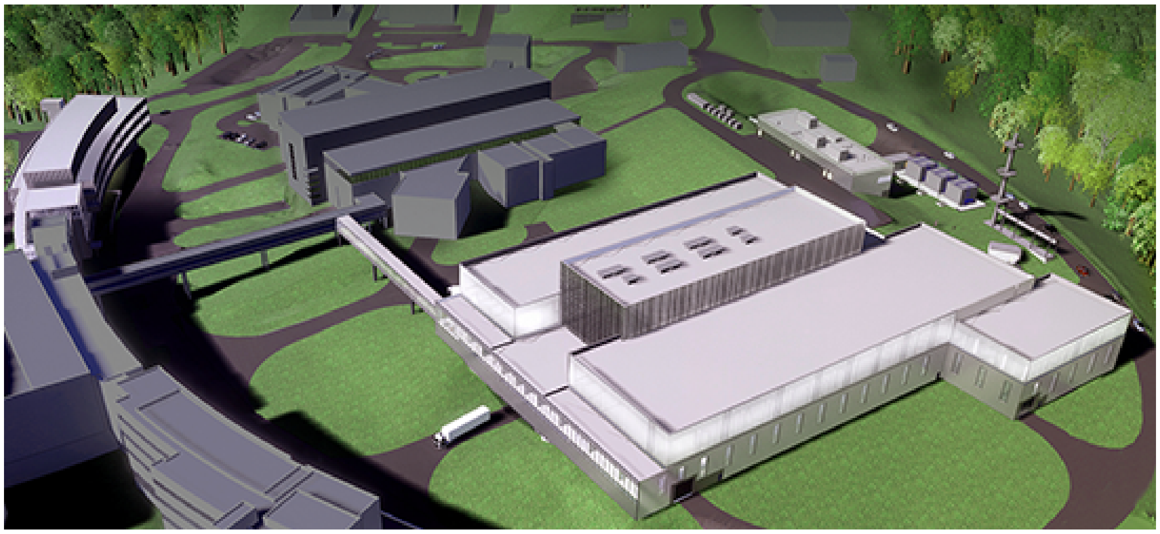
ORNL/TM-2024/3429

Uncertainty Quantification of Monolithic Tungsten Target Block Fatigue



Joseph B. Tipton, Jr.

March 2022

ORNL/TM-2024/3429

Second Target Station (STS) Project

UNCERTAINTY QUANTIFICATION OF MONOLITHIC TUNGSTEN TARGET BLOCK FATIGUE

Joseph B. Tipton, Jr.

March 2022

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831
managed by
UT-BATTELLE LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR2272

ABSTRACT

The current STS spallation target design consists of 21 water cooled, tantalum clad, monolithic tungsten blocks. The target wheel assembly rotates such that each target block receives a direct proton pulse every 1.4 seconds (21 blocks / 15 Hz pulses). The quasi-steady heating induced by the repetitive beam pulses creates a significant mean stress distribution in the target. On top of this, the rapid energy deposition from a proton pulse induces dynamic stress waves that are of the same order of magnitude as the mean stresses. Over the design goal of 5,000 operational hours per year for a decade, each target must withstand over 130 million repetitions of these dynamic stress events. The fatigue life prediction of the target blocks is a complex sequence of simulations that depend upon a set of input parameters. For the tungsten block, the end goal is the fatigue factor-of-safety (FOS).

This study documents a new structural analysis routine using software available to U.S. national laboratories (e.g. CUBIT, Sierra Multiphysics, Dakota) that enables more rapid target fatigue analysis. The increased throughput is amenable to parameter sensitivity and uncertainty quantification (UQ) studies. A set of input parameters (ultimate strength, fatigue strength, density, elastic modulus, Poisson's ratio, thermal conductivity, coefficient of thermal expansion, specific heat, flow rate) with expected ranges are documented based on literature data, recent experimental testing, and engineering judgement.

UQ bounds the resulting fatigue FOS at [0.5, 3.4] using epistemic interval analysis. These bounds are expected to be wider due to model effects (discretization) and beam effects (focus, offset) that were not included. Parameter sensitivity underscores the importance of ultimate strength and fatigue strength of irradiated tungsten on the fatigue calculations.

CONTENTS

TABLE OF FIGURES

1. OVERVIEW

1.1 INTRODUCTION & PURPOSE

The current STS spallation target design consists of 21 water cooled, tantalum clad, monolithic tungsten blocks. The target wheel assembly rotates such that each target block receives a direct proton pulse every 1.4 seconds (21 blocks / 15 Hz pulses).

The quasi-steady heating induced by the repetitive beam pulses creates a significant mean stress distribution in the target. On top of this, the rapid energy deposition from a proton pulse induces dynamic stress waves that are of the same order of magnitude as the mean stresses. Over the design goal of 5,000 operational hours per year for a decade, each target must withstand over 130 million repetitions of these dynamic stress events.

The fatigue life prediction of the target blocks is a complex sequence of simulations that depend upon a set of input parameters. For the tungsten block, the end goal is the fatigue factor-of-safety (FOS). The purpose of this report is to:

a) document a new software routine that enables more rapid target fatigue analysis
b) document current best understanding of nominal design input parameters as well as their uncertainties
c) characterize the sensitivity of the fatigue FOS to the input parameters
d) quantify the uncertainty of the fatigue FOS to the input parameters

1.2 SOFTWARE REQUIREMENTS

Design exploration, sensitivity analysis (SA), and uncertainty quantification (UQ) activities require a capability for large numbers of simulations within a reasonable amount of time. The target assembly group has developed a target block analysis capability using software that is unencumbered by license pools either by nature of being open-source or available to national laboratories through the United States government. The following software packages were utilized on the OZ3 high performance cluster (HPC) at SNS.

- ❖ CUBIT [1]
  - ➢ Version 15.6
  - ➢ Geometry and mesh generation toolkit
  - ➢ Freely available to US agencies

- ❖ Python
  - ➢ Version 2.7.5
  - ➢ Requires Numpy, Sklearn, and pyTetGen [2] modules
  - ➢ Open source

- ❖ Sierra Multiphysics
  - ➢ Version 4.56.4
  - ➢ Requires Sierra Solid Mechanics [3], Sierra Aria Thermal Mechanics [4], and Exodus Python Module
  - ➢ Available to US agencies with foreign national and/or export control restrictions

- ❖ Dakota [5]
  - ➢ Version 6.15

- ➢ Software for optimization and UQ
- ➢ Open source

❖ TORQUE
- ➢ HPC queueing system with MAUI job scheduler
- ➢ Required to run Sierra simulations in parallel on the OZ3 cluster

1.3 ASSUMPTIONS

Key assumptions made in these analyses are as follows:

Heat Transfer
- Tantalum cladding omitted
- Steady-state volumetric heating only considers block in 3 positions (direct beam pulse and immediate neighbor beam pulses)
- Fluid flow replaced with convection boundary conditions
- Constant properties
- Fully irradiated thermal conductivity
- Steady-state model does not consider transient effects between beam pulses

Structural
- Tantalum cladding omitted (In reality, the HIP process will create slight residual compressive stresses on the tungsten. It is assumed that radiation damage and stress relaxation will significantly reduce or eliminate any residual stress.)
- Constant properties
- No body accelerations
- No contacts
- No voids and perfectly smooth surfaces

Fatigue
- Tantalum fatigue omitted
- Tungsten failure mode is fully brittle fracture
- Goodman mean stress correction
- S-N fatigue curve modeled as linear log-log (Basquin's model)
- The logarithms of fatigue lives are normally distributed (log-normal distribution) with constant variance
- Irradiated knockdown factor determined from ESS testing of cross-rolled tungsten
- Beam trips not considered
- Effects of oscillations during beam pulse response are not counted in fatigue life. For a given beam pulse, the mean stress and stress amplitude are calculated from the largest and smallest stresses during the dynamic response.

2. TARGET ANALYSIS

2.1 OVERVIEW

The automated target block analysis routine has five major components as summarized in Figure 1. The inputs to the analysis are (a) CAD geometry, (b) neutronics heating, and (c) material and flow input parameters. The following sections explain the inputs and analysis steps in further detail.

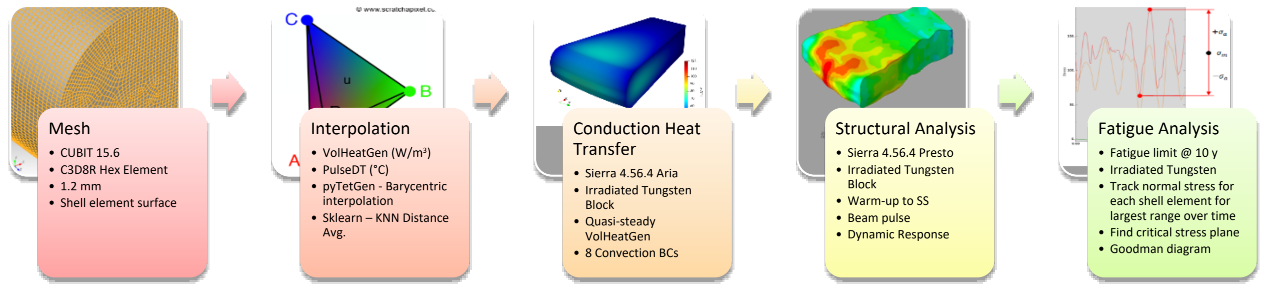


Figure 1: Summary of the automated and simplified target analysis routine for parameter sensitivity, uncertainty quantification, and optimization studies.

2.2 GEOMETRY

The CAD geometry used for analyses in this report is the November 2021 release of the "Cherry Cobbler" target configuration. The analysis domain is just the tungsten block. Figure 2 shows the geometry which has a flat rear and front with compound curvature. The STS_WORLD_CENTER coordinate system is also indicated.

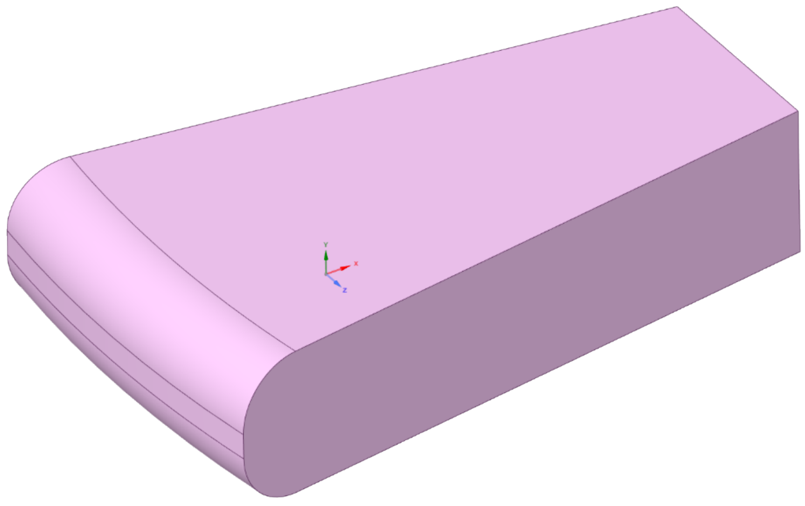


Figure 2: Tungsten block CAD geometry showing STS_WORLD_CENTER coordinate system.

2.3 MESH

The first step of the analysis creates a discretization of the target block geometry using CUBIT. A script file (see APPENDIX B) automates the mesh creation using the following steps:

- Imports CAD file in ACIS format ("OptTest_WBlock.sat")
- Meshes the target block with C3D8R Hex elements using a 1.2 mm nominal size

- Meshes the target block surface with SHELL4 elements that share nodes with the hex elements
- Creates 2 element blocks: surface shell elements & interior hex elements
- Creates a node set containing all nodes
- Creates a surface set for each of the 8 block surfaces
- Exports mesh in Exodus II format ("OptTest_WBlock.g")

Figure 3 shows a screen shot of the completed discretization including the named selections. In this example, W_BLOCK_SURF consists of 65,690 shell elements and W_BLOCK_INT consists of 914,204 hex elements.

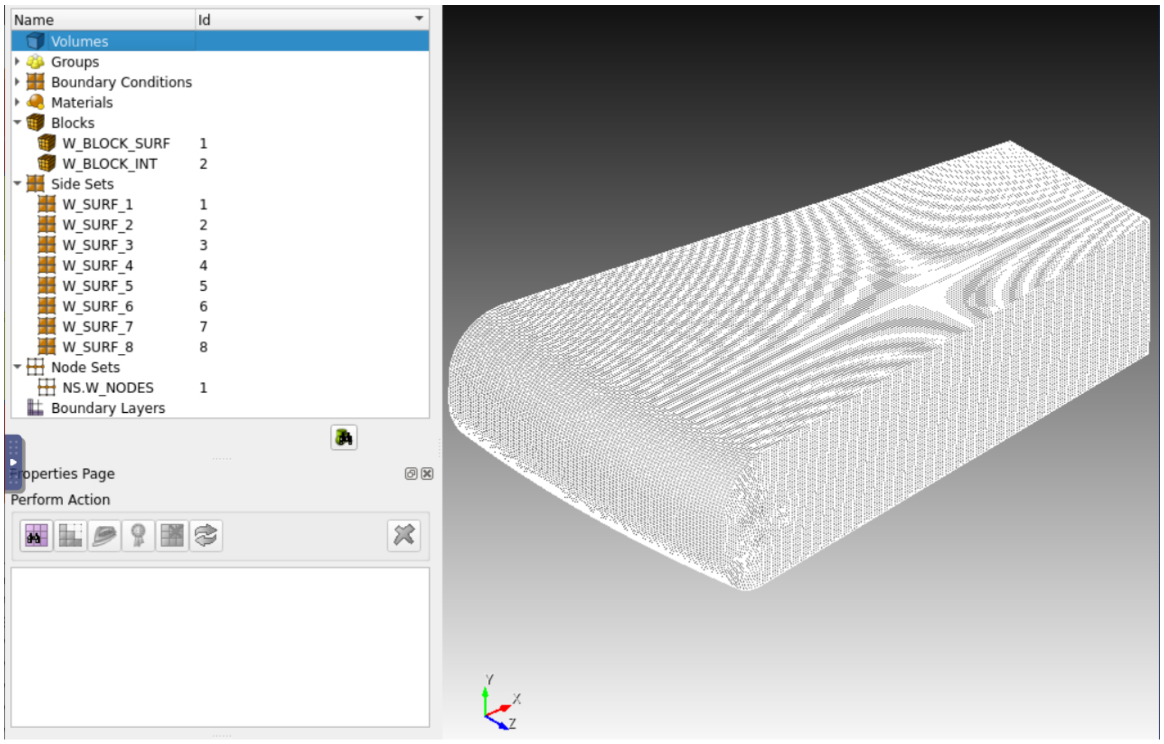


Figure 3: Screenshot of CUBIT showing target block mesh including named selections.

2.4 NEUTRONICS HEATING INTERPOLATION

Once the mesh is created for analysis, the neutronics heating data must be interpolated onto the mesh. This is performed with a custom python module (see APPENDIX C). The module constructs the neutronics heating point cloud as a tetrahedral mesh using the Delaunay triangularization method found in the pyTetGen module. Each node of the target block mesh is then located within a tetrahedron and is thus surrounded by 4 neutronic heating values. These are weighted by area using Barycentric interpolation. Any nodes external to the triangularization (due to coordinate rounding) are then extrapolated using a distance weighting of the 3 nearest neutronics heating neighbors via the scikit-learn module.

