction occurs when passing from 0.1 Hz to 2.0 Hz, while beyond this threshold the reactor transfer function starts to flatten. The IPK model captures the expected behavior of the transfer function. In fact, once a COLIBRI displacement is fixed and the simulations are performed considering various  $\omega_c$ , the amplitudes of  $CPSD_{x,5}$  decrease as the  $\omega_c$  increases up to an asymptotic value that corresponds to the frequency values where the transfer function starts to flatten. The computed results show that the amplitude decreases in the frequency range from 0.1 Hz to 2.0 Hz, according to what is expected from the theory, the CPSDs amplitudes are reduced by a factor  $\sim 1.76$  (on average), see Figure 6, while beyond the 2.0 Hz threshold, the reactor dynamics is less sensitive to the increase of  $\omega_c$ .

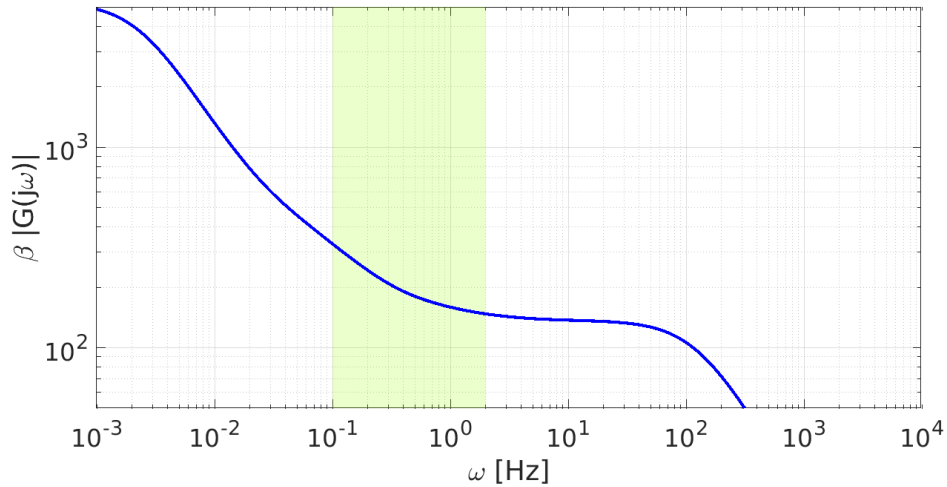


Figure 5. Magnitude of critical CROCUS reactor transfer function, when using classical point-kinetics with parameters from [16]. The frequency range of interest (0.1 Hz – 2.0 Hz) is highlighted in green.

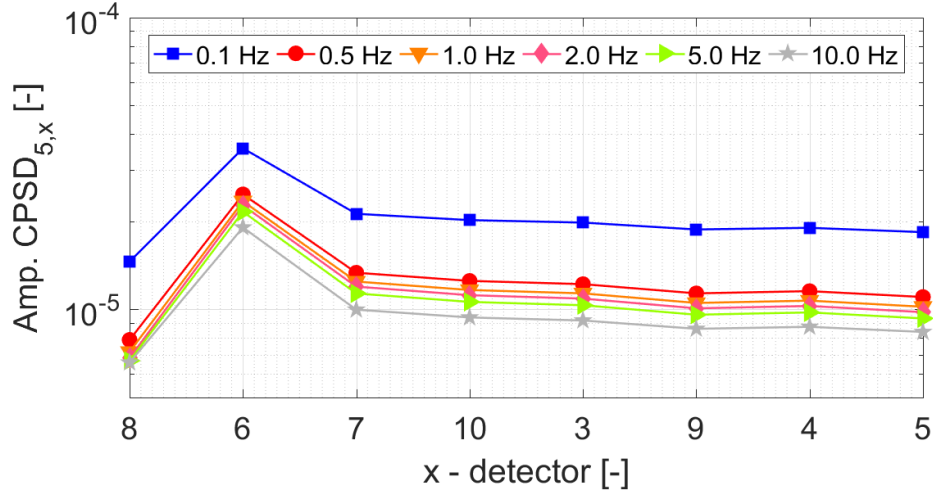


Figure 6. With a fixed oscillation amplitude of 1.85 mm,  $CPSD_{x,5}$  for a parametric study with various  $\omega_c$ .

### 3.3. Analysis and reconstruction of COLIBRI movement

In the domain of numerical simulations, the quality of the output results highly depends on the quality of the input parameters and in this case it is the oscillation amplitude. The fuel pins oscillate thanks to a crankshaft mechanism that moves the top COLIBRI plate, connected to the bottom one by an aluminum rod. Since the two plates are not rigidly jointed in the translational movement, some de-synchronization due to inertia effects may arise between the two. Moreover, due to the mechanical limitations of the machine, the imposed movement is not an ideal sine curve, see Figure 7, but it shows a flattened region on the top and bottom boundaries. The spectral analysis of this “almost-sinusoidal” movement, see Figure 8, shows higher frequencies, which can be either natural harmonics ( $\omega = n\omega_c, n \in \mathbb{N}^+$ ) or not, give indeed a contribution to the global movement. Therefore, it turns out that a simulation with an ideal sine curve may be a rude approximation causing some discrepancies.

