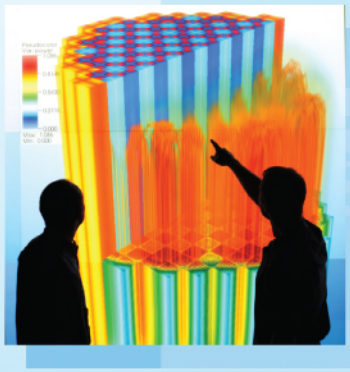
Power uprates
and plant life extension

CASL-U-2012-0237-000

Consortium for Advanced Simulation of LWRs



Engineering design
and analysis

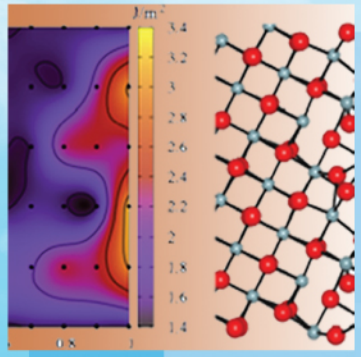
Enhancement of GTRF Modeling Fidelity

T. Li, P. Nath, N.K. Crane and T.-L. Sham

Sandia National Laboratories

March 31, 2012

Science-enabling
high performance
computing



Fundamental science

Plant operational data

CASL-U-2012-0237-000

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

Enhancement of GTRF Modeling Fidelity

T. Li, P. Nath, N.K. Crane and T.-L. Sham

Date Published: March, 2012

Prepared under the direction of the

U.S. Department of Energy

Office of Nuclear Energy

Advanced Modeling and Simulation Program

Consortium for Advanced Simulation of Light Water Reactors (CASL)
Energy Innovation Hub Project

CASL-U-2012-0237-000

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

CASL-U-2012-0237-000

Enhancement of GTRF Modeling Fidelity

T. Li, P. Nath, N.K. Crane* and T.-L. Sham

* Sandia National Laboratories

Date Published: March, 2012

Prepared under the direction of the

U.S. Department of Energy

Office of Nuclear Energy

Advanced Modeling and Simulation Program

Consortium for Advanced Simulation of Light Water Reactors (CASL)
Energy Innovation Hub Project

Prepared by

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6283
managed by
UT-BATTELLE, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CASL-U-2012-0237-000

Intentionally Blank

ACKNOWLEDGMENTS

We are grateful to the support by the U.S. DOE, Advanced Modeling and Simulation Program, Energy Innovation Hub Project, the Consortium for Advanced Simulation of Light Water Reactors (CASL). We gratefully acknowledge the programmatic directions provided by Brian Wirth (University of Tennessee), John Turner (ORNL) and Randall Summers (SNL).

Intentionally Blank

ABSTRACT

Grid to Rod Fretting (GTRF) is the most common cause of fuel failure in pressurized water reactor, and occurs due to flow induced vibration of the fuel rod causing a high-cycle fatigue interaction with the springs and dimples of the spacer grid. An objective of CASL is to develop high-fidelity Fluid-structural interaction tools to predict gap, turbulent flow excitation, rod vibration and wear so that designs can be optimized to reduce the potential for GTRF. This report describes the implementation of a wear model with oxide growth to enhance the GTRF modeling fidelity. A path forward plan to modify Sierra to incorporate variable wear and friction coefficients in the analysis is described. This report fulfills the completion requirement of CASL milestone L3:MPO.GTRF.P4.02 and L3:VRI.CM.P4.03.

Intentionally Blank

CONTENTS

Page

ACKNOWLEDGMENTS iii
ABSTRACT v
CONTENTS vii
LIST OF FIGURES ix
LIST OF TABLES ix
ACRONYMS xi
1. INTRODUCTION 1
2. IMPLEMENTATION OF WEAR MODEL IN SIERRA 2
2.1 HAYRAPETIAN-DEMKOWICZ-BLAU WEAR MODEL 2
2.1 Sierra wear subroutine 4
2.2 Access solution variables 7
2.3 Calculate wear quantities 7
2.4 Return the calculated wear quantities 8
3. TESTING OF WEAR SUBROUTINE 9
4. SIERRA ENHANCEMENT 11
4.1 Oxide growth model 11
4.2 Wear coefficient 11
5. CONCLUSION 13
REFERENCES 14
APPENDIX 15

Intentionally Blank

LIST OF FIGURES

Figure Page

Fig. 1 Scheme for oxide growth and fretting wear competition 3
Fig. 2. Workflow for calculations in wear subroutine 8
Fig. 3. Pin-on-slider geometry for the chatter wear verification problem..... 10
Fig. 4. Calculated total wear volume at a time of 1.47598 seconds..... 10
Fig. 5. Applied pressure time history for a location downstream from the mixing vanes 12
Fig. 6. Displacement time history at quarter-span location 12

LIST OF TABLES

Table **Page**

Table 1. index of *a* representing 4 different contact types..... 4
Table 2. Nodal contact output variables from Adagio/Presto accessed by the wear subroutine, and computed nodal variables in the wear subroutine that support the wear depth calculation 5
Table A. FORTRAN77 wear subroutine 15

Intentionally Blank

ACRONYMS

CASL	Consortium for Advanced Simulation of Light Water Reactors
DOE	Department of Energy
GTRF	Grid-To-Rod Fretting
MPO	Materials Performance and Optimization
ORNL	Oak Ridge National Laboratory
PWR	Pressurized Water Reactor
SNL	Sandia National Laboratories
VERA	Virtual Environment for Reactor Applications

Intentionally Blank

1. INTRODUCTION

In the U.S., 70% of the fuel leaking in pressurized water reactors (PWRs) is due to the Grid-to-Rod Fretting (GTRF) wear. Fuel rods are assembled in the spacer grid by friction forces between the fuel rods and the springs and dimples of the spacer grid. As the coolant flows through the fuel assembly, fluid forces generated by the flow field induce vibration in the fuel rod which causes small relative motions between the supports and the fuel rod and leads to the fretting wear. Moreover, due to the severe neutron flux working environment, the irradiation creep and growth will cause the cladding creep-down and the spacer grid grow resulting in preloads relaxation between the rod and the grid. Under these conditions, the spring and dimples may lose contacts with the fuel rod and form gaps between the spacer grid and the fuel rod. This magnifies the effects of flow-induced vibrations, introduces cyclic normal and tangential contact forces between the spring-dimple supports and the fuel rod, and further enhances fretting wear. The Consortium for Advanced Simulation of Light Water Reactors (CASL) has identified GTRF as one of the Challenge Problems that drive the requirement for the development and application of a modeling and simulation computational environment for predictive simulation of light water reactors. This environment, known as Virtual Environment for Reactor Applications (VERA), will incorporate science-based models, state-of-the-art numerical methods, modern computational science and engineering practices, and uncertainty quantification and validation against data from operating light water reactors.