The neutronics heating source files come from the neutronics analysis TO-001_S.03.02_2021-12-01_TARGET_HEAT_DPA and used a nominal beam (Super-Gaussian-TM2-1.2-2.45) centered on the target block. The file "o10b_W_1.csv" represents the energy deposition on the target block that receives the proton pulse. It is converted into an instantaneous temperature rise via the thermal capacitance ($\rho \cdot c_p$) of the tungsten. The file "o10b_W_combined.csv" represents the total energy deposition into a single target block in one revolution around the wheel assembly. Since the neutronics results files use an identical mesh, it is obtained by summing the energy deposition of files "o10b_W_1.csv" through "o10b_W_21.csv" directly. It is converted into a volumetric heat generation rate via the period between pulses for a single block (21 blocks / 15 hertz = 1.4 seconds).

Figure 4 shows the proton pulse energy deposition after interpolation onto the analysis mesh. The pulse temperature rise will follow this pattern but varies in magnitude according to the thermal capacitance. Figure 5 shows the quasi-static volumetric heat generation rate after interpolation onto the analysis mesh.

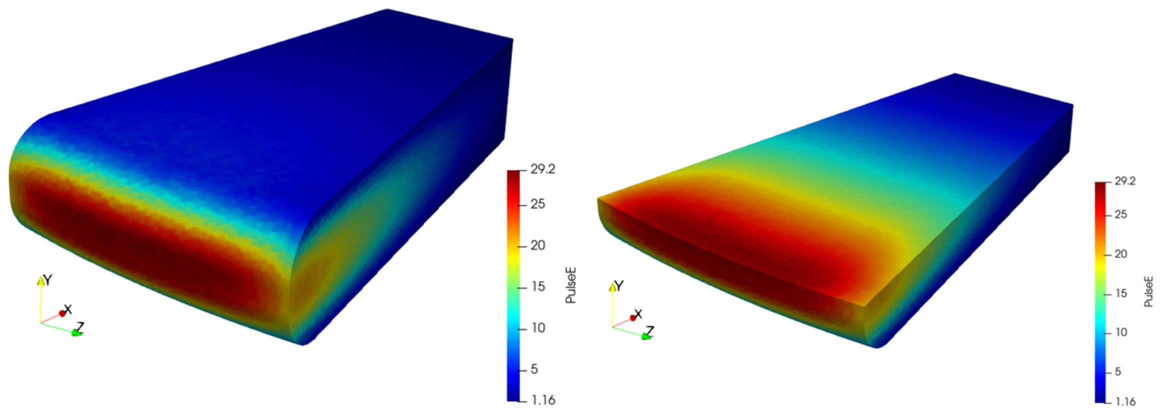


Figure 4: Energy deposited in a single pulse [J/cc/pulse].

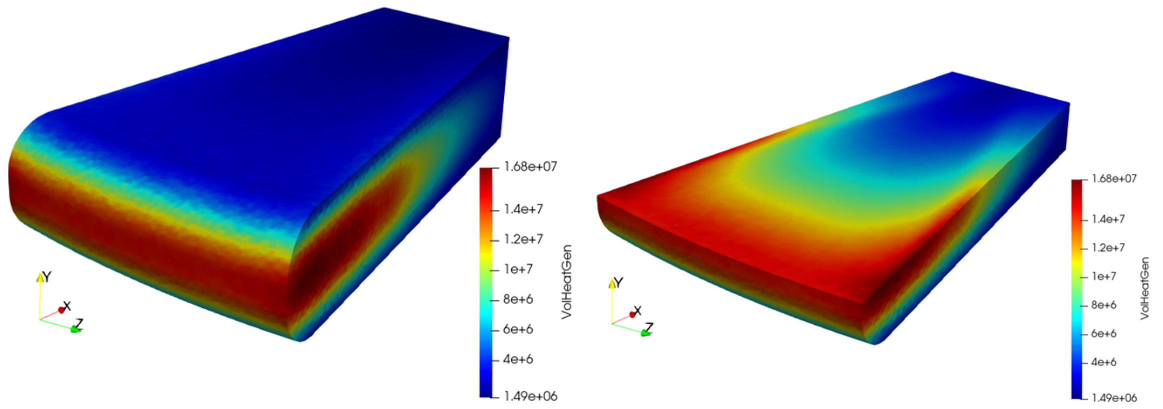
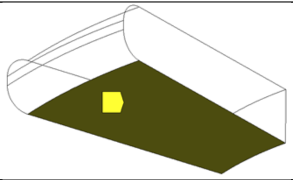
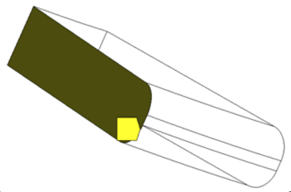
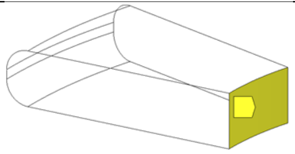
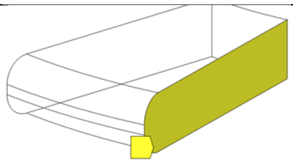
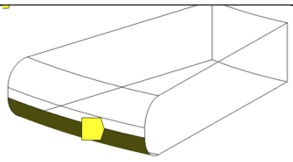
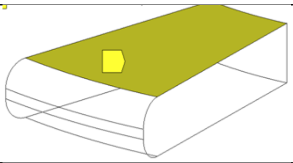
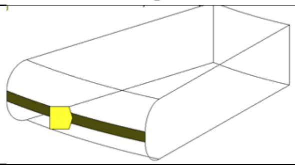
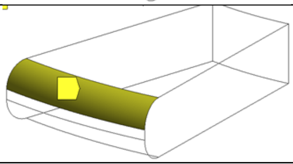


Figure 5: Total quasi-static volumetric rate of heat generation [W/m^3]. Created by summing energy deposition on a block in each position around the target wheel for a set of 21 beam pulses.

2.5 CONDUCTION HEAT TRANSFER

The next step in target block analysis is a quasi-steady heat transfer simulation using the input mesh and volumetric heat generation field. The Sierra Aria code is used to solve diffusion heat transfer in the target block (see APPENDIX D). Convection boundary conditions are defined on each surface of the target block to approximate the more accurate heat transfer effects of a conjugate heat transfer CFD simulation. Table 1 identifies the nominal boundary conditions for each of the 8 tungsten block surfaces. These boundary conditions were determined using a rough optimization (i.e. goal driven optimization combined with engineering judgement) in ANSYS Mechanical. Figure 6 shows a side-by-side comparison of the nominal temperature and static stress results against a full CFD conjugate heat transfer analysis performed by Min-Tsung Kao. The convection boundary conditions appear to capture the major trends of the static stress solution while underpredicting the peak static stress by approximately 13%.

Table 1: Convection boundary conditions for target block heat transfer analysis.

Boundary Name	**Bulk Fluid Temp.**	**Convection Coeff.**	**Location**
Convection S1	35 [°C]	7,000 [W/m^2-°C]	
Convection S2	35 [°C]	4,500 [W/m^2-°C]	
Convection S3	35 [°C]	3,500 [W/m^2-°C]	
Convection S4	35 [°C]	5,500 [W/m^2-°C]	
Convection S5	35 [°C]	7,000 [W/m^2-°C]	
Convection S6	35 [°C]	8,000 [W/m^2-°C]	
Convection S7	35 [°C]	7,000 [W/m^2-°C]	
Convection S8	35 [°C]	9,900 [W/m^2-°C]	

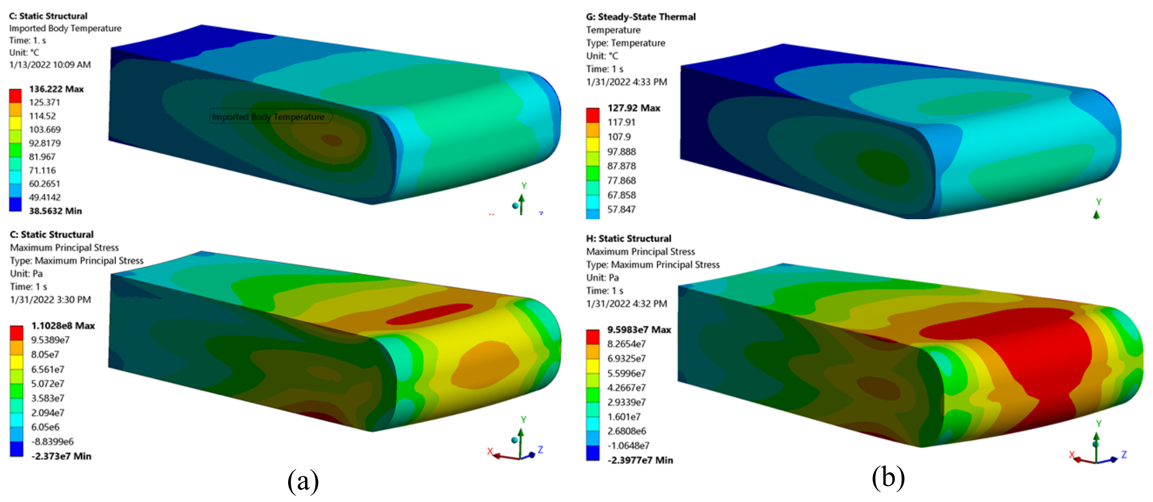


Figure 6: Target block quasi-steady temperature and stress distributions. (a) Full analysis (Steady CFD + Steady FEA); (b) "Rough" Optimization (Convection BCs + Steady FEA). The convection boundary conditions under-predict the maximum values while capturing the shape distributions. The contour plots bands are not aligned between images.

2.6 STRUCTURAL ANALYSIS

The fourth step in the automated target block analysis is a structural simulation using Sierra Presto (see APPENDIX E). The domain is the fully elastic tungsten block with no boundary conditions. The only loads are applied as temperature fields. A python script takes the heat transfer simulation results and neutronics pulse temperature rise fields and places them in the correct time order for the structural analysis.

There are 3 steps to the structural analysis. First, a quasistatic step is defined to simulate the warm-up to steady operation of the target. The temperature field from the heat transfer solution drives thermal stresses. Second, an explicit step is defined to apply the neutronics pulse temperature rise over 700 [ns]. Third and final, the target block is allowed to dynamically respond over 0.4 [ms] as the elastic stress waves propagate, reflect, and interfere throughout the domain. For file size management, output is saved for the surface elements only. And for each surface shell element, the transformed in-plane stresses are saved (at the element integration point). A more detailed document on the development of Sierra simulations for target analysis is documented elsewhere [6].

2.7 FATIGUE ANALYSIS

The fifth and final step in the automated target block analysis is a fatigue calculation using a custom python script (see APPENDIX F). The irradiated tungsten block is assumed to follow brittle failure theory. Accordingly, normal stress is the expected failure mode, and a fatigue calculation should calculate the largest normal stress amplitude as a function of stress plane orientation (critical plane analysis). Figure 7 explains the calculation using flow chart logic. Figure 8 shows a graphical example for a single element. Once the stress mean and amplitude are determined for each element, a fatigue factor-of-safety (FOS) is calculated using the modified Goodman equation as shown in Figure 18. The use of Goodman's linear relationship and the fatigue curve for irradiated tungsten are explained further in Section 3.3 "Fatigue Strength". The smallest FOS among all surface elements and among all stress planes is reported as the solution output.

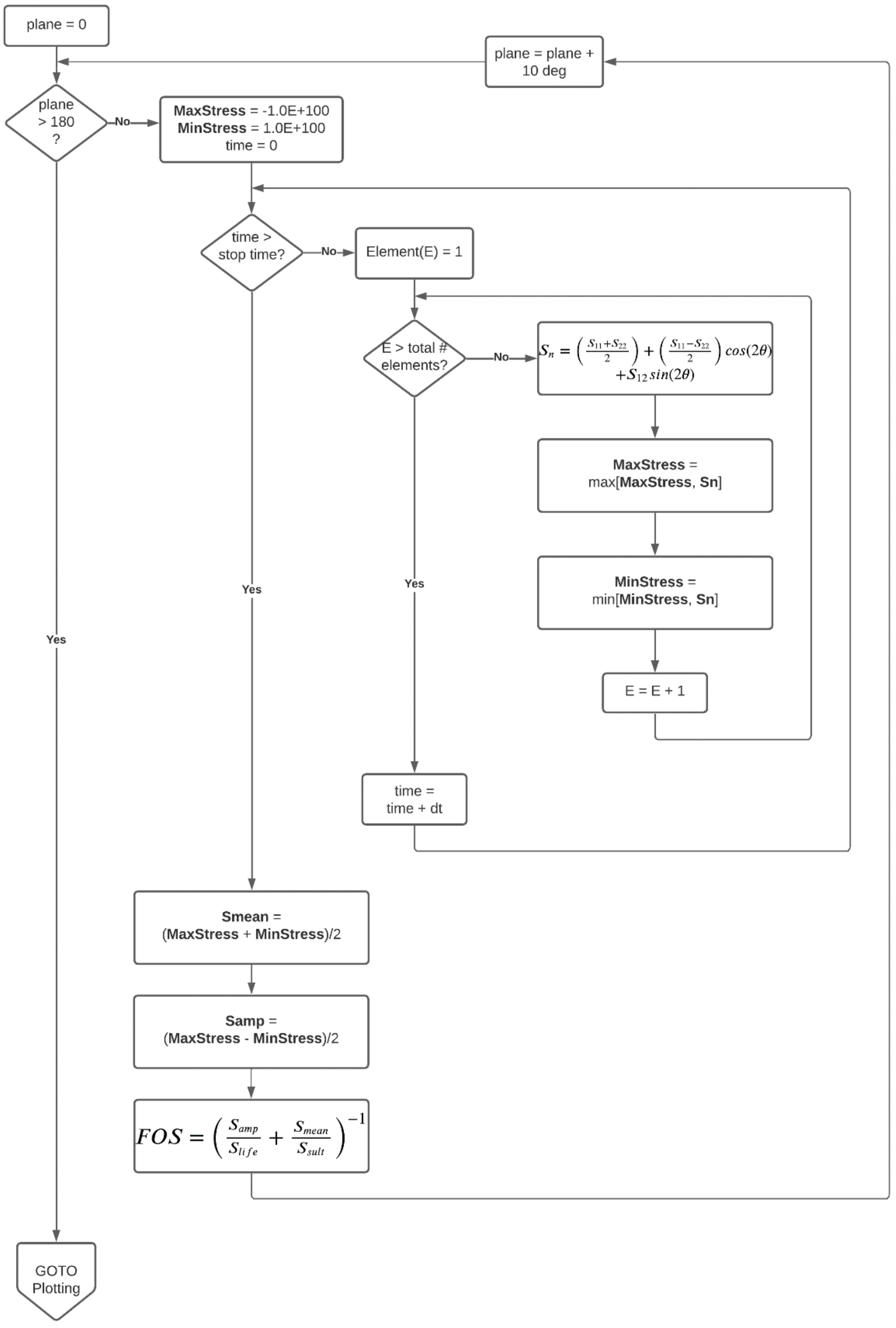

plane = 0
plane > 180 ?
No
MaxStress = -1.0E+100
MinStress = 1.0E+100
time = 0
plane = plane + 10 deg
time > stop time?
No
Element(E) = 1
E > total # elements?
No
$S_n = \left(\frac{S_{11}+S_{22}}{2}\right) + \left(\frac{S_{11}-S_{22}}{2}\right) cos(2\theta) + S_{12} sin(2\theta)$
MaxStress = max[MaxStress, Sn]
MinStress = min[MinStress, Sn]
E = E + 1
Yes
time = time + dt
Yes
Smean = (MaxStress + MinStress)/2
Samp = (MaxStress - MinStress)/2
$FOS = \left(\frac{S_{amp}}{S_{life}} + \frac{S_{mean}}{S_{sult}}\right)^{-1}$
Yes
GOTO Plotting

Figure 7: Program flow chart logic for calculation of elemental mean stress and stress amplitude for tungsten block dynamic response.