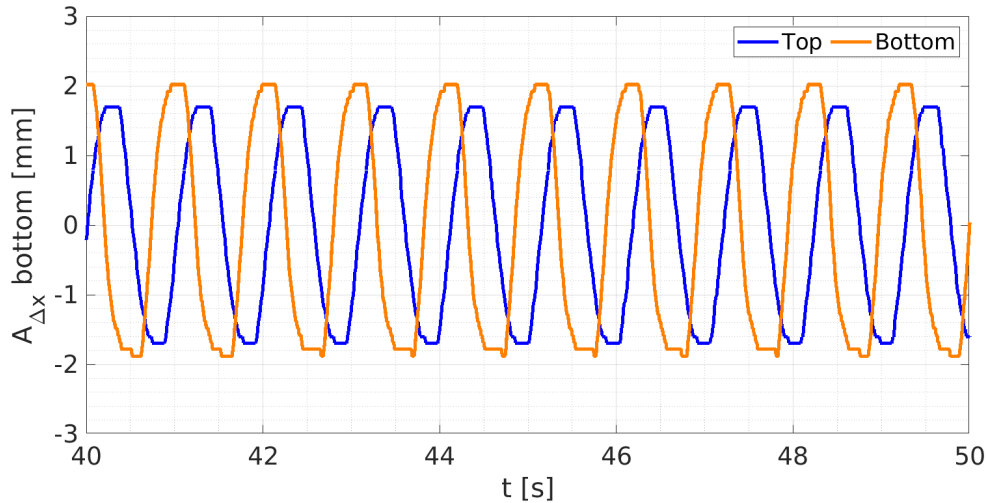
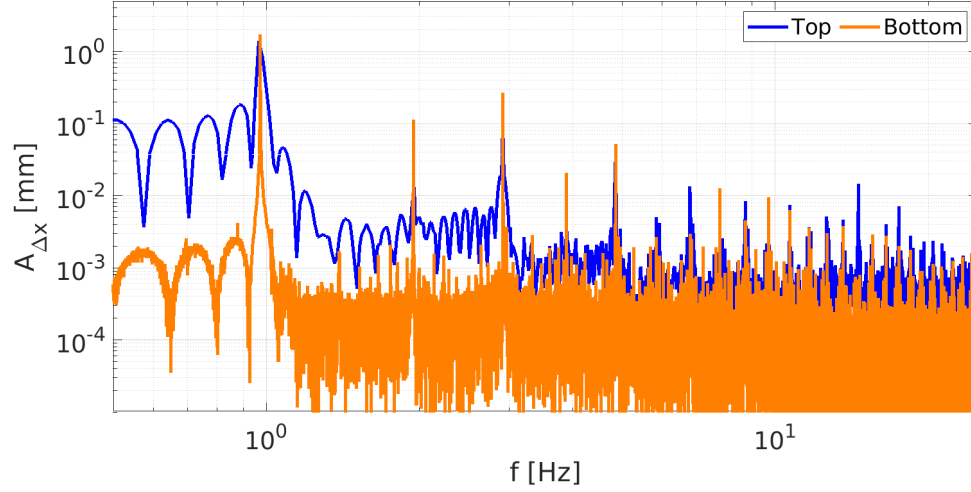


Figure 7. Time segments of raw top (blue) and bottom (orange) COLIBRI plates signals for experiment #13 ( $A = 2.0$  mm,  $\omega_c = 0.972$  Hz).

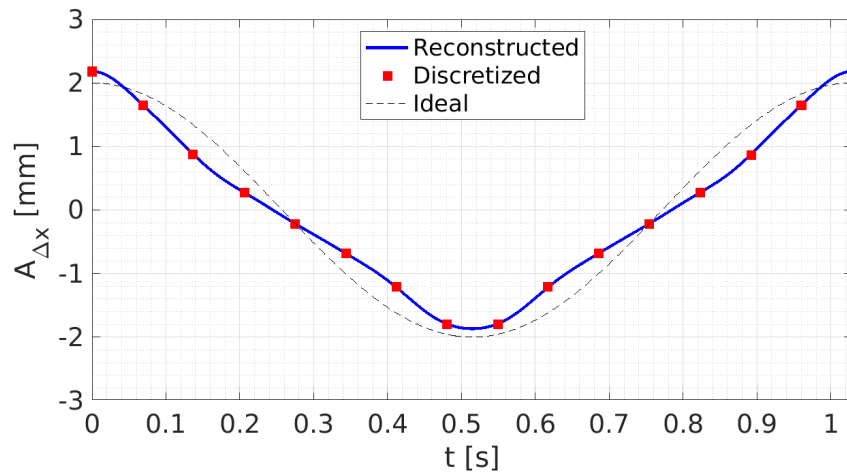


**Figure 8.** Frequency spectrum of top (blue) and bottom (orange) COLIBRI plates signals for experiment #13 ( $A = 2.0$  mm,  $\omega_c = 0.972$  Hz).

