VERIFICATION OF FUEL PERFORMANCE SIMULATION CODES: APPLICATION OF DISTRIBUTED COMPUTING

Y. Liu and C.J. Westbye[1]
Code Support & Model Development Section
Ontario Power Generation
700 University Avenue
Toronto, Ontario, Canada M5G 1X6

INTRODUCTION

The Canadian Nuclear Industry as a whole is attempting to find accurate, efficient and cost-effective methods to either test or demonstrate that complex simulation codes are behaving as they were intended to. One method that has been used at Ontario Power Generation is "stress testing": running the code or code suite a sufficient number of times with small variations in input data, and tracking the resulting change in calculated results. This technique has only recently become practical for large simulation codes with the increasing power of desktop computers and the ability to dispatch batch jobs across a network; the method used for the work performed to date is described in detail in [1].

The stress testing methodology is a balance between safety analysis needs and software engineering principles. It is not in general possible to completely cover the range of all input parameters and their interactions in a reasonable amount of time. Hence the number of input parameters to test was restricted to those that have the largest impact on the key output results (those outputs that are actually used directly or indirectly as safety criteria). Of particular value in identifying code deficiencies was "parameter scanning", where one input parameter was varied using a fine increment over a range encompassing the normal variation. The key output results were then plotted against the varying input parameter, and the results studied to ensure that the predicted trends were correct and no non-physical discontinuities were present.

This paper reports the results from two such stress testing exercises as examples of applications of the stress testing method. One application is on the ELESIM/ELOCA code suite [2] to show how stress testing can identify problems in code testing and verification. The other application is from the FACTAR code suite [3,4] focusing on how the stress testing method can be applied to a large code suite.

---

[1] Present address: Geodesic Consulting Inc., 1807-21 Vaughan Road, Toronto, ON M6G 2N2

RATIONALE

The Canadian Nuclear Industry is currently developing standards for the engineering and reverse-engineering of safety analysis simulation codes, in order to achieve a high level of confidence that they function correctly and accurately. Examples of this effort include the Nuclear Asset Optimization Project: Code Verification & Validation at Ontario Power Generation, and the Code Validation Project at Atomic Energy of Canada Limited. Both projects are similar in goal and approach. Central to both is the development of a set of documentation which rigorously specifies the characteristics of a given piece of software, to serve as a basis for verification and future development. Verification of the software, which is defined for the purposes of this paper as achieving reasonable confidence that a piece of software functions as intended and is free of errors under all conditions, is another central activity. Software validation, which is used in the engineering sense of quantifying the accuracy of a simulation code, completes the list of vital activities under these initiatives.

There are many techniques which can be employed in the verification of a simulation code. Comparison of the source code with a requirement specification, for example, can be a very thorough but time consuming (*i.e.*, expensive) method for determining that the software embodies all the requirements. This method is also somewhat unreproducible, in that it depends exclusively on human intervention. Static analysis of the source code using an automated tool is completely reproducible, cheap and efficient, but can only find gross errors or point to potential trouble spots through the use of quality metrics. This method, while useful for many purposes, does not address modelling correctness issues in any way. Dynamic software testing, where the code is executed over a range of conditions, offers a means to fill some of the gaps left by the above methods.

Testing a piece of software is hardly a novel concept; indeed, every simulation code will have been tested many times during its development and use. Many codes will have a standard test suite defined, which exercises the software over an appropriate range of conditions and model options. However, when the software consists of a large number of complex, coupled models representing the behaviour of a system under extreme conditions it is not always clear whether the software is behaving correctly. Engineers and scientists familiar with the system can analyze the simulation results and provide expert judgement regarding the fidelity of the results, at least in terms of the general trends and values reported. Comparison of the results to experimental data (validation) is an even better way to ensure the correctness of the simulation results, and is essential but suffers from many well known pitfalls, such as scarcity of experimental data, uncertainty in the measured results and boundary conditions, instrumentation which interferes with the effect being measured and the necessary use of non-prototypic conditions. Even if the software passes expert review and experimental validation, errors may still exist which are either not encountered for the conditions tested or whose effects are relatively small and are masked by the overall system response. Under a different set of conditions, the effect of such errors may be magnified and result in a non-conservative prediction of some key output parameter.

