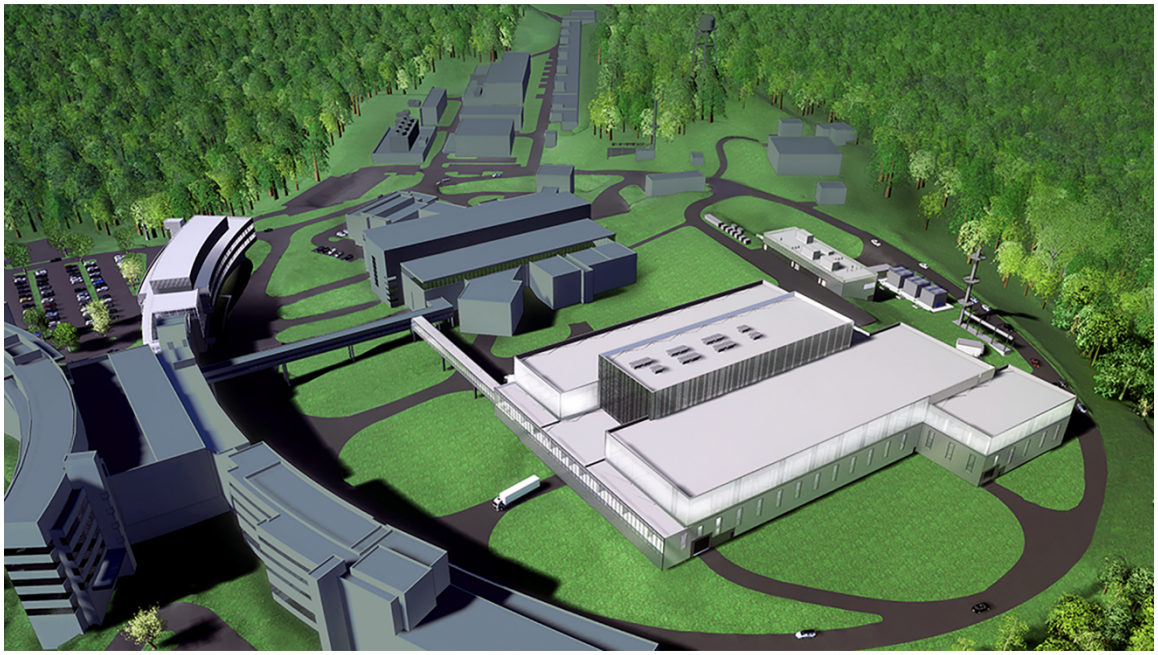
S03010000-TRT10000
ORNL/TM-2024/3432

Sierra/SolidMechanics for Target Pulse Analysis



Joseph B. Tipton, Jr.

October 2021

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

S03010000-TRT10000
ORNL/TM-2024/3432

Second Target Station (STS) Project

SIERRA/SOLIDMECHANICS FOR TARGET PULSE ANALYSIS

Joseph B. Tipton, Jr.

October 2021

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831
managed by
UT-BATTELLE LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

ABSTRACT

Sierra is a Multiphysics engineering simulation software suite developed and maintained by Sandia National Laboratories. Created for nuclear weapon needs, it is available to national laboratories with a foreign national restriction. The STS Target Systems analysis team is interested in using Sierra to perform explicit dynamics simulations of the tantalum clad tungsten target blocks as a response to sudden temperature changes induced by proton beam pulses. The capability already exists with Abaqus|Explicit, however, there are time and cost limitations due to licensing. The following document serves as a "quick-start" guide to the successful setup, simulation, and post-processing of a target model in Sierra on the OZ3 computing cluster at the Spallation Neutron Source.

1. Prerequisites

1.1 THINLINC[1]

Sierra resides on the OZ3 compute cluster at the Spallation Neutron Source (SNS). The Linux windows-like OS can be accessed remotely via a web browser using the ThinLinc client. Connect through ThinLinc through a web browser and note that you will need to proceed through security warnings. Once connected, login using your 3-character UID and password as shown in Figure 1. Figure 2 shows the main screen after login. Click "Activities" and then click the 3x3 dot icon to show all of the available applications. The "File", "Terminal", and "Text Editor" applications are the most useful. After a minute of inactivity, the desktop screen will lock. To unlock either (a) click on the blue screen and then press the spacebar key or (b) click on the blue screen and drag upward.

Figure 1: ThinLinc login screen

[1] This section was modified from an original written by Justin Mach. The OZ3 web address has been redacted.

Figure 2: OZ3 Linux GUI main screen. A list of all applications is shown by (1) clicking "Activities" followed by (2) clicking the 3x3 "Show Applications" button.

1.2 CUBIT

CUBIT™ is a licensed finite element meshing tool from Sandia National Laboratory. As of 1 October 2021, it is installed on the OZ3 cluster and can be opened via the **`cubit`** command in a terminal window. The installed version is:

```
$ which cubit
alias cubit='/data/Cubit-15.7/cubit &'
```

1.3 PARAVIEW

ParaView is an open-source visualization tool. As of 1 October 2021, it is installed on the OZ3 cluster and can be opened via the **`paraview`** command in a terminal window. The installed version is:

```
$ which paraview
alias paraview='MESA_GL_VERSION_OVERRIDE=3.3 /data/ParaView-5.9.1-MPI-
Linux-Python3.8-64bit/bin/paraview &'
```

1.4 SIERRA/SM

1.4.1 Installation

Sierra has already been installed on the OZ3 cluster but has a foreign national access limitation. For access, contact SNS linux support via email, and request to be added to the Sierra 'fem-users' group.

From a terminal window, type `gedit ~/.bash_profile &` and add the following lines of code:

```
ACCESS=/opt/seacas
export ACCESS
## Sierra environment variables
[ -f /data/fem/bash_environment ] && . /data/fem/bash_environment
```

Once saved, you will need to log out and log in again for the profile changes to take effect. To test access to Sierra, type `sierra --version`.

1.4.2 Documentation

The following documents are available for in-depth understanding of Sierra:

- Merewether, Mark Thomas, Plews, Julia A., de Frias, Gabriel Jose, Mosby, Matthew David, Porter, Vicki L., Shelton, Timothy, Thomas, Jesse David, Tupek, Michael R., Veilleux, Michael, Manktelow, Kevin, Beckwith, Frank, Belcourt, Kenneth Noel, Miller, Scott T, Treweek, Benjamin, Wagman, Ellen Brooke, & Koester, Jacob. *Sierra/SolidMechanics 4.56 User's Guide*.. United States. https://doi.org/10.2172/1608404

- Beckwith, Frank, Belcourt, Kenneth Noel, de Frias, Gabriel Jose, Koester, Jacob, Manktelow, Kevin, Merewether, Mark Thomas, Miller, Scott T, Mosby, Matthew David, Plews, Julia A., Porter, Vicki L., Shelton, Timothy, Thomas, Jesse David, Treweek, Benjamin, Tupek, Michael R., Veilleux, Michael, & Wagman, Ellen Brooke. *Sierra/SolidMechanics 4.56 Example Problems Manual*.. United States. https://doi.org/10.2172/1608083

- Veilleux, Michael, Beckwith, Frank, Belcourt, Kenneth Noel, de Frias, Gabriel Jose, Koester, Jacob, Manktelow, Kevin, Merewether, Mark Thomas, Miller, Scott T, Mosby, Matthew David, Plews, Julia A., Porter, Vicki L., Shelton, Timothy, Thomas, Jesse David, Treweek, Benjamin, Tupek, Michael R., & Wagman, Ellen Brooke. *Sierra/SolidMechanics 4.56 Verification Tests Manual*.. United States. https://doi.org/10.2172/1608403

- Beckwith, Frank, Belcourt, Kenneth Noel, de Frias, Gabriel Jose, Koester, Jacob, Manktelow, Kevin, Merewether, Mark Thomas, Miller, Scott T, Mosby, Matthew David, Plews, Julia A., Porter, Vicki L., Shelton, Timothy, Thomas, Jesse, Treweek, Benjamin, Tupek, Michael R., Veilleux, Michael, & Wagman, Ellen Brooke. *Sierra/SolidMechanics 4.56 Theory Manual*. United States. https://doi.org/10.2172/1608915

1.4.3 Scripting

Note that Sierra appears to require that all processors access a shared folder to run correctly. As a result, it must be run from the OZ3 headnode. The script file shown in Figure 3 should be used to run Sierra using the TORQUE job scheduler via the "qsub" command.

```
#!/bin/sh
#PBS -V
#PBS -j oe
#PBS -m abe
#PBS -l nodes=16:ppn=8         # specify # of nodes
#PBS -N Job_3_HIP_SS_Pulses    # specify the job name
#PBS -M UID@ornl.gov           # specify your email address
#PBS -q all                    # specify the queue: all / slow / fast
# #PBS -W depend=afterany:6923

cd $PBS_O_WORKDIR

# Define particulars of this run:
INPUT_FILENAME=Job_3_HIP_SS_Pulses.i

# Number of processors
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS processors

# Executable for sierra
EXEC=/data/fem/sierra_4.56.4b/install/sntools/engine/sierra #2020

echo $EXEC -j $NPROCS adagio -i $INPUT_FILENAME

# Run preprocessor to decompose mesh
$EXEC --pre -j $NPROCS adagio -i $INPUT_FILENAME

# Run solver
$EXEC --run -j $NPROCS adagio -i $INPUT_FILENAME

# Run postprocessor to rejoin decomposed results
$EXEC --post -j $NPROCS adagio -i $INPUT_FILENAME

sleep 60 # Give the log file a chance to complete

# clean up
rm -r *.e.*
rm -r *.g.*

exit
```

Figure 3: Script file to run Sierra using the TORQUE "qsub" command.

2. Discretization

2.1 MESH CREATION

Mesh creation can come from a variety of tools including CUBIT. This example uses a hex mesh created in Abaqus|CAE by Tom McManamy and consists of 3 parts: tantalum cladding, tungsten block surface elements, and tungsten block interior elements. The tungsten block surface and interior elements share common nodes at their boundary. (The tungsten block is split in this way to reduce postprocessing requirements for fatigue analysis.) The tantalum cladding and tungsten block surface elements have duplicate nodes at their boundary (and will be tied together in the analysis). Create a copy of the Abaqus input deck (*.inp), keep the **`*Part`** and **`*Assembly`** portions, and delete everything else.

2.2 MESH CONVERSION

CUBIT is used to import the Abaqus mesh and then save it in a format for Sierra.

- File > Import
- Abaqus (*.inp)
- Import Options: Free Mesh
- The file will load with the message "Parsing ABAQUS Input Deck"
- Figure 4 shows the mesh after import. Rename blocks and sets of interest as needed. Avoid minus "-" characters as these are invalid in Sierra. Delete any unnecessary node sets.
- File > Export
- Genesis (*.g)
- Export All

Figure 5 shows the resulting mesh and suggested block and set names.

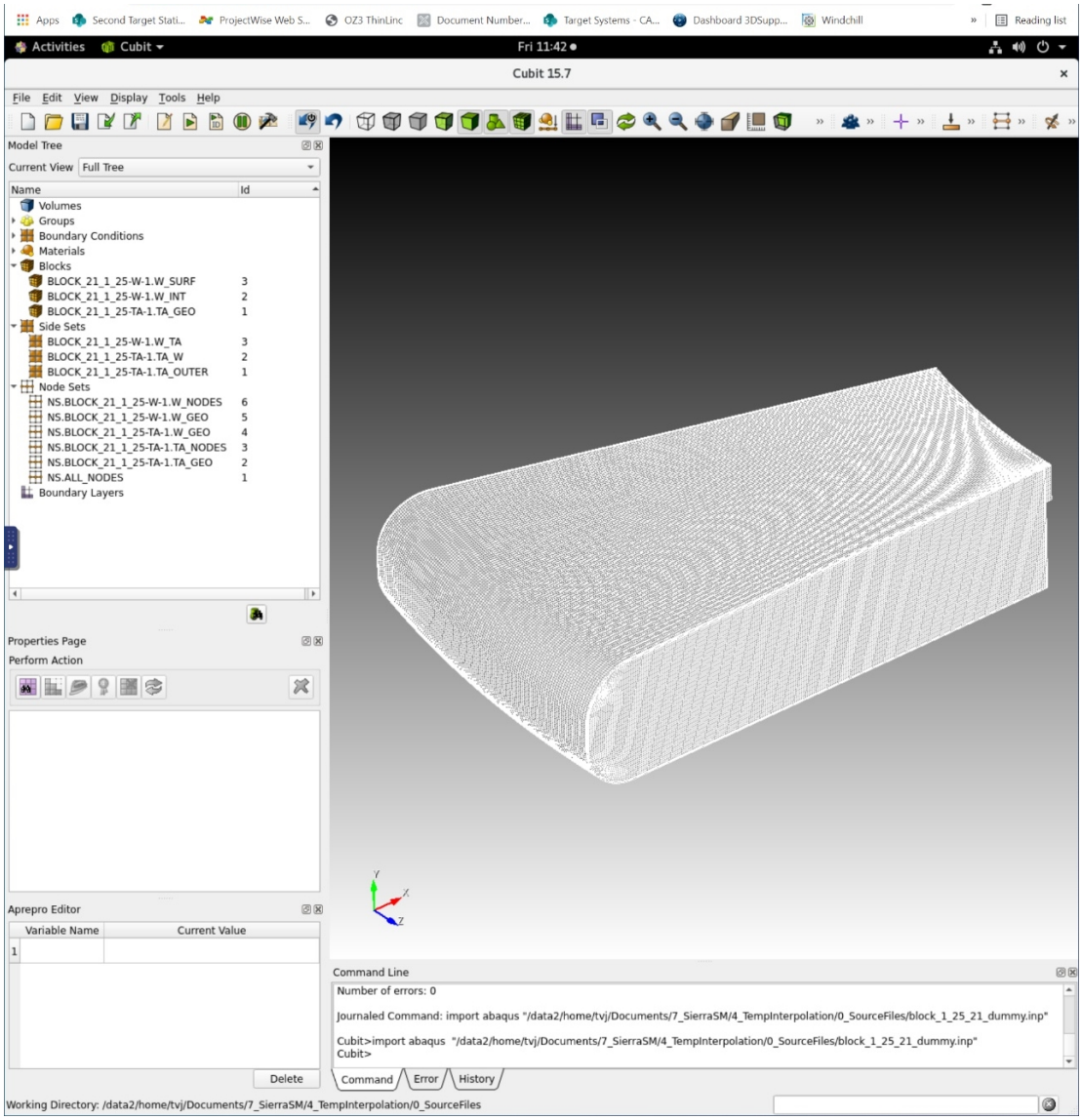


Figure 4: CUBIT screenshot showing successful import of Abaqus mesh.

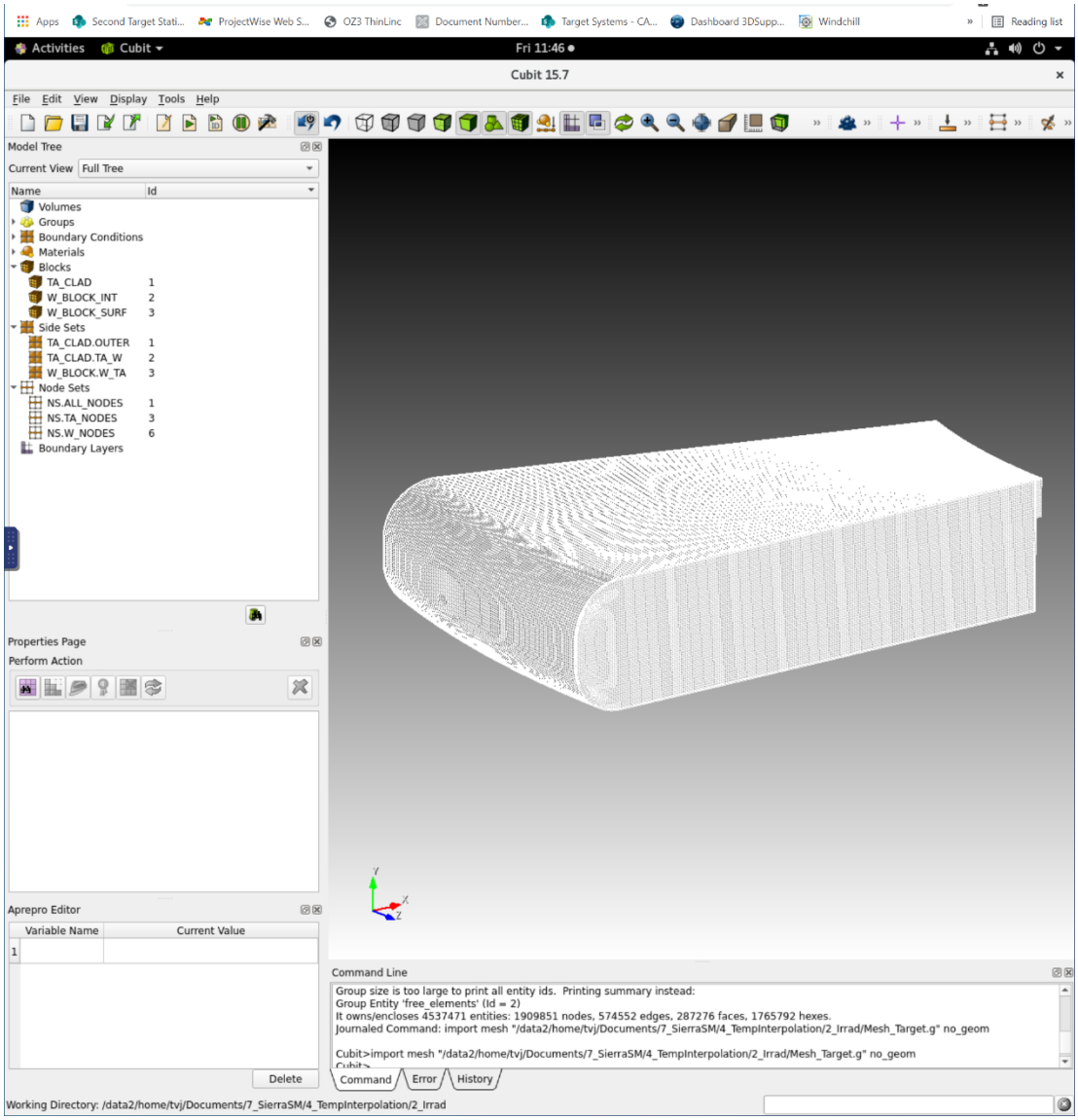


Figure 5: CUBIT screenshot showing the mesh after transformation into Sierra format including suggested block and set names.

3. Material Constitutive Models

The material constitutive model for tantalum is explored using a 1-element model. This section also serves to explain the general structure of a Sierra input file.

3.1 MODEL GEOMETRY

Figure 6 shows the single element mesh model created in Abaqus|CAE. This describes a 1 [mm] cube consisting of a C3D8R element with a single integration point. This mesh is converted into Sierra format using CUBIT.

```
*Heading
 Elastic-Plastic Test
** Job name: Job-1 Model name: Model-1
** Generated by: Abaqus/CAE 2020.HF4
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Part-1
*Node
      1, 0.00100000005, 0.00100000005, 0.00100000005
      2, 0.00100000005,           0., 0.00100000005
      3, 0.00100000005, 0.00100000005,           0.
      4, 0.00100000005,           0.,           0.
      5,           0., 0.00100000005, 0.00100000005
      6,           0.,           0., 0.00100000005
      7,           0., 0.00100000005,           0.
      8,           0.,           0.,           0.
*Element, type=C3D8R
1, 5, 6, 8, 7, 1, 2, 4, 3
*Nset, nset=Set-All, generate
 1,  8,  1
*Elset, elset=Set-All
 1,
** Section: Section-1
*Solid Section, elset=Set-All, material=Ta-JC-Table-Rate
,
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Part-1-1, part=Part-1
*End Instance
**
*Nset, nset=BC-4, instance=Part-1-1, generate
 1,  4,  1
*Nset, nset=Node-1, instance=Part-1-1
 8,
*Nset, nset=Node-2, instance=Part-1-1
 6,
*Nset, nset=Node-3, instance=Part-1-1
 5, 7
*End Assembly
```

Figure 6: Abaqus single element mesh.

3.2 MONOTONIC PULL TEST

Figure 7 through Figure 11 list the different sequential portions of the Sierra input file used to simulate a monotonic pull test of tantalum.

Figure 7 shows the header “begin sierra” and contains a listing of mathematical functions to be used in the analysis. This is similar to the Abaqus amplitude listings.

```
begin sierra

  ##
  ##-------------------------------------------------------------------
  ## FUNCTIONS
  ##-------------------------------------------------------------------
  ##

  begin definition for function TempFunc
    type is piecewise linear
    begin values
      0.0     200.0
      0.01     25.0
    end values
  end

  begin definition for function NodePull
    type is piecewise linear
    begin values
      0.0       0.0
      0.01      2.0e-06
    end values
  end
```

Figure 7: Sierra input deck (1 of 5) defining mathematical functions.

Figure 8 defines the material models. The function “CTE_Ta” defines the coefficient of thermal expansion as an analytical function. The initial temperature of the simulation will be the reference temperature, and the expression will be differentiated automatically to produce the correct CTE coefficient. The function “Yield_Ta” is an analytical expression of the normalized yield stress as a function of temperature. This is taken from a published Johnson-Cook constitutive model for tantalum [1]. The function “Hard25” is an analytical expression of the yield stress (Pa) as a function of plastic strain at 25°C. This also comes from the published Johnson-Cook constitutive model evaluated at room temperature (25°C) and at the reference rate of plastic strain (0.001/s).

The “Ta_Bilinear” material model defines the tantalum as a simple elastic-plastic material. A kinematic hardening model is selected via “BETA = 0.0”.

The “Ta_ThermoElasticPlastic” material model defines the tantalum as a thermoelastic-plastic material. This incorporates a temperature effect on the yield behavior. First, the YIELD STRESS is multiplied by the YIELD STRESS FUNCTION. Second, the list of HARDENING FUNCTIONS corresponds to the listed TEMPERATURES. Only one temperature 25°C is listed here due to a software bug in Sierra that does not correctly interpolate hardening between temperatures. This bug was identified during model testing and was reported to sierra-help@sandia.gov on 8/20/2021 with a help ticket ID of COMPSIMHD-11562.

```
##------------------------------------------------------------------------
## MATERIALS - Tantalum
##------------------------------------------------------------------------
begin function CTE_Ta
  type is analytic
  # Initial Temp is the Reference Temp
  evaluate expression is "6.3e-06 * x"
end

begin function Yield_Ta
  type is analytic
  evaluate expression is "1.0 - ((x - 25.0)/(2976.85 - 25.0))^0.4"
end

begin function Hard25
  type is analytic
  evaluate expression is "   \#
    (1.0) *                  \#
    0.6 *                    \#
    1.0e+6 *                 \#
    (340.0 + 750.0*x^0.7)    \#
    "
end

begin property specification for material Ta_Bilinear
  density = 16600.
  THERMAL LOG STRAIN FUNCTION = CTE_Ta
  begin parameters for model ELASTIC_PLASTIC
    youngs modulus = 1.88e+11
    poissons ratio = 0.35
    #
    #Hardening Behavior
    #
    YIELD STRESS = 2.0e+8
    BETA = 0.0
    HARDENING MODULUS = 1.0e+9
  end
end

begin property specification for material Ta_ThermoElasticPlastic
  density = 16600.
  THERMAL LOG STRAIN FUNCTION = CTE_Ta
  begin parameters for model THERMOELASTIC_PLASTIC
    youngs modulus = 1.88e+11
    poissons ratio = 0.35
    #
    # Hardening Behavior
    #
    BETA = 0.0
    YIELD STRESS = 204.0E+6
    YIELD STRESS FUNCTION = Yield_Ta
    TEMPERATURES = 25.0
    HARDENING FUNCTIONS = Hard25
  end
end
```

Figure 8: Sierra input deck (2 of 5) defining material models.

