KANT, the ASNR Probabilistic Software for Level 2 PSA

Gaëtan Bayle-Ruault des Courchamps[a], Nadia Rahni[a] and Emmanuel Raimond[a]
[a] ASNR, Fontenay-aux-Roses, France
gaetan.bayleruaultdescourchamps@asnr.fr, nadia.rahni@asnr.fr, emmanuel.raimond@asnr.fr

Abstract: ASNR (previously IRSN) has developed and maintained its probabilistic software KANT since the end of the 90s, to fulfil the needs of the ASNR level 2 PSA project.
KANT underwent major changes in 2025, with a rewrite of its graphical user interface, and shortened load times thanks to the adoption of a much faster interpreter for its runtime language.
The paper provides an overview of the current state of KANT, mainly regarding its modeling capabilities and its graphical interface features.

1. INTRODUCTION

ASNR have been developing level 2 probabilistic safety assessment (L2 PSAs) for the French PWRs since the end of 90s. These PSAs provide a risk ranking by combining the severe accident scenario frequencies with the radiological consequences they induce.

The ASNR probabilistic software KANT is used to develop L2 PSAs. It has been designed as a user-friendly tool allowing users to:

- integrate all information from L1 PSA;
- implement representative information about accident scenarios;
- relate the chronology of the severe accident and integrate physical, human errors and systems models;
- model all dependencies between events (physical, systems activations and failures);
- trace the value of any variable during the severe accident, such as physical variables (pressure, volume, mass, etc.) that describe the plant state;
- integrate uncertainties and perform uncertainty analysis by Monte-Carlo simulations;
- consider all the containment failure modes;
- quantify the frequency and the radiological consequences of severe accident sequences.

From its early versions dating back to 1997, KANT has been adapted to the evolution of hardware, software and users' expectations, while maintaining compatibility with its previous versions.

The paper first describes the main features of KANT. It gives also an overview of the KANT language and its exploration algorithm, an overview of the KANT Graphical User Interface (GUI) and introduces some of the next challenges and expectations for KANT.

2. MAIN SPECIFICITIES OF KANT

2.1 The L1-L2 PSA Interface

ASNR uses RiskSpectrum PSA software for the L1 PSA, which is appropriate for the systems modeling. Consequently, the only link between L1 PSA and L2 PSA events trees is the list of Plant Damage States (PDSs). PDSs are for L2 PSA equivalent to initiating events for L1 PSA, i.e. they are the initiating events in the L2 PSA event tree, but they include much more information.

Each PDS is associated to a frequency and is described by means of "PDS variables" providing all needed information from L1 PSA (initiating events, system availability and human action status etc.). ASNR developed a dedicated tool [1] to generate a matrix of PDS from the list of L1 PSA minimal cutsets. Each PDS variable has a user-defined set of possible integer values (for example, the Table 1 presents the PDS variable dedicated to the High Pressure Safety Injection system).

Table 1. Example of attributes for the interface variable HPSI

HPSI values	HPSI description
1	Available and started by the operators
2	Started by the operators and available until the switch in recirculation mode
3	Available but the operators have failed to start it (human error)
4	Available until the switch in recirculation mode but the operators failed to start it (human error)
5	Not available

The user describes all the PDS variables (names and allowed values) using the KANT GUI, and KANT then reads the PDS matrix (the columns of the matrix are the PDS numbers, the PDS interface variables attributes, the PDS frequency).

Around 50 PDS variables are defined for French PWRs L2 PSA, leading to large set of PDSs (around 10000).

KANT and its language provide a flexible enough environment to model the severe accident progression for all the accidental situations described by the PDSs.