To improve the representativeness of the imposed shift, rather than the ideal/monochromatic approximation used so far, a reconstructed/polychromatic signal has been build. For conservative reasons [17], the bottom plate signal is chosen as reference and so the new displacement has been obtained and by superimposing the first  $N_k = 10$  harmonics of the signal Fourier spectrum as:

$$A_{\Delta x}(t) = \sum_{k=1}^{N_k} A_k \cdot \cos(2\pi k \omega_c) \quad (6)$$

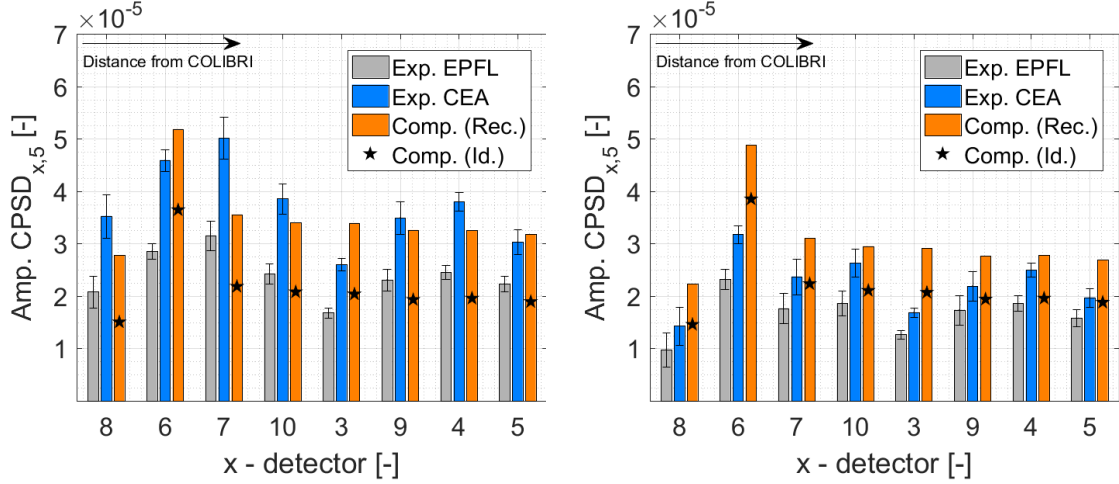
where  $A_{\Delta x}(t)$  is reconstructed shift of the pins and  $A_k$  is the amplitude corresponding to the  $k$ -th harmonic. Then as done for the simulation using the monochromatic displacement, a set of  $N$  discrete time equally spaced points are identified on the reconstructed signal, see Figure 9, and these are used as reference to compute the static flux distributions necessary to the IPK model.



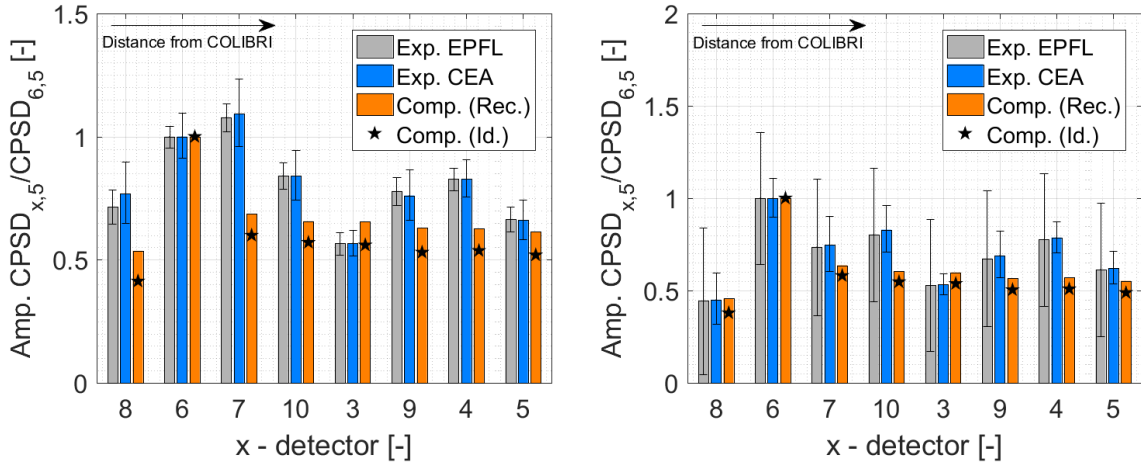
**Figure 9.** Bottom plate reconstructed (blue solid line) and monochromatic (dashed black line) signals for experiment #13. Equally spaced discrete points are also reported (red squares).



