Assembling Multiple Models Within the RAVEN Framework

A. Alfonsi*, C. Rabiti*, D. Mandelli*

* *Idaho National Laboratory*
andrea.alfonsi@inl.gov, cristian.rabiti@inl.gov, diego.mandelli@inl.gov

INTRODUCTION

RAVEN (Risk Analysis Virtual ENvironment) [1,2,3] is an INL-developed software tool that can be used to identify and increase the safety margin in complex systems.

As a generic software framework, RAVEN is designed to perform parametric and probabilistic analysis based on the response of complex system codes. RAVEN is capable of investigating the system response as well as the input space using Monte Carlo, Grid, or Latin Hyper Cube sampling schemes, but its strength is focused toward system feature discovery, such as limit surfaces, separating regions of the input space leading to system failure, using dynamic supervised learning techniques.

RAVEN is currently able to construct multi-targets Reduced Order Models [4], which are aimed to represent the response of a system (in a fixed configuration) for multiple Figures of Merits (FOMs) and time-dependent ROMs [5,6]. These capabilities represent the initial steps for a larger implementation about the interaction of multiple models. In fact, in several cases, multiple models need to interface with each other since the initial conditions of one are dependent on the outcomes of another.

To better understand the problem that here is solved, it is useful to consider a simple example:

- The following problem is considered: a weather forecast simulation code "a" is used to compute the external (i.e. ambient) temperature in a certain location. A second model "b" is inquired to compute the average temperature in a room having as boundary condition, among several others, the external ambient temperature. The response of the model "b" depends on the outcome of the model "a".

The reported example is only aimed to illustrate the reason why the creation of a framework to make interact different models is a key development for the advancement of RAVEN as a comprehensive calculation flow driver. Before reporting how the ensemble-models have been implemented, it is necessary to briefly describe the representative Model "entities" that are available in RAVEN.

MODELS IN RAVEN

The Model entity, in the RAVEN environment, represents a "connection pipeline" between the input and the output space. The RAVEN software itself does not own any physical model (i.e., it does not possess the equations needed to simulate a system), but implements APIs by which any generic model can be integrated and interrogated. In the RAVEN framework four different model categories (entities) are defined:

- Codes
- Externals
- ROMs
- Post-Processors

The Code model represents the interface object that establishes the communication pipe between RAVEN and any driven code. Currently, RAVEN has APIs for several different codes:

- RELAP5-3D and RELAP-7, a safety analysis codes (thermal-hydraulic) developed at INL;
- Any MOOSE-based application;
- Modelica, object-oriented, declarative, multi-domain modeling language for component-oriented modeling of complex systems;
- MELCOR, engineering-level computer code that models the progression of severe accidents in light-water reactor nuclear power plants (coupling under development by the University of Rome "La Sapienza");
- MAAP5, computer code that models the progression of severe accidents in light-water reactor nuclear power plants (coupling performed by the Ohio State University);
- And several others.

The data exchange between RAVEN and the driven code can be performed either by direct software interface or by files such as input files. If the system code is parallelized, the data exchanging by files is generally the way to follow since it can be much more optimized in large clusters.

The External model allows the user to create, in a Python file (imported, at run-time, in the RAVEN framework), its own model (e.g. set of equations representing a physical model, connection to another code, control logic, etc.). This model will be interpreted/used by the framework and, at run-time, will become part of RAVEN itself.

The ROM (Reduced Order Model) represents an API to several different algorithms. A ROM is a mathematical representation of a system, used to predict a selected output space of a physical system. The creation and sub-sequential usage of a ROM involves a procedure named "training". The "training" is a process that uses sampling of the

physical model to improve the prediction capability (capability to predict the status of the system given a realization of the input space) of the ROM. More specifically, in RAVEN the ROM is trained to emulate a high fidelity numerical representation (system codes) of the physical system.

The Post-Processor model is aimed to manipulate the data generated, for example, employing a sampling strategy. In RAVEN several different post-processors are available: 1) Statistics Post-Processor, aimed to compute all the statistical figure of merits (e.g. expected values, variance, skewness, covariance matrix, sensitivity coefficients, etc.); 2) Limit Surface, which computes the Limit Surface, inquiring a goal function (i.e. a function that determines if a certain coordinate in the input space led to a failure or success), and so many others.

