POLITECNICO DI TORINO

MASTER's Degree in ENERGY AND NUCLEAR ENGINEERING

MASTER's Degree Thesis

Benchmarking of the nuclear data uncertainty quantification capabilities of the SANDY code

Supervisors

Prof. Sandra DULLA

Co-supervisors

Dr. Luca FIORITO

Dr. Pablo ROMOJARO

Candidate

Luigi BARONE

November 2023

"Alla mia meravigliosa famiglia: Mamma, Papà e Lea"

"A te nonno Luigi, che saresti indubbiamente fiero di me"

"All'amorevole, instancabile e immenso Fò"

Abstract

Nuclear safety is one of the most popular concern about nuclear energy and, for this reason, knowing the safety-related parameters, at design stage, is fundamental to reduce substantially the risk. However, especially in the nuclear field, the uncertainty sources are very widespread and, as consequence, computing with accuracy important parameters results very difficult.
In this work, the effective multiplication factor (k_{eff}) uncertainty due to only nuclear data uncertainties is investigated. In order to perform it, different methods has been developed in the past, but the stochastic Monte Carlo method is chosen for this thesis.
The Monte Carlo method, for uncertainty propagation, has become lately attractive because of faster computation skill (and so better performance) of new computers, since it usually requires huge computational costs. Moreover, it can compute uncertainties that with traditional methods cannot be computed with accuracy due to their assumptions, approximations and simplifications.
To be applied, samples of nuclear data must be produced and the SANDY code has been developed to perform this task. SANDY can work easily with cross-sections data while, for the nubar and the energy distributions, there is not a clear method implemented. With the following thesis, the nubar is added to the already-existing methods of SANDY and, after it, the sampling is verified. Furthermore, the uncertainty quantification/propagation on the integral parameter k_{eff}, due to the nubar uncertainty and using the Monte Carlo method, is analysed for different benchmarks as well as the simultaneous effect of cross-sections and nubar nuclear data, comparing the results with already existing methods such as the sandwich rule and NDaST. Then, the treatment in SANDY of the energy distributions is started but not completed and verified.
In this way, the uncertainty quantification/propagation can be easily performed using SANDY, also for the nubar, and the results on integral responses, such as the k_{eff} or power density, can be useful for improving the nuclear facilities safety as well as helping the nuclear data evaluators to create increasingly reliable data.

Acknowledgements

I find it mandatory to thank people who are, directly or indirectly, linked to the achievement of this thesis and my Master's degree.

First and foremost, I must thank my promoter Prof. Sandra Dulla who, from the bachelor, shared her love for the nuclear energy in her lectures, helping me its understanding and discovering this passion other than bursting my knowledge. About passion for nuclear, I cannot not mention Prof. Piero Ravetto whose lectures really make anyone curious about the nuclear physics and the story behind it.

Many thanks to SCK CEN for allowing me conducting my Master thesis at its BR1 building and for giving me all the tools and supports needed to learn, grow and further develop my interest in the nuclear field. In particular, I would like to express my sincere gratitude to my mentor Dr. Luca Fiorito who immediately gave me the possibility to work on its SANDY, transmitting his passions and knowledge about many scientific fields as well as driving me to the right progress of my research. Moreover, I would also like to extend my thanks to my co-mentor Dr. Pablo Romojaro that helped me to solve many topic and non-topic related issues and, without its guidance and suggestions, I would have never finished my work on time.

Special thanks to my middle school Prof. Vittorio Marino and my high school Prof. Antonio Di Pasqua for the invaluable, tough but fair teaching and for their devotion to physics and mathematics which allowed me to uncover my aptitude to their subjects.

I would like to extend my sincere thanks to all my friends and colleagues with whom I shared my Erasmus and internship at SCK CEN in Belgium: there are no words about the contribution you gave to let me live and study happily and carelessly in Boeretang. Nevertheless, this endeavour would not have been possible without David, Mounir, Alex, Chino and Ines allegiance to convince me in choosing this thesis at SCK CEN, giving me practical advice other than almost forcing me

on it.

I would like to express my deepest gratitude to my family: to my mother Pina who transmitted to me her love for the learning and her devotion to school other than sharing with me my anxiety before exams and always pushing me in giving it all. To my father Rocco who always believes in me, never has doubts about my performances and who reassured me every time I had concerns with its strong temperament. To my sister Lea which, since the beginning of my university career, has been my model, my reference, my map to follow as well as for her experienced and professional advice.
Thanks even for your economic support.
I could not have undertaken this journey without all of you.

I would like to extend my sincere thanks to my grandmother Lea for her constant presence, always calling me wherever I am, genuinely asking how I am and, of course, if I have eaten because mental and physical health are the basis to study. Moreover, I want to thank my grandfather Luigi because, even if from far, he always protects, supports and drives me.

Also, thanks should go to my uncles, aunts, cousins and friends, with whose light but essential support, provided a non-negligible contribution to this goal. In particular, I would be remiss in not mentioning my friends in Turin for making the past 5 years much more enjoyable and keeping me sane throughout the whole process.

Absolutely last but not least, words cannot minimally express my gratitude to Alfonso Borrelli and his family, that can be called team. I could never explain his affection and attention towards me, the days spent together and the unconditional and precious help he provided during my first stay in Turin. I miss you, Fò.

Once again, thanks to everybody!

Luigi Barone

Table of Contents

List of Tables

List of Figures

Acronyms

$\beta_{\mathbf{eff}}$ effective delayed neutron fraction. 15, 34
$^{184}\mathbf{W}$ tungsten-184. xi, 87–89, 91
$^{233}\mathbf{U}$ uranium-233. 54
$^{235}\mathbf{U}$ uranium-235. ix, x, 29, 60, 61, 65–68, 70–72
$^{238}\mathbf{U}$ uranium-238. 34, 60
$^{239}\mathbf{Pu}$ plutonium-239. ix, x, 2, 4, 15, 36, 37, 50, 51, 55, 57–61, 78, 79, 81, 90
$^{240}\mathbf{Pu}$ plutonium-240. ix, 36, 37, 58–61

ACE A Compact ENDF. ix, 14, 16, 20, 21, 41–43, 63–65
ADS Accelerator Driven System. 5
ANL-W Argonne National Laboratory-West. 60
ANR Atlas of Neutron Resonances. 14
ASAP Adjoint Sensitivity Analysis Procedure. 34

CENDL Chinese Evaluated Nuclear Data Library. 13
CINDA Computer Index of Nuclear Reaction Data. 11, 12
CLT Central Limit Theory. 40, 66, 67, 81
CNDC China Nuclear Data Center. 13
CNDCN China Nuclear Data Coordination Network. 13
CSBEP Criticality Safety Benchmark Evaluation Project. 50
CSEWG Cross Section Evaluation Working Group. 13, 18
CTF Criticality Test Facility. ix, 56–60

DICE Database for the International handbook of evaluated Criticality safety benchmark Experiments. 49

ENDF Evaluated Nuclear Data File. ix, 15, 18–22, 27, 42–46, 64–67, 78, 81, 90
ENSDF Evaluated Nuclear Structure Data File. 12–15
EXFOR Experimental Nuclear Reaction Data Library. 12, 15

FRENDY FRom Evaluated Nuclear Data librarY to any application. 15, 33, 41
FSAP Forward Sensitivity Analysis Procedure. 34

GNDS Generalised Nuclear Database Structure. 20
GPT Generalised perturbation Theory. 34, 36

IAEA International Atomic Energy Agency. 12, 13
ICSBEP International Criticality Safety Benchmark Evaluation Project. 15, 48, 50, 51, 62

IDAT International reactor physics handbook Database and Analysis Tool. 49
IEA International Energy Agency. 1
IRPHeP The International Handbook of Evaluated Reactor Physics Benchmark Experiments. 48

JAEA Japan Atomic Energy Agency. 13
JEFF Joint Evaluated Fission and Fusion. ix, x, 12, 13, 18, 29, 66, 78
JENDL Japanese Evaluated Nuclear Data Library. 13
JNDC Japanese Nuclear Data Committee. 13

$\mathbf{k_{eff}}$ effective multiplication factor. ix–xi, 2–4, 32–34, 36, 37, 48, 49, 63, 73, 75, 78–87, 89–91
KS Kolmogorov–Smirnov. 66, 73, 75, 83, 84
kwargs keywords arguments. 65

LANL Los Alamos National Laboratory. 20, 61
LASL Los Alamos Scientific Laboratory. 51, 52, 61
LH Latin Hypercube. x, xi, 4, 76, 80, 83–85, 90
LLNL Lawrence Livermore National Laboratory. 18

MC Monte Carlo. 3, 4, 21, 22, 28, 36, 40, 41, 63, 73, 76, 79, 80, 82, 86, 87, 90, 91
MIRD Medical Internal Radiation Dose. 14
MSR Molten Salt Reactor. 5

NDaST Nuclear Data Sensitivity Tool. ix, xi, 16, 21, 40, 48–50, 86, 87, 89, 91
NDC Nuclear Data Center. 13
NDEC Nucelar Data Evaluation Cycle. 16
NDS Nuclear Data Section. 13
NEA Nuclear Energy Agency. 13, 48, 50, 51, 63
NRDC Nuclear Reaction Data Centres. 12
NRG Nuclear Research and consultancy Group. 13
NRT Norgett-Robinson-Torrens. 14
NSDD Nuclear Structure and Decay Data. 12
NSR Nuclear Science References. 11
nubar average fission neutron multiplicity. ix–xi, 2–4, 26, 29, 31, 32, 36, 37, 43, 47–49, 63, 65–68, 78–81, 85–87, 89–92, 94
NuDat National Nuclear Data Center. 13
NUDUNA NUclear Data UNcertainty Analysis. 42
NWC Nuclear Wallet Cards. 13
NZE Net Zero Emission by 2050 scenario. 1, 91

OECD Organisation for Economic Cooperation and Development. 12, 50, 51, 63

PCA Principal Component Analysis. 26
PFNS Prompt Fission Neutron Spectra. x, 2–4, 7, 26, 32, 47, 69–72, 90, 92
PSI Paul Scherrer Institute. 13
PU-MET-FAST Plutonium Metal Fast. ix–xi, 51–62, 80–83, 87, 89, 91

QA Quality Assurance. 14, 15

RIPL Reference Input Parameter Library. 14

RNDC Russian Nuclear Data Center. 13

RSICC Radiation Safety Information Computational Center. 63

SA Sensitivity Analysis. 17, 30, 32

SANDY Sampler of Nuclear Data uncertaintY. x, xi, 4, 14, 20, 26, 29, 40, 42–48, 50, 53, 63–65, 68–73, 78–81, 86, 87, 89–91

SCALE Standardized Computer Analyses for Licensing Evaluation. 36, 40, 48

SCK CEN Belgian Nuclear Research Centre. 42

SIMBAD Set of Identifications, Measurements, and Bibliography for Astronomical Data. 15

TENDL TALYS-based Evaluated Nuclear Data Library. 13, 42

UQ Uncertainty Quantification. ix, 3, 4, 17, 30, 31, 38–42, 48, 49, 73, 80, 81, 86–88, 90, 91

USDOE United States Department Of Energy. 50

VNIIEF All-Russian Scientific Research Institute of Experimental Physics. ix, 57–60

VNIITF All-Russian Scientific Research Institute of Technical Physics. 56

VTT Technical Research Centre of Finland. 63

XSUSA Cross Section Uncertainty and Sensitivity Analysis. 42

XUNDL Experimental Unevaluated Nuclear Data List. 11

ZPPR Zero Power Physics Reactor. 60

Introduction

Context

Nowadays, nuclear energy is candidate to play an important role to cope with two relevant worldwide crises: the climate and the energy crisis.

For what concern the energy crisis, after the Ukraine invasion by the Russian military forces and its consequent global energy supply cut, the governments have started to re-consider their energy security plans, focusing mainly on the development of domestic energy supplies as well as their diversification, and it is a common though nuclear energy is one of them [1]. Indeed, since the main purpose is to reduce the imported fossil fuels dependence, nuclear energy is a fair alternative, affordable for many countries.

About the climate crisis, Net Zero Emission by 2050 scenario (NZE) [2] is a normative scenario, made by International Energy Agency (IEA), aiming to reach a zero CO_2 net emission by 2050, following the given Roadmap, and reducing the global temperature increase of 1.5 °C. At the same time, the requirements of maintaining consistent and cost-effective energy provision, guaranteeing universal energy accessibility and fostering resilient economic expansion must be fulfilled. Therefore, it requires a complete transformation of the present energy systems which support the economy. According to this policy, the power mix will be mainly dominated by renewable sources, such as wind and solar energy, definitely eliminating the fossil fuels. However, renewables do not ensure important system services such as stability, flexibility and adequacy capability (to energy demand peak) and, for this reason, dispatchable energy sources are also needed [1] to secure the cited services and not only the electricity production. Nuclear energy is a large-scale low emissions energy source, capable to help the decarbonisation of the electricity supply as well as to produce heat and hydrogen, placing it as a valid dispatchable energy source.

Nuclear energy, operational in 32 countries and boasting a capacity of 413 gigawatts (GW), plays a dual role in advancing these objectives by preventing 1.5 gigatonnes (Gt) of worldwide emissions and reducing global gas demand by 180 billion cubic meters annually [1]. However, it is not accepted in all countries due to economic, safety, performance and waste management reasons. Indeed, the Fukushima-Daiichi power plant accident in 2011, further decreased public confidence on the nuclear energy, highlighting the crucial requirement for strong, autonomous regulatory supervision other than improving the safety design.

Nuclear safety for fission reactors and related data

Current nuclear reactors produce energy through the so-called **fission chain reaction** and so they are called fission nuclear reactors [3]. When a neutron interacts with a fissile nuclide, such as plutonium-239 (^{239}Pu), energy is released (based on mass defect) together with other two heavy nuclides and neutrons, whose number depends on the specific reaction [4]. The additional neutrons produced can lead to other fission reactions, causing a reactions cascade (chain reaction).
Nuclear safety is strictly related to the **criticality** concept, connected to the capability to control the chain reaction. In order to well understand it, the effective multiplication factor (k_{eff}) must be defined and it is the number of neutrons generated by fission divided by the number of neutrons lost by the system. Three cases are possible [5]:

- if $\mathbf{k_{eff} > 1}$ the average number of neutrons generated is higher than the one lost and the power increases exponentially. The system is called **supercritical**
- if $\mathbf{k_{eff} = 1}$ the average number of neutrons generated is equal to the one lost by the system and the power is constant. The system is called **critical**
- if $\mathbf{k_{eff} < 1}$ the average number of neutrons generated is lower than the one lost by the system and the power decreases exponentially. The system is called **subcritical**

Two of the major data affecting the k_{eff} are the average fission neutron multiplicity (nubar) and the Prompt Fission Neutron Spectra (PFNS) (part of the energy distributions):

* **nubar** is the average number of neutrons produced during the fission of a specific nuclide, for a specific neutron energy. It can be divided into:

  prompt nubar is the average number of prompt fission neutrons released per fission within 10^{-14} s of the fission event. In other words, the neutrons

generated in the same time fission occurs. The file number is MF=3 while the reaction one is MT=456.

delayed nubar is the average number of delayed neutrons released per fission. Their release occurs after the fission event itself, allowing the control of the current nuclear power plants. The file number is MF=3 while the reaction one is MT=455.

The sum of the two gives the **total nubar** o simply called **nubar**, represented by the file number MF=3 and reaction number MT=452.

* **PFNS** is the fraction of prompt neutrons emitted per unit energy (E, E+dE), for a specific fission reaction and for a specific incident neutron energy. It is a probability density function and, usually, the highest probability corresponds to a released neutron energy in the neighbourhood of 2 MeV (peak of the distribution). The file and reaction number are respectively MF=5 and MT=18.

While the nubar will be deeply analysed during this thesis, a small introduction to the PFNS will be also given.

The role of uncertainty quantification in nuclear reactor physics

Lately, Uncertainty Quantification (UQ)/propagation analysis has become largely adopted among engineering and science fields [6]. One of the main task is to determine how certain input data uncertainty propagates towards the response function uncertainty.

Different methods can be used, but in this dissertation the stochastic Monte Carlo (MC) method will be applied, that has grown thanks to modern computation performance of new computers.
In this framework, the focus is on the k_{eff} and, in particular, the uncertainty on the k_{eff} due to the nubar uncertainty. Indeed, the k_{eff} uncertainty determination is crucial for the reactor safety design. Moreover, offering accurate assessments of how nuclear data impacts the uncertainty associated with widely-recognised integral benchmarks is essential for the validation and verification procedures of a nuclear data library [7].

It is important to point out that the uncertainty on the k_{eff} does not depend only on the input nuclear data, but many other variables play a relevant role (e.g. geometry uncertainty, model assumptions and simplifications, design conditions,

etc.) and, thus, it is impossible to determine the complete uncertainty of any response function, included the k_{eff}.

The uncertainty science is a very complex field, impossible to study completely and precisely, because of its inherent significance, and therefore this dissertation will focus only on the input nuclear data as source of uncertainty.

"Einstein never accepted quantum mechanics because of this element of chance and uncertainty. He said: God does not play dice. It seems that Einstein was doubly wrong. The quantum effects of black holes suggests that not only does God play dice, He sometimes throws them where they cannot be seen"

Stephen Hawking

Thesis objectives

More specifically, during this thesis, the following goals will be reached:

- the nubar will be implemented in Sampler of Nuclear Data uncertaintY (SANDY) [8], that is a Python package developed in SCK CEN, aiming primarily to perturb nuclear data files.
- In addition, also a method for the energy distributions will be introduced, in particular the PFNS. However, this has not been completed and verified.
- Then, the code and methodology for the nubar must verified: specific benchmarks will be properly chosen and UQ/propagation on the k_{eff} will be performed, applying the MC stochastic method (the criticality calculations will be run through the Serpent2 code).
- Afterwards, the MC method results will be compared with other methods and the differences, if any, will be analysed. In particular, the main objective is to study the plutonium-239 (^{239}Pu) nubar uncertainty effect on some plutonium fast benchmarks k_{eff} uncertainty. Furthermore, also the UQ/propagation of all cross-sections and nubar (when fissile) of all the input nuclides on the selected benchmarks k_{eff} will be performed, and important considerations will be given.
- Lastly, the impact of a quasi Monte Carlo method, the Latin Hypercube (LH) sampling, will be investigated.

Chapter 1

Nuclear Data

Nuclear data describe the nuclear properties of atomic nuclei and their physical nuclear interactions. To perform simulations of nuclear applications, a complete collection of nuclear reactions and decay data are needed. Thus, nuclear data are the basic information to understand all the physical phenomena regarding all nuclear technologies. There are different categories of nuclear data applications that underline their importance [9](Figure 1.1):

Energy applications: this includes fission and fusion energy, nuclear fuel cycle, waste management and decommissioning, Accelerator Driven System (ADS). In particular, for nuclear fission reactors, nuclear data are necessary to evaluate and understand all the physical, engineering and nuclear processes (neutron/photon fluxes, reaction rates, activation and radionuclides inventory, radiation damages, etc.) as well as safety margins (shielding, criticality, reactivity coefficients, power and its distribution, burn-up and relative transmutation etc.). All these features will be crucial for the design of future reactors in order to ensure performance and mainly safety. Different forms of energy are included: heat, electricity and hydrogen production. To this category also nuclear submarines and shuttles belong. For example, the reaction cross-section ^{35}Cl(n,p) has a large impact on the criticality safety analysis for Molten Salt Reactor (MSR) in the energy range 0.1-10 MeV and, unfortunately, massive discrepancies are present between the evaluated data. As a consequence, it is crucial to delete or reduce this issue.

Non-energy applications: medicine (diagnosis and therapy), production of radioisotopes for industrial and medical applications (e.g. Technetium 99), dosimetry and radiation safety. Also, nuclear safety and material analysis (radiation-induced damages, ion beam analysis, detection of hazardous materials) as well as geological applications (exploration of particular nuclides, search for oil, studying of earth composition etc.)

Basic science applications: astrophysics, planetary gamma spectroscopy to find out the elements' abundance on a planet surface, planets and stars evolution and many others.

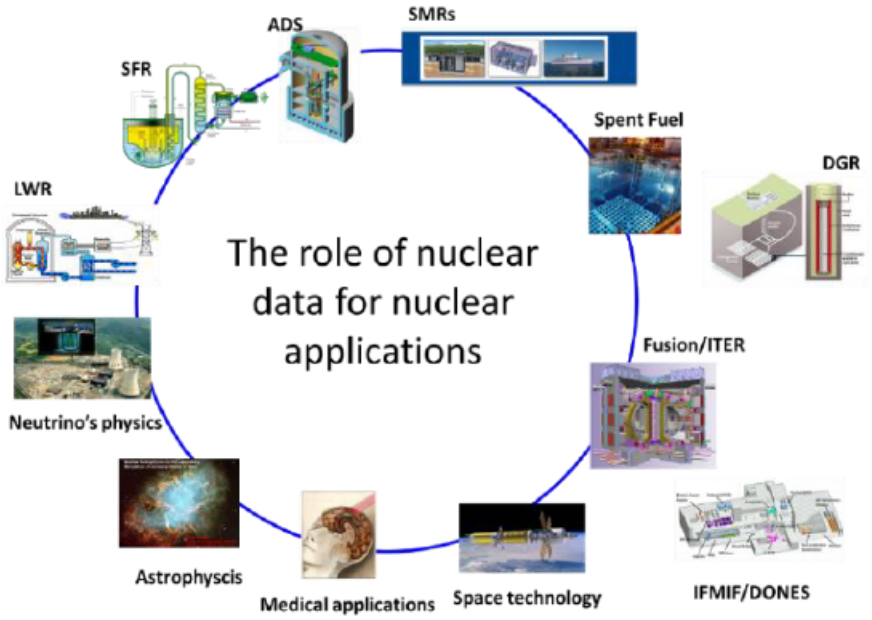


Figure 1.1: Nuclear applications playing important roles [9]

Nuclear data collection (Figure 1.2) must include all stable and radioactive isotopes present. The modelling nuclear systems are characterised by physical/mathematical equations and the nuclear data are fundamental to solve them:

The Boltzmann neutron transport equation: Considering t the time, $\mathbf{r} = (x, y, z)$ the position, E the energy and $\mathbf{\Omega}$ the direction of a neutron, the Equation 1.1 [10] is related to the motion and interactions of neutrons with matter, and simplified approaches can be used to solve it, such as the S_N method [11] and the diffusion equation [12]. Also, other particles transport equations can be derived.

$$\begin{aligned} &\frac{1}{v}\frac{\partial \Phi(\mathbf{r}, E, \mathbf{\Omega}, t)}{\partial t} + \nabla \cdot \mathbf{\Omega}\Phi(\mathbf{r}, E, \mathbf{\Omega}, t) + \Sigma(\mathbf{r}, E)\Phi(\mathbf{r}, E, \mathbf{\Omega}, t) = \\ &= \oint \mathrm{d}\Omega' \int \mathrm{d}E' \Sigma_s(\mathbf{r}, E')\Phi(\mathbf{r}, E', \mathbf{\Omega}', t) f_s(\mathbf{r}, E' \to E, \mathbf{\Omega}' \cdot \mathbf{\Omega}, t) + \\ &\qquad + S(\mathbf{r}, E, \mathbf{\Omega}, t) \end{aligned} \tag{1.1}$$

$\mathbf{r}, E, \mathbf{\Omega}$ is the phase space and:

- $\Phi(\mathbf{r}, E, \mathbf{\Omega}, t)$ is the angular particle flux in n/cm^2/s/eV/sr
- v is the particle speed in cm/s
- $\Sigma(\mathbf{r}, E)$ is the total macroscopic cross-section [cm^{-1}]. A macroscopic cross-section represents the probability of interaction, for a particle with the matter, per unit path [5]. For a neutron, two main events/interactions can occur: absorption and scattering. After an absorption, the interaction can lead to a radiative capture or to a fission event, if fissile material. Moreover, each interaction x has its macroscopic cross-section computed as $\Sigma_x(\mathbf{r}, E) = \sigma_x(E)N(\mathbf{r})$, where $\sigma_x(E)$ is the microscopic cross-section [cm^2] and $N(\mathbf{r})$ is the atomic density [cm^{-3}] of the matter. Since Σ is a probability per unit path, the probability must be conserved and, thus, the sum between the macroscopic radiative capture cross-section Σ_r and the macroscopic fission cross-section Σ_f gives the macroscopic absorption cross-section Σ_a. In principle, Σ also depends on the direction, but to simplify the explanation, the medium is assumed isotropic
- $\Sigma_s(\mathbf{r}, E) = N(\mathbf{r}) \cdot \sigma_s(E)$ is the scattering macroscopic cross-section
- $f_s(\mathbf{r}, E' \rightarrow E, \mathbf{\Omega}' \cdot \mathbf{\Omega}, t)$ is the *scattering probability density*, while E' and E are respectively the incident and the outgoing neutron energy of a scattering event. The same regard for the direction $\mathbf{\Omega}$. Particularly, for an isotropic medium, the probability does not depend on the incoming neutron energy but only on the angle formed by the two directions, i.e. the scalar product between them
- $S(\mathbf{r}, E, \mathbf{\Omega}, t)$ is the source term, defined as the sum between all the reactions leading to a neutron production and an external source S_{ext}. Two important reactions/events, producing neutrons, are the prompt (Equation 1.2) and the delayed fission (Equation 1.3) events [13].

✻ Prompt fission

$$\oint \mathrm{d}\Omega' \int \mathrm{d}E' \nu(\mathbf{r}, E') \Sigma_f(\mathbf{r}, E') \Phi(\mathbf{r}, E', \mathbf{\Omega}', t) \frac{\chi(\mathbf{r}, E)}{4\pi} \tag{1.2}$$

where:

$\nu_i(E')$ is the neutron multiplicity of prompt neutrons

$\chi(\mathbf{r}, E)$ is the Prompt Fission Neutron Spectra (PFNS), which represents the probability density function of a neutron to be reissued within the interval (E, E + dE). The term $\frac{1}{4\pi}$ is due to the isotropic nature of the fission.

✻ Delayed fission:

$$\sum_k \lambda_k \cdot C_k(\mathbf{r},t) \cdot \chi_{d,k}(\mathbf{r},E) \tag{1.3}$$

with C_k the concentration of the k$^{\text{th}}$ group (precursor) of delayed neutrons, $\chi_{d,k}(\mathbf{r},E)$ the delayed fission neutron spectra and:

$$\frac{\partial C_k}{\partial t} = -\lambda_k \cdot C_k(\mathbf{r},t) + \sum_i \oint \mathrm{d}\Omega' \int \mathrm{d}E' \Sigma_{f,i}(\mathbf{r},E')\Phi(\mathbf{r},E',\mathbf{\Omega}',t)\nu_{dk,i}(\mathbf{r},E') \tag{1.4}$$

where:

λ_k is the half-life of the k$^{\text{th}}$ group of delayed neutrons

$\nu_{dk,i}(\mathbf{r},E')$ is the neutron multiplicity of delayed neutrons for the reaction i

$\Sigma_{f,i}(\mathbf{r},E')$ is the i$^{\text{th}}$ reaction macroscopic fission cross-section.

Examples of derived quantities from the Boltzmann equation are (time dependency is omitted to simplify):

✻ Reaction rate

$$RR_{x,i}(\mathbf{r},E) = \oint \mathrm{d}\Omega' N_i(\mathbf{r}) \cdot \sigma_{x,i}(E)\Phi(\mathbf{r},E',\mathbf{\Omega}') = \Sigma_{x,i}(\mathbf{r},E)\Phi(\mathbf{r},E) \tag{1.5}$$

✻ Fission power

$$RR_{fission,i}(\mathbf{r},E) = \Sigma_{fission,i}(\mathbf{r},E) \cdot \Phi(\mathbf{r},E) \tag{1.6}$$

$$\int_V \mathrm{d}V \sum_i E_{i,fission} \int RR_{fission,i}(\mathbf{r},E)\,\mathrm{d}E = P[W] \tag{1.7}$$

with $E_{i,fission}$ the energy released in each specific fission reaction.

The Bateman equation: Equation 1.8 describes the isotopic inventory evolution [14]:

$$\boxed{\frac{\mathrm{d}N_i(t)}{\mathrm{d}t} = -(\lambda_i + r_i) \cdot N_i(t) + \sum_{i \neq j} (\lambda_{j\to i} + r_{j\to i}) \cdot N_j(t) + PF_i} \tag{1.8}$$

where:

λ_i is the half-life of the radionuclide i

$\lambda_{j\to i} = \lambda_j \cdot b_{j\to i}$, where b is the branching ratio

$r_{i\to j} = \int_V \int \Phi(\mathbf{r},E)\sigma_{i\to j}(E)\,\mathrm{d}E$ is the reaction term

$PF_i = \sum_h \int_V N_h(\mathbf{r}) \int \Phi(\mathbf{r}, E)\sigma_{f,h}(E)\gamma_{h\to i}(E)\,\mathrm{d}E$ is the fission source term (for the nuclides) and:

- the independent fission yields $\gamma_{h\to i}$ is the probability that a nuclide i is produced after a fission of the nuclide h due to the absorption of a neutron
- $\sigma_{f,h}$ is the microscopic fission cross-section of the nuclide h
- $\Phi(\mathbf{r}, E)$ is the direction-integrated neutron flux [n/cm^2/eV]

Examples of derived quantities from Equation 1.8 are:

✻ The activity

$$A_i(t) = \lambda_i \cdot N_i(t) \tag{1.9}$$

✻ The decay heat

$$DH_i(t) = \lambda_i \cdot N_i(t) \cdot E_i \tag{1.10}$$

and E_i is the average energy per decay that can come from electron-related radiation, electromagnetic radiation or heavy charged particles one.

Both equations use all nuclear data as parameters, and thus the accuracy of their solution depends on the accuracy of the nuclear data themselves. While for the Boltzmann equation the nuclear reaction data (cross-sections, angular and energy distributions, fission multiplicity etc.) are the most important, for the Bateman one nuclear structure and decay data play the key role (half-lives, decay and branching ratios and scheme, energy and intensity of emitted particles, independent and cumulative fission yields).

In this paper, nuclear data mainly related to Equation 1.1 will be investigated.

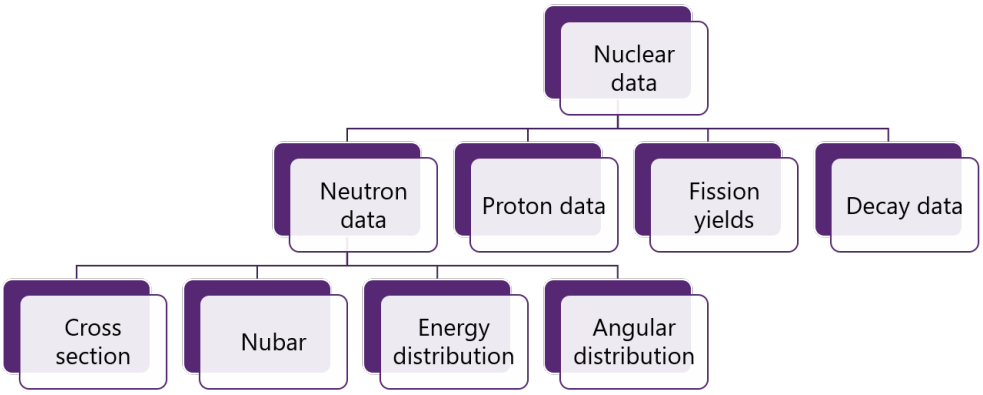


Figure 1.2: Nuclear data scheme

1.1 Activities associated with nuclear data

In the "nuclear data route" (Figure 1.3) the theoretical and experimental nuclear data are elaborated in order to make them suitable for the simulations of nuclear applications. All the steps of the route are complex, long and some feedbacks may be necessary to adjust them, leading to loops. The procedure can be summarised in 4 different tasks: **Compilation**, **Evaluation**, **Processing** and **Validation** [9].

1.1.1 Compilation of nuclear data and filling of databases

First of all, nuclear data production depends on two methods:

Experimental: here, nuclear data are measured through differential experiments in the laboratories, and then they are stored in different set of publications: lab report, journals, proceedings, etc.

Nuclear theory models: specific codes are developed based on nuclear theory models, and they are able to predict reaction cross-sections and other nuclear properties (e.g. fission neutron multiplicity).

Experiments are always preferred because the current accuracy of nuclear models is still very low, compared to the one of the experiments. However, experiments are not always possible because of different reasons:

- Impossibility to produce a very pure sample of the target nuclide.
- Elements with very low half-life so that it is impossible to perform the experiment before their decay.
- Rare nuclides that cannot be produced.
- Others.

Therefore, nuclear models are used to fill the gaps left by experiments or to interpolate or extrapolate nuclear data or to resolve discrepancies from different experiments.
Firstly, references and publications related to nuclear data are inserted in two compilations where no numerical data are present, and they are used as starting point in the evaluation process (subsection 1.1.2):

Nuclear Science References (NSR) is a set of references to publications on nuclear data (`https://www.nndc.bnl.gov/nsr/`). It contains structure, decay and non-neutron reaction data collected from 1910 to date [15].

Computer Index of Nuclear Reaction Data (CINDA) contains bibliographic references to measurements, reviews and evaluations of nuclear cross-sections (neutron and other incident particles), microscopic data and spontaneous fission data (`https://www-nds.iaea.org/exfor/cinda.htm`).

Subsequently, there are two important compilation databases that collect the real, numeric experimental results and other fundamental information, given by the authors, needed to properly evaluate nuclear data:

Experimental Unevaluated Nuclear Data List (XUNDL) represents the nuclear structure compilation database. Since data in XUNDL are organized by nuclide, a single publication may lead to the production of more than one XUNDL dataset. Also, each dataset is considered as a stand-alone work, and it is not mandatory to agree with existing, evaluated nuclear data of the nuclide it describes. Nevertheless, basic consistency checks are carried out during the XUNDL compilation and any internal conflicts are pointed out to the author in order to resolve them in the database. XUNDL is used both by nuclear structure evaluators as support and by nuclear structure researchers to find easily and promptly the current publications they need and, consequently, get access to data.