Using the reconstructed COLIBRI signal, the computed results show a good agreement for the computed results of the (to pair 6&5) CPSDs amplitudes and phases for experiment #12 and #13, see Figure 10- Figure 12. From the experimental point of view, the outcomes of EPFL and CEA analyses give consistent results. On the computational side, instead, both simulations with the ideal mechanical displacement, return slightly lower signal amplitude, possibly due to the sensitivity of the point-kinetics transfer function with  $\omega_c < 2\text{Hz}$  [15], see also Paragraph 3.2. The relative phase is within the error bars for experiment #12, with except of detector #7 showing an unexpected behavior currently under investigation. For experiment #13 with the higher frequency of oscillation, the relative phase measures few millisecond and the computed results fall mostly inside the error bars or they are outside the uncertainty intervals of few milliseconds.

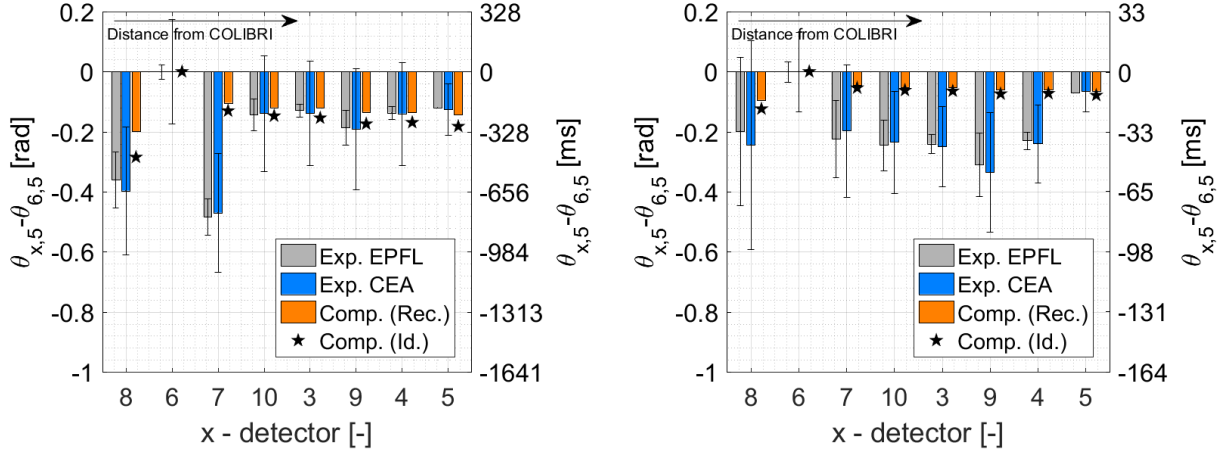


**Figure 10. Comparison of measures and computed  $\text{CPSD}_{x,5}$  amplitudes for experiment (a) #12 and (b) #13. Results obtained with reconstructed and ideal [5] signal are reported as orange bars and black stars, respectively.**



**Figure 11. Comparison of measures and computed relative (to pair 6&5)  $\text{CPSD}_{x,5}$  amplitudes for experiment (a) #12 and (b) #13. Results obtained with reconstructed and ideal [5] signal are reported as orange bars and black stars, respectively.**





**Figure 12. Comparison of measures and computed relative (to pair 6&5) CPSD<sub>x,5</sub> phases for experiment (a) #12 and (b) #13. Results obtained with reconstructed and ideal [5] signal are reported as orange bars and black stars, respectively.**

Concerning the simulation with the reconstructed COLIBRI signal, in principle, this should be a better representation of the actual shift. The absolute CPSDs show higher detector responses with respect to the case with the monochromatic displacement [5] and the reason behind this difference may lay behind the fact that the shapes used in the IPK model are those obtained from the static stimulations and that, differently from the traditional approaches [12], these are not updated at the end of the iterations using as correction algorithm of the kind presented in [10]. Concerning the phases, no major differences are observed when using the monochromatic or reconstructed COLIBRI signal, as the shape update would rather modify its normalization, therefore the overall level of the signal, rather than the spatial distribution.

#### 4. CONCLUSIONS

In the present work, the latest advancements in CEA analysis of the first CROCUS noise experimental campaign have been presented. The first part presents the methodology set up for the analysis of experimental data that gives consistent results with respect to those obtained in previous works. The second part presents some parametric and interpretive studies performed with the validated Improved Point-Kinetics noise model developed by CEA. The first include the analysis of the effects of the frequency of oscillation on the noise amplitude, showing that as expected from literature, with increasing frequencies, the amplitude of the reactor transfer function decreases. This should mean that the higher the frequency of oscillation, the more difficult it would be to detect the flux perturbation. On the other hand, the exercises using a reconstructed COLIBRI displacement signal show that higher order harmonics give a not negligible to the final amplitude of oscillation and therefore these stress the importance carefully determining the simulation input parameters. In conclusion, the present paper proved the accuracy of both CEA experimental and computational analysis tools in view of their future applications to industrial cases.