ENSEMBLE MODEL STRUCTURE IN RAVEN

As already mentioned, in several cases multiple models need to interface with each other since the initial conditions of some are dependent on the outcomes of others. In order to face this problem in the RAVEN framework, a new model category (e.g. class), named *EnsambleModel*, was implemented [7]. This class is able to assemble multiple models of other categories (i.e. Code, External Model, ROM), identifying the input/output connections, and, consequentially the order of execution and which sub-models can be executed in parallel.

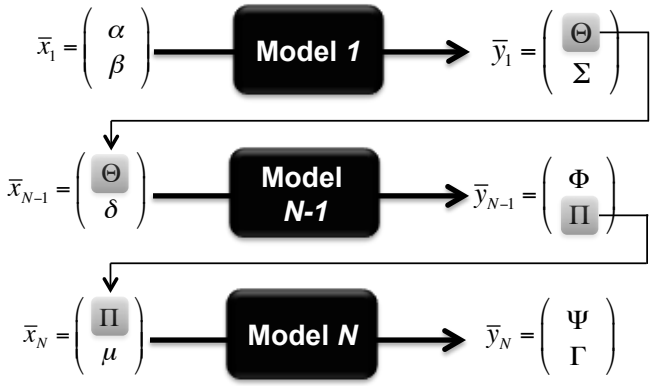

Figure 1 - Example of an *EnsembleModel* constituted by 3 sequential sub-models.

Figure 1 reports an example of an *EnsembleModel* that is constituted by 3 sub-models (ROMs, Codes, or External Models). As it can be noticed:

- The *Model 2* is connected with the *Model 1* through the variable Θ (Model 1 output and Model 2 input);
- The *Model 3* is connected with the *Model 2* through the variable Π (Model 2 output and Model 3 input);

In this case, the *EnsembleModel* is going to drive the execution of all the sub-models in a serial sequence, since each model (except the *Model 1*) is dependent on one of the outcomes of previously executed.

In several cases, the input of a model depends on the output of another model whose input is the output of the initial model. In this situation, the system of equation is non-linear and an iterative solution procedure needs to be employed. The *EnsembleModel* entity in RAVEN is able to detect the non-linearity of the sub-models' assembling and activate the non-linear solver: an iterative scheme. Figure 2 shows an example of when the *EnsembleModel* entity activates the iteration scheme, which ends when the residue norm (between an iteration and the other) falls below a certain input-defined tolerance.

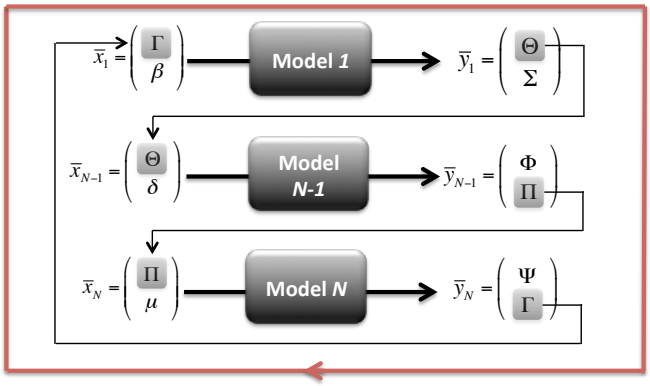

Figure 2 – *EnsembleModel* resolving in a non-linear system of equations – Numerical iterations.

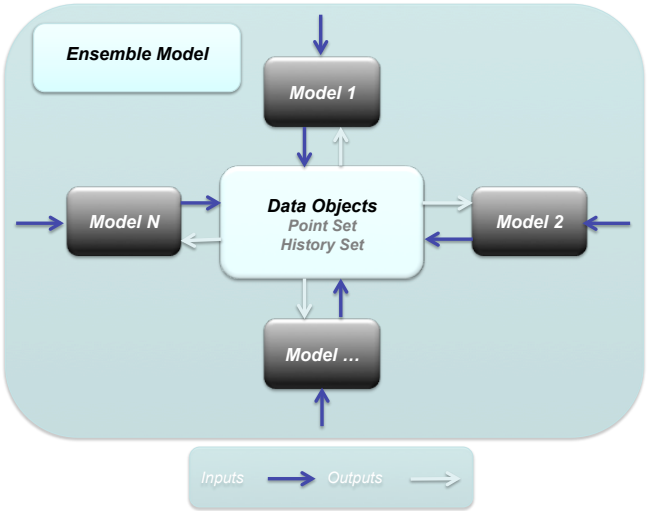

Figure 3 - *EnsembleModel* data exchange.