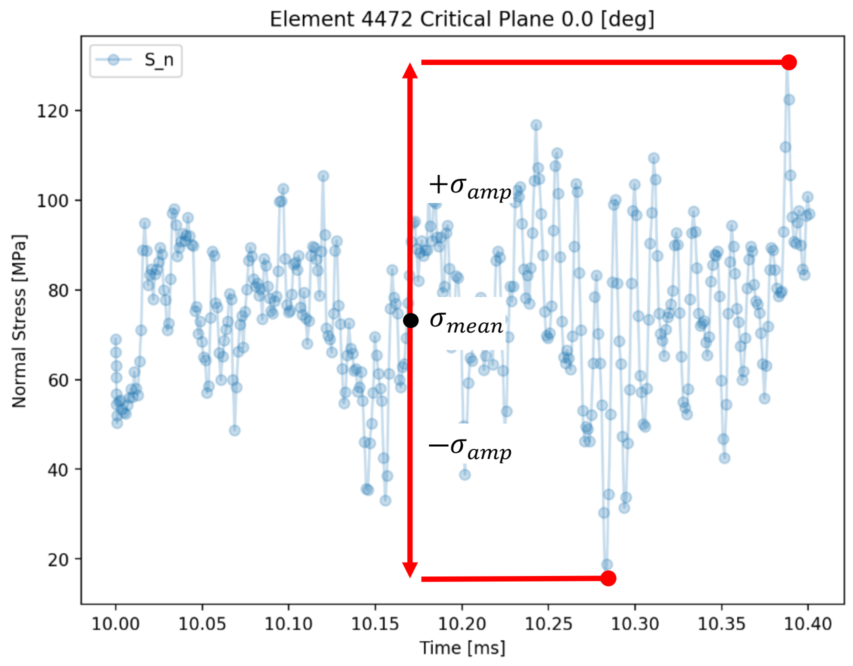


Figure 8: Example dynamic response of a tungsten block element illustrating the calculation of mean stress and stress amplitude from the normal stress of an element at a stress plane angle.

3. INPUT PARAMETER UNCERTAINTY

3.1 INPUT SUMMARY

Table 2 identifies the input parameters used throughout the target block analysis and which portion of the analysis they impact. The last parameter "flow" describes the effect of water cooling on the target block. The other parameters are all properties of tungsten (assumed to be irradiated).

Table 2: Analysis input parameters – definitions and regions of impact.

Symbol	Name	Analysis Impact
SultT	Ultimate tensile strength	Fatigue FOS – Goodman ultimate strength Fatigue FOS – Goodman fatigue strength
K_95	95% PI knockdown for fatigue strength at 10 yr life	Fatigue FOS – Goodman fatigue strength
rho	Density	Structural analysis – W mass Structural analysis – Beam pulse temperature rise
E	Modulus of elasticity	Structural analysis
poisson	Poisson's Ratio	Structural analysis
k	Thermal conductivity	Heat transfer analysis
CTE	Coeff. of thermal expansion	Structural analysis

Symbol	Name	Analysis Impact
cp	Specific Heat	Structural analysis – Beam pulse temperature rise
flow	Flow rate multiplier	Heat transfer analysis

Table 3 summarizes the nominal value for each parameter as well as the expected lower and upper bounds. The type of uncertainty is also listed. The parameters with epistemic uncertainty are intrinsic properties of the irradiated tungsten (e.g. density, elastic modulus, Poisson's ratio, thermal conductivity, coefficient of thermal expansion, and specific heat). These properties are assumed to be constant throughout the material. There is thus an actual value to the parameter that is unknown due to a lack of knowledge (epistemic). The nominal values for these parameters are inherited from published analyses of the ISIS spallation targets [7].

The parameters with aleatory uncertainty are assumed to have randomly varying values. The ultimate tensile strength and fatigue strength are assumed to vary randomly with space (location) as the strength of brittle materials is highly sensitive to inclusions, porosity, and surface effects. The flow rate of coolant over the target block is assumed to randomly vary with time due to the dynamic effects of fluid flow through a complex flow circuit.

The following sections further describe how the parameter bounds were determined.

Table 3: Analysis input parameters – nominal values with expected ranges and uncertainty type.

Symbol	Name	Lower Bound	Nominal	Upper Bound	Units	Type
SultT	Ultimate tensile strength	200	300	400	[MPa]	Aleatory
K_95	95% PI knockdown for fatigue strength at 10 yr life	0.78	1.0	1.2	[ ]	Aleatory
rho	Density	19,033	19,250	19,361	[kg/m^3]	Epistemic
E	Modulus of elasticity	398	398	477.6	[GPa]	Epistemic
poisson	Poisson's Ratio	0.266	0.28	0.294	[ ]	Epistemic
k	Thermal conductivity	78.3	87	175	[W/m-°C]	Epistemic
CTE	Coeff. of thermal expansion	4.05E-06	4.5E-06	4.95E-06	[1/°C]	Epistemic
cp	Specific Heat	115.47	128.3	141.13	[J/kg-°C]	Epistemic
flow	Flow rate multiplier	0.85	1.0	1.15	[ ]	Aleatory

3.2 ULTIMATE TENSILE STRENGTH (UTS)

The best knowledge of UTS comes from experiments at the European Spallation Source (ESS) by Habainy *et al*. [8]. Pure tungsten samples were irradiated in a spallation-relevant environment followed by 3-point bend tests. The tungsten samples were cut from high-purity, cross-rolled, 10-mm plate. Figure 9 summarizes the results for irradiation at room temperature. The bend strength is used as a substitute for UTS and the 3 data points for irradiated strength are assumed to follow a normal distribution. The 95% prediction interval is calculated using Student's t-statistic as shown in Figure 10. The result is an average irradiated strength of 303 MPa with a 95% PI of ± 103 MPa. The expected range of values for ultimate tensile strength of irradiated tungsten is thus 200 to 406 MPa.

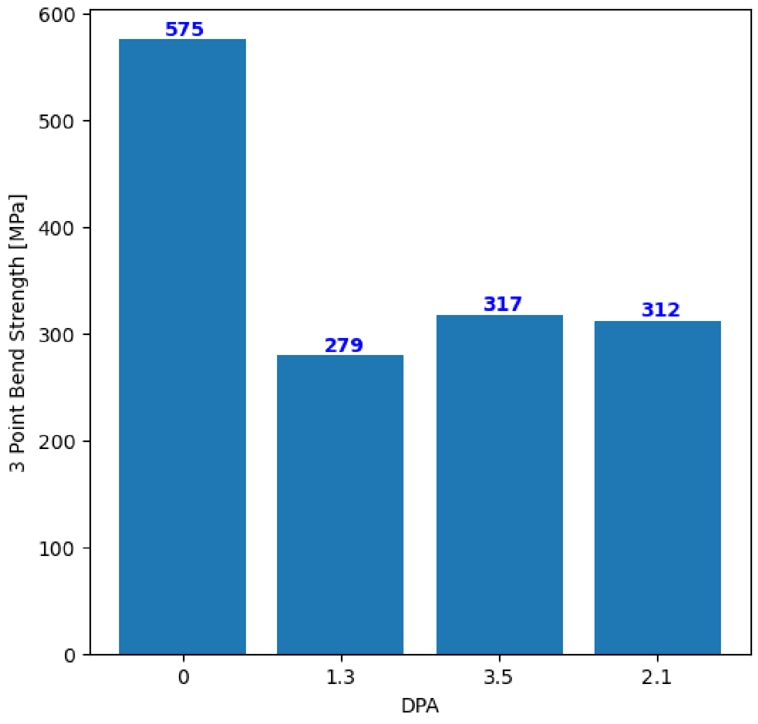


Figure 9: Strength testing of cross-rolled tungsten plate as a function of dpa from room-temperature irradiated in a spallation environment [8].

```
x = data
x_mean = np.mean(x)

n = len(x)                          # number of samples
m = 1                               # number of parameters
dof = n - m                         # degrees of freedom
t = stats.t.ppf(0.975, dof)         # Students statistic of interval confidence

residual = x - x_mean

std_dev = (np.sum(residual**2) / dof)**.5    # Standard deviation

# predicting interval
pi = t * std_dev * (1 + 1/n)**.5
```

Figure 10: Python script to calculate 95% prediction interval assuming normal statistics.

3.3 FATIGUE STRENGTH

3.3.1 Data Sources

There are 3 potential sources of fatigue data for unirradiated tungsten below the ductile-to-brittle transition temperature (DBTT). Schmidt and Ogden [9] reported fatigue data for transverse grain samples of 0.02 inch rolled tungsten sheet. The testing was completely reversed sheet-bending at room temperature. The data is summarized in Figure 11.

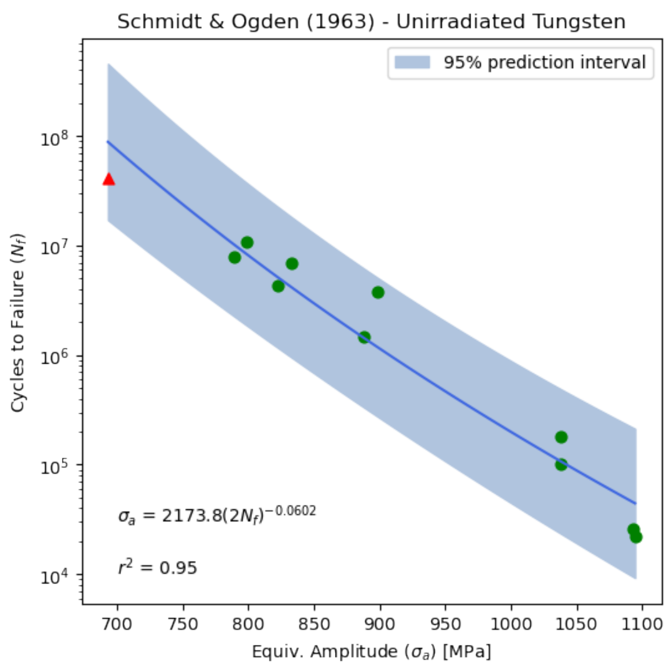


Figure 11: Bending fatigue of rolled tungsten sheet from Schmidt and Ogden [9]. The Basquin model fit parameters are included along with a 95% prediction interval assuming normal statistics. The triangle data are run-outs.

Habainy *et al*. [10] published fatigue tests of sintered and HIPed tungsten from Plansee. The samples were electropolished and tested at room temperature. Due to the brittle nature of the tungsten, the fatigue tests were performed only in the tensile regime. Figure 12 summarizes the data after using the Goodman linear relationship to adjust to a fully-reversing (R = -1) stress amplitude. The data was scaled using the lowest reported static tensile strength of 560 MPa.

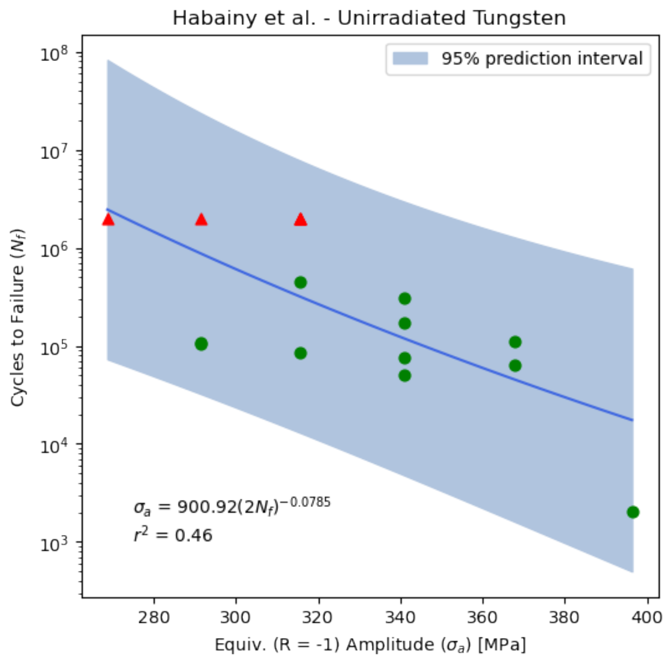


Figure 12: Tensile fatigue of sintered and HIPed tungsten from Habainy et al. [10]. The Basquin model fit parameters are included along with a 95% prediction interval assuming normal statistics. The triangle data are run-outs.

The Target Systems group at STS recently contracted with IMR Test Labs to perform rotating bending fatigue (RBF) tests of tungsten [11]. Samples were cut from 48 mm forged plate from Plansee. The samples were in-plane to the forging, finished with 600 grit sanding, and tested at room temperature. Figure 13 summarizes the data after using a 0.7 knockdown factor to account for the reduced tensile load area of rotating bending tests [12, p. 13].

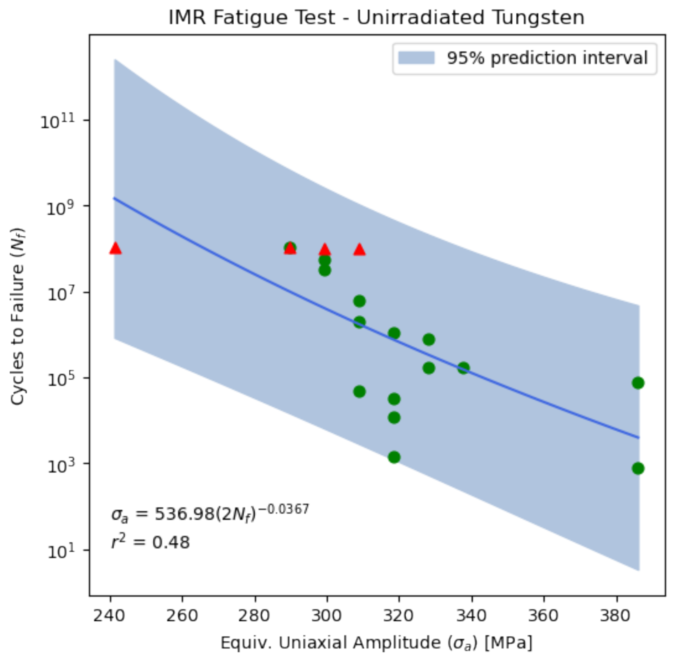


Figure 13: RBF fatigue of thick forged tungsten plate [11]. The Basquin model fit parameters are included along with a 95% prediction interval assuming normal statistics. The triangle data are run-outs.