Simulations of GTRF involve turbulence flow, structural dynamics, contacts, and wear. In addition, the changes in the mechanical behavior and wear characteristics of the materials in the irradiation environment during reactor operations have a direct effect on the GTRF mechanisms. An initial framework of the wear model was discussed in the milestone report of Hayrapetian, et al. (2011a). The finite element implementation of a wear model where the contacts are between two Zircaloy surfaces is described in the CASL milestone report by Sham, et al. (2012).

In this report, a wear model that includes oxide growth, Hayrapetian, et al. (2011b), is implemented in Sierra. Due to restrictions in Sierra Release 4.25, the wear model with oxide growth was only implemented for constant wear and friction coefficients. Generally, the wear and friction coefficients would depend on the material characteristics of the contact surfaces, e.g., zircaloy on zircaloy, zircaloy on zirconium oxide, or zirconium oxide on zirconium oxide. Within each time increment, whether the contact surface is zircaloy or zirconium oxide is resulted from a competition between oxide growth and wear. Variable wear and friction coefficients will be implemented in Sierra once the required enhancement is in place.

Section 2 discusses the implementation of the enhanced wear model accounting for the wear and oxide growth competition, Section 3 provides the details of the testing and the result of the wear subroutine, Section 4 lists the required simulation capability enhancements and future considerations; and finally, Section 5 gives the conclusion.

2. IMPLEMENTATION OF WEAR MODEL IN SIERRA

2.1 HAYRAPETIAN-DEMKOWICZ-BLAU WEAR MODEL

Hayrapetian et al. (2011a) has developed a "work-rate" model as the initial wear model for GTRF, where the wear rate of any contact point is related to the frictional power dissipated at that location. The Hayrapetian-Demkowicz-Blau wear model given in Hayrapetian et al. (2011a) employs a depth-based formulation where the work-rate is localized to a differential contact area. The total wear depth $D(x,y,t)$ is given by

$$D(x,y,t) = \int_0^t K q_t(x,y,t') v_t(x,y,t')\, \mathrm{d}t' \tag{1}$$

where x and y represent the coordinates on the wear surface, K is the wear coefficient, q_t is the shear traction, and v_t is the tangential relative slip velocities (i.e., tangential velocity jump across the contact surfaces at the point x and y). Hayrapetian et al. (2011a) recommend an expression for K as

$$K = C\exp(-l \cdot P), \quad C = 3.888\times10^{-6}\,\mathrm{mm^3N^{-1}m^{-1}}, \quad l = 0.05739\,\mathrm{N^{-1}}, \quad 10\,\mathrm{N} \le P \le 30\,\mathrm{N} \tag{2}$$

where P is the normal contact force in newton.

The total wear depth D in Eq. (1) is associated with the two contact surfaces and D represents the sum of the wear depths from both contacting materials. In order to partition the total wear depth into contributions for the cladding and for the spacer grid, the following expressions were recommended by Hayrapetian et al. (2011b):

$$\begin{aligned} D_{clad}(x,y,t) &= \gamma D(x,y,t), \quad 0 \le \gamma \le 1 \\ D_{grid}(x,y,t) &= (1-\gamma) D(x,y,t) \end{aligned} \tag{3}$$

where D_{clad} and D_{grid} are the wear depths for the fuel clad and the spacer grid, respectively and the wear partition parameter γ is an empirical factor. For identical surfaces, Hayrapetian et al. (2011b) recommended a value of $\gamma = 0.5$. It is noted that the wear coefficient and the wear partition parameter would in general depend on the specific combination of contacting materials. This also applies to the friction coefficient.

In a subsequent effort, Hayrapetian et al. (2011b) extended the wear model by including the effect of surface oxide growth. This is an important development because the presence of zirconium oxide layer over the Zircaloy surface can influence the wear characteristics. In the local contacting zone of the spring and dimples of the grid and the fuel cladding, the omnipresent corrosive environment causes the growth of an oxide layer on the surface of the zirconium. On the other hand, the contact fretting wears materials off the surface. Thus, the competition between the wear and the oxide growth determines the material type (i.e. zirconium or oxide) that participating in the contact. This leads to three different surface contact scenarios, i.e., oxide-oxide contact, zirconium-zirconium contact, and zirconium-oxide contact. Each scenario has different friction coefficient and wear coefficients in the wear model. And a local contact type can change from one to the other over time depending on corrosion and fretting wear processes. Therefore, oxide growth has great influence to the GTRF problem and tracking the presence of the oxide layer becomes necessary in the incremental wear calculation.

The oxide growth rate recommended by Hayrapetian et al. (2011b) is assumed to be related to the current oxide thickness and time which is,

$$\dot{G}_\alpha = k\left(S_\alpha, t\right) \tag{4}$$

where S_α is the oxide thickness; $\alpha = 1$ and $\alpha = 2$ represent for the fuel clad and the grid respectively; and k is the oxide rate function. Combining Eqs. (1), (3), and (4), within each incremental incidence, each contact surface will undergo an oxide growth and fretting wear competition, and the oxide thickness growth rate would be,

$$\dot{S}_\alpha = \dot{G}_\alpha - \dot{D}_\alpha \tag{5}$$

where $\dot{D}_\alpha$ is the wear rate for the fuel clad or grid. If integrate Eq. (5) explicitly, within each time increment, the oxide thickness at the end of the time step $S_\alpha^{t+\Delta t}$ is

$$S_\alpha^{t+\Delta t} = \left\langle S_\alpha^t + k\Delta t - \Delta D_\alpha \right\rangle \tag{6}$$

where S_α^t is the oxide thickness at the beginning of the time step; and Macaulay bracket means that if the fretting wear is more than the oxide depth grew in the current time step plus the oxide thickness on the surface, the oxide layer will be totally worn out and the oxide thickness should be zero. A scheme of the fretting wear and oxide competition is shown in Fig. 1. The surface after each time increment can be either increase if the oxide growth rate is more than the fretting wear rate or recede if the wear rate is more than the oxide growth rate. One thing need to be concerned is the sequence of oxide growth and fretting wear. For example, assuming the surface at time *t* in Fig. 1 is zircaloy and within the time step, if we first calculate wear depth and then add up the oxide growth, the final surface level would be the same as if we first add up the oxide growth and then calculate the fretting wear. However, the local material combination would not be the same because in the first scenario, the wear of the zircaloy is clearly more than the wear of the zircaloy in the second scenario. And it is the same for the oxide thickness. Moreover, when the two surfaces are in contact, if the oxide will still grow is also unknown. Therefore, what happens in the reality need to be addressed in order to acquire the correct localize oxide thickness information. In this report, we will always calculate oxide growth first and then count in the fretting wear assuming the oxide grows in an unimpeded manner on both the clad and the grid.