Experimental Nuclear Reaction Data Library (EXFOR) represents the nuclear reaction compilation database. It contains a comprehensive collection of experimental nuclear reaction data regarding neutrons, charged particles and photon incident particles. EXFOR includes all types of differential cross-sections, resonance parameters, polarization data, independent and cumulative fission yields and many other same-kind data [16]. Since the neutron discovery, neutron reactions have been compiled systematically, while less efforts have been put on charged particles and photons. All EXFOR data are connected to the CINDA bibliography. The EXFOR library includes both numerical and structured data with experimental and bibliographic information, and it is used in both nuclear reaction evaluation and research. The data have been systematically collected, expanded and developed by the Nuclear Reaction Data Centres (NRDC) through an international collaboration under the coordination of the International Atomic Energy Agency (IAEA) since 1970.

1.1.2 Evaluation

The evaluation activity is very specialised and is under the control of international and national nuclear data organisations [9]. Each data type is handled by different data file [17]:

- Structure and decay data evaluation is compiled in the Evaluated Nuclear Structure Data File (ENSDF). It contains numerous dataset types (e.g. prompt γ and particle spectra as well as decay data) for the most known nuclides (more info at: `https://www.nndc.bnl.gov/ensdf`). The international network of Nuclear Structure and Decay Data (NSDD) has evaluated and maintained the ENSDF data under the coordination of IAEA since 1974.

- Nuclear reaction and decay data evaluation is compiled in evaluated cross-section data libraries. In this procedure, data best-estimates and uncertainty are obtained through statistical/Bayesian procedures based on physical model calculations, using the EXFOR data as guide. The corresponding activities are mainly focused on neutron reaction data for the energy range 10^{-5} eV-20 MeV, even if, sometimes, energies up to 1 GeV are considered. Then, other incident particle reactions are taken into account: charged-particle and photon ones. Additionally, important nuclear data, other than reaction type, are considered: decay data, fission yields and neutron thermal scattering properties. The recently-released, most widely-used evaluated data libraries are [18]:

  Joint Evaluated Fission and Fusion (JEFF) data library belongs to the Organisation for Economic Co-operation and Development (OECD) and

it is produced by a coordinated group of the Nuclear Energy Agency (NEA) [19]. The latest version, the JEFF-3.3.1, was officially released in December 2019.

ENDF/B data library created by the Cross Section Evaluation Working Group (CSEWG), composed by a cooperation between national laboratories, industries and universities in the United States and Canada. The latest version is ENDF/B-VIII.0 and it was released in February 2018 in the USA.

Japanese Evaluated Nuclear Data Library (JENDL) generated by Nuclear Data Center (NDC) at the Japan Atomic Energy Agency (JAEA) together with Japanese Nuclear Data Committee (JNDC). The last version, JENDL-5.0, was released in December 2021.

TALYS-based Evaluated Nuclear Data Library (TENDL) is a bit different because it provides nuclear data based on the output of the TALYS [20] nuclear model code system. During the time, different groups were responsible for the production: up to 2014 the Nuclear Research and consultancy Group (NRG), then, since 2015, it has been developed by Paul Scherrer Institute (PSI) and by Nuclear Data Section (NDS) of the IAEA. TENDL-2021, the latest version, was released in December 2021.

Chinese Evaluated Nuclear Data Library (CENDL) produced by the China Nuclear Data Center (CNDC) together with the China Nuclear Data Coordination Network (CNDCN). The latest version, CENDL-3.2, was released in June 2020.

BROND is the Russian evaluated neutron data library and it is produced by the Russian Nuclear Data Center (RNDC). The latest version, BROND-3.1 was released in 2016.

There are other minor nuclear data libraries, while the most used ones in this framework are the JEFF and the ENDF/B. More details about the format will be explained in the section 1.2.

✻ Other important evaluations like:

- Nuclear Wallet Cards (NWC) that contains basic properties of ground and metastable states. `https://www.nndc.bnl.gov/nudat3/indx_sigma.jsp`
- Atomic Masses for the mass evaluation of more than 2900 nuclides. `https://www-nds.iaea.org/amdc/`.
- National Nuclear Data Center (NuDat) to search and plot nuclear structure and decay data from the ENSDF plus thermal neutron data. `https://www.nndc.bnl.gov/nudat3/`.

  - Atlas of Neutron Resonances (ANR) to collect neutron resonance parameters, thermal cross-sections, capture resonance integrals, average resonance parameters.
  - Reference Input Parameter Library (RIPL) that contains discrete and continuous nuclear structure data to be used in the nuclear model calculations `https://www-nds.iaea.org/RIPL-3/`.

1.1.3 Processing and benchmarking

During this phase, the evaluated data are prepared in a suitable form in order to be used in a particular application code. At this stage, a deep knowledge of the evaluation process physics is requested, as well as of the formatting issues of nuclear data and the code capabilities to manage them. First, the mandatory verification of the compliance with the format is performed (e.g. ENDF-6 or GNDS one). Subsequently, an exhaustive set of "quality" requirements must be verified to ensure they are fulfilled for all electronic files that contain nuclear data, through the Quality Assurance (QA) process [9]. In other words, the QA is a multi-stage process which aims to assess the validity of the physical information carried by an evaluated nuclear data file, comparing it with our best knowledge of available experimental data and theoretical models.

The processing, where application-specific databases are created and released, plays a fundamental role. Here examples of specialised data libraries:

- The Medical Internal Radiation Dose (MIRD) database is generated using the RADLST code, starting from the ENSDF evaluation. It is exploited to guide medical treatment and diagnosis, and it is made of nuclear and atomic radiations tables from nuclear decay schemes. Available at: `https://www.nndc.bnl.gov/nudat3/mird/`.

- The atomic displacement cross-sections using the standard Norgett-Robinson-Torrens (NRT) model or the gas production cross-sections can be generated by using the NJOY code [21] and consulting the evaluated nuclear data library. Available at: `https://www.oecd-nea.org/dbdata/jeff/jeff33/#dpa`. In particular, the NJOY code will be deepened in the section 1.3 as it is widely used in this thesis, launched through the SANDY code (section 2.3).

- For the continuous energy format, the A Compact ENDF (ACE) format is largely used, based on the evaluated nuclear data files. More important, it will be the format needed to run criticality simulations with the Serpent2 code [22] in order to produce the output dataset and carry out this thesis.

More in details, the QA processing, for the evaluated nuclear reaction data files, includes two type of complete set of tests [9]:

❶ For all **individual** evaluations:

ENSDF: the "ENSDF Analysis and Utility Programs" produces a collection of codes for checking, processing and analysis needed by the ENSDF users.

ENDF: there are tools to:

- Check format compliance by using utility codes such as CHECKR, INTER, etc [23].
- Process individual files with processing codes such as NJOY, AMPX [24], PREPRO [25], FRom Evaluated Nuclear Data librarY to any application (FRENDY) [26], etc.
- Compare differential data from EXFOR and other evaluations.
- Compute simple benchmarks cases on critical mass, shielding problems, transmutations, particle emissions, etc.

In this stage of individual evaluations check, feedbacks are provided to correct obsolete evaluated libraries (1^{st} pipeline loop in Figure 1.3).

❷ For the **complete** library that contains **all** individual evaluations. QA tests are performed to demonstrate the full library performance as compared with available internationally-evaluated integral benchmarks experiments:

– Integral tests, often referred to as benchmarks, represent straightforward single-physics experiments involving complex yet well-defined nuclear systems. They represent an essential reference and databases are produced for validation, such as International Criticality Safety Benchmark Evaluation Project (ICSBEP) [27]. In this context, an "integral benchmark" can be viewed as a meticulously assessed experiment and it undergoes a rigorous evaluation of experimental uncertainties, a process overseen by criticality experts, where the actual impact of nuclear data on the output (i.e. the integral experiment) will be assessed. In this framework, many of the ICSBEP benchmarks of plutonium-239 (^{239}Pu) will be considered and taken as reference for the method assessment.

– Traditional "benchmark suites" not only prioritise k_{eff} values, but also other measurements, including reaction rates and kinetic parameters, such as reactivity ρ, effective delayed neutron fraction (β_{eff}). An example of a database, containing this type of measurements, is Set of Identifications, Measurements, and Bibliography for Astronomical Data (SIMBAD) [28].

- Sensitivity and uncertainty analysis (SA/UA) serves a crucial function in establishing the link between nuclear data and integral parameters. Tools like Nuclear Data Sensitivity Tool (NDaST) [29] streamlines this analysis by employing a straightforward first-order approximation method. In the chapter 2, a more detailed description will follow since it is a fundamental pillar for this thesis.
- All the experiments performed here must not be used in the validation step.

Analysing the differences between predictions and integral experiments, basic nuclear data can be corrected to develop evaluated libraries of high "quality".

It is essential to ensure the accurate processing and reconstruction of the information contained in the evaluation to match the application conditions, such as energy grouping and operating temperatures. As a result, when a new nuclear reaction data library is released, it often includes a processed library in a specific format, such as the ACE format for the Serpent2 tool. Typically, these processes are carried out using software codes, making it feasible to automate them. This automation greatly simplifies the generation for the evaluators and ensures the correct utilisation for users. Different projects are working on it such as ADVANCE, MyENDF and Nucelar Data Evaluation Cycle (NDEC).

1.1.4 Validation

Finally, the nuclear data library evaluation must align with integral experiments that accurately represent the application to simulate [30]. In the realm of nuclear physics, "integral" data, such as k_{eff}, reaction rates, neutron leakage, and more, refer to data collected from complete systems. Therefore, nuclear reactors demand comprehensive physics coverage for all reactions, energy ranges and isotopes. Regarding nuclear data, the **validation** process can be subdivided in 4 broader subsets:

Verification: this step involves assessing the correct incorporation of nuclear data into evaluated files, ensuring the proper processing of nuclear data according to the evaluators' expectations and verifying solutions using integral benchmark suites which enable the quantification of the overall performance of a nuclear data library. There are various integral benchmarks for testing nuclear data, including those for reaction data (e.g. criticality, reaction rates and fission spectrum transmission benchmarks) and for fission yields and decay data (e.g. decay heat benchmarks). After the nuclear data library has been shown to accurately replicate these benchmarks, it can be committed for the user-applications.

Validation: this step concerns to more intricate experiments or integral measurements. It involves assessing the accuracy of a computational model by comparing it with experimental data that is assumed to faithfully represent reality. Practically, evaluated nuclear data is used in various user codes (e.g. Serpent [22], OpenMC [31], Mcnp [32], etc.) and user-defined problems to confirm that the results from the data correctly fit with real-world applications.

Sensitivity Analysis (SA): SA helps to identify how a nuclear data input influence the integral response and the trends, typically expressed in target system parameters (e.g. k_{eff}, reactivity coefficients, power distribution, nuclides concentration) [33]. This SA involves the calculation of sensitivity profiles using various techniques that will be deepened later.

Uncertainty Quantification (UQ): it aims to characterise all relevant uncertainties and to quantify their impact on integral data calculations [6]. Evaluating uncertainties in a "real" system presents a challenging aspect of UQ. In this particular scenario, the exclusive attention will be directed towards nuclear data as the predominant source of uncertainty. This phase of UQ is directly linked to end-user "applications," and it is imperative that nuclear data attain target accuracy demands, typically expressed through integral parameters essential for reactor design.

The SA&UQ are key aspects of this thesis and more details will be shown in the section 2.1 and section 2.2. Indeed, they are the two steps mainly treated in the present work.

The validation procedure can uncover other potential requirements for extra differential measurements or assessments, performing the loop until the desired level of precision is obtained. This represents the 3rd and last loop in the process. To sum up, the activity pipeline mainly focuses on providing better nuclear data to the final users.

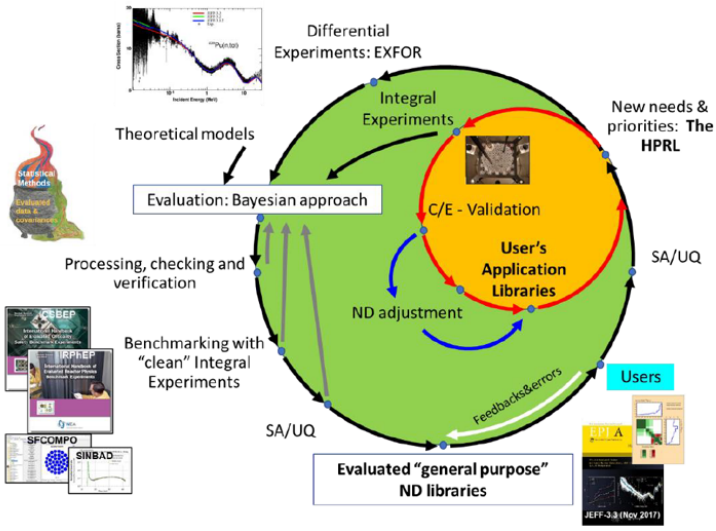


Figure 1.3: Nuclear data activities pipeline [9]

1.2 The ENDF6 format

The responsibility of providing evaluated nuclear data in a machine-readable format falls upon the nuclear data community. During the 1940s and 1950s, different formats for the evaluated nuclear data storage were offered, such as KEDAK in Germany, UKNDL in the UK, ENDL at Lawrence Livermore National Laboratory (LLNL) in the USA, ENDF in the rest of the USA, and SOKRATOR in the Soviet Union [9]. Regrettably, the incompatibility between these formats did not allow the exchange and no international effort was put for a common set of evaluated nuclear data. Later, in the 1980s, the United States released the ENDF/B-IV and ENDF/B-V evaluated nuclear data libraries, both of which adopted a subset of the ENDF format as their foundation. Meanwhile, in Europe, the JEFF-1 library and later the JEFF-2.2 library adopted the ENDF format, specifically the sixth version ENDF-6. Presently, all the major evaluated data libraries are released in the ENDF-6 format (universally accepted) and the responsibility for its development and maintenance has been entrusted to the US-CSEWG.
ENDF-6 format serves as a machine-readable format designed for the storage and retrieval of evaluated nuclear data, and it is used in nuclear technology applications. Its structure is hierarchical (Figure 1.4) and is based on 80-character records. Each record contains six data and, for each of them, 11 characters are used. The hierarchy of an ENDF-6 can be summarised as follows [18]:

Library (NLIB) is a set of evaluations produced by a specific evaluation group. It can encompass various type of nuclear data (referred to as sub-libraries)

and many different materials or isotopes. In general, a material correspond to a single nuclide, a natural element with its several isotopes, or a combination of several elements like compounds, alloys or molecules.

Sub-Library (NSUB) is a collection of evaluations designed to differentiate among various incident particles and types of data.

Material (MAT) is an integer number that uniquely identify a specific evaluated material. For more details about how a MAT number is generated, see Appendix C of the reference [34].

File (MF) is a two-digit number and is the division of materials into logical units, denoted as "File", each containing a distinct class of information such as fission multiplicity, cross-sections, decay data, energy and angular distributions, etc. From Table A.1, MF= 1, 3, 5, 31, 33, 35 play an important role in this framework.

Reaction (MT) is represented by a three-digit number, and it is the further division of a File into sections that provide details about specific reactions or type of supplementary information (see Table A.2). Here, the focus is mainly on MT= 1, 4, 18, 452, 455 and 456.

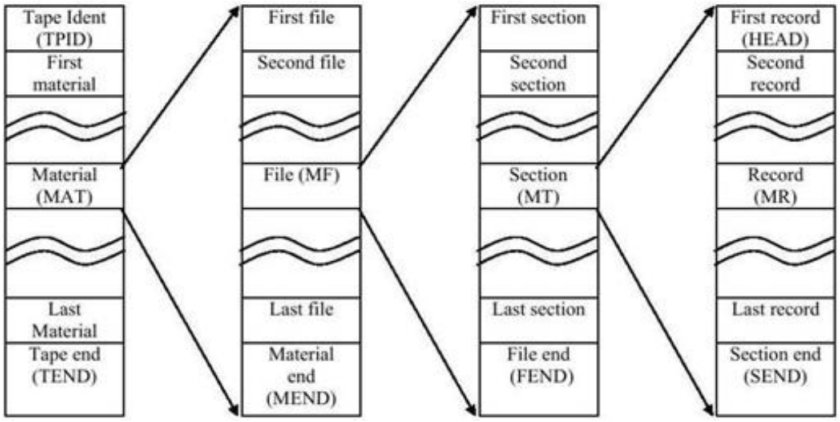


Figure 1.4: ENDF-6 file structure [18]

For more details about the ENDF-6 format, it is possible to consult the official manual [34].

Although ENDF-6 format has been widespread accepted [18], other nuclear applications (e.g. medical applications, space/radiation physics, non-proliferation,

waste management, etc.) discovered difficulties to exploit the ENDF-6 format for their planned use and the nuclear data community has acknowledged the necessity for a new nuclear data format. For this reason, in 2014, the WPEC Subgroup 38 [18], titled "Beyond the ENDF format: A modern nuclear database structure", was initiated with the objective of outlining the prerequisites for substituting the ENDF-6 format, initially devised for applications related to nuclear reactors. In 2020, the new format Generalised Nuclear Database Structure (GNDS) was finally released with its numerous advantages (e.g. xml and JSON acceptance). Currently, both formats coexist, but the ENDF-6 will be gradually substituted by the GNDS. Obviously, at this time the ENDF-6 is still predominant and, as a consequence, it is the format used to carry out this thesis.

1.3 NJOY

Processing codes are necessary to prepare evaluated nuclear data into the required format in order to be used in computer codes for simulation of a wide number of applications. As mentioned before, in subsection 1.1.3, NJOY [21] is the processing code chosen to prepare nuclear data in this thesis and it is launched by SANDY to create the ERRORR files (useful to produce the covariance matrices data) and to create the ACE files, requested by the Serpent2 code to run criticality simulations. The steps followed by NJOY, to perform the cited tasks, are shown respectively in the Figure 1.5 and Figure 1.6. Among the nuclear data community, NJOY stands out as the most widely used processing code. It is a complete computer code for specific-application libraries generation and for producing and processing pointwise and multigroup nuclear cross-sections - and related quantities - starting from an ENDF-6 data file. NJOY is developed and maintained by Los Alamos National Laboratory (LANL) and, in 2021, two version were available: NJOY2016 and NJOY2021. SANDY is currently built to work with NJOY2016 (Fortran version) and obviously this is the version adopted. NJOY2021 is written in C++. It was born to deal with the new format GNDS as well as the ENDF-6, and it is the forthcoming of NJOY.

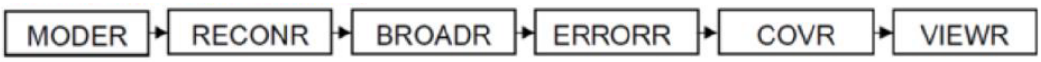


Figure 1.5: NJOY pipeline to process covariance matrices [18]

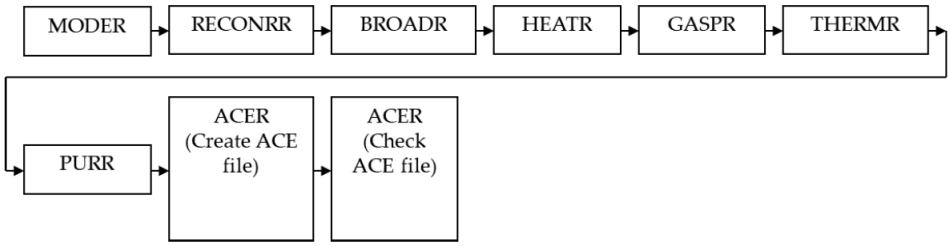


Figure 1.6: NJOY pipeline to convert evaluated files into ACE format [9]

NJOY comprises a series of core modules, each designed for specific processing functions, interconnected through input and output files. The main modules (most of them used to process cross-sections and covariance data and to create the ACE file) and the relative functions are listed here:

MODER: it converts ENDF-6 files changeably between the ASCII format and the distinctive NJOY blocked-binary format. Also, it performs the extraction of a single evaluation from a multi tape as well as the addition of an evaluation to an existing tape

RECONR: mainly reconstructs pointwise, energy-dependent cross-sections from resonance parameters and interpolation schemes. Linearisation and mesh unionization are two other important tasks

BROADR: to perform Doppler broadening depending on the user set temperature and to make the pointwise cross-sections thinner

UNRESR: unresolved resonance regions processing

GROUPR: for self-shielded multi-group cross-sections production and for group-to-group scattering and photon production matrices

ERRORR: to produce multi-group cross-sections and covariance data from point-wise data, consulting the uncertainties of the MF31-MF40 files

COVR: ERRORR covariance data processing. In particular, it processes the ERRORR output to BOXER format, an extremely compressed, card-image format for the covariance matrices storage. The BOXER format is largely used by NDaST, chosen for the validation of the results in the section 4.3.

ACER: it performs the conversion of the library to the ACE format for continuous energy Monte Carlo (MC) simulation codes: Serpent and Mcnp

WIMSR: it converts multigroup data into libraries compatible with the WIMSD and WIMSE codes

PLOTR: it prepares the ENDF, PENDF, GENDF cross-sections, distributions or matrices data for VIEWR

VIEWR: it displays plots from PLOTR and COVR in postscript format

PURR: it produces unresolved-region probability tables for MC and perform random sampling from the user defined amount

HEATR: to generate pointwise heat production (KERMA factors) and radiation-damage production cross-sections. In both cases, total or/and by reaction data types can be produced

GASPR: poitwise gas production cross-sections for p, d, t, ^{3}He and α

MIXR: combines cross-sections of different materials/reactions

THERMR: to generate cross-sections and energy-to-energy matrices for free or bound scatters in the thermal energy range.

More modules description and further information are found in the official NJOY manual [21].

1.4 Data uncertainty

Almost all the input data of a scientific field are affected by uncertainties, especially if they come from experimental measurements. It means it is not possible to compute precisely the value of a certain variable, but the latter is described by a mean value (or expected value/best estimate) and a variance. Whereas understanding the mean value could be a straightforward task, the variance concept could not be familiar and requires a short explanation to better understand it.

When dealing with an extensive dataset, it proves beneficial to summarise the whole dataset using a single value that characterises the "average" value of the entire collection. Statistically, this single value is referred to as a measure of central tendency, and the mean is a method employed to depict it. Given a certain set of N observations of a random variable x, the mean value is computed through the Equation 1.11. On the other hand, the random variable variance is a measure of the dispersion and describes how different or how spread out the data values are, and it is computed through the Equation 1.12 [35]:

$$\boxed{\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i} \tag{1.11}$$

$$\boxed{\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{1.12}$$

where x_i is the i$^{\text{th}}$ observation of the random variable x in the data sample and $\bar{x}$ is the variable x mean value. In other words, the variance is the mean quadratic error of the random variable x. The mean quadratic error has been chosen to represent the dispersion because, in case of positive and negative errors, they can combine and reduce the mean error values, giving a wrong statistical information. If there are more variables/parameters, also the covariance can be computed through the Equation 1.13 [35]:

$$\boxed{\sigma(x,y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})} \tag{1.13}$$

This represents the covariance between the variables x and y. The covariance quantitatively assesses how the variation of one variable, denoted as x, correlates with the variation of another variable, denoted as y, relative to their respective means. In other words, while the variance measures the spread of data along a single axis, the covariance measures the directional relationship between two variables. As a consequence, the variance can be considered as the covariance between a variable and itself (or the covariance as the variance between two variables) and it can be expressed as $\sigma(x,x)$. Furthermore, the variance/covariance can be expressed as relative term, called relative variance/covariance (Equation 1.14):

$$cov(x,y) = \frac{\sigma(x,y)}{\overline{x}\,\overline{y}} \tag{1.14}$$

The covariance can be positive, negative or zero:

Positive covariance describes two variables with the same behaviour and so they move in the same direction. For example, if higher values of one variable correspond to higher values of the other one, the covariance between them is positive.

Negative covariance describes two variables with opposite behaviour and so they move in opposite direction. In contrast to the positive one, larger values of one variable correspond to smaller values of the other one, and vice versa.

Zero covariance means that no relation is present between the two variables.

Another important statistical information of a dataset, between variables, is the correlation [36]. It is strictly related to the covariance and while the covariance

demonstrates how two variables change in relation to each other, the correlation assesses the strength and direction of this relationship (how intensely these two variables are linked). The correlation between two variables x and y is measured through the correlation coefficient ρ, expressed in the Equation 1.15. It is important to specify that the correlation coefficient is effective for assessing linear relationships, and it is not able to quantify non-linear ones. To extend it to non-linear relationships, alternative techniques, like variable transformation [37], are explored.

$$\boxed{\rho = \frac{\sigma(x,y)}{\sqrt{\sigma_x^2\,\sigma_y^2}}} \tag{1.15}$$

As previously stated, covariance and correlation are strongly connected concepts, they both measure linear relationship and can be positive, negative or zero (the concepts behind are the same). Simplifying, correlation is essentially a scaled or standardised version of covariance. In other words, correlation is a specific instance of covariance that come out when the data is in a standardised format. However, there are 3 substantial differences between the covariance and the correlation coefficient:

1. **Value range**: the correlation coefficient ranges from -1 to 1 while the covariance, in principle, from -∞ to +∞ and this explains the description above.

2. **Measurement units**: the correlation coefficient is dimensionless while the covariance is in units, given by the product of the two variables units.

3. **Scaling transformation**: the covariance is sensitive to changes in scale, the correlation coefficient not. For example, if all the values of both variables are multiplied by a constant (equal or different), the covariance changes while the correlation not.

1.4.1 Covariance matrix

In case of multidimensional data, the covariance concept is extended and generalised in the so-called covariance matrix. Indeed, the covariance matrix contains the variances in the diagonal values and the covariances in the others and for this reason it is also called variance-covariance matrix. To understand the meaning of each position in the matrix, it is fundamental to know how the covariance matrix is built. Given a matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ containing n observations of d random variables/parameters of a dataset, the associated covariance matrix $\mathbf{C}$ is computed as [38]:

$$\boxed{\mathbf{C} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{X_i} - \bar{\mathbf{X}})(\mathbf{X_i} - \bar{\mathbf{X}})^T} \tag{1.16}$$

where $\mathbf{X_i}$ is the i$^{\text{th}}$ observation vector of the d random variables and $\bar{\mathbf{X}}$ is the mean value vector of the d random variables. The number of random variables/parameters d is also called dimension because it determines the covariance matrix dimension. If the dataset has zero mean (i.e. the mean of all random variables is zero, $\bar{\mathbf{X}}$ is a null vector) the covariance matrix can be calculated with the Equation 1.17.

$$\boxed{\mathbf{C} = \frac{\mathbf{X}\mathbf{X}^T}{n-1}} \tag{1.17}$$

Therefore, $\mathbf{C} \in \mathbb{R}^{d\times d}$, explaining the definition of dimension for d. For example, the values of $\mathbf{C}$(4,3) is the covariance between the third and the fourth variable of the dataset and it is equal to $\mathbf{C}$(3,4). With the same consideration, it is possible to understand that the diagonal values are the variances. The relative covariance matrix is obtained by substituting the variances/covariances with the relative ones. The covariance matrix must be [38]:

Squared: indeed, $\mathbf{C} \in \mathbb{R}^{d\times d}$ and so the number of rows are equals to the number of columns.

Symmetric: since $\sigma(x, y) = \sigma(y, x)$, $\mathbf{C} = \mathbf{C}^T$($\mathbf{C}$ transpose matrix) and so the covariance matrix is symmetric.

Positive semi-definite: it means that:

- ✻ given a vector $\mathbf{u} \in \mathbb{R}^{n\times d}$, $\mathbf{uCu}^T \geq 0$
- ✻ all its eigenvalues are real and non-negative.

Nowadays, covariance and correlation matrices are becoming largely used in the scientific field, for different purposes. Some examples are [36]:

- In industries based on data, they play a pivotal role in recognising multivariate data, facilitating data processing and enabling efficient execution of analytical operations.
- Kalman filters: Kalman filters, also known as Kalman filtering, is an algorithm used to estimate unknown variables based on observed measurements over time. The covariance matrix is used to compute the weighted average that aids in predicting the intermediate system state lying between the predicted and measured states.

- In the Gaussian mixture models, the structure of a multivariate normal cluster can be interpreted by the covariance matrix.
- Knowing the correlation between data, and so their relationship, is the crucial step before implementing statistical models.
- The Principal Component Analysis (PCA): covariance matrices are employed to reduce the dimensions of extensive datasets, thereby improving their interpretability. Data scientists utilise PCA for tasks such as predictive analysis and exploratory data analysis.
- Analytical processes (e.g. multivariate analysis).
- In this thesis, a dataset is generated according to its mean and covariance matrix. This is called the **sampling process**. Therefore, starting from a covariance matrix and knowing the mean of a certain multivariate dataset, it is possible to create random samples but that reflect the statistical information of the covariance. Here, this process has been implemented in SANDY and the methodology will be explained in the section 2.3.

Summing up, the structure of a covariance matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$ is:

$$\mathbf{C} = \begin{cases} var(x_i) = \sigma_i^2 & \text{for } i = j \\ cov(x_i, x_j) = \rho_{ij}\sigma_i\sigma_j & \text{for } i \neq j \end{cases}$$

where x_i represent i-th variable and ρ_{ij} the correlation coefficients between x_i and x_j. In matrix form, given K parameters:

$$\mathbf{C} = \begin{bmatrix} var(x_1) & \dots & cov(x_1, x_K) \\ \vdots & \ddots & \vdots \\ cov(x_K, x_1) & \dots & var(x_K) \end{bmatrix}$$

In the instance of a nuclear-data covariance matrix, x_i is a nuclear data for a specific isotope/material, reaction (e.g. fission cross-section, nubar, PFNS, angular distributions, fission yields, ..., general MT number) and energy value.

Nuclear covariance matrix

Nuclear data, as many other data, are affected by uncertainty. In fact, they represent a substantial source of uncertainty in various reactor analyses, such as reactor criticality and safety assessments. For this reason, nuclear data evaluators initiated efforts several years ago to offer estimates of nuclear data uncertainties which characterise our understanding of the data in relation to their best fit with

experimental measurements. A short time ago, missing experimental data have been provided by including nuclear models uncertainties. Then, all this information has been integrated into public data evaluations, along with the corresponding nuclear data, in the form of covariance matrices. Furthermore, the ENDF-6 format itself has been modified in order to include also the new covariance information.

The main uncertainty sources of nuclear data arise from [39]:

Experimental measurements: these uncertainties come from performing experiments to measure nuclear properties such as cross-sections, decay rates, fission neutron multiplicity, decay yields or nuclear reaction energies. The measurement error is defined as the difference between the measured quantity and the reference quantity value, indicating a potential bias. It is important to specify that error and uncertainty are two different quantities. Indeed, uncertainty is more related to the dispersion of a distribution. About experimental measurements, it is possible to define the measurement **precision** and **accuracy** [9] (Figure 1.7). Precision serves as an indicator of the degree of agreement among measurements, specifically pertaining to the values of the measured quantity obtained through replicate measurements. On the other hand, accuracy indicates the degree of alignment of the measured values with the true/reference one.

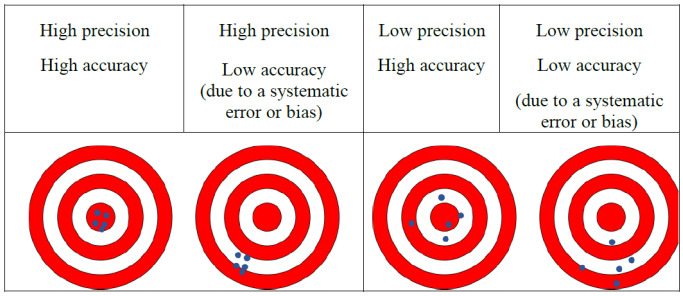


Figure 1.7: Precision and accuracy difference [9]

Connected with precision and accuracy are the **random** and **systematic** error. The first is related to the measurement precision and arises due to unpredictable or stochastic variations in influencing factors over time and space. Increasing the number of observations can mitigate this type of error. The second is related to the accuracy and occurs when a recognised influence quantity affects the measurement. In such cases, a correction factor can be applied to offset the impact of this factor. During measurement, systematic uncertainties can derive from:

- all the inputs uncertainty
- the incoming beam: the error results from the imprecise values of nuclear data standards used for beam calibration, which have an impact on measurements for the entire energy range
- the target: mass and impurities amount uncertainties give a systematic error on the measurements
- the detection device: attributed to partial understanding of measurement conditions, geometry, composition, dosimeter/detector location and efficiency
- differential approximations: they occur due to corrections related to multiple scattering based on MC methodologies
- from physics: because of insufficient expertise on nuclear data or models, like the model assumptions and defects.

Hence, all these errors lead to an uncertainty on the quantity to measure.

Theoretical models: frequently, theoretical nuclear models are trusted to interpolate or extrapolate nuclear data points. However, these models may carry inherent uncertainties due to simplifications, assumptions and their approximations.

Evaluation processes: the uncertainty depends on the evaluator itself and its way of working. It must be pointed out that "the" evaluator is a working team made of many people. It is fundamental that the evaluator has a wide knowledge of different field such as nuclear, statistics, informatics, etc. in order to properly evaluate the data, recognising inconsistencies and conflicts between them. During the evaluation process, also codes are used and they can lead to uncertainties if not verified and well-tested. The evaluator must ensure that all codes are reliable.

Processing: during this step, codes are used as well and, like the evaluation one, they must be reliable otherwise other uncertainties can arise. Here, the adjustment is performed and it is implemented when nuclear data are modified after new discovers and this operation can bring additional uncertainty.

In general, uncertainties from multiple sources can propagate through calculations, leading to cumulative uncertainties in derived nuclear data and quantifying it is mandatory to ensure nuclear data reliability.