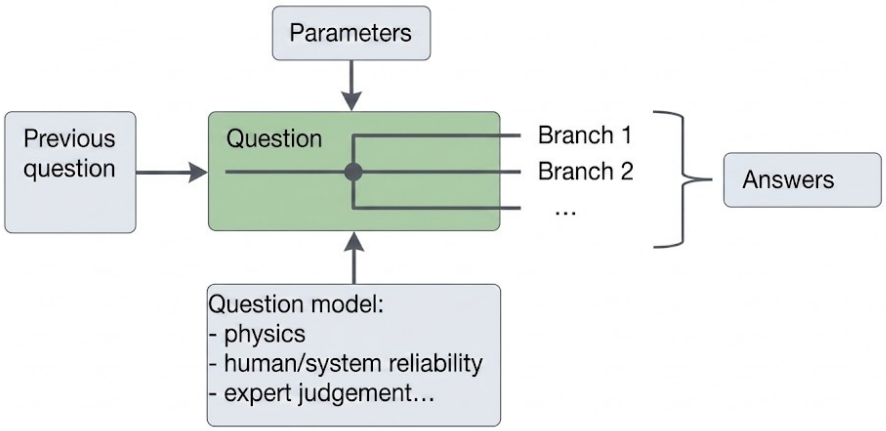
2.2 Representation of the Accident Progression Event Tree (APET) with KANT

ASNR applies the event tree formalism, which is well known and allows the L2 PSA developers to represent, in a simple and easily understandable way, the succession of events that would occur during a severe accident. The event can be binary (e.g. system activation) or more complex (e.g. conditional probability of a containment failure given some severe accident conditions).

2.2.1 Structure of the APET

The APET is structured as a succession of nodes or questions (i.e. the top events of the event tree) reflecting the chronology of the accident. Their inputs are a list of global variables (system availability, containment pressure, mass of hydrogen, containment breach size …) which describe the Nuclear Power Plant (NPP) state before the event and their outputs are, for each branch, an updated list of variables on the NPP state after the event and a conditional probability. This differs from classic event trees (where branches are defined in advance) because each node generates dynamically (during the quantification) one or several branches. The Figure 1 below shows the global structure of the APET's questions.

Figure 1: Representation of APET's questions



To clarify the structure of an APET, users can group consecutive questions into subtrees (potentially containing other subtrees). The graphical interface manages the APET, subtrees and questions by showing a foldable tree structure akin to a file hierarchy (see 4.1).

2.2.2 Physical Phenomena Modeling

KANT offers several possibilities to model physical phenomena:

- deterministic precalculated results: KANT can integrate accident simulation results from external software (for example the European severe accident integral code ASTEC) thanks to a C++ library that allows it to consult grids of results from inside the questions (results can be values at specific times, time series etc.);
- mathematical models written in the KANT language (e.g. response surfaces): the modeling is implemented in the questions themselves or in separate KANT internal functions (called by the questions);
- models through external libraries (for models that cannot be written in the KANT language or that already have an implementation in other languages): for instance, a dedicated C++ library developed for PWR L2 PSA, called MER, has been developed to assess amplitude and kinetics of radioactive releases [2]; other examples of C++ library are the modeling of hydrogen combustion phenomena (inside of PWR containment building) and the thermohydraulic modeling for a nuclear fuel reprocessing plant of La Hague.

2.2.3 Release Categories

Release categories (RCs) are formed by KANT at the end of the APET, with rules chosen so that situations leading to the same RCs are expected be associated with similar source term into the environment. A set of variables calculated through the APET (variables associated to the containment behavior, that determine the kinetics and level of release of an accident sequence) is used for the grouping of L2 PSA sequences into release categories. On the same principle as PDS variables, RC variables are defined by a set of integer values (example of RC variables: spray system availability before and after the vessel rupture, containment breach size, venting system opening ...).

The RC frequency is the sum of the associated sequence frequencies. A release calculation (amplitude and kinetics) is then performed, at the request of the user, for each RC.

Around 50 RC variables are defined for French PWRs L2 PSA, leading to approximatively 20000 RCs.

3. KANT ALGORITHM AND LANGUAGE

Since nuclear safety experts may not be experts at programming, the programming language of KANT was designed as a very simple way to represent L2 PSA concepts.

This section focusses on the parts of the language that relate to those concepts. First with an overview of the exploration algorithm used for KANT APETs, then with short descriptions of the variable types and of the syntax, and last with some considerations about the implementations of the language.

For a representative idea of the amount of KANT language in the current APETs, the current ones for French PWRs have between 80 and 100 questions, totalling 40000 lines. The additional user-created functions in the KANT Language (internal functions) fill 28000 more lines.

