Implementation of the System Thermal-Hydraulic code TRACE into SALOME Platform for Multi-Scale Coupling

***Kanglong Zhang, Victor Hugo Sanchez Espinoza, Robert Stieglitz**
Karlsruhe Institute of Technology (KIT)
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen, Germany
kanglong.zhang@kit.edu; victor.sanchez@kit.edu; robert.stieglitz@kit.edu

ABSTRACT

The paper discusses the procedures to implement the best-estimated system thermal hydraulic code TRACE into the open-source platform SALOME in order to enable TRACE for multi-scale coupling with other thermal-hydraulic solvers such as sub-channel or CFD codes. For this purpose, the source code of TRACE was re-organized and modularized as the primary step in order to meet the functional modularization request of SALOME-YACS module, which is the GUI-based supervision tool of the SALOME-platform. First of all, the TRACE-source code was wrapped with a newly-developed C++ envelope which supplies an interface between the basic Python layer of SALOME and the Fortran calculation engine of TRACE. With another newly developed SWIG-file, the communication channel between the C++ envelope and the SALOME Python layer was established. As a result, TRACE was fully implemented into SALOME as a component. An explicit mesh generation capability was also developed for TRACE in order to map the computational domains (meshes) between different codes and also for post-processing purpose. This capability is based on SALOME-MED module. After the implementation, the code was then tested with a 3D problem of the reactor pressure vessel. The verification work is undergoing and the current results are very encouraging, which show that the developmental work discussed here is consistent.

Key Words: TRACE, SALOME, MED, YACS

1. INTRODUCTION

Simulation tools in engineering science are becoming more sophisticated and detailed regarding the spatial discretization thanks to the advances in the involved physical fields and the increasing computer power: a notable trend also in nuclear engineering. Different thermal hydraulic simulation tools with rather different spatial nodalization were developed over the past decades in order to improve the description of the complex fluid dynamic and thermal hydraulic taking place within the primary/secondary circuit including the core of nuclear power plant. For example, the system thermal hydraulic codes are focused on simulation of the integral behavior of a nuclear power plant. Hence the models are mainly one dimensional or coarse mesh-3D for some part of a NPP e.g. the reactor pressure vessel. On the other hand,

the sub-cannel codes are developed to enhance the thermal hydraulic phenomena within the reactor core and for this purpose they rely on quasi-3d model (singe and two-phase flow). Finally, the use of the general purpose computational fluid dynamic (CFD) codes is rapidly increasing in the nuclear community for both design optimization and safety-related investigations. However, two-phase flow models are still not yet mature for the application in the nuclear industry. On the other hand, the simulation of a complete nuclear power plant with a detailed core model that permits the prediction of local safety parameters is not yet possible since the problem is too large to be loaded in the memory of current computers. Consequently, for the time being KIT is engaged in the development of multi-scale simulation tools to take profit of the best features of system, sub-channel and CFD codes in solving the key-safety relevant phenomena of different reactor concepts.
Considering the peculiarities of the open-source SALOME-platform regarding the meshing, coupling options and post-processing, the investigations of KIT are concentrated in the multi-scale coupling of the system code TRACE, the sub-channel code SUBCHANFLOW and the open-source CFD code TrioCFD. Especially, the SALOME-platform provides powerful in-build functions for standardized coupling approaches of different solvers using the modules such as YACS and MED. Lots of codes were already successfully implemented into the SALOME platform for coupling purpose, including TrioCFD [1-5]. The work of this paper integrated the best-estimate system thermal hydraulic code TRACE developed by U.S. NRC into the SALOME-platform for multi-scale coupling e.g. with CFD or sub-channel codes.

2. CODES AND TOOLS

2.1 TRACE

TRACE is the abbreviation of TRAC/RELAP Advanced Computational Engine which is formerly called TRAC-M. It is the latest in a series of advanced, best-estimate reactor systems codes developed by U.S. NRC for analyzing neutronic thermal-hydraulic behavior in light water reactors. TRACE takes a component-based approach to modeling a reactor system. Each physical piece of equipment in a flow loop can be represented as some type of component. VESSEL is the special 3D component which can model the Reactor Pressure Vessel and other components in which 3D phenomena take place. The basic governing equation set of TRACE includes six equations which can simulate a full two-fluid hydrodynamic like the gas-liquid flow. In addition, two more equations are applied to describe the non-condensable gas field and to track dissolved solute [6].