All the nuclear data uncertainties are stored in the covariance matrix. Usually, different parameters are taken under consideration to build it: the material/isotope,

the reaction type and a variable linked to it, such as the energy. In SANDY, nuclear data are processed using the ERRORR module of NJOY which produces energy multi-group covariance matrices. In the Figure 1.8 is shown the multi-group energy relative covariance matrix of the uranium-235 (^{235}U) for the total nubar, considering the JEFF-33 library, obtained using SANDY.

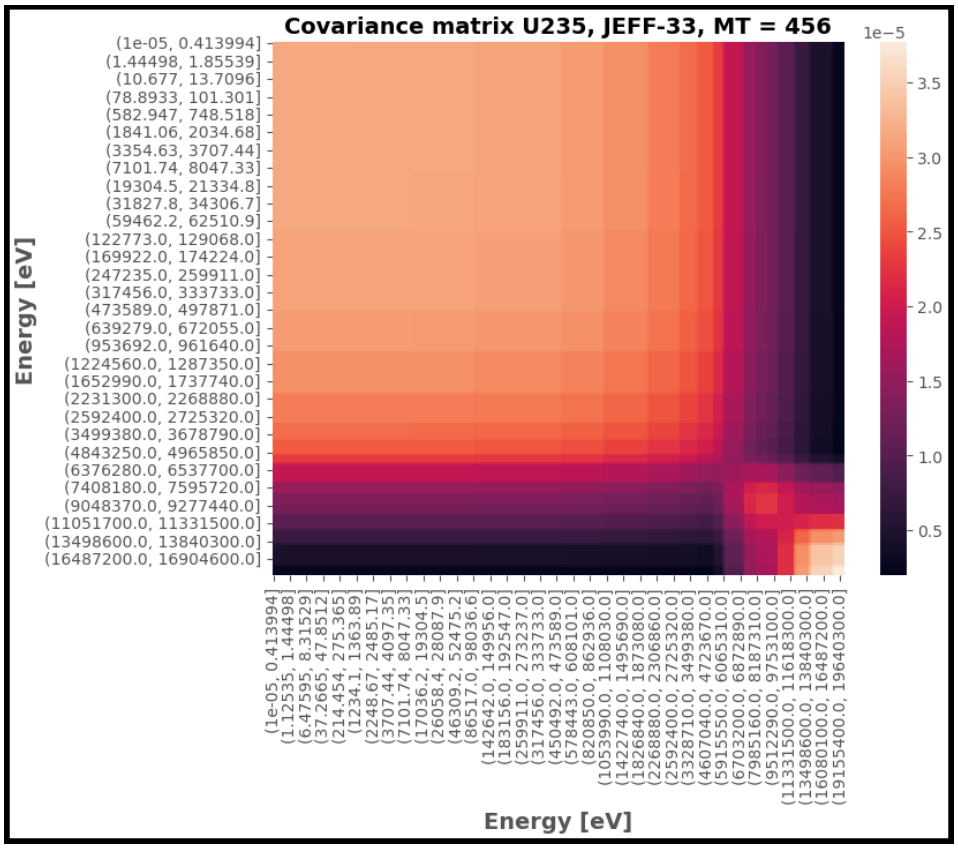


Figure 1.8: ^{235}U prompt nubar covariance matrix (JEFF-33)

Chapter 2

Sensitivity Analysis and Uncertainty Quantification

In the subsection 1.1.4, Sensitivity Analysis (SA) and Uncertainty Quantification (UQ) SA&UQ have been already mentioned. However, in this chapter, a more complete description will follow, especially in the calculation methods, and their role in the thesis will be shown. Lately, many engineering and science fields have largely started to increase the application of the sensitivity and uncertainty analysis, including activities related to experimental data processing, computational modelling and process simulation.
In order to perform these analyses, methods based either on deterministic or stochastic approach have been developed [40]. Generally, deterministic methods are based on precise, known inputs and mathematical equations, and they produce consistent and same outcomes for the same set of input data while stochastic methods involve randomness and uncertainty in the input data or in the process modelled and, thus, they provide probabilistic outcomes (with their distributions) that can vary each time the method is run, even with the same initial conditions. More in detail about this thesis, deterministic methods focus mainly in the local analysis and so the system response behaviour is investigated locally, around a chosen fixed point of the parameters and state variables phase space (i.e. input space). However, local sensitivities provide only the linear (or first order) contributions to the overall response variation and are considered valid only if the perturbations of the relative input parameters are small or, in other words, only around small neighbourhoods of the parameters' best estimate. On the other hands, stochastic methods explore all the input space leading to a global analysis where the system response is studied taking under consideration the whole variation range of the input parameters, and not only their closest neighbourhoods [18]. The pros and cons of the two methods are summarised in the Table 2.1 [41].

Table 2.1: Pros and cons of deterministic and stochastic methods

Method	Pros	Cons
Deterministic	• Very computational efficient	• Only linear system response • Only for small perturbations of the input variables (i.e. small input uncertainties). It means, the system response is studied around the neighbourhoods of the input parameters' best estimate • Only one parameter at time
Stochastic	• For any uncertainty magnitude of the input variables because it explores all the inputs space • More variables at time • Suitable also for non-linear functions or system responses	• Complex systems, with many input parameters, require a very high computational cost

Reactor systems are highly complex, influenced by thousands of input parameters, from nuclides and reaction types and energy values [42]. In addition, reactor physics predominantly deals with integral quantities which exhibit linearity concerning real and/or adjoint neutron fluxes (e.g. fission neutron multiplicity, cross-sections, average fission spectra, ...). Consequently, the assumption of linearity holds when dealing with the sensitivity of these reactor criticality safety quantities. As consequence, this property will be exploited later, when the sensitivity vector of the nubar will be needed to verify the computations of the method implemented in this framework (i.e. a stochastic one) for the UQ, since linearity is ensured and no (or almost null) errors will be introduced on the sensitivity calculation. Following, the methods' description with some tools/codes that are currently used to apply

them other than the one used to carry on this thesis.

2.1 Sensitivity Analysis

SA aims to precisely and effectively determine how the system's response changes when the system's parameters vary within their nominal values. Given a function $y = f(x_1, x_2, ..., x_d)$, representing a response of the system, the sensitivity of x_i with respect to the response y is defined as [18]:

$$\boxed{S_i = \left.\frac{\partial y}{\partial x_i}\right|_{\bar{\mathbf{x}}}} \tag{2.1}$$

where $\bar{\mathbf{x}} = [\bar{x}_1, \bar{x}_2, ..., \bar{x}_d]$ is the vector of all variables nominal values. Sensitivity coefficients, or sensitivities, can be eventually used to rank them and assess their relative importance on the response, evaluate alterations in the response resulting from parameter fluctuations, conduct uncertainty analysis employing the sandwich formula and be applied in the optimisation process, such as establishing target accuracy levels for reducing uncertainty [9]. Thus, a complete sensitivity analysis requires accurate sensitivity coefficients for all the system materials (recognising the most important isotope in each), the most relevant reactions (nubar, radiative capture cross-section, PFNS and in general energy distributions, etc.), the neutron multi-group or continuous energies. All of this must be computed for all the most relevant system responses (k_{eff}, decay heat, isotopic inventory, etc...). In this thesis, the response that will be analysed is the k_{eff} of well-known benchmarks (section 2.5). Sensitivity can be evaluated with either deterministic or stochastic methods [18].

2.1.1 Deterministic methods

Direct perturbation: the "brute force"

It consists of applying the finite difference approximation:

$$S_i = \left.\frac{\partial y}{\partial x_i}\right|_{\bar{\mathbf{x}}} \simeq \left.\frac{\Delta y}{\Delta x_i}\right|_{\bar{\mathbf{x}}} = \frac{y(x_1, x_2, ..., \bar{x}_i + \delta x_i, ..., \bar{x}_d) - y(\bar{\mathbf{x}})}{\delta x_i} \quad i = 1,2,...d \tag{2.2}$$

The Equation 2.2 requires (1+d) model computations (1 nominal + d direct perturbations) while the central difference implemented in Equation 2.3 requires 2*d computations.

$$S_i = \left.\frac{\partial y}{\partial x_i}\right|_{\bar{\mathbf{x}}} \simeq \left.\frac{\Delta y}{\Delta x_i}\right|_{\bar{\mathbf{x}}} = \frac{y(x_1, x_2, ..., \bar{x}_i + \delta x_i, ..., \bar{x}_d) - y(x_1, x_2, ..., \bar{x}_i - \delta x_i, ..., \bar{x}_d)}{2\delta x_i} \tag{2.3}$$

This method is very easy to implement and does not require other models development. However, the bottlenecks are the high computational cost and the trial-and-error loop to better select the perturbations δx_i. A short time ago, the computer code FRom Evaluated Nuclear Data librarY to any application (FRENDY) has been developed to compute sensitivity coefficients with direct perturbation method other than nuclear data perturbation capabilities (`https://rpg.jaea.go.jp/main/en/program_frendy/`). For example, it can be applied to compute the k_{eff} sensitivity:

$$S_i = \frac{k_{eff}(\bar{x}_1, \bar{x}_2, ..., \bar{x}_i + \delta x_i, ..., \bar{x}_d) - k_{eff}(\bar{\mathbf{x}})}{\delta x_i} \tag{2.4}$$

1st order Perturbation Theory

Given a vector $\mathbf{x} \in \mathbb{R}^{d\times 1}$ of nominal values for the input parameters for a response function y and their uncertainties (δx_1, δx_2,..., δx_d), the function y can be written using the Taylor series, expanded on the mean value $\bar{\mathbf{x}}$ up to the n^{th} order with variations δ_{x_i} as:

$$\begin{aligned} y(x_1, x_2, ..., x_d) = y(\bar{\mathbf{x}}) &+ \sum_{i_1=1}^{d} \left.\frac{\partial y}{\partial x_{i_1}}\right|_{\bar{\mathbf{x}}} \delta x_{i_1} + \frac{1}{2} \sum_{i_1,i_2=1}^{d} \left.\frac{\partial^2 y}{\partial x_{i_1}\, \partial x_{i_2}}\right|_{\bar{\mathbf{x}}} \delta x_{i_1} \delta x_{i_2} \\ &+ ... + \frac{1}{n!} \sum_{i_1,i_2,...,i_n=1}^{d} \left.\frac{\partial^n y}{\partial x_{i_1}, \partial x_{i_2}, ..., \partial x_{i_n}}\right|_{\bar{\mathbf{x}}} \delta x_{i_1}, \delta x_{i_2}, ..., \delta x_{i_n} \end{aligned} \tag{2.5}$$

Therefore, if linearity is assumed, the Taylor expansion can be truncate to the first order and the response function can be written as:

$$\boxed{y(x_1, x_2, ..., x_d) = y(\bar{\mathbf{x}}) + \sum_{i_1=1}^{d} \left.\frac{\partial y}{\partial x_{i_1}}\right|_{\bar{\mathbf{x}}} \delta x_{i_1} = y(\bar{\mathbf{x}}) + \sum_{i=1}^{d} S_i \delta x_i} \tag{2.6}$$

where $S_i = \left.\frac{\partial y}{\partial x_i}\right|_{\bar{\mathbf{x}}}$ is the parameter x_i sensitivity for the response y. This procedure is suitable to calculate sensitivities for a response where the dependence on x_i is exclusively direct, like the k_{eff} sensitivity.

Generalised Perturbation Theory

Certainly, the k_{eff} is not the sole focus of interest in reactor sensitivity investigations. Generally, in transport calculations, the sensitivity coefficients can be split in two terms [9]:

- the **explicit** sensitivity coefficient, related to a problem-specific (self-shielded) multi-group parameter $x^i_{a,g}$, featured in the transport calculation, on the response y, is formally defined as:

$$(S_{x^i_{a,g}})_{explicit} = \frac{\partial y/y}{\partial x^i_{a,g}/x^i_{a,g}} \tag{2.7}$$

  where $x^i_{a,g}$ is the parameter for the reaction a of the nuclide i within energy group g.

- the **implicit** sensitivity coefficient takes into account the impact that alterations in one cross-section may have on other problem-specific multi-group cross-sections due to self-shielding perturbations. The summation encompasses all parameters linked to the same group-wise cross-section, i.e. nuclide-reaction pairs a, b and energy groups h, g for the nuclides i, j.

$$(S_{x^i_{a,g}})_{implicit} = \sum_j \sum_h \frac{\partial y/y}{\partial x^j_{b,h}/x^j_{b,h}} \times \frac{\partial x^j_{b,h}/x^j_{b,h}}{\partial x^i_{a,g}/x^i_{a,g}} \tag{2.8}$$

Therefore, the "total" sensitivity coefficient is computed as:

$$(S_{x^i_{a,g}})_{total} = (S_{x^i_{a,g}})_{explicit} + (S_{x^i_{a,g}})_{implicit} \tag{2.9}$$

For example, the elastic scattering of the hydrogen not only influences k_{eff} due to the effects of hydrogen moderation, but it leads to alterations in the self-shielded uranium-238 (^{238}U) cross-section as well and, as a consequence, k_{eff} is indirectly changed by it. While the explicit term can be solved by using the first order perturbation method, the implicit one needs to satisfy an orthogonality condition, that goes very far from this thesis purpose. If a response satisfies the previous condition, the impact of a perturbation of a general parameter on the response can be computed through the so-called Generalised perturbation Theory (GPT) [43] and the latter is based on a linear perturbation approach and can be used, for example, to compute sensitivities of the effective generation neutron time, the reactivity worth and the effective delayed neutron fraction (β_{eff}). Certainly, GPT necessitates the computation of generalised solutions, which can be more challenging to calculate due to the presence of the indirect term.

Other deterministic methods, based on linearity assumptions, are Forward Sensitivity Analysis Procedure (FSAP) and Adjoint Sensitivity Analysis Procedure (ASAP). However, they are much distant from the objective of this thesis and detailed description can be obtained from the reference [44].

2.1.2 Stochastic methods

Sobol's method

From a variance analysis study, carried by Sobol [45], it came out that the response function can be decomposed as:

$$f(\mathbf{x}) = f(\bar{\mathbf{x}}) + \sum_{i<j} f_{ij}(x_i, x_j) + ... + f_{1,2,...,d}(x_1, x_2, ..., x_d) \tag{2.10}$$

In order to apply the Equation 2.10, its terms must be orthogonal so that they can be expressed as integrals of $f(\mathbf{x})$ and the following condition must be satisfied:

$$\int_0^1 f_{i_1,i_2,...,i_d(x_{i_1},x_{i_2},...,x_{i_d}})dx_k = 0 \text{ for } k = i_1, i_2, ..., i_d \tag{2.11}$$

$f(\mathbf{x})$ is assumed square integrable and, as a consequence, Equation 2.10 can be first squared and then integrated [18]:

$$\int f^2(\mathbf{x})dx - f^2(\bar{\mathbf{x}}) = \sum_{s=1}^{d} \sum_{i_1<...<i_s}^{d} \int f^2_{i_1,i_2,...,i_d} dx_{i_1}, dx_{i_2}, ..., dx_{i_d} \tag{2.12}$$

Analysing Equation 2.12, the left-hand side is the variance of y definition, while the right one indicates the terms of the decomposed variance. Finally, the variance can be written as:

$$V_y = \sum_{i=1}^{d} V_i + \sum_{i<j}^{d} V_{i,j} + ... + V_{1,2,...,d} \tag{2.13}$$

V_i is the response marginal variance of the conditional mean $E[y|x_i]$, computed across all values of x_i and over all factors except for x_i. In mathematical terms:

$$V_i = V_{x_i}(E[y|x_i]) \tag{2.14}$$

Also, the variance of the response is influenced by the interaction between x_i and x_j and this contribution is described as:

$$V_{ij} = V_{x_i,x_j}(E[y|x_i, x_j]) - V_{x_i}(E[y|x_i]) - V_{x_j}(E[y|x_j]) \tag{2.15}$$

At the same way, the contributions to the response variance, due to the interaction between more than 2 variables, can be computed as well until the number of random variables/parameters d. Furthermore, $S_{i,...,j} = \frac{V_{i,...,j}}{V_y}$ are called global sensitivity indices and introducing them into Equation 2.13, the following results are obtained:

$$\sum_{i=1}^{d} S_i = \sum_{i<j}^{d} S_{i,j} + ... + S_{1,2,...,d} = 1 \tag{2.16}$$

and S_i is called individual sensitivity index of x_i and it corresponds to the sensitivity definition of the section 2.1.1. Finally, the total effect sensitivity index S_{T_i} can be computed and it represents the individual contribution (the linear one) plus all the possible higher orders effects of x_i:

$$S_{T_i} = S_i + \sum_{i \neq j}^{d} S_{i,j} + ... + S_{1,2,...,d} \tag{2.17}$$

Regarding nuclear field, there are different codes to compute sensitivities [9], based on the described methods:

- **Standardized Computer Analyses for Licensing Evaluation (SCALE)** code system (`https://www.ornl.gov/scale`) has developed the TSUNAMI code [46], designed for conducting sensitivity analyses related to nuclear data. Three different versions are available: TSUNAMI-1D/2D for one or two dimensions sensitivity analysis based on GPT and TSUNAMI-3D for three dimensions analysis, based on the KENO (a MC) code, which gives sensitivity profiles in either continuous or multi-group energy.

- **MCNP** code (`https://mcnp.lanl.gov/`) for continuous energy

- **Serpent** code, specialised in computing sensitivities for bilinear ratios. Serpent utilises an implementation based on the collision history, equivalent to GPT, to compute the sensitivities of different responses to various perturbations. In the 2.1.31 version, these capabilities were improved to enable the use of Serpent estimates for uncertainty propagation. In addition, starting from version 2.2.1 and onwards, this methodology is applicable not only to criticality calculations but also external source ones have been implemented.

Figure 2.1 and Figure 2.2 show respectively the sensitivity profiles of some plutonium benchmarks (section 2.5) k_{eff} to the total nubar of ^{239}Pu and the sensitivity profiles of Jezebel benchmark k_{eff} to the total nubar and some important cross-sections of ^{239}Pu and plutonium-240 (^{240}Pu), computed with TSUNAMI 1D. It is possible to see that the sensitivities are higher at energy of about 2 MeV, that corresponds to the average neutron energy generated by fission. Furthermore, the nubar of ^{239}Pu is the reaction with the highest sensitivity because these benchmarks are referred to very simple system, with few materials, working mainly with fast neutrons. Since the ^{239}Pu is the main fissile isotope in these benchmarks and due to their simplicity, they have been chosen for the verification of the methodology and the code implemented (chapter 3 and section 2.3). Indeed, from the Figure 2.1 and Figure 2.2, the highest sensitivity correspond to the ^{239}Pu nubar at the energy of 2 MeV, demonstrating what has been explained above. In general, the highest

sensitivities, of a fixed reaction, is located at the energy of 2 MeV because they are fast nuclear systems.

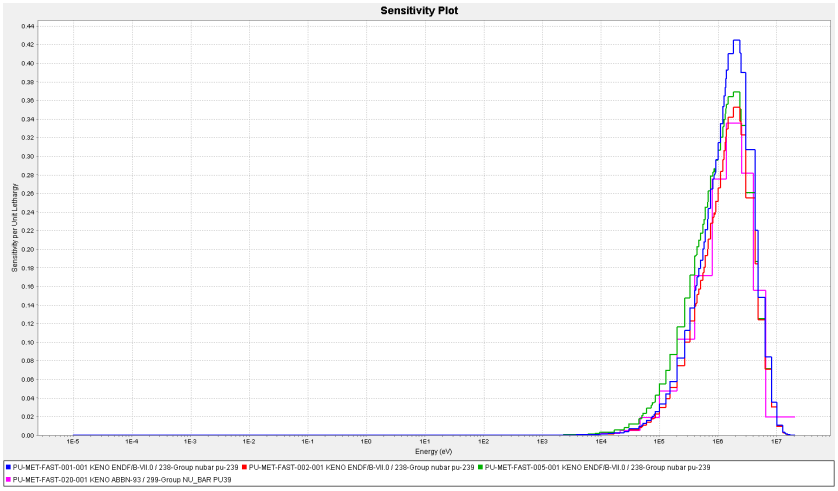


Figure 2.1: Sensitivity profile of some plutonium benchmarks k_{eff} to the ^{239}Pu total nubar. Computed with TSUNAMI 1-D code.

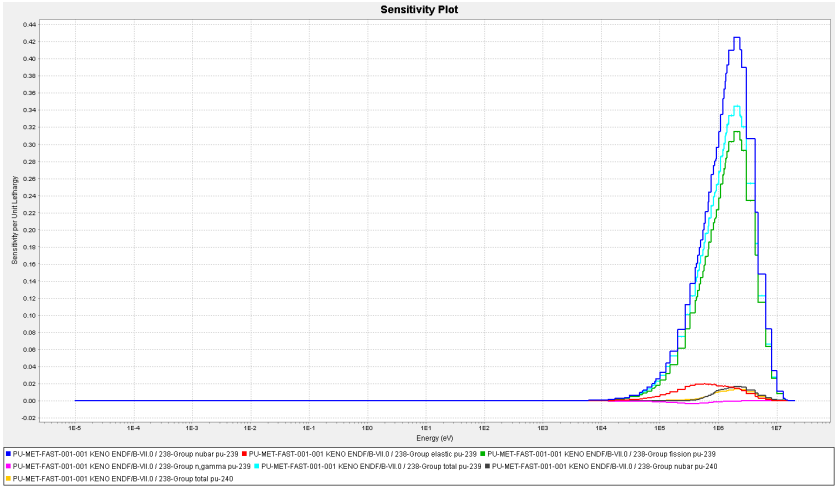


Figure 2.2: Sensitivity profile of the Jezebel benchmark k_{eff} to the total nubar and total, elastic scattering, radiative capture and fission cross-section of ^{239}Pu and ^{240}Pu. Computed with TSUNAMI 1-D code.

2.2 Uncertainty Quantification

In general, the objective of UQ is to rank all important uncertainties of a certain input dataset and to compute their impact on the response function. More in general, it aims to compute the statistical moments for a given response function y. The corresponding equations for these moments, such as the expected value $E[y]$ (1^{st} moment), the variance $var(y)$ (2^{nd} moment), and so on, are referred to as the moment propagation equations. In particular, the uncertainty propagation analysis aims to quantify the uncertainty on the response function due to the uncertainty of the input parameters/variables. UQ/propagation analysis can be performed both with deterministic and stochastic methods that will be detailed in the following two subsections.

2.2.1 Deterministic methods

The sandwich rule

The sandwich rule [47] assumes linearity of the general response function y and thus section 2.1.1 is exploited to perform the analysis. Given a set of r response functions $\mathbf{y} \in \mathbb{R}^{r\times 1}$ and a set of d random variables/parameters $\mathbf{x} \in \mathbb{R}^{d\times 1}$ and assuming linearity of the response functions, the Equation 2.6 can be written in matrix form as:

$$\boxed{\mathbf{y} - \bar{\mathbf{y}} = \mathbf{S}(\mathbf{x} - \bar{\mathbf{x}})} \tag{2.18}$$

where $\bar{\mathbf{y}}$ is $f(\bar{\mathbf{x}})$ and $\mathbf{S} \in \mathbb{R}^{r\times d}$ is the sensitivity coefficients matrix (or vector in case of one response) of the response functions to all system parameters:

$$\mathbf{S} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1}\Big|_{\bar{\mathbf{x}}} & \cdots & \frac{\partial y_1}{\partial x_d}\Big|_{\bar{\mathbf{x}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_r}{\partial x_1}\Big|_{\bar{\mathbf{x}}} & \cdots & \frac{\partial y_r}{\partial x_d}\Big|_{\bar{\mathbf{x}}} \end{bmatrix}$$

Generally, for a nuclear system d = number of isotopes x number of reactions x number of energy bins, and the element at position (ij) represents the sensitivity of the i^{th} response to the j^{th} input parameter. Applying the definition of variance/covariance, in matrix form:

$$cov(\mathbf{y}) = E[(\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T] = E[(\mathbf{S}(\mathbf{x} - \bar{\mathbf{x}}))(\mathbf{S}(\mathbf{x} - \bar{\mathbf{x}})^T)$$

Since the sensitivity matrix is, by definition, a linear function, it can go out the linear operator E:

$$cov(\mathbf{y}) = \mathbf{S}E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T]\mathbf{S}^T$$

$E[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T]$ is nothing but the covariance matrix definition by using the linear operator E[]. Hence:

$$\boxed{cov(\mathbf{y}) = \mathbf{S}\mathbf{C}\mathbf{S}^T} \tag{2.19}$$

in non-matrix form:

$$\boxed{cov(\mathbf{y}) = \sum_{i=1}^{d} S_i^2 var(x_i) + 2\sum_{i\neq j=1}^{d} S_i S_j cov(x_i, x_j)} \tag{2.20}$$

Equation 2.19 is called **sandwich rule** or **sandwich formula** [18]. If only one response, cov($\mathbf{y}$) is the variance of it and corresponds to a single value. However, in the most general case, more than one response is investigated and thus cov($\mathbf{y}$) contains the covariance matrix of the different responses, with their variances (on the diagonal) and covariances (non-diagonal terms). Hence, once the sensitivity matrix (or vector) is obtained, knowing the covariance matrix (i.e. uncertainties) of the input data, it is possible to compute the uncertainty of the output (i.e. the response). As previously described, this is only valid for linear responses and in case of non-linear one, the sandwich rule is not precise, it becomes an approximation and gives an error. The latter is larger as the larger are the uncertainties on the input data. Also, for linear responses:

$$E[\mathbf{y}] = E[\bar{\mathbf{y}} + \mathbf{S}(\mathbf{x}-\bar{\mathbf{x}})] = \bar{\mathbf{y}} + \mathbf{S}E[\mathbf{x}] - \mathbf{S}E[\bar{\mathbf{x}}] = \bar{\mathbf{y}} + \mathbf{S}\bar{\mathbf{x}} - \mathbf{S}\bar{\mathbf{x}}$$

Therefore, for linear systems:

$$E[\mathbf{y}] = \bar{\mathbf{y}} = f(\bar{\mathbf{x}}) \tag{2.21}$$

while for non-linear systems:

$$E[\mathbf{y}] \neq f(\bar{\mathbf{x}})$$

The main drawbacks of using the sandwich rule as UQ/propagation technique are [9]:

1. inefficiency with numerous response functions: the method exhibits reduced efficiency when dealing with numerous response functions because the count of adjoint functions enhances linearly with the number of system responses

2. limited applicability for small uncertainties: its applicability is confined to small uncertainties since it relies on a linear approach and substantial uncertainties might extend beyond the model's linear range

3. constraints due to non-linearity in multi-physics: the method faces limitations arisen from the inherent non-linearity of multi-physics aspects, such as neutronics, thermal-hydraulics, depletion, etc..., within nuclear reactor calculations.

Some codes have been developed to perform UQ/propagation analysis. The most important are [9]:

- **TSUNAMI** code represents a module of the SCALE code system, useful to compute sensitivities and perform UQ/propagation analysis.
- **Nuclear Data Sensitivity Tool (NDaST)** code is based on Java and specifically developed for conducting uncertainty calculations on sensitivity files for benchmark cases related to evaluated nuclear covariance data. The uncertainty propagation is performed by applying the sandwich rule (Equation 2.19). This software will be used as verification of the method based on SANDY, implemented in this framework. More details in the section 2.4.

2.2.2 Stochastic methods

Monte Carlo technique

The UQ/propagation method, based on MC technique, performs a random sampling of nuclear data libraries and, for each sample generated, a separated calculation is carried out. In particular, the random sampling is a stochastic technique in which variables characterised by uncertainties, treated as random variables, are sampled according to their mean values and covariance matrices, along with their corresponding probability distribution functions. Consequently, each sample becomes the input for the mathematical model or code that computes the response functions to be analysed [48]. At the end of the procedure, the outputs are processed using the statistical analysis in order to compute the total uncertainty in the response functions arisen from the input variables/parameters uncertainties. The MC method is not based on approximations, like first order or liner approximation, and hence any-order effect is investigated. However, it is a stochastic method and thus, since it carries inherent uncertainty due to the statistics, proportional to $1/\sqrt{n}$ for the Central Limit Theory (CLT), a substantial quantity of samples is necessary to reach statistical convergence of the response functions/outputs. Lately, computer performances have been improved and this, as well as model simplifications and dimensionality reductions, have led to a greater consideration of the described method. In addition, since the input parameters are affected by uncertainty, also the sampling process itself must reach a convergence of the samples. In other words, the generated samples must reflect the statistical information of the inputs (i.e.

mean values, covariance matrices and distributions) and their convergence must be reached before performing all the calculations. This method is employed to address the global impact of uncertainties related to all nuclear data, but nevertheless it can be also utilised for examining the specific impact of each individual system parameter on the total uncertainty of all the response functions. Furthermore, its utilisation with any model is simple since it doesn't interact with the model itself but solely with the model inputs and so there is no necessity to create complex algorithms for computing the adjoint functions of the model. Figure 2.3 shows the scheme adopted for the implementation of the MC technique, where n is the number of samples.

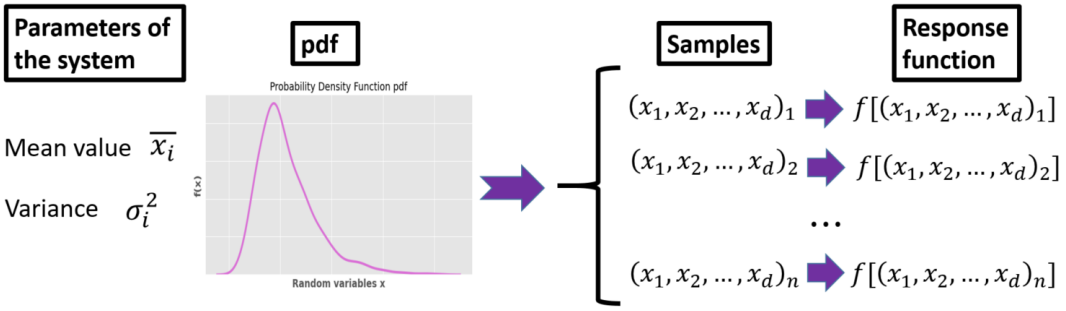


Figure 2.3: MC technique for UQ/propagation scheme

After n calculations have been performed, the mean value and the variance/-covariance of the outputs are computed respectively through Equation 2.22 and Equation 2.23 and they refer to the most general case (i.e. multiple response function and multiple variables). In case of a single response function, as described before, $cov(\mathbf{y})$ is simply the variance of the single output y.

$$\boxed{E[\mathbf{y}] = \bar{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n} y_i} \tag{2.22}$$

$$\boxed{cov(\mathbf{y}) \approx \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x_i} - \bar{\mathbf{x}})(\mathbf{x_i} - \bar{\mathbf{x}})^T} \tag{2.23}$$

Different tools have been implemented to perform the stochastic sampling of nuclear data libraries [9]:

- **FRENDY** code is able to perturb directly the ACE file of a nuclear data, based on its covariance matrix. First, the perturbation factors are generated using random sampling according to the covariance matrix in GENDF format, which typically is handled through the ERRORR module of NJOY. After,

these factors are applied to the ACE file with the "ACE file perturbation tool", making them suitable to codes like Serpent

- **TENDL**: as described in subsection 1.1.2, it is a nuclear library that come from the output of TALYS nuclear model code system for immediate utilisation in both fundamental physics and practical applications. The samples are generated by varying randomly the TENDL model parameters. The last release TENDL 2021 performs sampling for fission yields, thermal scattering, ENDF-6 and so ACE files

- **NUclear Data UNcertainty Analysis (NUDUNA)** [49] takes as input the information given by nuclear data evaluator or covariance matrices and then, the derived perturbations factors are applied to continuous energy data

- **XSUSA** [50] utilises the same technique as NUDUNA but here the perturbation factors are applied to multi-group energy data

- **SANDY** is a Python package and has been created with the purpose of generating perturbed nuclear data files starting from sampling coefficients which has been produced in accordance with the related evaluated nuclear data covariance matrices and the probability density functions. These files can be subsequently employed to propagate uncertainties throughout any compatible system through a brute force approach. As mentioned in the section , this framework aims to extend/improve the use of SANDY and to verify it and the UQ/propagation analysis derived from it.

2.3 SANDY code

SANDY has been developed at SCK CEN [51] by Luca Fiorito. Basically, SANDY is a Python package [52] designed to handle nuclear data files in the ENDF-6 format, offering capabilities for reading, writing, and executing various operations on them [8] (available at `https://github.com/luca-fiorito-11/sandy`). However, the primary objective of SANDY is to perturb nuclear data files and, consequently, create nuclear data samples, in a proper format, to be utilised in nuclear reactors' simulation codes like Serpent and Mcnp. For parsing ENDF-6 files, extracting nuclear data, constructing covariance matrices, extracting sample parameters from probability distributions and executing other tasks related to nuclear data, a sophisticated Python interface has been developed. For this reason, it is open-source and can be utilised by everyone unlike many other nuclear sampling codes, other than having advantages on the UQ/propagation [53] given by a stochastic sampling (see Table 2.1). The process used by SANDY to create perturbed nuclear data files is:

1. First, the nuclear data are read, extracting them from the ENDF-6 file, in continuous energy range. Usually it ranges from 10^{-5} eV to 20 MeV.

2. After, the covariance matrix is extracted from the ENDF-6, but the latter is first processed by NJOY and, as consequence, the resulting covariance matrix is in multi-group energy.

3. Then, the perturbation coefficients are extracted from the correspondent covariance matrix, after a distribution (probability density function) has been assumed since the covariance matrix does not contain information about it. The distribution is an assumption and usually the multivariate Normal distribution is used, even if SANDY can work also with the uniform and log-Normal distribution [53]. Since the covariance matrix is in multi-group energy also the derived perturbation coefficients will have the same structure.

4. Finally, the perturbed coefficients are applied to the continuous energy nuclear data and the perturbed nuclear data are stored in a file with any format, suitable to be inserted in a specific nuclear code. In this thesis, the ACE format will be deployed.