In RAVEN all the models' outputs (e.g. whatever code output, etc.) are collected in internal containers (named *DataObjects*) that are aimed to store time-series and input/output data relations in a standardized fashion; in this way, the communication of the output information among different entities (i.e. Models) can be completely agnostic with respect to the particular type of output generated by a model. The Ensemble-Model entity fully leverages this

peculiarity in order to transfer the data from a Model to the other(s).

Based on the Input/Output relations of each sub-models, the *EnsembleModel* entity constructs the order of their execution and, consequentially, the links among the different entities.

ENSEMBLE MODEL FOR 1-D FIGURE OF MERITS

The initial infrastructure of the *EnsembleModel* was able to transfer information among different models just in case of scalar quantities (e.g. peak clad temperature, constant thermal conductivities, etc.). This limitation was connected to the fact that in RAVEN the concept of "input realization" was limited to scalar values (i.e. the RAVEN code was able to perturb the input space in terms of scalar quantities). Since the increasing interest in using RAVEN also oriented to interconnection among heterogeneous Models' entities, the concept of treatable "input realizations" has been revised, including the possibility to process 1-D realizations.

Figure 3 schematically shows the communication piping established by the *EnsembleModel* entity. It can be noticed how the sub-models share information (inputs/outputs data) using the *DataObjects* entity as communication network.

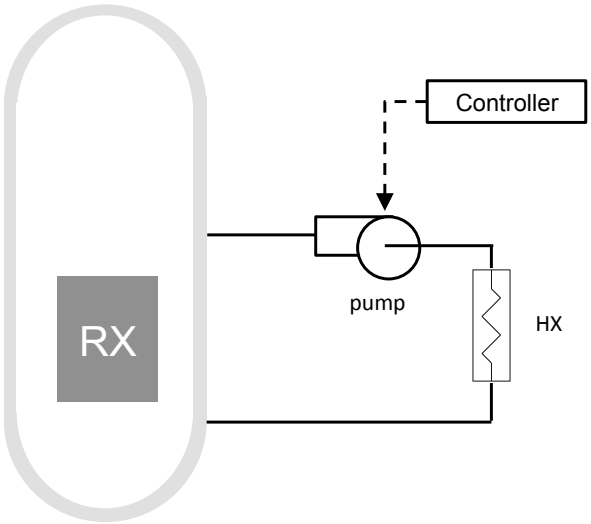

Figure 4 – Pump controller model scheme

APPLICATION EXAMPLE

In order to test the newly developed capability to process 1-D FOMs in an *EnsembleModel* configuration, two models have been considered:

- A pump controller model (***Model B***) for a hypothetical simplified PWR model (see Figure 4) has been used. It consists of the following components:
  - Reactor core (RX)
  - Motor operated pump
  - Pump digital controller
  - Heat exchanger (HX)

This system is responsible to remove the decay heat generated from the core (RX) in order to avoid damage of the core itself. While we assumed that both the HS and the pump are perfectly reliable components (i.e., no failure can be introduced) the digital pump controller reliability model has been performed using a continuous time Markov Chain formulation.

In more detail, the controller has been modeled using 4 states (Figure 5):

- Operating: controller operating as designed
- Failed closed: controller failed by sending close signal to pump (i.e., pump not running)
- Failed stuck: controller failed by sending oldest valid signal to pump
- Failed random: controller failed by sending close signal to pump

Since the scope of this exercise is to show the new capability in RAVEN, an additional

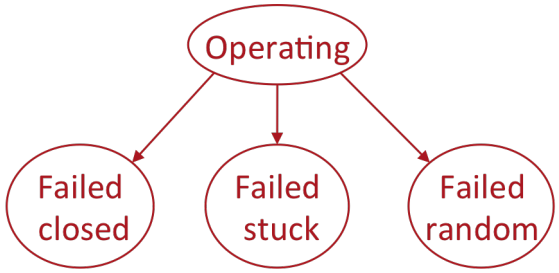

Figure 5 - Continuos time Markov model for the pump controller

In order to perform such analysis the model has been coded as a RAVEN external model (see Sect. 2.2) which determine the temporal profile of core temperature give the two stochastic parameters:

- Pump controller failure time (***CNTR***$_{f,time}$)
- Pump controller failure mode (***CNTR***$_{f,mode}$)

The dynamic of the hypothetical system has been modeled using basic mass and energy conservation laws so no effective engineering conclusions can be gathered by this example.