Figure 9 defines the mesh. It identifies the mesh file "SingleElem.g" created from the CUBIT export. For the mesh file, each block name must be assigned a material definition and material model. In this case there is only one block, named "BLOCK1" and it is assigned as thermoelastic_plastic tantalum.

```
##------------------------------------------------------------------
## MESH DEFINITION
##------------------------------------------------------------------

begin finite element model BlockMesh
  database name = SingleElem.g
  database type = exodusII

  begin parameters for block BLOCK1
    material Ta_ThermoElasticPlastic
    solid mechanics use model THERMOELASTIC_PLASTIC
  end

end
```

Figure 9: Sierra input deck (3 of 5) defining mesh and assigning material models.

Figure 10 gives the solution control settings and output settings. "Presto" identifies the explicit dynamics solver in Sierra. The procedure is given a descriptive name of "Matl_Test". The "time control" section defines a start time at 0.0 [s] and an end time at 0.01 [s] for the "TargetBlock" region. The "Step Interval" simply defines the frequency to write energy history data to the log file and does not change the solution.

The "TargetBlock" region is defined next. First, the mesh name "BlockMesh" references the same name used in Figure 9. Next, two different output files are listed. The "history output" file writes solution information at 0.1 ms time increments to a binary file in ExodusII format that can be read with ParaView. The "heartbeat output" file writes solution information at 0.1 ms time increments to an ASCII text file. This file can be monitored in real time during longer simulations.

```
##-------------------------------------------------------------------------
## ANALYSIS SETTINGS
##-------------------------------------------------------------------------

begin presto procedure Matl_Test

  begin time control
    begin time stepping block
      start time = 0.0
      begin parameters for presto region TargetBlock
        Step Interval = 1000
      end
    end
    termination time = 0.01
  end

  begin presto region TargetBlock
    use finite element model BlockMesh

    begin history output
      database name = %B.e
      at time 0.0 increment = 0.0001
      global timestep as DT
      global internal_energy as IE
      global kinetic_energy as KE
      global strain_energy as SE
      global hourglass_energy as HGE
      global external_energy as ExtE
      global contact_energy as ContE
      global time as TIME
      node displacement at node 1 as Disp
      node temperature at node 1 as Temp
      element eqps at element 1 as Eqps
      element stress at element 1 as Stress
      element yield_stress at element 1 as Ystress
      element yield_flag at element 1 as Yflag
      element von_mises at element 1 as Mises
      element log_strain at element 1 as LogStrain
      element effective_log_strain at element 1 as EffLogStrain
    end

    begin heartbeat output
      stream name = %B.hrt
      append = OFF
      format = CSV
      labels = OFF
      legend = ON
      precision = 6
      timestamp format = "[%H:%M:%S],"
      at time 0.0       increment = 1.0e-04
      global time as TIME
      node displacement at node 1 as Disp
      node temperature at node 1 as Temp
      element eqps at element 1 as Eqps
      element stress at element 1 as Stress
      element yield_stress at element 1 as Ystress
      element yield_flag at element 1 as Yflag
      element von_mises at element 1 as Mises
      element log_strain at element 1 as LogStrain
      element effective_log_strain at element 1 as EffLogStrain
    end
```

Figure 10: Sierra input deck (4 of 5) defining solution controls and output requests.

Figure 11 is the final portion of the Sierra input file and defines loads and boundary conditions. First, all blocks are assigned a time varying temperature via the "TempFunc" function defined at the beginning of the file. Second, 3 nodes are assigned simple fixed displacement boundary conditions to make the block minimally constrained on one side. Third, the other side of the element is given a time varying displacement via the "NodePull" function defined at the beginning of the file.

The last 3 lines close the "presto" control definitions and terminate the Sierra file.

```
      begin prescribed temperature
        include all blocks
        function = TempFunc
      end

      begin fixed displacement
        node set = NS.NODE_1
        components = X Y Z
      end fixed displacement

      begin fixed displacement
        node set = NS.NODE_2
        components = X Y
      end fixed displacement

      begin fixed displacement
        node set = NS.NODE_3
        components = X
      end fixed displacement

      begin prescribed displacement
        node set = NS.BC_4
        component = X
        function = NodePull
      end prescribed displacement

    end presto region TargetBlock

  end presto procedure Matl_Test

end sierra
```

Figure 11: Sierra input deck (5 of 5) defining loads and boundary conditions and ending the input file.

This simulation can be run with 1 compute node and 1 processor using the TORQUE script given in Figure 3.

Figure 12 shows the results compared against an equivalent simulation in Abaqus. Note that the simulation produces a 1 μm pull over 10 ms while the temperature linearly decreases from 200°C to 25°C. The tantalum expands elastically and then yields. Once it has yielded, the material follows the yield strength curve as the material continues to cool. Eventually, the temperature lowers to the point that the material returns to elastic deformation. At this point the two simulations show a slight deviation in stress. One possible explanation is that Sierra uses an analytical expression for yield stress as a function of temperature while Abaqus performs a linear interpolation.

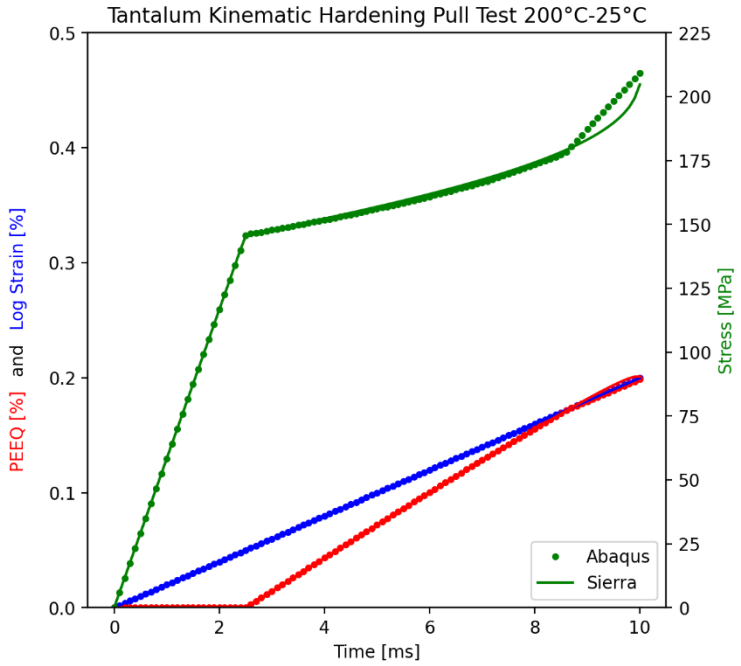


Figure 12: Monotonic pull test simulation comparison between Abaqus and Sierra.

3.3 STRAIN CONTROLLED CYCLIC TEST

The monotonic pull test simulation file can be easily modified to perform a strain controlled cyclic test. Figure 13 shows how the "NodePull" function is changed to represent 20 Hz cycling at progressively larger strains. In addition to this modification, the "time control" settings are changed to have a 0.5 s termination time. Finally, the history and heartbeat output settings are changed to have a 1 ms time increment.

Figure 14 and Figure 15 show the results. In this case, experimental data is available from Hellebrant and Stephens [2], however, this is for extremely large-grained tantalum (~ 500 µm). The curve shows a small amount of kinematic hardening, as expected.

```
##
##-------------------------------------------------------------------
## FUNCTIONS
##-------------------------------------------------------------------
##

begin definition for function TempFunc
  type is piecewise linear
  begin values
    0.0     25.0
    0.5     25.0
  end values
end

begin definition for function NodePull
  type is piecewise linear
  begin values
    0.00000E+00    0.0
    1.66667E-02    3.00E-07
    3.33333E-02   -3.00E-07
    5.00000E-02    0.0
    6.66667E-02    5.00E-07
    8.33333E-02   -5.00E-07
    1.00000E-01    0.0
    1.16667E-01    1.00E-06
    1.33333E-01   -1.00E-06
    1.50000E-01     0.0
    1.66667E-01     1.45E-06
    1.83333E-01   -1.45E-06
    2.00000E-01    0.0
    2.16667E-01    2.00E-06
    2.33333E-01   -2.00E-06
    2.50000E-01    0.0
    2.66667E-01    2.50E-06
    2.83333E-01   -2.50E-06
    3.00000E-01    0.0
    3.16667E-01    3.00E-06
    3.33333E-01   -3.00E-06
    3.50000E-01    0.0
    3.66667E-01    4.00E-06
    3.83333E-01   -4.00E-06
    4.00000E-01    0.0
    4.16667E-01    5.00E-06
    4.33333E-01   -5.00E-06
    4.50000E-01    0.0
    4.66667E-01    6.00E-06
    4.83333E-01   -6.00E-06
    5.00000E-01    0.0
  end values
end
```

Figure 13: Sierra input deck modification to describe a 20 Hz strain controlled cyclic test at progressively larger strains.

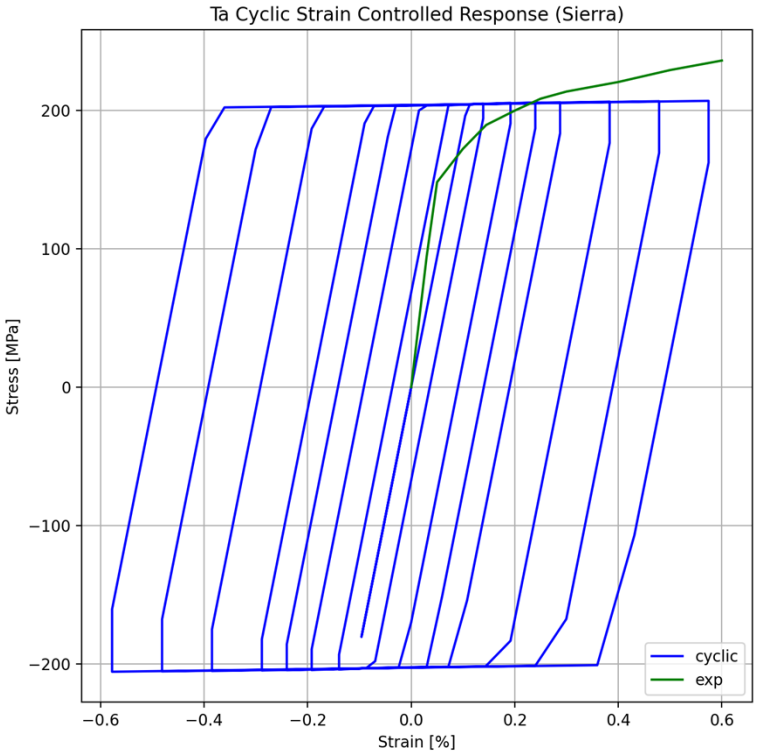


Figure 14: Tantalum cyclic strain controlled response. Experimental curve from [2].

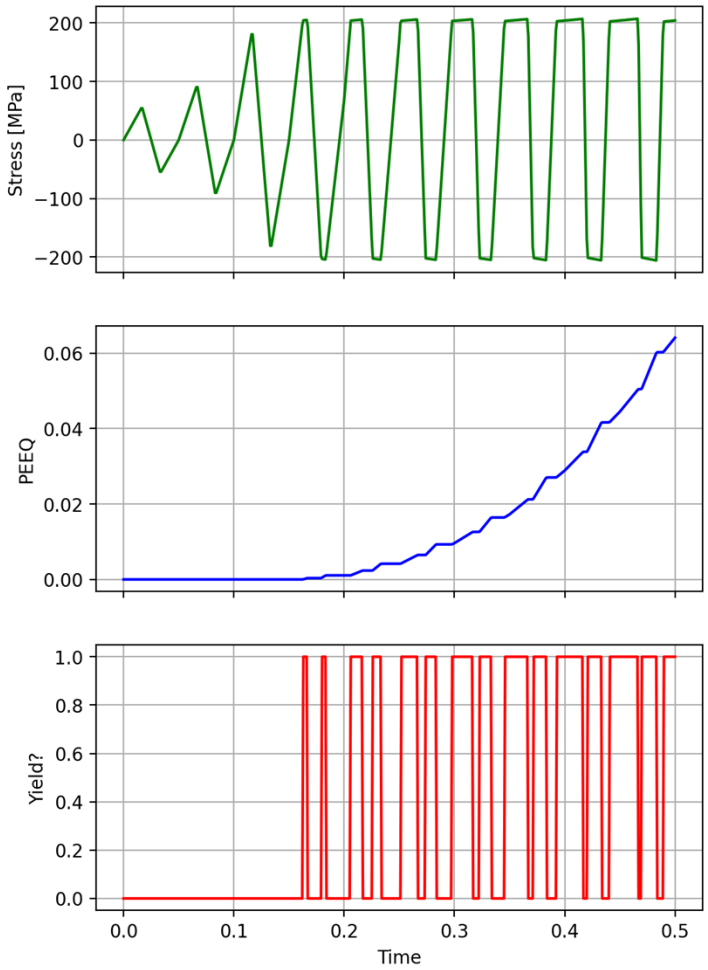


Figure 15: Tantalum cyclic strain controlled response as a function of time.

4. TEMPERATURE INTERPOLATION

4.1 SOURCE DATA FOR INTERPOLATION

This document uses the following configuration for target block analysis:

- Straight-through block geometry
- tantalum rear shelf
- unstructured mesh neutronics
- nominal beam placed between segments
- coordinate system prealigned to geometry by T. McManamy

The neutronics pulse heating files were sourced from the STS SharePoint site at **S.03.02 Target Assembly > heating_numbers > o28b_straight_through_210125MT**. The quasi-steady-state temperature distribution for unirradiated conditions was sourced from **S.03.02 Target Assembly > CFD_Temperature_Files > 2021_03_02_Temperature_straight-through_model_constant_conductivity**. The quasi-steady-state temperature distribution for irradiated conditions was sourced from **S.03.02 Target Assembly > CFD_Temperature_Files > 2021_03_11_Temperature_straight-through_model_dpa-dependent_conductivity**.

4.2 INTERPOLATION PROGRAM

The temperature interpolation program was composed in Python and was published online [3]. The "fitData.py" file is a more general Python module that requires the Numpy, PyTetGen, and SKLearn modules (already available on the OZ3 cluster). It uses Delaunay triangularization to mesh a source point cloud. For a given set of target nodal coordinates, it indexes which node falls inside which tetrahedron and then interpolates the target value using Barycentric (areal) weighting. If any target nodes fall outside the source temperature point cloud, it uses the K-Nearest Neighbors algorithm to find the 3 nearest target values and averages using a distance weighting.

The "fitTemps.py" file is specific to this analysis using Sierra. It requires the SEACAS Exodus Python module (already available on the OZ3 cluster). It uses the "fitData" module to interpolate temperature point clouds onto a mesh in ExodusII format and then saves the temperature fields to a new ExodusII file as a function of time. Figure 16 through Figure 18 break the file into three divisions.

Figure 16 loads the mesh of the target block and cladding. The arrays "targetID", "targetIndex", and "targetCoord" contain the IDs, indices, and special coordinates for all nodes in the mesh (that serve as the *target* for interpolation). The user must interrogate the "mesh.get_node_set_ids" and "mesh.get_node_set_names" to determine which node set ID numbers identify the tantalum nodes and tungsten nodes respectively. These IDs are then used to create the tantalum and tungsten target sets.

```
from fitData import *
from exodus import exodus, copyTransfer

#=====================================================================
# FIT POINT CLOUD TEMPERATURE SETS TO SIERRA MESH
#=====================================================================

#
# Load Target Mesh
#
mesh = exodus('Mesh_Target.g',mode='r',array_type='numpy')

targetID = np.array(mesh.get_node_id_map())
targetIndex = np.arange(1,len(targetID)+1)
targetCoord = np.transpose(mesh.get_coords())

mesh.get_elem_blk_names()
mesh.get_node_set_ids()
mesh.get_node_set_names()

mesh.get_node_set_name(3)
tantalumIndex = mesh.get_node_set_nodes(3)
tantalumID = tantalumIndex - 1
tantalumCoord = targetCoord[tantalumID]

mesh.get_node_set_name(6)
tungstenIndex = mesh.get_node_set_nodes(6)
tungstenID = tungstenIndex - 1
tungstenCoord = targetCoord[tungstenID]

mesh.close()
```

Figure 16: "fitTemps.py" Python program (Part 1 of 3) where the target mesh is loaded.

Figure 17 gives the second portion of the program. The *source* neutronics pulse temperature rise and steady-state temperature fields are loaded. Then the "fitData" module is invoked to interpolate the *source* data onto the *target* mesh. This is performed separately on the tantalum and tungsten nodes. Finally, the interpolated temperatures on the tantalum and tungsten nodes are combined into 3 data sets: "finalPulseA" is the temperature rise from a pulse on one corner; "finalPulseB" is the temperature rise from a pulse on the other corner; "finalSS" is the steady-state temperature.

```
#
# Load Source Mesh and Temperatures and Fit the Data
#

source = np.loadtxt(open('DTTa','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
TaPulseA= fitData(sourceCoord,sourceTemp,tantalumID,tantalumIndex,tantalumCoord)

source = np.loadtxt(open('DTTa121','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
TaPulseB = fitData(sourceCoord,sourceTemp,tantalumID,tantalumIndex,tantalumCoord)

source = np.loadtxt(open('Tantalum_Temperature_deg_C.csv','rb'), delimiter=',',
skiprows=1)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
TaSS = fitData(sourceCoord,sourceTemp,tantalumID,tantalumIndex,tantalumCoord)

source = np.loadtxt(open('DTW','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
WPulseA= fitData(sourceCoord,sourceTemp,tungstenID,tungstenIndex,tungstenCoord)

source = np.loadtxt(open('DTW121','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
WPulseB = fitData(sourceCoord,sourceTemp,tungstenID,tungstenIndex,tungstenCoord)

source = np.loadtxt(open('Tungsten_Temperature_deg_C.csv','rb'),
                    delimiter=',', skiprows=1)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
WSS = fitData(sourceCoord,sourceTemp,tungstenID,tungstenIndex,tungstenCoord)

finalPulseA = np.hstack((TaPulseA,WPulseA))
finalPulseB = np.hstack((TaPulseB,WPulseB))
finalSS = np.hstack((TaSS,WSS))
```

Figure 17: "fitTemps.py" Python program (Part 2 of 3) where the source temperature fields are interpolated onto the target mesh.

Figure 18 gives the third and final portion of the program. The input mesh "Mesh_Target.g" is copied to a new mesh as "Temps_Target.g" and a new nodal variable named "NodalTempField" is added. Seven different nodal temperature field results are added to the mesh file with corresponding times. These times match the simulation times used between steps in the analysis as listed in Table 1.

```
#
# Save Temperature/Time Fields to New Exodus File
#
addGlobalVariables = []
addNodeVariables = ["NodalTempField"]
addElementVariables = []
# copy exodus file and add variables to all nodes
exo = copyTransfer('Mesh_Target.g','Temps_Target.g','ctype',
                   addGlobalVariables,addNodeVariables, addElementVariables)

times = np.array([0.0,
                  0.007,
                  0.01,
                  0.0100007,
                  0.0140007,
                  0.0140014,
                  0.0180014])

temps = np.zeros([len(targetID),len(times)])

temps[:,0] = 450.0
temps[:,1] = 30.0
temps[:,2] = finalSS['Temp']
temps[:,3] = finalSS['Temp'] + finalPulseA['Temp']
temps[:,4] = finalSS['Temp'] + finalPulseA['Temp']
temps[:,5] = finalSS['Temp'] + finalPulseB['Temp']
temps[:,6] = finalSS['Temp'] + finalPulseB['Temp']

for ii in range(len(times)):
  exo.put_node_variable_values('NodalTempField',ii+1,temps[:,ii])
  exo.put_time(ii+1,float(times[ii]))

exo.close()
```

Figure 18: "fitTemps.py" Python program (Part 3 of 3) where the interpolated temperatures are saved with time stamps onto the mesh for use in the full Sierra simulation.

Table 1: Sierra simulation steps.

Start Time	End Time	Event
0.0	0.007	HIP
0.007	0.01	Warm-up to Steady-State
0.01	0.0100007	Beam Pulse on Corner
0.0100007	0.0140007	Stress/Strain Response
0.0140007	0.0140014	Beam Pulse on Other Corner
0.0140014	0.0180014	Stress/Strain Response

4.3 INTERPOLATION RESULTS

Figure 19 through Figure 22 show the results of the temperature interpolation.

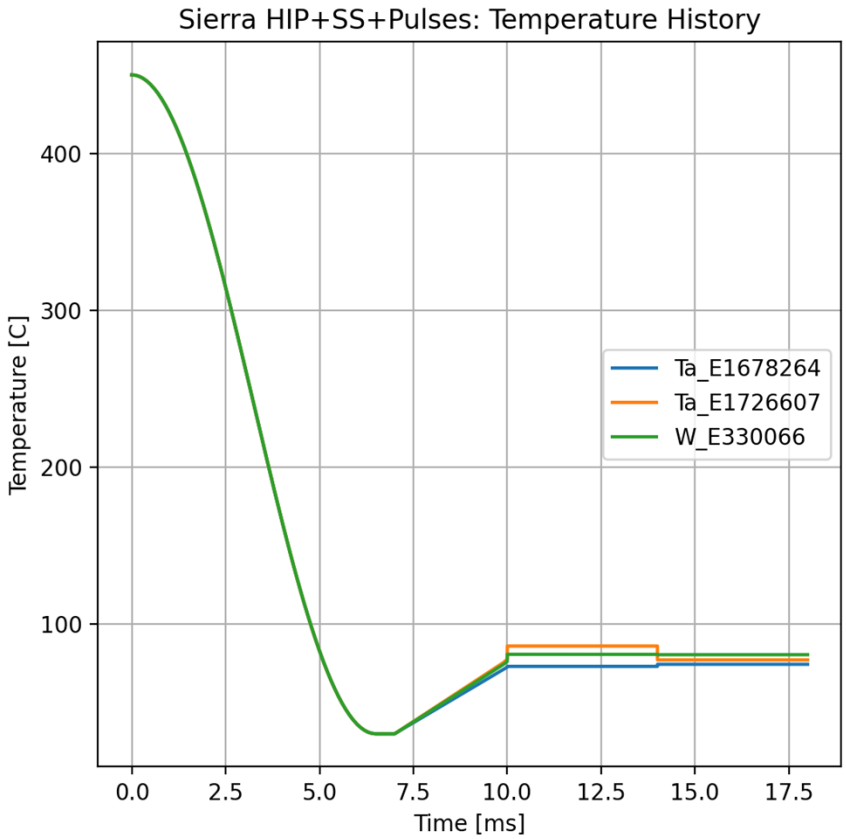


Figure 19: Temperature of 3 elements as a function of time as given in Table 1, for the unirradiated case.

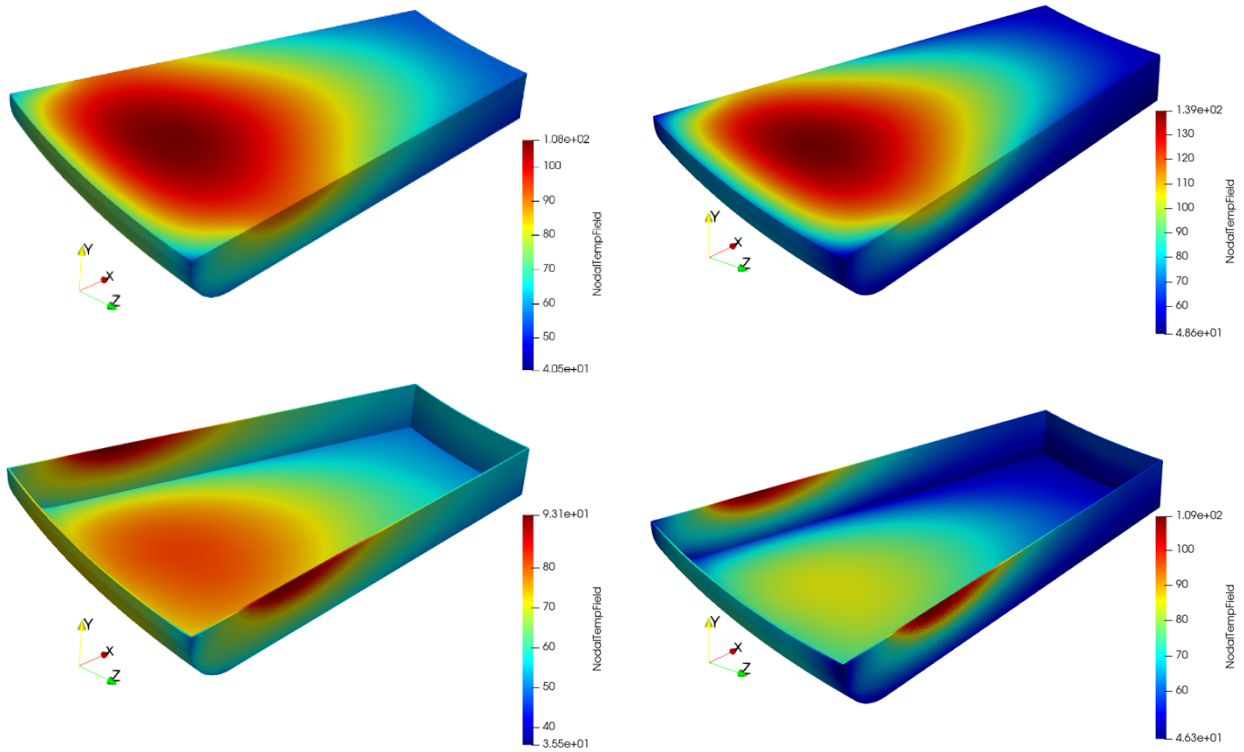


Figure 20: Quasi-steady state temperature distribution [°C]. Unirradiated condition (L) and irradiated condition (R).