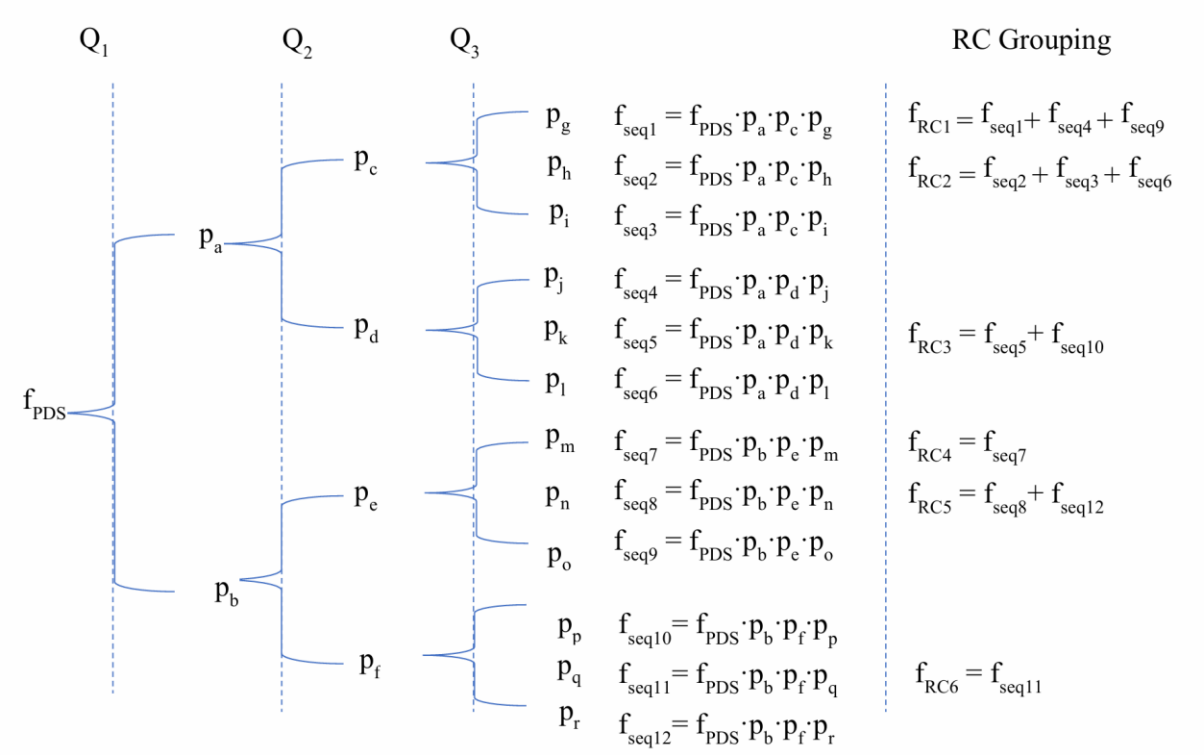
3.1 Exploration Algorithm

For each PDS, KANT explores the questions and their possible answers recursively, following the order of questions in the APET and backtracking when the frequency goes below a predefined cutoff value.

When KANT executes a question, it retrieves the conditional probabilities of the answers (filled by the KANT language instructions of the question) and it updates the frequency accordingly before moving to the next question.
Every time KANT reaches the last question, it uses the values of the RC variables to form an index. If no RC exists for this index, KANT creates a new one with the frequency of the current sequence. Otherwise, KANT just increases the frequency of the existing RC.

The Figure 2 below shows the formation of RCs for a single PDS. When KANT loops over all the PDSs and finds a sequence that leads to an existing RC, it keeps increasing the RC frequency like it did with a single PDS.

Figure 2: KANT Exploration Algorithm for a single PDS



When all the RCs have been computed, KANT can start the computation of release.

For uncertainty analysis, KANT repeats the exploration multiple times with the sampled values of the parameters that are marked as random (see 4.2.1).