The "stress testing" technique offers a complementary testing strategy which is very inexpensive and reproducible, and can find classes of errors which more traditional

methods could easily overlook. A set of input conditions defining a reference case is defined (in a comprehensive test, many such reference cases would be used). Then, selected input parameters are varied over a selected range. This range could be determined by physical considerations, for example representing the uncertainty in a given parameter, or could be a very broad range defined simply to test the code. Selecting input parameters which are known to have a significant effect on the simulation results offers the quickest return on investment and the best chance of finding important errors, but any parameter available through the code input could in practice be selected. Key output parameters are then identified, based on the scenario under consideration. For example, peak sheath temperature is a key output parameter for large break loss of coolant accidents because it is directly used to assess fuel channel integrity.

Once a suitable set of key input and output parameters is identified, the code or code suite is executed a large number of times, with each case differing only in the value of certain input parameters. After each execution, the key output variables are extracted. Retaining only the key results is done for practicality, otherwise physical limits on computer disk space would eventually be reached.

Selecting the input parameter space is a particular difficulty. A complete combinatorial test matrix would not typically be possible; for example, if five input parameters are selected for variation over a range which includes only 20 points each, the total number of simulations would be 20^5 or 3.2×10^6. Such a large number of simulations would be difficult to achieve for large, long-running codes and the input space is probably not sufficiently dense to achieve good statistical results. An alternate approach which has succeeded in demonstrating software faults is to perform single parameter scans, where all but one input parameter is held constant at the reference value. Dual parameter variations, where a second parameter is changed, are also quite achievable.

The results from this exercise are then analyzed by plotting the key output parameters against the varying input parameter. Any discontinuities in the output are an immediate indication that (i) a software fault exists, or (ii) a physical regime has been crossed, an example of a so-called "cliff edge" effect. Discontinuities of type (ii) can generally be readily identified by an expert in the physical process or system, and so anomalous results arising from software errors are easily determined. As a further check on the fidelity of the software model, the trends exhibited can be analyzed for correctness and consistency with the expectation of a subject matter expert.

This method can be used to find certain classes of errors, for example discontinuities between models of different regimes, non-convergencies, or non-protected code in areas which are not normally exercised. Since an integrated test is performed, data handling between modules which function properly when tested independently is also indirectly assessed. The power of the method lies in its simplicity, ease of use and interpretation, process independent from human intervention and the very quick (and therefore inexpensive) results that can be obtained. The quick response is largely due to advances in desktop computing power and the technology to utilize multiple workstations, which is briefly described in the following section.

COMPUTATIONAL TECHNIQUE

Running a nuclear safety simulation code a sufficient number of times to obtain good coverage of the input domain has only been practical in the past for small codes, *i.e.*, those with quick turn-around times. As computing power has continued to increase, this restriction has become less limiting but is still an issue for codes or code suites which require at least several hours to run.

At Ontario Power Generation, most safety analysis computing is performed on an IBM SP2 cluster consisting of eight RS/6000 nodes. As an example of the difficulty in performing an adequate stress test, consider the FACTAR code suite (described in more detail below). For the first phase of the stress test, 182 cases based on a large break loss of coolant accident scenario were defined. Using the SP2 platform under average load conditions (at the time the analysis was performed), conservative estimates indicate that the absolute minimum turn-around time for this test suite would be 23 days, and that estimate is likely too low by a factor of two or three. To avoid this problem, a novel approach was adopted: a total of 30 desktop computers (all Intel Pentiums, ranging from 75 MHz to 166 MHz) were used to run all 182 simulations overnight. This was made possible by developing batch job scheduling software that sends jobs to computers connected to a Local Area Network. This software, named *SheepDog*, is discussed in detail in [1].

Use of this testing technique does not rely on *SheepDog*, of course. The results presented below for ELESIM/ELOCA code suite were generated using the standard SP2 computing platform, since the turn-around time for this code is very small. For a more time consuming code suite, exemplified in this paper by FACTAR, use of the standard computing facilities would have been prohibitively slow and would have negatively impacted upon other computer users. *SheepDog* was therefore invaluable to make stress testing of a large code suite practical.