#### ACKNOWLEDGMENTS

The CORTEX project received funding from the Euratom Research and Training Programme 2014-2018 under grant agreement No 754316. The CEA team would like to thank the EPFL team for providing their experimental results and for the fruitful discussions during all project activities.

#### REFERENCES

- [1] V. Lamirand, A. Rais, S. Hübner, C. Lange, J. Pohlus, U. Paquee, C. Pohl, O. Pakari, P. Frajtag, D. Godat, M. Hursin, A. Laureau, G. Perret, C. Fiorina and A. Pautz, "Neutron noise experiments in the AKR-2 and CROCUS reactors for the European project CORTEX," *ANIMMA 2019, EPJ Web of Conferences 225, 04023 (2020)*, 2019, <https://doi.org/10.1051/epjconf/202022504023>.
- [2] V. Lamirand, P. Frajtag, D. Godat, M. Hursin, G. Perret, O. Pakari, A. Rais, C. Fiorina and A. Pautz, "The COLIBRI programme in CROCUS : characterisation of the fuel rods oscillator," *ANIMMA 2019, EPJ Web of Conferences 225, 04020 (2020)*, 2019, <https://doi.org/10.1051/epjconf/202022504020>.
- [3] V. Lamirand, M. Hursin, A. Rais, S. Hubner, C. Lange, J. Pohlus, U. Paquee, C. Pohl, O. Pkari and A. Laureau, "Experimental Report of the 1st campaign at AKR-2 and CROCUS," CORTEX - D2.1, 03 Dec. 2018.
- [4] A. Brighenti, S. Santandrea, I. Zmijarevic and Z. Stankovski, "Interpretation of COLIBRI measurements in the CROCUS research reactor using a point-kinetics reactor model," *submitted to the PHYTRA5 conference*, 2020.
- [5] A. Brighenti, S. Santandrea, I. Zmijarevic and Z. Stankovski, "Validation of a time-dependent deterministic model for neutron noise on the first CROCUS experimental measurements," *submitted to ANS M&C 2021 Conference*, 2020.
- [6] V. Lamirand, A. Rais, O. Pakari, M. Hursin, A. Laureau, J. Pohlus, U. Paquee, C. Pohl, S. Hubner, C. Lange, P. Frajtag, D. Godat, G. Perret, C. Fiorina and A. Pautz, "Analysis of the first COLIBRI neutron noise campaign in the CROCUS reactor for the European project CORTEX," in *PHYSOR2020: Transition to a Scalable Nuclear Future*, Cambridge, United Kingdom, March 29th-April 2nd, 2020.
- [7] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*, Boca Raton (FL): Chapman & Hall/CRC, 1993.
- [8] D. Radulović, "On the Bootstrap and Empirical Processes for Dependent Sequences," Boston (MA), Birkhäuser Boston, 2002, p. 345–364.
- [9] E. Paparoditis, "Frequency Domain Bootstrap for Time Series," Boston (MA), Birkhäuser Boston, 2002, p. 365–381.
- [10] A. Gammicchia, S. Santandrea, I. Zmijarevic, R. Sanchez, Z. Stankovski, S. Dulla and P. Mosca, "A MOC-based neutron kinetics model for noise analysis," *Ann. Nucl. Energy*, vol. 137, 2019.
- [11] A. F. Henry, *Nuclear-Reactor Analysis*, Cambridge (MA): The MIT Press, 1986.
- [12] S. Dulla, E. H. Mund and P. Ravetto, "The quasi-static method revisited," *Prog. Nucl. Energ.*, vol. 50, no. 8, pp. 908-920, 2008.
- [13] S. Santandrea, L. Graziano, I. Zmijarevic and B. Vezzoni, "A Leakage synthetic algorithm and a Krylov approach for thermal iterations in APOLLO3 code in support to industrial applications," *submitted to ANS M&C 2021 Conference*.
- [14] G. R. Keepin, *Physics of Nuclear Kinetics*, Addison-Wesley Publishing Co., 1965.
- [15] D. L. Hetrick, *Dynamics of Nuclear Reactors*, The University of Chicago Press, 1971.
- [16] OECD/NEA-4440, "Benchmark on the kinetics parameters of the CROCUS reactor," *Plutonium recycling*, vol. 9, 2007.
- [17] V. Lamirand, *Private communications*, 06 Oct. 2020.