It should be noted that for a given question:

- the sum of the conditional probabilities of the possible answers is always 1;
- if the contents of a question are independent from the answers to the upstream questions, the conditional probability of its answers is independent from the branches explored previously. For instance, assuming such independence for Q2 in Figure 1, we have $p_c=p_e$ and $p_d=p_f$.

3.2 Language syntax

The syntax of the KANT language is close to other simple languages. It includes:

- control instructions: if, then, else, else_if, end_if; while, end_while; for, end_for; switch, case, default, end_switch; stop;
- variable types: integer, real, boolean, string;
- arithmetic operators: +, -, *, /, =, <, <=, >, >=, <>;
- boolean logic: and, or, not, false, true;
- integrated functions: MIN, MAX, power, log etc.

The KANT language also has keywords that are more closely tied to the exploration algorithm:

- the instruction “go_to” that changes the next question is not an immediate jump, and more of an instruction for KANT to move to the mentioned question after the current one is finished. No backward jump is allowed, and if no “go_to” is encountered, the next question is just the one that follows in the APET;
- to ease the search for errors, a “break” instruction gives control to the debugger, and another instruction lists the sequence along with all the current variable values;
- for some cases that need information about the current accidental sequence, the language offers instructions like “write_sequence” that displays the whole sequence, or “answer()” that reminds what branch is being explored for a previous step of the sequence.

3.3 Implementation

Currently, KANT runs on Windows 7 and its successors. Its computation engine also runs on Linux. The engine itself has parallel implementations for Windows and Linux: they rely on the Message Passing Interface (MPI) standard [7,8].

From its first versions in 1997 up to 2025, the KANT Language was compiled into an interpreted execution language called p-code. This was inspired by the compilation courses of the time and the popular “Dragon books” about compilers [3].

However, over the years the KANT house-made interpreter has become a bottleneck, both for performance and for maintenance reasons. Thus, it was decided to shift the compilation target to a widely adopted, simple, fast and open interpreted programming language: Lua [4]. While this language may be known to the public for its use in videogaming, it is also quite popular in embedded devices, and, more importantly for KANT, it is designed to integrate easily with programs written in C and C++.

The move to Lua has been completed recently and has yielded two benefits at least:

- much faster load time when starting a computation; for current APETs, load times at startup went down from 30s to less than 1s;
- proper capabilities to implement some long-awaited features for the language, such as better support for character strings.

4. KANT GRAPHICAL USER INTERFACE (GUI)

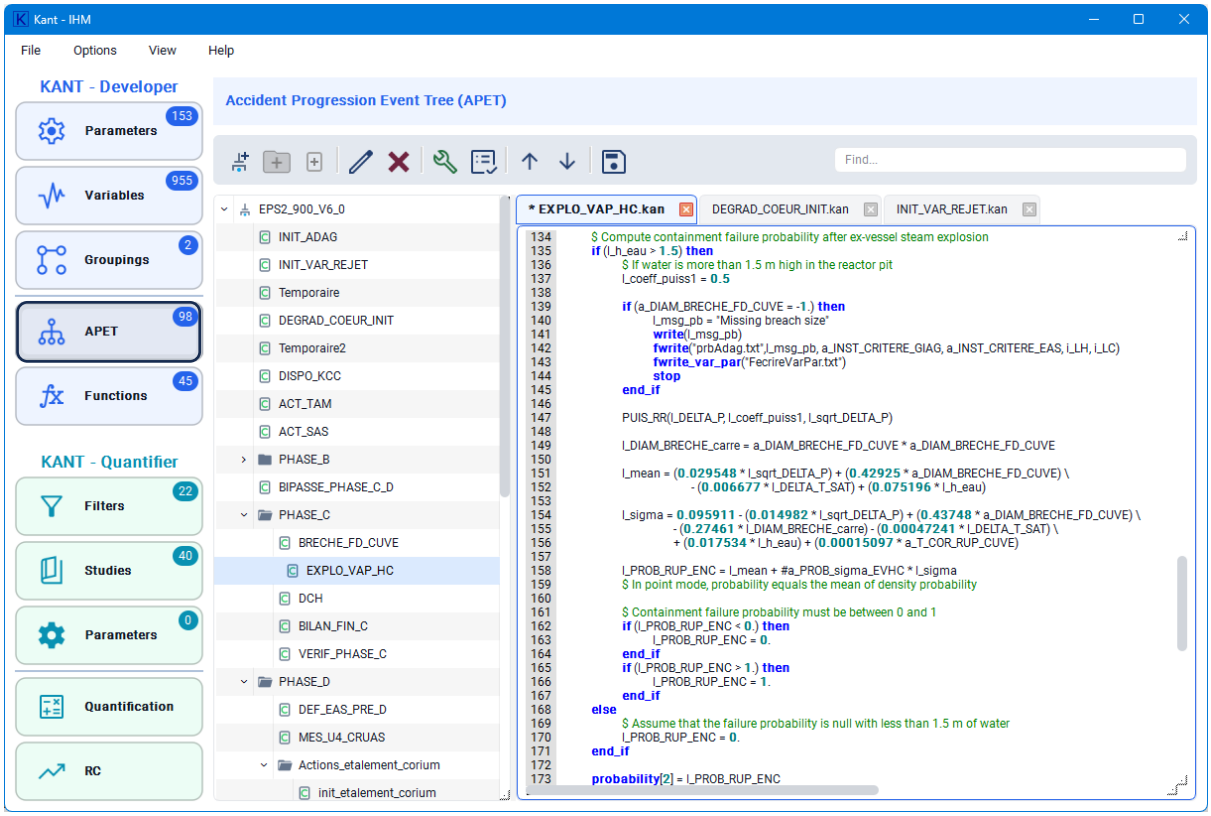
The GUI is written using the QT framework [6]. It is organized in pages, each one of which relates to a step in the development and the quantification of an APET.