Thus, after perturbed data files are generated, these can be subsequently employed to propagate uncertainties throughout any compatible system through a brute force approach, hence with the stochastic method described in section 2.2.2. This procedure for propagating uncertainties can be employed with any model, response or computer code, as far as they are compatible with ENDF-6 format files, either through direct compatibility or through data conversion. Moreover, the modified files are not influenced by any inherent effects arising from processing, such as multi-group assumptions, self-shielding, or temperature influences [8]. At the time this framework started, SANDY was implemented to work with cross-sections, fission yields and decay data. In subsection 3.2.1, the methodology adopted to extend the use of SANDY with the average fission neutron multiplicity (nubar) will be explained. Moreover, many steps about the energy distributions data has been performed but not completed (and so verified) and in the subsection 3.2.2 they will be described.

2.3.1 Theory: sampling methodology

The general sampling methodology will be explained but before some assumptions, to simplify the understanding and to make it "general", have to be stated. The sampling methodology for a specific nuclear data are described in depth in the subsection 3.2.1 and subsection 3.2.2. Suppose that within a given ENDF-6 file, there exists a single covariance matrix Σ so that it is [8]:

- ✻ defined over G energy groups
- ✻ a relative covariance matrix (see subsection 1.4.1)
- ✻ characteristic of a continuous energy data type

With the previous assumptions, generality is still applicable, and they are common to almost all covariance matrices present in the ENDF-6 format. As described, SANDY extracts the covariance matrix from the ENDF-6 file and, according to the statistical information included, it constructs a multivariate Normal distribution $\mathbf{N}(\mu, \mathbf{\Sigma})$. The first step is to generate a matrix $\mathbf{X} \in \mathbb{R}^{G \times n}$ of G independent variables $\mathbf{x} = [x_1, x_2, ..., x_g]$, all normally distributed $\mathbf{N}(0,1)$ (so with mean value 0 and variance 1) with a number of samples equal to n.

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ x_g^{(1)} & x_g^{(2)} & \dots & x_g^{(n)} \end{bmatrix}$$

Then, a linear operator $\mathbf{L}$ such that, applied to the uncorrelated samples $\mathbf{X}$, turns them into $\mathbf{K}$. $\mathbf{K}$ contains the samples distributed as $\mathbf{N}(0, \Sigma)$:

$$\mathbf{K} = \mathbf{L}\mathbf{X}$$

and it is possible to demonstrate that $\mathbf{L}$ does not influence the mean of the distribution, since L is linear. For the same reason, also the type of the distribution is preserved.

$$E[\mathbf{K}] = E[\mathbf{L}\mathbf{X}] = \mathbf{L}E[\mathbf{X}] = 0$$

Since the distribution mean is zero, Equation 1.17 can be employed to compute the covariance matrix of $\mathbf{K}$:

$$\mathbf{\Sigma} = E[\mathbf{K}\mathbf{K}^\mathbf{T}] = E[(\mathbf{L}\mathbf{X})(\mathbf{L}\mathbf{X})^T] = \mathbf{L}E[\mathbf{X}\mathbf{X}^\mathbf{T}]\mathbf{L}^\mathbf{T}$$

$E[\mathbf{X}\mathbf{X}^T]$ is, by definition, the covariance matrix of the $\mathbf{X}$ samples and is equal to the identity matrix. Thus:

$$\mathbf{L}\mathbf{L}^\mathbf{T} = \mathbf{\Sigma} \tag{2.24}$$

and the matrix $\mathbf{L}$ is computed through the eigendecomposition of $\mathbf{\Sigma}$, performed by SANDY. To be more useful, the $N(0, \mathbf{\Sigma})$-distributed matrix $\mathbf{K}$ can be converted to a $N(1, \mathbf{\Sigma})$-distributed matrix $\mathbf{P} \in \mathbb{R}^{G \times n}$ with all the values of the mean vector equal 1:

$$\mathbf{P} = \begin{bmatrix} p_1^{(1)} & p_1^{(2)} & \dots & p_1^{(n)} \\ p_2^{(1)} & p_2^{(2)} & \dots & p_2^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ p_g^{(1)} & p_g^{(2)} & \dots & p_g^{(n)} \end{bmatrix} = \mathbf{K} + \mathbf{1} \tag{2.25}$$

where $\mathbf{1} \in \mathbb{R}^{G \times n}$ is a matrix with all values equal to 1 and the p_i^j represent the j^{th} perturbation coefficient for the i^{th} energy group. Thanks to this, given the matrix of the related variables nominal values $\mathbf{V} \in \mathbb{R}^{G \times G}$:

$$\boxed{\mathbf{S} = \mathbf{VP}} \tag{2.26}$$

where:

$$\mathbf{V} = \begin{bmatrix} v_1 & 0 & \dots & 0 \\ 0 & v_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & v_g \end{bmatrix}$$

and $\mathbf{S}$ is the matrix containing the perturbed nuclear data.
Let's demonstrate that the statistical information included are coherent with the purpose of the sampling. First, the mean value is (being $\mathbf{V}$ a constant):

$$\bar{\mathbf{S}} = E[\mathbf{S}] = E[\mathbf{VP}] = \mathbf{V}E[\mathbf{P}] = \mathbf{V}$$

and so it is verified. The covariance matrix of $\mathbf{S}$ is:

$$E[(\mathbf{S} - \bar{\mathbf{S}})(\mathbf{S} - \bar{\mathbf{S}})^T] = E[(\mathbf{VP} - \mathbf{V})(\mathbf{VP} - \mathbf{V})^T] = \mathbf{V}E[(\mathbf{P} - \mathbf{1})(\mathbf{P} - \mathbf{1})^T]\mathbf{V}^T$$

Since $E[(\mathbf{P} - \mathbf{1})(\mathbf{P} - \mathbf{1})^T]$ is the definition of the covariance matrix of $\mathbf{P}$:

$$E[(\mathbf{S} - \bar{\mathbf{S}})(\mathbf{S} - \bar{\mathbf{S}})^T] = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T$$

and since $\boldsymbol{\Sigma}$ is the relative covariance matrix, $\mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T$ represents the absolute covariance matrix and so the sampling has been performed correctly.

The data presented in the ENDF-6 files are not structured according to multi-group energy; instead, the nominal values are defined for a continuous-energy domain and consist of explicitly provided energy-value pairs along with interpolation rules [8]. Given a certain energy group $[e_i, e_{i+1}]$ with its perturbation coefficient p_i^j, SANDY perturbs all the nominal values v_k whose energy e_k is such that $e_i \leq e_k \leq e_{i+1}$, applying the formula [8]:

$$v_k^{(j)} = v_k p_i^{(j)}$$

2.3.2 Negative eigenvalues

As see in the subsection 1.4.1, the covariance matrix is positive-definite and so all its eigenvalues are positive. Moreover, this property is mandatory in order to apply the eigendecomposition to solve the Equation 2.24. However, often the covariance matrices, stored in the ENDF-6 file, do not satisfy this requirement. Obviously, SANDY is not able to solve directly this issue and find its origin, but another option has been considered. SANDY sets all the negative eigenvalues to zero and, then, it reconstructs a new covariance matrix $\tilde{\mathbf{\Sigma}}$ [8]. The previous approximation is valid when dealing with small negative eigenvalues that are prone to derive from rounding or truncation procedures. In these instance, $\tilde{\mathbf{\Sigma}} \approx \mathbf{\Sigma}$.

2.3.3 Negative samples

When the standard deviations are large, in particular when the relative standard deviation start to approach 40 %, the resulting perturbation coefficients, assuming Normal distribution, are more probable to be smaller than zero, producing negative samples when they are applied to evaluated nuclear data. Many data, such as cross-sections, fission neutron multiplicity or energy distributions, cannot be negative because they lost their physical meaning. In SANDY two methods have been suggested [8]:

Method 1: perturbation coefficients smaller than 0 or higher than 2 are set to 1

Method 2: perturbation coefficients smaller than 0 or higher than 2 are set respectively to 0 or 2

The upper limit 2 is set in order to have a symmetric probability density function and, thus, is just a matter of choice. However, cutting the probability density function deletes some statistical information and, hence, the standard deviation of the samples will be lower than the one contained in the covariance matrix. Obviously, the higher is the standard deviation of the covariance matrix, the higher will be the error introduced. Moreover, since the method 2 leads to lower reduction of the standard deviation, it has been implemented in the code. Furthermore, to better deal with this problem, a log-Normal distribution can be adopted, avoiding the negative samples.

Indeed, recently, a new solution has been found: instead of a Normal distribution, a log-Normal one is assumed, removing the issue related to negative samples [54]. Specific details will be given later in this paper.

2.3.4 Constraints

After the perturbations are applied to the nuclear data, physical constraints must be ensured, like the redundancy and the normalisation. Therefore, data are further modified and additional levels of correlations are imposed on the random samples, potentially introducing correlations that were not originally present in the covariance matrix.

Redundancy

The data stored in the different MT sections often exhibit redundancy, meaning that certain cross-section data can be derived from others using summation rules. For instance, the total cross-section can be reconstructed by adding up all the individual (non-redundant) cross-sections or the total nubar can be reconstructed by summing the prompt and delayed contribution. In simpler terms, a cross-section (or in general a data) is considered redundant when it can be entirely calculated by combining other cross-sections/data in accordance with conservation laws and SANDY has been developed to automatically perform the task of reconstructing redundant cross-sections for each modified file. Therefore, redundancy is a constraint belonging to nubar and cross-sections, and others, and it will be taken into account in the implementation of the nubar perturbation in SANDY. Furthermore, if perturbations are available exclusively for a redundant cross-section and not for its individual components, the code extends the redundant cross-section perturbations also to its components and, in this way, the perturbation is considered and the redundancy constraint is respected.

Normalisation

Some nuclear data, such as the energy distributions, are represented by probability density functions that have the constraint to obey the normalisation condition represented in the Equation 2.27.

$$\boxed{\int_{-\infty}^{+\infty} f(x)dx = 1} \tag{2.27}$$

The zero-sum rule imposes that, if the normalisation had been previously integrated into the covariance matrix, the totals of the elements in each row (as well as in each column) would be precisely zero. SANDY is capable to normalise all the perturbed distribution for covariance matrices that do not comply to the zero-sum rule. In particular, the methodology followed to include the PFNS in the code will be treated in the subsection 3.2.2.

2.4 NDaST

Nuclear Data Sensitivity Tool (NDaST) (`https://www.oecd-nea.org/jcms/pl_32450/nuclear-data-sensitivity-tool-ndast`) is an openly accessible Java-based web tool created to conduct computations using sensitivity files for nuclear data of well-known benchmarks, and it was developed in 2015 under the auspices of the NEA [29]. NDaST is able to conduct two types of tasks:

- ✻ Assessing the effect of perturbations in nuclear data on computed criticality scenarios, which enables the identification of specific reactions' contribution to the results and facilitates the analysis of competing impacts.
- ✻ Evaluating the uncertainty in computed outcomes arising from the evaluated nuclear covariance data.

This tool helps nuclear data evaluators to rapidly analyse and test the impact of nuclear data and nuclear data covariance matrices perturbations on the final outputs, for an extensive range of criticality benchmarks from the ICSBEP and IRPHeP handbooks (see section 2.5). Unfortunately, NDaST contains important drawbacks that must be considered:

- it uses the sandwich rule for the uncertainty propagation and, more in particular, the sensitivity profiles are obtained using a first-order approximation method (section 2.1.1). Thus, higher-order effects are not taken into account and their contribution to the response could not be negligible, leading NDaST to give wrong results
- not all the sensitivities are included in the ICSBEP database (only 85 %)
- sensitivities are primarily given based on the SCALE 238 energy group structure, which may not be sufficiently suitable for specific system configurations and perturbations
- it is still under development concerning the energy-dependent nubar, energy and angular distributions.

2.4.1 Methodology

NDaST will be used subsequently to verify the UQ/propagation analysis performed by the stochastic method where the nuclear data samples have been produced by SANDY. In particular, it will be useful to verify the effects of all nuclides cross-sections and nubar perturbation on important benchmarks k_{eff}. However, since NDaST does not work with nubar yet, to compare the results the variance on the k_{eff} due to all input nuclides data will be subtracted by the variance due to

only the nubar (obviously for nuclides that can produce fission). Then, the latter will be compared with the variance on the k_{eff} due to all nuclides cross-sections, obtained using NDaST (in fact the related standard deviations). More details and the correspondent results will be shown in the section 4.3.

The software is basically made of 3 sections [29]:

Sensitivities: this section has been used to select the proper benchmarks to analyse. It is possible to select between two databases: Database for the International handbook of evaluated Criticality safety benchmark Experiments (DICE) [55] and International reactor physics handbook Database and Analysis Tool (IDAT) [56]. The interested plutonium benchmarks are present in the DICE database and consequently all the chosen benchmarks (section 2.5) can be selected, with their different models, codes and libraries to evaluate the sensitivities.

Perturbations: the uncertainty propagation can be carried out by simply choosing a fixed nuclear data perturbation and analysing the output uncertainty. However, this is not the goal of this paper and so this section has not been used.

Covariances: here, it is possible to select the covariance of multiple isotopes and nuclear reactions. The UQ/propagation will be performed based on the selected covariance data.

The steps are summarised in the Figure 2.4.

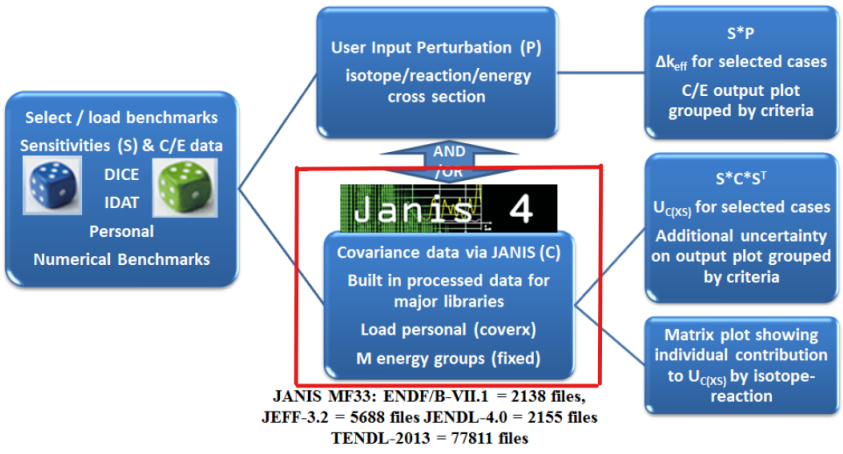


Figure 2.4: NDaST flow chart [9] with emphasis on the specific path followed

Therefore, once all the inputs have been selected, NDaST extracts the sensitivities of all the selected isotopes and their relative nuclear reactions as well as the related covariance matrices. Finally, it prepares them for the sandwich rule (Equation 2.19) and the latter is performed. These are the steps followed in this framework and, for more details about NDaST, the manual [29] can be consulted.

2.5 Benchmarks

Criticality safety organisations around the world frequently need to compare the outcomes generated by their computational methods with experimental data. Historically, this involved a challenging procedure of searching for experimental data related to criticality safety in journals, publications, or reports [27]. From the early years of the nuclear industry, numerous experiments covering criticality, subcriticality, shielding, radiation transport and fundamental physics have been conducted, all of which hold significance for criticality safety applications. A substantial portion of these experiments can serve as benchmarks for validating criticality safety calculation methods, while the remaining part needs to be reviewed, increasing the quality assurance degree and documenting them properly.

For the verification of the stochastic method, implemented with SANDY, different benchmarks have been exploited. In particular, the ^{239}Pu-based benchmarks, that come from the International Criticality Safety Benchmark Evaluation Project (ICSBEP), have been chosen for the uncertainty propagation and the reasons are clearly drawn up at the end of section 2.1.

Regarding benchmarks, everything started in 1992, when the USDOE launched the Criticality Safety Benchmark Evaluation Project (CSBEP) and it rapidly evolved into a global collaboration as scientists from various nations joined in. Therefore, later, in 1995, the ICSBEP was formally adopted as an initiative under the OECD NEA [27]. The OECD serves as a distinctive forum where 38 member nations collaborate to tackle the economic, social, and environmental challenges posed by globalisation. Additionally, the OECD plays a leading role in comprehending and assisting governments in addressing new developments and emerging issues, including corporate governance, the digital economy and the complexities associated with the ageing of the population. The NEA was established on the 1st February 1958 by OECD and it is made of 34 members [19]. It aims to support member nations in enhancing global collaboration to ensure safe, sustainable, and cost-effective utilisation of nuclear energy for peaceful purposes through scientific, technological and legal advancement. Moreover, it delivers authoritative assessments and fostering shared perspectives on critical nuclear energy policy matters to inform

government decisions and contribute to OECD's broader policy initiatives in energy and sustainable development [19]. The NEA specialises in various fields, including the safety and regulation of nuclear operations, management of radioactive waste, radiological protection, nuclear science, economic and technical evaluations related to the nuclear fuel cycle, nuclear law and liability and dissemination of public information. Additionally, the NEA Data Bank [57] offers participating nations nuclear data and computer program services.

ICSBEP handbook [27] comprises specifications for criticality safety benchmarks obtained from experiments conducted at nuclear critical facilities worldwide. These benchmark specifications serve as a resource for criticality safety engineers, aiding the verification of calculation methods for establishing the minimum subcritical margins during operations involving fissile materials, as well as determining criticality alarm requirements and their placement. Moreover, numerous specifications within this handbook can be valuable for testing nuclear data. ICSBEP main objective is to consolidate benchmark-experiment data related to criticality safety into a standardised format, making it convenient for criticality safety analysts to employ the data for verifying calculation methods and cross-sections and other data. However, it is clearly specified that "this work does not constitute a validation of codes or cross-section data. Furthermore, sample input listings may not have been verified and are not intended to be used directly for validation efforts" [27]. Nevertheless, they can be used as, at most, a verification tool to compare results using similar calculation methods and input data, that is the intended purpose. Other information about ICSBEP, such as its other purposes and acceptance and quality assurance criteria, can be found by consulting the reference [27].

ICSBEP is made of nine volumes, while the ones related to critical and subcritical benchmarks ranges from *volume* I to VII. The differentiation is based on the fissile material, its composition/physical state and the neutron energy region where it works. In particular, the benchmarks used for this thesis comes from the *volume* I, related to plutonium systems. More in detail, the plutonium benchmarks utilised belong to the Plutonium Metal Fast (PU-MET-FAST) and, therefore, they are plutonium fissile-material based, in metallic state and exploiting fast region spectrum. Here, the list of them with few details [27]:

- **PU-MET-FAST-001**, also called Jezebel, was a critical assembly constructed and operated at Los Alamos Scientific Laboratory (LASL). Jezebel was an unreflected bare δ-phase ^{239}Pu critical assembly, nearly spherical in shape and was composed by Pu and 1.02 wt.% Ga (gallium-69 and gallium-71). The plutonium composition was 95.2 at.%^{239}Pu, 4.5 at.% ^{240}Pu and 0.3 at.% ^{241}Pu. Criticality calculations were made, but it was used also to compute the critical

mass of the plutonium and the result was 16.2 kg at a density of 15.8 $\mathrm{g\,cm^{-3}}$, even if the reference to the plutonium pure mass or Pu-alloy (with gallium) was not specified. For safety considerations, the nearly spherical mass was manufactured in four primary components, each with approximately equal mass, and then assembled to create a three-part division for operational safety. The assembly was meticulously designed to be easily reproducible and to minimise reflection while preserving experimental versatility. Indeed, there were mass adjustment plugs (common to all PU-MET-FAST) which tuned the core mass in the central hole (called "glory hole") and the control rod could run through the whole core, making the system subcritical or critical (Figure 2.5).

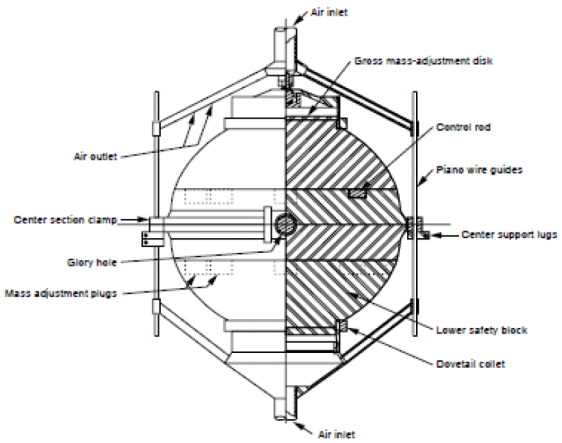


Figure 2.5: Jezebel assembly scheme [27]

✻ **PU-MET-FAST-002**, called also Jezebel dirty, was similar to Jezebel in dimension and configuration, but the composition was different with an higher percentage of ^{240}Pu (20.4 at.%) as well as the presence of ^{241}Pu (3.1 at.%) and ^{242}Pu(0.4 at.%). Operated for many years, a large number of reactivity worth calculations were made until its disassembling, when the different part were used for other purposes.

✻ **PU-MET-FAST-005** was a plutonium sphere reflected by tungsten, also from LASL. The experiment was conducted utilising the Planet universal assembly apparatus and the core was divided in two hemispheres of δ-phase plutonium and a cavity in their centre while tungsten-based hemishells were built to enclose the plutonium core. From Figure 2.6, the upper portion of the plutonium assembly, comprising the core and reflector, was supported by a stainless steel diaphragm. The lower part of the assembly was raised using a

hydraulic lift, with the portion of the hydraulic lift in contact with the lower hemisphere being a cylindrical, hollow aluminium tube. The elevation of the lift was raised step by step and the multiplication factor was measured at each interval until reaching closure. The sphere was made of 99.0 wt.% plutonium and 1 wt.% gallium. The plutonium composition was 94.79 at.% ^{239}Pu, 4.9 at.% ^{240}Pu and 0.31 at.% ^{241}Pu while the reflector composition was 91.3 wt.% W, 5.5 wt.% Ni, 2.5 wt.% Cu and 0.7 wt.% Zr. The massive presence of the tungsten, in particular ^{184}W, will be crucial for the verification of the results (subsection 4.3.1) and it will show up that the source of error comes from the stochastic sampling with SANDY.

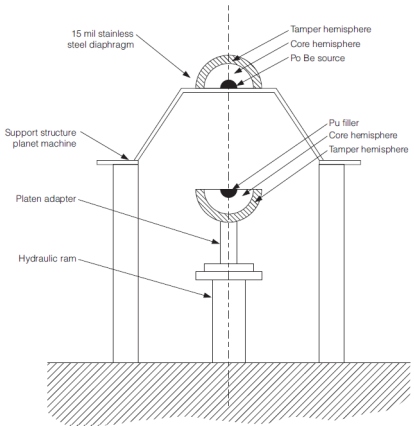


Figure 2.6: PU-MET-FAST-005 assembly scheme (PLANET) [27]

✻ **PU-MET-FAST-006** core was made of δ-phase plutonium metal alloy, surrounded by a natural uranium reflector, and it was divided in two hemispheres. The reflector consists of a fixed hemishell and two movable quadrants that together form the second hemishell. The core halves were positioned on a conventional uranium pedestal support that moves along a track and similarly, the two movable quadrants had their own designated tracks. The assembly process involved placing the core within the stationary reflector hemishell and subsequently positioning the individual quadrants around the core, completing the assembly (Figure 2.7). The approach to reaching criticality was monitored by analysing the position of the control rod, and the rapid dismantling was achieved by swiftly extracting the two standard uranium components. Moreover, the core composition was 93.8 wt.% ^{239}Pu, 4.8 wt.% ^{240}Pu, 0.3 wt.% ^{241}Pu and 1.1 wt.% Ga.

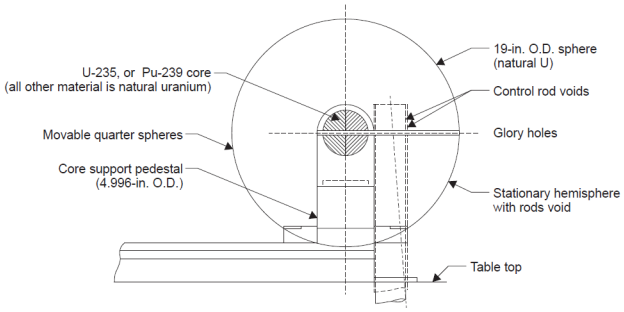


Figure 2.7: PU-MET-FAST-006 assembly scheme (FLATTOP), elevation view [27]

- **PU-MET-FAST-008** was a δ-phase plutonium spherical mass closely surrounded by an annular cylindrical reflector (closed up and down thanks to safety blocks) made by thorium. The main purpose of this experiment was to investigate the neutronic characteristics of thorium, with the ultimate goal of advancing the development of a potential fast breeder reactor based on uranium-233 (^{233}U). The experiment was executed with the Thor assembly machine, adapted from the Planet one. The adaptations primarily involved adjustments to the control rod drive and safety mechanisms, while the plutonium core consisted of three significant components: an upper polar cap, a lower polar cap, and a central section (with a glory hole). Whereas the upper polar cap and central section remained immobile, the lower polar cap was positioned atop a pneumatic lift to put them together. Moreover, reactivity was fine-tuned by employing filler pieces for the glory hole. In case of an emergency shutdown (scram), swift disassembly was achieved by retracting both the upper and lower safety blocks along with the control rod. The core composition was 93.59 wt.% ^{239}Pu, 5.10 wt.% ^{240}Pu, 0.3 wt.% ^{241}Pu and 1.01 wt.% Ga and the reflector was made of 100 wt.% ^{232}Th.

- **PU-MET-FAST-009** was a plutonium spherical mass reflected by type 2014 aluminium reflector. The experiment was carried out utilising the Comet universal assembly machine (from the Planet one): two hemispheres were crafted from a δ-phase plutonium alloy and the upper hemisphere featured a cylindrical central source cavity, designed to accommodate either a fission source or a closely fitting piece of plutonium. Concentric hemispherical shells, made of 2014 aluminium, were fabricated to encapsulate the plutonium hemispheres, making up the reflector. The upper portion of the plutonium core was positioned atop a stainless-steel diaphragm, while the lower part of the assembly, which included both the reflector and the core, was affixed to a hydraulic cylindrical lift. The approach to criticality was monitored

by consulting a neutron detector (inside the source cavity) count rate, as the separation distance between the upper and lower halves of the assembly decreased, thanks to the lift. In contrast, rapid disassembly was achieved by lowering the hydraulic lift, causing the lower portion of the assembly to drop, when limit values on the count rate were reached. The plutonium composition is not far from the previous benchmark while the reflector was made of 93.4 wt.% Al, 4.4 wt.% Cu, 0.4 wt.% Si, 0.8 wt.% Mn and 0.5 wt.% Mg.

- **PU-MET-FAST-010** was a δ-phase plutonium sphere reflected by natural uranium. This benchmark is a part of a series, to which also PU-MET-FAST-005 belongs. Therefore, the experiment configuration is exactly the same of PU-MET-FAST-005 as well as the plutonium composition. The only difference was reflector composition, here made of natural uranium (not tungsten).

- **PU-MET-FAST-011**: the experiment was conducted with a δ-phase plutonium sphere, meticulously crafted to achieve a high level of purity and density, surrounded by a water reflector. As the Figure 2.8 shows, the plutonium sphere was positioned on a Lucite stand, supported by three legs, designed with a circular seat featuring a central hole. The stand was placed within a cylindrical aluminium-run tank, connected to the filling tank through a flexible hose. The water level inside the run tank was increased by raising the filling tank using a hydraulic lift and to lower the water level, the operator could either use the drain valve or low the filling tank with the hydraulic lift. Thanks to neutrons detectors, the criticality was monitored and measured as the water height in the run tank changed. The plutonium composition was 94.5 at.% ^{239}Pu, 5.18 at.% ^{240}Pu, 0.3 at.% ^{241}Pu and 0.2 at.% ^{242}Pu.

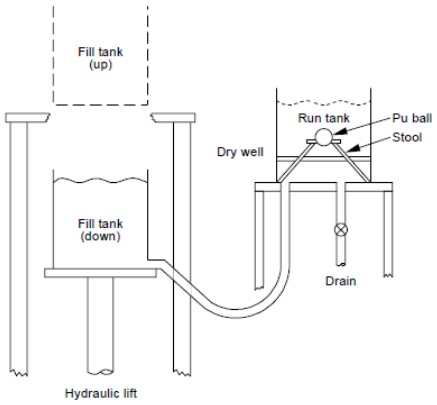


Figure 2.8: PU-MET-FAST-011 assembly scheme [27]

- **PU-MET-FAST-018** was a plutonium sphere surrounded by beryllium

reflector. It is part of the same experiment of PU-MET-FAST-005 and PU-MET-FAST-010. Again, the experimental configuration and the plutonium composition was the same. The difference was the reflector, composed by 98.0 wt.% beryllium with 2.0 wt.% oxygen.

✻ **PU-MET-FAST-019** was a plutonium sphere reflected by beryllium, built in 1983 in the All-Russian Scientific Research Institute of Technical Physics (VNIITF) Criticality Test Facility (CTF). This experiment was employed to confirm the accuracy of neutron data related to beryllium. The assembly (Figure 2.9) comprised a spherical structure divided into two sections with a gap in between and its description involved two additional parameters: the critical separation $\mathbf{h_{cr}}$, that is the distance between the centres of the opposing hemispheres, and the separation radius $\mathbf{R_{sep}}$, i.e. the radius of the upper hemisphere portion that remains connected to the lower mass that can move thanks to a shaft linked to a piston. The latter described how the assembly was separated. The upper part of the assembly was upheld by a steel annular diaphragm that laid on a stand made by two rings locked by three rods. Thus, the top assembly was not movable. The criticality was measured by detectors, as a function of the gap width **h**, reduced by the lift. The plutonium and beryllium composed matching hemispherical shells, and the plutonium ones had polar cylindrical holes to be filled with the same plutonium plugs. Different composition for the plutonium were used while the reflector was made of beryllium containing small percentages of impurities such as C, O and Fe.

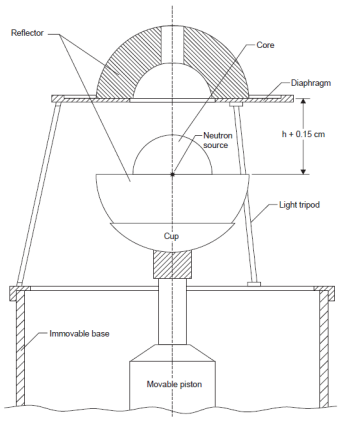


Figure 2.9: PU-MET-FAST-019 assembly scheme [27]

✻ **PU-MET-FAST-020** was a plutonium sphere reflected by depleted uranium. The experimental configuration and the plutonium composition were the same

of PU-MET-FAST-019, located at the same place, and the only difference was made by the reflector. The latter was depleted uranium with 99.5 wt.% ^{238}U and 0.5±0.05 wt.% ^{235}U. This experiment was conducted to verify the neutron data regarding uranium isotopes.

✻ **PU-MET-FAST-021** was made of δ-phase plutonium cylinders surrounded by beryllium or beryllium oxide that constitute the reflector. Therefore, two different configurations were constructed in order to compare the impact of the reflective materials and neutron data validation was performed. In this framework, the first configuration has been chosen. Figure 2.10 shows the configuration where the critical assembly featured a uniform cylindrical core composed of 95 at.% ^{239}Pu and was equipped with beryllium end reflectors. Two identical and symmetric sections were present, separated by a gap, with a mobile lower portion and a fixed upper portion. The criticality was reached by moving the lower part close to the upper (i.e. reducing the gap distance) and detectors were used to measure it. The plutonium composition was 95.21 at.% ^{239}Pu, 4.59 at.% ^{240}Pu, 0.2 at.% ^{241}Pu while the reflector was beryllium with 1.51 wt.% oxygen and other impurities with very low content.

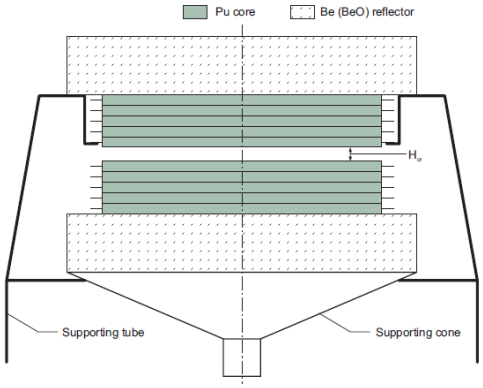
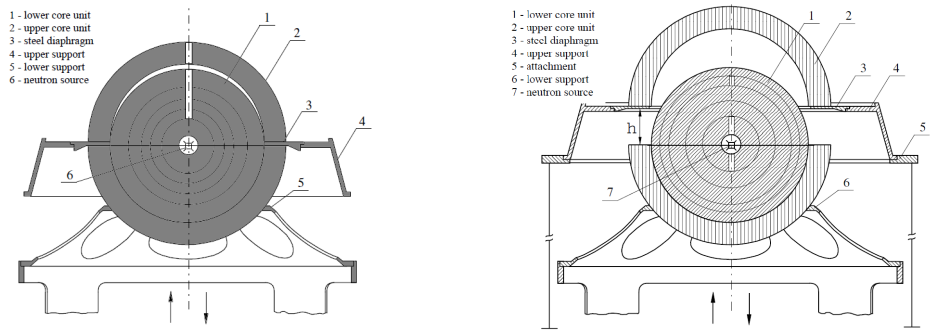


Figure 2.10: PU-MET-FAST-021 assembly scheme [27]

✻ **PU-MET-FAST-022** was a bare metal plutonium ^{239}Pu(δ, 98%) spherical core, and the experiments were performed at the All-Russian Scientific Research Institute of Experimental Physics (VNIIEF) CTF. The assembly (Figure 2.11) consisted of six spherical layers, each made of different composition, and each was constructed using two hemispherical components that featured cylindrical pole holes. This assembly comprised two distinct units: the upper unit consisted of a single hemispherical piece from layer 6, which was positioned on top of a diaphragm, and the lower unit (made of the remaining part). In the standard test assembly, the pole holes of layer 1 were closed with

specialised stoppers made of core material. Moreover, the core presented a central cavity needed for the neutron source. The criticality was reached and always measured with the same procedure, so making the parts closer and using detectors. The average bare plutonium composition was 98.2 wt.% ^{239}Pu and 1.8 wt.% ^{240}Pu while the core contained also gallium (wt.2%).