- A power history generator model (***Model A***) has been used. It employs of the following simple equation:

$$Power(t) = 1{,}500 * scaling_P * \exp(-t)$$

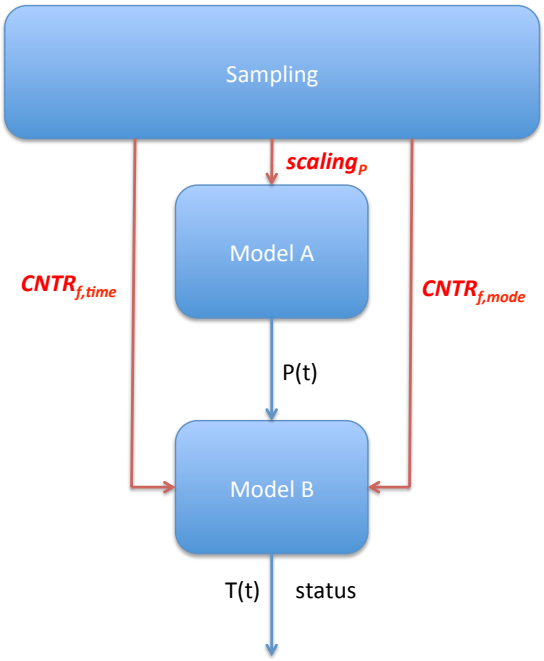

Figure 6 - Ensemble Model scheme for PWR controller example

The power multiplier (***scaling$_P$***) is an additional stochastic parameter in this example analysis (Uniform between 0.5 and 1.2).

The *EnsembleModel* data flow is shown in Figure 6. The Model A generates a Power history (time-dependent) that is passed into the Model B that employs the balance analysis. Even if the presented example is quite simplified, it shows the potential of this added capability.

By using RAVEN we sampled the three stochastic parameters using a Monte-Carlo algorithm and generated 1500 simulations as shown in Figures Figure 7 and Figure 8. Note in Figure 8 (i.e., Model B) that in some cases elevated temperatures are recorded due to the potential failures of the pump in the system.

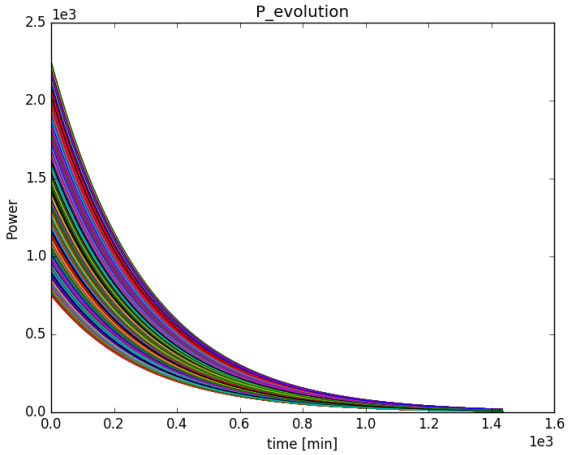

Figure 7 - Plot of the 1500 histories (Power - Model B) generated by RAVEN

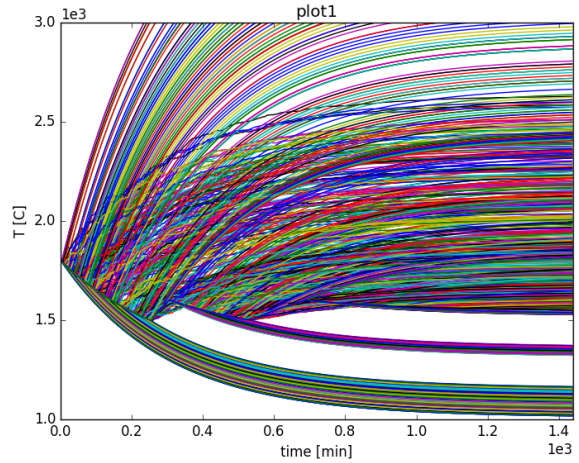

Figure 8 - Plot of the 1500 histories (Temperature – Model A) generated by RAVEN

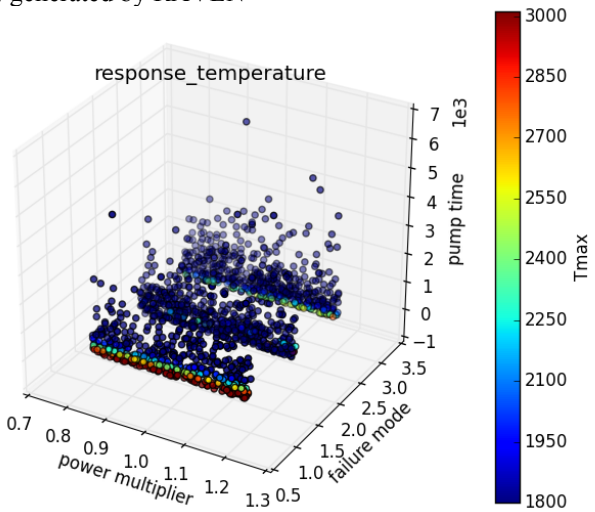

Figure 9 – Input space partitioning with respect to maximum temperature in the system