4.1 The GUI for APET Development : “KANT – Developper”

The first set of pages, associated with the blue tiles (see Figure 3), deals with the development of an APET. They allow users to manage parameters, variables, L2 PSA sequences grouping (rules for the formation of RCs), APET elements (subtrees, questions, compilation) and internal functions.

Several tiles display a number inside a blue label. It counts the number of objects associated with the tile (number of parameters, variables, grouping rules, questions in the APET, internal functions). Similarly, the number of questions or functions that fail to compile is shown inside a red label.

Figure 3: APET visualisation



4.1.1 Pages “Variables”, “Parameters” and “Groupings”

Global variables are APET-wide and are managed from the “Variables” page. They usually represent the plant state:

- some variables are marked as PDS variables (meaning they belong to the L1-L2 PSA interface);
- some are marked as Release variables (meaning they will be computed only after the RCs have been identified);
- any variable that is used for the accident progression modelling or for sequences grouping into RCs must also be declared here.

Local variables don’t need any dedicated page, since they are defined directly in the source files of the question or functions.

KANT also considers parameters, that behave as constant global variables that any question can read. The parameters page allows to specify parameter attributes:

- names and types are required so that questions can use a parameter;
- minimum and maximum values, as well as the unit (the unit is informative, bounds will be checked at execution if they are defined);
- the “Release” attribute signals the parameters that are needed for release computations.

The “Groupings” page displays and edits the grouping rules for the formation of RCs. Those rules specify:

- which variables will be considered for the constitution of the RC;

- the allowed (integer) values for each one of these variables (with an information field for the meaning of each value).

4.1.2 Page "APET"

KANT manages most of the development operations of the APET from this page, which has two main purposes:

- creation, edition, deletion of APET elements: for example, users can create/modify contents of a question, move a question in the tree etc.;
- compilation: users can compile the whole APET, or a selected subtree, or a single question; compiler messages and errors appear in a dedicated window.

4.1.3 Page "Functions"

This page manages internal functions written in the KANT language. As in other languages, users may create functions to avoid repetitions, and to keep the code of the questions simple.

4.2 The GUI for APET Quantification : "KANT – Quantifier"

The second set of pages, associated with the green tiles, deals with the quantification of a given APET.

The APET quantification follows the steps below:

- selection of the APET (page "Studies") and the PDS set: the pages "Filters" defines boolean expressions to restrict the PDSs to consider, if needed (for instance to consider only a subset of L1 initiator events);
- assignment of parameter values;
- quantification execution.