2.2 SALOME

The SALOME platform is an open source software framework for numerical pre- post processing and integration of numerical solvers in various scientific domains. It is supported by CEA, EDF and OPENCASCADE. SALOME has already been employed to perform a wide range of simulations, which are typically related to industrial equipment in power plants (nuclear power plants, wind turbines, dams...). SALOME is mainly composed of 8 modules (Figure 1), among which, KERNEL and GUI provide the core functionalities of SALOME, GEOM is for CAD usage, MESH is in charge of generating computational grids, PARAVIS is nothing but PARAVIEW which is post processing professional, MED contains lots of mesh interpolation tools and also supplies a universal data format standard for all other modules, YACS is used to organize and monitor calculation chains, the last stands for the user-developed modules. In sum total, KERNEL and GUI are the base, GEOM, MESH and PARAVIS are for pre and post processing, MED, YACS and user-module are closely related to coupling issues.

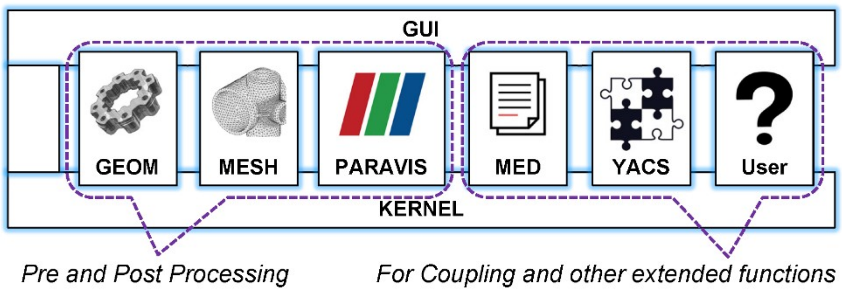

Figure 1. Diagram of SALOME architecture

3. DEVELOPMENT OF TRACE MESH BASED ON MED

For a successful coupling, physical field mapping between different code meshes is one of the critical issues, which largely determines the efficiency and even validity of the coupling. But first of all, the meshes should be properly and explicitly defined as the essential prerequisites, no matter in a file or in the memory. Almost all CFD codes have their own meshes isolated from the physical fields and the numerical definition of a problem. But for TRACE, the mesh is implicitly defined in the input file which mixes the numerical data and mesh description together. Thus the overriding work is developing an explicit TRACE mesh based on MED format. Specially, since the coupling only concerns the VESSEL component, only the MED mesh of VESSEL should be developed. SALOME-MED principally supports five cell types: tetrahedron, triangular prism, hexahedron, hexagonal prism and polyhedron. The former four (Figure 2.a) however, can't be applied to represent a TRACE cell. This is because the cells in TRACE are quite unusual, which are either fan-shaped or annular (Figure 2.b). So, the only remaining option is polyhedron.

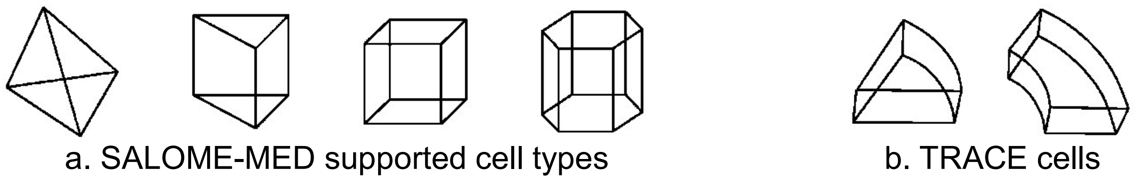

Figure 2. Cells of SALOME-MED and TRACE

Normally, only four points could be derived from TRACE input file for one single cell (Figure 3.a). But they are not sufficient to form an unbroken cell which contains space curves. Some assistant points should be inserted to complete the cell (Figure 3.b). Finally, a multi-faces wrapped TRACE cell could be built (Figure 3.c).