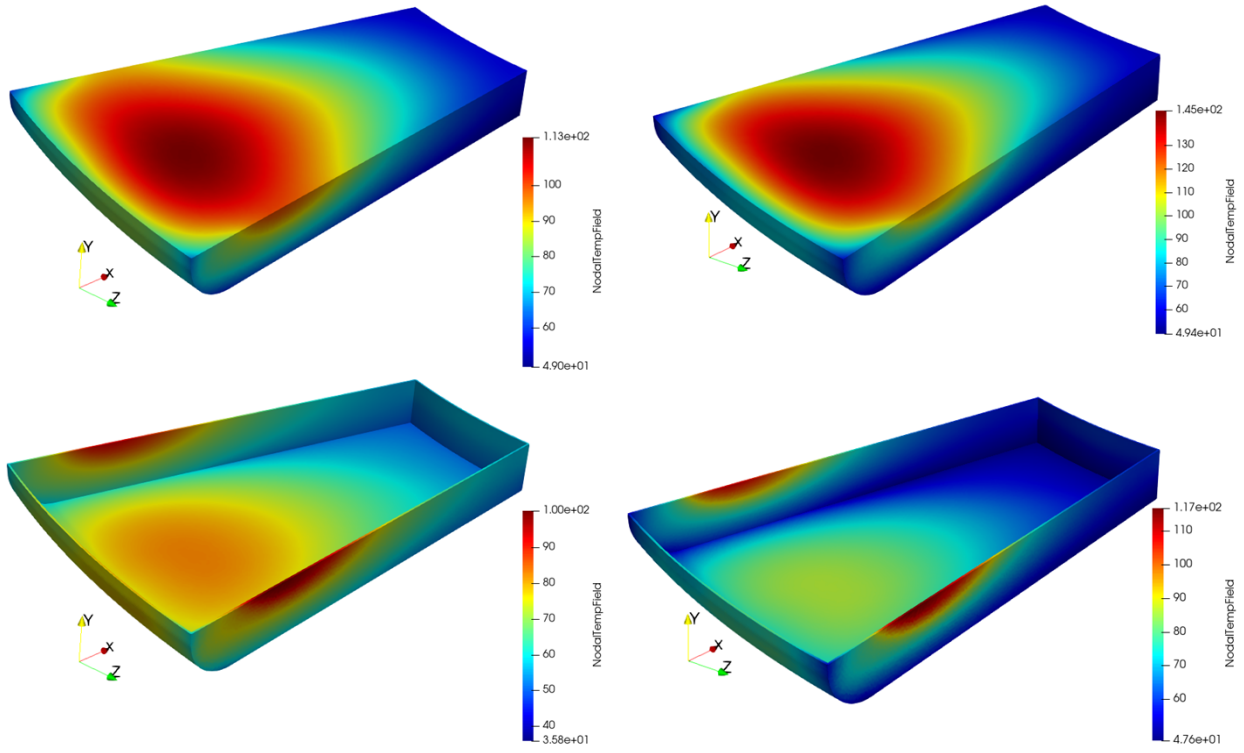


Figure 21: Quasi-steady state + Pulse A temperature distribution [°C]. Unirradiated condition (L) and irradiated condition (R).

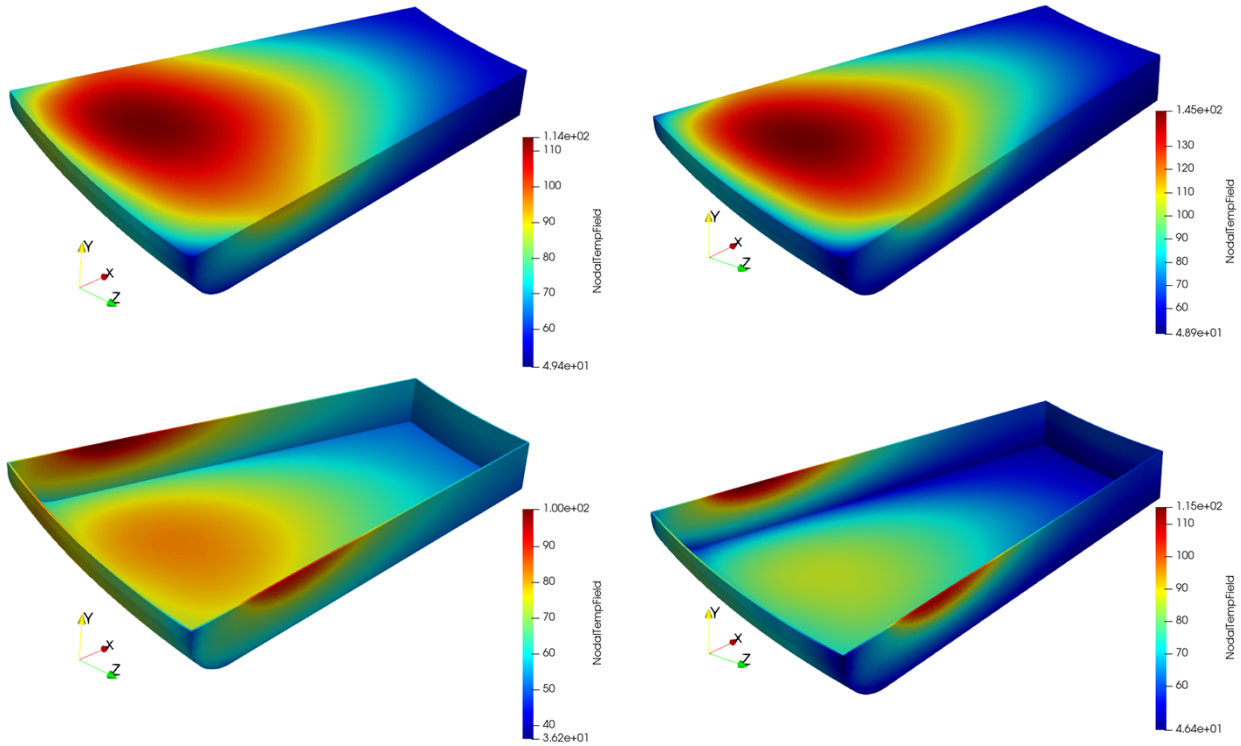


Figure 22: Quasi-steady State + Pulse B temperature distribution [°C]. Unirradiated condition (L) and irradiated condition (R).

5. SIERRA SIMULATION

5.1 INPUT FILE

5.1.1 Unirradiated Condition

Appendix 8.4 lists the Sierra input file for the full analysis as defined in Table 1. The structure of the input file follows the explanations given in Section 3.2 with three exceptions. First, the HIP and steady-state warm-up steps are run in quasistatic mode [4]. This is achieved with the command "number of quasistatic time steps". In general, the number of steps entered should yield a timestep that is an order of magnitude larger than the stable explicit timestep. The duration of the total time and timestep were determined through iterative testing while observing energy history along with stress-strain history of a tantalum element.

Second, the "prescribed temperature" fields were used to load different time-temperature functions. Figure 23 lists the relevant code. For the HIP step, the "SmoothTempHIP" analytical function uses a cosine ramp function to bring the target from a uniform 450°C to 30°C in a way that helps with quasistatic solution convergence. The other analysis steps then use the "TargetBlockTemps" nodal temperature fields from the interpolation performed in Section 4.

```
begin prescribed temperature
  include all blocks
  active periods = Step_1_HIP
  function = SmoothTempHIP
end

begin prescribed temperature
  include all blocks
  active periods = Step_2_SS Step_3_Pulses
  copy variable = NodalTempField from model TargetBlockTemps MAP_BY_ID
end prescribed temperature
```

Figure 23: Sierra commands to define temperature changes with time.

Third, the tungsten block nodes are tied to the tantalum clad nodes as shown in Figure 24 using tied multi-point constraint (MPC). The Sierra User's Manual states that "[t]his type of MPC is equivalent to using pure master/slave tied contact and is significantly faster than standard tied contact for explicit dynamics analyses" [5]. "W_BLOCK.W_TA" and "TA_CLAD.TA_W" define the sidesets (or surfaces) named in the mesh discretization for the contact surfaces. To interrogate contact status, the nodal variable "Status_Constraint_Flag" can be requested as output for visualization.

```
begin tied mpc

  tied faces = W_BLOCK.W_TA

  tied nodes = TA_CLAD.TA_W

end tied mpc
```

Figure 24: Sierra commands to define temperature changes with time.

5.1.2 Irradiated Condition

The input file for the irradiated condition is also provided in Appendix 8.4 and includes minor modifications from the unirradiated condition. First, the mesh definition section references the irradiated temperature file. Second, the "Step_1_HIP" time stepping block is removed from the time control. Third, the initial condition block is changed to a magnitude of 30°C. Fourth, the prescribed temperature block for "Step_1_HIP" is removed.

The combined effect of these changes will force the simulation to start at 0.007 seconds which corresponds to the uniform 30°C temperature state. The block will be unstressed at this temperature which simulates the complete relaxation of residual stresses from HIP. The simulation then proceeds using temperature fields from tungsten with reduced conductivity from irradiation.

5.2 PERFORMANCE SCALING ON OZ3

Figure 25 compares solution times for Sierra versus Abaqus. The OZ3 "fast" queue is used preferentially. The HIP and stead-state warm-up steps of the target simulation were used with roughly equivalent information written as output. Figure 26 explores Sierra scaling performance for the full target simulation. It appears that Abaqus is faster in all cases, but Sierra scales more strongly. One key difference in execution of the two software programs is that Abaqus is able to run from the compute node (and therefore take advantage of the faster local file system) while Sierra must run from the headnode (to use a file system that is slower but shared between all nodes). Running Sierra from the headnode also introduces a confounding effect as the headnode load varies as it serves other users. For example, Figure 25 shows a difference in execution time for Sierra with and without file output while Figure 26 does not. Thus, this scaling performance study is very specific to the OZ3 hardware configuration. Finally, it appears that 16 compute nodes is optimal for this model.

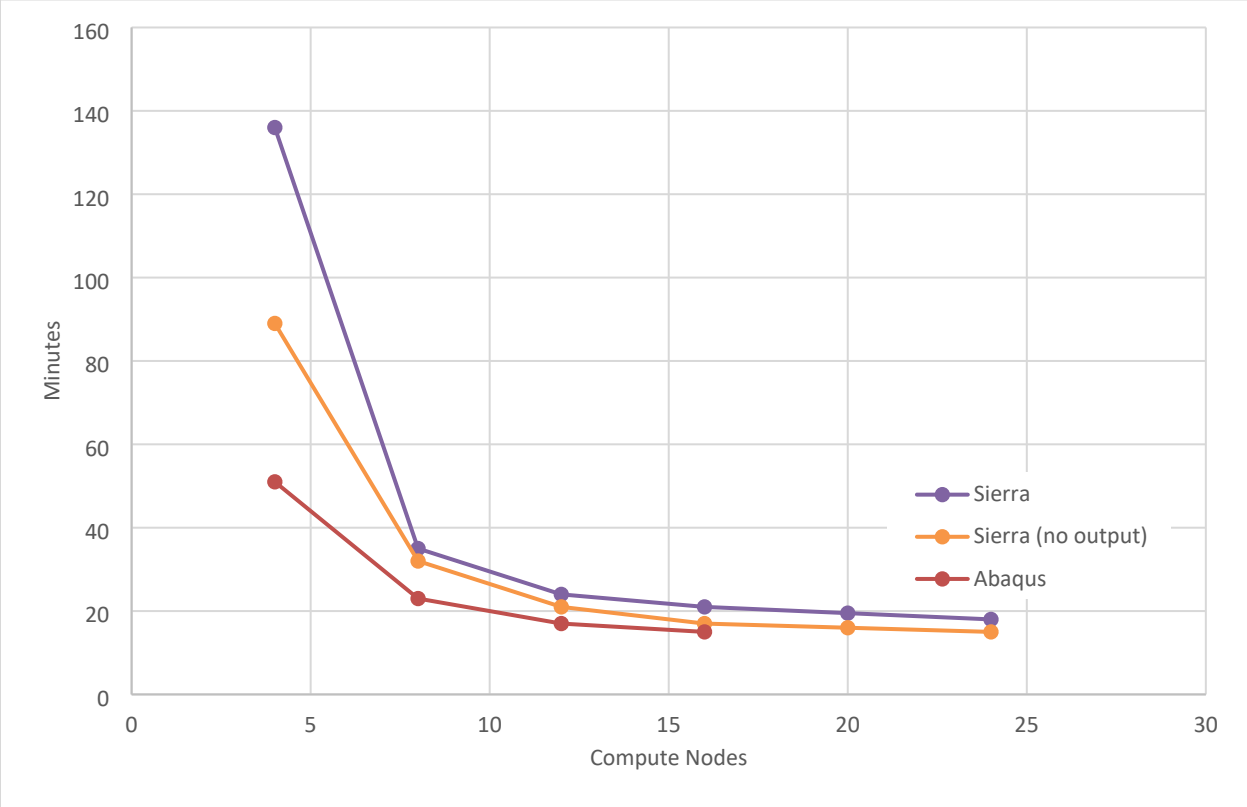


Figure 25: Scaling performance on the OZ3 cluster using the HIP and steady-state warm-up steps of the target simulation.

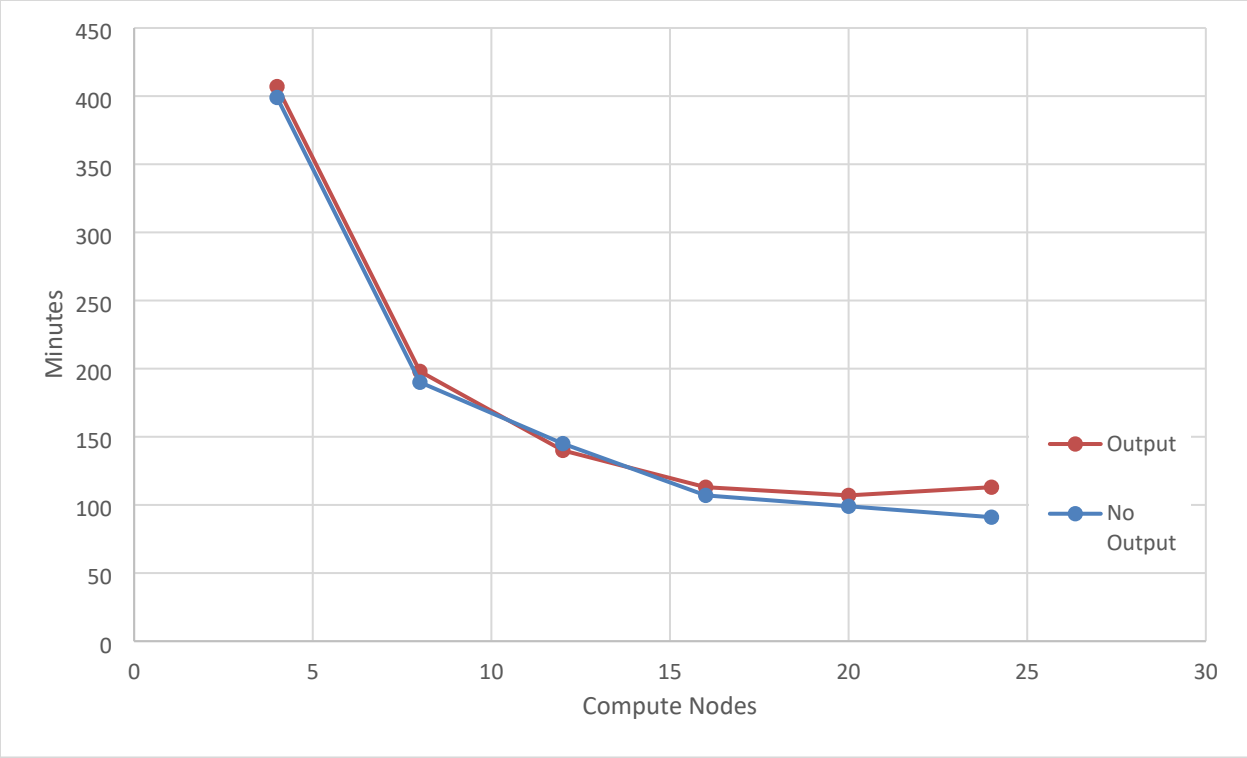


Figure 26: Scaling performance on the OZ3 cluster using the full target simulation.

5.3 RESULTS

Figure 27 through Figure 37 give results of the Sierra simulation (unirradiated condition) along with comparisons against an equivalent analysis in Abaqus. Overall, comparisons show excellent agreement. It should be noted that stress contour plots in ParaView show the integration point data while stress contour plots in Abaqus show an extrapolation of integration point data to the surface. Appendix 8.5 gives the Python scripts used to generate the plots.

5.3.1 HIP

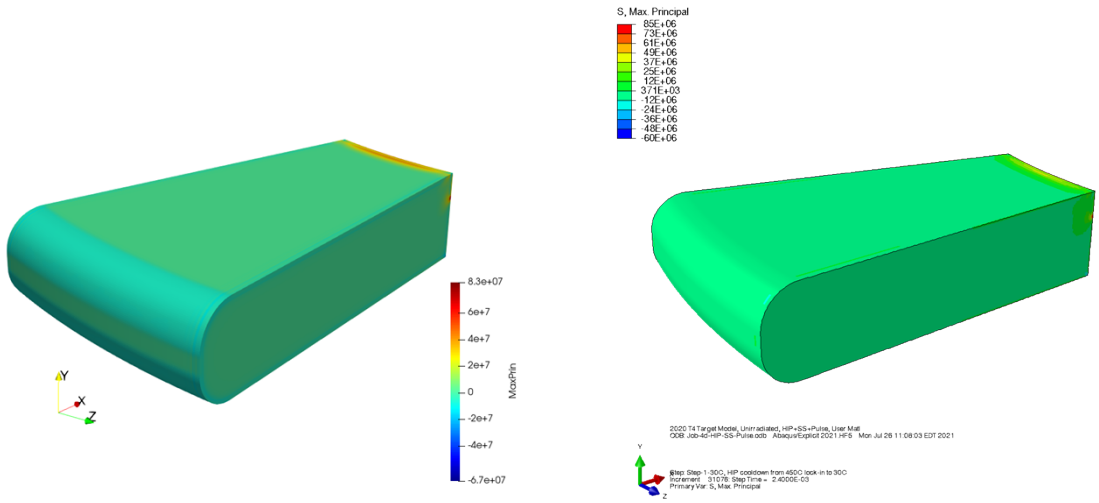


Figure 27: Sierra (L) and Abaqus (R) contour plots of tungsten maximum principal stress after the HIP step.

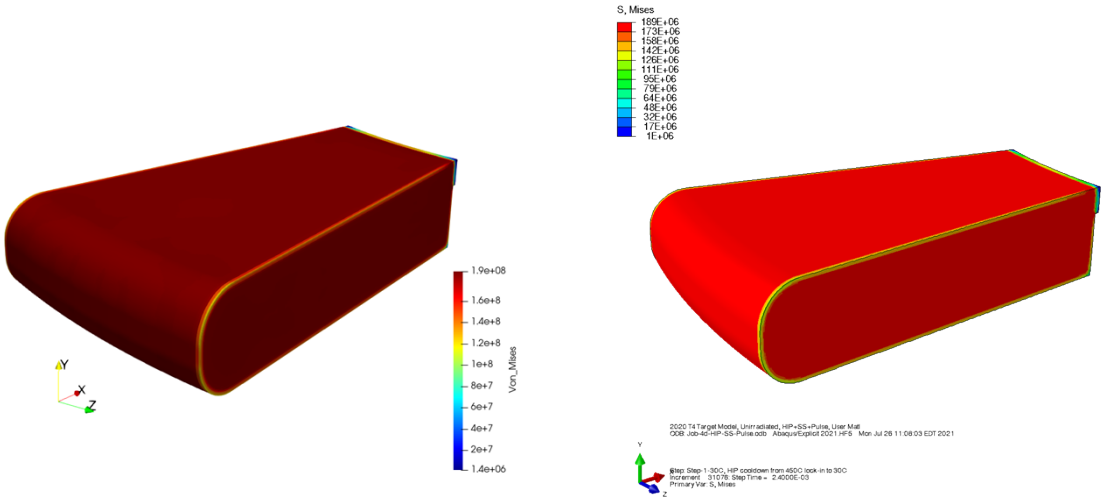


Figure 28: Sierra (L) and Abaqus (R) contour plots of tantalum von Mises equivalent stress after the HIP step.

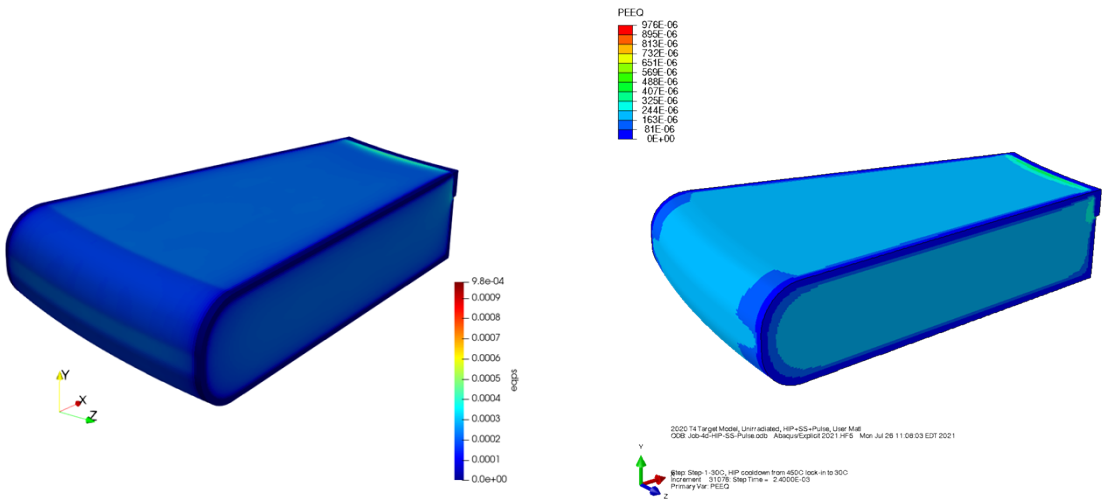


Figure 29: Sierra (L) and Abaqus (R) contour plots of tantalum cumulative equivalent plastic strain after the HIP step.