(a) PU-MET-FAST-022 **(b)** General scheme

Figure 2.11: PU-MET-FAST-022 assembly scheme and general scheme for the VNIIEF CTF [27]

- **PU-MET-FAST-023** was a bare δ-phase metal plutonium sphere surrounded by a graphite reflector. The configuration was similar to the one of PU-MET-FAST-022 (indeed both were performed at the VNIIEF CTF) but here a reflector was present and also the layers characteristics were different. In this case, the plutonium upper unit (the unit 2 in Figure 2.11) and the 6th layer in the lower unit were replaced by a graphite hemispherical piece of 2.35 cm. Moreover, also the core composition was similar as the previous benchmark while the reflector was made of graphite (carbon), where the impurities (such as Al, Mg, B, Fe, Mn, Cu, Ti and Si) concentrations were very low.

- **PU-MET-FAST-024** was similar to PU-MET-FAST-023 (same configuration Figure 2.11, same facility and similar plutonium composition) but in this case the graphite reflector was substituted by a 1.55-cm polyethylene reflector, having density of 0.904 g cm^{-3}.

- **PU-MET-FAST-025** was part of the PU-MET-FAST-024 experiments but in this case the polyethylene was replaced by steel to compose the reflector and this made the only difference with the previous benchmark unless the layers properties (also same thickness). The steel contained, other than iron, 0.3 wt.% C, 0.4 wt.% Mn, 0.1 wt.% Si, 0.3 wt.% Cr, 0.3 wt.% Ni and 0.3 wt.% Cu with a density of 7.505 g cm^{-3}.

- **PU-MET-FAST-026** was similar to PU-MET-FAST-025, but here the upper unit (see Figure 2.11) and part of the lower unit were replaced by two concentric spherical layers with a combined thickness measuring 11.9 cm. All the other parameters remained the same as PU-MET-FAST-025 with exceptions for the layers characteristics.

- **PU-MET-FAST-028** was a metal plutonium ^{239}Pu(δ, 89%) spherical core, surrounded by a steel reflector, and the experiments were performed at the VNIIEF CTF. Indeed, it was similar to the previous 4 benchmarks (layers features were different), with a different plutonium concentration but with the same exactly configuration, represented in Figure 2.11. The steel reflector consisted of two spherical layers, with their interface positioned at a radius of 9.15 cm. The overall thickness of the reflector was equal 19.65 cm and its composition was the same as PU-MET-FAST-025. However, the plutonium composition changed with 89.7 wt.% ^{239}Pu, 9.3 wt.% ^{240}Pu and 1.0 wt.% ^{241}Pu with the usual gallium content in the core.

- **PU-MET-FAST-029** was a bare metal plutonium spherical core and was always based on the configuration depicted in Figure 2.11, with different layers aspects. The noticeable difference is the plutonium phase (α) and ^{239}Pu content (88%). Indeed, in all the previous examples, the plutonium was in its δ-phase. Instead of 6, 5 spherical layers were present: the three inner and the two outer had different thickness. Moreover, the upper unit included two outer hemispherical layers while the other parts were all comprised in the lower, movable unit. The average plutonium composition changed with 88.6 wt.% ^{239}Pu, 9.9 wt.% ^{240}Pu and 1.6 wt.% ^{241}Pu other than the usual gallium content in the core.

- **PU-MET-FAST-031** was also performed at VNIIEF CTF and, thus, the experiment was carried out in the same way as before. Again, the configuration was similar to Figure 2.11 with different layers customs. ^{239}Pu(α, 88%) was used as fissile material, while the reflector was made of polyethylene composed by two spherical layers. The upper unit included a single hemispherical part of polyethylene and the remained parts composed the lower unit. The average plutonium composition was slightly different from PU-MET-FAST-029, with 88.9 wt.% ^{239}Pu, 9.7 wt.% ^{240}Pu and 1.4 wt.% ^{241}Pu other than the usual gallium content in the core.

- **PU-MET-FAST-032** was a α-phase plutonium sphere, surrounded by 4.49-cm steel reflector. The experiments were performed at the VNIIEF CTF and, hence, all the consideration done before can be applied to this case. The average plutonium composition was equal to the one in PU-MET-FAST-031 while the steel composition was equal to PU-MET-FAST-031.

✻ **PU-MET-FAST-033** was a cylindrical assembly of metallic plutonium surrounded by graphite. The experiments were performed thanks to the Zero Power Physics Reactor (ZPPR) [58] fast CTF at the Argonne National Laboratory-West (ANL-W), built to acquire the neutron physics data required for the design of fast breeder reactors. The assemblies were equipped with graphite reflector and varying core fuel compositions containing varying combinations of plutonium, uranium and zirconium to form a mock-up Pu/U/zirconium fuel, in metallic form, plate-type cells. The chosen configuration is with depleted uranium (0.21 at.% ^{235}U), plutonium (94.0 wt.% ^{239}Pu, 5.8 wt.% ^{240}Pu and 0.2 wt.% ^{241}Pu) and zirconium. As can be seen in Figure 2.12, the geometry/configuration was very complex and, thus, a simplified one was adopted in the simulations. In order to reach criticality, which was monitored through detectors, the radial reflector thickness was adjusted (thus, also the radial reflector and the room-return shield distance) until the goal was reached.

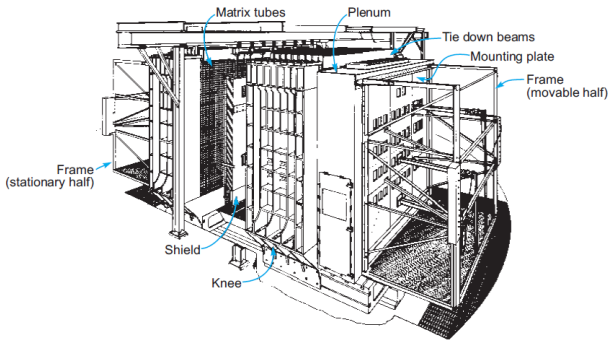


Figure 2.12: PU-MET-FAST-033 assembly scheme [27]

✻ **PU-MET-FAST-041** was a metal plutonium ^{239}Pu(α, 88%) spherical core, surrounded by a depleted-uranium reflector. Still the experiments were conducted in the VNIIEF CTF but with different configuration (Figure 2.13). The sphere was made by many layers, and each layer (for both core and reflector) was constituted by two hemispherical parts. Moreover, both the core and the reflector layers contained cylindrical pole holes to be plugged with the same material of the layer during the measurements. The assembly consisted of two distinct components: the upper unit, which consisted of a single hemispherical part of depleted uranium of 2 cm, suspended using a slender cable made of depleted uranium as well, and the lower movable unit which encompassed the remaining components. Criticality was monitored as the lower part was made closer to the upper. The average plutonium composition was 88.3 wt.% ^{239}Pu, 10 wt.% ^{240}Pu and 1.7 wt.% ^{241}Pu while the reflector was 99 wt.% ^{238}U and

0.44 wt.% ^{235}U.

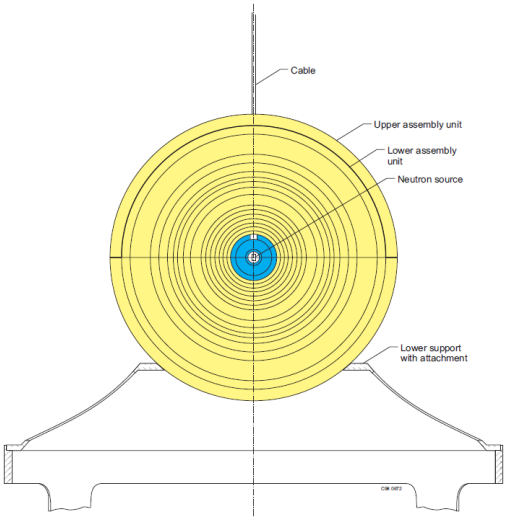


Figure 2.13: PU-MET-FAST-041 assembly scheme [27]

✻ **PU-MET-FAST-044** was a nearly spherical core made of a plutonium alloy, encased by a closely fitting inner tamper shell and an outer shell made of polyethylene for reflection. The main objective of these experiments was to determine the critical thickness of polyethylene reflectors required to reach criticality with five different tampers (beryllium, graphite, aluminium, iron, and molybdenum) surrounding the plutonium-alloy core. For the thesis purpose, the configuration with molybdenum tamper has been studied. The experiments were performed at LASL (now know as LANL) and the Comet machine was adopted. Figure 2.14 shows the plutonium-alloy core made of an upper and a lower semi-sphere and a central section. Moreover, also here a glory hole was present, where plutonium was laid. For each tamper material, the criticality was reached by increasing the polyethylene thickness by adding matching concentric layers. The average plutonium composition was 94.54 wt.% ^{239}Pu, 5.11 wt.% ^{240}Pu and 0.35 wt.% ^{241}Pu with a gallium percentage of 0.95 wt.% in the plutonium alloy. Furthermore, the molybdenum composition was 14.53 at.% ^{92}Mo, 9.15 wt.% ^{94}Mo, 15.84 wt.% ^{95}Mo, 1.667 at.% ^{96}Mo, 9.6 wt.% ^{97}Mo, 24.39 wt.% ^{98}Mo and 9.82 wt.% ^{100}Mo.

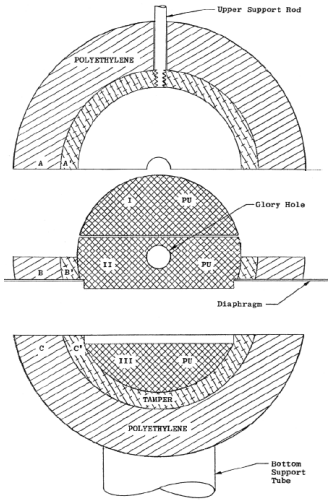


Figure 2.14: PU-MET-FAST-044 plutonium-alloy core, surrounded by tamper and polyethylene reflector, scheme [27]

Furthermore, all the benchmark experiments were performed at a temperature of about 300 K. Moreover, some materials (e.g. the structural) has not been mentioned, but they have been considered in the simulations and always the simplified models have been chosen to run the calculations. For all the deepest details about the single benchmarks, the ICSBEP handbook is available [27].

Chapter 3

Methodology

In this chapter, the methodology to perturb nuclear data with SANDY and to process the Serpent output files will be detailed. In other words, the uncertainty propagation due to the uncertainty of the nubar (or other input nuclear data) on nuclear parameters will be explained. The uncertainty propagation has been performed for the plutonium benchmarks listed in the section 2.5 and, in particular, the effective multiplication factor (k_{eff}) has been chosen as response function (computed through the MC code Serpent), since the nubar mainly affects it (i.e. high sensitivity). Thus, the uncertainty on the output is not negligible and the effects can be investigated in depth. To sum up, n samples of the nubar are produced (together with their ACE files) and n simulations (each one with a different sample) are run, on a specific benchmark.

3.1 Serpent2 tool

For the criticality calculations, Serpent2 [22] has been used, as well as the for the sensitivity vectors (sensitivity analysis) computation for the other methods implemented. Serpent2 is a versatile three-dimensional code for the continuous-energy transport of neutrons and photons, which has been under development at the Technical Research Centre of Finland (VTT) since 2004, and its distribution is managed by VTT, while it is accessible through two key data centres: the OECD/Nuclear Energy Agency (NEA) Data Bank and the Radiation Safety Information Computational Center (RSICC) in the United States. For further details on the distribution of Serpent, please refer to the users' sub-page (`http://montecarlo.vtt.fi/`) while the manual is available in the reference [22].

3.2 Samples generation methodology

Firstly, the chosen nuclear data must be perturbed and each of it has a proper Python [52] method for different reasons: they depend on different variables (energy, input energy, output energy etc.), they are processed and stored differently (as their covariances) and other. Consequently, to obtain the perturbed nuclear data, different class and methods has been implemented. However, the general approach is the same:

1. The nuclear data class is generated. It is a Python dataframe with the main independent variable as index (e.g. the energy range) and the others as column multi-indexes (e.g. MAT, MF and MT number). The nuclear data are taken usually from the ENDF-6 file, saved in a `ENDF6` instance, and they are the best estimate, in continuous energy, that comes from the evaluation process. The `ENDF6` instance is extracted giving the library, the data type (here xs, neutron nuclear data) and the zam. Moreover, the zam package, in SANDY, can be used to obtain the zam knowing the nuclide symbol and its mass number (e.g. Pu239).

2. The covariance matrix is extracted from the ERRORR file and the information is stored in a dataframe where the index and the columns are identical multi-index(i.e. the parameters). Indeed, from subsection 1.4.1, the covariance matrix is squared. For the ERRORR file and the covariance matrix, the respectively `ERRORR` and `Cov` class has been created with its energy groups.

3. From the covariance matrix, the perturbation coefficients are generated and stored in a `Samples` instance. The relative dataframe has a multi-index index where the last level corresponds to the energy groups while the other two levels corresponds to the MAT and MT numbers. The sample numbers are located along the columns.

4. Finally, the perturbation coefficients are applied to the nuclear data instance, creating perturbed data according to their uncertainty stored in the covariance matrix. Then, it is possible to create the perturbed ACE files to be run in the Serpent code.

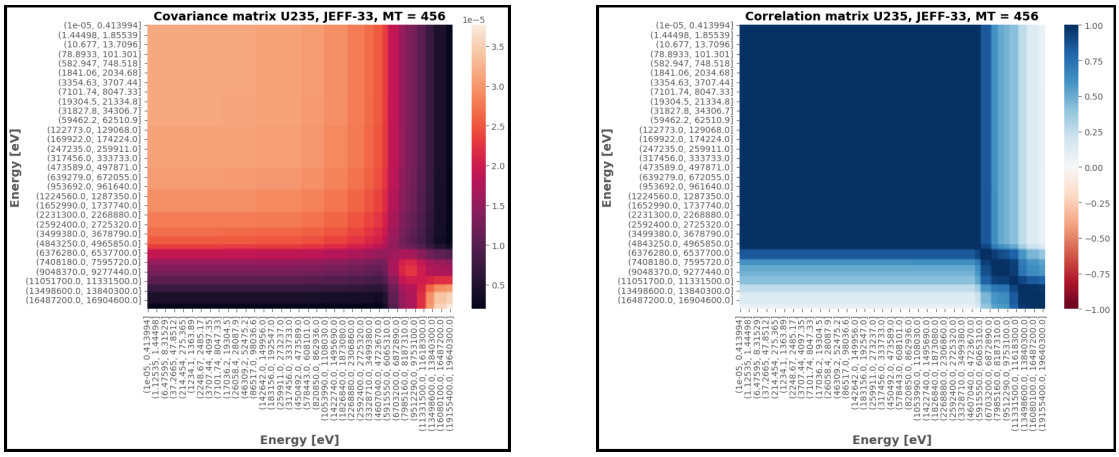
The single perturbation process, for different nuclear data, is implemented and, then, it can be recalled directly from the `ENDF6` instance through the methods `get_perturbations` (section B.1) and `apply_perturbations` (section B.2) to respectively obtain the perturbation coefficients and the perturbed nuclear data. `apply_perturbations` is connected to the `endf6_perturb_worker` function. During the sampling process implementation, the `Samples` instance will be slightly modified in a way that each continuous energy value of the nuclear data instance

has its perturbation coefficients of the related energy group and, if for a continuous energy value no energy group value corresponds (on the perturbation coefficients energy groups), the perturbation coefficient is set to 1. This is, basically, the implementation of the `sampling` method, used to apply the perturbations. Finally, by setting the variable `to_ace` and `to_file`, according to the keywords arguments (kwargs) they belong, to the value `True`, after the perturbation process, SANDY returns the ACE file needed for the simulations, using already implemented methods (e.g. `get_ace`).

3.2.1 Average Fission Neutron multiplicity

Since the average fission neutron multiplicity (nubar) is structured similarly to the cross-sections (they both depend only on the neutron energies), the class `Xs`, already present in SANDY, has been slightly modified in order to work properly with the nubar as well. Indeed, during its instance creation, only the cross-section information were filtered (`endf6.filter_by`). Then the MT numbers for prompt, delayed and total nubar (see Table A.2) have been filtered. Before performing this, it is important to verify that the nubar values are all got by linear interpolation from the ENDF-6, making it suitable for the correct instance creation since eventually missed values of specific energies are automatic obtained by interpolation in SANDY and since linear interpolation is exploited after the perturbations. The ERRORR instance is obtained by filtering "errorr31" (because 31 is the MF number for the nubar) in the dict returned by applying the `get_errorr` method to the `ENDF6` instance, already implemented in SANDY. Subsequently, since the ERRORR file is standard, the format is equal to the one of the cross-sections and so the covariance matrices and the corresponding perturbation coefficients are generated with the same method of the cross-sections (giving a number of samples). With the perturbation coefficients, properly stored in a `Samples` instance in order to process the data with the already existing functions in SANDY, the method `_perturb` can be applied to the `Xs` instance, giving a generator with whom the nubar perturbed data are finally constructed and stored back in the `ENDF6` instance. However, before, the perturbed nubar data must accomplish with the redundancy constraints (see subsection 2.3.4), performed by the method `reconstruct_sum`.

Once the code has been updated, its correctness must be verified. First, 1000 samples of ^{235}U prompt nubar have been produced, according to the covariance matrix (Figure 3.1). The latter shows also the correlation matrix, where the correlation coefficients are always positive: the nubar, at any energy, increases with the neutron energy itself and so the correlation between the different energy values is positive.



(a) Covariance matrix **(b)** Correlation matrix

Figure 3.1: Covariance and correlation matrix of the ^{235}U prompt nubar from JEFF-33 library

Then, the mean values have been compared to the best estimates stored in the ENDF-6 file. For example, in Figure 3.2 is shown the distribution of the samples, with its mean, and the best estimate, for a specific energy value (1.5 MeV). It can be seen that the best estimate and the mean value of the distribution correspond. Moreover, the distribution is "almost" Normal because, as the Central Limit Theory (CLT) explains, to be Normal the number of samples must approach the infinity. Furthermore, a Kolmogorov–Smirnov (KS) [59] test has been performed (see subsection 3.3.2), giving a p-value of 0.926, higher than the threshold 0f 0.05, and so the distribution follows the Normal distribution.

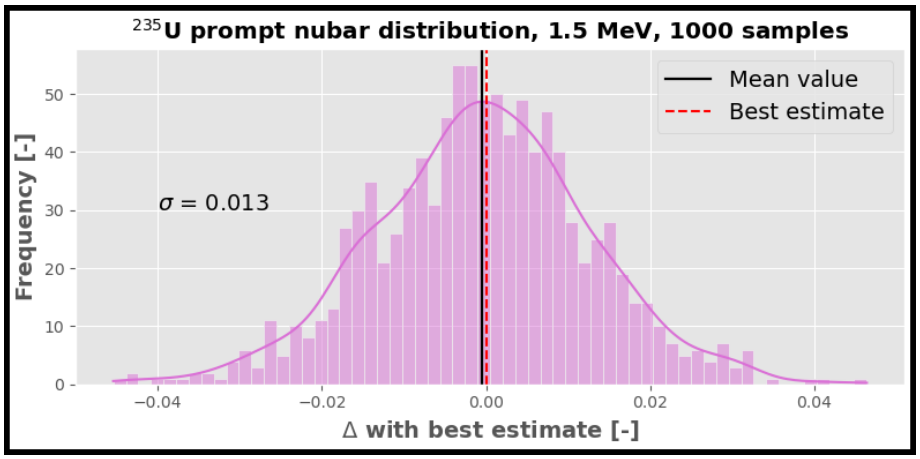


Figure 3.2: ^{235}U prompt nubar 1000 samples distribution, at 1.5 MeV

In addition, in the Figure 3.3, the best estimate (i.e. the value stored in the ENDF-6 file) has been compared with the output resulting from the sampling, with its mean and uncertainty 1σ[1]. Also, the Figure 3.4, shows the standard deviation of the perturbed data compared to the value given by the matrix as well as the standard deviation of the sample mean. The latter decreases as $1/\sqrt{N}$, confirming the CLT theorem. The sample standard deviation is practically always overestimated, and it tends to converge towards the matrix value with the increase of the number of samples. This overestimation propagates towards the response function, and in the section 4.1 it will be demonstrated and deepened.

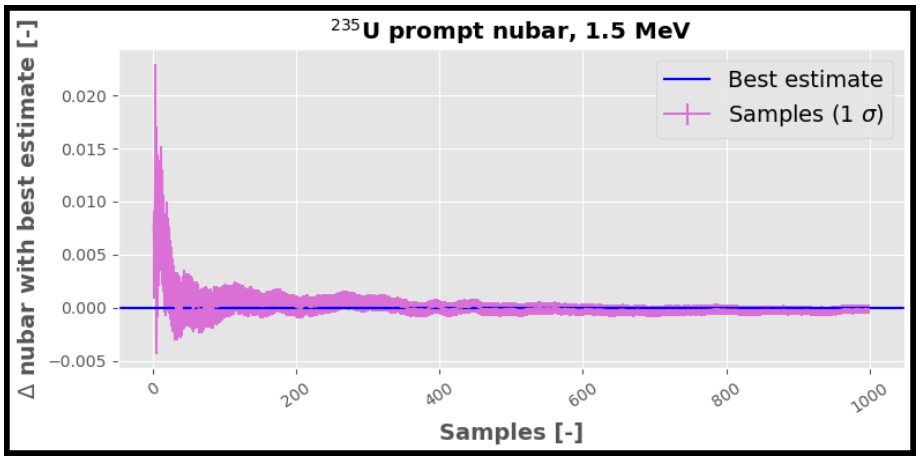


Figure 3.3: ^{235}U prompt nubar samples convergence, at 1.5 MeV

[1] 1 σ means there is a probability of 68 % that the real value falls in the given interval

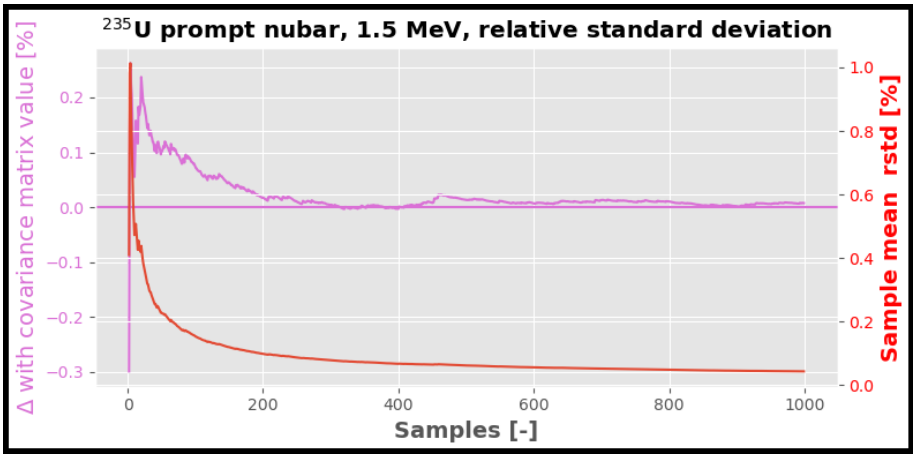


Figure 3.4: ^{235}U prompt nubar samples standard deviation and sample mean standard deviation, as function of the number of samples

Figure 3.5 shows the relative standard deviations of the ^{235}U prompt nubar, as function of the energy, stored in the covariance matrix and the ones resulting from 1000 samples (perturbation coefficients), generated with SANDY. The maximum relative difference registered is about 2.7 % (maybe due to numerical computation errors or statistical fluctuations), only for restricted energy values, and the SANDY sampling can be considered verified.

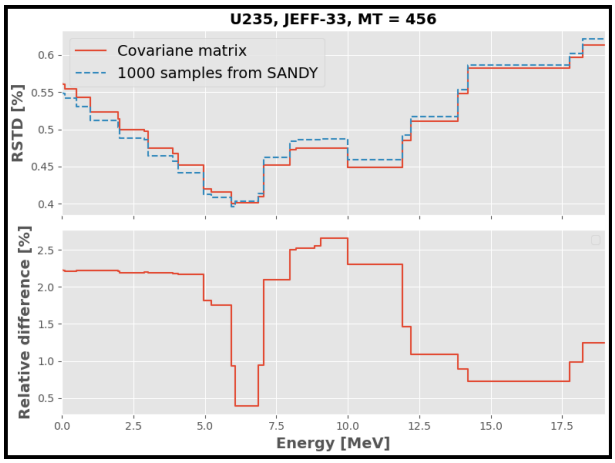


Figure 3.5: Difference between the relative standard deviations of the ^{235}U prompt nubar stored in the covariance matrix and the ones resulting from 1000 samples, as function of the energy

3.2.2 Energy distributions: Prompt Fission Neutron Spectra

Provided that the following modifications has not been verified (just a small beginning for the next steps) for the energy distributions, in particular the Prompt Fission Neutron Spectra (PFNS), a different approach has been conducted because such nuclear data depends on two variables: the input energy (i.e. the energy of the input particle) and the output energy (i.e. the energy of the outlet particle). In other words, the energy distribution of the emitted fission neutron depends also on the initial neutron energy. For these reasons, the class `Edistr` has been built: its dataframe contains the output energy on the index and the MAT, MT, K and input energy on the different levels of the multi-index columns. The K number is related to the type of neutron generated and, as first study, only the values 0 (i.e. prompt neutron) has been added. Moreover, a small part of SANDY has been modified to process the energy distribution with ERRORR and extract the covariance matrix. In this case, there are different covariance matrices for different input energies and to deal with two approaches are possible:

1. Since the cross-correlation, between the different input energy matrices, is very small, a unique covariance matrix can be chosen (at a specific input energy). However, each time, the assumption of zero cross-correlation must be justified.

2. The different covariance matrices are considered, but in this case the different distributions could not agree each other (e.g. the distributions are not normalised). Therefore, it must be verified that the cross-correlation is considered during the sampling.

For this initial step, the method 2 has been chosen and `get_perturbations` has been modified (subsection B.3.3). Thus, for each input energy, different perturbation coefficients, for the correspondent output energies, has been created and applied with a method similar to the `Xs` class. Indeed, the process has been implemented in a way that the `Cov` class would be extended for the energy distributions. However, in this case, to produce back the `ENDF6` instance, the function `write_mf5` (subsection B.3.2) and the method `to_endf6` have been added (subsection B.3.1).

Being probability density functions, the constraint applied after the sampling, is the normalisation (Equation 2.27), using the `normalize` method. Finally, its application in the `get_perturbations` and `apply_perturbations` (methods of the class `ENDF6`), has been performed. It is important to point out that all these steps, related to the PFNS, are not already in the official version of SANDY, because they must be updated, improved and verified, but it is a good starting point for the next. In the section B.3, are shown all the scripts implemented.

Figure 3.6 shows the best estimate and a perturbed sample (with SANDY) of the ^{235}U PFNS, according to the related covariance matrices (e.g. Figure 3.7). The peak corresponds to around 2 MeV, as it should be. Furthermore, the perturbed data is normalised, but it cannot be seen easily because the probability decrease is not concentrated as the increase at around 2 MeV. Moreover, the correlation matrix (Figure 3.7) shows also negative values: indeed, since the normalisation constraints must be verified, if the probability, at a certain energy, increases, a correspondent decrease in another energy values must follow.

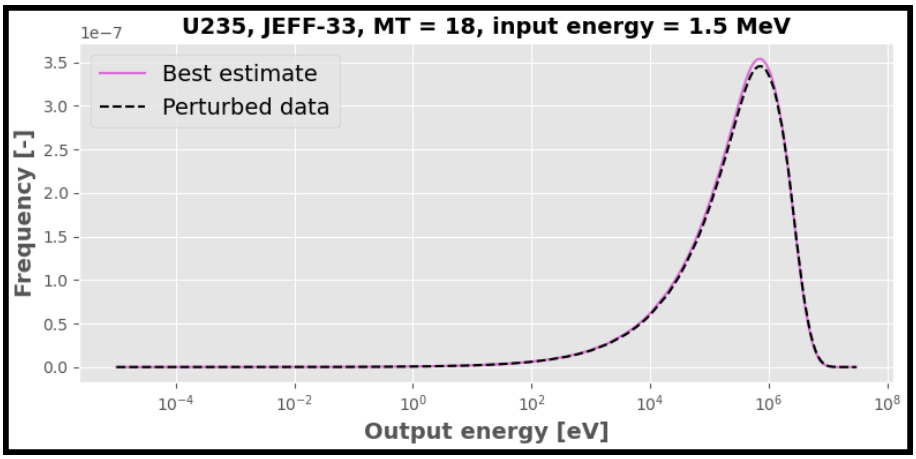
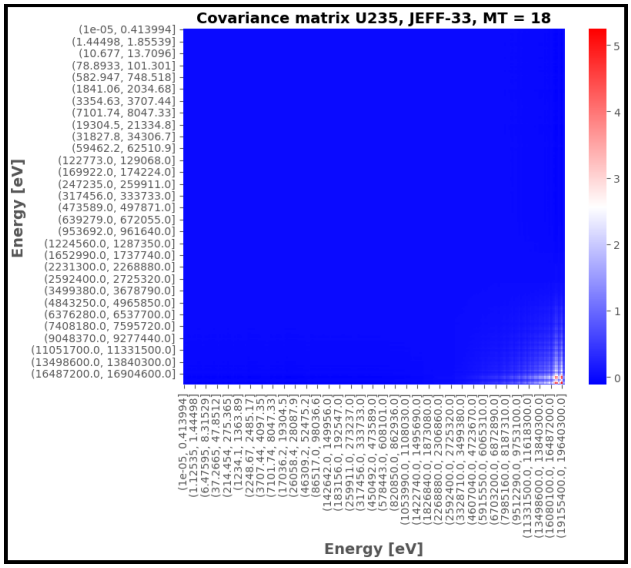
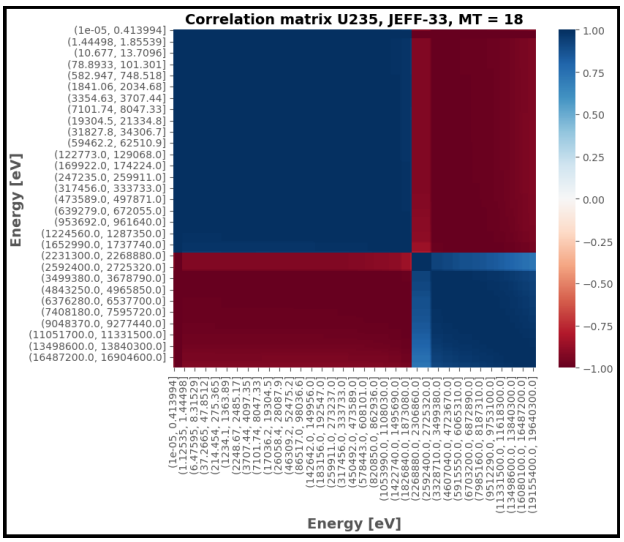


Figure 3.6: Best estimate and one sample, from the covariance matrix of 1.5 MeV input energy, of ^{235}U PFNS



(a) Relative covariance matrix



(b) Correlation matrix

Figure 3.7: Relative covariance and correlation matrix of the ^{235}U PFNS, at input energy of 1.5 MeV

In the Figure 3.8, it is clear that the sampling has not been performed correctly with SANDY and it has a precise reason: SANDY does not properly work with nuclear data characterised by very high standard deviations (see subsection 2.3.3), if Normal distribution is assumed. The PFNS shows very high uncertainties at high output energies (overcoming the 150 %), where SANDY fails.

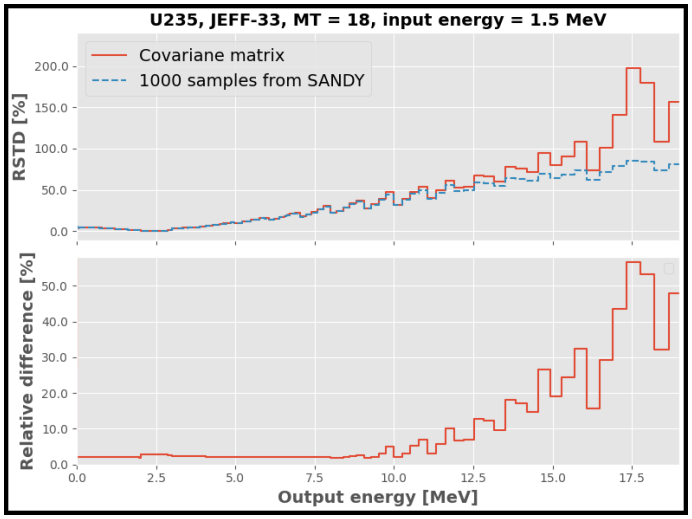


Figure 3.8: Difference between 1000 samples (perturbation coefficients generated with SANDY and assuming Normal distribution) and the covariance matrix (at input energy of 1.5 MeV) relative standard deviations of the ^{235}U PFNS, as function of the output energies

A solution can be found by assuming a log-Normal distribution and Figure 3.9 is the proof.