3.3.2 Data Selection

The three potential data sets are plotted together and summarized in Figure 14. The circled data are excluded from analysis. The entirety of the Schmidt & Ogden data is obviously too different. The ESS testing data is kept except for the runouts which are too short-lived for STS target design. The IMR test data is kept save for a single sample (20RBF) that was the only test at 50 ksi (241 MPa) and ended in a runout. This sample has a high leverage on the statistics.

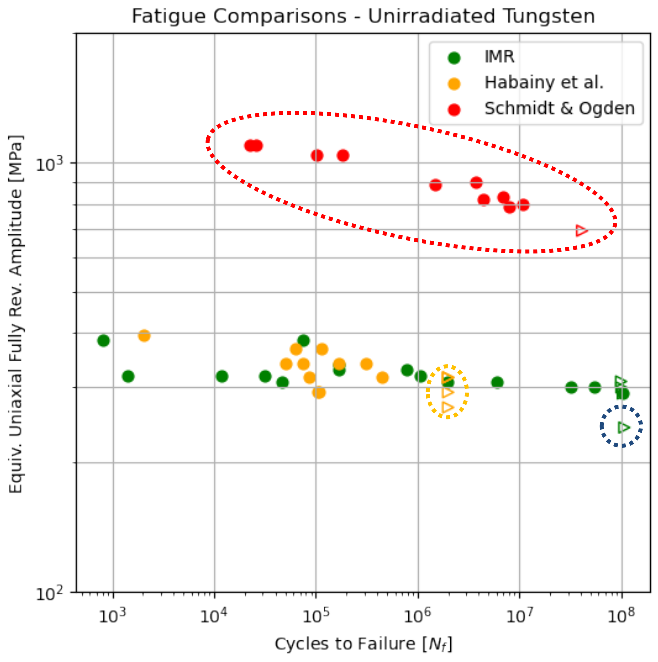


Figure 14: Summary of data used for tungsten fatigue analysis. The encircled data points are excluded.

3.3.3 Basquin Model Fit

Figure 15 shows the Basquin model fit of the combined fatigue data. A linear regression analysis is performed on a log-log transformation of the data where the testing amplitude is the independent variable. The data is then replotted without the transformation.

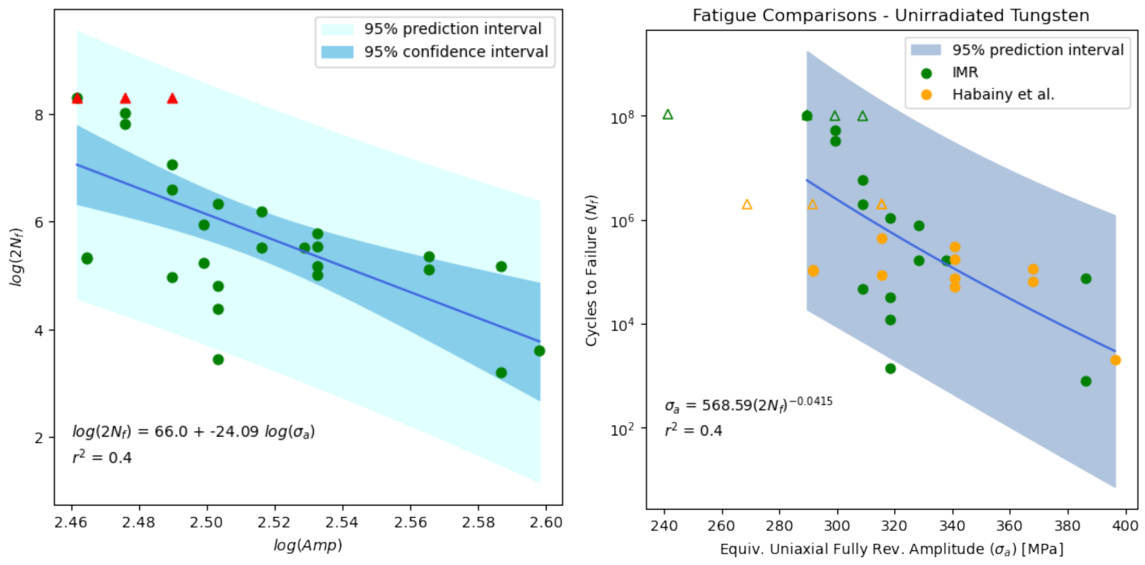


Figure 15: Basquin model fit for unirradiated tungsten fatigue. (L) Linear regression analysis; (R) Basquin model and 95% PI overlaid with complete ESS and IMR data sets.

3.3.4 Irradiated Knockdown Factor

The present analysis treats the tungsten target as fully irradiated for the entirety of the analysis. A knockdown factor is assumed to account for the effects of irradiation on the fatigue of tungsten. The fatigue strength coefficient, ($\boldsymbol{\sigma'_f}$ **= 569 MPa**), of the Basquin model fit is the extrapolated strength at one fatigue cycle which is conceptually equal to the static fracture stress of the material. Indeed, this value is within 1% of the reported bend strength of unirradiated tungsten from ESS (see Figure 9). An irradiation knockdown factor, $\boldsymbol{K_{irr}}$, is thus used to scale the Basquin model such that the effective fatigue strength coefficient is **300 MPa**. Figure 16 shows the resulting shifted fatigue curve. This methodology implicitly assumes the slope and scatter (i.e. 95% PI) of the Basquin model does not change. The lower 95% PI is algebraically modeled with a separate Basquin model fit and uses an additional knockdown factor ($\boldsymbol{K_{95}}$). For uncertainty quantification, the lower bound of $\boldsymbol{K_{95}}$ is thus 0.78 and the upper bound is 1.2 (roughly the upper 95% PI).

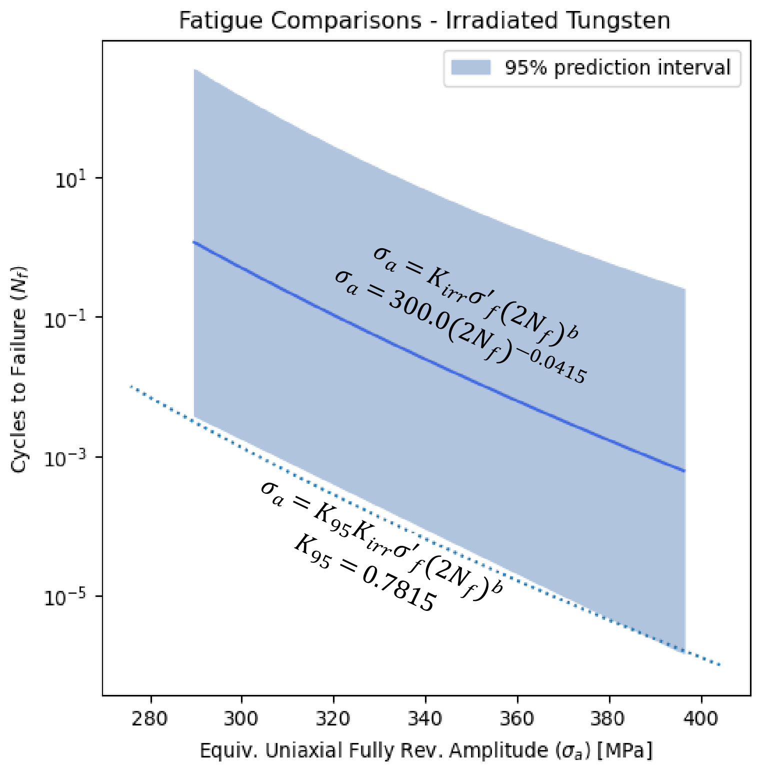


Figure 16: Basquin model fit for irradiated tungsten fatigue using an irradiation knockdown factor. The dotted line represents a separate Basquin model fit to algebraically approximate the lower 95% PI using an additional knockdown factor, K_{95}. (Note that the curves are extended into relevant cycle ranges in Figure 18.)

3.3.5 Irradiated Fatigue – Best Guess

Figure 18 summarizes the best engineering guess for fatigue analysis of the tungsten block stress response. The left image shows the original Basquin model fit with the unirradiated fatigue data. Also shown is the knocked-down curve for irradiated fatigue with the corresponding lower 95% PI. Two points are indicated on the figure corresponding to a 10 year life requirement for tungsten blocks. Figure 17 shows the 10 year operational life consists of 130 million beam pulse events on an individual tungsten block. Accordingly, the fatigue strength at 130 million cycles to failure is 134 MPa with a lower 95% PI of 105 MPa.

```
Life = 10                        # years
Year_to_Sec = 365 * 24 * 3600    # sec in a year
Uptime = 5000 / (365*24)         # hours operation / year
PulseFreq = 15                   # pulses / sec
NumTarget = 21
Pulses = Life * Year_to_Sec * Uptime * PulseFreq / NumTarget

print('Number pulses received by an individual target for design life = ', Pulses)
Number pulses received by an individual target for design life =  128571428.5
```

Figure 17: Python code snippet to calculate the total number of beam pulse events on an individual tungsten block over the 10 year design life requirement.

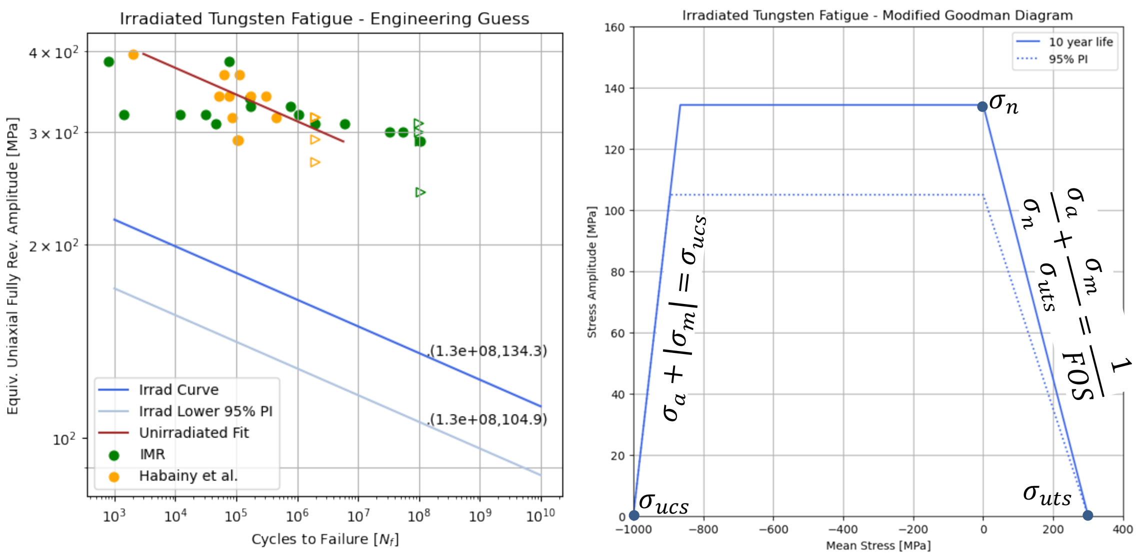


Figure 18: Tungsten fatigue (L) S-N curve for unirradiated and irradiated conditions; (R) Modified Goodman diagram for irradiated condition at 10 year design life.

The right image in Figure 18 shows the 10 year fatigue life design envelope as a modified Goodman diagram. The compressive strength regime is not an active constraint in practice and is ignored here. Instead, the Goodman equation is used to calculate the FOS for a given fatigue analysis. For uncertainty quantification, the fatigue strength is calculated as a function of both irradiated ultimate strength (*UTS*) and 95% PI knockdown (K_{95}) as shown in Equation (1).

$$\sigma_n = K_{95} \cdot \sigma_{uts} \tag{1}$$

3.4 DENSITY

Knowledge of tungsten density comes from testing Plansee 48 mm and 58 mm plate forgings. Testing was performed by Plansee and independently by IMR, both using the Archimedes method. The results are summarized in Figure 19. The 95% PI is calculated using normal statistics (see Figure 10). The result is an average density of 19,197 with a 95% PI of ± 164 kg/m^3. The expected range of values for analysis is thus 19,033 to 19,361 kg/m^3.

Density is assumed to be unaffected by irradiation. This is supported by experiments finding negligible radiation-induced swelling for tungsten at accelerator relevant conditions [13] [14].

It should be noted that lower density material is usually indicative of increased porosity and correlates with weaker strength. These effects are not captured in this report.

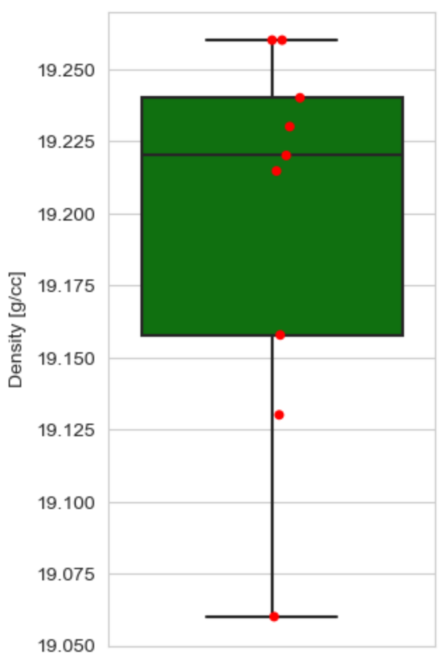


Figure 19: Graphical summary of density measurements for unirradiated tungsten. The data are summarized using a boxplot with individual measurements overlaid in red.

3.5 ELASTIC MODULUS

For irradiated tungsten, the lower bound (398 GPa) is equal to the nominal value. This is conservative since increase in stiffness drives the stress response amplitude. The upper bound comes from reported results that initially suggested a 15% increase in stiffness with irradiation [15]. These results are now in question, however, as described below [16]

> *In some of previous thermomechanical analyses of a tungsten block for the STS target, an increased modulus of elasticity for irradiated tungsten was used compared to that of unirradiated one. This was mainly motivated by a paper Habainy et al., where it was reported that uranium beam irradiated pure tungsten up to 0.1 dpa showed about 20% increase in stiffness [Note this was actually a 15% observed increase along with a further 5% increase for analysis conservatism - JBT]. However, it was found that the reported increase of elastic modulus is questionable. The experimental method that was used to derive an increase in elastic modulus in the Habainy et al. paper was critically reviewed by ORNL experts in nuclear materials. It was concluded that the observed increase of elastic modulus [would] most probably be a result of measurement artifacts rather than a true nature of irradiated tungsten. Following the expert's opinion, we decided to remove the redundant conservatism by taking the elastic modulus of unirradiated tungsten for thermomechanical analyses of tungsten blocks for the STS target design.*
>
> *On the other hand, it is still not clear whether the elastic modulus would change as a result of radiation damage. To resolve this issue, we are working on measuring elastic modulus of tungsten specimens irradiated at HFIR, in collaboration with ORNL experts. Additional radiation and PIE campaign will be planned for higher precision measurements in 2022.*

For now, the 20% increase of 477 GPa is kept as an upper bound until ORNL completes further testing of irradiated tungsten samples.

3.6 POISSON'S RATIO

In the absence of information, a 5% variation in Poisson's ratio is assumed.