5.3.2 Steady-State

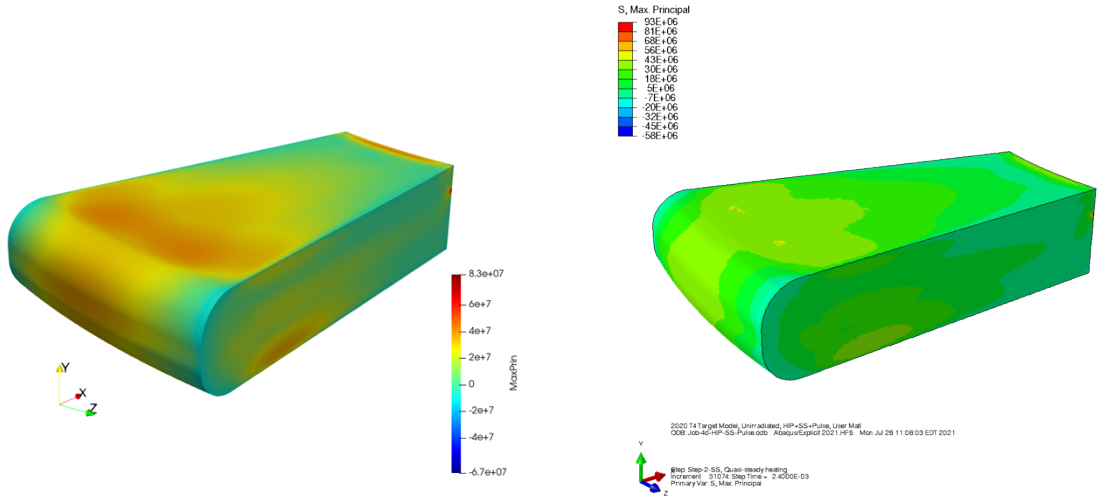


Figure 30: Sierra (L) and Abaqus (R) contour plots of tungsten maximum principal stress after the steady-state warm-up step.

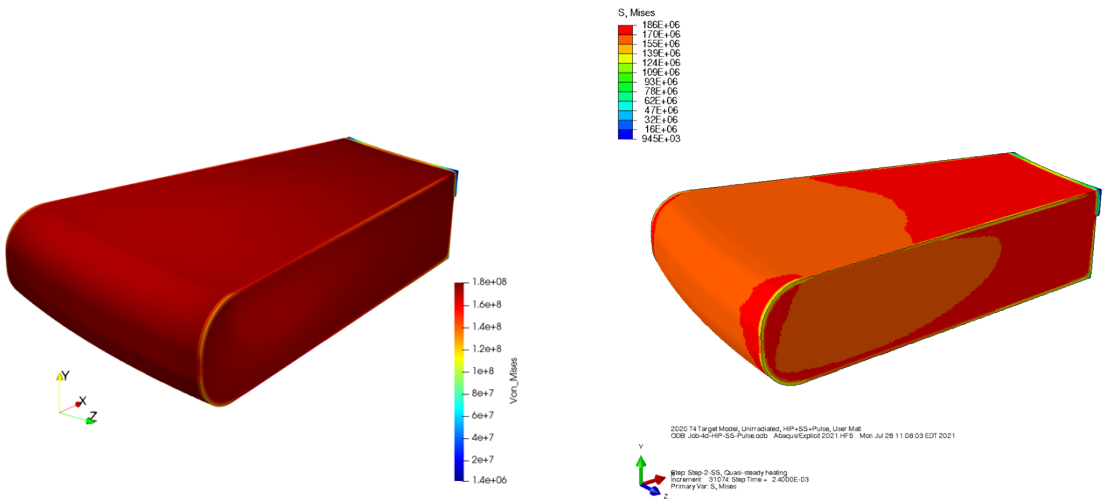


Figure 31: Sierra (L) and Abaqus (R) contour plots of tantalum von Mises equivalent stress after the steady-state warm-up step.

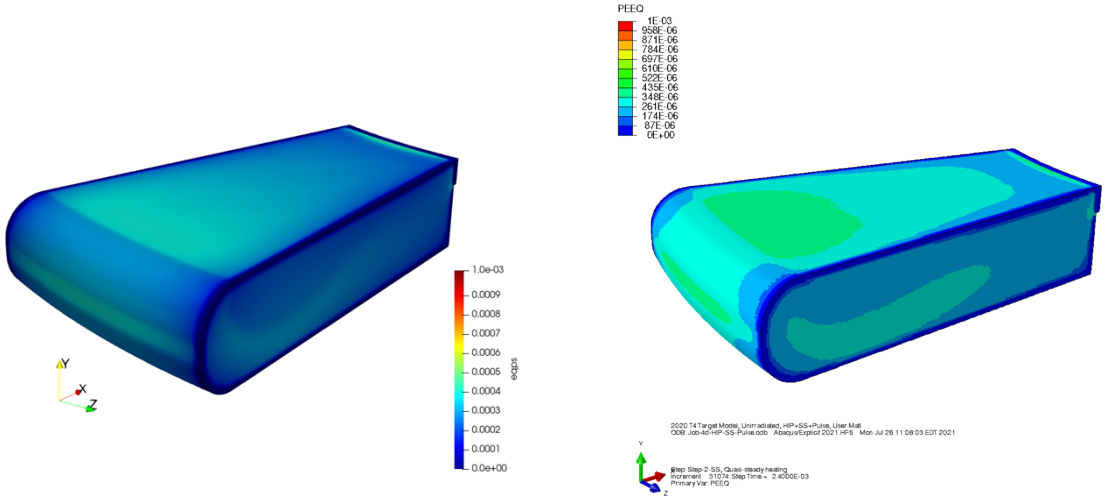


Figure 32: Sierra (L) and Abaqus (R) contour plots of tantalum cumulative equivalent plastic strain after the steady-state warm-up step.

5.3.3 Dynamic Response

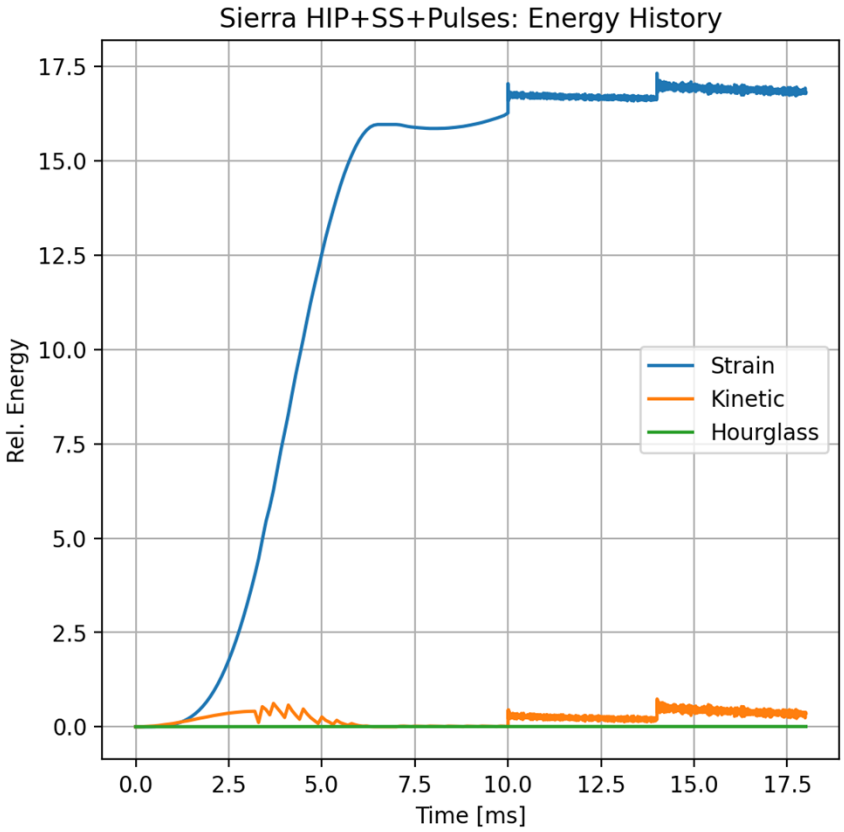


Figure 33: Energy history showing minimal hourglass and kinetic effects compared to strain.

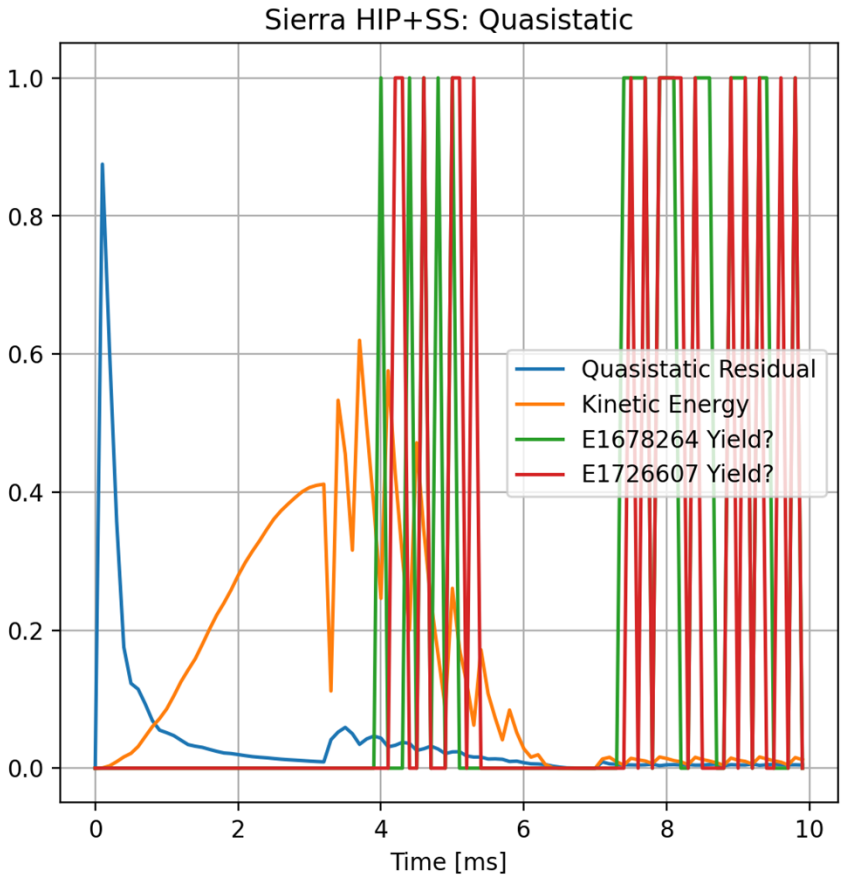


Figure 34: Evolution of the quasistatic HIP and SS steps showing converging quasistatic residual. The oscillations in kinetic energy align with incipient yielding of the tantalum.

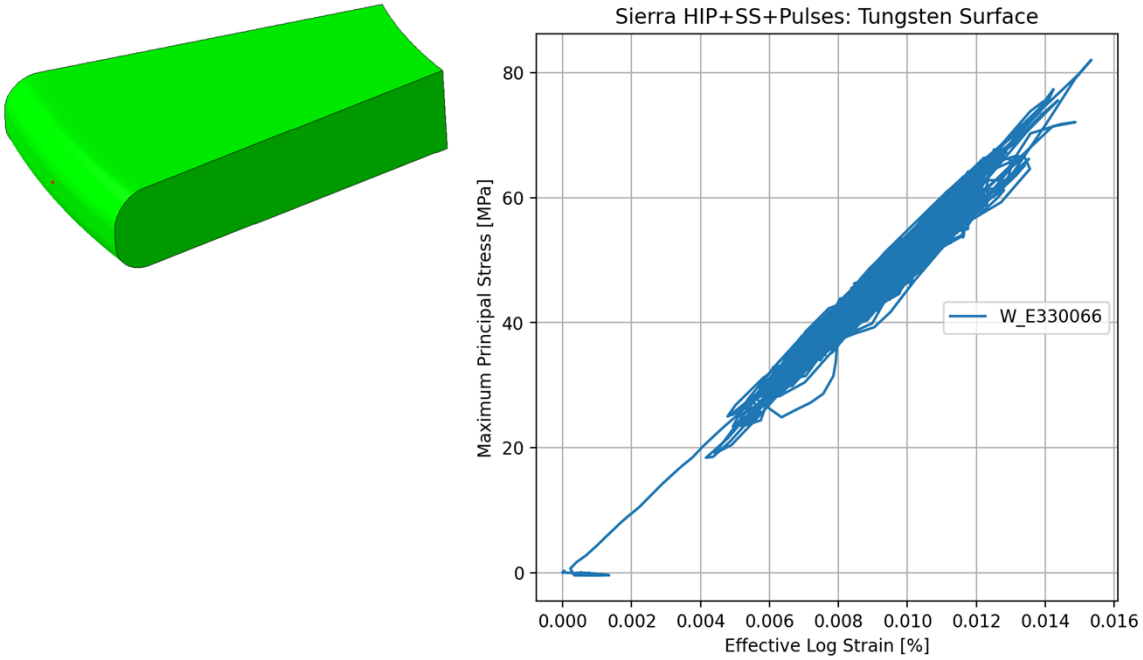


Figure 35: Stress-strain response of a tungsten surface element.

Figure 36: Stress-strain response of a tantalum surface element showing close alignment with Abaqus.

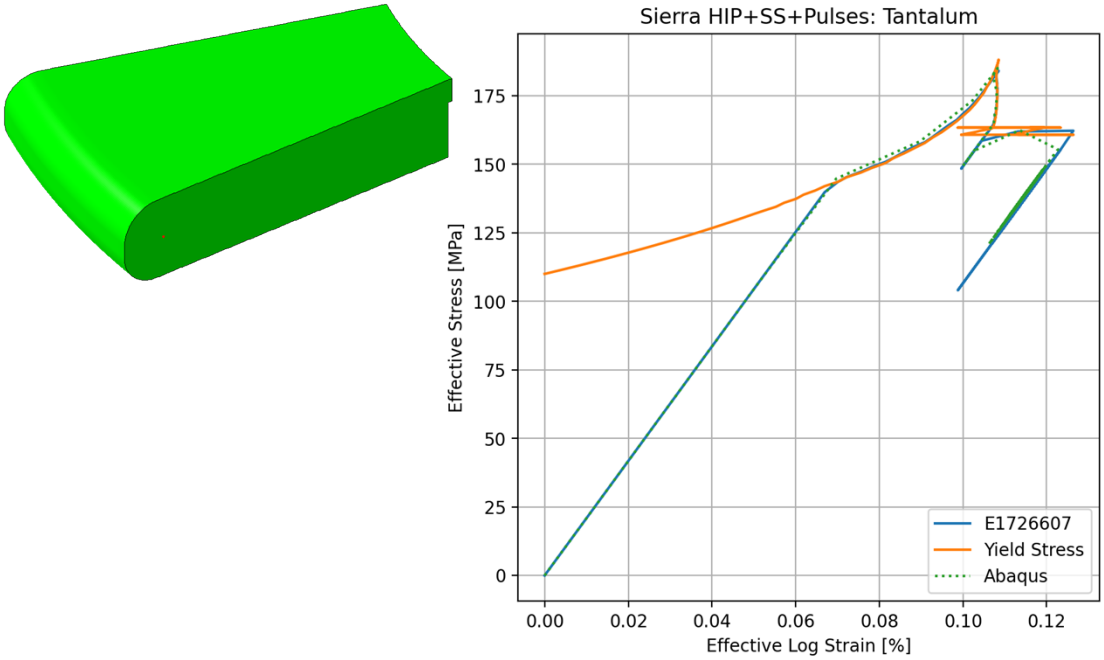


Figure 37: Stress-strain response of a tantalum surface element. The oscillation strain amplitude is larger than Abaqus predictions for the same element.

6. FATIGUE ANALYSIS USING SIERRA RESULTS

Dassault Systèmes SIMULIA's fe-safe® software is used by the STS Target Analysis team for fatigue analysis of the tungsten block and tantalum cladding. A procedure, detailed in the following section, has been created to transfer the Sierra simulation pulse results into an equivalent Abaqus ODB file which can then be read by fe-safe®. Fatigue analysis can then proceed for the tantalum cladding and tungsten block components.

6.1 SIERRA SOLUTION EXPORT TO ABAQUS ODB

6.1.1 Step 1 – Mesh Transfer to Abaqus

This step creates an Abaqus ODB results file with the mesh. First, the Sierra mesh in Genesis format (e.g. Mesh_Target.g) is imported in CUBIT and then exported as Abaqus INP format. This creates a full Abaqus input deck including fake material definition and fake solution step. Second, the Abaqus "datacheck" command can be used to convert the input file into an ODB results file. The results file will be empty save for the mesh. This can be invoked from a terminal window as:

```
abaqus datacheck job=<jobname> input=<inputfile>.inp interactive
```

6.1.2 Step 2 – Sierra Export to Python

This step uses the SEACAS exodus python module to post-process the Sierra simulation output file. It gathers the elemental stress tensors at all available time steps and saves the results in binary Numpy format. The code is listed in Appendix 8.6 and can be invoked from a terminal window as:

```
python sierraExport.py
```

Note that the Sierra input file was set to create a results file that only contained the tantalum cladding and only wrote results for the pulse and dynamic response steps. This is done purposefully such that the resulting fatigue analysis will only have the relevant data.

6.1.3 Step 3 – Python Export to Abaqus

This step uses Abaqus python to write the Sierra stress tensor results into the empty ODB file. The script first checks if a step named "Sierra" exists in the ODB file. If it does not, the step is created. Then a count-controlled loop writes time frames and stress results using the previously exported Numpy arrays. The code is listed in Appendix 8.7 and can be invoked from a terminal window as:

```
qsub sierra2ODB.sh
```

The TORQUE script is invoked to run the Abaqus python code from an OZ3 compute node as these have much more RAM than the head node. Even so, the test case presented in this report ended with a "Killed" full memory error after 3666 time frames had been processed. The solution is to run the above "qsub" command twice. If the step "Sierra" already exists in the ODB, the code script will find the last written time frame and then continue writing data from that point.

6.2 FATIGUE ANALYSIS

6.2.1 Input Files

A fatigue analysis using fe-safe® can be performed using the script files listed in Appendix 8.8 "FESAFE Fatigue Analysis". (Throughout this appendix listing, key file and directory inputs are shown in bold.) A separate analysis is performed for tungsten block fatigue and for tantalum cladding fatigue.

An analysis is started using the Torque job scheduler via the **`qsub FESAFE_run.sh`** terminal command. This executes fe-safe® from a compute node on the OZ3 cluster and runs a macro. The macro file switches to the active directory, loads the Abaqus ODB file, searches for any load steps containing the phrase "Sierra", and scans those frames for elemental stresses. This can take several hours depending on the file size. Once the scan is complete, the appropriate element group is selected (e.g. "PART-DEFAULT_1_TA_CLAD" or "PART-DEFAULT_1_W_BLOCK_SURF") and the analysis file is run.

Two analysis files are listed in the appendix. One is for fatigue analysis of the tungsten block which calculates the fatigue life under fully irradiated conditions (i.e. reduced tungsten thermal conductivity and no residual HIP stresses). To reduce file size, only the surface elements of the tungsten block are analyzed. The other analysis file is for fatigue analysis of the tantalum cladding which calculates the fatigue life under unirradiated conditions (i.e. maximal tensile HIP stresses). The fatigue parameters and material characteristics are described in other DAC publications [6].

Finally, the analysis file references a loading file. The loading file instructs fe-safe® to treat a cycle as a simple sequential loading of all timesteps. These are called "datasets" (ds) in fe-safe® and should match the total number of timesteps loaded from the Abaqus ODB file.

Once analysis is complete, results can be graphically post-processed from an Abaqus ODB results file.

6.2.2 Tungsten Fatigue Results

Figure 38 through Figure 41 summarize the results of the tungsten fatigue analysis using irradiated conditions.

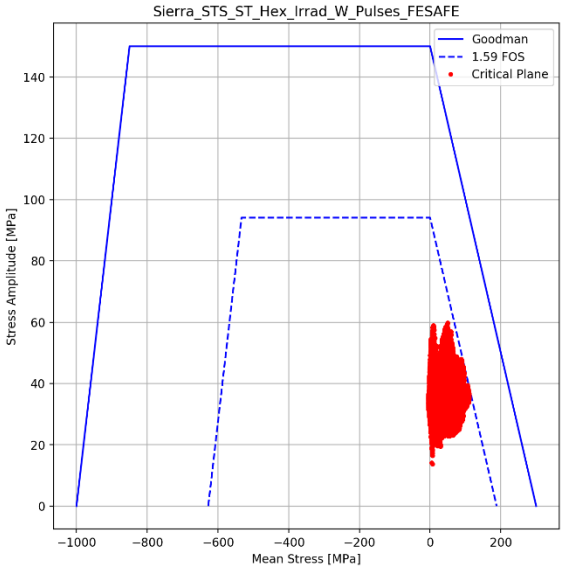


Figure 38: Goodman-Haigh diagram for tungsten fatigue assuming irradiated conditions. Only surface elements were included in the analysis as interior elements experience more compression. Each red dot represents a surface node result and was

analyzed using normal stresses critical plane theory. For a defined life of 130 million cycles, the factor of safety is 1.6 when compared to the modified Goodman failure envelope.

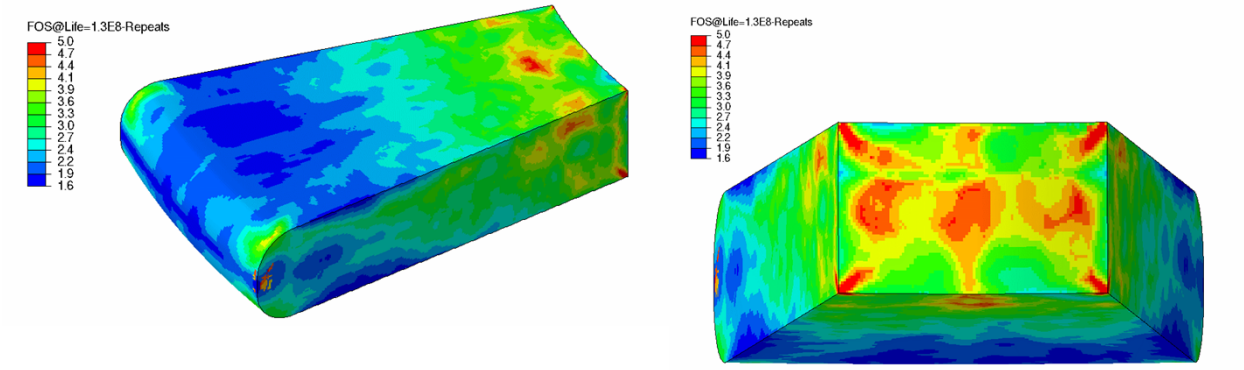


Figure 39: Contour plot of unaveraged fatigue factor-of-safety for the tungsten block under irradiated conditions.

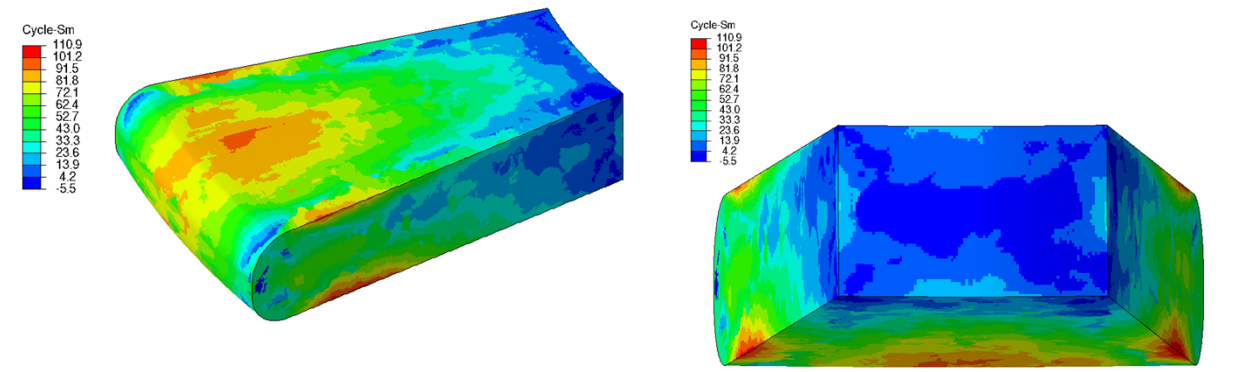


Figure 40: Contour plot of unaveraged nodal mean stress for the tungsten block under irradiated conditions.