# Analysis and interpretation of the first CROCUS reactor neutron noise experiments using an improved point-kinetics model

## Appendix A MATLAB scripts for experimental data processing

### A.1 Load of experimental data

```
function [rawe,dt] = loadExp(ntest,daq,DET_ID)
    disp(['Loading Exp ' num2str(ntest) ' data...'])
    dt_old = 0;
    nDet = length(DET_ID);
    for iDet = 1:nDet
        try
            sig0 = load(['herefosethepathofthefile']);
            fprintf('Loaded IstEC_%.d\r',DET_ID(iDet));

            dt = sig0(1);    sig0(1) = [];
            if dt_old ~= dt % Check if the detector have the same sampling time
                disp(['WARNING:> dt different for det: ' num2str(DET_ID(iDet))]);
            end
            dt_old = dt;
            rawe(:,DET_ID(iDet)) = sig0;
        end
    end
end
```

### A.2 Manipulation of experimental data

```
clear all
close all
clc
s
ntest = [2:20]; % Number of the tests to be analyzed
daq = 'istec'; % Name of the DAS used

ARRAY_DET = 3:10; % Array of selected detectors for the analysis
nDet = max(ARRAY_DET); % Number of detectors
DET_ORDER = [8 6 7 10 3 9 4 5]; % Order of detectors with increasing distance from COLIBRI

refDet = 5; % Reference detector for CPSD signal
jDetRef = 6;
nBootstrap = 1000; % Number of bootstrap resampling

lbl = ''; % Label for the file output name

% 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
freq_nom = [0.0 1.0 1.0 0.5 1.5 2.0 0.1 0.5 1.0 0.1 0.5 0.1 1.0 0.5 0.1 1.0 1.5 2.0 0.1 1.0];
omega_fact = [0.8 1.2]; % Frequency lower/upper multiplication factor for maximum search

loadExpAgain = 1; % Call loadExp function
disp('Loading static experimental results-----')

for nn = ntest
    clear rawn rawe
    if loadExpAgain == 1
        [rawe,dt] = loadExp(nn,daq,ARRAY_DET); % Load the data
        dataR(nn).o = rawe; % Store the raw data into a struct in order to be able to use them
    later
        Fs = 1/dt;
        if (nn == ntest(end))
            save('dataR.mat','dataR','Fs');
        end
    else
        if nn == ntest(1)
            load('dataR.mat');
        end
        rawe = dataR(nn).o;
    end

    % Normalize the signal
```

```

for iDet = ARRAY_DET
    rawn(:,iDet) = rawe(:,iDet) / mean(rawe(:,iDet));
end

% Initialize bootstrap variables
teta_b = zeros(nDet,nBootstrap); teta2_b = zeros(nDet,nBootstrap);
f_b = zeros(nDet,nBootstrap); f2_b = zeros(nDet,nBootstrap);
A_b = zeros(nDet,nBootstrap); A2_b = zeros(nDet,nBootstrap);
Y_b = zeros(nDet,nBootstrap); Y2_b = zeros(nDet,nBootstrap);
auto_b = zeros(nDet,nBootstrap); auto2_b = zeros(nDet,nBootstrap);
autoY_b = zeros(nDet,nBootstrap); autoY2_b = zeros(nDet,nBootstrap);

for bb = 1:nBootstrap
    clear sigr
    for iDet = ARRAY_DET
        if bb == 1
            freq_s = freq_nom(nn);
            period = 1/freq_s;
            sigr(:,iDet) = rawn(:,iDet);
        elseif bb > 1 % Resampling only if bootstrap is used
            freq_s = mean(mean(f_b(DET_ORDER,1:(bb-1))));
            period = 1/freq_s;
            [sigr(:,iDet),dt_new] = myBootstrap(iDet,rawn(:,iDet),period,dt);
            Fs = 1/dt_new;
        end
    end
    % Reinitialize variables
    teta_ = zeros(nDet,1); f_ = zeros(nDet,1); A_ = zeros(nDet,1); Y_ = zeros(nDet,1);
    teta2_ = zeros(nDet,1); f2_ = zeros(nDet,1); A2_ = zeros(nDet,1); Y2_ = zeros(nDet,1);
    auto_ = zeros(nDet,1); autoY_ = zeros(nDet,1); auto2_ = zeros(nDet,1); autoY2_ =
zeros(nDet,1);
    for iDet = ARRAY_DET
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        fprintf('Test %d, Resampling %d/%d: doing APSD %d -
%d.\r',nn,bb,nBootstrap,iDet,iDet);
        [f,P1PA,~,~,~,Y2] = myCPSDlast(sigr(:,iDet),sigr(:,iDet),Fs); % Fourier Transform
        % Find the APSD peak in the interval around the nominal frequency and the second
harmonic
        % First harmonic
        ind1 = find(abs(f - omega_fact(1)*freq_s) == min(abs(f - omega_fact(1)*freq_s)));
        ind2 = find(abs(f - omega_fact(2)*freq_s) == min(abs(f - omega_fact(2)*freq_s)));
        P1_c = P1PA(ind1+1:(ind2-1));
        Y2_c = Y2(ind1+1:(ind2-1));
        ind_max = find(P1_c == max(P1_c));
        auto_(iDet) = P1_c(ind_max);
        autoY_(iDet) = Y2_c(ind_max);

        % Second harmonic
        ind1_2 = find(abs(f - omega_fact(1)*2*freq_nom(nn)) == min(abs(f -
omega_fact(1)*2*freq_nom(nn))));
        ind2_2 = find(abs(f - omega_fact(2)*2*freq_nom(nn)) == min(abs(f -
omega_fact(2)*2*freq_nom(nn))));
        P2_c = P1PA(ind1_2+1:(ind2_2-1));
        Y2_c = Y2(ind1_2+1:(ind2_2-1));
        ind_max_2 = find(P2_c == max(P2_c));
        auto2_(iDet) = P2_c(ind_max_2);
        autoY2_(iDet) = Y2_c(ind_max_2);

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        clear f P1PA ind_max ind_max_2
        fprintf('Test %d, Resampling %d/%d: doing CPSD %d -
%d.\r',nn,bb,nBootstrap,refDet,iDet);
        [f,P1PA,teta,YOUT,PA,~,Y2] = myCPSDlast(sigr(:,iDet),sigr(:,refDet),Fs);

        % First harmonic
        ind1 = find(abs(f - omega_fact(1)*freq_nom(nn)) == min(abs(f -
omega_fact(1)*freq_nom(nn))));
        ind2 = find(abs(f - omega_fact(2)*freq_nom(nn)) == min(abs(f -
omega_fact(2)*freq_nom(nn))));
        f_c = f(ind1+1:(ind2-1));
        P1_c = P1PA(ind1+1:(ind2-1));