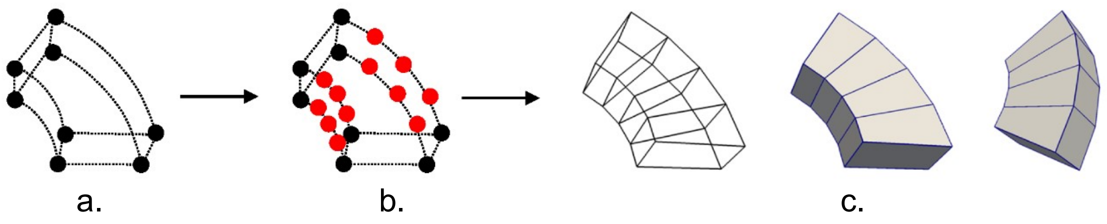

Figure 3. Building process of a TRACE typical cell

Up to this point, a TRACE mesh could be explicitly built from the input file with the help of SALOME-MED capabilities. No additional definitions are required. A VVER model was utilized to test the newly-developed mesh generation function of TRACE. Figure 4 presents the normal-cell-based TRACE mesh for cell-data (pressure, density .et al) post-processing. Figure 5 displays the tetra-cell-based TRACE mesh which was expressly developed for mesh-interpolation purpose, since the interpolation tools of SALOME-MED couldn't recognize normal TRACE cells (fan-shaped and annular cells). Moreover, an edge-based mesh (Figure 6) has to be developed for both post-processing and interpolation of edge-data like velocity and mass flow. Completed with the development of some other key functions, totally 21 kinds of data could be written to the three meshes for both post processing and data interpolation.