SAMPLE RESULTS

Two sample applications of the methodology described above are presented in the following sections. The first considers a relatively small code where the input parameters could be sampled at quite a high resolution. The second considers a larger code suite which would normally prohibit a very large test matrix; these results are included to demonstrate that the method is generally applicable to virtually any simulation code.

ELESIM/ELOCA CODE SUITE

The sample application of the stress testing method described in this section is to demonstrate how this method can reveal potential problems in the code or code suite.

The code suite selected consists an ELESIM code version and an ELOCA code version for a single fuel element performance simulation. ELESIM performs the simulation under normal operating condition and provides the initial fuel element condition for ELOCA to perform an accident transient simulation. The key output parameter selected to demonstrate the stress testing result is the maximum fuel centreline temperature from

ELOCA transient simulations. The input parameters selected, UO_2 grain size and density, are two manufacturing data required in ELESIM simulation.

The stress testing result displayed in Figure 1 (with square marks) is the maximum centreline temperature from 150 ELESIM/ELOCA code suite simulations with small variations in UO_2 grain size while the rest of the input parameters are kept as constants. The result indicates that a discontinuity occurs between grain size values of 22.425 μm and 22.427 μm. Further assessment indicates that the discontinuity appears when the radial gap between fuel and sheath approaches to zero and an interfacial pressure is suddenly introduced. This discontinuity results from the way ELESIM treats the radial gap in the fuel-to-sheath heat transfer calculation. Some modifications are introduced to ELESIM based on the stress testing result. Similar simulation results from a modified ELESIM/ELOCA code suite are shown in Figure 1 as the circle-marked line which indicates that the modification resolved the discontinuity and the code suite predicts a consistently lower centreline temperature.

In Figure 2, fuel density is used as another input parameter to demonstrate the stress testing results. In Figure 2, the square-marked line shows the stress testing result for the density variation. The stress testing shows an anomalous point at density of 10.72254 Mg/m^3. It was caused by an anomalous opening of a very small radial gap (0.888×10^{-15} μm) between fuel and sheath. It was concluded that the simulation result is not converged at this density value. In addition, the square-marked line in Figure 2 displays that the radial gap between fuel and sheath experiences three different phases from low density to high density. There is a residual gap at low density. At high density, the radial gap is completely closed. Around 10.65 MG/m^3 the gap is very small and the slope indicates that the simulation result is more sensitive to fuel density than in the other two density regions. In Figure 2, the circle-marked line shows that the modifications that improve the stress testing result for UO_2 grain size variation does not provide acceptable result for density variation. Although the modifications fix the anomalous point at 10.72254 Mg/m^3, it brings instability when the radial gap approaches zero. The stress testing result indicates that the modifications made to the code suite are either not acceptable or not good enough, and more improvement is required.

The sample application of the stress testing method shown in Figures 1 and 2 indicates that the stress testing is a powerful tool for code development, code testing and verification. The method can easily identify modelling, coding related problems in the given ranges of the input parameters. It is a tool to determine the full impact of code modification.

FACTAR CODE SUITE

FACTAR (Fuel And Channel Temperature And Response) is used at Ontario Power Generation to calculate the transient fuel and fuel channel thermal/mechanical response to conditions which range from normal operation to severely degraded cooling. The code suite has two components. FACTAR_SS (Steady-State) [3], based on ELESIM-II (MOD10), determines the initial fuel conditions prior to the accident scenario. FACTAR 2.0 [4], based on ELOCA.Mk4, determines the transient response of the fuel channel components.

To perform the stress test, a reference large break loss of coolant accident scenario was selected. This case represents a typical 50% pump discharge break downstream of the pressurizer.

A total of eight FACTAR_SS/FACTAR input parameters were selected for variation. These parameters represent those that were previously known to have an impact on the key output parameters for the scenario under investigation:

i) fuel thermal conductivity;
ii) thermal expansion coefficient;
iii) grain size;
iv) fuel density;
v) fuel/sheath diametral clearance;
vi) fission gas volume;
vii) fuel-to-sheath heat transfer coefficient; and
viii) fuel heat capacity.