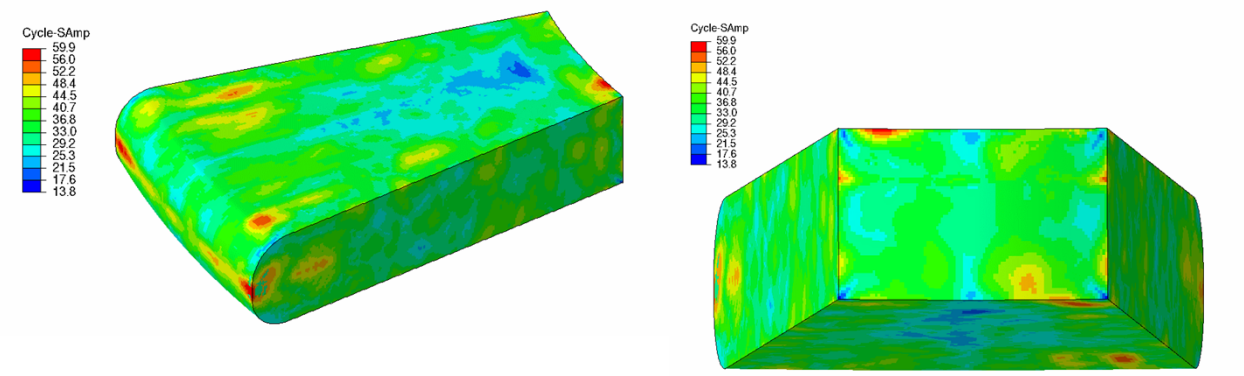


Figure 41: Contour plot of unaveraged nodal worst-case stress amplitude for the tungsten block under irradiated conditions.

6.2.3 Tantalum Fatigue Results

Figure 42 through Figure 45 summarize the results of the tantalum fatigue analysis using unirradiated conditions.

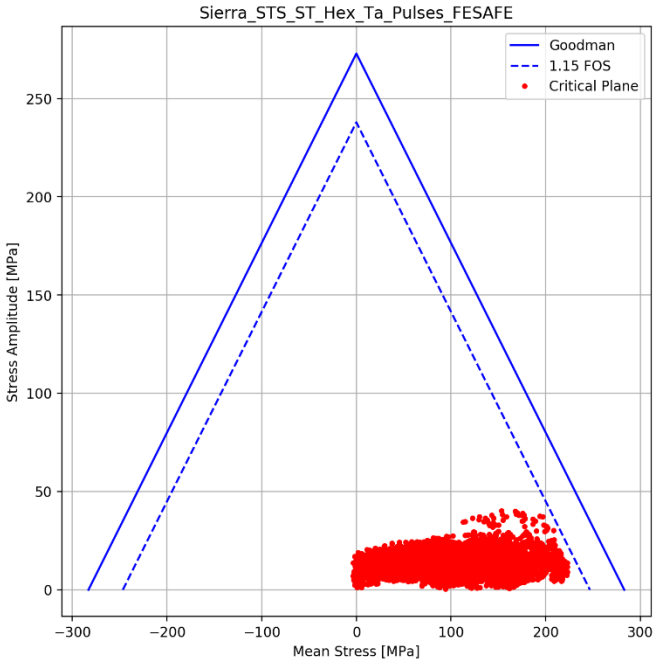


Figure 42: Goodman-Haigh diagram for tantalum cladding fatigue assuming unirradiated conditions. Each red dot represents a nodal result and was analyzed using stress-based Brown-Miller fatigue theory with Goodman mean stress correction . For a defined life of 130 million cycles, the factor of safety is ~1.1 when compared to the modified Goodman failure envelope.

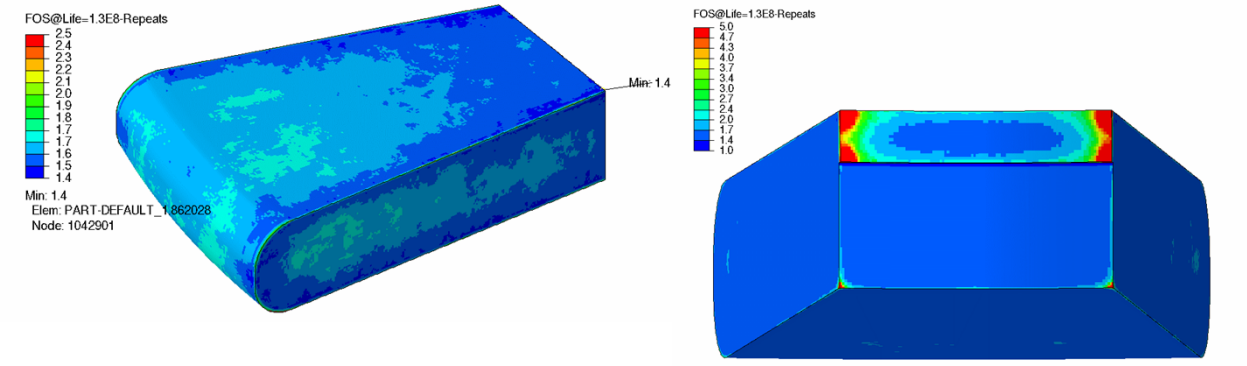


Figure 43: Contour plot of unaveraged fatigue factor-of-safety for the tantalum cladding under unirradiated conditions. The left plot clips the rear shelf feature (which has a stress concentration) to better show safety factors away from this region.

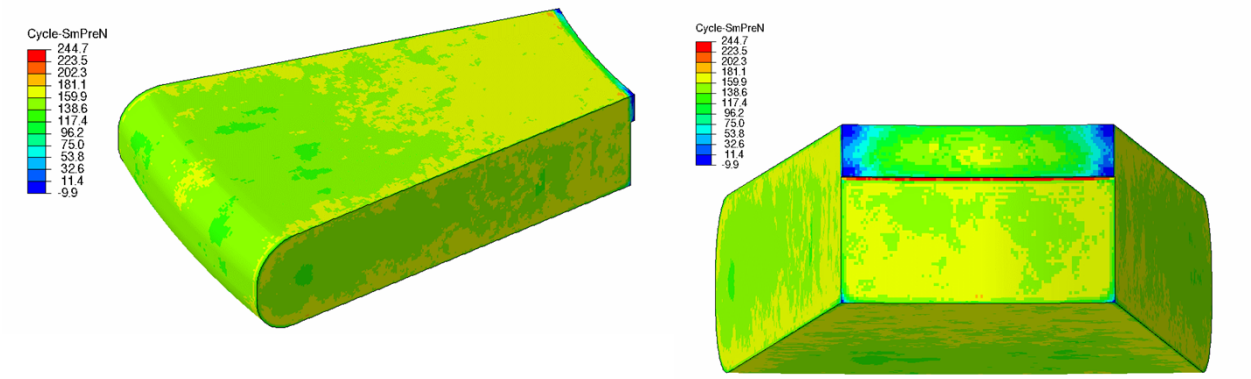


Figure 44: Contour plot of unaveraged nodal mean stress for the tantalum cladding under unirradiated conditions.

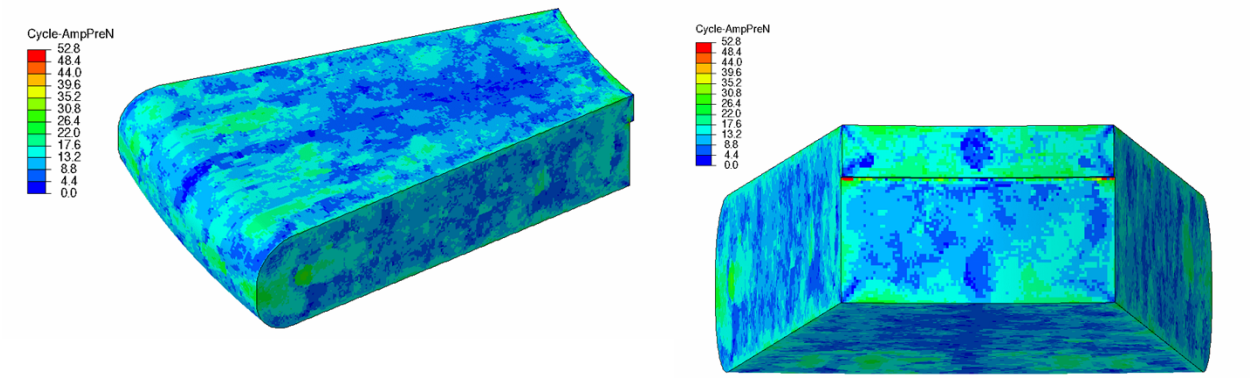


Figure 45: Contour plot of unaveraged nodal worst-case stress amplitude for the tantalum cladding under unirradiated conditions.

7. REFERENCES

[1] S. R. Chen and G. T. I. Gray, "Constitutive Behavior of Tantalum and Tantalum-Tungsten Alloys," *Metallurgical and Materials Transactions A,* vol. 27A, pp. 2994-3006, 1996.

[2] D. R. Helebrant and R. I. Stephens, "Cyclic Yield Behavior of Tantalum," *Journal of Materials,* vol. 7, no. 4, pp. 536-541, 1972.

[3] "sierraMISC: Miscellaneous Python tools for use with Sierra FEA software," 17 August 2021. [Online]. Available: https://github.com/jtipton2/sierraMISC. [Accessed 21 September 2021].

[4] "3.1.7. Explicit Quasistatic Mode," in *Sierra/SolidMechanics 4.56 User's Guide (SAND2020-3547)*, Sandia National Laboratory, 23 March 2020, pp. 98-100.

[5] "7.14.8. Tied Multi-Point Constraints," in *Sierra/SolidMechanics 4.56 User's Guide (SAND2020-3547)*, Sandia National Laboratory, 23 March 2020, pp. 594-597.

[6] T. McManamy, J. Tipton, M.-T. Kao, I. Remec and A. Jacques, "Target Block Lifetime Design Analysis Calculations (Preliminary)," Second Target Station (STS) Project, S03020100-DA0001, Oak Ridge National Laboratory, February 2021.

8. APPENDICES - PROGRAM FILES

8.1 TORQUE JOB SCHEDULER SIERRA SCRIPT

```
#!/bin/sh
#PBS -V
#PBS -j oe
#PBS -m abe
#PBS -l nodes=16:ppn=8                # specify # of nodes
#PBS -N Job_3_HIP_SS_Pulses           # specify the job name
#PBS -M UID@ornl.gov                  # specify your email address
#PBS -q all                           # specify the queue: all / slow / fast
# #PBS -W depend=afterany:6923

cd $PBS_O_WORKDIR

# Define particulars of this run:
INPUT_FILENAME=Job_3_HIP_SS_Pulses.i

# Number of processors
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS processors

# Copy files to compute node for I/O.
# NOTE: BE CAREFUL. THIS WILL COPY ALL FILES IN THE DIRECTORY
#ssh $PBS_O_HOST
#WORK_DIR=/local/$USER/$PBS_JOBID/
#mkdir -p $WORK_DIR
#cp -r $PBS_O_WORKDIR/* $WORK_DIR/
#cd $WORK_DIR

# Executable for sierra
EXEC=/data/fem/sierra_4.56.4b/install/sntools/engine/sierra #2020

echo $EXEC -j $NPROCS adagio -i $INPUT_FILENAME

$EXEC --pre -j $NPROCS adagio -i $INPUT_FILENAME

$EXEC --run -j $NPROCS adagio -i $INPUT_FILENAME

$EXEC --post -j $NPROCS adagio -i $INPUT_FILENAME

sleep 60 # Give the log file a chance to complete

# clean up
rm -r *.e.*
rm -r *.g.*

# Copy completed job files from compute node to job submit directory.
#cp -r $WORK_DIR/* $PBS_O_WORKDIR/
#cd $PBS_O_WORKDIR
#rm -rf $WORK_DIR/

exit
```

8.2 TEMPERATURE INTERPOLATION PYTHON MODULE

```
import numpy as np
import pytetgen as pytet
from sklearn import neighbors

def fitData(sourceCoord,sourceTemp,targetID,targetIndex,targetCoord):
    """Fit point cloud temperatures from a neutronics mesh onto FEA mesh nodes for
    thermostructural simulations.  Interpolation uses Delaunay triangularization
    with a Barycentric interpolation.  Any nodes external to the triangularization
    (due to coordinate rounding) are then extrapolated using a distance weighting of the
    3 nearest neighbors.

    Parameters:

    sourceCoord (float): x,y,z coordinates (m) of source point cloud
    sourceTemp (float): temperatures (deg C) of source point cloud
    targetID (int): Sierra mesh node_id_map
    targetIndex (int): Sierra mesh node index (1 to num_nodes)
    targetCoord (float): Sierra mesh node x,y,z coordinates (m)

    Returns:

    Numpy structured array of target Sierra nodes with columns:
    'Index' (int): echo of targetIndex
    'ID' (int): echo of targetID
    'x' (float): echo of targetCoord[:,0]
    'y' (float): echo of targetCoord[:,1]
    'z' (float): echo of targetCoord[:,2]
    'Temp' (float): nodal temperatures (deg-C)

    """

    #
    # Build Delaunay Tet Mesh from Source Point Cloud
    #
    tri = pytet.Delaunay(sourceCoord)
    # Any targets that fall outside the Delaunay cells return a
    # simplex value of -1.  Need to filter these and later
    # determine via some sort of extrapolation.
    tetsRaw = tri.find_simplex(targetCoord)
    outMask = tetsRaw > 0
    tets = tetsRaw[outMask]
    R = targetCoord[outMask]


    #
    # Calculate Barycentric Coordinates | Local Weights for Each Target
    # Coordinate Inside Source Tetrahedron
    #
    """
    Background:  Interpolation of point clouds in Abaqus is painfully slow.  SciPy has a way to
do this
    using LinearNDInterpolator, however, it is also painfully slow.  I tried using
<stackoverflow.com/a/56628900>
    and discovered that it is the Delaunay triangularization that is taking a long time.
Amazingly, a set of
    points that still had not completed in over a day was handled in seconds using pyTetGen.  The
issue is
    that SciPy works in python while pyTetGen interfaces directly with C libraries.  What follows
is then my
    reverse engineering of barycentric interpolation

    Basic idea and formula in:
    https://codeplea.com/triangular-interpolation
    https://en.wikipedia.org/wiki/Barycentric_coordinate_system

    Great code example:  https://codereview.stackexchange.com/a/41089
    But uses scipy.spatial.Delaunay.transform function
    I had to work from the Wikipedia example to get a workaround.
```

```
    If I use scipy.spatial.Delaunay(points), I get the same answer as
    in the link.

    How to piecewise invert a collection of matrices:
    https://stackoverflow.com/questions/17924411/vectorized-partial-inverse-of-an-nmm-tensor-
with-numpy

    Using np.newaxis to add a dimension to an array:
    https://stackoverflow.com/questions/26333005/numpy-subtract-every-row-of-matrix-by-
vector/26333184

    How to transpose individual matrices nested inside an array:
    https://jameshensman.wordpress.com/2010/06/14/multiple-matrix-multiplication-in-numpy/
    """
    T = sourceCoord[tri.simplices[tets,:3]] - sourceCoord[tri.simplices[tets,3]][:, np.newaxis]
    Ttrans = np.transpose(T,(0,2,1))
    Tinv = np.linalg.inv(Ttrans)

    RminusR4 = R - sourceCoord[tri.simplices[tets,3]]

    b = np.einsum('ijk,ik->ij', Tinv, RminusR4)

    bcoords = np.c_[b, 1 - b.sum(axis=1)]


    #
    # Calculate Interpolated Target Temperatures with Coordinates and Node ID
    #
    outTemp = np.multiply(sourceTemp[tri.simplices[tets]],bcoords).sum(axis=1)
    outCoord = targetCoord[outMask]
    outID = targetID[outMask]
    outIndex = targetIndex[outMask]


    #
    # Unmatched Nodes - KNN Distance Weighted Averaging
    #
    # np.savetxt('unmatched.txt',targetCoord[outMask!=True],delimiter=',')
    #
    # (Visual inspection of this node set in ParaView showed it to be the
    #  outer surface nodes.  There's a rounding problem on node coords
    #  between Abaqus, Atilla, and Sierra.  Use a distance weighted average
    #  of the 3 closest neighbors for these elements.)
    #
    # https://stackoverflow.com/questions/48312205/find-the-k-nearest-neighbours-of-a-point-in-
3d-space-with-python-numpy
    #
    outerCoord = targetCoord[outMask!=True]
    outerID = targetID[outMask!=True]
    outerIndex = targetIndex[outMask!=True]
    knn = neighbors.KNeighborsRegressor(3, weights='distance')
    outerTemp = knn.fit(sourceCoord,sourceTemp).predict(outerCoord)


    #
    # Join the results from the interpolation and extrapolations
    # Join everything into 1 array
    # Sort the array by Sierra node index
    #
    finalIndex = np.hstack((outIndex,outerIndex))
    finalID = np.hstack((outID,outerID))
    finalCoord = np.vstack((outCoord,outerCoord))
    finalTemp = np.hstack((outTemp,outerTemp))

    #final = np.column_stack([finalIndex, finalID, finalCoord, finalTemp])
    #final[final[:,0].argsort()]

    final = np.empty(len(finalIndex), dtype=[('Index', int),
                                             ('ID', int),
                                             ('x', float),
                                             ('y', float),
```

```
                                        ('z', float),
                                        ('Temp', float)]))

final['Index'] = finalIndex
final['ID'] = finalID
final['x'] = finalCoord[:,0]
final['y'] = finalCoord[:,1]
final['z'] = finalCoord[:,2]
final['Temp'] = finalTemp

final.sort(order='Index')

return final
```

8.3 TEMPERATURE INTERPOLATION SCRIPT

```
from fitData import *
from exodus import exodus, copyTransfer

#=========================================================================
# FIT POINT CLOUD TEMPERATURE SETS TO SIERRA MESH
#=========================================================================

#
# Load Target Mesh
#
mesh = exodus('Mesh_Target.g',mode='r',array_type='numpy')

targetID = np.array(mesh.get_node_id_map())
targetIndex = np.arange(1,len(targetID)+1)
targetCoord = np.transpose(mesh.get_coords())

mesh.get_elem_blk_names()
mesh.get_node_set_ids()
mesh.get_node_set_names()

mesh.get_node_set_name(3)
tantalumIndex = mesh.get_node_set_nodes(3)
tantalumID = tantalumIndex - 1
tantalumCoord = targetCoord[tantalumID]

mesh.get_node_set_name(6)
tungstenIndex = mesh.get_node_set_nodes(6)
tungstenID = tungstenIndex - 1
tungstenCoord = targetCoord[tungstenID]

mesh.close()

#
# Load Source Mesh and Temperatures and Fit the Data
#

source = np.loadtxt(open('DTTa','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
TaPulseA= fitData(sourceCoord,sourceTemp,tantalumID,tantalumIndex,tantalumCoord)

source = np.loadtxt(open('DTTa121','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
TaPulseB = fitData(sourceCoord,sourceTemp,tantalumID,tantalumIndex,tantalumCoord)

source = np.loadtxt(open('Tantalum_Temperature_deg_C.csv','rb'), delimiter=',', skiprows=1)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
TaSS = fitData(sourceCoord,sourceTemp,tantalumID,tantalumIndex,tantalumCoord)

source = np.loadtxt(open('DTW','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
WPulseA= fitData(sourceCoord,sourceTemp,tungstenID,tungstenIndex,tungstenCoord)

source = np.loadtxt(open('DTW121','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
WPulseB = fitData(sourceCoord,sourceTemp,tungstenID,tungstenIndex,tungstenCoord)

source = np.loadtxt(open('Tungsten_Temperature_deg_C.csv','rb'), delimiter=',', skiprows=1)
sourceCoord = source[:,:3]
sourceTemp = source[:,3]
WSS = fitData(sourceCoord,sourceTemp,tungstenID,tungstenIndex,tungstenCoord)

finalPulseA = np.hstack((TaPulseA,WPulseA))
finalPulseB = np.hstack((TaPulseB,WPulseB))
```

```
finalSS = np.hstack((TaSS,WSS))


#
# Save Temperature/Time Fields to New Exodus File
#
addGlobalVariables = []
addNodeVariables = ["NodalTempField"]
addElementVariables = []
# copy exodus file and add variables to all nodes
exo = copyTransfer('Mesh_Target.g','Temps_Target.g','ctype',addGlobalVariables,addNodeVariables,
addElementVariables)

times = np.array([0.0,
                  0.0024,
                  0.0048,
                  0.0048007,
                  0.0088007,
                  0.0088014,
                  0.0128014])

temps = np.zeros([len(targetID),len(times)])

temps[:,0] = 450.0
temps[:,1] = 30.0
temps[:,2] = finalSS['Temp']
temps[:,3] = finalSS['Temp'] + finalPulseA['Temp']
temps[:,4] = finalSS['Temp'] + finalPulseA['Temp']
temps[:,5] = finalSS['Temp'] + finalPulseB['Temp']
temps[:,6] = finalSS['Temp'] + finalPulseB['Temp']

for ii in range(len(times)):
  exo.put_node_variable_values('NodalTempField',ii+1,temps[:,ii])
  exo.put_time(ii+1,float(times[ii]))

exo.close()
```