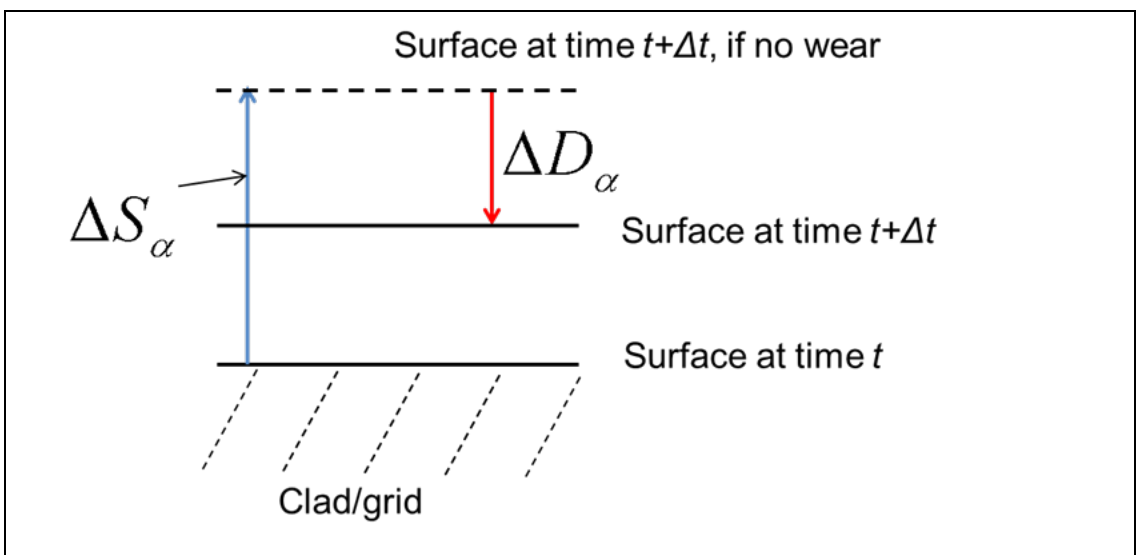


Fig. 1 Scheme for oxide growth and fretting wear competition

At the beginning of each time step, there are four different contact types as listed in Table 1 based on

the material exposing on the clad and the grid. Either the clad or the grid can be covered by oxide layer or its base material. From the literature reviewed, Hayrapetian et al. (2011b) do not recommend any difference in wear coefficient, K^a, for different combinations of contacting surfaces. However, they do recommend using a different coefficient of friction in calculating the contact tractions. Hayrapetian et al. (2011b) recommend the range of friction coefficient for three different surface combinations are, between 0.12 and 0.26 for zirconia on zirconia, between 0.20 and 0.35 for zirconia on zirconium alloy, and between 0.28 and 0.38 for zirconium alloy on zirconium alloy. In the current structural analysis, a value of 0.3 is used for the coefficient of friction for all contact surface combinations and the wear coefficient used is as in Eq. (2).

Table 1. index of *a* representing 4 different contact types

a	Clad surface	Grid surface
1	Metal 1	Metal 2
2	Metal 1	Oxide 2
3	Oxide 1	Metal 2
4	Oxide 1	Oxide 2

For the finite element implementation of the wear model considering the oxide growth, one first has to know which part of the assembly (i.e. the fuel clad or the grid) does the current node belong to in order to determine Eq. (3). Then, one can know what is the material exposing to contact for the current node by reading the oxide thickness stored as internal variable. If the oxide thickness is larger than zero, it is oxide exposing to contact. If the oxide thickness is zero, the node is wearing on the zircaloy. At last, one has to know what material that the current point is contacting with (i.e. zircaloy or oxide), so that the wear coefficients can be determined. The friction coefficient that used in Sierra needs to be determined from the above information in the same manner. In the current analysis, because of the difficulty for the wear routine to know what material it is contacting with, the fuel clad and the grid use the same wear coefficients. A sierra contact routine to pass information of nodes on the opposite surface to user routines is under development and a framework will be discussed in the later section of this report.

The implementation of the Sierra wear subroutine discussed above is described in the following section, as a compliment to the workflow outlined in the milestone report, ORNL-TM-2012-3-MPO.P3.04 (Sham, et al. 2012).

2.1 Sierra wear subroutine

A Sierra wear subroutine has been developed in FORTAN77 to perform the wear depth calculation for each time increment in the structural analysis. The ‘raw variable access method’ is used to access the solution data structure of Sierra and to retrieve the nodal contact parameters given in Table 2 for calculating the wear depth D of Eq. (1) and the oxide growth in Eq. (6). The magnitude of the incremental slip vector, Δu_t, tangential to the contact surfaces respectively, are calculated as indicated in Table 2.

Table 2. Nodal contact output variables from Adagio/Presto accessed by the wear subroutine, and computed nodal variables in the wear subroutine that support the wear depth calculation		
Variable	Sierra contact variable name	Description
flag	`contact_status`	Status of the interactions at the node. Possible values are as follows: 0.0 = Node is not a contact node (not in a defined contact surface) 0.5 = Node is not in contact 1.0 = Node is in contact and is slipping −1.0 = Node is in contact and is sticking
$\underset{\sim}{n}$	`contact_normal_direction`	Vector direction of the constraint. This is, in general, the normal of the face in the interaction.
$\underset{\sim}{t}$	`contact_tangential_direction`	Vector direction of the contact tangential force.
P	`contact_normal_force_magnitude`	Magnitude of the contact force at the node in the direction normal to the contact face (`contact_normal_direction`).
q_n	`contact_normal_traction_magnitude`	Traction normal to the contact face, i.e., `contact_normal_force_magnitude` divided by `contact_area`. If there are multiple interactions for this node, the traction only for the last interaction is given.
q_t	`contact_tangential_traction_magnitude`	Traction in the plane of the contact face, i.e., `contact_traction_force_magnitude` divided by `contact_area`. If there are multiple interactions for this node, the traction only for the last interaction is given.
Δu	`contact_incremental_slip_magnitude`	Magnitude of incremental slip over the current time step.
$\underset{\sim}{s}$	`contact_incremental_slip_direction`	Normalized direction of incremental slip over the current time step.
area	`contact_area`	Contact area for the node. This is the tributary area around the node for this interaction. If there are multiple interactions, the reported area is the area associated with the last interaction.
Δt	`delta_t`	Current time increment
Calculated nodal variable		Description
$\Delta \underset{\sim}{u} = \Delta u \ \underset{\sim}{s}$		Incremental slip vector
$\Delta u_n = \lvert \Delta \underset{\sim}{u} \cdot \underset{\sim}{n} \rvert$		Magnitude of incremental slip vector normal to the contact surface
$\Delta u_t = \lvert \Delta \underset{\sim}{u} \cdot \underset{\sim}{t} \rvert$		Magnitude of incremental slip vector tangential to the contact surface

Table 2. Nodal contact output variables from Adagio/Presto accessed by the wear subroutine, and computed nodal variables in the wear subroutine that support the wear depth calculation	
depth(8)	depth()=: 1: total material worn of the clad and the grid 2: total worn of the current clad/grid 3: wear rate of the clad and the grid 4: total wear volume of the clad and the grid 5: total oxide depth grew of the clad and the grid 6: total oxide depth grew of the clad/grid 7: oxide thickness of the clad/grid 8: total zircaloy worn depth of the clad/grid