4.2.1 Page "Parameters"

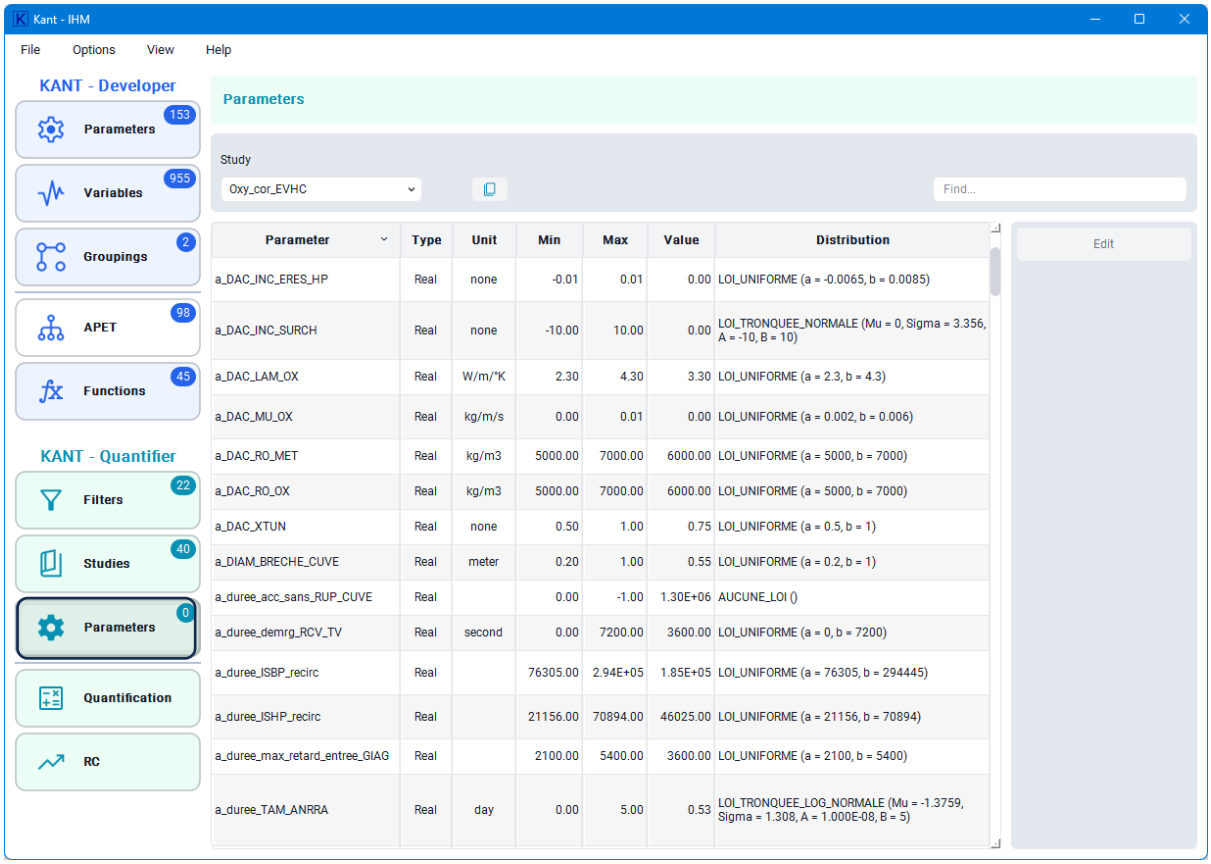
The parameters page assigns values (or numeric distributions) to each parameter as shown in Figure 4.

Parameter values are assigned on a per-study basis. This concerns:

- all data used for the quantification of the study;
- parameters reflecting epistemic or stochastics uncertainties. KANT supports several classic probabilistic laws (uniform, log-uniform, normal, log-normal and their truncated variants);
- values of parameters that can be changed to perform sensibility analysis.

The tile associated with the quantifier displays a number inside a green label. It is the number of objects associated with the page (filters, studies) or information about some work to be done (for the parameter pages, it is the number of missing values).