8.4 SIERRA SIMULATION INPUT FILE

8.4.1 Unirradiated Condition

```
begin sierra

  ##
  ##-------------------------------------------------------------------
  ## FUNCTIONS
  ##-------------------------------------------------------------------
  ##

  begin function SmoothTempHIP
    evaluate expression = "450.0 - 420.0 * cos_ramp(t, 0.0, 0.0065)"
  end

  begin definition for function T450to30
    type is piecewise linear
    begin values
      0.0     450.0
      0.0005  450.0
      0.0025  250.0
      0.007    30.0
    end values
  end

  ##-------------------------------------------------------------------
  ## MATERIALS - Tantalum
  ##-------------------------------------------------------------------

  begin function CTE_Ta
    type is analytic
    # Initial Temp is the Reference Temp
    evaluate expression is "6.3e-06 * x"
  end

  begin function Yield_Ta
    type is analytic
    evaluate expression is "1.0 - ((x - 25.0)/(2976.85 - 25.0))^0.4"
  end

  begin function Hard25
    type is analytic
    evaluate expression is "   \#
      (1.0) *                  \#
      0.6 *                    \#
      1.0e+6 *                 \#
      (340.0 + 750.0*x^0.7)    \#
      "
  end

  begin function Hard200
    type is analytic
    evaluate expression is "                          \#
      (1 - ((200.0 - 25.0)/(2976.85 - 25.0))^0.4) *  \#
      0.6 *                                          \#
      1.0e+6 *                                       \#
      (340.0 + 750.0*x^0.7)                          \#
      "
  end

  begin function Hardening25
    type is piecewise linear
    begin values
       0.0     204953384.3
       0.005   216032789.4
    end values
  end

  begin function Hardening200
```

```
  type is piecewise linear
  begin values
     0.0     138757254.4
     0.005   146258217.8
  end values
end

begin property specification for material Ta_Bilinear
  density = 16600.
  THERMAL LOG STRAIN FUNCTION = CTE_Ta
  begin parameters for model ELASTIC_PLASTIC
    youngs modulus = 1.88e+11
    poissons ratio = 0.35
    #
    #Hardening Behavior
    #
    YIELD STRESS = 2.0e+8
    BETA = 0.0
    HARDENING MODULUS = 1.0e+9
  end
end

begin property specification for material Ta_ThermoElasticPlastic
  density = 16600.
  THERMAL LOG STRAIN FUNCTION = CTE_Ta
  begin parameters for model THERMOELASTIC_PLASTIC
    youngs modulus = 1.88e+11
    poissons ratio = 0.35
    #
    # Hardening Behavior
    #
    BETA = 0.0
    YIELD STRESS = 204.0E+6
    YIELD STRESS FUNCTION = Yield_Ta
    TEMPERATURES = 200.0
    HARDENING FUNCTIONS = Hard200
  end
end


##------------------------------------------------------------------------
## MATERIALS - Tungsten
##------------------------------------------------------------------------

begin function CTE_W
  type is analytic
  # Initial Temp is the Reference Temp
  evaluate expression is "4.5e-06 * x"
end

begin property specification for material W_Elastic
  density = 19250.
  THERMAL LOG STRAIN FUNCTION = CTE_W
  begin parameters for model ELASTIC
    youngs modulus = 3.98e+11
    poissons ratio = 0.28
  end
end


##------------------------------------------------------------------------
## MESH DEFINITION
##------------------------------------------------------------------------

begin finite element model TargetBlockMesh
  database name = Mesh_Target.g
  database type = exodusII

  begin parameters for block W_BLOCK
    material W_Elastic
    solid mechanics use model ELASTIC
```

```
  end

  begin parameters for block TA_CLAD
    material Ta_ThermoElasticPlastic
    solid mechanics use model THERMOELASTIC_PLASTIC
  end

end


begin finite element model TargetBlockTemps
  database name = Temps_Target.g
  database type = exodusII

  begin parameters for block W_BLOCK
    material W_Elastic
    solid mechanics use model ELASTIC
  end

  begin parameters for block TA_CLAD
    material Ta_ThermoElasticPlastic
    solid mechanics use model THERMOELASTIC_PLASTIC
  end

end

##-------------------------------------------------------------------------
## ANALYSIS SETTINGS
##-------------------------------------------------------------------------

begin presto procedure HIP_SS

  begin time control
    begin time stepping block Step_1_HIP
      start time = 0.0
      begin parameters for presto region TargetBlock
        Step Interval = 1000
        number of quasistatic time steps = 17500
      end
    end
    begin time stepping block Step_2_SS
      start time = 0.007
      begin parameters for presto region TargetBlock
        Step Interval = 1000
        number of quasistatic time steps = 7500
      end
    end
    begin time stepping block Step_3_Pulses
      start time = 0.01
      begin parameters for presto region TargetBlock
        Step Interval = 1000
      end
    end
    termination time = 0.0180014
  end

  begin presto region TargetBlock
    use finite element model TargetBlockMesh

    begin heartbeat output
      stream name = %B.hrt
      append = OFF
      format = CSV
      labels = OFF
      legend = ON
      precision = 6
      timestamp format = "[%H:%M:%S],"
      at time 0.0       increment = 1.0e-04
      at time 0.01      increment = 1.0e-07
      at time 0.0100007 increment = 1.0e-06
      at time 0.0140007 increment = 1.0e-07
```

```
  at time 0.0140014 increment = 1.0e-06
  global timestep as DT
  global internal_energy as IE
  global kinetic_energy as KE
  global strain_energy as SE
  global hourglass_energy as HGE
  global external_energy as ExtE
  global contact_energy as ContE
  global time as TIME
  global quasistatic_residual as QuasiRES
  # ABAQUS ELEMENT 330066 IP COORD
  element element_id nearest location -80.2509E-03, -500.0E-06, -84.1315E-03 as W_Elem_ID
  element temperature at element 330066 as W_Elem_Temp
  element max_principal_stress at element 330066 as W_Elem_MaxPrin
  element effective_log_strain at element 330066 as W_Elem_EffLE
  # ABAQUS ELEMENT 1678264 IP COORD
  element element_id nearest location -30.7043E-03, 29.75E-03, -105.866E-03 as Ta_Elem1_ID
  element temperature at element 1678264 as Ta_Elem1_Temp
  element von_mises at element 1678264 as Ta_Elem1_Mises
  element effective_log_strain at element 1678264 as Ta_Elem1_EffLE
  element yield_stress at element 1678264 as Ta_Elem1_YS
  element yield_flag at element 1678264 as Ta_Elem1_Yflag
  element eqps at element 1678264 as Ta_Elem1_EQPS
  # ABAQUS ELEMENT 1726607 IP COORD
  element element_id nearest location -57.4985E-03, -500.0E-06, -1.24983E-03 as Ta_Elem2_ID
  element temperature at element 1726607 as Ta_Elem2_Temp
  element von_mises at element 1726607 as Ta_Elem2_Mises
  element effective_log_strain at element 1726607 as Ta_Elem2_EffLE
  element yield_stress at element 1726607 as Ta_Elem2_YS
  element yield_flag at element 1726607 as Ta_Elem2_Yflag
  element eqps at element 1726607 as Ta_Elem2_EQPS
  # Note that these are ExodusII node INDECES
  #node displacement at node 213540 as Disp_Ta_213540
  #node temperature at node 213540 as Temp_Ta_213540
  #node displacement at node 1474864 as Disp_W_1474864
  #node temperature at node 1474864 as Temp_W_1474864
end

begin results output
  database name = %B_HIP_SS.e
  at time 0.0 increment = 0.001
  termination time = 0.01
  Nodal Variables = Velocity As V
  Nodal Variables = Displacement As U
  Nodal Variables = Temperature As NT11
  Element Variables = Stress
  Element Variables = Log_Strain
  Element Variables = Von_Mises
  Element Variables = max_principal_stress as MaxPrin
  Element Variables = Effective_Log_Strain As ELS
  Element Variables = eqps
  Nodal Variables = Status_Constraint_Flag as Status
end

#      begin results output
#        database name = %B_W.e
#        at time 0.0       increment = 0.0005
#        at time 0.01      increment = 1.0e-07
#        at time 0.0100007 increment = 1.0e-06
#        at time 0.0120007 increment = 0.001
#        at time 0.0140007 increment = 1.0e-07
#        at time 0.0140014 increment = 1.0e-06
#        at time 0.0160014 increment = 0.001
#        include = W_BLOCK
#        Nodal Variables = Displacement As U
#        Element Variables = max_principal_stress as MaxPrin
#        Element Variables = Effective_Log_Strain As ELS
#      end

begin results output
  database name = %B_Ta_Pulses.e
```

```
        start time = 0.01
        at time 0.01      increment = 1.0e-07
        at time 0.0100007 increment = 1.0e-06
        at time 0.0120007 increment = 0.001
        at time 0.0140007 increment = 1.0e-07
        at time 0.0140014 increment = 1.0e-06
        at time 0.0160014 increment = 0.001
        include = TA_CLAD
        Nodal Variables = Displacement As U
        Element Variables = Effective_Log_Strain As ELS
        Element Variables = Von_Mises
        Element Variables = EQPS
        Element Variables = Stress
      end

      begin tied mpc
        tied faces = W_BLOCK.W_TA
        tied nodes = TA_CLAD.TA_W
      end tied mpc

      begin initial condition
        include all blocks
        initialize variable name = temperature
        variable type = node
        magnitude = 450.0
      end

      begin prescribed temperature
        include all blocks
        active periods = Step_1_HIP
        function = SmoothTempHIP
      end

      begin prescribed temperature
        include all blocks
        active periods = Step_2_SS Step_3_Pulses
        copy variable = NodalTempField from model TargetBlockTemps MAP_BY_ID
      end prescribed temperature

    end presto region TargetBlock

  end presto procedure HIP_SS

end sierra
```

8.4.2 Irradiated Condition

```
begin sierra

  ##
  ##-------------------------------------------------------------------
  ## FUNCTIONS
  ##-------------------------------------------------------------------
  ##

  begin function SmoothTempHIP
    evaluate expression = "450.0 - 420.0 * cos_ramp(t, 0.0, 0.0065)"
  end

  begin definition for function T450to30
    type is piecewise linear
    begin values
      0.0     450.0
      0.0005  450.0
      0.0025  250.0
      0.007    30.0
    end values
  end
```

```
##------------------------------------------------------------------------
## MATERIALS - Tantalum
##------------------------------------------------------------------------

begin function CTE_Ta
  type is analytic
  # Initial Temp is the Reference Temp
  evaluate expression is "6.3e-06 * x"
end

begin function Yield_Ta
  type is analytic
  evaluate expression is "1.0 - ((x - 25.0)/(2976.85 - 25.0))^0.4"
end

begin function Hard25
  type is analytic
  evaluate expression is "   \#
    (1.0) *                  \#
    0.6 *                    \#
    1.0e+6 *                 \#
    (340.0 + 750.0*x^0.7)    \#
    "
end

begin function Hard200
  type is analytic
  evaluate expression is "                          \#
    (1 - ((200.0 - 25.0)/(2976.85 - 25.0))^0.4) *  \#
    0.6 *                                          \#
    1.0e+6 *                                       \#
    (340.0 + 750.0*x^0.7)                          \#
    "
end

begin function Hardening25
  type is piecewise linear
  begin values
     0.0     204953384.3
     0.005   216032789.4
  end values
end

begin function Hardening200
  type is piecewise linear
  begin values
     0.0     138757254.4
     0.005   146258217.8
  end values
end

begin property specification for material Ta_Bilinear
  density = 16600.
  THERMAL LOG STRAIN FUNCTION = CTE_Ta
  begin parameters for model ELASTIC_PLASTIC
    youngs modulus = 1.88e+11
    poissons ratio = 0.35
    #
    #Hardening Behavior
    #
    YIELD STRESS = 2.0e+8
    BETA = 0.0
    HARDENING MODULUS = 1.0e+9
  end
end

begin property specification for material Ta_ThermoElasticPlastic
  density = 16600.
  THERMAL LOG STRAIN FUNCTION = CTE_Ta
  begin parameters for model THERMOELASTIC_PLASTIC
    youngs modulus = 1.88e+11
```

```
    poissons ratio = 0.35
    #
    # Hardening Behavior
    #
    BETA = 0.0
    YIELD STRESS = 204.0E+6
    YIELD STRESS FUNCTION = Yield_Ta
    TEMPERATURES = 200.0
    HARDENING FUNCTIONS = Hard200
  end
end


##-------------------------------------------------------------------
## MATERIALS - Tungsten
##-------------------------------------------------------------------

begin function CTE_W
  type is analytic
  # Initial Temp is the Reference Temp
  evaluate expression is "4.5e-06 * x"
end

begin property specification for material W_Elastic
  density = 19250.
  THERMAL LOG STRAIN FUNCTION = CTE_W
  begin parameters for model ELASTIC
    youngs modulus = 3.98e+11
    poissons ratio = 0.28
  end
end


##-------------------------------------------------------------------
## MESH DEFINITION
##-------------------------------------------------------------------

begin finite element model TargetBlockMesh
  database name = Mesh_Target.g
  database type = exodusII

  begin parameters for block W_BLOCK_INT
    material W_Elastic
    solid mechanics use model ELASTIC
  end

  begin parameters for block W_BLOCK_SURF
    material W_Elastic
    solid mechanics use model ELASTIC
  end

  begin parameters for block TA_CLAD
    material Ta_ThermoElasticPlastic
    solid mechanics use model THERMOELASTIC_PLASTIC
  end

end


begin finite element model TargetBlockTemps
  database name = Temps_Target_Irrad.g
  database type = exodusII

  begin parameters for block W_BLOCK_INT
    material W_Elastic
    solid mechanics use model ELASTIC
  end

  begin parameters for block W_BLOCK_SURF
    material W_Elastic
    solid mechanics use model ELASTIC
```

```
    end

    begin parameters for block TA_CLAD
      material Ta_ThermoElasticPlastic
      solid mechanics use model THERMOELASTIC_PLASTIC
    end

  end

  ##-------------------------------------------------------------------
  ## ANALYSIS SETTINGS
  ##-------------------------------------------------------------------

  begin presto procedure HIP_SS

    begin time control
#      begin time stepping block Step_1_HIP
#        start time = 0.0
#        begin parameters for presto region TargetBlock
#          Step Interval = 1000
#          number of quasistatic time steps = 17500
#        end
#      end
      begin time stepping block Step_2_SS
        start time = 0.007
        begin parameters for presto region TargetBlock
          Step Interval = 1000
          number of quasistatic time steps = 7500
        end
      end
      begin time stepping block Step_3_Pulses
        start time = 0.01
        begin parameters for presto region TargetBlock
          Step Interval = 1000
        end
      end
      termination time = 0.0180014
    end

    begin presto region TargetBlock
      use finite element model TargetBlockMesh

      begin heartbeat output
        stream name = %B.hrt
        append = OFF
        format = CSV
        labels = OFF
        legend = ON
        precision = 6
        timestamp format = "[%H:%M:%S],"
        at time 0.0       increment = 1.0e-04
        at time 0.01      increment = 1.0e-07
        at time 0.0100007 increment = 1.0e-06
        at time 0.0140007 increment = 1.0e-07
        at time 0.0140014 increment = 1.0e-06
        global timestep as DT
        global internal_energy as IE
        global kinetic_energy as KE
        global strain_energy as SE
        global hourglass_energy as HGE
        global external_energy as ExtE
        global contact_energy as ContE
        global time as TIME
        global quasistatic_residual as QuasiRES
        # ABAQUS ELEMENT 330066 IP COORD
        element element_id nearest location -80.2509E-03, -500.0E-06, -84.1315E-03 as W_Elem_ID
        element temperature at element 330066 as W_Elem_Temp
        element max_principal_stress at element 330066 as W_Elem_MaxPrin
        element effective_log_strain at element 330066 as W_Elem_EffLE
        # ABAQUS ELEMENT 1678264 IP COORD
        element element_id nearest location -30.7043E-03, 29.75E-03, -105.866E-03 as Ta_Elem1_ID
```

```
  element temperature at element 1678264 as Ta_Elem1_Temp
  element von_mises at element 1678264 as Ta_Elem1_Mises
  element effective_log_strain at element 1678264 as Ta_Elem1_EffLE
  element yield_stress at element 1678264 as Ta_Elem1_YS
  element yield_flag at element 1678264 as Ta_Elem1_Yflag
  element eqps at element 1678264 as Ta_Elem1_EQPS
  # ABAQUS ELEMENT 1726607 IP COORD
  element element_id nearest location -57.4985E-03, -500.0E-06, -1.24983E-03 as Ta_Elem2_ID
  element temperature at element 1726607 as Ta_Elem2_Temp
  element von_mises at element 1726607 as Ta_Elem2_Mises
  element effective_log_strain at element 1726607 as Ta_Elem2_EffLE
  element yield_stress at element 1726607 as Ta_Elem2_YS
  element yield_flag at element 1726607 as Ta_Elem2_Yflag
  element eqps at element 1726607 as Ta_Elem2_EQPS
  # Note that these are ExodusII node INDECES
  #node displacement at node 213540 as Disp_Ta_213540
  #node temperature at node 213540 as Temp_Ta_213540
  #node displacement at node 1474864 as Disp_W_1474864
  #node temperature at node 1474864 as Temp_W_1474864
end

begin results output
  database name = %B_HIP_SS.e
  at time 0.0 increment = 0.001
  termination time = 0.01
  Nodal Variables = Velocity As V
  Nodal Variables = Displacement As U
  Nodal Variables = Temperature As NT11
  Element Variables = Stress
  Element Variables = Log_Strain
  Element Variables = Von_Mises
  Element Variables = max_principal_stress as MaxPrin
  Element Variables = Effective_Log_Strain As ELS
  Element Variables = eqps
  Nodal Variables = Status_Constraint_Flag as Status
end

begin results output
  database name = %B_W_Pulses.e
  start time = 0.01
  at time 0.01      increment = 1.0e-07
  at time 0.0100007 increment = 1.0e-06
  at time 0.0120007 increment = 0.001
  at time 0.0140007 increment = 1.0e-07
  at time 0.0140014 increment = 1.0e-06
  at time 0.0160014 increment = 0.001
  include = W_BLOCK_SURF
  Nodal Variables = Displacement As U
  Element Variables = max_principal_stress as MaxPrin
  Element Variables = Effective_Log_Strain As ELS
  Element Variables = Stress
end

begin results output
  database name = %B_Ta_Pulses.e
  start time = 0.01
  at time 0.01      increment = 1.0e-07
  at time 0.0100007 increment = 1.0e-06
  at time 0.0120007 increment = 0.001
  at time 0.0140007 increment = 1.0e-07
  at time 0.0140014 increment = 1.0e-06
  at time 0.0160014 increment = 0.001
  include = TA_CLAD
  Nodal Variables = Displacement As U
  Element Variables = Effective_Log_Strain As ELS
  Element Variables = Von_Mises
  Element Variables = EQPS
  Element Variables = Stress
end

begin tied mpc
```

```
        tied faces = W_BLOCK.W_TA
        tied nodes = TA_CLAD.TA_W
      end tied mpc

      begin initial condition
        include all blocks
        initialize variable name = temperature
        variable type = node
        magnitude = 30.0
      end

#      begin prescribed temperature
#        include all blocks
#        active periods = Step_1_HIP
#        function = SmoothTempHIP
#      end

      begin prescribed temperature
        include all blocks
        active periods = Step_2_SS Step_3_Pulses
        copy variable = NodalTempField from model TargetBlockTemps MAP_BY_ID
      end prescribed temperature

    end presto region TargetBlock

  end presto procedure HIP_SS

end sierra
```

8.5 SIERRA SIMULATION POSTPROCESSING SCRIPTS

8.5.1 ParaView PostProcessing Script – Ta1

```
# trace generated using paraview version 5.9.1


from paraview.simple import *
# disable automatic camera reset on 'Show'
paraview.simple._DisableFirstRenderCameraReset()




#--------------------------------------------------------------------
# LOAD DATA
#--------------------------------------------------------------------
job = ExodusIIReader(registrationName='Job_3_HIP_SS.e', FileName=['Job_3_HIP_SS.e'])
job.ElementVariables = []
job.PointVariables = []
job.GlobalVariables = []
job.NodeSetArrayStatus = []
job.SideSetArrayStatus = []

# get animation scene
animationScene1 = GetAnimationScene()

# update animation scene based on data timesteps
animationScene1.UpdateAnimationUsingDataTimeSteps()

# Properties modified on job
job.GenerateObjectIdCellArray = 0
job.GenerateGlobalElementIdArray = 0
job.ElementVariables = ['ELS', 'Von_Mises', 'eqps']
job.GenerateGlobalNodeIdArray = 0
job.PointVariables = ['NT11']
job.ElementBlocks = ['ta_clad']
job.FilePrefix = ''
job.FilePattern = ''




#--------------------------------------------------------------------
# SET DISPLAY SIZE & VIEW
#--------------------------------------------------------------------

# get active view
renderView1 = GetActiveViewOrCreate('RenderView')

# show data in view
jobDisplay = Show(job, renderView1, 'UnstructuredGridRepresentation')

# reset view to fit data
renderView1.ResetCamera()

# get the material library
materialLibrary1 = GetMaterialLibrary()

# update the view to ensure updated data information
renderView1.Update()

# get layout
layout1 = GetLayout()

# layout/tab size in pixels
layout1.SetSize(924, 1033)

# current camera placement for renderView1
renderView1.CameraPosition = [-0.4493305225375842, 0.2670909045539559, 0.2989763507381588]
```

```
renderView1.CameraFocalPoint = [0.060020272687684076, 0.030191248202313503, -0.08313984357630592]
renderView1.CameraViewUp = [0.26739968175979606, 0.937065191174243, -0.22451333520381206]
renderView1.CameraParallelScale = 0.17702867337271413

RenderAllViews()



#------------------------------------------------------------------
# PLOT HIP
#------------------------------------------------------------------

# get HIP time frame
animationScene1.GoToLast() # frame 10
animationScene1.GoToPrevious() # frame 9
animationScene1.GoToPrevious() # frame 8
animationScene1.GoToPrevious() # frame 7


#### PLOT

# set scalar coloring
#ColorBy(jobDisplay, ('POINTS', 'NT11'))
ColorBy(jobDisplay, ('CELLS', 'Von_Mises'))

# rescale color and/or opacity maps used to include current data range
jobDisplay.RescaleTransferFunctionToDataRange(True, False)

# show color bar/color legend
jobDisplay.SetScalarBarVisibility(renderView1, True)

# change color transfer function/color map
plotLUT = GetColorTransferFunction('Von_Mises')
plotLUT.ApplyPreset('Jet', True)

# get opacity transfer function/opacity map
plotPWF = GetOpacityTransferFunction('Von_Mises')

# Properties modified on renderView1
renderView1.Background = [1.0, 1.0, 1.0]

# get the material library
materialLibrary1 = GetMaterialLibrary()

# Properties modified on renderView1
renderView1.OrientationAxesLabelColor = [0.0, 0.0, 0.0]

# Properties modified on renderView1
renderView1.OrientationAxesOutlineColor = [0.0, 0.0, 0.0]

# get color legend/bar for maxPrinLUT in view renderView1
plotLUTColorBar = GetScalarBar(plotLUT, renderView1)

# Properties modified on maxPrinLUTColorBar
plotLUTColorBar.TitleColor = [0.0, 0.0, 0.0]
plotLUTColorBar.LabelColor = [0.0, 0.0, 0.0]

# rescale color and/or opacity maps used to exactly fit the current data range
jobDisplay.RescaleTransferFunctionToDataRange(False, True)

# save screenshot
SaveScreenshot('Sierra_Ta_Mises_HIP.png', renderView1, ImageResolution=[924, 1033])



#------------------------------------------------------------------
# PLOT SS
#------------------------------------------------------------------
```

```
#### CHOOSE TIME FRAME - SS
animationScene1.GoToLast()
Show(job, renderView1)

# rescale color and/or opacity maps used to exactly fit the current data range
jobDisplay.RescaleTransferFunctionToDataRange(False, True)

# save screenshot
SaveScreenshot('Sierra_Ta_Mises_SS.png', renderView1, ImageResolution=[924, 1033])
```

8.5.2 ParaView PostProcessing Script – Ta2

```
# trace generated using paraview version 5.9.1


from paraview.simple import *
# disable automatic camera reset on 'Show'
paraview.simple._DisableFirstRenderCameraReset()




#--------------------------------------------------------------------
# LOAD DATA
#--------------------------------------------------------------------
job = ExodusIIReader(registrationName='Job_3_HIP_SS.e', FileName=['Job_3_HIP_SS.e'])
job.ElementVariables = []
job.PointVariables = []
job.GlobalVariables = []
job.NodeSetArrayStatus = []
job.SideSetArrayStatus = []

# get animation scene
animationScene1 = GetAnimationScene()

# update animation scene based on data timesteps
animationScene1.UpdateAnimationUsingDataTimeSteps()

# Properties modified on job
job.GenerateObjectIdCellArray = 0
job.GenerateGlobalElementIdArray = 0
job.ElementVariables = ['ELS', 'Von_Mises', 'eqps']
job.GenerateGlobalNodeIdArray = 0
job.PointVariables = ['NT11']
job.ElementBlocks = ['ta_clad']
job.FilePrefix = ''
job.FilePattern = ''




#--------------------------------------------------------------------
# SET DISPLAY SIZE & VIEW
#--------------------------------------------------------------------

# get active view
renderView1 = GetActiveViewOrCreate('RenderView')

# show data in view
jobDisplay = Show(job, renderView1, 'UnstructuredGridRepresentation')

# reset view to fit data
renderView1.ResetCamera()

# get the material library
materialLibrary1 = GetMaterialLibrary()

# update the view to ensure updated data information
renderView1.Update()
```

```
# get layout
layout1 = GetLayout()

# layout/tab size in pixels
layout1.SetSize(924, 1033)

# current camera placement for renderView1
renderView1.CameraPosition = [-0.4493305225375842, 0.2670909045539559, 0.2989763507381588]
renderView1.CameraFocalPoint = [0.060020272687684076, 0.030191248202313503, -0.08313984357630592]
renderView1.CameraViewUp = [0.26739968175979606, 0.937065191174243, -0.22451333520381206]
renderView1.CameraParallelScale = 0.17702867337271413

RenderAllViews()



#-----------------------------------------------------------------------
# PLOT HIP
#-----------------------------------------------------------------------

# get HIP time frame
animationScene1.GoToLast() # frame 10
animationScene1.GoToPrevious() # frame 9
animationScene1.GoToPrevious() # frame 8
animationScene1.GoToPrevious() # frame 7


#### PLOT

# set scalar coloring
#ColorBy(jobDisplay, ('POINTS', 'NT11'))
ColorBy(jobDisplay, ('CELLS', 'eqps'))

# rescale color and/or opacity maps used to include current data range
jobDisplay.RescaleTransferFunctionToDataRange(True, False)

# show color bar/color legend
jobDisplay.SetScalarBarVisibility(renderView1, True)

# change color transfer function/color map
plotLUT = GetColorTransferFunction('eqps')
plotLUT.ApplyPreset('Jet', True)

# get opacity transfer function/opacity map
plotPWF = GetOpacityTransferFunction('eqps')

# Properties modified on renderView1
renderView1.Background = [1.0, 1.0, 1.0]

# get the material library
materialLibrary1 = GetMaterialLibrary()

# Properties modified on renderView1
renderView1.OrientationAxesLabelColor = [0.0, 0.0, 0.0]

# Properties modified on renderView1
renderView1.OrientationAxesOutlineColor = [0.0, 0.0, 0.0]

# get color legend/bar for maxPrinLUT in view renderView1
plotLUTColorBar = GetScalarBar(plotLUT, renderView1)

# Properties modified on maxPrinLUTColorBar
plotLUTColorBar.TitleColor = [0.0, 0.0, 0.0]
plotLUTColorBar.LabelColor = [0.0, 0.0, 0.0]

# rescale color and/or opacity maps used to exactly fit the current data range
jobDisplay.RescaleTransferFunctionToDataRange(False, True)

# save screenshot
SaveScreenshot('Sierra_Ta_EQPS_HIP.png', renderView1, ImageResolution=[924, 1033])
```

```
#--------------------------------------------------------------------
# PLOT SS
#--------------------------------------------------------------------

#### CHOOSE TIME FRAME - SS
animationScene1.GoToLast()
Show(job, renderView1)

# rescale color and/or opacity maps used to exactly fit the current data range
jobDisplay.RescaleTransferFunctionToDataRange(False, True)

# save screenshot
SaveScreenshot('Sierra_Ta_EQPS_SS.png', renderView1, ImageResolution=[924, 1033])
```