Using these nodal quantities, the incremental total wear depth ΔD_k at a node for a given time increment k is calculated as

$$\Delta D_k = K \times q_t \times \Delta u_t \tag{7}$$

(and the cumulative nodal total wear depth D up to the Nth time increment is given by

$$D = \sum_{k=1}^{N} \Delta D_k \ . \tag{8}$$

The nodal total wear depth rate $\dot{D}_k$ is calculated as

$$\dot{D}_k = \Delta D_k / \Delta t \ . \tag{9}$$

The incremental total wear volume ΔV_k at a node for a given time increment k is calculated as

$$\Delta V_k = \Delta D_k \times area_k \tag{10}$$

and the cumulative nodal total wear volume V up to the Nth time increment is given by

$$V = \sum_{k=1}^{N} \Delta V_k \ . \tag{11}$$

The wear depth rate and wear volume for the fuel clad and spring and dimples are apportioned similarly as

$$\begin{aligned} \dot{D}_1(x,y,t) &= \gamma \dot{D}(x,y,t), \quad \dot{D}_2(x,y,t) = (1-\gamma)\dot{D}(x,y,t) \\ V_1(x,y,t) &= \gamma V(x,y,t), \quad V_2(x,y,t) = (1-\gamma)V(x,y,t) \ . \end{aligned} \tag{12}$$

The workflow of the calculations in the wear subroutine is shown in Fig. 2. For each time increment of the analysis, it involves retrieving the necessary nodal contact variables, calculating the wear quantities, and returning the calculated wear quantities to Sierra.

The wear subroutine is a so-called ‘node set subroutine’ in Sierra as it operates on and returns values of nodal quantities. The interface to the subroutine is as follow:

```
subroutine wear_calc(int num_nodes, int num_vals, real eval_time,
int nodeID[], real values, int flags, int ierror)
```

where

`num_nodes =` (integer) number of nodes passed in by Sierra.
`num_vals =` (integer) number of values per nodes passed in by Sierra.

`eval_time` = (real) time at which the wear quantities are evaluated
`nodeID[]` = (integer) node set passed in
`values` = (real) calculated wear values that are returned per node

There are three calculated wear values in the current implementation and values is declared as real values(3), currently, no information is exchanged with Sierra through values().

`flags` = (integer) flag for Sierra to interpret current returned values
`ierror` = (integer) error code returned by the subroutine, a non-zero value implies an error

A check is included in the subroutine to check if the number of nodes passed in is zero, in the following manner.

```
if(num_nodes.eq.0) then
   ierror = 0
   return
endif
```

The wear subroutine is written in FORTRAN77 which is currently supported by Sierra. The code is included in the Appendix. It works with static arrays, and hence all arrays have to be pre-allocated. Thus, the number of nodes passed in and the size of the corresponding nodal contact variables need to be checked against the pre-allocated space. If the pre-allocated space is not large enough, an error code `ierror` is returned to Sierra.

2.2 Access solution variables

The availability of a defined solution variable is first checked by using the query function `aupst_check_node_var`. If it is available the query function `aupst_get_node_var` is used to retrieve the nodal variable. For example, to retrieve the contact variable `contact_normal_direction` and to store it in a variable called `cndir` in the subroutine, the syntax is:

```
call aupst_get_node_var(num_nodes, num_comp_check, nodeID, cndir,
"contact_normal_direction", ierror)
```

To retrieve the current time increment, `delta_t`, which is a global variable in Sierra, and to store it in a local variable `time_step`, the query function `aupst_get_global_var` is used:

```
call aupst_get_global_var(1, time_step, "delta_t", ierror)
```

2.3 Calculate wear quantities

The local variable `"depth"` is a user-defined nodal variable with eight components, and it is required to be specified in the input file with initial values. The values for this nodal variable at each time instant are calculated and updated in the subroutine. The eight components are:

1) The total material has worn from both the clad and the grid (i.e. Eq. (8))
2) The total material has worn for the current node (i.e. clad or grid)
3) The wear rate of the clad and the grid (i.e. Eq. (9))
4) The total worn volume of the clad and the grid (i.e. Eq. (11))
5) The total oxide grew on the clad and the grid
6) The total oxide grew on the current node (i.e. clad or grid)
7) oxide thickness of the current node (i.e. clad or grid)
8) total zircaloy depth has worn of the current node (i.e. clad or grid)

The user-defined nodal variable "depth" needs to be defined in the input file. The syntax is as follows:

```
begin USER VARIABLE depth
   TYPE = NODE real length=8
   INITIAL VALUE =0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
   USE WITH RESTART
end USER VARIABLE depth
```

To tell the wear subroutine if the current node belongs to the clad or the grid, the following needs to be included in the input file:

```
begin USER OUTPUT
   block = block set names
   node set subroutine = subroutine name
   subroutine integer parameter: is_grid = 1
   end
end
```

where the variable "is_grid" is 1 for the grid nodes and 0 for the clad nodes.

2.4 Return the calculated wear quantities

The query function `aupst_put_node_var` is used to return the calculated wear quantities stored in the local vector `depth` to the user-defined nodal variable `depth` in Sierra. The syntax is as follow:

```
call aupst_put_node_var(num_nodes, num_comp_check, nodeID, depth,
"depth", ierror)
```

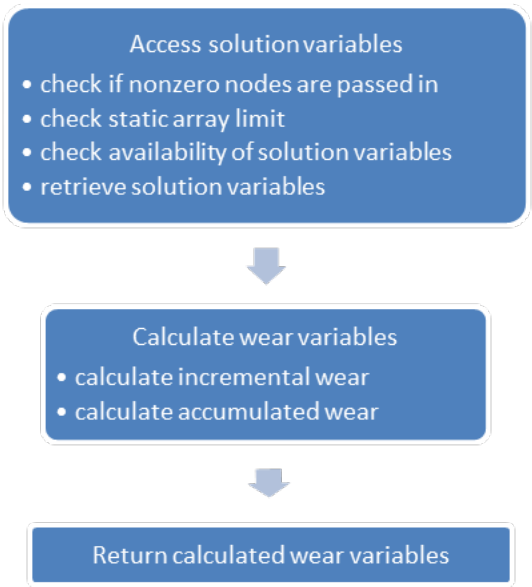


Fig. 2. Workflow for calculations in wear subroutine

3. TESTING OF WEAR SUBROUTINE

We use the same testing problem in our last MPO report (Sham, et al., 2012) to test the new subroutine. A pin under a chattering load is in contact with a slider, which is shown in Fig. 3. The circular pin with 1.5 m in diameter and 0.35 m in height, is sitting on a slider which is 10 m in length, 2 m in width and 0.25 m in height. The top 0.1 m part of the pin is modeled as rigid body, and the rest of the pin and the slider are modeled with 3D continuum elements. All the translational degrees of freedom of the slider are fixed. The pin is constrained on the top rigid part from any rotation about the three mutually perpendicular axes. The displacements of the rigid part along the z-direction are also prohibited. Vertical force along the y-direction is applied on all nodes of the rigid part of the pin to introduce contacts between the pin and the slider and generates contact tractions and relative slips at the pin-slider contact surfaces. This force quickly ramps up in 0.1s and remains constantly to a total duration of 3 seconds. The pin is held steady for the first 2s and then starts sliding in the *x* direction. The wear subroutine retrieves various contact variables from Sierra and performs calculations to determine the wear and oxide quantities, as described above. The material properties for the deformable part of the pin and slider are: Young's modulus = 7.42 X10^6 Pa, Poisson's ratio = 0.3, and density = 100 kg/m^3. For a given total wear between the pin and the slider, it is assumed that the contacting type is not important and 70% of it is contributed by the slider and 30% from the pin (i.e. $\gamma^a = 0.3$).