3.7 THERMAL CONDUCTIVITY

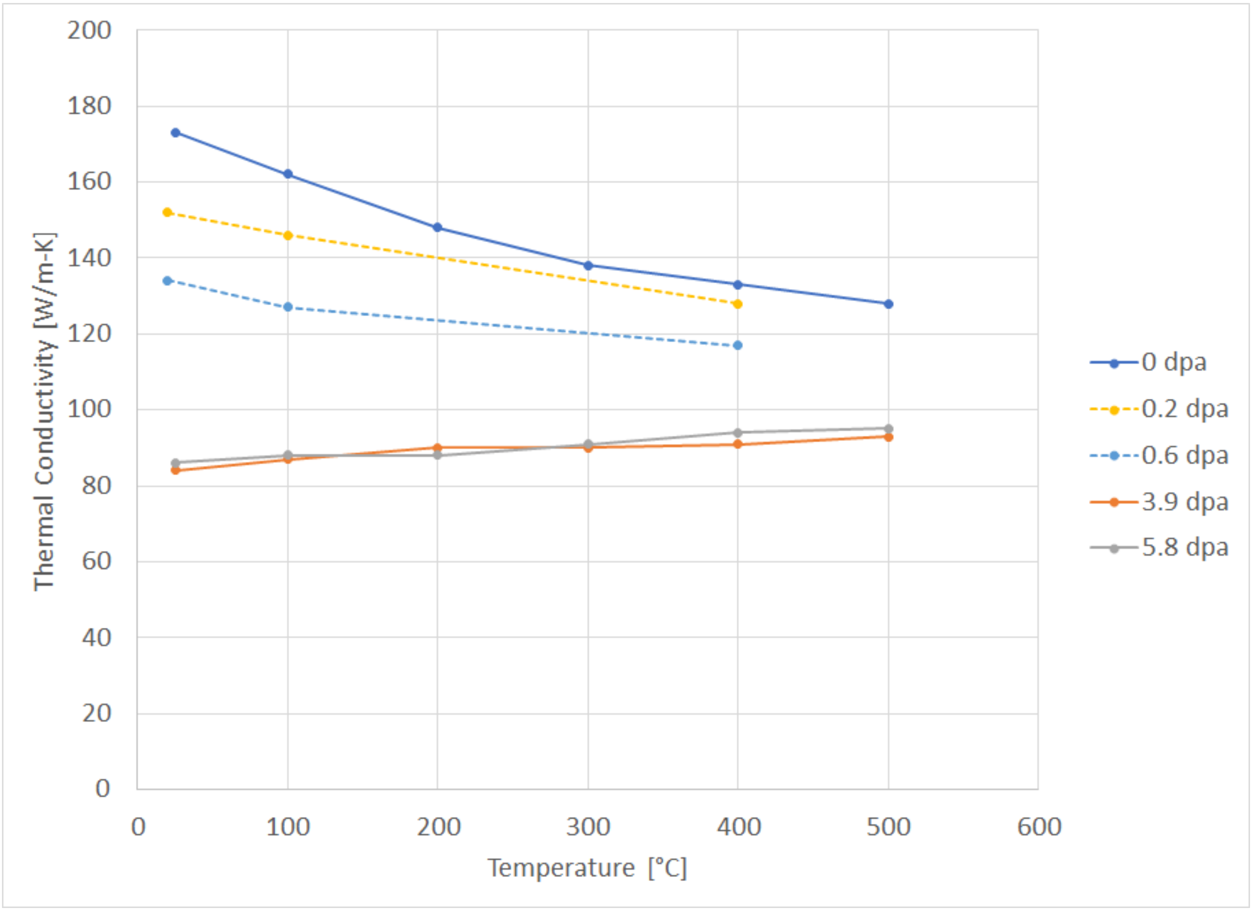


Figure 20: Thermal conductivity variation of tungsten as a function of temperature and DPA damage. Plot produced with data tabulated from [17].

As seen in Figure 20, the thermal conductivity of tungsten decreases upon irradiation and appears to saturate by at least 3.9 dpa of damage. The temperature dependence also disappears with this level of irradiation. Accordingly, simulations of irradiated tungsten blocks use a constant thermal conductivity of 87 W/m-K. The thermal diffusivity data reported by Habainy is consistent with other published thermal diffusivity experiments on rolled and annealed Tungsten [18].

For the purposes of uncertainty quantification, a lower bound of 78.3 W/m-K is assumed (10% below the nominal 87 W/m-K for irradiated tungsten). The upper bound is 175 W/m-K which corresponds to the unirradiated thermal conductivity at room temperature.

3.8 COEFFICIENT OF THERMAL EXPANSION

In the absence of information, a 10% variation in the coefficient of thermal expansion is assumed.

3.9 SPECIFIC HEAT

In the absence of information, a 10% variation in the specific heat is assumed.

3.10 FLOW RATE

As a rough approximation, the effect of flow rate on the simulation can be achieved by placing a multiplier on the convection boundary coefficients since these scale roughly linearly with the Reynolds number. A 15% variation is assumed.

4. UNCERTAINTY ANALYSIS

Dakota [5] is open-source software for optimization and UQ that is maintained by Sandia National Laboratory. The software is command-line driven using ASCII input files. A preprocessor, named *dprepro*, provides an easy method to pass parameters into other analysis software input files. For each analysis iteration, Dakota executes a driver script to pass the parameters, complete the analyses, and report the result. APPENDIX A "Dakota Scripts" contains these input files and the raw output.

This study includes 9 input parameters where 6 are epistemic and 3 are aleatory. Very little is known about the distribution of the parameters. With so little knowledge, an epistemic interval analysis is chosen for the uncertainty quantification [5, p. 69]:

> *Interval analysis is often used to model epistemic uncertainty. In interval analysis, one assumes that nothing is known about an epistemic uncertain variable except that its value lies somewhere within an interval. In this situation, it is NOT assumed that the value has a uniform probability of occurring within the interval. Instead, the interpretation is that any value within the interval is a possible value or a potential realization of that variable. In interval analysis, the uncertainty quantification problem is one of determining the resulting bounds on the output (defining the output interval) given interval bounds on the inputs. Again, any output response that falls within the output interval is a possible output with no frequency information assigned to it.*
>
> *We have the capability to perform interval analysis using either global or local methods. In the global approach, one uses either a global optimization method (based on a Gaussian process surrogate model) or a sampling method to assess the bounds. The local method uses gradient information in a derivative-based optimization approach, using either SQP (sequential quadratic programming) or a NIP (nonlinear interior point) method to obtain bounds.*

This study uses the efficient global optimization (EGO) with a Gaussian process surrogate model that is a high-order polynomial. The EGO is controlled automatically within Dakota using the Surfpack Kriging toolkit [19].

5. RESULTS

5.1 MESH CONVERGENCE

Figure 21 summarizes a mesh refinement study performed using a sample set of input parameters. The first study was a uniform mesh refinement using HEX elements and a FOS calculated from the maximum principal stress. As shown in Table 4, the mesh seed size in the CUBIT meshing script was changed for each iteration starting with 5 mm and decreasing by 30% each iteration. Mesh convergence was not achieved after 7 size reductions. A key limitation is that Sierra outputs the stress results at the element integration points; thus, the surface stress is never fully resolved.

Table 4: Mesh refinement study results for nominal input parameters.

Element	Seed [mm]	Elements	min(FOS)
C3D8R	5	13,332	1.775
C3D8R	3.5	37,417	1.654
C3D8R	2.45	103,312	1.642
C3D8R	1.715	309,897	1.584
C3D8R	1.2	902,447	1.550
C3D8R	0.84	2,633,406	1.503
C3D8R	0.588	7,657,335	1.414
C3D8R	0.4116	22,270,512	1.315

Next, the same HEX mesh was utilized along with the addition of SHELL elements over the surface. The SHELL elements share nodes with the HEX mesh and are defined with a very small thickness (e.g. 0.5 µm) to provide no additional stiffness. As a result, the stress state for these elements is naturally extrapolated to the surface of the block. The same mesh refinement produces a much more consistent FOS result although mesh convergence was not achieved.

A study on the dynamic response of the SNS mercury target found that tetrahedral C3D10 elements provided a consistently 10% higher predicted stress than hexahedral C3D8 [20]. With this knowledge, a further discretization study was performed using TET C3D10 elements with a TRI3 shell surface. First, a "fixed core" study used a varying surface mesh with a 1.05 geometric growth rate to a fixed 1.2 [mm] nominal mesh in the core of the volume. Second, a "fixed surface" study used a fixed 1.2 [mm] nominal surface mesh with a 1.05 geometric growth rate to a larger interior mesh in the core of the volume. As Figure 21 shows, fatigue FOS results were around 4% lower than the corresponding HEX meshes. Stable mesh convergence was still not achieved, however, and the simulation time increased with the increased node count. With this in mind, the UQ analyses presented in this report use a 1.2 [mm] nominal HEX C3D8R mesh with shell surface elements.

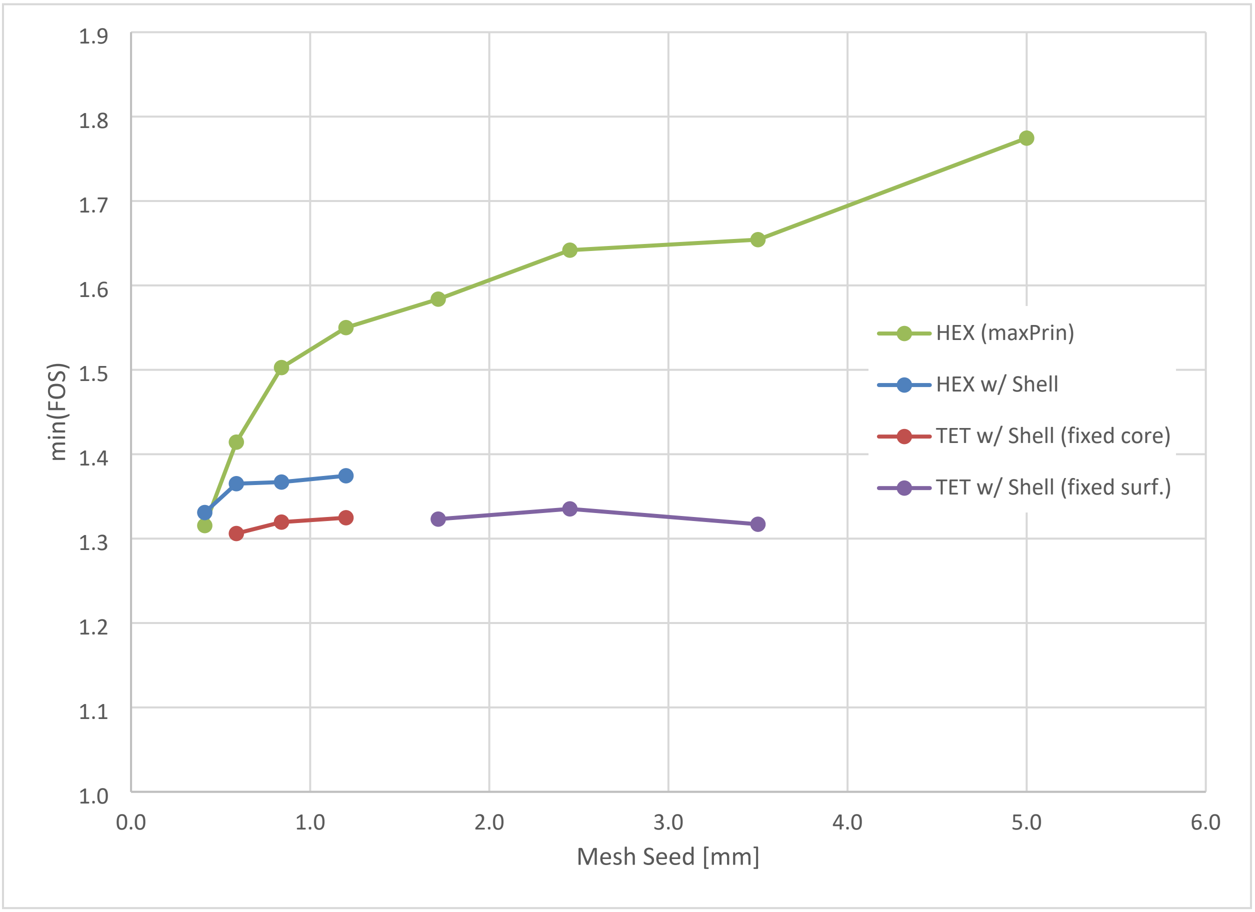


Figure 21: Mesh refinement study results for nominal input parameters.

5.2 UNCERTAINTY QUANTIFICATION VIA INTERVAL ANALYSIS

The UQ study successfully completed after 76 iterations. Figure 22 shows a histogram of the various fatigue FOS results that were computed. The epistemic interval analysis yielded a bound on the fatigue FOS as [0.462; 3.132]. It should be realized that the UQ analysis did not consider uncertainties due to modeling (e.g. convection approximation) as well as proton beam uncertainties (e.g. beam focus and beam offset). These effects would be expected to further widen the UQ interval.

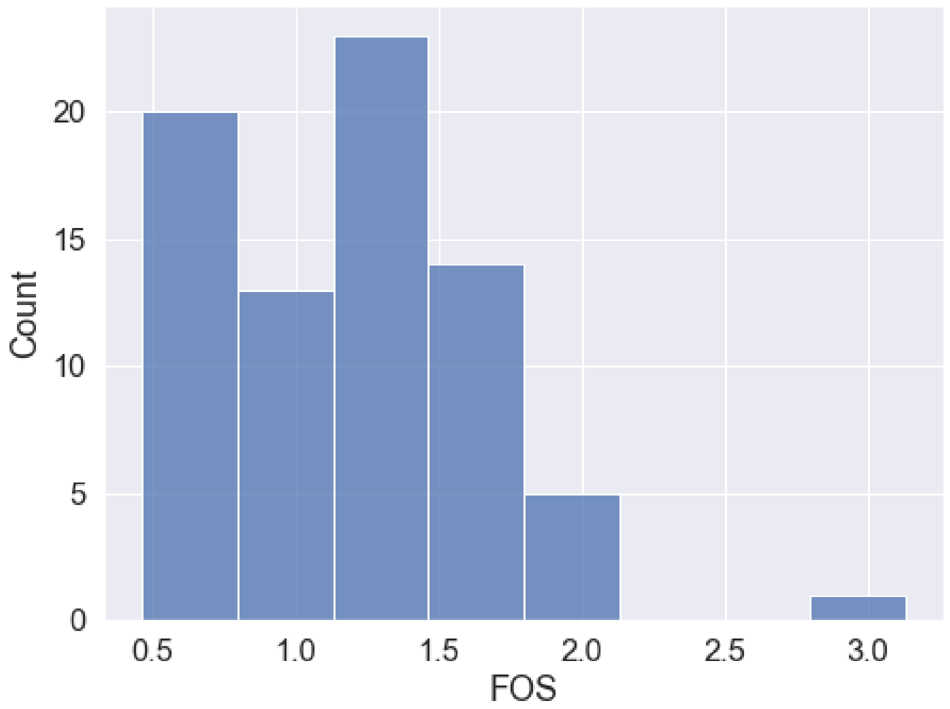


Figure 22: Histogram of fatigue FOS results from 76 iterations in a Dakota epistemic interval analysis.