All parameters except the last are inputs to FACTAR_SS. Only parameters (i), (ii), (vii) and (viii) are direct inputs to FACTAR 2.0, although the other parameters indirectly affect the initial conditions that are passed to FACTAR 2.0.

FACTAR_SS and/or FACTAR 2.0 only permit certain types of variations for each of the input parameters. Most commonly, a new value can be specified, overriding the code default. For some models, the code allows the user to input a multiplication factor which is applied to the final result from a particular calculation. In another case, an increment can be specified which is added to the result of a calculation. The default values of each parameter, which define the reference case, and the types of permissible variations are provided in the following list:

Input Parameter	**Symbol**	**Type of Variation**	**Default Value**
Fuel thermal conductivity	k	increment	0.0 W/m·K
Thermal expansion coefficient	α	multiplier	1.0
Grain size	Gsize	value	7.5 μm
Fuel density	ρ_{fuel}	value	10.75 g/cm^3
Fuel/sheath diametral clearance	Diacl	value	0.08 mm
Fission gas volume	GasVol	value	1870 mm^3
Fuel-to-sheath heat transfer coefficient	$h_{fs,mult}$	multiplier	1.0
Fuel heat capacity	$\rho c_{p,mult}$	multiplier	1.0

A total of three key output parameters were selected for consideration. These output parameters (maximum fuel centreline temperature, maximum fuel sheath temperature and maximum sheath strain) are the parameters predicted by FACTAR that are most significant from a safety and licensing point of view.

In the first phase of this study, 182 cases were defined which scanned through each of the input parameters while holding the others constant. Dual parameter variations of particularly important quantities (*e.g.*, thermal conductivity and fuel density) were also considered. This test matrix is too sparse to actually achieve the desired confidence in the correctness and robustness of the simulation code, and was defined largely to demonstrate that such a testing strategy was possible and useful for a larger code suite. Complete results from the study are reported in [5], but selected results are discussed below.

Figures 3 and 4 show sample results from the stress testing exercise. In Figure 3, the variation in the peak sheath temperature is shown as a function of fuel thermal conductivity increment for several values of fuel density. It is observed that maximum sheath temperature generally decreases with increasing thermal conductivity, as expected due to the flattening of the fuel pellet radial temperature profile (not shown). The effect of fuel density is also as expected: lower density leads to lower thermal inertia and hence higher peak fuel temperatures, which determine the maximum sheath temperature in this case. The general trends are therefore consistent with expectations, and are free of obviously spurious results for this limited test matrix.

Figure 4 shows the variation in peak fuel centreline temperature as a function of fuel-to-sheath heat transfer coefficient multiplier and fuel heat capacity multiplier for two different UO_2 grain sizes. Higher values of fuel heat capacity result in lower peak fuel temperatures due to the higher thermal inertia of the fuel, as expected. Higher values of fuel-to-sheath heat transfer coefficients also result in lower peak fuel temperatures since more energy can escape from the fuel, but the sensitivity is lower than when the heat capacity is altered. In both cases, for each grain size considered, the variation is smooth and continuous with respect to variations in the input parameters.

These results are generally indicative of the type of results obtained from this exercise. No results were obtained which were obviously erroneous in nature, although in several cases the test matrix was too sparse to completely understand the code response, and these gaps will be filled with further testing. In general, the code response matched expectations which provides added confidence that the models are implemented properly and are free of significant errors.

CONCLUSIONS

It was found that the stress testing technique was an extremely powerful and inexpensive way to test simulation codes. The bulk of the effort is automated; the only manual effort is preparing test matrices and analysing the output. Since the analysis can largely be performed by visually inspecting plots of results, discontinuities that would otherwise be nearly impossible to detect are easily found.

REFERENCES

1. D. Evens and R. Rock, "Verification and Uncertainty Analysis of Fuel Codes Using Distributed Computing", presented at the Sixth Annual CANDU Fuel Conference, Niagara Falls, Ontario, September 26-29, 1999.

2. ELESIM and ELOCA documents.

3. H.E. Sills and Y. Liu, "FACTAR_SS 1.0: A Computer Program to Model Fuel Behaviour Under Normal Operating Conditions for a Single Channel in a Reactor Core, Program Description and User Manual", Ontario Hydro Reactor Safety and Operational Analysis Department report No. N-03553-940033, Rev. 0, April 1994.