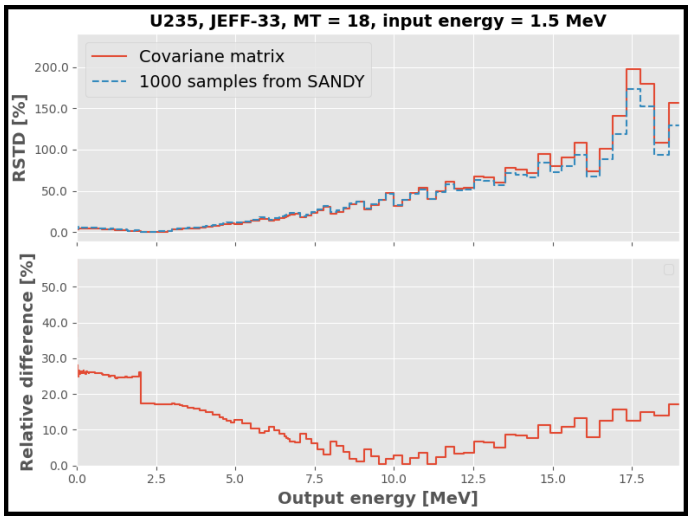


Figure 3.9: Difference between 1000 samples (perturbation coefficients generated with SANDY and assuming a log-Normal distribution) and the covariance matrix (at input energy of 1.5 MeV) relative standard deviations of the ^{235}U PFNS, as function of the output energies

3.3 Outputs processing

Once the n outputs (i.e. the dataset) of Serpent has been obtained, they must be processed to extract all the statistical information they contain, in order to apply the MC method (section 2.2.2) for the UQ/propagation. Thus, the mean value and the variance of the dataset are computed respectively through Equation 2.22 and Equation 2.23. Indeed, since the response function is only one (i.e. the k_{eff}), the covariance of the outputs is simply the variance. However, the values given is simply an estimation of the real value and the 95 % confidence interval of the k_{eff} variance is computed (see subsection 3.3.1): it means there is a probability of 95 % that the real value falls within the given interval. Moreover, a KS test is needed to verify the distribution followed by the dataset and, thus, since the response function is linear, to verify that it follows a Gaussian distribution. Finally, in the subsection 3.3.3, a variance-reduction method is explained, and it will be deepened in the section 4.2, after being implemented in SANDY.

3.3.1 Chi-square test for a population variance

The sample chi-square test for a population variance is used to verify if a dataset, with unbiased estimated population variance $\tilde{s}^2$, come from a population with variance σ^2 and it is based on the chi-square distribution (with test statistic represented by χ) [60]. In other words, the test is useful to know if the estimated variance is likely to belong to a population with a certain variance. For these reasons, it has been exploited to compute the confidence interval of the variance, with its probability: the p confidence interval represents the interval where there is a probability of p that the real value belongs to it [61]. If the test yields a significant result, it can be inferred that there is an high probability that the sample comes from a population with a different variance value, than the one hypothesised. The chi-square test relies on 2 fundamental hypotheses [62]:

1. The investigated dataset must be Normally distributed. To verify it, the KS test can be utilised (see subsection 3.3.2).
2. The sample chosen from the population is randomly taken.

If both are not respected, the test is not considered reliable. The chi-square distribution is an asymmetrical probability density function, with values between 0 and $+\infty$ and so it cannot have negative values [62]. There is an infinite number of chi-square distributions that depends on the degree of freedom df:

- As the degrees of freedom decrease, the distribution becomes increasingly positively skewed. Hence, there is a greater concentration of scores at the lower end of the distribution.

- The higher the degrees of freedom, the more the distribution tends to become symmetric.

To perform the test, different hypothesis can be done [60]:

Null hypothesis: $\tilde{s}^2 = \sigma^2$

Alternative hypothesis: it is divided in **directional** alternative hypothesis ($\tilde{s}^2 < or > \sigma^2$, assessed with a one-tailed test) and **non-directional** alternative hypothesis ($\tilde{s}^2 \neq \sigma^2$, assessed with a two-tailed test).

Since the variance confidence interval is the target, the non-directional alternative hypothesis is exploited and, thus, a two-tailed test. In this case, the null hypothesis can be rejected whether the chi-square value χ^2 (Equation 3.1) is equal or greater than the tabled critical two-tailed chi value in the upper tail of the chi-square distribution, at the specific level of confidence ($1 - \alpha$, related to the probability), or equal or less than the tabled critical two-tailed chi value in the upper tail of the chi-square distribution, at the specific level of confidence [62]. α is called percentile or significance level.

$$\chi^2 = \frac{(n-1)\tilde{s}^2}{\sigma^2} \tag{3.1}$$

where $\tilde{s}^2$ is the unbiased estimation of σ^2, computed with Equation 1.12.
Practically, to not reject the null hypothesis, the χ^2 value must be higher than the tabled two-tailed critical value in the lower tail of the distribution and lower than the tabled two-tailed critical value in the upper tail of the distribution, at the specific level of confidence. Using Equation 3.1, the formula for the $1 - \alpha$ confidence interval is:

$$\boxed{\frac{(n-1)\tilde{s}^2}{\chi^2_{(1-\frac{\alpha}{2})}} \leq \sigma^2 \leq \frac{(n-1)\tilde{s}^2}{\chi^2_{(\frac{\alpha}{2})}}} \tag{3.2}$$

where $\chi^2_{(\frac{\alpha}{2})}$ is the tabled critical two-tailed value in the chi-square distribution above which a percentage equal to [1 – (α/2)] of the cases falls. Subtracting the portion of the distribution that falls within the confidence interval (probability), α is obtained. As a consequence, the confidence interval has $1-\alpha$ as level of confidence. Moreover, since there is a lower and upper tail, $\alpha/2$ is taken (and in the upper tail $1 - \alpha/2$). However, if the degrees of freedom are larger than 30, the chi-square distribution can be approximated and replaced by the Normal distribution for the test and the confidence interval. The Normal distribution symmetry has been exploited (in particular the portion below the mean value) and its value is:

$$z = \frac{\tilde{s} - \sigma}{\frac{\sigma}{\sqrt{2n}}} \tag{3.3}$$

Again, with the same consideration of the chi-square distribution, the $1 - \alpha$ confidence interval, using the Normal distribution with its symmetry, is computed as:

$$\frac{\tilde{s}}{1 + \frac{z_{(\frac{\alpha}{2})}}{\sqrt{2n}}} \leq \sigma \leq \frac{\tilde{s}}{1 - \frac{z_{(\frac{\alpha}{2})}}{\sqrt{2n}}} \tag{3.4}$$

and $z_{(\frac{\alpha}{2})}$ has the same meaning of $\chi^2_{(\frac{\alpha}{2})}$, but for a Normal distribution. In this thesis, the 95 % confidence interval ($\alpha = 5\%$) for the variance of the k_{eff} outputs has been computed and, since the number of samples is always larger than 30, the Normal distribution for the chi-square test has been used. Moreover, the tabled critical two-tailed value $z_{(0.025)}$ is 1.96 [62].

3.3.2 Kolmogorov-Smirnov test

The Kolmogorov–Smirnov (KS) test allows verifying whether the dataset/samples/-variable empirical distribution (i.e. the population distribution) follows a specific theoretical distribution (or more in general if two distributions are equal) [59]. In other words, it determines whether the observations derive from the specified distribution, based on the goodness of fit.

Specifically, the test compares the two cumulative distribution functions and computes the maximum difference registered, in absolute value, called D or test statistic. Another important parameter is the p-value: it is the probability of the found D statistic, if the null hypothesis is true. The test outcome depends on the percentile or significance level α and the lower is the highest is the probability to reject the null hypothesis (i.e. the distributions are identical). Thus, the p-value is the smallest α value for which the null hypothesis can be rejected, and it is usual to set the threshold to 0.05. Therefore, if the p-value is higher than 0.05 the null hypothesis cannot be rejected and the two distributions are assumed to be equal.

The KS test has been performed using the *scipy* package [63] of Python, in particular `scipy.stats.ks_1samp` to compare the empirical dataset distribution with the Normal one. However, the dataset must be standardised (i.e. transformed in the same distribution but with mean value zero and variance one) before the test is performed, otherwise the test is compromised because the two distributions have different mean and variance and the comparison is not compatible. Indeed, the KS test compares the shape more than the precise values of the distribution parameters.

3.3.3 Latin Hypercube sampling

Latin Hypercube (LH) is a sampling method, based on MC sampling, developed by McKay, Conover and Beckman in 1979 [64]. This approach allows reducing the statistical uncertainty during sampling (i.e. the sampling is more accurate, given a precise number of samples) and thus the variance of the sample mean is reduced. Hence, given a preselected number of samples, the LH allows to obtain a more accurate representation of the statistical information as well as to help the convergence of the solution/outputs. Given d input variables and n samples to generate, the approach is simple:

- First, the domain of each variable is divided into n non-overlapping intervals, based on equal probability (i.e. all the intervals have the same probability). See example with normal distribution and 5 samples in Figure 3.10.

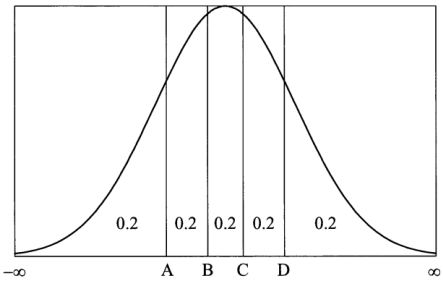


Figure 3.10: Division of the input variable domain in n=5 intervals, representing the same probability [64]

- Then, in each interval, a value is taken according to the probability density function contained in it. In other words, the value is selected based on the probability density function height within the interval, performing a simple MC sampling.

- Finally, all the variables samples are randomly paired (usually by linking a random arrangement of the initial n integers to each input variable) to form the n samples input vectors of dimension d.

To summarise, generally, a collection of n LH sample points in d-dimensional Euclidean space comprises one point within every interval for each of the d variables. Therefore, the matrix $\mathbf{X}$ of the independent variables $\mathbf{x}$, normally distributed (see subsection 2.3.1), is obtained applying the Equation 3.5 [65]:

$$X_{i,j} = \frac{\pi_i(i-1) + U_{i,j}}{n}, \quad 1 \leq i \leq d \quad 1 \leq j \leq n \tag{3.5}$$

where $\pi_1, \pi_2, ..., \pi_d$ are the {0,1,...,n-1} uniform random permutations and $U_{i,j}$~**U**(0,1), both independent. The Equation 3.5 is applied using the `qmc` module (and its `LatinHypercube` method) of the package *scipy*.

Chapter 4

Results

Two main investigations have been performed: first, the uncertainty on the k_{eff} due to the uncertainty of the nubar of the main fissile nuclide and second, the uncertainty on the k_{eff} due to the uncertainty of the nubar and all the cross-sections of the input nuclides. Besides, for the first task, the ENDF/B-VII.I library has been chosen while for the second one the JEFF-33. The process has been explained in the chapter 3. Therefore, after the n samples has been produced, the different benchmarks have been simulated n times and the correspondent outputs have been analysed. Moreover, to verify the results, different methods have been exploited.

4.1 Uncertainty on the k_{eff} due to ^{239}Pu nubar perturbed data

First, 200 ^{239}Pu nubar samples have been produced with SANDY and then the uncertainty on the k_{eff}, for the different benchmarks (section 2.5) and due to ^{239}Pu nubar uncertainty, has been investigated. Moreover, the same uncertainty has been computed also through the sandwich formula (see section 2.2.1). In particular, the sensitivity vector (vector since only one response function) of the k_{eff} on the ^{239}Pu nubar, for each benchmark, has been computed through Serpent2 while the relative covariance matrix (independent on the benchmarks) has been extracted with SANDY from the ENDF-6 relative file and the Equation 2.19 has been applied.
An example of sensitivity vector is shown in Figure 4.1 and, as expected, the sensitivity is high in the fast spectrum, in particular at 2 MeV (i.e. the average energy of the neutron generated by fission).

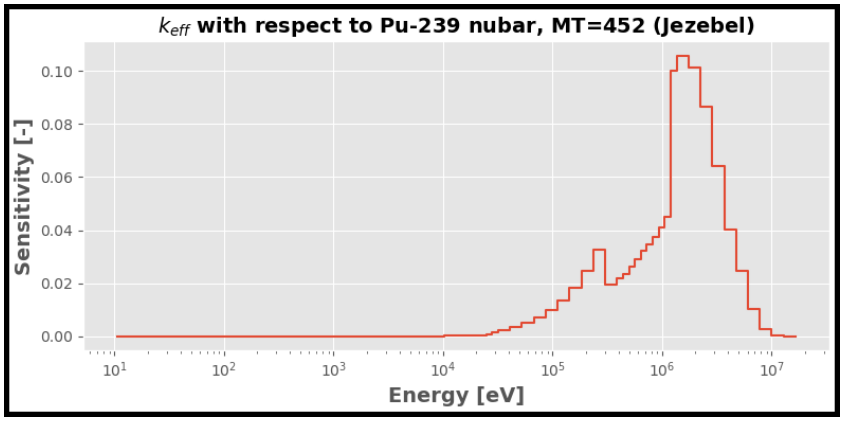


Figure 4.1: Sensitivity profile of the Jezebel benchmark k_{eff} to the ^{239}Pu total nubar

It is important to specify that the sensitivity vector, since it is computed through a MC code, has its uncertainty, but the latter has been investigated, and it came out to be negligible (i.e. precise value, no confidence interval). On the other hands, since a statistical post-process has been preformed for the n outputs, the SANDY method has its 95 % confidence interval, computed as explained in subsection 3.3.1. Figure 4.2 shows the results obtained for the different benchmarks:

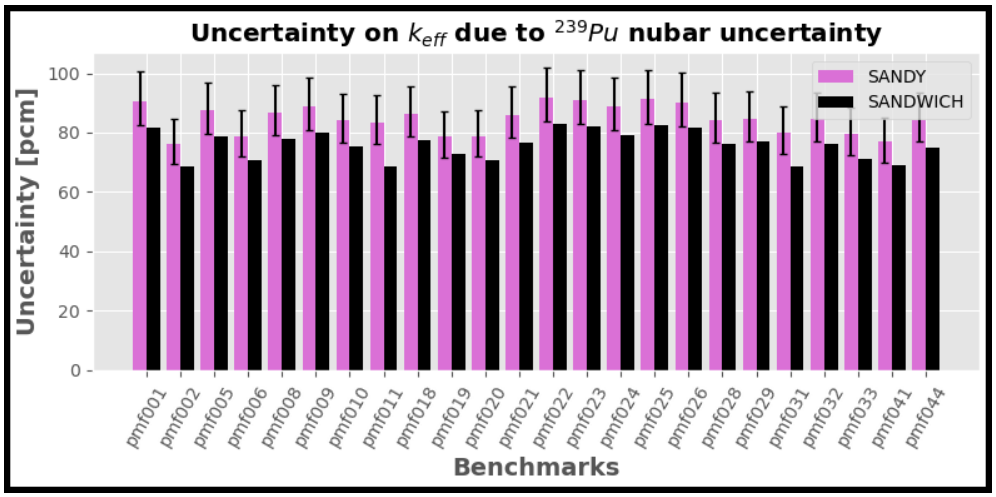
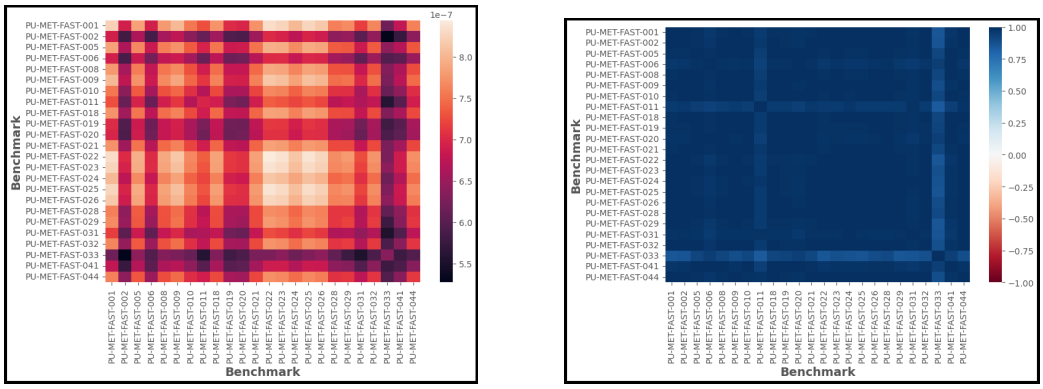


Figure 4.2: Uncertainty on the k_{eff} due to the ^{239}Pu nubar uncertainty, for the different plutonium benchmarks

the UQ/propagation with SANDY (also called here SANDY method[1]) leads to a systematic overestimation, compared to the implemented sandwich rule. Most likely, the overestimation is due to the random samples generated and, maybe, the convergence with 200 samples has not been reached, and the samples generated do not correctly represent the statistics of the covariance matrix. Indeed, since the possible non-converged samples used are the same, the overestimation is experienced in all the benchmarks. For this reason, the PU-MET-FAST-005 (one of the most "unreliable") has been analysed more in the detail in the subsection 4.1.1 to better prove the overestimation cause.

Overall, the results can be considered verified (at least the values are close) as well as the SANDY sampling. Furthermore, the sandwich rule does not give the real value, since uncertainty is a very complex field to verify and validate. In the Figure 4.3 is represented the covariance and the correlation matrix of the different benchmarks response function (so the parameters are the benchmarks themselves). As predictable, the correlation coefficients are all positive because if the nubar increases the k_{eff} increases (and vice versa), for all the benchmarks.



(a) Covariance matrix **(b)** Correlation matrix

Figure 4.3: Covariance ($cov(\mathbf{y})$) and correlation matrix of the k_{eff} response function for the different benchmarks

Finally, the LH sampling has been performed and investigated for the Jezebel benchmark in the section 4.2.

[1]because the samples have been produced with SANDY, but it is the stochastic MC method

4.1.1 Overestimation cause: in-depth analysis using the PU-MET-FAST-005

First, the number of samples has been increased from 200 to 500 and Figure 4.4 shows the results obtained. As expected, even if still the uncertainty is overestimated, the overestimation has been reduced and the statistical value is closer (with a reduction of the confidence interval) to the one from the sandwich rule, further verifying the SANDY sampling and the UQ/propagation method.

Given that the overestimation is systematic (i.e. present also for all the other benchmarks), it can come from the fact that the other 300 samples have been added and so the first 200 samples are the same. It could have been interesting producing 500 samples independently from the first 200 and maybe systematic overestimation would not have been produced.

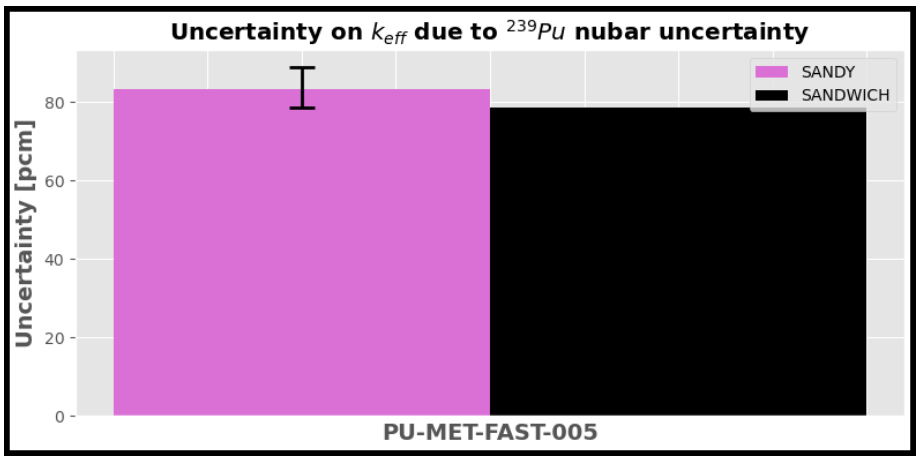
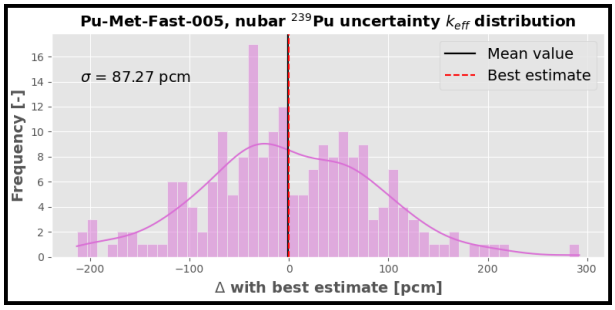
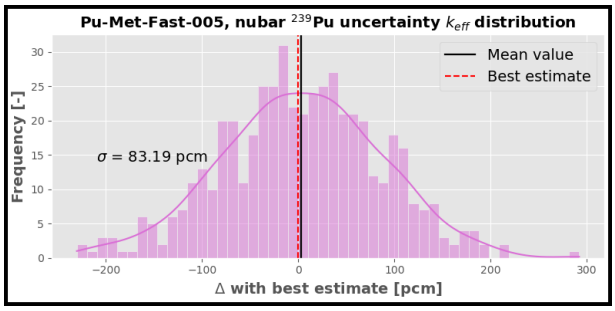


Figure 4.4: Uncertainty on the k_{eff} due to the ^{239}Pu nubar uncertainty, for the PU-MET-FAST-005 plutonium benchmark, using 500 samples

In the Figure 4.5, the k_{eff} outputs distributions are shown, with 200 and 500 samples. The distribution with 500 samples looks more "Normal", as it should be from CLT. Moreover, the mean value of the outputs and the best estimate (i.e. the run with the best estimate value of the nubar, stored in the ENDF-6) are almost equal, proving the linearity of the response function.



(a) 200 samples



(b) 500 samples

Figure 4.5: k_{eff} outputs distribution with different number of samples, for PU-MET-FAST-005

Moreover, Figure 4.6 proves that the outputs values, with their uncertainty, contain the best estimate run, confirming the MC technique.

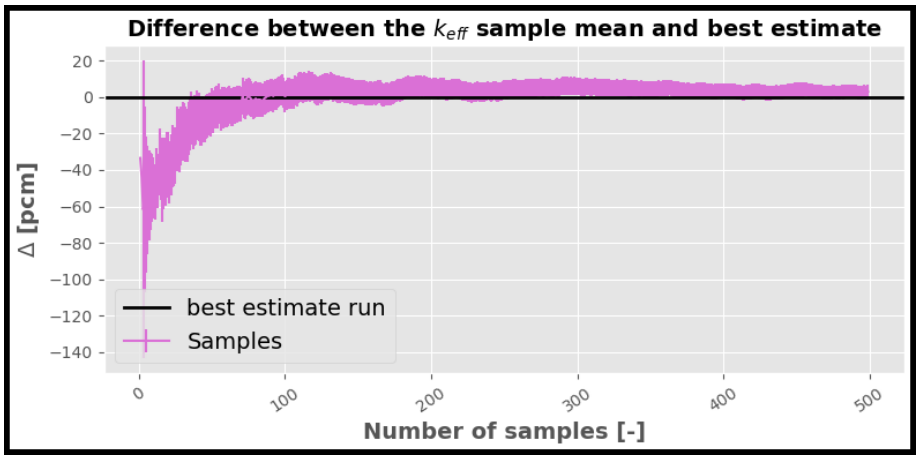


Figure 4.6: k_{eff} mean value and uncertainty, as function of the number of samples, and comparison with the best estimate run for the PU-MET-FAST-005 benchmark

Finally, the KS test (see subsection 3.3.2) has been performed (Figure 4.7) and, as it should be, by increasing the number of samples, the p-value increases and the test statistic decreases. Furthermore, from a number of samples equal to about 35 onward, the null hypothesis cannot be rejected and the distribution is assumed Normal.

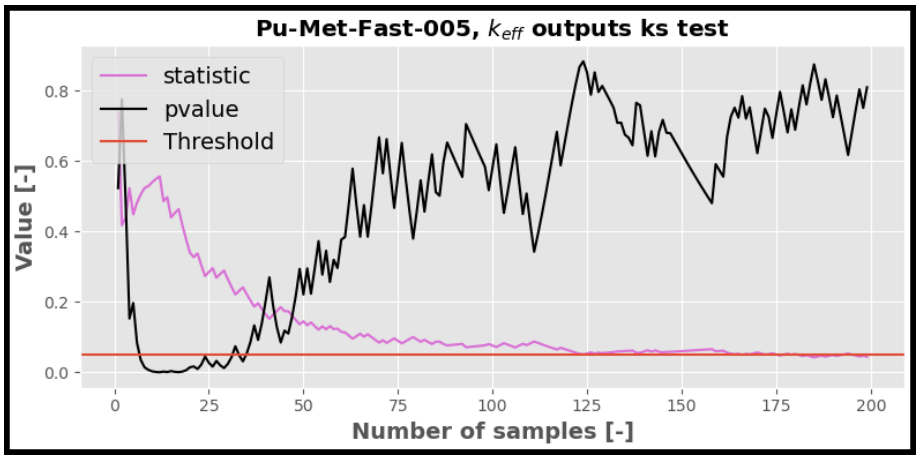


Figure 4.7: p-value and test statistic as function of the number of samples, for the PU-MET-FAST-005 benchmark

4.2 Latin Hypercube applied to Jezebel

Here, the results with the standard and the Latin Hypercube (LH) method will be compared, for the Jezebel benchmark.

First, the standard deviation of the k_{eff} outputs (Figure 4.8): the trend given by the LH sampling anticipates the other, even if conclusions cannot be made.

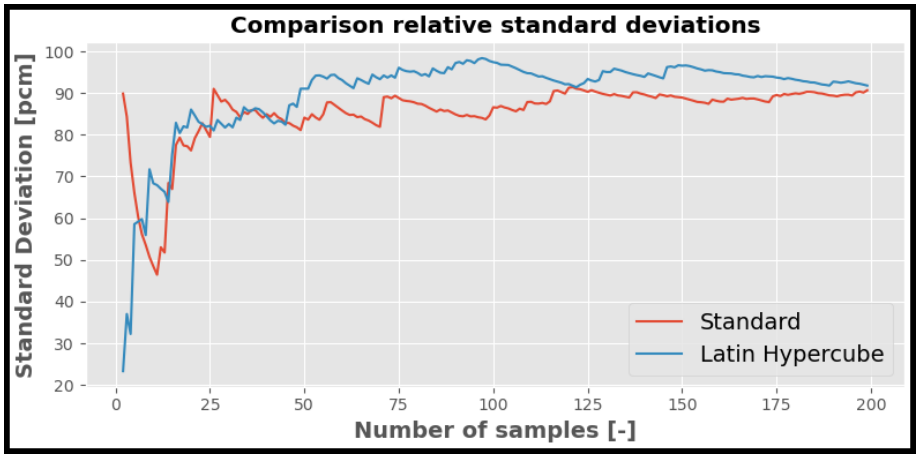


Figure 4.8: Comparison between the k_{eff} standard deviations using the standard and the LH sampling method, for the Jezebel benchmark

An interesting figure is the Figure 4.9 where the KS test has been performed on the k_{eff} outputs, for the two methods, and the following observations can be done:

1. The p-value using the LH is always larger than the one using the standard method (considering the same number of samples) and, thus, it better represents the statistic information.

2. The LH is very useful with a small number of samples, where it works very well, while at higher ones the two methods leads to the same results.

3. The LH method gives p-values always larger than the threshold, highlighting its sampling power

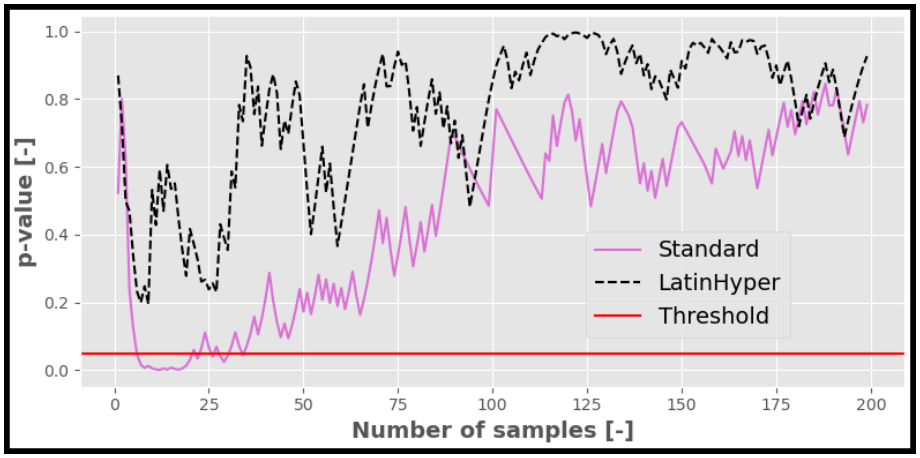


Figure 4.9: p-value as function of the number of samples, for the Jezebel benchmark, using the standard and the LH method

All the result information are summarised in the Figure 4.10.

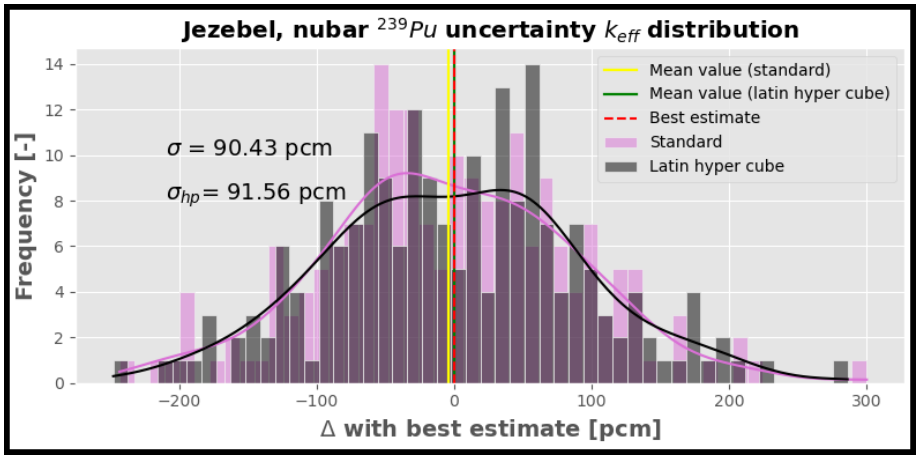


Figure 4.10: k_{eff} outputs distribution with the standard and the LH sampling method

4.3 Uncertainty on the k_{eff} due to all cross-sections and nubar perturbed data

In this section the uncertainty on the k_{eff}, for the different benchmarks (section 2.5) and due to the cross-sections and nubar uncertainty of all the nuclides, has been investigated and the results are shown in the Figure 4.11, with their 95 % confidence interval.

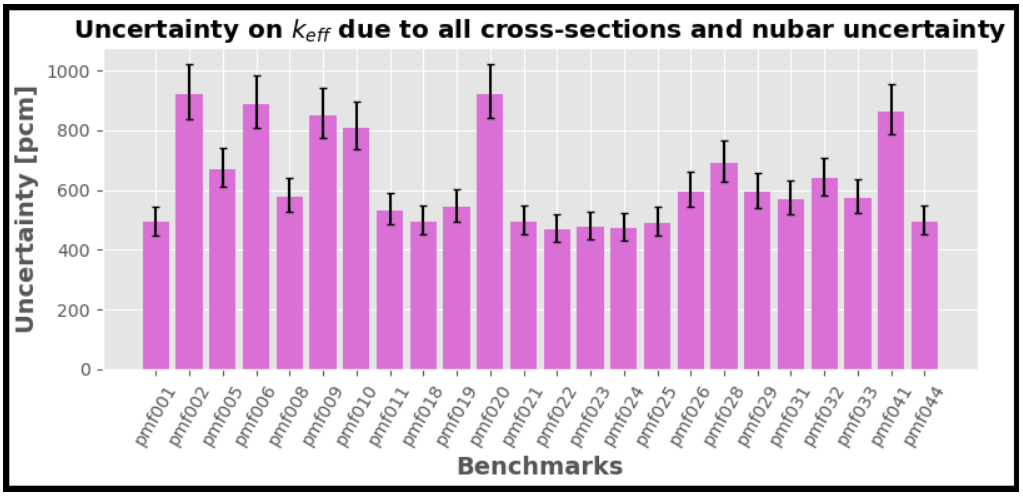


Figure 4.11: Uncertainty on the k_{eff} due to the cross-sections and nubar (only for suitable nuclides) uncertainty of all the nuclides, for the different plutonium benchmarks

Unlike the section 4.1, the results using MC technique (with the sampling made by SANDY) has been compared and verified with NDaST (see subsection 2.4.1). However, NDaST does not work with the nubar and an indirect comparison has been made. In particular, the uncertainty on the k_{eff} only due to all the cross-sections has been chosen and, in order to compute it for the SANDY method, the following computation has been done: the variance of the k_{eff} due to nubar has been subtracted to all cross-sections and nubar variance, giving the variance due to only the cross-sections (for all the nuclides) and therefore the uncertainty (by applying the square root).
The comparison is shown in the Figure 4.12 and, overall, the SANDY sampling, with its relative UQ/propagation MC method, is verified.

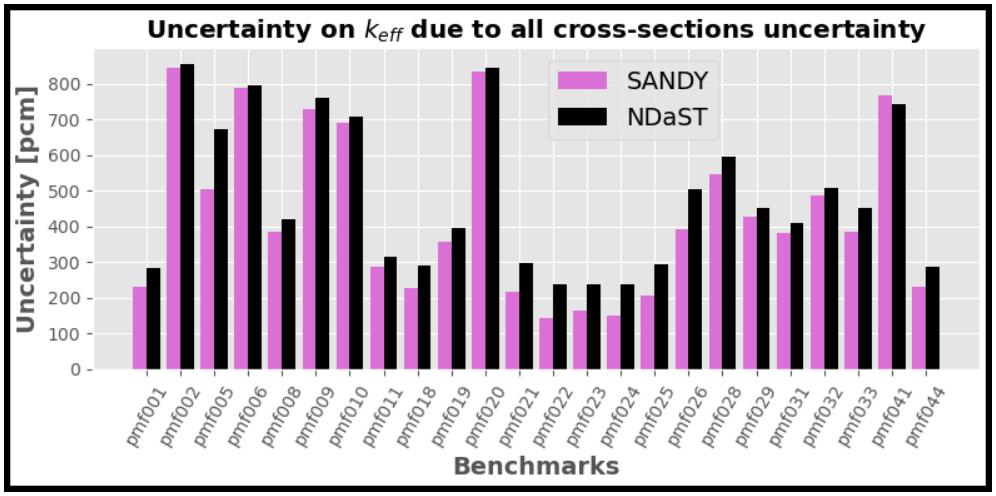


Figure 4.12: Comparison between SANDY and NDaST for the uncertainty on the k_{eff} due to the all cross-sections uncertainty, for the different plutonium benchmarks

The general systematic underestimation is a direct consequence of the overestimation described in the section 4.1, since the variance due to nubar has been subtracted. However, for some benchmarks, the values are very distant, especially for the PU-MET-FAST-005. For this reason, the latter has been studied in detail in the subsection 4.3.1.