Table 5 gives the parameters associated with the UQ interval bound on fatigue FOS and compares them with the input parameters. The low bound is most interesting since it is the limiting performance case and any solution below unity represents a failed target. The worst case condition (a 0.46 FOS) is associated with the following:

- Low ultimate strength
- Low fatigue strength
- Low density
- High elastic modulus
- High Poisson's ratio
- Low thermal conductivity
- High coefficient of thermal expansion
- Low specific heat
- Low flow rate

These are in line with the sensitivity analysis presented in the following section. The astute eye will note the flow parameter is unchanged between the maximum and minimum FOS cases. As will be seen, this parameter has the least leverage on the simulation output and the UQ algorithm was not able to discern an appreciable impact.

Table 5: Comparison of input parameters for the UQ bounding cases against the defined ranges.

Symbol	Units	Lower Bound	Nominal	Upper Bound	min(FOS) = 0.46	max(FOS) = 3.13
SultT	[MPa]	200	300	400	201.2	398.8
K_95	[ ]	0.78	1	1.2	0.783	1.20
rho	[kg/m^3]	19033	19250	19361	19035	19359
E	[GPa]	398	398	477.6	477.1	398.6
poisson	[ ]	0.266	0.28	0.294	0.294	0.266
k	[W/m-°C]	78.3	87	175	78.9	174.4
CTE	[1/°C]	4.05E-6	4.5E-6	4.95E-6	4.94E-6	4.056E-6
cp	[J/kg-°C]	115.47	128.3	141.13	115.66	140.94
flow	[ ]	0.85	1	1.15	0.852	0.852

5.2.1 Solution at UQ Lower Bound

Figure 23 through Figure 26 give detailed results for the UQ lower bound solution. For reference, the simulation was rerun with the same parameters while the elastic modulus was reduced to nominal. The fatigue FOS increased by 20% (from 0.5 to 0.6). Clearly, the static strength and fatigue strength have the largest influence on the fatigue FOS output. This can be observed qualitatively in Figure 26 where restoring the static and fatigue strengths (i.e. the blue boundary lines) to their nominal values would restore the fatigue FOS to near unity.

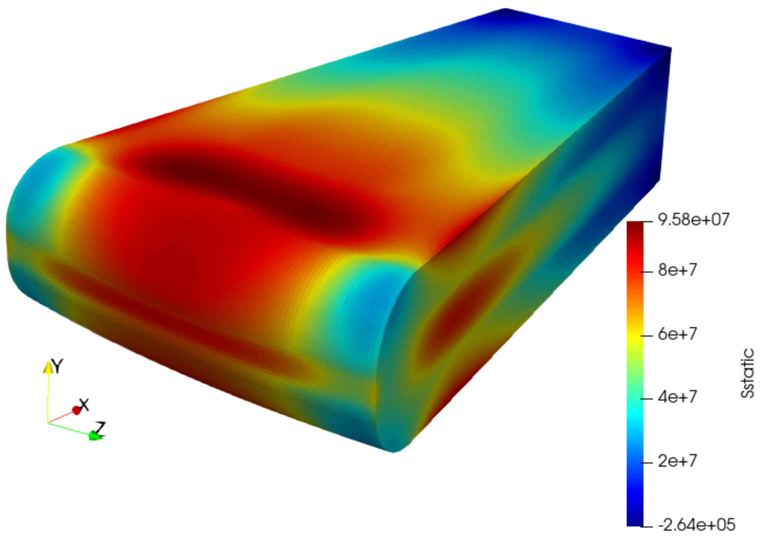


Figure 23: UQ lower bound solution – Static, maximum principal stress distribution [MPa].

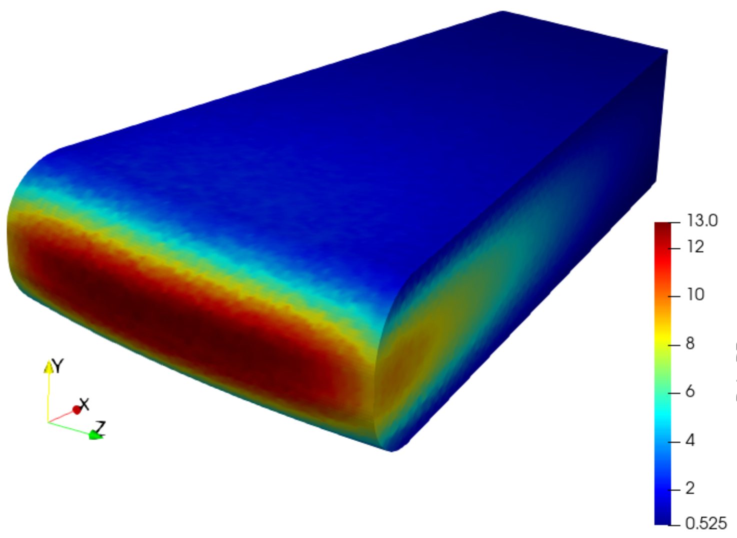


Figure 24: UQ lower bound solution – beam pulse instantaneous temperature rise [°C].

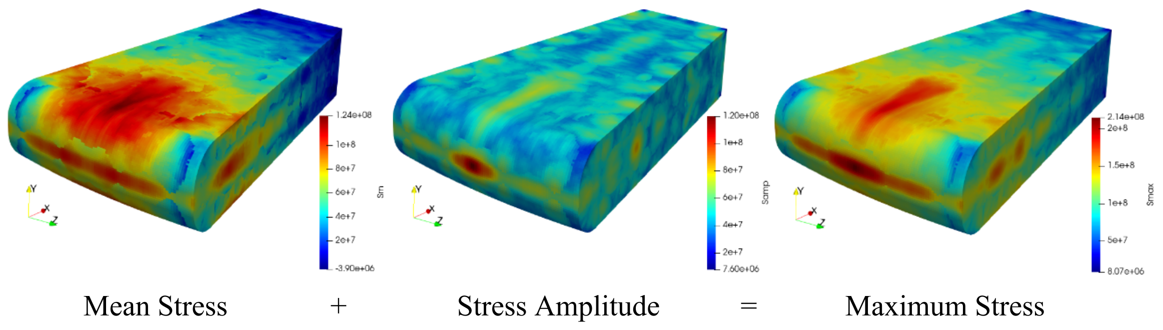


Figure 25: UQ lower bound solution – fatigue mean stress, worst case stress amplitude, and maximum stress [MPa].

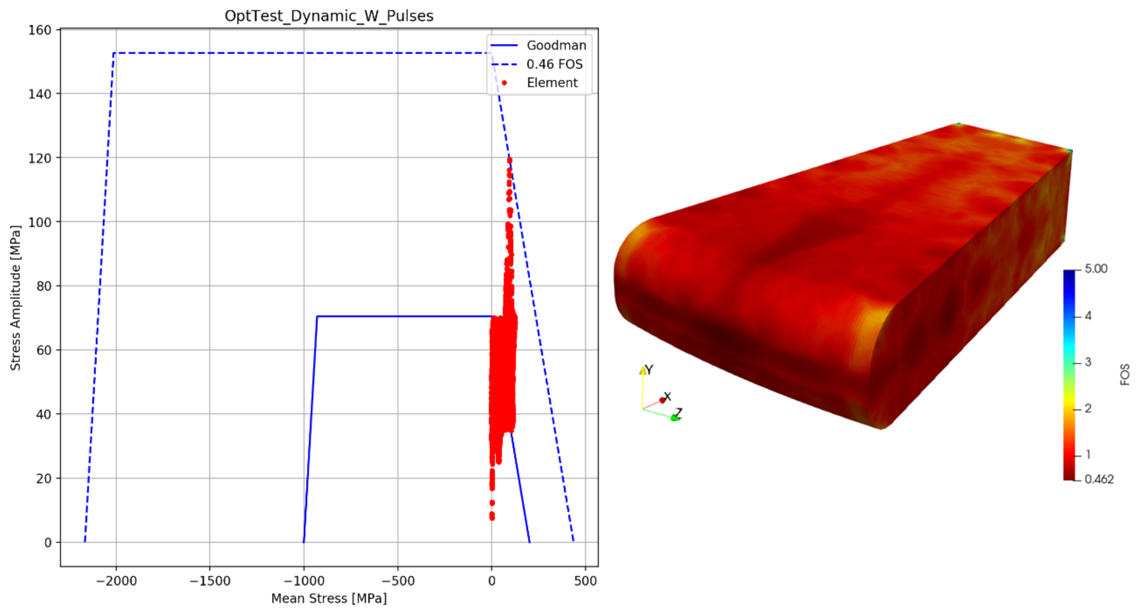


Figure 26: UQ lower bound solution – fatigue FOS contour plot and Goodman diagram.

5.2.2 Solution at UQ Upper Bound

Figure 27 through Figure 30 give detailed results for the UQ upper bound solution

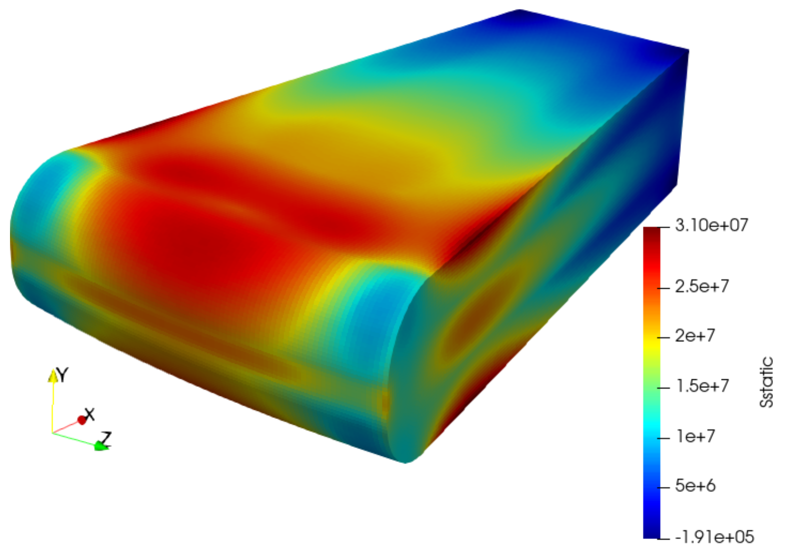


Figure 27: UQ upper bound solution – Static, maximum principal stress distribution [MPa].

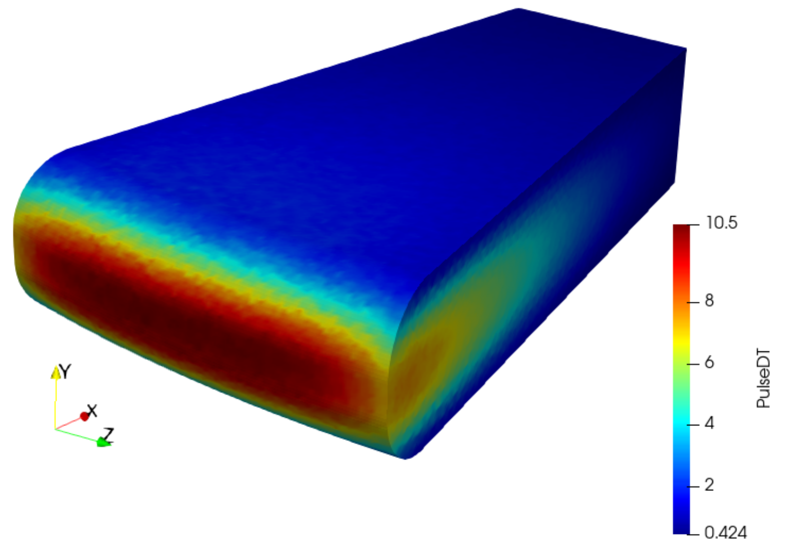


Figure 28: UQ upper bound solution – beam pulse instantaneous temperature rise [°C].

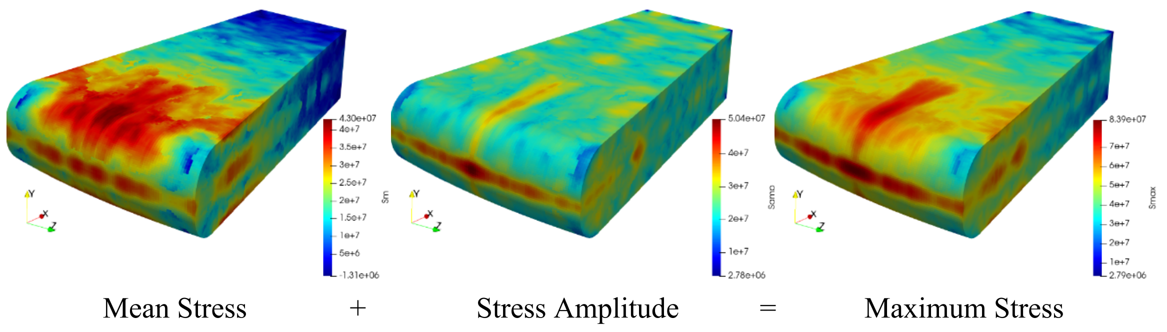


Figure 29: UQ upper bound solution – fatigue mean stress, worst case stress amplitude, and maximum stress [MPa].

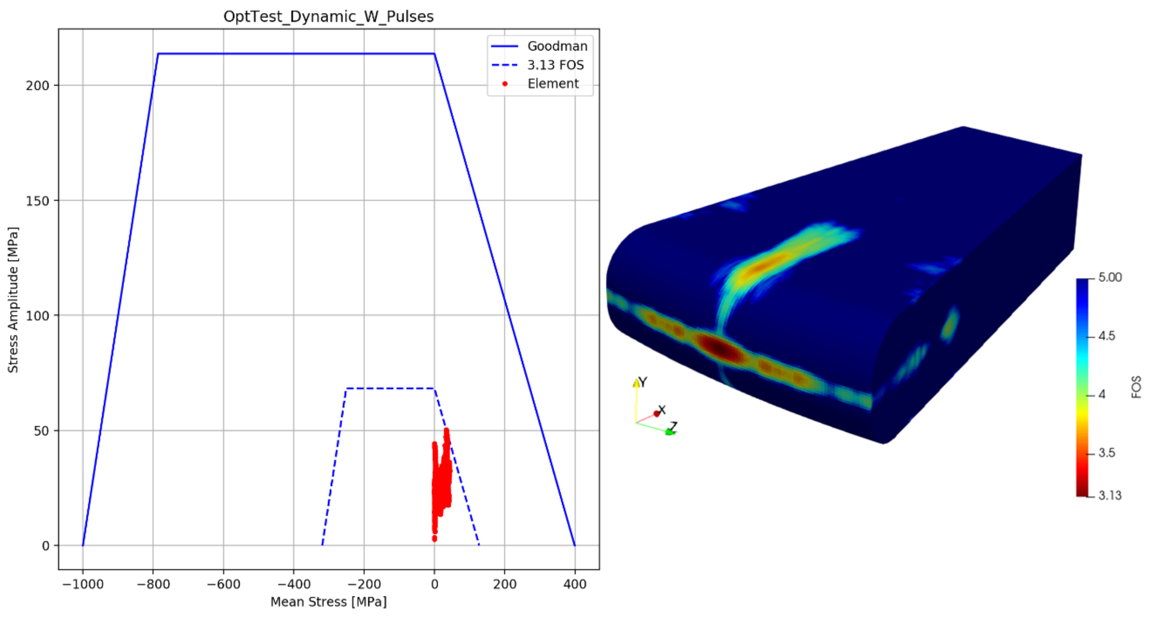


Figure 30: UQ upper bound solution – fatigue FOS contour plot and Goodman diagram.