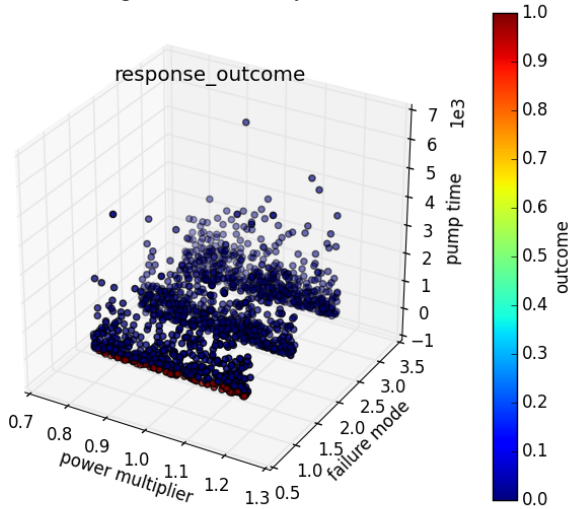

Figure 10 - Input space partitioning with respect to the outcome of scenario (failure/success)

In Figures Figure 9 and Figure 10 it can be seen as most of the failures happened in failure mode 1 (i.e. controller failed by sending close signal to pump (i.e., pump not running).
This simple example testifies how RAVEN can share 1-Dimensional Figure of Merits, abstracting the concept of "input realization" from scalars (e.g. uncertainty on thermal conductivity) to vectors (e.g. power histories).

FINAL REMARKS

The upgrade of the *EnsembleModel* capability in RAVEN in order to process 1-Dimensional Figure of Merits represents a development that determined the need to abstract the concept of "input realization" in the RAVEN code. RAVEN has extended the concept of "input realization" to any scalar and 1-Dimensional Figure of Merits (i.e. parameter and vectorial uncertainties). This abstraction makes the *EnsembleModel* infrastructure capable to handle the current and future challenges and needs within the RISMC "Industrial Applications" since they rely on multiple models (i.e. codes), communicate through a common interfaced infrastructure, and combine high-fidelity codes with surrogate modeling. Future work in RAVEN is to extend the *EnsembleModel* infrastructure in order to process High-Density fields (both as input and output spaces). This development is needed in order to make the framework able to interface surrogate models that are "trained" on mesh-data (e.g. 3-Dimensional power maps, etc.).

REFERENCES

1. A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and R. Kinoshita, "RAVEN as a tool for dynamic probabilistic risk assessment: Software overview," in Proceeding of M&C2013 International Topical Meeting on Mathematics and Computation, CD-ROM, American Nuclear Society, LaGrange Park, IL, 2013.
2. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, A. Naviglio, "Dynamic Event Tree Analysis Through RAVEN", International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA 2013), September 22-26, Columbia, SC, USA, (2013).
3. A. A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and R. Kinoshita, "RAVEN: Development of the adaptive dynamic event tree approach," Tech. Rep. INL/MIS-14-33246, Idaho National Laboratory (INL), (2014)
4. A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, S. Sen, and C. Smith, "Improving limit surface search algorithms in raven using acceleration schemes", INL/EXT-15-36100 (July 2015).
5. D. Mandelli, C. Smith, A. Alfonsi, C. Rabiti, J. Cogliati, H. Zhao, I. Rinaldi, D. Maljovec, P.Talbot, B. Wang, V. Pascucci "Reduced Order Model Implementation in the Risk-Informed Safety Margin Characterization Toolkit." INL/EXT-15-36649 (September 2015)
6. C. Rabiti, A. Alfonsi, D. Huang, F. Gleicher, B. Wang, H. S. Abdel-Khalik, V. Pascucci, and C. L. Smith, "System Reliability Analysis Capability and Surrogate Model Application in RAVEN", INL/EXT-16-37243 (November 2016).
7. A. Alfonsi, C. Rabiti, D. Maljevoic, D. Mandelli, J. Cogliati, "Enhancements to the RAVEN code in FY16", INL/EXT-16-40094 (September 2016).