The oxide growth model in Eq. (4) is assumed to be a linear relationship with time, i.e.,

$$\Delta G_\alpha = k \cdot t \tag{13}$$

where *k* is the mean oxide growth rate suggested to be 0.01~0.1 $\mu m \cdot d^{-1}$ (Hayrapetian et al., 2011b). This report use $k = 0.025$ represents the oxide growth for Zircalloy-4 in water at 360°C. It is worth noting that the finite element implementation discussed in this report is in general expression and allows for implementation of the more generalize oxide growth law in Eq. (4).

During the computation, all the nodal contact variables retrieved from Sierra and the wear and oxide quantities calculated in the subroutine for each increment are stored as internal variables which are accessible from the Sierra output file using visualization software such as *Ensight* or *VisIt*. The computation is solved with an explicit dynamic solver, Presto, provided by Sierra. A representative result at time=2.54s is shown in Fig. 4. The wear depths of the pin and the slider are shown in colored spectrum. The pin is clockwise rotated with 90 degree along the *x* axis and then translated along the *y* axis in order to show the contact surface. Because the pin pressed steadily for 2s and slid afterwards, the red colored zone near the edge of the slider has the most wear of the slider. The contact surfaces of the pin and the slider have different wear depth distribution as the current model discriminate the wear of the pin and the slider.

To verify the combination model of the wear and the oxide growth, the time history of the wear depth, the oxide growth, and the oxide thickness of a single node are shown in the embedded plots in Fig. 4. The picked node is shown as the white dot on the slider. From the wear depth versus time plot, there is no wear for the first 2s until the pin start to slide and introduce tangential slip between the contact surfaces. As a consequence, the wear depth begins to grow and remains a constant when the pin slides away from the node. In the oxide growth versus time plot, the oxide growth shows a linear relationship with the time as the Eq. (4) suggests. After combing the wear and oxide growth together, the oxide thickness versus time curve in green shows nonlinearity. The oxide thickness first increases linearly with time since there is no wear on the slider in the first 2s. Then the pin starts to slide and wears out the oxide layer on the slider leading to the oxide growth rate drop. After the picked node no longer feel the contact force from the pin, the oxide thickness will grow linearly with time again.

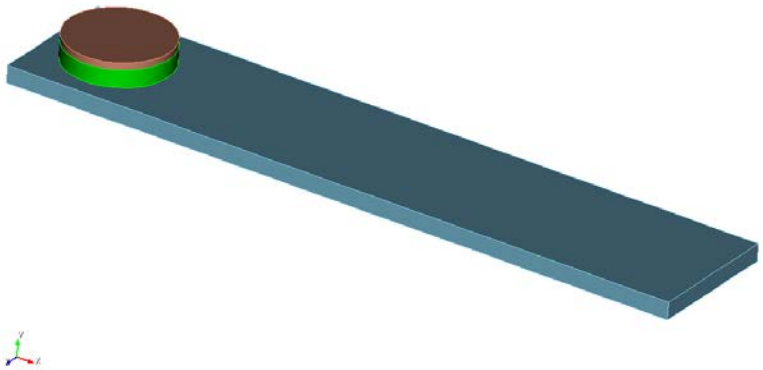


Fig. 3. Pin-on-slider geometry for the chatter wear verification problem

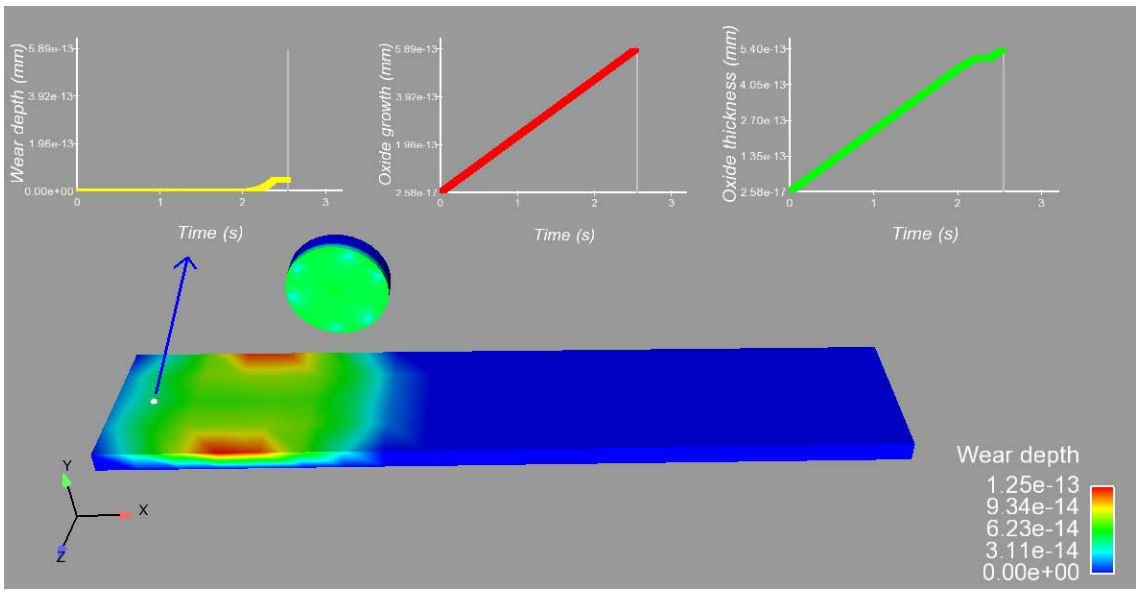


Fig. 4. Calculated total wear volume at a time of 1.47598 seconds

4. SIERRA ENHANCEMENT

The current wear model can be improved in following matters in the future.

4.1 Oxide growth model

The oxide growth model right now is assumed to be linear with time. However, experiments have shown that the growth rate is related to the current oxide thickness and has a cubic periodic behavior (Hayrapetian et al, 2011a and 2011b). The current model also omits the effects of heat flux from the nuclear fuel, neutron irradiation, and the presence of hydride. A more sophisticate and robust oxide growth model based on real physical mechanism which will count in all issues mentioned above would be a good attempt in order to capture the contact defects in the GTRF problem.