5.3 PARAMETER SENSITIVITY ANALYSIS

The iteration history from the Dakota UQ analysis can also be used to illustrate parameter sensitivity. Figure 31, Figure 32, and Figure 33 use linear regression to determine the correlation between all parameters and between parameters and the output fatigue FOS. The flow parameter has the least effect on the analysis. Static strength and fatigue strength have the largest impacts on the analysis. The elastic modulus and Poisson's ratio both have comparable effects on the analysis. This is reasonable considering they both appear in the stiffness matrix for a finite element stress solution.

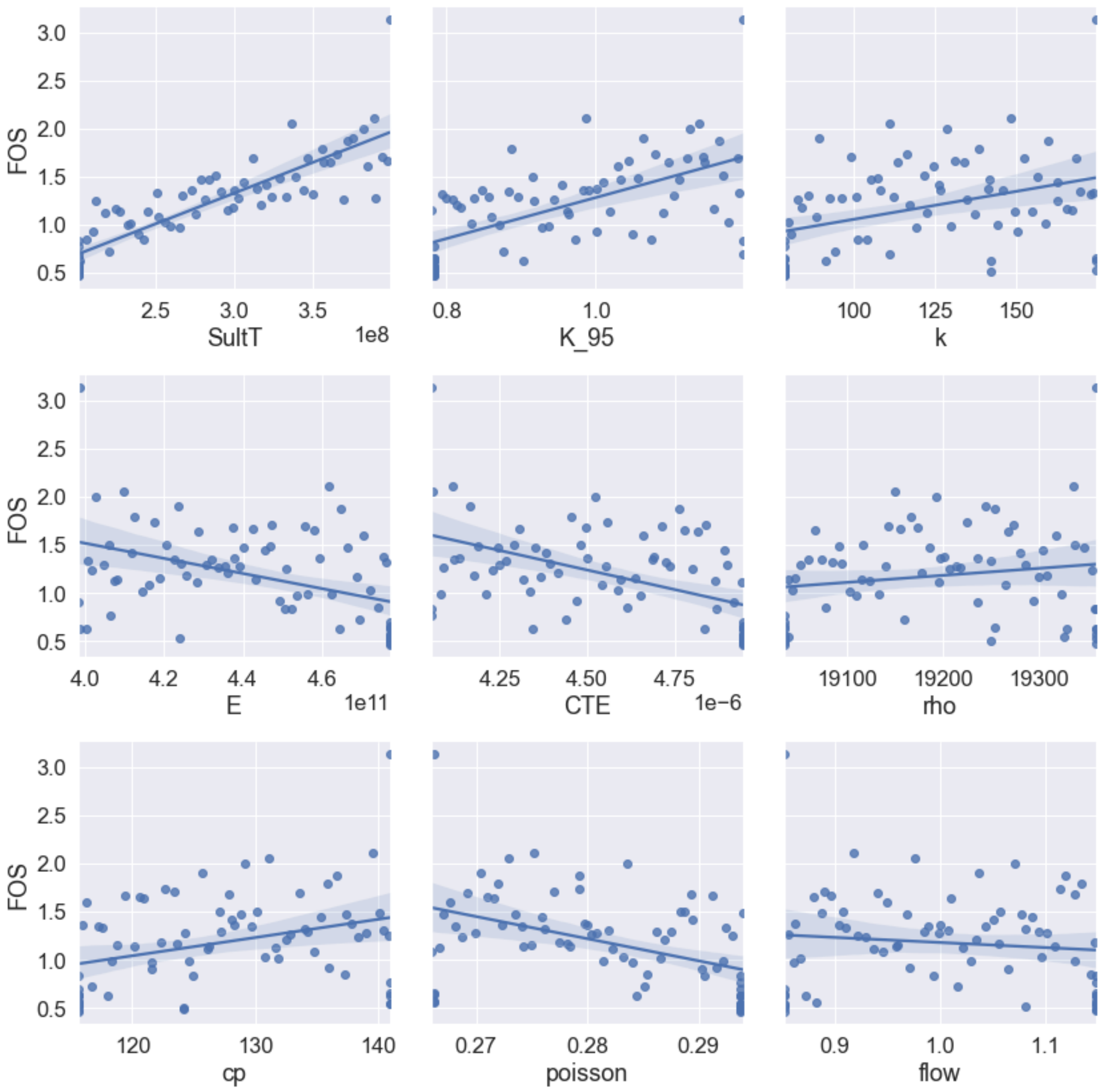


Figure 31: Graphical correlations of fatigue FOS against input parameters.

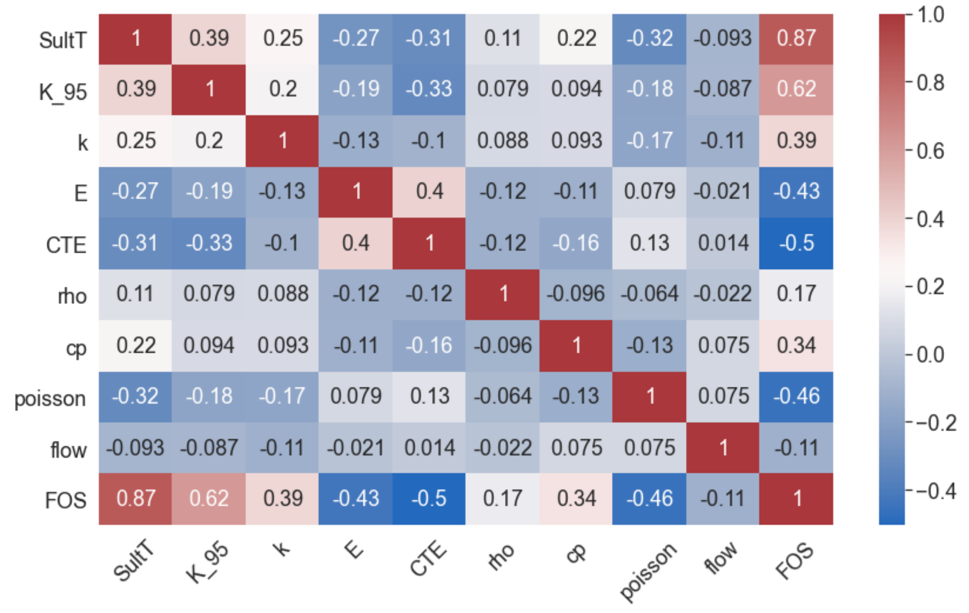


Figure 32: Relative linear regression correlations matrix.

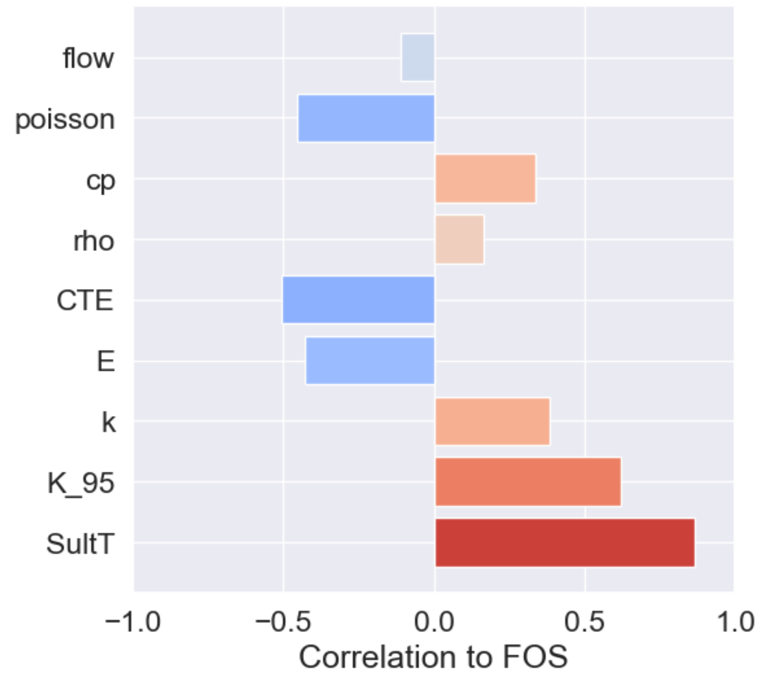


Figure 33: Relative linear regression correlations of input parameters against output fatigue FOS.

6. CONCLUSIONS

This study attempts to quantify the 10 year fatigue lifetime factor-of-safety (FOS) for a proposed monolithic tungsten target block. The current state of knowledge is presented for the input parameters (in the irradiated condition) along with best estimates of the variation in those parameters. UQ bounds the resulting tungsten fatigue FOS at [0.5, 3.4] using epistemic interval analysis. These bounds are expected to be wider due to model effects (discretization) and beam effects (focus, offset) that were not included. Mesh studies indicate a further FOS reduction of around 0.1 is plausible with refinement.

Parameter sensitivity underscores the importance of ultimate strength and fatigue strength of irradiated tungsten on the fatigue calculations. Recent R&D activities are revealing more variation and anisotropy in thick tungsten forgings compared to the smaller plate material used elsewhere. This study highlights the importance of understanding and characterizing those properties for more accurate prediction of target lifetime performance.

7. REFERENCES

[1] "The CUBIT™ Geometry and Mesh Generation Toolkit," Sandia National Laboratory, [Online]. Available: https://cubit.sandia.gov/. [Accessed 4 March 2022].

[2] H. Si, "TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator," *ACM Trans. on Mathematical Software,* vol. 41, no. 2, p. 36, 2015.

[3] SIERRA Solid Mechanics Team, "Sierra/SolidMechanics 4.56 User's Guide," Sandia Technical Report SAND2020-3547, March 2020.

[4] SIERRA Thermal/Fluid Development Team, "SIERRA Multimechanics Module: Aria User Manual – Version 4.56," Sandia Technical Report SAND2020-4000, April 2020.

[5] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, M. Khalil, K. A. Maupin, J. A. Monschke, E. M. Ridgway, A. A. Rushdi, D. T. Seidl, J. A. Stephens, L. P. Swiler and J. G. Winokur, "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.15 User's Manual," Sandia Technical Report SAND2020-12495, November 2021.

[6] J. J. B. Tipton, "Second Target Station (STS) Project: Sierra/SolidMechanics for Target Pulse Analysis," S03010000-TRT10000-R00, Oak Ridge National Laboratory, 22 October 2021.

[7] D. Wilcox, P. Loveridge, T. Davenne, L. Jones and D. Jenkins, "Stress levels and failure modes of tantalum-clad tungsten target at ISIS," *Journal of Nuclear Materials,* vol. 506, no. 1, pp. 76-82, 2018.

[8] J. Habainy, Y. Dai, Y. Lee and S. Iyengar, "Mechanical properties of tungsten irradiated with high-energy protons and spallation neutrons," *Journal of Nuclear Materials,* vol. 514, pp. 189-195, 2019.

[9] Schmidt and Ogden, "The Engineering Properties of Tungsten and Tungsten Alloys," Defense Military Information Center Report 191, 1963.

[10] J. Habainy, A. Lovberg, S. Iyengar, Y. Lee and Y. Dai, "Fatigue properties of tungsten from two different processing routes," *Journal of Nuclear Materials,* vol. 506, no. 1, pp. 83-91, 2018.

[11] "IMR Report Number 202103793 Preliminary," IMR Test Labs, Lansing, NY, 8 February 2022.

[12] J. A. Bannantine, J. J. Comer and J. L. Handrock, Fundamentals of Metal Fatigue Analysis, Englewood Cliffs, New Jersey: Prentice Hall, 1990.

[13] S. A. Maloy and E. J. Pitcher, "Materials issues associated with the design of the materials test station," *LA-UR-08-07052,* 2008.

[14] R. B. Jones, G. Hall, B. Marsden and C. A. English, "On the Irradiation Response of Be, W and Graphite for Proton Accelerator Applications," *National Nuclear Laboratory Literature Review, UK,* November 2013.

[15] J. Habainy, Y. Lee, K. Surreddi, A. Prosvetov, P. Simon, S. Iyengar, Y. Dai and M. Tomut, "Study of heavy ion beam induced damage in tungsten for high power target applications," *Nuclear*

Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms, vol. 439, pp. 7-16, 2019.

[16] Y. J. Lee, *Personal Communication,* 7 January 2022.

[17] J. Habainy, Y. Day, Y. Lee and S. Iyengar, "Thermal diffusivity of tungsten irradiated with protons up to 5.8 dpa," *Journal of Nuclear Materials,* vol. 509, pp. 152-157, 2018.

[18] A. Reza, H. Yu, K. Mizohata and F. Hofmann, "Thermal diffusivity degradation and point defect density in self-ion implanted tungsten," *Acta Materialia,* vol. 193, pp. 270-279, 2020.

[19] A. A. Giunta, L. P. Swiler, S. L. Brown, M. S. Eldred, M. D. Richards and E. C. Cyr, "The Surfpack Software Library for Surrogate Modeling of Sparse Irregularly Spaced Multidimensional Data," *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference,* p. 7049, 2006.

[20] S. Kaminskas, "Hexahedral and Tetrahedral Finite Element Mesh Comparison for 2 MW Mercury Target Simulations," SNS Neutron Technologies Division, PPUP-509-TR0001, R00, Oak Ridge National Laboratory, 11 April 2019.