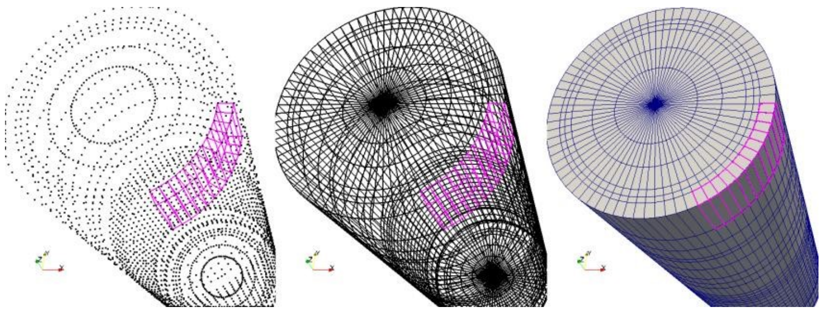

Figure 4. MED normal-cell-based-mesh of TRACE for VVER

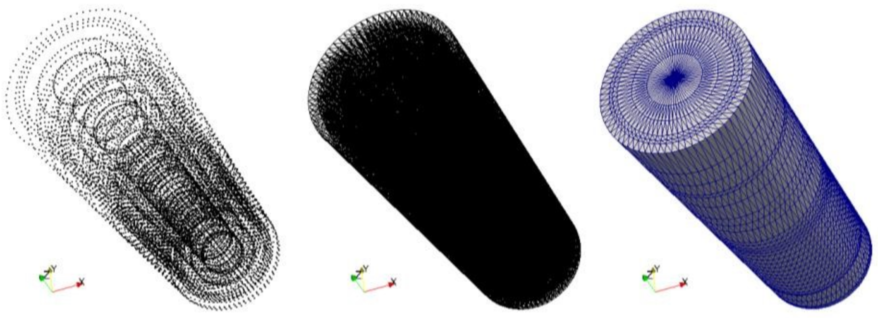

Figure 5. MED tetra-cell-based-mesh of TRACE for VVER

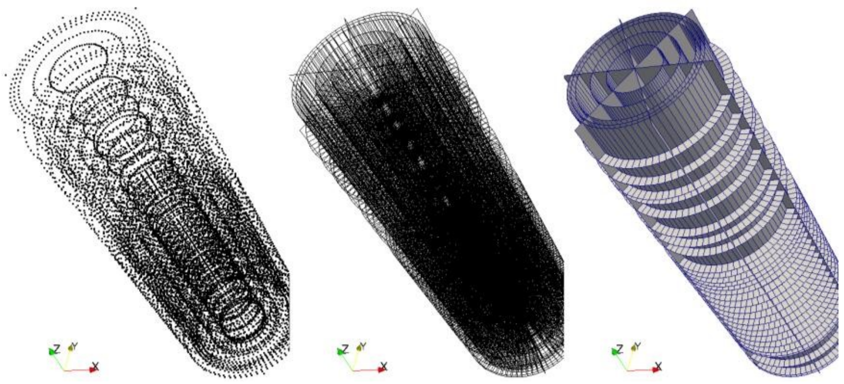

Figure 6. MED edge-based-mesh of TRACE for VVER

4. IMPLEMENTATION OF TRACE TO SALOME-YACS

SALOME-YACS provides user a friendly GUI to organize the calculation by dragging and connecting some basic functional-components, just as Figure 7 shows. Apparently, only explicit mesh is not enough to achieve a successful coupling. The single TRACE executable has to be further divided into several basic functional components such as “read input file”, “initialize”, “run a time step calculation” and so on. This operation is compulsory because TRACE has to supply various accessible input and output spots in order to be able to interact with other codes flexibly.

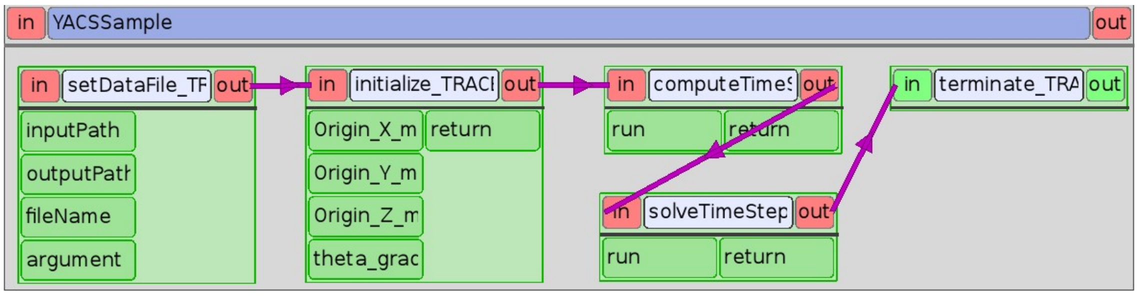

Figure 7. Typical calculation chain in SALOME-YACS

The implementation of TRACE to SALOME-YACS is a little bit complicated. Nevertheless, with the help of SALOME-tools (a toolkit aims to standardize developing work within SALOME), the entire process could be generalized into four main steps (Figure 8).

1) Re-written and re-organize the original TRACE source code to meet the functional modularized request of SALOME-YACS.
2) Develop a C++ envelope to wrap the lower TRACE Fortran computing engine forming a so called TRACE-SALOME internal object.
3) Develop a SWIG file to stick the internal object to SALOME-YACS python layer forming a so called TRACE-SALOME local python object.
4) Develop a CORBA file to establish the communication channels for TRACE module forming the final so called TRACE-SALOME Component.

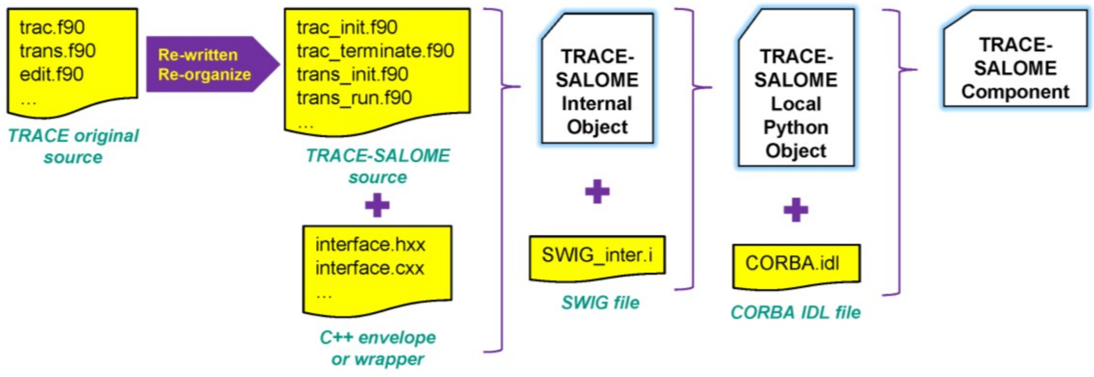

Figure 8. Four steps of implementing TRACE into SALOME

Step 4 could be automatically done by SALOME-tools, which means users don’t need to pay any attention on the CORBA layer despite its enormous complexity. The SWIG file of step 3 has to be developed by users but this is relatively simple since there are already some off-the-shelf templates available. The real challenges are step 1 and 2 which calls for in-depth knowledge of TRACE source code and involves substantial programming effort. The discussion is not only appropriate for TRACE but also applies to other codes-to-SALOME implementation work. The completed functional components of TRACE in SALOME-YACS are listed in Table 1. Details of the components are not further described due to the lack of space. Nevertheless, their key functions could be perceived at once from the names.