Figure 4: Assignation of Values and Probabilistic Laws to Parameters



Parameter	Type	Unit	Min	Max	Value	Distribution
a_DAC_INC_ERES_HP	Real	none	-0.01	0.01	0.00	LOI_UNIFORME (a = -0.0065, b = 0.0085)
a_DAC_INC_SURCH	Real	none	-10.00	10.00	0.00	LOI_TRONQUEE_NORMALE (Mu = 0, Sigma = 3.356, A = -10, B = 10)
a_DAC_LAM_OX	Real	W/m/°K	2.30	4.30	3.30	LOI_UNIFORME (a = 2.3, b = 4.3)
a_DAC_MU_OX	Real	kg/m/s	0.00	0.01	0.00	LOI_UNIFORME (a = 0.002, b = 0.006)
a_DAC_RO_MET	Real	kg/m3	5000.00	7000.00	6000.00	LOI_UNIFORME (a = 5000, b = 7000)
a_DAC_RO_OX	Real	kg/m3	5000.00	7000.00	6000.00	LOI_UNIFORME (a = 5000, b = 7000)
a_DAC_XTUN	Real	none	0.50	1.00	0.75	LOI_UNIFORME (a = 0.5, b = 1)
a_DIAM_BRECHE_CUVE	Real	meter	0.20	1.00	0.55	LOI_UNIFORME (a = 0.2, b = 1)
a_duree_acc_sans_RUP_CUVE	Real		0.00	-1.00	1.30E+06	AUCUNE_LOI ()
a_duree_demrg_RCV_TV	Real	second	0.00	7200.00	3600.00	LOI_UNIFORME (a = 0, b = 7200)
a_duree_ISBP_recirc	Real		76305.00	2.94E+05	1.85E+05	LOI_UNIFORME (a = 76305, b = 294445)
a_duree_ISHP_recirc	Real		21156.00	70894.00	46025.00	LOI_UNIFORME (a = 21156, b = 70894)
a_duree_max_retard_entree_GIAG	Real		2100.00	5400.00	3600.00	LOI_UNIFORME (a = 2100, b = 5400)
a_duree_TAM_ANRRA	Real	day	0.00	5.00	0.53	LOI_TRONQUEE_LOG_NORMALE (Mu = -1.3759, Sigma = 1.308, A = 1.000E-08, B = 5)

4.2.2 Page “Quantification”

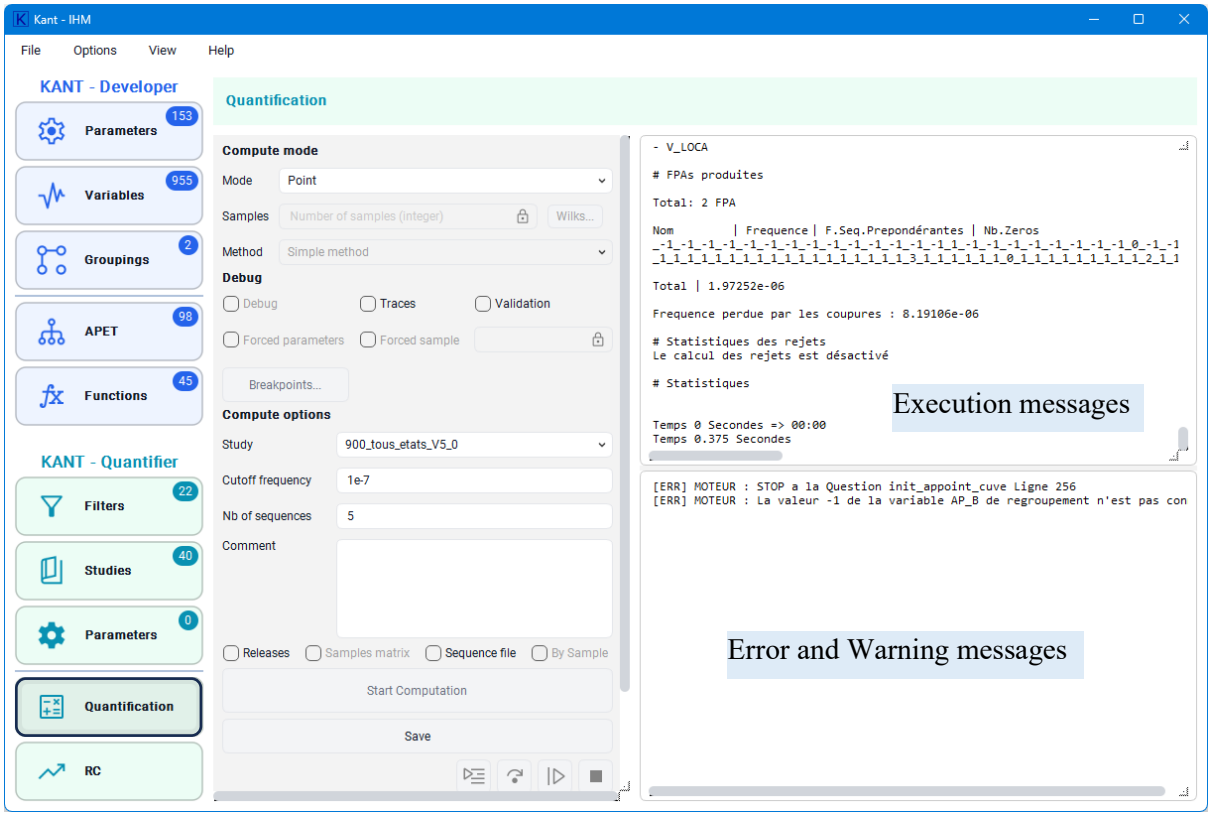
The page shown in Figure 5 offers several means to control the execution of the quantification:

- selection of the calculation mode:
  - “point mode” calculation: quantification is performed with default values, ignoring uncertainties (ex: 50% quantile value of the uncertain parameters);
  - “uncertainty mode” calculation: uncertainty analysis is performed by sampling the parameters values according to their associated probabilistic laws. KANT currently has two sampling options: Monte Carlo sampling or Latin Hypercube Sampling (LHS);
- selection of the cut-off frequency (to reduce the number of sequences to compute);
- calculation of the RC frequencies;
- calculation (or not) of the release associated with each RC.

It also offers debugging features:

- for the Monte-Carlo mode: turn off pseudo-random generator initialization, store or reuse the parameter sample values, replay a given set of samples (useful for non-regression analysis);
- for all modes: show variables, choose an answer for the current question, replay a given sequence or a recorded one.

Figure 5: Quantification execution



5. CONCLUSION

KANT is a user-friendly software used to perform ASNR's L2 PSA. It allows to consider (among other things) detailed L1/L2 PSA interfaces, detailed APETs with large number of events, user functions for implementation of physical models, uncertainties modeling and term source calculations in the APET.

The latest evolutions of KANT have been a major step in its adaptation to the evolution of hardware, software and user expectations, while remaining able to run studies made with earlier versions.

Some evolutions of the KANT language are planned, such as better type and argument checking for external calls to functions, multi-dimensional matrix or the possibility to use global variables in internal functions.

Acknowledgements

ASNR would like to thank Thierry Diffo and Jérôme Bourgeois (CODRA company) who implemented the new GUI, and Kevin Juilly and Sébastien Monot (AS+ company) who ported the compilation target to Lua.

References

[1] N. Duflot, N. Rahni, T. Durin, Y. Guigueno and E. Raimond. "*A new interfacing approach between level 1 and level 2 PSA*", PSAM 12, 2014
[2] T. Durin, J. Denis, E. Raimond and Y. Guigueno. "*Very fast running codes for the characterization of severe accident radiological consequences and the results presentation in level 2 PSA*", PSAM11, 2012
[3] Aho, Sethi, Ullman, *"Compilers: Principles, Techniques, and Tools"*, Pearson, 1985
[4] Roberto Ierusalimschy, "*Programming in Lua, fourth edition*", Lua.org, 2016

[5] QT Framework documentation https://qt.org/
[6] Lua reference manual https://www.lua.org/
[7] Message Passing Interface tutorial https://hpc-tutorials.llnl.gov/mpi/
[8] MPI specifications https://www.mpi-forum.org/index.html/