4.2 Wear coefficient

Another improvement of the current model is the wear and friction coefficients. For the current model, the wear and friction coefficients are constant and irrelevant with the contact type. However, in the reality, it should be a variable parameter that depends on if the contact is between the oxide and oxide, oxide and zircaloy, and zircaloy and zircaloy. In order to determine the contact type, each node on the contact surface has to know the material information of the surface that it is contacting with. For example, as is shown in Fig. 5, for any node in blue on the contact surface of the Rod, the current surface information of this node can be retrieved from its internal variable, i.e. “oxide thickness”. If the oxide thickness for this node is zero, then the node is zircaloy. And if the oxide thickness is larger than zero, then the node is oxide. In order to determine the contact type and the friction and wear coefficients of this node, one has to know if the node is contacting on oxide or zircaloy. Therefore, each node on the rod contact surface has to know the information of all nodes which belong to the surface that it is contacting with. It creates difficulty because those nodes may not be assigned to the same processor, and the subroutine in one processor cannot get information from other processors. Another difficulty is the internal variable updating algorithm. In order to post-process the wear and oxide information, it is suggested to store those values as the internal variables of each node. Current wear subroutine will update the internal variables after each call to keep tracking the wear and the oxide thickness. However, when introducing variable friction coefficient in the contact routine, Sierra has to keep calling the wear subroutine during the contact algorithm because each time the contact algorithm change the contact force during iteration, the wear coefficients has to be changed accordingly. This may lead to a scenario that the wear subroutine will keep adding the oxide thickness and the wear information after each call and large error is convoyed because the wear subroutine should not update information if convergence in the contact iteration is not achieved.

To address the above issues, we have set up a new framework as is shown in Fig. 6 to implement in the next milestone. In Sierra level, two new data are created for each node in contact algorithm, i.e. `oxide_curr` and `oxide_oppo`. `Oxide_curr` will store the oxide thickness of the node as well as the `oxide_oppo` will store the oxide thickness of the point that the node is contacting with. During the contact iteration, both `oxide_oppo` and `oxide_curr` will be passed into the wear subroutine, and the wear subroutine will determine the contact type, compute the wear and oxide growth, and pass back the contact force and updated `oxide_curr`. However, at the end of the wear subroutine, all those wear and oxide information won’t be stored to the internal variables in the wear subroutine, but are merely passed back to Sierra. After Sierra gets this information, it will keep iterating the contact algorithm and calculate the new `oxide_oppo` based on `oxide_curr` and contact information until a converged solution is achieved. Then, after a stable solution is obtained, Sierra will send the `oxide_oppo` and `oxide_curr` that it used in the converged solution to the wear subroutine one more time, and the wear subroutine will store those to its internal variables for post-processing.

Although this method increases the memeroy usage because two more global like variables are

introduced, it will finish the task accurately and effectivly.

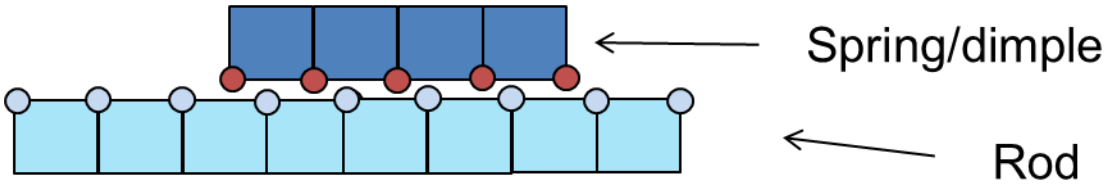


Fig. 5. Applied pressure time history for a location downstream from the mixing vanes

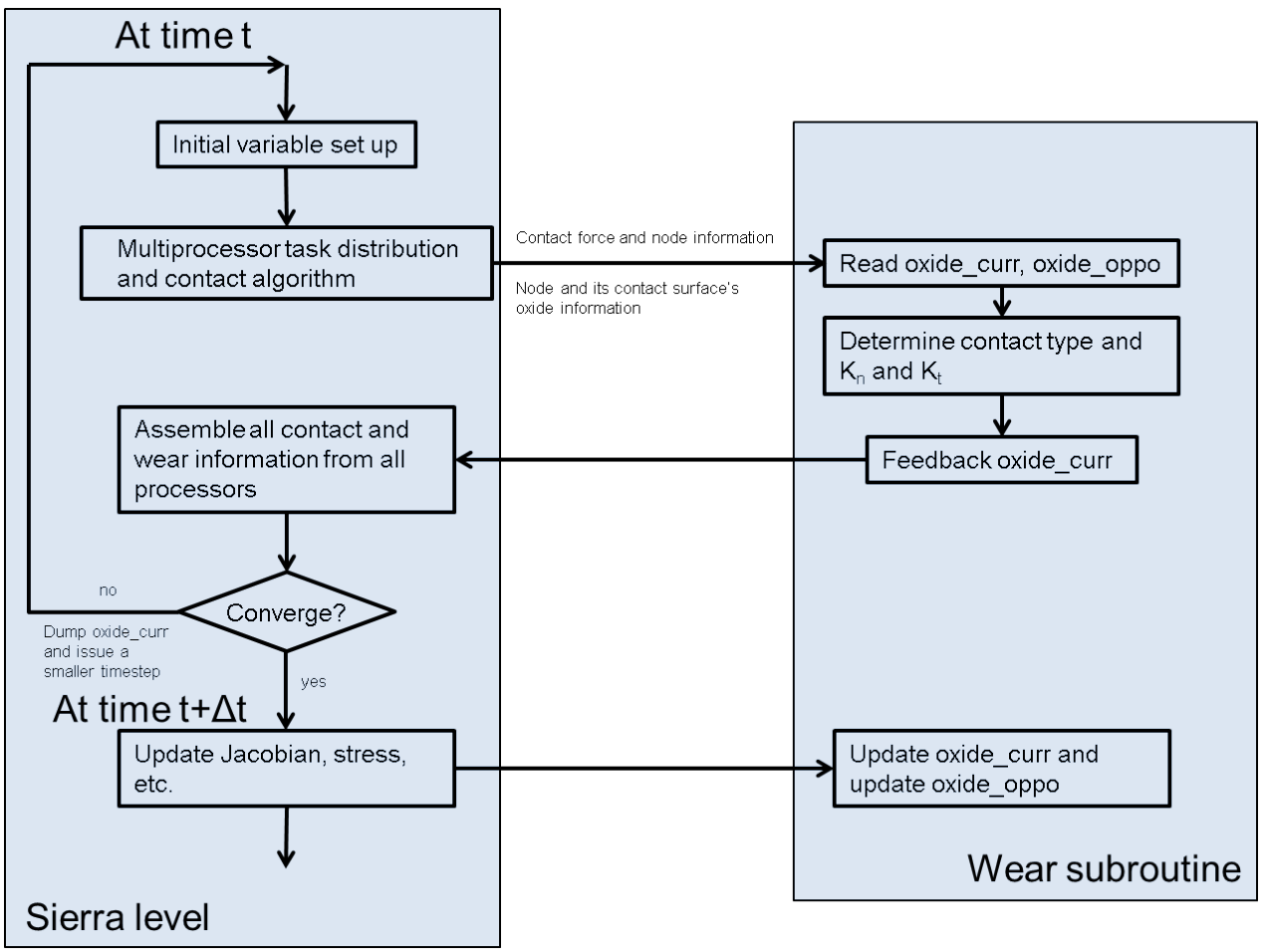


Fig. 6. Displacement time history at quarter-span location

5. CONCLUSION

The enhanced MPO wear models of Hayrapetian et al. (2011a, 2011b) have been implemented as a Sierra user subroutine for the calculation of fuel clad and spacer grid wear depths and oxide growth. A benchmark case is tested to verify the wear model and subroutine. Required capability enhancements and future considerations have been outlined for improvements in the ability to change the friction and wear coefficients.