4.3.1 Tungsten issue: in-depth analysis

In order to find the error source in the PU-MET-FAST-005, a first investigation has been made: analysing the cross-section, with its nuclide, that strongly affects the uncertainty on the k_{eff}. After a full quest, one of the most affecting uncertainty is the one from the inelastic cross-section (MT=4, Table A.2) of the tungsten-184 (^{184}W) and, thus, an UQ/propagation, due to only its uncertainty, on the k_{eff} has been performed with the MC method (based on SANDY). Indeed, the SANDY method result has given an underestimation of k_{eff} uncertainty, compared to NDaST.

Analysing the ^{184}W relative covariance matrix, for MT=4 (see Figure 4.13 and Figure 4.14), it has been noticed that the relative standard deviation is very high, close to 100 % at energy about 0.5 MeV, and in the subsection 2.3.3 is clearly explained that SANDY does not work correctly, if a Normal distribution is assumed.

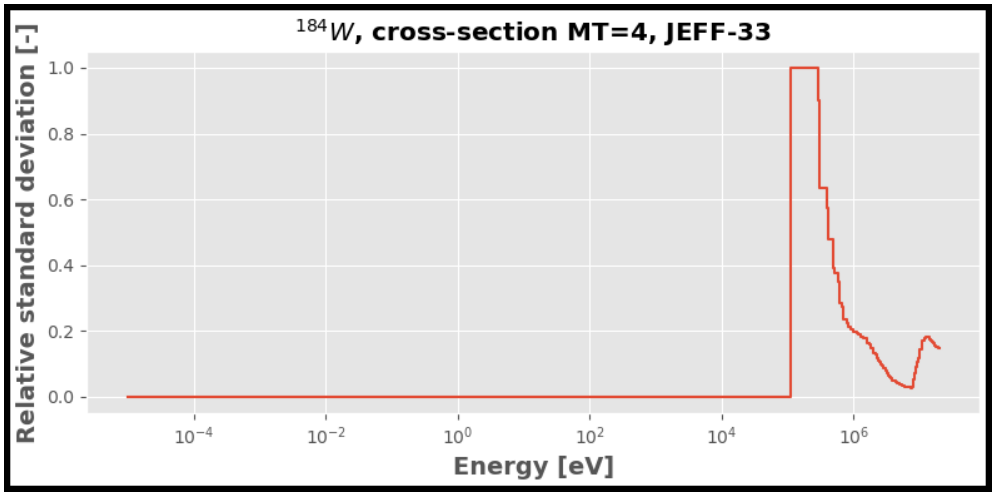


Figure 4.13: Relative standard deviations, as function of the energy, of the ^{184}W for the reaction MT=4

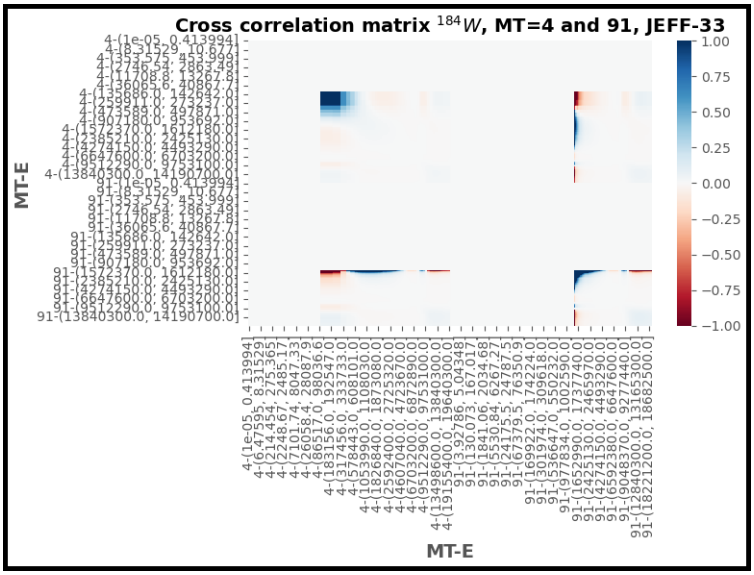


Figure 4.14: Cross-correlation matrix of the ^{184}W for the reactions MT=4 and MT=91 and different energies

For this reason, the same UQ/propagation has been performed, but this time, the log-Normal distribution of the ^{184}W samples has been assumed. In the Figure 4.15 are shown the results, showing the correctness of the log-Normal distribution assumption and giving evidence of the considerations made.

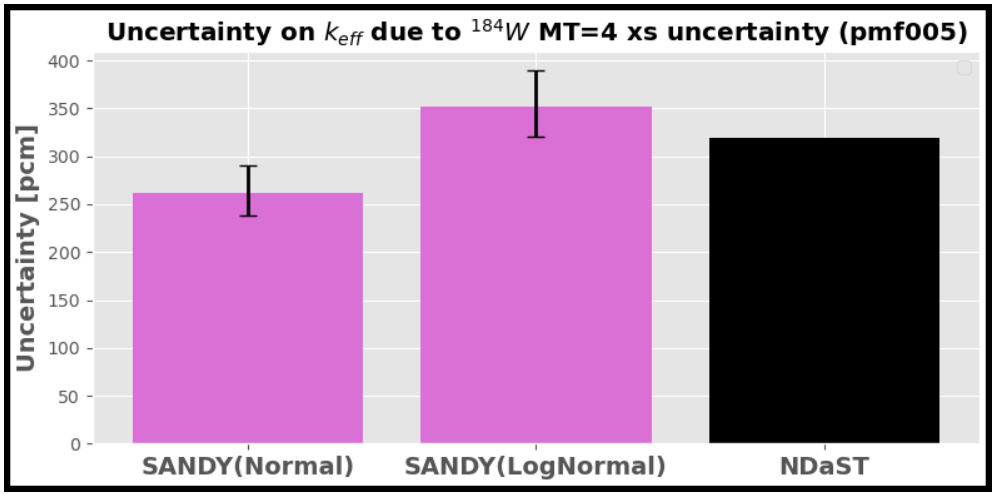


Figure 4.15: Comparison between SANDY (Normal and log-Normal distribution) and NDaST for the uncertainty on the k_{eff} due to MT=4 ^{184}W cross-section uncertainty, for PU-MET-FAST-005

As a consequence, also for the other tungsten-uncertainty-affected benchmarks k_{eff} the same consideration can be done. More generally, the latter can be applied to all the benchmarks whose cross-sections or nubar uncertainty is very high and strongly affects the k_{eff}.

Conclusion

This dissertation aimed to implement the nubar nuclear data into SANDY and then to verify it performing an UQ/propagation on pre-selected benchmarks k_{eff}.

First, the nubar has been correctly added in the SANDY code. Moreover, by generating 1000 nubar samples, their mean values and standard deviations has been compared respectively with the best estimate (stored in the ENDF-6) and with the standard deviations (stored in the covariance matrix), giving a relative small difference and proving the sampling.
The same job has been done for the energy distributions nuclear data (PFNS) but the latter has not been verified and completed. However, it has been discovered that, since the energy distribution nuclear data contains high uncertainties, the latter cannot be sampled assuming a Normal distribution and a log-Normal one gave interesting results.

As first integral study, the effect of the ^{239}Pu nubar uncertainty on specific benchmarks k_{eff} has been investigated performing a MC method, and the results has been compared with another method: the sandwich rule, where the sensitivity vector has been computed through Serpent2 and the covariance matrix extracted by SANDY. The two outputs were very close (i.e. the 95 % confidence interval, obtained with the stochastic method, of the k_{eff} standard deviation included the value from the other method), although a systematic overestimation came out. Nevertheless, increasing the number of samples, the overestimation has been reduced and the results are even closer, verifying first the method implemented and then the sampling one with SANDY.
Afterwards, the LH sampling has been tested: it came out that the latter gives advantages, reducing the variance of the sample mean, only if a small number of samples is taken.

In the second study, a more complete investigation has been made. Here, all the cross-sections of all the benchmarks nuclides has been perturbed, together with the fissile nuclides nubar and their uncertainty propagation has been computed

on the k_{eff} (same benchmarks as before), always with a MC approach. This time, NDaST has been chosen as comparison method and the MC results have showed a systematic underestimation: it is nothing but the consequence of the nubar overestimation, since to compare the methods a variance subtraction has been adopted.
Again, the results are quite reliable, except for some benchmarks whose nuclides cross-sections were characterised by high uncertainties, such as the MT=4 reaction cross-section of the ^{184}W in the PU-MET-FAST-005. Indeed, in cases of high uncertainties and assumed Normal distribution, an underestimation occurs and SANDY does not work well. After assuming a log-Normal distribution also those benchmarks has been verified.

Therefore, SANDY has new methods that can be easily recalled for the perturbation of the nubar nuclear data and, thanks to it and adopting a MC method, UQ/propagation studies can be carried out. The latter is not only a check, but it is useful for the nuclear libraries evaluators (being part of the nuclear data activities) other than for design purposes. Indeed, knowing the uncertainty of some neutronic parameters, as the k_{eff}, allows reducing the risk associated with nuclear power plants and, maybe with the right information, more consent and approval can be gained from the population. The policy in favour of new nuclear power plants, together with the extension of the old one, can truly help to reach the NZE.

Recommendations

Despite the positive outcomes of this thesis, it is dutiful to give some recommendations.

First, the integral benchmarks' verification is valid only for the selected benchmarks themselves. Therefore, a different benchmark or a specific reactor project could not produce reliable results (for example due to particular nuclides, geometries or simply due to other hidden uncertainties) and, thus, a further research is needed to amplify the described method field of application.

Furthermore, to better understand the systematic overestimation, a deeper study could be useful. For example, more and different samples for each benchmark can be produced, even if not crucial (since the confidence interval is compliant with the other method) and not necessarily an explanation can arise.

Finally, the implementation of energy distributions has started but not completed. I would recommend, to the one pursuing this job, the same methodology applied

for the nubar but something must be changed: the Normal distribution cannot be assumed. Moreover, it will be important to verify that each output energy beam is correctly represented: indeed, since the PFNS values depend on the output energy and also on the input one, the related different (even if it is small difference) input energy covariance matrices can produce not reliable results, if no cross-correlation is present between them.
Last but not least, the same benchmarks can be used as verification, although the latter is limited to them.

Appendix A

ENDF-6 MF and MT tables

Table A.1: MF definitions [9]

MF	Description
1	General information and fission multiplicities
2	Resolved and unresolved resonance parameter data
3	Reaction cross-sections
4	Angular distributions for secondary particles
5	Energy distributions for secondary particles
6	Energy-angle distributions for emitted particles
7	Thermal neutron scattering law data S(α, β)
8	Radioactivity and fission-product yield data
9	Multiplicities for radioactive nuclide production
10	Cross-sections for radioactive nuclide production
11	General comments of photon production
12	Photon Production and Multiplicities and Transition Probability Arrays
13	Photon production cross-sections
14	Photon Angular Distribution
15	Continuous Photon Energy Spectra
...	
23	Photon interaction cross-section
...	
26	Electro-atomic angle and energy distribution
27	Atomic From Factors or Scattering Functions
28	Atomic Relaxation data
...	
31	Data covariances for average fission neutron multiplicity (nubar)
32	Data covariances for resonance parameters

33	Data covariances for reaction cross-sections
34	Data covariances for angular distributions of secondary particles
35	Data covariances for energy distributions of secondary particles
...	
39	Data covariances for radionuclide production yields
40	Data covariances for radionuclide production cross-sections
...	

Table A.2: MT definitions [9]

MT	Description
1	(z,total)
2	(z,z0) elastic scattering cross-section
3	(z,non-elastic) Sum of MT=4-5, 11, 16-17, 22-37, 41-42,...
4	(z,n') cross-section for the emission of one neutron in the exit channel. Sum of MT=50-91
5	(z, anything) the cross-section for all other reactions, whose MT number is not explicitly present
...	
16	(n,2n)
17	(n,3n)
18	(z,fission) Total fission cross-section. Sum of MT=19, 20, 21 and 38
19	(n,f) first-change neutron induced fission
...	
51	(z,n_1) Production of a neutron, with residual in the 1^{st} excited state
52	(z,n_2) Production of a neutron, with residual in the 2^{nd} excited state
...	
91	(z,n_c)Generation of a neutron within the continuous spectrum that is not accounted for in the discrete representation mentioned earlier
...	
101	Neutron disappearance. Sum of MT=102-117, 155, 182, 191-193, 197
102	(n,γ) radioactive capture cross-section
103	(n,p) Proton production after neutron absorption cross-section. Sum of MT=600-649
...	
107	(n,α) α-particle (i.e. He nucleus) production after neutron absorption cross-section. Sum of MT=800-849
452	Total average fission neutron multiplicity
455	Delayed average fission neutron multiplicity
456	Prompt average fission neutron multiplicity
...	

Appendix B

SANDY code modified scripts

B.1 Obtain the samples or perturbation coefficients

```
    def get_perturbations(
        self,
        nsmp,
        to_excel=None,
        njoy_kws={},
        smp_kws={},
        **kwargs,
        ):
        """
        Construct multivariate distributions with a unit vector for
        mean and with relative covariances taken from the evaluated files
        processed with the NJOY module ERRORR.

        Perturbation factors are sampled with the same multigroup structure of
        the covariance matrix and are returned by nuclear datatype as a `dict`
        of `pd.Dataframe` instances .

        Parameters
        ----------
        nsmp : TYPE
            DESCRIPTION.
```

```
22          to_excel : TYPE, optional
23              DESCRIPTION. The default is None.
24          njoy_kws : TYPE, optional
25              DESCRIPTION. The default is {}.
26          smp_kws : TYPE, optional
27              DESCRIPTION. The default is {}.
28          **kwargs : TYPE
29              DESCRIPTION.
30
31          Returns
32          -------
33          smp : TYPE
34              DESCRIPTION.
35
36          Examples
37          --------
38          Generate a couple of samples from the H1 file of JEFF-3.3.
39          >>> njoy_kws = dict(err=1, errorr_kws=dict(mt=102))
40          >>> tape = sandy.get_endf6_file("jeff_33", "xs", 10010)
41          >>> smps = tape.get_perturbations(nsmp=2, njoy_kws=njoy_kws)
42          >>> assert len(smps) == 1
43          >>> assert isinstance(smps[33], sandy.Samples)
44          >>> assert (smps[33].data.index.get_level_values("MT") ==
    102).all()
45          """
46          smp = {}
47          seeds = {}
48
49          outs = self.get_errorr(**njoy_kws)
50
51          if "errorr31" in outs:
52              smp_kws["seed"] = seed = smp_kws.get("seed31", sandy.
    get_seed())
53              seeds["errorr31"] = seed
54              smp[31] = outs["errorr31"].get_cov().sampling(nsmp, **
    smp_kws)
55          if "errorr33" in outs:
56              smp_kws["seed"] = seed = smp_kws.get("seed33", sandy.
    get_seed())
57              seeds["errorr33"] = seed
58              smp[33] = outs["errorr33"].get_cov().sampling(nsmp, **
    smp_kws)
59          if to_excel and smp:
60              with pd.ExcelWriter(to_excel) as writer:
61                  for k, v in smp.items():
62                      v.to_excel(writer, sheet_name=f'MF{k}')
63          return smp
```

B.2 Obtain the perturbed nuclear data

```
    def get_perturbations(
        self,
        nsmp,
        to_excel=None,
        njoy_kws={},
        smp_kws={},
        **kwargs,
        ):
        """
        Construct multivariate distributions with a unit vector for
        mean and with relative covariances taken from the evaluated files
        processed with the NJOY module ERRORR.

        Perturbation factors are sampled with the same multigroup structure of
        the covariance matrix and are returned by nuclear datatype as a `dict`
        of `pd.Dataframe` instances .

        Parameters
        ----------
        nsmp : TYPE
            DESCRIPTION.
        to_excel : TYPE, optional
            DESCRIPTION. The default is None.
        njoy_kws : TYPE, optional
            DESCRIPTION. The default is {}.
        smp_kws : TYPE, optional
            DESCRIPTION. The default is {}.
        **kwargs : TYPE
            DESCRIPTION.

        Returns
        -------
        smp : TYPE
            DESCRIPTION.

        Examples
        --------
        Generate a couple of samples from the H1 file of JEFF-3.3.
        >>> njoy_kws = dict(err=1, errorr_kws=dict(mt=102))
        >>> tape = sandy.get_endf6_file("jeff_33", "xs", 10010)
        >>> smps = tape.get_perturbations(nsmp=2, njoy_kws=njoy_kws)
        >>> assert len(smps) == 1
```

```
43          >>> assert isinstance(smps[33], sandy.Samples)
44          >>> assert (smps[33].data.index.get_level_values("MT") ==
    102).all()
45          """
46          smp = {}
47          seeds = {}
48
49          outs = self.get_errorr(**njoy_kws)
50
51          if "errorr31" in outs:
52              smp_kws["seed"] = seed = smp_kws.get("seed31", sandy.
    get_seed())
53              seeds["errorr31"] = seed
54              smp[31] = outs["errorr31"].get_cov().sampling(nsmp, **
    smp_kws)
55          if "errorr33" in outs:
56              smp_kws["seed"] = seed = smp_kws.get("seed33", sandy.
    get_seed())
57              seeds["errorr33"] = seed
58              smp[33] = outs["errorr33"].get_cov().sampling(nsmp, **
    smp_kws)
59          if to_excel and smp:
60              with pd.ExcelWriter(to_excel) as writer:
61                  for k, v in smp.items():
62                      v.to_excel(writer, sheet_name=f'MF{k}')
63          return smp
64
65      def apply_perturbations(self, smps, processes=1, njoy_kws={}, **
    kwargs):
66          """
67          Apply perturbations to the data contained in ENDF6 file. At
    the
68          moment only the procedure for cross sections is implemented.
    Options
69          are included to directly convert perturbed pendf to ace and
    write data
70          on files.
71
72          Parameters
73          ----------
74          smps : samples obtained taking the relative covariances from
    the
75          evaluated files and a unit vector as mean.
76          processes : number of processes used to complete the task.
77                      Creation of perturbed pendf files and conversion
    to ACE
78                      format is done in parallel if processes>1.
79                      The default is 1.
80          njoy_kws: keyword argument to pass to "tape.get_pendf()".
```

```
81          **kwargs : keyword argument to pass to "tape.get_ace()" plus
    keywords
82                     to pass to "endf6_perturb_worker".
83
84          Returns
85          -------
86          A dictionary of endf/pendf file or ace files depending on
    to_ace.
87
88          Notes
89          -----
90          .. note:: ACE file temperature. Two options are implemented:
91                    - Generation of pendf file at 0K and broadening to
    the
92                      required temperature when ACE file is created.
93                    - Generation of pendf file at temperature and
    broadening not
94                      performed when ACE is created. This approach
    takes into
95                      account implicit effect.
96
97          Examples
98          --------
99          Example to produce and apply perturbations to Pu-239 xs and
    nubar.
100         >>> tape = sandy.get_endf6_file("jeff_33", "xs", 942390)
101         >>> smps = tape.get_perturbations(2, njoy_kws=dict(err=1, chi
    =False, mubar=False, errorr33_kws=dict(mt=[2, 4, 18]),), smp_kws=
    dict(seed31=1, seed33=3))
102
103         Let's apply both nubar and xs perturbations, then only nubar
    and then only xs.
104         >>> outs_31_33 = tape.apply_perturbations(smps, njoy_kws=dict
    (err=1), processes=1)
105         >>> outs_31 = tape.apply_perturbations({31: smps[31]},
    njoy_kws=dict(err=1), processes=1)
106         >>> outs_33 = tape.apply_perturbations({33: smps[33]},
    njoy_kws=dict(err=1), processes=1)
107
108         Check that files are different for different samples.
109         >>> for i in range(2):
110         ...     assert(outs_33[i]["endf6"].data == tape.data)
111         ...     assert(outs_31[i]["endf6"].data != tape.data)
112         ...     assert(outs_31[i]["endf6"].data == outs_31_33[i]["
    endf6"].data)
113         ...     assert(outs_33[i]["pendf"].data != outs_31[i]["pendf
    "].data)
114         ...     assert(outs_33[i]["pendf"].data == outs_31_33[i]["
    pendf"].data)
```

```
115
116         Check that method is consistent only nubar, only xs or both
    nubar and xs are perturbed.
117         >>> assert outs_33[0]["pendf"].data != outs_33[1]["pendf"].
    data
118         >>> assert outs_33[0]["endf6"].data == outs_33[1]["endf6"].
    data
119         >>> assert outs_31[0]["pendf"].data == outs_31[1]["pendf"].
    data
120         >>> assert outs_31[0]["endf6"].data != outs_31[1]["endf6"].
    data
121
122         Check that redundant nubar is also perturbed.
123         >>> nu0 = sandy.Xs.from_endf6(outs_31[0]["endf6"].filter_by(
    listmt=[452, 455, 456]))
124         >>> nu1 = sandy.Xs.from_endf6(outs_31[1]["endf6"].filter_by(
    listmt=[452, 455, 456]))
125         >>> assert not nu0.data[9437, 452].equals(nu1.data[9437,
    452])
126         >>> assert nu0.data[9437, 455].equals(nu1.data[9437, 455])
127         >>> assert not nu0.data[9437, 456].equals(nu1.data[9437,
    456])
128
129         Check that redundant and partial cross sections are correctly
     perturbed.
130         >>> xs0 = sandy.Xs.from_endf6(outs_33[0]["pendf"].filter_by(
    listmf=[3]))
131         >>> xs1 = sandy.Xs.from_endf6(outs_33[1]["pendf"].filter_by(
    listmf=[3]))
132         >>> assert not xs0.data[9437, 1].equals(xs1.data[9437, 1])
133         >>> assert not xs0.data[9437, 2].equals(xs1.data[9437, 2])
134         >>> assert not xs0.data[9437, 4].equals(xs1.data[9437, 4])
135         >>> assert not xs0.data[9437, 18].equals(xs1.data[9437, 18])
136         >>> assert not xs0.data[9437, 51].equals(xs1.data[9437, 51])
137         >>> assert xs0.data[9437, 16].equals(xs1.data[9437, 16])
138         >>> assert xs0.data[9437, 102].equals(xs1.data[9437, 102])
139         >>> assert xs0.data[9437, 103].equals(xs1.data[9437, 103])
140         >>> assert xs0.data[9437, 107].equals(xs1.data[9437, 107])
141
142         Check that ENDF6 and PENDF output filenames are correct
143         >>> endf6 = sandy.get_endf6_file('jeff_33', 'xs', 10010)
144         >>> smps = endf6.get_perturbations(2, njoy_kws=dict(err=0.1))
145         >>> outs = endf6.apply_perturbations(smps, to_file=True)
146         >>> assert outs[0]["endf6"] == '1001_0.endf6' and os.path.
    isfile('1001_0.endf6')
147         >>> assert outs[0]["pendf"] == '1001_0.pendf' and os.path.
    isfile('1001_0.endf6')
148         >>> assert outs[1]["endf6"] == '1001_1.endf6' and os.path.
    isfile('1001_1.endf6')
```

```
149         >>> assert outs[1]["pendf"] == '1001_1.pendf' and os.path.
    isfile('1001_1.pendf')
150 
151         Check that ACE output filenames are correct
152         >>> outs = endf6.apply_perturbations(smps, to_file=True,
    to_ace=True, ace_kws=dict(err=1, temperature=300, purr=False,
    heatr=False, thermr=False, gaspr=False))
153         >>> assert outs[0]["ace"] == '1001_0.03c' and os.path.isfile
    ('1001_0.03c')
154         >>> assert outs[0]["xsdir"] == '1001_0.03c.xsd' and os.path.
    isfile('1001_0.03c.xsd')
155         >>> assert outs[1]["ace"] == '1001_1.03c' and os.path.isfile
    ('1001_1.03c')
156         >>> assert outs[1]["xsdir"] == '1001_1.03c.xsd' and os.path.
    isfile('1001_1.03c.xsd')
157         """
158 
159         if 33 not in smps and 31 not in smps:
160             logging.info("no perturbation coefficient was found.")
161             return
162 
163 
164         pendf = self.get_pendf(**njoy_kws)
165 
166         seqs = []
167         ids = []
168         if 31 in smps:
169             seq_nu = smps[31].iterate_xs_samples()
170             seqs.append(seq_nu)
171             ids.append("pnu")
172         if 33 in smps:
173             seq_xs = smps[33].iterate_xs_samples()
174             seqs.append(seq_xs)
175             ids.append("pxs")
176         data = dict(zip(ids, seqs))
177 
178         if processes == 1:
179             outs = {}
180 
181             while True:
182                 kws = {}
183                 for k, v in data.items():
184                     item = next(v, False)
185                     if not item:
186                         break
187                     n, s = item
188                     kws[k] = s
189                 if not item:
190                     break
```

```
191                 kws.update(**kwargs)
192                 outs[n] = endf6_perturb_worker(self.data, pendf.data,
    n, **kws)
193
194         elif processes > 1:
195             pool = mp.Pool(processes=processes)
196             outs = {}
197
198             while True:
199                 kws = {}
200                 for k, v in data.items():
201                     item = next(v, False)
202                     if not item:
203                         break
204                     n, s = item
205                     kws[k] = s
206                 if not item:
207                     break
208                 kws.update(**kwargs)
209                 outs[n] = pool.apply_async(
210                     endf6_perturb_worker,
211                     (self.data, pendf.data, n),
212                     kws,
213                     )
214
215             outs = {n: out.get() for n, out in outs.items()}
216             pool.close()
217             pool.join()
218
219         # if we keep ENDF6 and PENDF files in memory, convert them
    back into
220         # sandy Endf6 instances (must do it here because Endf6 object
     cannot be pickled)
221         if not kwargs.get("to_file", False) and not kwargs.get("
    to_ace", False):
222             outs = {k: {k1: sandy.Endf6(v1) for k1, v1 in v.items()}
    for k, v in outs.items()}
223         return outs
224
225
226
227 def endf6_perturb_worker(e6, pendf, n,
228                          pxs=None,
229                          pnu=None,
230                          plpc=None,
231                          pedistr=None,
232                          verbose=False,
233                          to_ace=False,
234                          to_file=False,
```

```
                                filename="{ZA}_{SMP}",
                                ace_kws={},
                                **kwargs):
    """


    Parameters
    ----------
    e6 : TYPE
        DESCRIPTION.
    pendf : TYPE
        DESCRIPTION.
    n : TYPE
        DESCRIPTION.
    pxs : TYPE
        DESCRIPTION.
    verbose : TYPE, optional
        DESCRIPTION. The default is False.
    to_ace : TYPE, optional
        DESCRIPTION. The default is False.
    to_file : TYPE, optional
        DESCRIPTION. The default is False.
    filename : TYPE, optional
        DESCRIPTION. The default is "{ZA}_{SMP:d}".
    ace_kws : TYPE, optional
        DESCRIPTION. The default is {}.
    **kwargs : TYPE
        DESCRIPTION.

    Returns
    -------
    TYPE
        DESCRIPTION.

    """
    # default initialization
    endf6_pert = sandy.Endf6(e6.copy())
    pendf_pert = sandy.Endf6(pendf.copy())


    # filename options, in case we write to file
    mat = endf6_pert.mat[0]
    intro = endf6_pert.read_section(mat, 1, 451)
    za = int(intro["ZA"])
    meta = int(intro["LISO"])
    zam = sandy.zam.za2zam(za, meta=meta, method=False)
    zaid = ace_kws.get("zaid", "nndc")
    if zaid == "nndc":
        za = sandy.zam.zam2za(zam, method=zaid)[0]
```

```
284     params = dict(
285         MAT=mat,
286         ZAM=zam,
287         META=meta,
288         ZA=za,
289         SMP=n,
290         )
291     fn = filename.format(**params)
292 
293     # apply nubar perturbation
294     if pnu is not None:
295         nu = sandy.Xs.from_endf6(endf6_pert.filter_by(listmt=[452,
    455, 456]))
296         nu_pert = sandy.core.xs.xs_perturb_worker(nu, n, pnu, verbose
    =verbose)
297         endf6_pert = nu_pert.reconstruct_sums(drop=True).to_endf6(
    endf6_pert).update_intro()
298 
299     # apply lpc perturbation
300     if plpc is not None:
301         pass
302 
303     # apply edistr perturbation
304     if pedistr is not None:
305         pass
306 
307     # apply xs perturbation
308     if pxs is not None:
309         xs = sandy.Xs.from_endf6(pendf_pert)
310         xs_pert = sandy.core.xs.xs_perturb_worker(xs, n, pxs, verbose
    =verbose)
311         pendf_pert = xs_pert.reconstruct_sums(drop=True).to_endf6(
    pendf_pert).update_intro()
312 
313     # Run NJOY and convert to ace
314     if to_ace:
315         temperature = ace_kws.get("temperature", 0)
316         suffix = ace_kws.get("suffix", "." + sandy.njoy.
    get_temperature_suffix(temperature))
317         ace = endf6_pert.get_ace(pendf=pendf_pert, **ace_kws)
318 
319         if to_file:
320             outfiles = {}
321             file = f"{fn}{suffix}c"
322             with open(file, "w") as f:
323                 if verbose:
324                     print(f"writing to file '{file}'")
325                 f.write(ace["ace"])
326             outfiles["ace"] = file
```

```
327             file = f"{file}.xsd"
328             with open(file, "w") as f:
329                 if verbose:
330                     print(f"writing to file '{file}'")
331                 f.write(ace["xsdir"])
332             outfiles["xsdir"] = file
333             return outfiles
334         return ace
335 
336     else:
337         out = {
338             "endf6": endf6_pert.data,
339             "pendf": pendf_pert.data,
340             }
341 
342         if to_file:
343             outfiles = {}
344             file = f"{fn}.endf6"
345             if verbose:
346                 print(f"writing to file '{file}'")
347             endf6_pert.to_file(file)
348             outfiles["endf6"] = file
349             if pendf_pert:
350                 file = f"{fn}.pendf"
351                 if verbose:
352                     print(f"writing to file '{file}'")
353                 pendf_pert.to_file(file)
354                 outfiles["pendf"] = file
355             return outfiles
356 
357         return out
```

B.3 A first trial: energy distributions (not present in the official release yet)