8.5.3 ParaView PostProcessing Script – W

```
# trace generated using paraview version 5.9.1


from paraview.simple import *
# disable automatic camera reset on 'Show'
paraview.simple._DisableFirstRenderCameraReset()




#--------------------------------------------------------------------
# LOAD DATA
#--------------------------------------------------------------------
job = ExodusIIReader(registrationName='Job_3_HIP_SS.e', FileName=['Job_3_HIP_SS.e'])
job.ElementVariables = []
job.PointVariables = []
job.GlobalVariables = []
job.NodeSetArrayStatus = []
job.SideSetArrayStatus = []

# get animation scene
animationScene1 = GetAnimationScene()

# update animation scene based on data timesteps
animationScene1.UpdateAnimationUsingDataTimeSteps()

# Properties modified on job
job.GenerateObjectIdCellArray = 0
job.GenerateGlobalElementIdArray = 0
job.ElementVariables = ['ELS', 'MaxPrin']
job.GenerateGlobalNodeIdArray = 0
job.PointVariables = ['NT11']
job.ElementBlocks = ['w_block']
job.FilePrefix = ''
job.FilePattern = ''




#--------------------------------------------------------------------
# SET DISPLAY SIZE & VIEW
#--------------------------------------------------------------------

# get active view
renderView1 = GetActiveViewOrCreate('RenderView')

# show data in view
jobDisplay = Show(job, renderView1, 'UnstructuredGridRepresentation')

# reset view to fit data
renderView1.ResetCamera()

# get the material library
materialLibrary1 = GetMaterialLibrary()

# update the view to ensure updated data information
renderView1.Update()

# get layout
layout1 = GetLayout()

# layout/tab size in pixels
layout1.SetSize(924, 1033)

# current camera placement for renderView1
renderView1.CameraPosition = [-0.4493305225375842, 0.2670909045539559, 0.2989763507381588]
renderView1.CameraFocalPoint = [0.060020272687684076, 0.030191248202313503, -0.08313984357630592]
renderView1.CameraViewUp = [0.26739968175979606, 0.937065191174243, -0.22451333520381206]
```

```
renderView1.CameraParallelScale = 0.17702867337271413

RenderAllViews()



#-----------------------------------------------------------------------
# PLOT HIP STRESS
#-----------------------------------------------------------------------

# get HIP time frame
animationScene1.GoToLast() # frame 10
animationScene1.GoToPrevious() # frame 9
animationScene1.GoToPrevious() # frame 8
animationScene1.GoToPrevious() # frame 7


#### STRESS PLOT

# set scalar coloring
#ColorBy(jobDisplay, ('POINTS', 'NT11'))
ColorBy(jobDisplay, ('CELLS', 'MaxPrin'))

# rescale color and/or opacity maps used to include current data range
jobDisplay.RescaleTransferFunctionToDataRange(True, False)

# show color bar/color legend
jobDisplay.SetScalarBarVisibility(renderView1, True)

# change color transfer function/color map
plotLUT = GetColorTransferFunction('MaxPrin')
plotLUT.ApplyPreset('Jet', True)

# get opacity transfer function/opacity map
plotPWF = GetOpacityTransferFunction('MaxPrin')

# Properties modified on renderView1
renderView1.Background = [1.0, 1.0, 1.0]

# get the material library
materialLibrary1 = GetMaterialLibrary()

# Properties modified on renderView1
renderView1.OrientationAxesLabelColor = [0.0, 0.0, 0.0]

# Properties modified on renderView1
renderView1.OrientationAxesOutlineColor = [0.0, 0.0, 0.0]

# get color legend/bar for maxPrinLUT in view renderView1
plotLUTColorBar = GetScalarBar(plotLUT, renderView1)

# Properties modified on maxPrinLUTColorBar
plotLUTColorBar.TitleColor = [0.0, 0.0, 0.0]
plotLUTColorBar.LabelColor = [0.0, 0.0, 0.0]

# rescale color and/or opacity maps used to exactly fit the current data range
jobDisplay.RescaleTransferFunctionToDataRange(False, True)

# save screenshot
SaveScreenshot('Sierra_W_MaxPrin_HIP.png', renderView1, ImageResolution=[924, 1033])


#### Y PLANE SLICE

# create a new 'Clip'
clip1 = Clip(registrationName='Clip1', Input=job)
clip1.ClipType = 'Plane'
clip1.HyperTreeGridClipper = 'Plane'

# init the 'Plane' selected for 'ClipType'
```

```
clip1.ClipType.Origin = [0.045296136289834976, 0.0, -0.08404913311824203]

# toggle 3D widget visibility (only when running from the GUI)
Hide3DWidgets(proxy=clip1.ClipType)

# Properties modified on clip1.ClipType
clip1.ClipType.Normal = [0.0, 1.0, 0.0]

# show data in view
clip1Display = Show(clip1, renderView1, 'UnstructuredGridRepresentation')

# hide clipped data
Hide(job, renderView1)

# update the view to ensure updated data information
renderView1.Update()

# save screenshot
SaveScreenshot('Sierra_W_MaxPrin_HIP_Clip.png', renderView1, ImageResolution=[924, 1033])

# delete clip
Delete(clip1)
del clip1



#--------------------------------------------------------------------
# PLOT SS STRESS
#--------------------------------------------------------------------

#### CHOOSE TIME FRAME - SS

animationScene1.GoToLast()

Show(job, renderView1)

# rescale color and/or opacity maps used to exactly fit the current data range
jobDisplay.RescaleTransferFunctionToDataRange(False, True)

# save screenshot
SaveScreenshot('Sierra_W_MaxPrin_SS.png', renderView1, ImageResolution=[924, 1033])


#### Y PLANE SLICE

# create a new 'Clip'
clip1 = Clip(registrationName='Clip1', Input=job)
clip1.ClipType = 'Plane'
clip1.HyperTreeGridClipper = 'Plane'

# init the 'Plane' selected for 'ClipType'
clip1.ClipType.Origin = [0.045296136289834976, 0.0, -0.08404913311824203]

# toggle 3D widget visibility (only when running from the GUI)
Hide3DWidgets(proxy=clip1.ClipType)

# Properties modified on clip1.ClipType
clip1.ClipType.Normal = [0.0, 1.0, 0.0]

# show data in view
clip1Display = Show(clip1, renderView1, 'UnstructuredGridRepresentation')

# hide clipped data
Hide(job, renderView1)

# update the view to ensure updated data information
renderView1.Update()

# save screenshot
SaveScreenshot('Sierra_W_MaxPrin_SS_Clip.png', renderView1, ImageResolution=[924, 1033])
```

8.5.4 Python PostProcessing Script

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug 25 15:09:36 2021

@author: tvj

Post-processes Sierra Heartbeat output file and
plots comparisons against Abaqus

"""

import numpy as np
import matplotlib.pyplot as plt

f = np.genfromtxt('Job_3_HIP_SS_Pulses.hrt', dtype=float, delimiter=",", names=True)

abq = np.genfromtxt('Abaqus_STS_ST_Hex.csv', dtype=float, delimiter=",", names=True,
skip_header=3)

#f.dtype
#abq.dtype

#
# PLOT
#
fig1, ax1 = plt.subplots(figsize=(6,6), dpi=200)

ax1.plot(f['TIME']*1.E3,
         f['SE'],
         label="Strain")

ax1.plot(f['TIME']*1.E3,
         f['KE'],
         label="Kinetic")

ax1.plot(f['TIME']*1.E3,
         f['HGE'],
         label="Hourglass")

ax1.set_xlabel('Time [ms]')
ax1.set_ylabel('Rel. Energy')


ax1.set_title('Sierra HIP+SS+Pulses: Energy History')
plt.legend(loc='center right')
plt.grid(which='major', axis='both')

fig1.savefig('Sierra_STS_ST_Hex_fig1.png', bbox_inches = "tight")


#
# PLOT
#
fig2, ax2 = plt.subplots(figsize=(6,6), dpi=200)

ax2.plot(f['TIME']*1.E3,
         f['Ta_Elem1_Temp'],
         label='Ta_E'+f['Ta_Elem1_ID'][0].astype(int).astype(str)
         )

ax2.plot(f['TIME']*1.E3,
         f['Ta_Elem2_Temp'],
         label='Ta_E'+f['Ta_Elem2_ID'][0].astype(int).astype(str)
         )

ax2.plot(f['TIME']*1.E3,
         f['W_Elem_Temp'],
         label='W_E'+f['W_Elem_ID'][0].astype(int).astype(str)
```

```
        )

ax2.set_xlabel('Time [ms]')
ax2.set_ylabel('Temperature [C]')


ax2.set_title('Sierra HIP+SS+Pulses: Temperature History')
plt.legend(loc='center right')
plt.grid(which='major', axis='both')

fig2.savefig('Sierra_STS_ST_Hex_fig2.png', bbox_inches = "tight")


#
# PLOT
#
nQuasi = np.searchsorted(f['TIME'],0.01)

fig3, ax3 = plt.subplots(figsize=(6,6), dpi=200)

ax3.plot(f['TIME'][:nQuasi]*1.E3,
         f['QuasiRES'][:nQuasi],
         label='Quasistatic Residual'
         )

ax3.plot(f['TIME'][:nQuasi]*1.E3,
         f['KE'][:nQuasi],
         label='Kinetic Energy'
         )

ax3.plot(f['TIME'][:nQuasi]*1.E3,
         f['Ta_Elem1_Yflag'][:nQuasi],
         label='E'+f['Ta_Elem1_ID'][0].astype(int).astype(str)+' Yield?'
         )

ax3.plot(f['TIME'][:nQuasi]*1.E3,
         f['Ta_Elem2_Yflag'][:nQuasi],
         label='E'+f['Ta_Elem2_ID'][0].astype(int).astype(str)+' Yield?'
         )

ax3.set_xlabel('Time [ms]')

ax3.set_title('Sierra HIP+SS: Quasistatic')
plt.legend(loc='center right')
plt.grid(which='major', axis='both')

fig3.savefig('Sierra_STS_ST_Hex_fig3.png', bbox_inches = "tight")


#
# PLOT
#
fig4, ax4 = plt.subplots(figsize=(6,6), dpi=200)

ax4.plot(f['W_Elem_EffLE']*1.E2,
         f['W_Elem_MaxPrin']/1.E6,
         label='W_E'+f['W_Elem_ID'][0].astype(int).astype(str)
         )

ax4.set_xlabel('Effective Log Strain [%]')
ax4.set_ylabel('Maximum Principal Stress [MPa]')

ax4.set_title('Sierra HIP+SS+Pulses: Tungsten Surface')
plt.legend(loc='center right')
plt.grid(which='major', axis='both')

fig4.savefig('Sierra_STS_ST_Hex_fig4.png', bbox_inches = "tight")


#
# PLOT
```

```
#
fig5, ax5 = plt.subplots(figsize=(6,6), dpi=200)

ax5.plot(f['Ta_Elem1_EffLE']*1.E2,
         f['Ta_Elem1_Mises']/1.E6,
         label='E'+f['Ta_Elem1_ID'][0].astype(int).astype(str)
         )

ax5.plot(f['Ta_Elem1_EffLE']*1.E2,
         f['Ta_Elem1_YS']/1.E6,
         label='Yield Stress'
         )

ax5.plot(abq['E1678264_E'],
         abq['E1678264_S'],
         label='Abaqus',
         linestyle=':'
         )

ax5.set_xlabel('Effective Log Strain [%]')
ax5.set_ylabel('Effective Stress [MPa]')

ax5.set_title('Sierra HIP+SS+Pulses: Tantalum')
plt.legend(loc='center right')
plt.grid(which='major', axis='both')

fig5.savefig('Sierra_STS_ST_Hex_fig5.png', bbox_inches = "tight")


#
# PLOT
#
fig6, ax6 = plt.subplots(figsize=(6,6), dpi=200)

ax6.plot(f['Ta_Elem2_EffLE']*1.E2,
         f['Ta_Elem2_Mises']/1.E6,
         label='E'+f['Ta_Elem2_ID'][0].astype(int).astype(str)
         )

ax6.plot(f['Ta_Elem2_EffLE']*1.E2,
         f['Ta_Elem2_YS']/1.E6,
         label='Yield Stress'
         )

ax6.plot(abq['E1726607_E'],
         abq['E1726607_S'],
         label='Abaqus',
         linestyle=':'
         )

ax6.set_xlabel('Effective Log Strain [%]')
ax6.set_ylabel('Effective Stress [MPa]')

ax6.set_title('Sierra HIP+SS+Pulses: Tantalum')
plt.legend(loc='lower right')
plt.grid(which='major', axis='both')

fig6.savefig('Sierra_STS_ST_Hex_fig6.png', bbox_inches = "tight")
```

8.6 PYTHON SIERRA EXPORT SCRIPT

```
"""
sierraExport.py

Post-processes Sierra output file to save elemental stress tensors
at all available time steps.  Saves results in binary numpy format.

Built by Joseph B. Tipton, Jr. from original concept by Lianshan Lin (ORNL).
"""

import numpy as np
from exodus import exodus, copyTransfer
import exodusCalcs


#read stress data from sierra output .e file
filename = 'Sierra_STS_ST_Hex_Irrad_W_Pulses'
ei = exodus(filename+'.e',mode='r',array_type='numpy')


# exodus block dictionary
blkIDs = np.array(ei.get_elem_blk_ids())   # block integer ids
blkNames = np.array(ei.get_elem_blk_names()) # block names
blkDict = {blkNames[i]: blkIDs[i] for i in range(len(blkIDs))}


#element IDs
eleIDs = np.transpose(ei.get_elem_num_map())


# Lists the times saved in seconds
# The exodus time_step index starts at 1 (not zero)
timeArray = ei.get_times()
numTimes = len(timeArray)


# Preallocate master array
numElem = ei.num_elems()
outData = np.empty((numTimes,numElem,6))


for jj in range(numTimes):
  # assumes output file has only 1 block and selects that ID
  ii = blkIDs[0][0]
  # read the stress components into array
  #                                (elem_blk_id, evar_name, time_step)
  evar_xx = ei.get_element_variable_values(ii, 'Stress_xx', jj+1)
  evar_yy = ei.get_element_variable_values(ii, 'Stress_yy', jj+1)
  evar_zz = ei.get_element_variable_values(ii, 'Stress_zz', jj+1)
  evar_xy = ei.get_element_variable_values(ii, 'Stress_xy', jj+1)
  evar_zx = ei.get_element_variable_values(ii, 'Stress_zx', jj+1)
  evar_yz = ei.get_element_variable_values(ii, 'Stress_yz', jj+1)

  # construct stress array
  evar_all=(np.array([evar_xx, evar_yy, evar_zz, evar_xy, evar_zx, evar_yz])).T

  # add to output array
  outData[jj] = evar_all


ei.close()

# save data
with open(filename+'.npy', 'wb') as f:
  np.save(f, eleIDs)
  np.save(f, blkDict)
  np.save(f, timeArray)
  np.save(f, outData)
```

8.7 ABAQUS PYTHON IMPORT SCRIPT

8.7.1 Torque Job Scheduler Script

```
#!/bin/sh
#PBS -V
#PBS -j oe
#PBS -m abe
#PBS -l nodes=1:ppn=1                          # specify # of nodes
#PBS -N sierra2ODB                             # specify the job name
#PBS -M UID@ornl.gov                           # specify your email address
#PBS -q all                                    # specify the queue: all / slow
# #PBS -W depend=afterany:6923

cd $PBS_O_WORKDIR

# Define particulars of this run:
INPUT_FILENAME=sierra2ODB.py
JOBNAME=sierra2ODB

# Number of processors
NPROCS=`wc -l < $PBS_NODEFILE`
echo This job has allocated $NPROCS processors


# Executable for Abaqus
EXEC=/usr/SIMULIA/EstProducts/2020/linux_a64/code/bin/ABQLauncher  #2020


#
# To manage abaqus jobs, you need to catch signals
# and use "abaqus terminate to stop the job
#
exit_gracefully () {
$EXEC terminate job=$JOBNAME
echo Abaqus job $JOBNAME terminated
exit
}


#
# Invoke abaqus in the background on the compute node
#
trap exit_gracefully SIGUSR2
$EXEC python $INPUT_FILENAME


#
# Sleep until lock file disappears
#
sleep 60
while [ -f ${JOBNAME}.lck ]; do
  sleep 5
done


exit
```

8.7.2 Python Script

```
"""
sierra2ODB.py

Depends upon sierraExport.py script which saves Sierra simulation stress
tensors in numpy binary format.  This program loads this data along with
an Abaqus ODB that already contains the corresponding mesh.  It then writes
the data along with time steps to the ODB.

Built by Joseph B. Tipton, Jr. from original concept by Lianshan Lin (ORNL).

"""

from odbAccess import *
from abaqusConstants import *
import numpy as np


#
# LOAD DATA FROM SIERRA
#
filename = 'Sierra_STS_ST_Hex_Ta_Pulses'

with open(filename+'.npy', 'rb') as f:
  eleIDs = np.load(f, allow_pickle=True).tolist()
  blkDict = np.load(f, allow_pickle=True)
  timeArray = np.load(f, allow_pickle=True)
  eleStress = np.load(f, allow_pickle=True)


#
# ACCESS ODB and ADD STEP with FRAMES
#
odb = openOdb(filename+'.odb',readOnly=False)
allInstances = odb.rootAssembly.instances.keys()
odbInstance = odb.rootAssembly.instances[allInstances[-1]]

if odb.steps.keys()[-1] != 'Sierra':
  #
  # The data transfer has not yet started,
  # so, make the step and start the transfer
  #
  step1 = odb.Step(name='Sierra',
                   description='Sierra pulse results',
                   domain=TIME,
                   timePeriod=timeArray[-1])

  for ii in range(len(timeArray)):
    print('Starting frame ',ii)
    Sframe = step1.Frame(incrementNumber=ii,
                         frameValue=timeArray[ii])
    Sfield = Sframe.FieldOutput(name='S',
                                description='Stress',
                                type=TENSOR_3D_FULL,
                                componentLabels=('S11', 'S22', 'S33', 'S12', 'S13', 'S23'),
                                validInvariants=(MISES,TRESCA,MAX_PRINCIPAL))
    Sfield.addData(position=INTEGRATION_POINT,
                   instance=odbInstance,
                   labels=eleIDs,
                   data=eleStress[ii].tolist())
    # save every 1000th time step
    if (ii % 1000 == 0):
      odb.save()
      # gc.collect(generation=2) doesn't really help
else:
  #
  # The data transfer previously started but might
  # have ended prematurely due to a memory error.
  # Pickup where we left off...
```

```
  #
  step1 = odb.steps['Sierra']
  for ii in range(len(step1.frames),len(timeArray)):
    print('Starting frame ',ii)
    Sframe = step1.Frame(incrementNumber=ii,
                         frameValue=timeArray[ii])
    Sfield = Sframe.FieldOutput(name='S',
                                description='Stress',
                                type=TENSOR_3D_FULL,
                                componentLabels=('S11', 'S22', 'S33', 'S12', 'S13', 'S23'),
                                validInvariants=(MISES,TRESCA,MAX_PRINCIPAL))
    Sfield.addData(position=INTEGRATION_POINT,
                   instance=odbInstance,
                   labels=eleIDs,
                   data=eleStress[ii].tolist())
    # save every 1000th time step
    if (ii % 1000 == 0):
      odb.save()
      # gc.collect(generation=2) doesn't really help


odb.save()
odb.close()
```

8.8 FESAFE FATIGUE ANALYSIS

8.8.1 Torque Job Scheduler Script (FESAFE_run.sh)

```
#!/bin/sh
#PBS -V
#PBS -j oe
#PBS -m abe
#PBS -l nodes=1:ppn=8
#PBS -N FESAFE_Ta
#PBS -M UID@ornl.gov
#PBS -q all
# #PBS -W depend=afterany:9408

cd $PBS_O_WORKDIR


# Define particulars of this run:
INPUT_FILENAME=/data2/home/tvj/Documents/7_SierraSM/6_FESAFE/1_Unirrad/FESAFE/Sierra_Fatigue.macro
JOBNAME=Sierra_Fatigue


#executable for fe-safe
EXEC=/usr/SIMULIA/EstProducts/2021/fe-safe_cl


#
# To manage fe-safe jobs, you need to catch signals
# and use "pkill -9" to kill the fe-safe job
#

exit_gracefully () {
pkill -9 fe-safe_cl
echo fe-safe job $JOBNAME terminated
exit
}


# invoke fe-safe in the background on the compute node:
trap exit_gracefully SIGUSR2
$EXEC macro=$INPUT_FILENAME >FESAFE_run.log
```

8.8.2 FE-SAFE Macro (Sierra_Fatigue.macro)

```
#
# ACTIVE DIRECTORY
# (can use relative links from here)
```

```
#
SwitchToProject /data2/home/tvj/Documents/7_SierraSM/6_FESAFE/1_Unirrad/FESAFE

# PRE-SCAN PRE-SCAN COMMANDS
pre-scan files ../Sierra_STS_ST_Hex_Ta_Pulses.odb

# PRE-SCAN POSITION COMMAND
pre-scan position elemental
pre-scan deselect all

# PRE-SCAN OPTIONS
pre-scan select groups
pre-scan select detect-surface

# PRE-SCAN SELECT COMMANDS
pre-scan select step "*Sierra*" stress

# PRE-SCAN OPEN COMMAND
pre-scan open selected

# GROUPS COMMANDS
groups list deselect all

# GROUPS COMMANDS
groups list select "PART-DEFAULT_1_TA_CLAD"

# RUN FE-SAFE ANALYSIS
fe-safe b=Sierra_Fatigue_Ta.stlx
```

8.8.3 FE-SAFE Analysis File

8.8.3.1 Unirradiated Condition (Sierra_Fatigue_Ta.stlx)

```
<!DOCTYPE settings>
<root>
    <!--fe-safe Settings File-->
    <meta version="3"/>
    <data>
        <group name="project">
            <group name="interfaces">
                <boolean name="load all groups">true</boolean>
                <boolean name="add surface groups">true</boolean>
            </group>
            <directory name="generated results directory">./results</directory>
            <integer name="diagnostic level">0</integer>
            <group name="model">
                <file name="mesh source file">../Sierra_STS_ST_Hex_Unirrad_Ta_Pulses.odb</file>
                <file name="source file">../Sierra_STS_ST_Hex_Unirrad_Ta_Pulses.odb</file>
                <group name="stress units">
                    <string name="description">Pa</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <group name="strain units">
                    <string name="description">strain</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <group name="temperature units">
                    <string name="description">degC</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <group name="force units">
                    <string name="description">N</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <group name="distance units">
```