Table 1. Functional components of TRACE in SALOME

setDataFile	presenttime	getOutputFieldsNames
initialize	isStationary	getOutputMEDField
computeTimeStep	getInputFieldsNames	terminate
initTimeStep	getInputMEDFieldTemplate	
solveTimeStep	setInputMEDField	

5. CODE TESTING

A simple 3D coolant mixing case was built to test TRACE-SALOME. The model includes four primary loops, corresponds to four sectors of the VESSEL component (Figure 9). The transient process is described in Table 2.

Table 2. Transient process of coolant mixing

	Loop #1 Inlet			*Loop #2 #3 #4 Inlet*
Time (t)	0s ~ 20s	20s ~ 30s	30s ~ 50s	0s ~ 50s
Temperature	400K	400 + (t-20)*100K	500K	400K
Velocity	5m/s			5m/s
Solute mass ratio	0.01			0.01

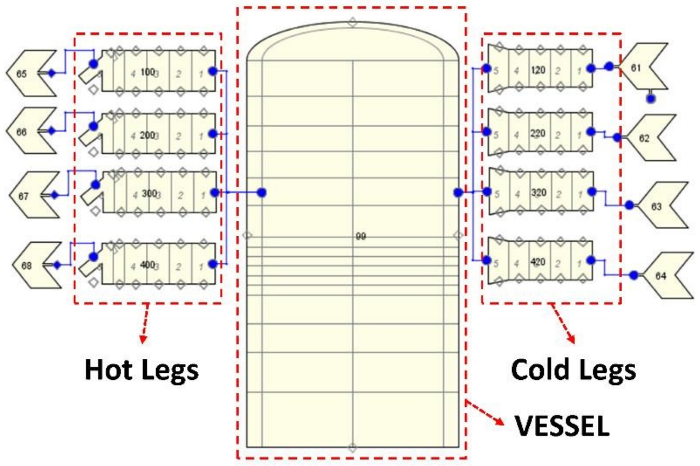

Figure 9. Coolant mixing test case for TRACE-SALOME

Curves in Figure 10 displays the coolant temperature changes over time at the four hot legs, covering both TRACE standalone and TRACE-SALOME. The various degrees of coolant temperature rise at hot leg 2, 3 and 4 come from the coolant mixing effect in the VESSEL. It could be found that the temperature curves of TRACE-SALOME perfectly cover the curves of TRACE standalone. This is obvious because nothing was done to the underlying calculation engine. What had been performed are just some modifications and supplements on the code's framework. Now, results of TRACE are more easily accessible thanks to the explicit mesh and field extraction capabilities. Figure 11 presents the coolant temperature together with the mesh in the VESSEL. All of the current out comings indicate that the implementation of TRACE to SALOME was correctly done.