B.3.1 The class Edistr

```
1 class Edistr():
2     """
3     Object to store tabulate energy distributions.
4 
5     Attributes
6     ----------
7     data : `pandas.DataFrame`
```

```
        dataframe of energy distribution data

    Methods
    -------
    filter_by
        apply condition to source data and return filtered results
    from_endf6
        extract energy distributions from `Endf6` instance
    get_integrals
        calculate the integral of each energy distribution
    normalize
        renormalize each outgoing energy distribution to 1
    to_endf6
        Given the `Edistr` instance, the `Endf6` instance is obtained
        with the relative information on the energy distribution
    """

    _indexname = ("EOUT")
    _columnsnames = ("MAT", "MT", "K", "EIN")

    def __repr__(self):
        return self.data.__repr__()

    def __init__(self, df, **kwargs):
        self.data = pd.DataFrame(df, **kwargs)

    @property
    def data(self):
        """
        Dataframe of energy distribution data with index as outgoing
    energy
        and columns as pd.MultiIndex with last level as incident
    energy

        Returns
        -------
        `pandas.DataFrame`
            tabulated energy distribution

        Notes
        -----
        .. note :: tabulated values are assumed to be interpolated
    linearly

        Examples
        --------
        >>> Edistr(minimal_edistrtest)
        MAT                  9437
        MT                     18
```

```
54          K                       0
55          EIN          1.00000e+00 2.00000e+00
56          EOUT
57          1.00000e-05 4.00000e-01 0.00000e+00
58          1.00000e-04 0.00000e+00 2.00000e-01
59          1.00000e+00 0.00000e+00 7.00000e-01
60          1.00000e+07 0.00000e+00 1.00000e-01
61          2.00000e+07 6.00000e-01 0.00000e+00
62          """
63          return self._data
64
65      @data.setter
66      def data(self, data):
67          self._data = data.rename_axis(self.__class__._indexname, axis
    =0)\
68                           .rename_axis(self.__class__._columnsnames,
    axis=1)
69          if not data.index.is_monotonic_increasing:
70              raise ValueError("energy grid is not monotonically
    increasing")
71
72      def filter_by(self, key, value):
73          """
74          Apply condition to source data and return filtered results.
75
76          Parameters
77          ----------
78          `key` : `str`
79              any label present in the columns of `data`
80          `value` : `int` or `float`
81              value used as filtering condition
82
83          Returns
84          -------
85          `sandy.Edistr`
86              filtered dataframe of energy distributions
87
88          Raises
89          ------
90          `sandy.Error`
91              if applied filter returned empty dataframe
92
93          Notes
94          -----
95          .. note:: The primary function of this method is to make sure
     that
96                    the filtered dataframe is still returned as a `
    Edistr`
97                    object.
```

```
98
99          Examples
100         --------
101         >>> Edistr(minimal_edistrtest).filter_by("EIN", 2)
102             MAT  MT  K          EIN          EOUT         VALUE
103         0  9437  18  0 2.00000e+00 1.00000e-04 2.00000e-01
104         1  9437  18  0 2.00000e+00 1.00000e+00 7.00000e-01
105         2  9437  18  0 2.00000e+00 1.00000e+07 1.00000e-01
106         """
107         condition = self.data[key] == value
108         out = self.data.copy()[condition].reset_index(drop=True)
109         if out.empty:
110             raise sandy.Error("applied filter returned empty
    dataframe")
111         return self.__class__(out)
112
113     def get_integrals(self):
114         """
115         Calculate the integral of each energy distribution.
116
117         Returns
118         -------
119         `pandas.DataFrame`
120             dataframe of integrals
121
122         Examples
123         --------
124         >>> Edistr(minimal_edistrtest).get_integrals()
125                                 INTEGRAL
126         MAT  MT K EIN
127         9437 18 0 1.00000e+00 3.00000e+06
128                   2.00000e+00 4.50000e+06
129         """
130         data = self.data
131         integrals = data.apply(lambda x: np.trapz(x, x.index))
132         df = integrals.to_frame(name="INTEGRAL")
133         return df
134     def normalize(self):
135         """
136         Renormalize each outgoing energy distribution to 1.
137
138         Returns
139         -------
140         `sandy.Edistr`
141             renormalized energy distributions
142
143         Examples
144         --------
145         >>> new = Edistr(minimal_edistrtest).normalize()
```

```
        >>> new
        MAT                  9437
        MT                     18
        K                       0
        EIN           1.00000e+00  2.00000e+00
        EOUT
        1.00000e-05  1.33333e-07  0.00000e+00
        1.00000e-04  0.00000e+00  4.44444e-08
        1.00000e+00  0.00000e+00  1.55556e-07
        1.00000e+07  0.00000e+00  2.22222e-08
        2.00000e+07  2.00000e-07  0.00000e+00

        >>> new.get_integrals()
                                      INTEGRAL
        MAT  MT K EIN
        9437 18 0  1.00000e+00  1.00000e+00
                   2.00000e+00  1.00000e+00
        """
        integrals = self.get_integrals()
        data = self.data
        b = np.array([[val]*data.shape[0]
                      for val in integrals.values.squeeze()]).T
        df = pd.DataFrame(data.values/b,
                          index=data.index,
                          columns=data.columns)
        return self.__class__(df)

    def _perturb(self, s):
        """
        Apply perturbations to energy distributions.

        Parameters
        ----------
        s : `pandas.DataFrame` or :func:`~sandy.Samples`
            input perturbations or samples.
            If `s` is a `pandas.DataFrame`, its index and columns must have the
            same names and structure as in `self.data`.

            .. note:: the energy grid of `s` must be multigroup, i.e.,
                      rendered by a (right-closed) `pd.IntervalIndex`.

            If `s` is a :func:`~sandy.Samples` instance

        Returns
        -------
        Edistr : :func:`~Edistr` or `dict` of :func:`~Edistr`
            perturbed chi object if `s` is a `pandas.DataFrame`,
```

```
193             otherwise dictionary of perturbed chi objects with
194             sample numbers as key.
195
196         Examples
197         --------
198         Get plutonium energy distributions
199         >>> e6 = sandy.get_endf6_file("jeff_33", "xs", 942390)
200         >>> edistr = sandy.Edistr.from_endf6(e6)
201
202         Apply multiplication coefficient equal to 1 to outgoing
    energy up to 3e7 eV (upper xs energy limit)
203         and incidnet energy of 2e6 eV.
204         >>> index = pd.IntervalIndex.from_breaks([9e-6, 3e7], name="
    EOUT", closed="right")
205         >>> columns = pd.MultiIndex.from_product([[9437], [18], [0],
    [2e6]], names=["MAT", "MT", "K", "EIN"])
206         >>> s = pd.DataFrame(1, index=index, columns=columns)
207         >>> ep = edistr._perturb(s)
208         >>> assert ep.data.equals(edistr.data)
209
210         Apply multiplication coefficients equal to 1.20 to outgoing
    energy up to 3e7 eV (upper xs energy limit)
211         and incidnet energy of 2e6 eV.
212         >>> s = pd.DataFrame(1.20, index=index, columns=columns)
213         >>> ep = edistr._perturb(s)
214         >>> assert not ep.data.equals(edistr.data)
215         >>> assert ep.data.loc[:, ep.data.columns != (9437, 18, 0, 2
    e6)].equals(edistr.data.loc[:, edistr.data.columns != (9437, 18,
    0, 2e6)])
216         >>> assert ep.data[(9437, 18, 0, 2e6)].equals(edistr.data
    [(9437, 18, 0, 2e6)] * 1.20)
217
218         """
219         x = self.data
220
221         # reshape indices (energy)
222         idx = s.index.get_indexer(x.index)
223         # need to copy, or else it returns a view
224         # seed https://pandas.pydata.org/pandas-docs/stable/
    user_guide/indexing.html#returning-a-view-versus-a-copy
225         s_ = s.iloc[idx].copy()
226         s_.index = x.index
227         s_.loc[idx < 0, :] = 1.  # idx = -1 indicates out of range
    lines
228
229         # reshape columns (MAT and MT)
230         idx = s_.columns.get_indexer(x.columns)
231         s_ = s_.iloc[:, idx].copy()
232         s_.columns = x.columns
```

```
233         s_.loc[:, idx < 0] = 1.  # idx = -1 indicates out of range
    lines
234
235         return self.__class__(s_ * x)
236
237     @classmethod
238     def from_endf6(cls, endf6):
239         """
240         Extract energy distributions from `Endf6` instance.
241
242         Parameters
243         ----------
244         endf6 : `sandy.Endf6`
245             object containing the ENDF-6 text
246
247         Returns
248         -------
249         `sandy.Edistr`
250             object with tabulated energy distributions
251
252         Show content of `sandy.Edistr` instance.
253         >>> e6 = sandy.get_endf6_file("jeff_33", "xs", 942390)
254         >>> sandy.Edistr.from_endf6(e6).data.head().iloc[:, 1:5]
255         MAT                  9437
256         MT                     18
257         K                       0
258         EIN           5.00000e+02 1.00000e+03 1.00000e+04 5.00000e+04
259         EOUT
260         1.00000e-05 1.85342e-12 1.85341e-12 1.85329e-12 1.85274e-12
261         2.00000e-05 2.62113e-12 2.62112e-12 2.62095e-12 2.62017e-12
262         4.00000e-05 3.70684e-12 3.70683e-12 3.70658e-12 3.70548e-12
263         6.00000e-05 4.53994e-12 4.53992e-12 4.53962e-12 4.53827e-12
264         8.00000e-05 5.24227e-12 5.24225e-12 5.24190e-12 5.24035e-12
265         """
266         tape = endf6.filter_by(listmf=[5])
267         for mat, mf, mt in tape.data:
268             sec = tape.read_section(mat, mf, mt)
269             for k, pdistr in sec["PDISTR"].items():
270                 if pdistr["LF"] != 1:
271                     msg = "non-tabulated distribution for " +\
272                           f"MAT{mat}/MF{mf}/MT{mt}, subsection {k}"
273                     logging.warning(msg)
274                     continue
275                 if list(filter(lambda x: x["INT"] != [2], pdistr["EIN
    "].values())):
276                     msg = "found non-linlin interpolation, skip " +\
277                           f"distribution for MAT{mat}/MF{mf}/MT{mt},"
    +\
278                           f" subsection {k}"
```

```
279                     logging.warning(msg)
280                     continue
281                 data = []
282                 for ein, v in sorted(pdistr["EIN"].items()):
283                     series  = pd.Series(v["EDISTR"],
284                                         index=v["EOUT"],
285                                         name=(mat, mt, k, ein)).
    to_frame()
286                     data.append(series)
287         if not data:
288             raise sandy.Error("no tabulated energy distribution was
    found")
289         df = pd.concat(data, axis=1).interpolate(method='slinear',
    axis=0) \
290                        .fillna(0)
291         return cls(df)
292     def to_endf6(self, endf6):
293         """
294         Update energy distributions in `Endf6` instance with those
    available
295         in a `Edistr` instance.
296 
297         .. warning:: only chi with `(MAT,MT)` combinations that are
    originally
298                      present in the `Endf6` instance are modififed,
    the others
299                      are discarded.
300                      The reason behind this is that to reconstruct a
    endf6
301                      section we need info that is not available in the
     `Edistr`
302                      instance itself.
303 
304         Parameters
305         ----------
306         `endf6` : `sandy.Endf6`
307             `Endf6` instance
308 
309         Returns
310         -------
311         `sandy.Endf6`
312             `Endf6` instance with updated chi
313         """
314         data = endf6.data.copy()
315         mf = 5
316         for (mat, mt, k), edistr in self.data.groupby(axis=1, level
    =[0,1,2]):
317             # Must read original section to extract info not given in
     `Xs`
```

```
318             # instance, e.g. QI, QM
319             frame = edistr[mat, mt, k]
320             if (mat, mf, mt) not in endf6.keys:
321                 continue
322             sec = endf6.read_section(mat, mf, mt)
323             sec["PDISTR"][k]["NBT_EIN"] = [len(frame.columns.unique()
    )]
324             sec["PDISTR"][k]["E_P"] = frame.columns.unique().values
    [[0,-1]]
325             for ein, fk in frame.iteritems():
326                 sec["PDISTR"][k]["EIN"][ein]["EOUT"] = fk.index.
    values
327                 sec["PDISTR"][k]["EIN"][ein]["EDISTR"] = fk.values
328                 sec["PDISTR"][k]["EIN"][ein]["NBT"] = [len(fk.index)]
329             data[mat, mf, mt] = sandy.mf5.write_mf5(sec)
330         return sandy.Endf6(data)
```

B.3.2 Write the sections MF=5

```
1  def write_mf5(sec):
2      """Write MT section for MF5
3
4      Parameters
5      ----------
6      sec : `sandy.utils.Section`
7          dictionary with MT section for MF5
8
9      Returns
10     -------
11     `str`
12         section content in a single string
13     """
14     text = sandy.write_cont(sec["ZA"], sec["AWR"], 0, 0, len(sec["
    PDISTR"]), 0)
15     for k, sub in sorted(sec["PDISTR"].items()):
16         U = sub['U'] if 'U' in sub else 0
17         text += sandy.write_tab1(U, 0, 0, sub["LF"], sub["NBT_P"],
    sub["INT_P"], sub["E_P"], sub["P"])
18         if sub["LF"] == 1:
19             text += sandy.write_tab2(0, 0, 0, 0, len(sub['EIN']), sub
    ["NBT_EIN"], sub["INT_EIN"])
20             for ein, distr in sorted(sub['EIN'].items()):
21                 text += sandy.write_tab1(0, ein, 0, 0, distr["NBT"],
    distr["INT"], distr["EOUT"], distr["EDISTR"])
22         elif sub["LF"] == 5:
```

```
23              text += sandy.write_tab1(0, 0, 0, 0, sub["NBT_THETA"],
    sub["INT_THETA"], sub["E_THETA"], sub["THETA"])
24              text += sandy.write_tab1(0, 0, 0, 0, sub["NBT_G"], sub["
    INT_G"], sub["E_G"], sub["G"])
25          elif sub["LF"] in (7,9):
26              text += sandy.write_tab1(0, 0, 0, 0, sub["NBT_THETA"],
    sub["INT_THETA"], sub["E_THETA"], sub["THETA"])
27          elif sub["LF"] == 11:
28              text += sandy.write_tab1(0, 0, 0, 0, sub["NBT_A"], sub["
    INT_A"], sub["E_A"], sub["A"])
29              text += sandy.write_tab1(0, 0, 0, 0, sub["NBT_B"], sub["
    INT_B"], sub["E_B"], sub["B"])
30          elif sub["LF"] == 12:
31              text += sandy.write_tab1(0, 0, 0, 0, sub["NBT_TM"], sub["
    INT_TM"], sub["E_TM"], sub["TM"])
32      textout = []
33      iline = 1
34      for line in text:
35          if iline > 99999:
36              iline = 1
37          textout.append("{:<66}{:4}{:2}{:3}{:5}\n".format(line, sec["
    MAT"], sec["MF"], sec["MT"], iline))
38          iline += 1
39      return "".join(textout)
```

B.3.3 Modified get_perturbations

```
1       def get_perturbations(
2           self,
3           nsmp,
4           to_excel=None,
5           njoy_kws={},
6           smp_kws={},
7           **kwargs,
8           ):
9           """
10          Construct multivariate distributions with a unit vector for
11          mean and with relative covariances taken from the evaluated
    files
12          processed with the NJOY module ERRORR.
13
14          Perturbation factors are sampled with the same multigroup
    structure of
15          the covariance matrix and are returned by nuclear datatype as
     a `dict`
16          of `pd.Dataframe` instances .
```

```
17
18          Parameters
19          ----------
20          nsmp : TYPE
21              DESCRIPTION.
22          to_excel : TYPE, optional
23              DESCRIPTION. The default is None.
24          njoy_kws : TYPE, optional
25              DESCRIPTION. The default is {}.
26          smp_kws : TYPE, optional
27              DESCRIPTION. The default is {}.
28          **kwargs : TYPE
29              DESCRIPTION.
30
31          Returns
32          -------
33          smp : TYPE
34              DESCRIPTION.
35
36          Examples
37          --------
38          Generate a couple of samples from the H1 file of JEFF-3.3.
39          >>> njoy_kws = dict(err=1, errorr_kws=dict(mt=102))
40          >>> tape = sandy.get_endf6_file("jeff_33", "xs", 10010)
41          >>> smps = tape.get_perturbations(nsmp=2, njoy_kws=njoy_kws)
42          >>> assert len(smps) == 1
43          >>> assert isinstance(smps[33], sandy.Samples)
44          >>> assert (smps[33].data.index.get_level_values("MT") ==
    102).all()
45          """
46          smp = {}
47          seeds = {}
48
49          outs = self.get_errorr(**njoy_kws)
50
51          if "errorr31" in outs:
52              smp_kws["seed"] = seed = smp_kws.get("seed31", sandy.
    get_seed())
53              seeds["errorr31"] = seed
54              smp[31] = outs["errorr31"].get_cov().sampling(nsmp, **
    smp_kws)
55          if "errorr33" in outs:
56              smp_kws["seed"] = seed = smp_kws.get("seed33", sandy.
    get_seed())
57              seeds["errorr33"] = seed
58              smp[33] = outs["errorr33"].get_cov().sampling(nsmp, **
    smp_kws)
59          if "errorr35" in outs:
```

```
60              smp_kws["seed"] = seed = smp_kws.get("seed35", sandy.
    get_seed())
61              seeds["errorr35"] = seed
62              njoy_kws["xs"] = njoy_kws["nubar"] = njoy_kws["mubar"] =
    False
63              mf35_eg = self.get_incident_ene()
64              frames = []
65              for ifissp, ein in enumerate(mf35_eg, 1):
66                  errorr = self.get_errorr(errorr35_kws=dict(ifissp=
    ifissp), **njoy_kws)
67                  data = errorr["errorr35"].get_cov().sampling(nsmp, **
    smp_kws).data
68                  data.index = pd.MultiIndex.from_product([self.mat,
    [18], [0], [ein], data.index.get_level_values(2).unique()],
69                                                          names=[*
    sandy.Edistr._columnsnames, sandy.Edistr._indexname])
70                  frames.append(data)
71              smp[35] = sandy.Samples(pd.concat(frames))
72          if to_excel and smp:
73              with pd.ExcelWriter(to_excel) as writer:
74                  for k, v in smp.items():
75                      v.to_excel(writer, sheet_name=f'MF{k}')
76          return smp
```

Bibliography

[1] International Energy Agency IEA. *Nuclear Power and Secure Energy Transitions - From today's challenges to tomorrow's clean energy systems*. Tech. rep. CC BY 4.0. Paris, June 2022. URL: https://www.iea.org/reports/nuclear-power-and-secure-energy-transitions (cit. on pp. 1, 2).

[2] International Energy Agency (IEA). *Net Zero by 2050: A Roadmap for theGlobal Energy Sector*. 2021, p. 224. DOI: https://doi.org/https://doi.org/10.1787/c8328405-en. URL: https://www.oecd-ilibrary.org/content/publication/c8328405-en (cit. on p. 1).

[3] «Neutron Chain Fission Reactors». In: *Nuclear Reactor Physics*. John Wiley and Sons, Ltd, 2007. Chap. 2, pp. 33–42. ISBN: 9783527611041. DOI: https://doi.org/10.1002/9783527611041.ch2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9783527611041.ch2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9783527611041.ch2 (cit. on p. 2).

[4] R. Kohale. *Fundamentals of Nuclear Physics*. Bentham Science Publishers, 2023. ISBN: 9789815049916. URL: https://books.google.it/books?id=Vr_KEAAAQBAJ (cit. on p. 2).

[5] J.R. Lamarsh and A.J. Baratta. *Introduction to Nuclear Engineering*. Addison-Wesley series in nuclear science and engineering. Prentice Hall, 2011. ISBN: 9780132764575. URL: https://books.google.it/books?id=wEk1KQEACAAJ (cit. on pp. 2, 7).

[6] D.L. Smith. «Nuclear Data Uncertainty Quantification: Past, Present and Future». In: *Nuclear Data Sheets* 123 (2015). Special Issue on International Workshop on Nuclear Data Covariances April 28 - May 1, 2014, Santa Fe, New Mexico, USA http://t2.lanl.gov/cw2014, pp. 1–7. ISSN: 0090-3752. DOI: https://doi.org/10.1016/j.nds.2014.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0090375214006814 (cit. on pp. 3, 17).

[7] Michael Fleming Luca Fiorito James Dyrda. *JEFF-3.3 covariance application to ICSBEP using SANDY and NDaST*. Tech. rep. NEA: Nuclear Energy Agency, 2019. DOI: https://doi.org/10.1051/epjconf/201921107003 (cit. on p. 3).

[8] Luca Fiorito. *SANDY Documentation*. 2018. URL: https://luca-fiorito-11.github.io/sandy-docs/index.html (Accessed: 13 Sept. 2023) (cit. on pp. 4, 42, 43, 45, 46).

[9] Oscar Cabellos. *Handbook on Nuclear Data*. Ed. by Great-Pioneer EU Project. Sept. 2022 (cit. on pp. 5, 6, 10, 12, 14, 15, 18, 21, 27, 32, 33, 36, 39–41, 49, 94, 96).

[10] G.I. Bell and S. Glasstone. *Nuclear Reactor Theory*. Van Nostrand Reinhold Company, 1970. URL: https://books.google.it/books?id=RNQmAQAAMAAJ (cit. on p. 6).

[11] N.M. Schaeffer, U.S. Atomic Energy Commission. Division of Reactor Development, and Technology. *Reactor Shielding for Nuclear Engineers. N.M. Schaeffer, Editor*. National Technical Information Service, 1973. URL: https://books.google.it/books?id=GtZOHAAACAAJ (cit. on p. 6).

[12] «Neutron Diffusion Theory». In: *Nuclear Reactor Physics*. John Wiley and Sons, Ltd, 2007. Chap. 3, pp. 43–100. ISBN: 9783527611041. DOI: https://doi.org/10.1002/9783527611041.ch3. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9783527611041.ch3. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9783527611041.ch3 (cit. on p. 6).

[13] «Neutron Transport Theory». In: *Nuclear Reactor Physics*. John Wiley and Sons, Ltd, 2007. Chap. 9, pp. 303–383. ISBN: 9783527611041. DOI: https://doi.org/10.1002/9783527611041.ch9. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9783527611041.ch9. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9783527611041.ch9 (cit. on p. 7).

[14] Jerzy Cetnar. «General solution of Bateman equations for nuclear transmutations». In: *Annals of Nuclear Energy* 33.7 (2006), pp. 640–645. ISSN: 0306-4549. DOI: https://doi.org/10.1016/j.anucene.2006.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0306454906000284 (cit. on p. 8).

[15] B. Pritychenko, E. Běták, M.A. Kellett, B. Singh, and J. Totans. «The Nuclear Science References (NSR) database and Web Retrieval System». In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 640.1 (2011), pp. 213–218. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2011.

03.018. URL: https://www.sciencedirect.com/science/article/pii/S0168900211005584 (cit. on p. 11).

[16] Semkova, Valentina, Otuka, Naohiko, Mikhailiukova, Marina, Pritychenko, Boris, and Cabellos, Oscar. «EXFOR - a global experimental nuclear reaction data repository: Status and new developments». In: *EPJ Web Conf.* 146 (2017). DOI: 10.1051/epjconf/201714607003. URL: https://doi.org/10.1051/epjconf/201714607003 (cit. on p. 12).

[17] M.B. Chadwick et al. «ENDF/B-VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technology». In: *Nuclear Data Sheets* 107.12 (Dec. 2006), pp. 2931–3060. DOI: 10.1016/j.nds.2006.11.001 (cit. on p. 12).

[18] Pablo Romojaro Otero. «Nuclear Data Analyses for Improving the Safety of Advanced Lead-Cooled Reactors». PhD thesis. Polytechnic University of Madrid, 2019 (cit. on pp. 12, 18–20, 30, 32, 35, 39).

[19] Uichiro Yoshimura. «The Nuclear Energy Outlook—A New Book From the OECD Nuclear Energy Agency». In: *Health Physics* 100.1 (2011), pp. 14–19 (cit. on pp. 13, 50, 51).

[20] Arjan Koning, Stephane Hilaire, and M Duijvestijn. «TALYS-1.2 A nuclear reaction program, User manual». In: *NRG, Netherlands* (Jan. 2009), pp. 16–18 (cit. on p. 13).

[21] Muir D.W, Boicourt R.M., Kahler A.C., Conlin J.L., and Haeck W. *The NJOY Nuclear Data Processing System*. Version 2016. Los Alamos National Laboratory. Nov. 2019 (cit. on pp. 14, 20, 22).

[22] Jaakko Leppänen. *Serpent – a Continuous-energy Monte CarloReactor Physics Burnup Calculation Code*. (V.T.T.) Finnish Technical Research Centre. June 2015 (cit. on pp. 14, 17, 63).

[23] C. L. Dunford. «ENDF Utility Codes». In: 7.01/02. Apr. 2005 (cit. on p. 15).

[24] M Dunn and N Greene. «AMPX-2000: A cross-section processing system for generating nuclear data for criticality safety applications». In: *Trans. Am. Nucl. Soc.* 86 (Jan. 2002) (cit. on p. 15).

[25] Dermott E. Cullen. «PREPRO 2023: 2023 ENDF/B Pre-processing Codes"». In: *Nuclear Data Services (International Atomic Energy Agency)* (June 2023) (cit. on p. 15).

[26] Kenichi Tada, Akio Yamamoto, Satoshi Kunieda, Chikara Konno, Ryoichi Kondo, Tomohiro Endo, Go Chiba, Michitaka Ono, and Masayuki Tojo. «Development of nuclear data processing code FRENDY version 2». In: *Journal of Nuclear Science and Technology* (2023), pp. 1–10. DOI: 10.1080/00223131.2023.2278600. eprint: https://doi.org/10.1080/00223131.2023.2278600. URL: https://doi.org/10.1080/00223131.2023.2278600 (cit. on p. 15).

[27] John D Bess, Tatiana Ivanova, Lori Scott, and Ian Hill. «International Handbook of Evaluated Criticality Safety Benchmark Experiments». In: *7952* (2021). Ed. by Nuclear Energy Agency (NEA). URL: https://www.oecd-nea.org/jcms/pl_20291/icsbep-handbook (cit. on pp. 15, 50–58, 60–62).

[28] Marc Wenger et al. «The SIMBAD astronomical database». In: *Astronomy and Astrophysics Supplement Series* 143 (Mar. 2000). DOI: 10.1051/aas:2000332 (cit. on p. 15).

[29] Ian Hill. *The Nuclear Data Sensitivity Tool (NDaST) 'How to' Beginner's Guide*. 1.6. Jan. 2022 (cit. on pp. 16, 48–50).

[30] Andrej Trkov. «From Basic Nuclear Data to Applications». In: Jozef Stefan Institute, Lubiana, Slovenia, Apr. 2000 (cit. on p. 16).

[31] Paul K. Romano, Nicholas E. Horelik, Bryan R. Herman, Adam G. Nelson, Benoit Forget, and Kord Smith. «OpenMC: A state-of-the-art Monte Carlo code for research and development». In: *Annals of Nuclear Energy* 82 (2015), pp. 90–97. ISSN: 0306-4549. DOI: https://doi.org/10.1016/j.anucene.2014.07.048. URL: https://www.sciencedirect.com/science/article/pii/S030645491400379X (cit. on p. 17).

[32] Joel Aaron Kulesza et al. *MCNP® Code Version 6.3.0 Theory & User Manual*. Tech. rep. LA-UR-22-30006, Rev. 1. Los Alamos, NM, USA: Los Alamos National Laboratory, Sept. 2022. DOI: 10.2172/1889957. URL: https://www.osti.gov/biblio/1889957 (cit. on p. 17).

[33] Andrea Saltelli. «Sensitivity Analysis for Importance Assessment». In: *Risk Analysis* 22.3 (2002), pp. 579–590. DOI: https://doi.org/10.1111/0272-4332.00040. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/0272-4332.00040. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/0272-4332.00040 (cit. on p. 17).

[34] Herman M and Trkov A. *ENDF-6 Formats Manual*. Tech. rep. Brookhaven National Lab. (BNL), Upton, NY (United States), July 2010. URL: www.nndc.bnl.gov (cit. on p. 19).

[35] Mukesh Srivastava. «Analysis of Variance and Covariance». In: *Journal of Applied Statistics* 37.10 (2010), pp. 1781–1782. DOI: 10.1080/02664760902919747. eprint: https://doi.org/10.1080/02664760902919747 (cit. on pp. 22, 23).

[36] Nimra Ejaz. *What's the Difference Between Covariance and Correlation?* 2023. URL: https://careerfoundry.com/en/blog/data-analytics/covariance-vs-correlation (Accessed: 18 Nov. 2023) (cit. on pp. 23, 25).

[37] M-C Casabán, J-C Cortés, J-V Romero, and M-D Roselló. «Probabilistic solution of random SI-type epidemiological models using the Random Variable Transformation technique». In: *Communications in Nonlinear Science and Numerical Simulation* 24.1-3 (2015), pp. 86–97 (cit. on p. 24).

[38] «Chapter 7 The Variance-Covariance Matrix». In: *Experimental Design: A Chemometric Approach*. Ed. by Stanley N. Deming and Stephen L. Morgan. Vol. 3. Data Handling in Science and Technology. Elsevier, 1987, pp. 105–116. DOI: https://doi.org/10.1016/S0922-3487(08)70278-8. URL: https://www.sciencedirect.com/science/article/pii/S0922348708702788 (cit. on pp. 24, 25).

[39] Talou, Patrick. «Evaluating nuclear data and their uncertainties». In: *EPJ Nuclear Sci. Technol.* 4 (2018), p. 29. DOI: 10.1051/epjn/2018032. URL: https://doi.org/10.1051/epjn/2018032 (cit. on p. 27).

[40] S. Lahaye, T. D. Huynh, and A. Tsilanizara. «Comparison of deterministic and stochastic approaches for isotopic concentration and decay heat uncertainty quantification on elementary fission pulse». In: *European Physical Journal Web of Conferences*. Vol. 111. European Physical Journal Web of Conferences. Mar. 2016, 09002, p. 09002. DOI: 10.1051/epjconf/201611109002 (cit. on p. 30).

[41] Majdi I Radaideh and Mohammad I Radaideh. «Application of stochastic and deterministic techniques for uncertainty quantification and sensitivity analysis of energy systems». In: *arXiv preprint arXiv:1901.05566* (2019) (cit. on p. 30).

[42] Ivo Kodeli. «Sensitivity analysis and uncertainty propagation from basic nuclear data to reactor physics and safety relevant parameters». In: *Report OECD Nuclear Energy Agency (NEA), https://www. oecd-nea. org/nsd/reports/2007/nea6053/Sessi II-Methods-for-Uncertainty-Assessment/Paper-5_kodeli. pdf* (2007) (cit. on p. 31).

[43] Nicolò Abrate, Sandra Dulla, and Piero Ravetto. «Generalized perturbation techniques for uncertainty quantification in lead-cooled fast reactors». In: *Annals of Nuclear Energy* 164 (2021), p. 108623 (cit. on p. 34).

[44] Emilio Castro González. «Methodologies for Sensitivity/Uncertainty Analysis using Reactor Core Simulations with Application to Pressurized Water Reactors». PhD thesis. Polytechnic University of Madrid, 2018 (cit. on p. 34).

[45] Gildas Mazo. «The Sobol method in sensitivity analysis for stochastic computer models». working paper or preprint. Apr. 2019. URL: https://hal.science/hal-02113448 (cit. on p. 35).

[46] Bradley T. Rearden and Matthew Anderson Jessee. «SCALE Code System». In: (Mar. 2018). DOI: 10.2172/1426571. URL: https://www.osti.gov/biblio/1426571 (cit. on p. 36).

[47] Andy Rivas, Nicolas P. Martin, Samuel E. Bays, Giuseppe Palmiotti, Zhiwen Xu, and Jason Hou. «Nuclear data uncertainty propagation applied to the versatile test reactor conceptual design». In: *Nuclear Engineering and Design* 392 (2022), p. 111744. ISSN: 0029-5493. DOI: https://doi.org/10.1016/j.nucengdes.2022.111744. URL: https://www.sciencedirect.com/science/article/pii/S002954932200098X (cit. on p. 38).

[48] Fiorito Luca Cabellos Oscar. «Examples of Monte Carlo techniques applied for nuclear data uncertainty propagation». In: OECD (Nuclear Energy Agency). Department of Energy Engineering, University Polythecnic of Madrid, 28006 Madrid, Spain: EPJ Web of Conferences, 2019. DOI: 10.1051. URL: https://doi.org/10.1051/epjconf (cit. on p. 40).

[49] Oliver Buss, Axel Hoefer, and Jens Neuber. «NUDUNA, Nuclear Data Uncertainty Analysis». In: Apr. 2013 (cit. on p. 42).

[50] Zwermann, W., Gallner, L., Klein, M., Krzykacz-Hausmann, B., Pasichnyk, I., Pautz, A., and Velkov, K. «Status of XSUSA for Sampling Based Nuclear Data Uncertainty and Sensitivity Analysis». In: *EPJ Web of Conferences* 42 (2013). DOI: 10.1051/epjconf/20134203003. URL: https://doi.org/10.1051/epjconf/20134203003 (cit. on p. 42).

[51] Maarten Ooms Lucia Popescu Hamid Aït Abderrahim and Koen Hasaers. «The Belgian Nuclear Research Centre (SCK CEN)». In: *Nuclear Physics News* 32.2 (2022), pp. 4–8. DOI: 10.1080/10619127.2022.2062995. eprint: https://doi.org/10.1080/10619127.2022.2062995. URL: https://doi.org/10.1080/10619127.2022.2062995 (cit. on p. 42).

[52] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697 (cit. on pp. 42, 64).

[53] L. Fiorito, G. Žerovnik, A. Stankovskiy, G. Van den Eynde, and P.E. Labeau. «Nuclear data uncertainty propagation to integral responses using SANDY». In: *Annals of Nuclear Energy* 101 (Mar. 2017), pp. 359–366. DOI: 10.1016/j.anucene.2016.11.026 (cit. on pp. 42, 43).

[54] Enrica Belfiore. «Sensitivity and uncertainty analysis for nuclear data of relevance in spent nuclear fuel characterization». MA thesis. Politecnico di Torino, Mar. 2023 (cit. on p. 46).

[55] J. Blair Briggs Ali Nouri Pierre Nagel and Tatiana Ivanova. «DICE: Database for the International Criticality Safety Benchmark Evaluation Program Handbook». In: *Nuclear Science and Engineering* 145.1 (May 2003). DOI: 10.13182/NSE03-15. eprint: https://doi.org/10.13182/NSE03-15 (cit. on p. 49).

[56] Ian Hill. *IDAT User Manual*. 1.7. May 2021. URL: https://www.oecd-nea.org/jcms/pl_20296/international-reactor-physics-handbook-database-and-analysis-tool-idat (cit. on p. 49).

[57] G. Coddens. «The NEA Data Bank». In: *Nuclear Data for Science and Technology*. Ed. by K. H. Böckhoff. Dordrecht: Springer Netherlands, 1983, pp. 993–996 (cit. on p. 51).

[58] Harold F. McFarlane et al. «Analysis and Evaluation of ZPPR Critical Experiments for a 100 Kilowatt-Electric Space Reactor». In: (Apr. 1990). URL: https://www.osti.gov/biblio/7270789 (cit. on p. 60).

[59] Vance W. Berger and YanYan Zhou. «Kolmogorov–Smirnov Test: Overview». In: *Wiley StatsRef: Statistics Reference Online*. John Wiley and Sons, Ltd, 2014. ISBN: 9781118445112. DOI: https://doi.org/10.1002/9781118445112.stat06558. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat06558. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat06558 (cit. on pp. 66, 75).

[60] «Chi-Square Test». In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 77–79. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_57. URL: https://doi.org/10.1007/978-0-387-32833-1_57 (cit. on pp. 73, 74).

[61] Douglas Curran-Everett. «Explorations in statistics: confidence intervals». In: *Advances in physiology education* 33.2 (2009), pp. 87–90 (cit. on p. 73).

[62] David J.Sheskin. *Handbook of PARAMETRIC and NONPARAMETRIC STATISTICAL PROCEDURES SECOND EDITION*. Ed. by CHAPMAN and HALL/CRC. Second. Western Connecticut State University, 2000 (cit. on pp. 73–75).

[63] Pauli Virtanen et al. «SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python». In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 75).

[64] Laura Swiler and G. Wyss. «A User's Guide to Sandia's Latin Hypercube Sampling Software: Lhs Unix Library/Standalone Version». In: *SAND Report* (July 2004) (cit. on p. 76).

[65] Art B. Owen. «Monte Carlo theory, methods and examples». In: Stanford University, 2013. Chap. Advanced variance reduction (chapter 10). URL: `https://artowen.su.domains/mc/Ch-var-adv.pdf` (cit. on p. 76).