```
                    <string name="description">m</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <boolean name="surface as nodal">true</boolean>
                <enumerator name="extract strain type">Total</enumerator>
                <boolean name="extract forces">false</boolean>
                <boolean name="extract strains">false</boolean>
                <boolean name="extract groups">true</boolean>
                <string name="debug items">None</string>
            </group>
        <group name="job">
            <enumerator name="type">general</enumerator>
            <group name="material databases">
                <array name="materials" size="2">
                    <entry index="1">
                        <group name="material">
                            <file name="database">/data2/home/tvj/fe-safe.2020/local.dbase</file>
                            <string name="fatigue strength coefficient (sf')">346.066</string>
                            <string name="Young's modulus">188000</string>
                            <string name="ultimate tensile strength">283</string>
                            <string name="CAEL">4E8</string>
                            <string name="material name">Tantalum</string>
                            <string name="temperature list">0</string>
                            <group name="stress-life curve">
                                <string name="control series">289.95, 271.5266</string>
                                <string name="sample series">927490, 1.96488322E8</string>
                            </group>
                            <group name="torsion-life curve">
                                <string name="control series">0</string>
                            <string name="ultimate compressive strength">283</string>
                            <string name="Poisson's ratio">0.35</string>
                            <string name="algorithm">(shear+direct)Stress:-Goodman</string>
                            <string name="class">Other</string>
                            <string name="units">Use system default</string>
                            <string name="quality">Caution_Approximate!!</string>
                            <string name="comment 1">Properties calculated from Papakyriacou et
al. (2001)</string>
                            <string name="revision number">20</string>
                            <string name="revision date">Thu Jul 22 08:30:22 2021</string>
                            <string name="default
MSC">"/SIMULIA/EstProducts/2020/win_b64/Durability_resources/data/goodman.msc"</string>
                            <string name="Convert SN To TN">1</string>
                        </group>
                    </entry>
                </array>
                <boolean name="nodal property mapping enabled">true</boolean>
                <boolean name="stress-life table enabled">true</boolean>
                <boolean name="abort if no stress life data">true</boolean>
                <boolean name="correct elastic stress">false</boolean>
                <enumerator name="temperature interpolation">linear</enumerator>
                <boolean name="temperature clamp warning">false</boolean>
                <boolean name="material extrapolation warning">false</boolean>
                <boolean name="R-ratio clamp warning">false</boolean>
                <real name="Downing coefficient">2.55</real>
                <real name="Downing exponent">-0.8</real>
            </group>
            <array name="groups" size="2">
                <entry index="0">
                    <group name="group">
                        <string name="number">0</string>
                        <string name="group name">Default</string>
                        <string name="display name">Default</string>
                        <string name="algorithm">NoAnalysis</string>
                        <string name="mean stress correction">Unknown MSC</string>
                        <integer name="subgroup">1</integer>
                        <file name="user MSC file"/>
                        <string name="frf life">Infinite</string>
                    </group>
                </entry>
                <entry index="1">
```

```
                    <group name="group">
                        <string name="number">1</string>
                        <string name="group name">PART-DEFAULT_1_TA_CLAD</string>
                        <string name="display name">PART-DEFAULT_1_TA_CLAD</string>
                        <string name="algorithm">(shear+direct)Stress</string>
                        <string name="mean stress correction">Goodman</string>
                        <integer name="subgroup">0</integer>
                        <file name="user MSC file"/>
                        <string name="frf life">Infinite</string>
                    </group>
                </entry>
            </array>

                <group name="gating">
                    <real name="tensor gate">5</real>
                    <real name="load history gate">5</real>
                    <real name="near zero stress">10</real>
                    <real name="near zero strain">10</real>
                    <boolean name="tensor gate enabled">true</boolean>
                    <boolean name="load history gate enabled">false</boolean>
                    <boolean name="CAEL gate enabled">true</boolean>
                    <boolean name="trigonometric look-up tables">true</boolean>
                </group>
                <group name="probability">
                    <string name="target lives">1000</string>
                </group>
                <group name="exports">
                    <group name="results">
                        <file name="FER file">./fesafe.fer</file>
                        <integer name="contour policy">0</integer>
                    </group>
                    <boolean name="logarithmic lives">true</boolean>
                    <group name="plots">
                        <group name="worst node">
                            <boolean name="Haigh critical plane">true</boolean>
                            <boolean name="Smith critical plane">false</boolean>
                            <boolean name="Von Mises stress">false</boolean>
                            <boolean name="ignore overflows">true</boolean>
                        </group>
                        <group name="worst cycle">
                            <boolean name="Haigh">true</boolean>
                            <boolean name="Smith">false</boolean>
                        </group>
                        <boolean name="Haigh">false</boolean>
                        <boolean name="Smith">false</boolean>
                        <boolean name="Von Mises stress">false</boolean>
                        <boolean name="stress tensors">false</boolean>
                        <boolean name="stress tensors after gating">false</boolean>
                        <boolean name="principals">false</boolean>
                        <boolean name="critical plane normals">false</boolean>
                        <boolean name="plasticity corrected critical plane
normals">false</boolean>
                        <boolean name="all plane normals">false</boolean>
                        <boolean name="Dang Van">false</boolean>
                        <boolean name="damage">false</boolean>
                        <boolean name="TURBOlife">false</boolean>
                        <boolean name="modal">false</boolean>
                        <boolean name="critical distance plots">false</boolean>
                        <boolean name="critical distance tensor histories">false</boolean>
                        <boolean name="PSD Frequency Response Function plots">false</boolean>
                    </group>
                    <group name="contours">
                        <boolean name="life">true</boolean>
                        <boolean name="damage">false</boolean>
                        <boolean name="block damage">false</boolean>
                        <boolean name="FRF horizontal">false</boolean>
                        <boolean name="FRF vertical">false</boolean>
                        <boolean name="FRF radial">false</boolean>
                        <boolean name="FRF worst">false</boolean>
                        <boolean name="largest loading stress">true</boolean>
                        <boolean name="SMAXYS">false</boolean>
```

```
                        <boolean name="SMAXUTS">false</boolean>
                        <boolean name="SM">false</boolean>
                        <boolean name="SMPreNeuber">true</boolean>
                        <boolean name="SRP nodal temperatures">false</boolean>
                        <boolean name="critical planes all">false</boolean>
                        <boolean name="critical planes worst">false</boolean>
                        <boolean name="critical distance success">false</boolean>
                        <boolean name="critical distance diagnostics">false</boolean>
                        <boolean name="critical distance cycle">false</boolean>
                        <boolean name="turbolife accumulated stress strain">false</boolean>
                        <boolean name="NASA life2">false</boolean>
                        <boolean name="temperature-dependent UTS and FS">false</boolean>
                        <boolean name="maximum temperature">false</boolean>
                    </group>
                    <group name="logs">
                        <boolean name="load_sensitivity">false</boolean>
                        <boolean name="worst n lives">false</boolean>
                        <integer name="n">100</integer>
                        <boolean name="ranked elimination table">false</boolean>
                        <boolean name="nodal information">false</boolean>
                        <boolean name="block life table">false</boolean>
                        <boolean name="plane life table">false</boolean>
                        <boolean name="critical plane life table">false</boolean>
                        <boolean name="all planes life table">false</boolean>
                        <boolean name="fos planes">false</boolean>
                        <boolean name="grey iron damage">false</boolean>
                        <boolean name="dataset stresses">false</boolean>
                        <boolean name="ep residuals">false</boolean>
                        <boolean name="stress and strain tensors">false</boolean>
                        <boolean name="principals">false</boolean>
                        <boolean name="TURBOlife">false</boolean>
                        <boolean name="critical distance items">false</boolean>
                        <boolean name="critical distance summary">false</boolean>
                        <boolean name="PSD items">false</boolean>
                        <enumerator name="history indexing">One</enumerator>
                    </group>
                    <boolean name="only analyse listed items">false</boolean>
                    <boolean name="traffic lights">false</boolean>
                    <real name="traffic lights lower">1e+06</real>
                    <real name="traffic lights upper">1e+07</real>
                    <enumerator name="unit system">Metric</enumerator>
                </group>
                <group name="fos">
                    <real name="design life">1.3e+08</real>
                    <boolean name="infinite design life">false</boolean>
                    <real name="maximum">5</real>
                    <real name="fine maximum">1.5</real>
                    <real name="fine minimum">0.8</real>
                    <real name="minimum">0.5</real>
                    <integer name="maximum coarse iterations">20</integer>
                    <integer name="maximum fine iterations">16</integer>
                </group>
                <group name="critical distance">
                    <boolean name="apply corrections">false</boolean>
                    <enumerator name="method">point</enumerator>
                    <boolean name="use safety-factor thresholds">true</boolean>
                    <real name="safety-factor ceiling">10</real>
                    <real name="safety-factor floor">0</real>
                    <boolean name="allow quadratic interpolation">true</boolean>
                </group>
                <group name="welds">
                    <real name="proportional variance threshold">0.8</real>
                    <boolean name="use modified wang_brown">true</boolean>
                    <boolean name="use nonlinear shear correction">true</boolean>
                    <enumerator name="structural stress calculation mode">Basic Nodal
Force</enumerator>
                    <boolean name="automatically bound plate thickness">false</boolean>
                    <real name="plate bound median margin">0.25</real>
                    <boolean name="split incomplete toes">false</boolean>
                    <real name="min line length post split">0.75</real>
                    <boolean name="reject no through nodes">false</boolean>
```

```
                    <boolean name="auto-detect toe failures">true</boolean>
                    <boolean name="auto-detect fusion failures">true</boolean>
                    <boolean name="auto-detect throat failures">false</boolean>
                    <boolean name="auto-detect root failures">false</boolean>
                    <boolean name="use compact contour mode">true</boolean>
                    <real name="co-planarity tolerance">5</real>
                    <real name="cone tolerance">0.05</real>
                    <enumerator name="Ir function">IrDisplacement</enumerator>
                    <integer name="smoothing iterations">0</integer>
                    <enumerator name="weld end correction">VirtualNode</enumerator>
                    <enumerator name="throat/fusion failure sense">FromSurface</enumerator>
                    <real name="wrap angle">90</real>
                    <real name="blunt root threshold">0.25</real>
                </group>
                <group name="planes">
                    <integer name="critical plane search count">18</integer>
                </group>
                <group name="psd">
                    <enumerator name="PSD response">Von Mises</enumerator>
                    <boolean name="Apply nodal filtering">false</boolean>
                    <integer name="Number of stress range intervals">1000</integer>
                    <real name="RMS stress cut-off multiple">10</real>
                    <real name="Combined shear-normal K">0.25</real>
                    <boolean name="Bound Dirlik at UTS">false</boolean>
                    <enumerator name="Weld SS response">Normal</enumerator>
                </group>
                <file name="loading file">./Sierra_Fatigue.ldf</file>
                <file name="output file">./Sierra_STS_ST_Hex_unirrad_Ta_Pulses_FESAFE.odb</file>
                <string name="loading mode">LDF</string>
                <real name="scale factor">1</real>
                <real name="overflow value">0</real>
                <real name="infinite life value">-1</real>
                <integer name="modal samples per cycle">10</integer>
                <real name="modal gate percentage">0.1</real>
                <boolean name="disable temperature analysis">false</boolean>
                <string name="life units description">Repeats</string>
                <real name="life units factor">1</real>
                <boolean name="disable triaxiality">false</boolean>
                <boolean name="disable failed directional cosine to XYZ">false</boolean>
                <boolean name="Von Mises signed">false</boolean>
                <boolean name="ignore compressive cycles">false</boolean>
                <boolean name="enforce thread affinity">false</boolean>
                <integer name="max overflows">1000</integer>
            </group>
        </group>
    </data>
</root>
```

8.8.3.2 Irradiated Condition (Sierra_Fatigue_W.stlx)

```
<!DOCTYPE settings>
<root>
    <!--fe-safe Settings File-->
    <meta version="3"/>
    <data>
        <group name="project">
            <group name="interfaces">
                <boolean name="load all groups">true</boolean>
                <boolean name="add surface groups">true</boolean>
            </group>
            <directory name="generated results directory">./results</directory>
            <integer name="diagnostic level">0</integer>
            <group name="model">
                <file name="mesh source file">../Sierra_STS_ST_Hex_Irrad_W_Pulses.odb</file>
                <file name="source file">../Sierra_STS_ST_Hex_Irrad_W_Pulses.odb</file>
                <group name="stress units">
                    <string name="description">Pa</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
```

```
                <group name="strain units">
                    <string name="description">strain</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <group name="temperature units">
                    <string name="description">degC</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <group name="force units">
                    <string name="description">N</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <group name="distance units">
                    <string name="description">m</string>
                    <real name="scale">1</real>
                    <real name="offset">0</real>
                </group>
                <boolean name="surface as nodal">true</boolean>
                <enumerator name="extract strain type">Total</enumerator>
                <boolean name="extract forces">false</boolean>
                <boolean name="extract strains">false</boolean>
                <boolean name="extract groups">true</boolean>
                <string name="debug items">None</string>
            </group>
            <group name="job">
                <enumerator name="type">general</enumerator>
                <group name="material databases">
                    <array name="materials" size="2">
                        <entry index="1">
                            <group name="material">
                                <file name="database">C:/Users/tvj/Documents/fe-
safe.2020/local.dbase</file>
                                <string name="fatigue strength coefficient (sf')">474</string>
                                <string name="fatigue ductility coefficient (Ef')">0</string>
                                <string name="fatigue strength exponent (b)">-0.06037</string>
                                <string name="fatigue ductility exponent (c)">0</string>
                                <string name="K'">0</string>
                                <string name="Young's modulus">398000</string>
                                <string name="ultimate tensile strength">300</string>
                                <string name="n'">0</string>
                                <string name="CAEL">1E11</string>
                                <string name="material name">TungstenIrradiated</string>
                                <string name="temperature list">0</string>
                                <group name="stress-life curve">
                                    <string name="control series">0</string>
                                    <string name="sample series">0</string>
                                </group>
                                <group name="torsion-life curve">
                                    <string name="control series">0</string>
                                </group>
                                <string name="Poisson's ratio">0.28</string>
                                <string name="algorithm">NormalStress:-Goodman</string>
                                <string name="class">Other</string>
                                <string name="units">Use system default</string>
                                <string name="quality">Caution_Approximate!!</string>
                                <string name="comment 1">Parameters calculated using Haibany and
approximated for irradiated tungsten</string>
                                <string name="revision number">97</string>
                                <string name="revision date">Thu Oct 22 16:13:11 2020</string>
                                <string name="default
MSC">"C:\SIMULIA\EstProducts\2020\win_b64\Durability_resources\data\goodman.msc"</string>
                                <string name="grey iron index">None</string>
                                <string name="Convert SN To TN">1</string>
                            </group>
                        </entry>
                    </array>
                    <boolean name="nodal property mapping enabled">true</boolean>
                    <boolean name="material diagnostics">true</boolean>
```

```
    <boolean name="stress-life table enabled">false</boolean>
    <boolean name="abort if no stress life data">true</boolean>
    <boolean name="correct elastic stress">false</boolean>
    <enumerator name="temperature interpolation">linear</enumerator>
    <boolean name="temperature clamp warning">false</boolean>
    <boolean name="material extrapolation warning">false</boolean>
    <boolean name="R-ratio clamp warning">false</boolean>
    <real name="Downing coefficient">2.55</real>
    <real name="Downing exponent">-0.8</real>
</group>
<array name="groups" size="2">
    <entry index="0">
        <group name="group">
            <string name="number">0</string>
            <string name="group name">Default</string>
            <string name="display name">Default</string>
            <string name="algorithm">NoAnalysis</string>
            <string name="mean stress correction">Unknown MSC</string>
            <integer name="subgroup">1</integer>
            <file name="user MSC file"/>
            <string name="frf life">Infinite</string>
        </group>
    </entry>
    <entry index="1">
        <group name="group">
            <string name="number">1</string>
            <string name="group name">PART-DEFAULT_1_W_BLOCK_SURF</string>
            <string name="display name">PART-DEFAULT_1_W_BLOCK_SURF</string>
            <string name="algorithm">NormalStress</string>
            <string name="mean stress correction">Goodman</string>
            <integer name="subgroup">0</integer>
            <file name="user MSC file"/>
            <string name="frf life">Infinite</string>
        </group>
    </entry>
</array>
<group name="gating">
    <real name="tensor gate">5</real>
    <real name="load history gate">5</real>
    <real name="near zero stress">10</real>
    <real name="near zero strain">10</real>
    <boolean name="tensor gate enabled">true</boolean>
    <boolean name="load history gate enabled">false</boolean>
    <boolean name="CAEL gate enabled">true</boolean>
    <boolean name="trigonometric look-up tables">true</boolean>
</group>
<group name="probability">
    <string name="target lives">1000</string>
</group>
<group name="exports">
    <group name="results">
        <file name="FER file">./fesafe.fer</file>
        <integer name="contour policy">0</integer>
    </group>
    <boolean name="logarithmic lives">true</boolean>
    <group name="plots">
        <group name="worst node">
            <boolean name="Haigh critical plane">true</boolean>
            <boolean name="Smith critical plane">false</boolean>
            <boolean name="Von Mises stress">false</boolean>
            <boolean name="ignore overflows">true</boolean>
        </group>
        <group name="worst cycle">
            <boolean name="Haigh">true</boolean>
            <boolean name="Smith">false</boolean>
        </group>
        <boolean name="Haigh">false</boolean>
        <boolean name="Smith">false</boolean>
        <boolean name="Von Mises stress">false</boolean>
        <boolean name="stress tensors">false</boolean>
        <boolean name="stress tensors after gating">false</boolean>
```

```
                        <boolean name="principals">false</boolean>
                        <boolean name="critical plane normals">false</boolean>
                        <boolean name="plasticity corrected critical plane
normals">false</boolean>
                        <boolean name="all plane normals">false</boolean>
                        <boolean name="Dang Van">false</boolean>
                        <boolean name="damage">false</boolean>
                        <boolean name="TURBOlife">false</boolean>
                        <boolean name="modal">false</boolean>
                        <boolean name="critical distance plots">false</boolean>
                        <boolean name="critical distance tensor histories">false</boolean>
                        <boolean name="PSD Frequency Response Function plots">false</boolean>
                    </group>
                    <group name="contours">
                        <boolean name="life">true</boolean>
                        <boolean name="damage">false</boolean>
                        <boolean name="block damage">false</boolean>
                        <boolean name="FRF horizontal">false</boolean>
                        <boolean name="FRF vertical">false</boolean>
                        <boolean name="FRF radial">false</boolean>
                        <boolean name="FRF worst">false</boolean>
                        <boolean name="largest loading stress">true</boolean>
                        <boolean name="SMAXYS">false</boolean>
                        <boolean name="SMAXUTS">false</boolean>
                        <boolean name="SM">true</boolean>
                        <boolean name="SMPreNeuber">false</boolean>
                        <boolean name="SRP nodal temperatures">false</boolean>
                        <boolean name="critical planes all">false</boolean>
                        <boolean name="critical planes worst">false</boolean>
                        <boolean name="critical distance success">false</boolean>
                        <boolean name="critical distance diagnostics">false</boolean>
                        <boolean name="critical distance cycle">false</boolean>
                        <boolean name="turbolife accumulated stress strain">false</boolean>
                        <boolean name="NASA life2">false</boolean>
                        <boolean name="temperature-dependent UTS and FS">false</boolean>
                        <boolean name="maximum temperature">false</boolean>
                    </group>
                    <group name="logs">
                        <boolean name="load_sensitivity">false</boolean>
                        <boolean name="worst n lives">false</boolean>
                        <integer name="n">100</integer>
                        <boolean name="ranked elimination table">false</boolean>
                        <boolean name="nodal information">false</boolean>
                        <boolean name="block life table">false</boolean>
                        <boolean name="plane life table">false</boolean>
                        <boolean name="critical plane life table">false</boolean>
                        <boolean name="all planes life table">false</boolean>
                        <boolean name="fos planes">false</boolean>
                        <boolean name="grey iron damage">false</boolean>
                        <boolean name="dataset stresses">false</boolean>
                        <boolean name="ep residuals">false</boolean>
                        <boolean name="stress and strain tensors">false</boolean>
                        <boolean name="principals">false</boolean>
                        <boolean name="TURBOlife">false</boolean>
                        <boolean name="critical distance items">false</boolean>
                        <boolean name="critical distance summary">false</boolean>
                        <boolean name="PSD items">false</boolean>
                        <enumerator name="history indexing">One</enumerator>
                    </group>
                    <boolean name="only analyse listed items">false</boolean>
                    <boolean name="traffic lights">false</boolean>
                    <real name="traffic lights lower">1e+06</real>
                    <real name="traffic lights upper">1e+07</real>
                    <enumerator name="unit system">Metric</enumerator>
                </group>
                <group name="fos">
                    <real name="design life">1.3e+08</real>
                    <boolean name="infinite design life">false</boolean>
                    <real name="maximum">5</real>
                    <real name="fine maximum">1.5</real>
                    <real name="fine minimum">0.8</real>
```

```
                    <real name="minimum">0.5</real>
                    <integer name="maximum coarse iterations">20</integer>
                    <integer name="maximum fine iterations">16</integer>
                </group>
                <group name="critical distance">
                    <boolean name="apply corrections">false</boolean>
                    <enumerator name="method">point</enumerator>
                    <boolean name="use safety-factor thresholds">true</boolean>
                    <real name="safety-factor ceiling">10</real>
                    <real name="safety-factor floor">0</real>
                    <boolean name="allow quadratic interpolation">true</boolean>
                </group>
                <group name="welds">
                    <real name="proportional variance threshold">0.8</real>
                    <boolean name="use modified wang_brown">true</boolean>
                    <boolean name="use nonlinear shear correction">true</boolean>
                    <enumerator name="structural stress calculation mode">Basic Nodal
Force</enumerator>
                    <boolean name="automatically bound plate thickness">false</boolean>
                    <real name="plate bound median margin">0.25</real>
                    <boolean name="split incomplete toes">false</boolean>
                    <real name="min line length post split">0.75</real>
                    <boolean name="reject no through nodes">false</boolean>
                    <boolean name="auto-detect toe failures">true</boolean>
                    <boolean name="auto-detect fusion failures">true</boolean>
                    <boolean name="auto-detect throat failures">false</boolean>
                    <boolean name="auto-detect root failures">false</boolean>
                    <boolean name="use compact contour mode">true</boolean>
                    <real name="co-planarity tolerance">5</real>
                    <real name="cone tolerance">0.05</real>
                    <enumerator name="Ir function">IrDisplacement</enumerator>
                    <integer name="smoothing iterations">0</integer>
                    <enumerator name="weld end correction">VirtualNode</enumerator>
                    <enumerator name="throat/fusion failure sense">FromSurface</enumerator>
                    <real name="wrap angle">90</real>
                    <real name="blunt root threshold">0.25</real>
                </group>
                <group name="planes">
                    <integer name="critical plane search count">18</integer>
                </group>
                <group name="psd">
                    <enumerator name="PSD response">Von Mises</enumerator>
                    <boolean name="Apply nodal filtering">false</boolean>
                    <integer name="Number of stress range intervals">1000</integer>
                    <real name="RMS stress cut-off multiple">10</real>
                    <real name="Combined shear-normal K">0.25</real>
                    <boolean name="Bound Dirlik at UTS">false</boolean>
                    <enumerator name="Weld SS response">Normal</enumerator>
                </group>
                <file name="loading file">./Sierra_Fatigue.ldf</file>
                <file name="output file">./Sierra_STS_ST_Hex_Irrad_W_Pulses_FESAFE.odb</file>
                <string name="loading mode">LDF</string>
                <real name="scale factor">1</real>
                <real name="overflow value">0</real>
                <real name="infinite life value">-1</real>
                <integer name="modal samples per cycle">10</integer>
                <real name="modal gate percentage">0.1</real>
                <boolean name="disable temperature analysis">false</boolean>
                <string name="life units description">Repeats</string>
                <real name="life units factor">1</real>
                <boolean name="disable triaxiality">false</boolean>
                <boolean name="disable failed directional cosine to XYZ">false</boolean>
                <boolean name="Von Mises signed">false</boolean>
                <boolean name="ignore compressive cycles">false</boolean>
                <boolean name="enforce thread affinity">false</boolean>
                <integer name="max overflows">1000</integer>
            </group>
        </group>
    </data>
</root>
```

8.8.4 FE-SAFE Loading File (Sierra_Fatigue.ldf)

```
INIT
transitions=Yes
END

BLOCK n=1
ds=1-4019
END
```