4. C.J. Westbye, "FACTAR 2.0 (LOCA) User Manual", Ontario Hydro Reactor Safety and Operational Analysis Department report No. N-03553-965146, November 1996.

5. C.J. Westbye, memorandum to J.C. Luxat, "Assessment of the 'Stress Testing' Methodology as Applied to the FACTAR Code", Ontario Hydro File No. N-06631.01 T5 FACTAR.V&V Tests 0.0, April 22, 1998.

Figure 1
The Maximum Centerline Temperature Predicted by during the Transient

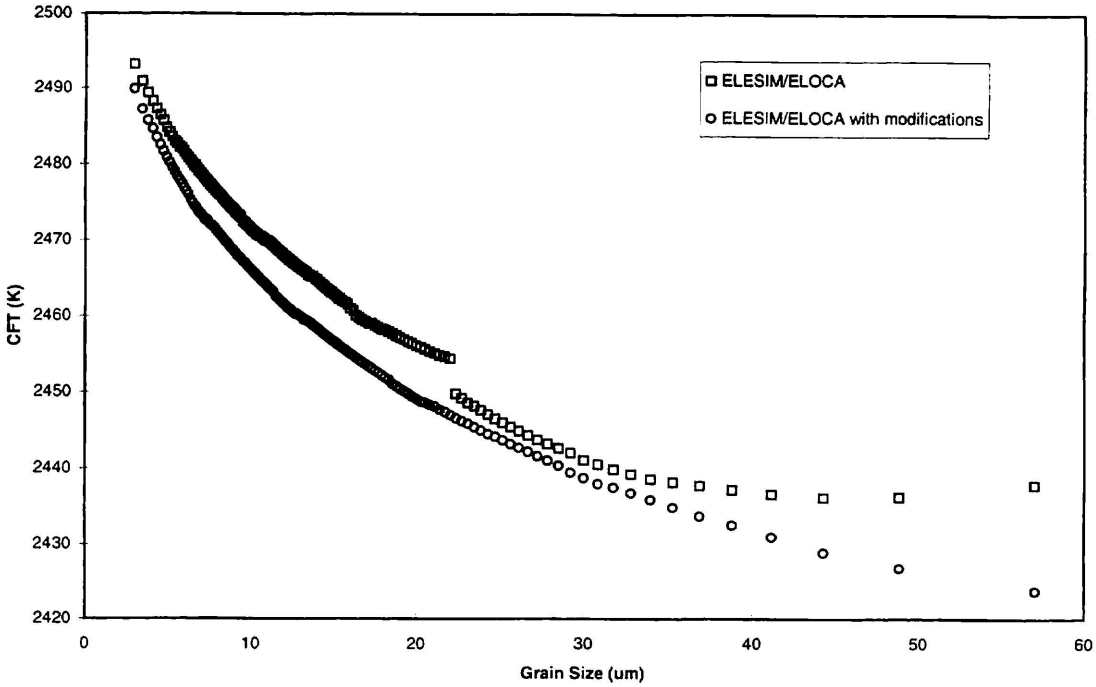

Figure 2
The Maximum Centerline Temperature during the Transient

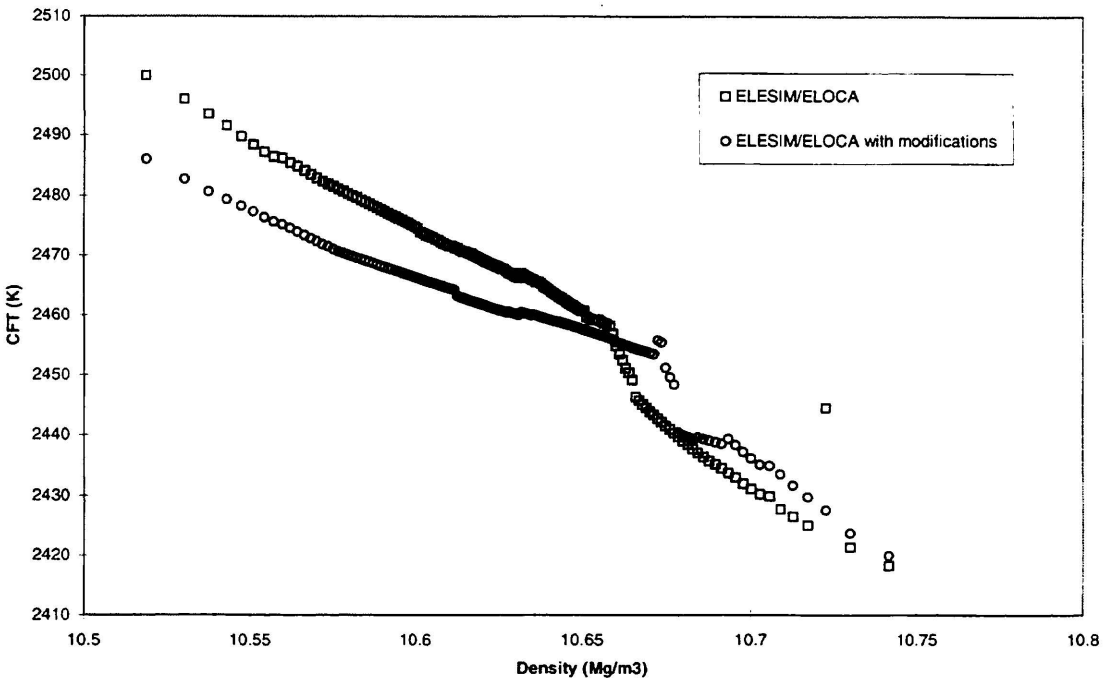

Figure 3

Variation in Maximum Sheath Temperature with Fuel Thermal Conductivity Offset

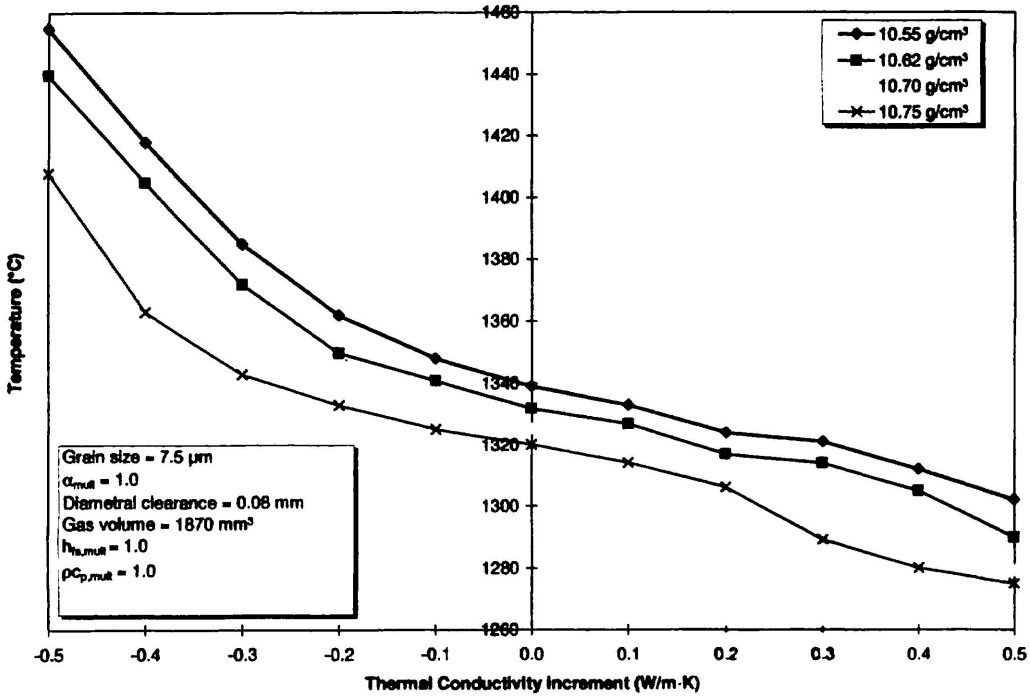

Figure 4

Variation in Maximum Fuel Centreline Temperature with Fuel Heat Capacity and Fuel/Sheath HTC