```

## Analysis and interpretation of the first CROCUS reactor neutron noise experiments using an improved point-kinetics model

```

YOUT_c = YOUT((ind1+1):(ind2-1));
teta_c = teta((ind1+1):(ind2-1));
ind_max = find(P1_c == max(P1_c));
if isempty(ind_max) == 0
    teta(iDet) = teta_c(ind_max);
    f_(iDet) = f_c(ind_max);
    A_(iDet) = P1_c(ind_max);
    Y_(iDet) = YOUT_c(ind_max);
else
    teta(nn,iDet) = 0;
    f_(iDet) = 0;
    A_(iDet) = 0;
end

% Second harmonic
ind1_2 = find(abs(f - omega_fact(1)*2*freq_nom(nn)) == min(abs(f -
omega_fact(1)*2*freq_nom(nn))));
ind2_2 = find(abs(f - omega_fact(2)*2*freq_nom(nn)) == min(abs(f -
omega_fact(2)*2*freq_nom(nn))));
f2_c = f((ind1_2+1):(ind2_2-1));
P2_c = P1PA((ind1_2+1):(ind2_2-1));
YOUT2_c = YOUT((ind1_2+1):(ind2_2-1));
teta_c2 = teta((ind1_2+1):(ind2_2-1));
ind_max_2 = find(P2_c == max(P2_c));
if isempty(ind_max_2) == 0
    teta2(iDet) = teta_c2(ind_max_2);
    f2_(iDet) = f2_c(ind_max_2);
    A2_(iDet) = P2_c(ind_max_2);
    Y2_(iDet) = YOUT2_c(ind_max_2);
else
    teta2(iDet) = 0;
    f2_(iDet) = 0;
    A2_(iDet) = 0;
end
end % iDet
teta_b(:,bb) = teta; teta2_b(:,bb) = teta2;
f_b(:,bb) = f_; f2_b(:,bb) = f2_;
A_b(:,bb) = A_; A2_b(:,bb) = A2_;
Y_b(:,bb) = Y_; Y2_b(:,bb) = Y2_;
auto_b(:,bb) = auto_;
autoY_b(:,bb) = autoY_;
auto2_b(:,bb) = auto2_;
autoY2_b(:,bb) = autoY2_;
end % bb

for iDet = ARRAY_DET
    % Auto-Power Spectral Density
    CPSD(1).auto(nn,iDet) = mean(auto_b(iDet,:)); CPSD(1).autostd(nn,iDet) =
std(auto_b(iDet,:));
    CPSD(1).autoY(nn,iDet) = mean(autoY_b(iDet,:)); CPSD(1).autoYstd(nn,iDet) =
std(autoY_b(iDet,:));
    CPSD(2).auto(nn,iDet) = mean(auto2_b(iDet,:)); CPSD(2).autostd(nn,iDet) =
std(auto2_b(iDet,:));
    CPSD(2).autoY(nn,iDet) = mean(autoY2_b(iDet,:)); CPSD(2).autoYstd(nn,iDet) =
std(autoY2_b(iDet,:));
    % First harmonic
    CPSD(1).t(nn,iDet) = mean(teta_b(iDet,:)); CPSD(1).tstd(nn,iDet) = std(teta_b(iDet,:));
    CPSD(1).f(nn,iDet) = mean(f_b(iDet,:)); CPSD(1).fstd(nn,iDet) = std(f_b(iDet,:));
    CPSD(1).A(nn,iDet) = mean(A_b(iDet,:)); CPSD(1).Astd(nn,iDet) = std(A_b(iDet,:));
    CPSD(1).Y(nn,iDet) = mean(Y_b(iDet,:)); CPSD(1).Ystd(nn,iDet) = std(Y_b(iDet,:));
    % Second harmonic
    CPSD(2).t(nn,iDet) = mean(teta2_b(iDet,:)); CPSD(2).tstd(nn,iDet) = std(teta2_b(iDet,:));
    CPSD(2).f(nn,iDet) = mean(f2_b(iDet,:)); CPSD(2).fstd(nn,iDet) = std(f2_b(iDet,:));
    CPSD(2).A(nn,iDet) = mean(A2_b(iDet,:)); CPSD(2).Astd(nn,iDet) = std(A2_b(iDet,:));
    CPSD(2).Y(nn,iDet) = mean(Y2_b(iDet,:)); CPSD(2).Ystd(nn,iDet) = std(Y2_b(iDet,:));

end
end % nn
save(['CPSD_' num2str(refDet) '_x' lbl '_b' num2str(nBootstrap) '.mat'],'CPSD');
disp('##### END #####')