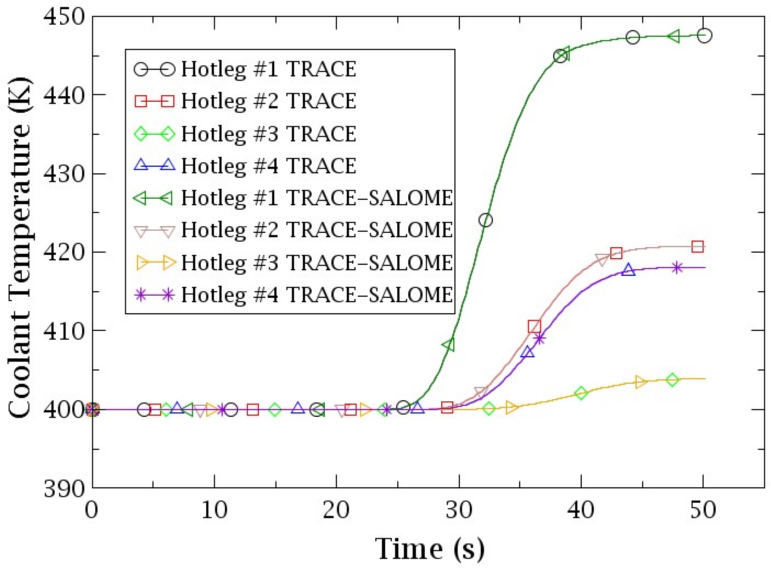

Figure 10. Coolant temperature curves at hot legs of TRACE and TRACE-SALOME

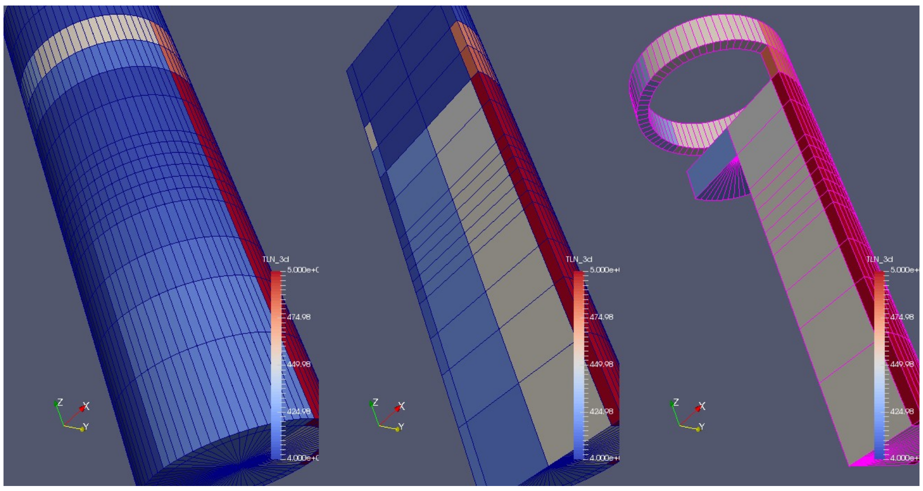

Figure 11. Post processing of coolant temperature in VESSEL of TRACE-SALOME

6. CONCLUSION

In this paper, the implementation of the system code TRACE in the SALOME-platform for multi-scale coupling is described. As a result, three different meshes were developed for the TRACE VESSEL-component based on SALOME-MED: the normal-cell-mesh for the post-processing of cell-based data, the tetra-cell-mesh for mesh interpolation of cell-based data (code coupling) and the edge-mesh for both post-processing and mesh interpolation of edge-based data. In addition, the TRACE code was modularized into 13 functional components, which is an important prerequisite for a flexibly coupling of TRACE with other thermal hydraulic odes. The new TRACE-functionalities inside the SALOME-platform e.g. the supervision YACS-module were used to defined four main computational steps in the GUI. Finally, a 3D coolant mixing problem in a VVER RPV was simulated and the new functionalities were successfully demonstrated. Summarizing it can be stated that the TRACE code is ready for coupling with another thermal hydraulic solver using the SALOME functionalities.

REFERENCES

1. B. Chanaron, C. Ahnert, N. Crouzet and V. Sanchez, “Advanced multi-physics simulation for reactor safety in the framework of the NURESAFE project”, Annals of Nuclear Energy, Volume 84, October 2015, Pages 166-177.
2. Y. Perin, “Development of a Multi-Physics, Multi-Scale simulation tool for LWR safety analysis”, Munich, Technical University Munich, 2016.
3. A. Hebert, “Integration of the DRAGON5/DONJON5 codes in the SALOME platform for performing multi-physics calculations in nuclear engineering”, Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013), La Cité des Sciences et de l’Industrie, Paris, France, October 27–31, 2013.
4. M. Calleja, V. Sanchez, J. Jimenez and U. Imke, “Coupling of COBAYA3/SUBCHANFLOW inside the NURESIM platform and validation using selected benchmarks”, Annals of Nuclear Energy, Volume 71, March 2014, Pages 145-158.
5. A. Cervone, D. Cerroni, R. Da Vià and F. Menghini, “Integration of the FELMORE Code in the Salome Platform”, September 2014, ENEA, CIRTEN.
6. U. S. Nuclear Regulatory Commission, “TRACE TRACE V5.1051 theory manual”, Washington, DC.