REFERENCES

Hayrapetian, A.V., Demkowicz, M.J., and Blau, P.J. (2011a) "Initial wear model for VI implementation for GTRF challenge problem," CASL Milestone Report, L4.MPO.VI.P3.01, Oak Ridge National Laboratory, Oak Ridge, TN, June.

Hayrapetian, A.V., Demkowicz, M.J., and Blau, P.J. (2011b) "Updated wear model accounting for surface oxide layers for use in GTRF simulations," CASL Milestone Report, L3.MPO.GTRF.P4.01, Oak Ridge National Laboratory, Oak Ridge, TN, December.

Sham, T.-L., Crane, N.K., Nath, P., Hu, Z., Turner, D.Z., Li, T., and Sankaran, R. (2012) "Three-Dimensional, High Resolution, Coupled Fluid/Materials Demonstration and Assessment of Grid-to-Rod Fretting (GTRF) Test Problem," CASL Milestone Report L2: MPO.P3.04, ORNL/TM-2012/3, Oak Ridge National Laboratory, Oak Ridge, TN, January.

APPENDIX

Table A. FORTRAN77 subroutine for wear with oxide growth, constant wear and friction coefficients

```
c    user subroutine wear_calc
c       with signature void ()(int const*, int const*, double const*,
c          int const*, double*, int*, int*)
      subroutine wear_calc(num_nodes, num_vals, eval_time,
     $   nodeID, values, flags, ierror)

      implicit none
      integer ispace
      parameter (ispace = 100000)

      integer num_nodes
      integer num_vals

c     current evaluation time
      double precision eval_time

c     array of node number for nodes processed
      integer nodeID(num_nodes)

c     number of values returned per node passed in
      double precision values(3)

      integer flags(1)
      integer ierror
      integer ierror1
      integer ierror2
      integer ierror3
      integer ierror4
      integer inode

c     100000 is of entities that can be processed using static arrays
c     it may be necessary to change this for a given mesh / cpus

ccccccc     contact related variables

c     cndir is sierra variable "contact_normal_direction"
      double precision cndir(3,ispace)

c     ctdir is sierra variable "contact_tangential_direction"
      double precision ctdir(3,ispace)

c     cnrmag is sierra variable "contact_normal_traction_magnitude"
      double precision cnrmag(ispace)

c     cnfmag is sierra variable "contact_normal_force_magnitude"
      double precision cnfmag(ispace)

c     ctrmag is sierra variable "contact_tangential_traction_magnitude"
      double precision ctrmag(ispace)

c     cslmag is sierra variable "contact_incremental_slip_magnitude"
      double precision cslmag(ispace)

c     csdir is sierra variable "contact_incremental_slip_direction"
      double precision csdir(3,ispace)

c     cs is sierra variable "contact_status"
```

Table A. FORTRAN77 subroutine for wear with oxide growth, constant wear and friction coefficients

```
      double precision cs(ispace)

c     carea is sierra variable "contact_area"
      double precision carea(ispace)

c     components are wear depth, wear depth rate, wear volume
c     oxide thickness increment
      double precision ddoxide,dwear,dwear_zr

c     judge if it is grid or rod
      integer is_grid

c     gamma for the grid and rod
      double precision gamma_value
      parameter (gamma_value = 0.3d0)

cccccc    Time step information

c     time_step is sierra variable "delta_t"
      double precision time_step

cccccc   wear related variales

c     k_t is wear depth slip increment coefficient
      double precision k_t(ispace)

c     k_n is wear depth normal increment coefficient
      double precision k_n(ispace)

c     ddepth is incremental wear depth
      double precision ddepth(ispace)

c     components are wear depth, wear depth rate, wear volume,
c                     current oxide thickness, total oxide grown,
c                     wear of the zircaloy
c     depth is sierra analysis variable "depth" defined in input file
      double precision depth(8,ispace)

c     used in aupst procedure to get / check variable type
c     scalar / vector / tensor
      integer num_comp_check

c     depth is defined as a vector with eight components
      integer num_depth_components
      num_depth_components=8

c     proceed if num_nodes passed in is greater than zero
c     number of nodes passed in for the processor in an increment

      if(num_nodes .eq. 0) then
         ierror = 0
         return
      end if

c     check data passed in will fit into the
c     statically allocated array

      if(num_nodes .gt. ispace) then
         write(6,*) 'ERROR in sphere disp,
     $      num_nodes exceeds static array size: ',num_nodes
         ierror = 1
```

Table A. FORTRAN77 subroutine for wear with oxide growth, constant wear and friction coefficients

```
          return
       end if

cccccc accessing information from sierra analysis

c    aupst_check_node_var() checks availability of "variable"

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_normal_direction", ierror)

      if(ierror .ne. 0) return

c    num_comp_check to check if scalar, vector, tensor
c    contact normal direction vector has three components

      if(num_comp_check .ne. 3) return

c    aupst_get_node_var() gets "variable"
c    gets variable "contact_normal_direction", stores in cndir(3,ispace)

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, cndir, "contact_normal_direction", ierror)
c
      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_tangential_direction", ierror)

      if(ierror .ne. 0) return
      if(num_comp_check .ne. 3) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, ctdir, "contact_tangential_direction", ierror)

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_status", ierror)

      if(ierror .ne. 0) return
      if(num_comp_check .ne. 1) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, cs, "contact_status", ierror)

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_normal_force_magnitude", ierror)

      if(ierror .ne. 0) return
      if(num_comp_check .ne. 1) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, cnfmag, "contact_normal_force_magnitude", ierror)

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_normal_traction_magnitude", ierror)

      if(ierror .ne. 0) return
      if(num_comp_check .ne. 1) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, cnrmag, "contact_normal_traction_magnitude", ierror)

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_tangential_traction_magnitude", ierror)
```