```

## A.2 CPSD function

```
function [f,AMP_CPSD,PHS_CPSD,YOUT,P1,PA,Y_,Y2_conj_] = myCPSDlast(sig1,sig2,Fs)
% For sig1
Y = fft(sig1); % Fourier Transform
L = length(sig1);
P2 = abs(Y/L);
P1 = P2(1:ceil(L/2)+1);
P1(2:end-1) = 2*P1(2:end-1); % Extract amplitude
P1 = P1/P1(1);
Y_ = Y(1:ceil(L/2)+1)/L;
Y_(2:end-1) = 2 * Y_(2:end-1);
Y_ = Y_/P1(1);
Y = Y/P1(1);

% For sig2
Y2 = fft(sig2); % Fourier Transform
L2 = length(sig2);
Y2_conj = conj(Y2);
PB = abs(Y2_conj/L2);
PA = PB(1:ceil(L2/2)+1);
PA(2:end-1) = 2*PA(2:end-1); % Extract amplitude
PA = PA/PA(1);
Y2_conj_ = Y2_conj(1:ceil(L2/2)+1)/L2; % Similarly to what is done to P1
Y2_conj_(2:end-1) = 2 * Y2_conj_(2:end-1);
Y2_conj_ = Y2_conj_/PA(1);
Y2_conj = Y2_conj/PA(1);

% Prepare output files
f = [Fs*(0:ceil(L/2))/L]'; % Extract frequency
AMP_CPSD = P1.*PA; % Extract amplitude
PHS_CPSD = angle((Y.*Y2_conj)/L^2); % Extract phase
PHS_CPSD=PHS_CPSD((1:ceil(L/2)+1));

% Compute the product of two complex numbers by hand
aY = real(Y_); bY = imag(Y_);
cY2 = real(Y2_conj_); dY2 = imag(Y2_conj_);
p_real = (aY.*cY2 - bY.*dY2);
p_imag = (aY.*dY2 + bY.*cY2);

YOUT = p_real + 1i .* p_imag;
end
```

## A.3 Bootstrap resampling subroutine

```
function [sig_out, dt_out] = myBootstrap(iDet,sig_in,period,dt)
% The period is not a multiple of dt, so it may happen that the number
% of points per period (i.e. period/dt) is not an integer number.
% Therefore I have to interpolate sig_in over a time vector with a dt
% compatible with the period.
mm = length(sig_in); % Number of points in the signal
time_0 = dt * [0:(mm-1)]; % Initial time array

points_per_period = floor(period/dt); % Set number of points per period
period_1 = round(period * 1e4)/1e4; % Find an 'approximated' oscillation period
to remove some digits
dt_out = period_1/points_per_period; % Set the chosen sampling time
num_av_period = floor(time_0(end)/period_1); % Find number of available periods
time_end = num_av_period*period_1; % Find final time of interpolating time vector
time_1 = [0:dt_out:time_end]; % Build time array for interpolation
sig_ip = interp1(time_0,sig_in,time_1,'linear'); % Interpolate the signal on the time array
sig_ip = sig_ip'; % Transpose the interpolated vector into a
column vector

nOffset = 100; % Number of periods to be cut at the beginning
of the signal

tot_p = floor(length(sig_ip)/points_per_period); % Total available periods
```

## Analysis and interpretation of the first CROCUS reactor neutron noise experiments using an improved point-kinetics model

```
% Do not consider the first 'nOffset' periods
sig = sig_ip([(nOffset*points_per_period+1):(tot_p*points_per_period)]);
nn = length(sig); % Number of points in the signal
n_period_disp = floor(nn/points_per_period); % Number of available periods

indBoot = randi(n_period_disp,n_period_disp,1);

tmp = [];
for pp = 1:length(indBoot)
    bb = indBoot(pp);
    indPoints = ((bb-1)*points_per_period+1):(bb*points_per_period);
    tmp = [tmp; sig(indPoints)];
end
sig_out = tmp;
end
```