Table A. FORTRAN77 subroutine for wear with oxide growth, constant wear and friction coefficients

```
      if(ierror .ne. 0) return
      if(num_comp_check .ne. 1) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, ctrmag, "contact_tangential_traction_magnitude",
     $   ierror)

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_incremental_slip_magnitude", ierror)

      if(ierror .ne. 0) return
      if(num_comp_check .ne. 1) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, cslmag, "contact_incremental_slip_magnitude", ierror)

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_incremental_slip_direction", ierror)

      if(ierror .ne. 0) return
      if(num_comp_check .ne. 3) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, csdir, "contact_incremental_slip_direction", ierror)

      call aupst_check_node_var(num_nodes, num_comp_check,
     $   nodeID, "contact_area", ierror)

      if(ierror .ne. 0) return
      if(num_comp_check .ne. 1) return

      call aupst_get_node_var(num_nodes, num_comp_check,
     $   nodeID, carea, "contact_area", ierror)

c    global variable of "delta_t" for time increment
      call aupst_get_global_var(1, time_step, "delta_t", ierror)

c    check for "depth" vector as defined in sierra analysis input file

      call aupst_check_node_var(num_nodes,num_depth_components,
     $   nodeID, "depth", ierror1)

c    get "depth" vector store in depth(8,100000)

      call aupst_get_node_var(num_nodes,num_depth_components,
     $   nodeID, depth, "depth", ierror4)

c    get the information for grid or rod
      call aupst_get_integer_param("is_grid",is_grid,ierror)
      if(ierror .ne. 0) then
         write(6,*) 'ERROR in isgrid,',is_grid,ierror
         return
      end if
      if ((is_grid.eq.1).or.(is_grid.eq.0)) then
      else
         write(6,*) 'ERROR in isgrid,',is_grid,ierror
         ierror = 1
        return
      end if
```

Table A. FORTRAN77 subroutine for wear with oxide growth, constant wear and friction coefficients

```
cccccc wear depth calculations

      do inode=1,num_nodes

         k_t(inode)=(3.888d-15)*dexp(-0.05739d0*cnfmag(inode))
         k_n(inode)=0.3d0*k_t(inode)

c     MPO.GTRF.P2.05-Completion-Memo & L3 report equation 1
c        k_t(inode)=(3.888E-6)*dexp(-0.05739*cnfmag(inode))
c        converting depth to m from mm, so the factor is d-15, not d-6

         ddepth(inode)=k_t(inode)*cslmag(inode)*(ctrmag(inode)*
     $      dabs(csdir(1,inode)*ctdir(1,inode)+
     $         csdir(2,inode)*ctdir(2,inode)+
     $         csdir(3,inode)*ctdir(3,inode))+
     $      (k_n(inode)*cnrmag(inode)*
     $      dabs(csdir(1,inode)*cndir(1,inode)+
     $         csdir(2,inode)*cndir(2,inode)+
     $         csdir(3,inode)*cndir(3,inode))))
         call oxide_depth(ddoxide,time_step)
         if (is_grid.eq.1) then
            dwear=gamma_value*ddepth(inode)
         else
            dwear=(1.d0-gamma_value)*ddepth(inode)
         end if
c depth = 1: for total wear of the clad and the grid
c         2: wear of the current clad/grid
c         3: wear rate of the clad and the grid
c         4: total wear volume of the clad and the grid
c         5: total oxide thickness of the clad and the grid
c         6: total oxide growth of the clad/grid (right assume to be
c            the same for clad and grid)
c         7: total oxide thickness of the clad/grid
c         8: total wear depth of the clad/grid
c
         depth(1,inode)=depth(1,inode)+ddepth(inode)
         depth(2,inode)=depth(2,inode)+dwear
         depth(3,inode)=ddepth(inode)/time_step
         depth(4,inode)=depth(4,inode)+carea(inode)*ddepth(inode)
         depth(5,inode)=depth(5,inode)+2.d0*ddoxide-ddepth(inode)
         if (depth(5,inode).lt.0.d0) depth(5,inode)=0.d0
         depth(6,inode)=depth(6,inode)+ddoxide
         depth(7,inode)=depth(7,inode)+ddoxide-dwear
         if (depth(7,inode).lt.0.d0) depth(7,inode)=0.d0
         if (ddoxide.lt.dwear) then
             dwear_zr=dwear
         else
             dwear_zr=ddoxide
         end if
         depth(8,inode)=depth(8,inode)+dwear_zr
c
      end do
      do inode=1,num_nodes
C      write(*,*)'node 150',nodeID(inode)
      if(nodeID(inode).eq.332) then
      write(*,*)'node number',nodeID(inode)
      write(*,*)'cs, time(2)',cs(inode),eval_time,time_step
      write(*,*)'csdir',csdir(1,inode),csdir(2,inode),csdir(3,inode)
      write(*,*)'cndir',cndir(1,inode),cndir(2,inode),cndir(3,inode)
      dwear=sqrt(cndir(1,inode)*cndir(1,inode)+cndir(2,inode)*cndir(2,
     $inode)+cndir(3,inode)*cndir(3,inode))
```

Table A. FORTRAN77 subroutine for wear with oxide growth, constant wear and friction coefficients

```
      write(*,*) 'magenetude of cndir',dwear
      write(*,*)'ctdir',ctdir(1,inode),ctdir(2,inode),ctdir(3,inode)
      dwear=sqrt(ctdir(1,inode)*ctdir(1,inode)+ctdir(2,inode)*ctdir(2,
     $inode)+ctdir(3,inode)*ctdir(3,inode))
      write(*,*) 'magenetude of ctdir',dwear
      dwear=(cndir(1,inode)*ctdir(1,inode)+cndir(2,inode)*ctdir(2,inode)
     $+cndir(3,inode)*ctdir(3,inode))
      write(*,*) 'cndir dot ctdir',dwear
      dwear=(csdir(1,inode)*ctdir(1,inode)+csdir(2,inode)*ctdir(2,inode)
     $+csdir(3,inode)*ctdir(3,inode))
      write(*,*) 'csdir dot ctdir',dwear
      dwear=(csdir(1,inode)*cndir(1,inode)+csdir(2,inode)*cndir(2,inode)
     $+csdir(3,inode)*cndir(3,inode))
      write(*,*) 'csdir dot cndir',dwear
      write(*,*)'cnfmag',cnfmag(inode)
      write(*,*)'cnrmag',cnrmag(inode)
      write(*,*)'ctrmag',ctrmag(inode)
      write(*,*)'cslmag',cslmag(inode)
      end if
      end do

cccccc wear depth calculations end

c     write depth to sierra analysis variable "depth" which can be
c     requested for output in the *.e file

      call aupst_put_node_var(num_nodes,num_depth_components,
     $   nodeID, depth, "depth", ierror2)

      call aupst_check_node_var(num_nodes,num_depth_components,
     $   nodeID, "depth", ierror3)

      call aupst_get_node_var(num_nodes,num_depth_components,
     $   nodeID, depth, "depth", ierror4)

      return
      end


c     subroutine for calculating the oxide growth ddoxide
      subroutine oxide_depth(ddoxide,time_step)

      implicit none
      double precision time_step, ddoxide,k

c     array of node number for nodes processed
c      integer nodeID(num_nodes)

c     current model is a linear growth model s=k*t
c     initial k is in micron/day
      k=0.01d0
c      convert k to m/s
      k=k*1.d-6/24.d0/3600.d0
      ddoxide=k*time_step
      return
      end
```