APPENDIX A. Dakota Scripts

APPENDIX A.1 DAKOTA INPUT

```
environment
  results_output
    hdf5
  tabular_data
    tabular_data_file = 'Dakota_UQ_v3.dat'


method
    global_interval_est ego


variables
    continuous_interval_uncertain = 9
    num_intervals =      1          1        1       1         1         1         1       1          1
    interval_probabilities = 1.0   1.0      1.0     1.0       1.0       1.0       1.0     1.0        1.0
    descriptors         'SultT'    'K_95'   'k'     'E'       'CTE'     'rho'     'cp'    'poisson'  'flow'
#   initial_point =      300.0e6    1.0       87.0   398.0e9   4.5e-6   19250.0    128.3   0.28       1.0
    lower_bounds =       200.0e6    0.78      78.3   398.1e9   4.05e-6  19033.0    115.5   0.266      0.85
    upper_bounds =       400.0e6    1.20     175.0   477.6e9   4.95e-6  19361.0    141.1   0.294      1.15


interface
    fork asynchronous evaluation_concurrency = 3
    analysis_drivers = 'Dakota_Driver.sh'
    work_directory named 'UQiter3'
       copy_files 'templateV3/*'
       directory_tag


responses
  response_functions = 1
  descriptors  'FOS'
  no_gradients
  no_hessians
```

APPENDIX A.2 DAKOTA DRIVER SCRIPT

```
#!/bin/bash
# $1 and $2 are special variables in bash that contain the 1st and 2nd
# command line arguments to the script, which are the names of the
# Dakota parameters and results files, respectively.

params=$1
results=$2

###############################################################################
##
## Pre-processing Phase -- Generate/configure an input file for your simulation
##  by substiting in parameter values from the Dakota paramters file.
##
###############################################################################

dprepro $params postFatigue.template postFatigue.py
dprepro $params fitTemps.template fitTemps.py
dprepro $params OptTest_Thermal.template OptTest_Thermal.i
dprepro $params OptTest_Dynamic.template OptTest_Dynamic.i

###############################################################################
##
## Execution Phase -- Run your simulation
##
###############################################################################

#
# Track time
#
start=`date +%s`

#
# Generate Mesh
#
/data/Cubit-15.7/cubit -batch -nographics -nojournal OptTest_WBlock.jou > cubit.log
echo =
echo ======================================================================
echo Mesh completed...
echo ======================================================================
echo =

#
# Fit Neutronics Data onto Mesh
#
python fitTemps.py
echo =
echo ======================================================================
echo Neutronics data interpolation completed...
echo ======================================================================
echo =

#
# Heat Transfer Analysis
#
touch thermal.lck
qsub OptTest_Thermal.sh
sleep 60
while [ -f thermal.lck ]; do
  sleep 5
done
echo =
echo ======================================================================
echo Heat transfer analysis completed...
echo ======================================================================
echo =

#
# Temperature History File
```

```
#
python makeTemps.py
echo =
echo ==========================================================================
echo Temperature history file completed...
echo ==========================================================================
echo =

#
# Explicit Dynamic Structural Analysis
#
touch dynamic.lck
qsub OptTest_Dynamic.sh
sleep 60
while [ -f dynamic.lck ]; do
  sleep 5
done
echo =
echo ==========================================================================
echo Explicit dynamic structural analysis completed...
echo ==========================================================================
echo =

#
# Fatigue Analysis
#
python postFatigue.py
FOSmin=$(<Goodman_FOS.out)
echo =
echo ==========================================================================
echo Fatigue analysis completed...
echo ==========================================================================
echo =

#
# Remove Files to Prepare for Next Run
#
rm -f *.g
rm -f *.e
rm -f *.log
rm -f *.lck

echo =
echo ==========================================================================
echo Sierra results files removed...
echo ==========================================================================
echo =

#
# Track time
#
end=`date +%s`
runtime=$((end-start))
hours=$((runtime / 3600))
minutes=$(( (runtime % 3600) / 60 ))
seconds=$(( (runtime % 3600) % 60 ))
echo =
echo ==========================================================================
echo "Iteration Runtime: $hours:$minutes:$seconds (hh:mm:ss)"
echo ==========================================================================
echo =

##############################################################################
##
## Post-processing Phase -- Extract (or calculate) quantities of interest
##  from your simulation's output and write them to a properly-formatted
##  Dakota results file.
##
##############################################################################

echo "$FOSmin FOS" > $results
```

APPENDIX A.3 DAKOTA RAW OUTPUT

	SultT	K_95	k	E	CTE	rho	cp	poisson	flow	FOS
%eval_id										
56	201234568	0.782593	78.896914	4.77E+11	0.000005	19035.025	115.65803	0.293827	0.851852	0.461754
58	201234568	0.782593	78.896914	4.77E+11	0.000005	19358.975	115.65803	0.293827	1.148148	0.46553
64	201234568	0.782593	78.896914	4.77E+11	0.000005	19035.025	124.19136	0.293827	1.148148	0.48834
65	201234568	0.782593	78.896914	4.77E+11	0.000005	19249.642	124.19136	0.293827	0.851852	0.496633
70	201234568	0.782593	142.16975	4.77E+11	0.000005	19035.025	115.65803	0.293827	1.081481	0.508808
59	201234568	0.782593	78.896914	4.24E+11	0.000005	19035.025	115.65803	0.293827	1.148148	0.523778
60	201234568	0.782593	174.40309	4.77E+11	0.000005	19358.975	115.65803	0.293827	0.851852	0.524985
57	201234568	0.782593	78.896914	4.77E+11	0.000005	19039.074	140.94198	0.293827	0.851852	0.537625
63	201234568	0.782593	78.896914	4.77E+11	0.000005	19326.58	140.94198	0.293827	1.148148	0.542894
62	201234568	0.782593	78.896914	4.77E+11	0.000005	19035.025	115.65803	0.266173	0.881481	0.560887
66	201234568	0.782593	78.896914	4.77E+11	0.000005	19358.975	115.65803	0.266173	1.148148	0.564566
61	201234568	0.782593	174.40309	4.77E+11	0.000005	19035.025	140.94198	0.293827	1.148148	0.619856
13	201533518	0.902478	91.341978	4.64E+11	0.000005	19329.242	117.95137	0.284441	0.871981	0.624736
74	201234568	0.782593	142.16975	3.99E+11	0.000005	19358.975	115.65803	0.293827	1.148148	0.625321
73	201234568	0.782593	78.896914	4.01E+11	0.000004	19358.975	115.65803	0.293827	0.851852	0.629297
68	201234568	0.782593	174.40309	4.77E+11	0.000005	19253.691	115.65803	0.266173	0.851852	0.643602
67	201234568	0.782593	78.896914	4.77E+11	0.000005	19035.025	140.94198	0.266173	1.148148	0.649233
72	201234568	1.197407	111.13025	4.77E+11	0.000005	19035.025	115.65803	0.293827	0.851852	0.689526
21	220477789	0.875434	94.508529	4.70E+11	0.000004	19159.874	116.6806	0.285146	1.016751	0.71788
69	201234568	0.782593	78.896914	4.06E+11	0.000004	19035.025	140.94198	0.293827	1.148148	0.766403
71	201234568	1.197407	78.896914	4.51E+11	0.000004	19358.975	115.65803	0.293827	1.148148	0.827254
43	242576919	0.97211	104.2716	4.52E+11	0.000005	19358.601	124.948	0.290551	0.994085	0.836997
7	205608560	1.075097	101.23692	4.74E+11	0.000005	19077.457	137.28702	0.285432	1.14355	0.845768
32	238795198	1.050074	80.705512	3.99E+11	0.000005	19235.721	121.52816	0.290331	1.064846	0.897569
28	210164978	1.001779	150.52589	4.49E+11	0.000004	19294.856	136.03466	0.291637	0.97063	0.921748
19	264855947	0.928666	119.31415	4.54E+11	0.000005	19109.233	121.57142	0.28403	0.859813	0.964339
27	259604364	0.936415	129.87477	4.56E+11	0.000004	19133.463	118.35769	0.292203	1.127825	0.978312
6	231887184	0.870443	144.3623	4.62E+11	0.000004	19323.549	124.56055	0.281397	1.029022	0.991117
51	233680286	0.832878	159.17314	4.14E+11	0.000004	19103.21	131.84324	0.286535	0.866583	1.007583
25	256307841	1.177625	79.97136	4.72E+11	0.000005	19042.816	130.79858	0.283223	1.096189	1.029945
53	251359888	0.859848	88.593944	4.16E+11	0.000005	19264.501	134.79324	0.266003	0.945573	1.078965
23	275442012	0.963826	137.24794	4.28E+11	0.000005	19196.201	126.15731	0.282304	0.936412	1.106812
41	217766776	1.091544	122.38244	4.08E+11	0.000005	19122.989	131.63421	0.266635	1.021301	1.124013
55	227109556	1.019723	155.00362	4.08E+11	0.000005	19038.321	126.21648	0.278388	0.95859	1.137868
46	244716154	0.96211	149.44092	4.43E+11	0.000004	19114.616	120.13581	0.274178	1.109158	1.139012
16	295773790	0.780495	167.11523	4.19E+11	0.000005	19045.02	118.79764	0.274891	0.959966	1.144067
4	224472911	1.1584	165.48171	4.69E+11	0.000004	19300.369	123.6736	0.278177	1.056559	1.15954
1	298994031	0.819301	84.143335	4.26E+11	0.000004	19308.191	122.28116	0.277453	1.147296	1.173223
26	317300504	0.814148	117.10417	4.36E+11	0.000004	19177.668	132.52506	0.286919	1.034336	1.207195
33	211569777	0.917821	162.73541	4.02E+11	0.000004	19354.992	138.41323	0.268729	0.928465	1.238804
34	369327412	0.808571	134.69687	4.51E+11	0.000005	19207.042	140.83454	0.293047	0.921318	1.254461
49	281270891	0.944616	82.964457	4.34E+11	0.000004	19217.992	132.84374	0.280388	0.854866	1.256756
42	302772760	0.836473	96.335768	4.35E+11	0.000005	19213.841	139.01168	0.26994	1.102109	1.274904
52	389708075	0.799038	92.775262	4.39E+11	0.000005	19140.297	124.33518	0.28086	0.998475	1.277365
47	332756398	0.856307	112.5236	4.05E+11	0.000005	19286.425	134.24979	0.286261	0.985301	1.283998
44	323811832	0.895722	100.84475	4.31E+11	0.000004	19051.517	127.25244	0.287544	1.094255	1.284618
20	266870596	1.10471	86.15765	4.24E+11	0.000004	19094.356	140.48013	0.281921	1.007229	1.302341
39	349848363	0.794052	172.84577	4.76E+11	0.000005	19084.543	134.09231	0.276191	1.080979	1.311264
5	250599406	1.192413	173.7056	4.01E+11	0.000004	19249.993	117.56991	0.292509	0.910298	1.327206
9	291238369	0.884024	169.87158	4.23E+11	0.000005	19059.756	129.65553	0.268164	0.988952	1.341154
48	272550696	0.983275	146.03514	4.32E+11	0.000004	19072.789	117.23905	0.274088	1.042953	1.350641
3	343918032	0.847586	108.35214	4.59E+11	0.000004	19236.536	115.91653	0.272303	0.903091	1.352335
76	300000000	0.99	126.65	4.38E+11	0.000005	19197	128.3	0.28	1	1.360953
37	314681484	1.001305	141.54554	4.75E+11	0.000005	19200.698	137.85761	0.279757	0.862094	1.367352

12	320498207	0.954524	126.09814	4.12E+11	0.000004	19280.405	128.11186	0.289502	1.05132	1.4187
54	306190000	1.011043	162.54426	4.45E+11	0.000005	19303.793	135.30368	0.275965	1.088128	1.443446
17	279064074	1.034223	105.26766	4.40E+11	0.000004	19186.434	137.38543	0.267071	0.96847	1.470393
15	283905670	1.112766	141.78289	4.67E+11	0.000004	19347.453	128.85493	0.273525	1.077131	1.471344
8	328483051	1.056954	107.3181	4.47E+11	0.000004	19090.865	140.13206	0.293966	0.886815	1.481731
29	338515976	0.915741	156.42725	4.06E+11	0.000004	19337.332	127.06828	0.288737	1.057508	1.491587
11	287704422	1.171714	121.65284	4.21E+11	0.000004	19116.606	130.1106	0.288349	0.907638	1.504696
40	384911538	1.030636	124.71439	4.70E+11	0.000005	19317.609	116.30904	0.267656	0.949145	1.600553
31	356525557	1.09801	113.58783	4.29E+11	0.000005	19268.078	120.91207	0.271583	1.010197	1.642227
14	360578080	1.146766	133.86293	4.58E+11	0.000005	19065.927	120.54695	0.270937	0.878803	1.654014
36	397561964	1.045197	131.25017	4.42E+11	0.000004	19155.843	119.38095	0.291282	0.895572	1.660151
18	346292917	1.122237	168.48914	4.37E+11	0.000004	19172.982	127.8824	0.289326	1.129	1.683837
38	312167016	1.191837	152.41182	4.56E+11	0.000005	19142.357	133.56702	0.269248	0.939878	1.690932
22	393663304	1.145164	99.067172	4.47E+11	0.000005	19272.826	123.39496	0.276977	0.889053	1.702958
24	365057702	1.079774	116.45938	4.17E+11	0.000005	19225.514	122.62511	0.279221	1.113796	1.732697
2	355808446	0.887784	138.73307	4.13E+11	0.000004	19166.02	135.91764	0.271922	1.13473	1.789517
50	372352427	1.165375	159.74112	4.65E+11	0.000005	19253.786	136.60969	0.279261	1.119183	1.869445
30	375212744	1.064487	89.33246	4.24E+11	0.000004	19244.666	125.6492	0.270356	1.037189	1.90329
35	381791497	1.126986	128.71336	4.03E+11	0.000005	19192.338	129.11877	0.283523	1.070788	2.001533
45	336690738	1.138834	111.11755	4.10E+11	0.000004	19149.751	131.11856	0.272896	0.975732	2.052814
10	389044893	0.987402	148.37872	4.62E+11	0.000004	19335.871	139.5959	0.275245	0.918103	2.10221
75	398765432	1.197407	174.40309	3.99E+11	0.000004	19358.975	140.94198	0.266173	0.851852	3.13232

APPENDIX B. Mesh Generation Scripts

APPENDIX B.1 CUBIT MESH GENERATION SCRIPT

```
## /data/Cubit-15.7/bin/clarox
## Cubit Version 15.7
## Cubit Build b6d5011

import acis "OptTest_WBlock.sat" nofreesurfaces attributes_on separate_bodies

volume 1 size 0.0012
mesh volume 1
create group "surfelems"
group "surfelems" add surface 1 2 3 4 5 6 7 8
group "allelems" add hex all
set duplicate block elements off
block 1 add group 2
block 1 name "W_BLOCK_SURF"
set duplicate block elements off
block 2 add group 3
block 2 name "W_BLOCK_INT"

sideset 1 add surface 3
sideset 1 name "W_SURF_1"

sideset 2 add surface 5
sideset 2 name "W_SURF_2"

sideset 3 add surface 4
sideset 3 name "W_SURF_3"

sideset 4 add surface 2
sideset 4 name "W_SURF_4"

sideset 5 add surface 8
sideset 5 name "W_SURF_5"

sideset 6 add surface 1
sideset 6 name "W_SURF_6"

sideset 7 add surface 7
sideset 7 name "W_SURF_7"

sideset 8 add surface 6
sideset 8 name "W_SURF_8"

nodeset 1 add node all
nodeset 1 name "NS.W_NODES"
set exodus netcdf4 off
set large exodus file on
export mesh "OptTest_WBlock.g" overwrite
```

APPENDIX C. Neutronics Heating Interpolation Scripts

APPENDIX C.1 NEUTRONICS HEATING INTERPOLATION SCRIPT

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

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


#
# Target Parameters
#
WDensity = {rho}       # [kg/m^3]
WSpecHeat = {cp}        # [J/kg-K]
WThermCond = {k}        # [W/m-K]
PulseFreq = 15.0 / 21.0  # [s^-1]


#
# Load Target Mesh
#
mesh = exodus('OptTest_WBlock.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(1)
tungstenIndex = mesh.get_node_set_nodes(1)
tungstenID = tungstenIndex - 1
tungstenCoord = targetCoord[tungstenID]

mesh.close()

#
# Load Source Mesh and Energy Deposition and Fit the Data
#
source = np.loadtxt(open('o10b_W_combined.csv','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]*0.01                          # [m]
sourceVal = source[:,3]*1.0e6*PulseFreq                  # [W/m^3]
WSteady= fitData(sourceCoord,sourceVal,tungstenID,tungstenIndex,tungstenCoord)

source = np.loadtxt(open('o10b_W_1.csv','rb'), delimiter=',', skiprows=0)
sourceCoord = source[:,:3]*0.01                          # [m]
sourceVal = source[:,3]*1.0e6/WDensity/WSpecHeat         # [degC/pulse]
WPulse= fitData(sourceCoord,sourceVal,tungstenID,tungstenIndex,tungstenCoord)


#
# Save Heat Generation Values to New Exodus File
#
addGlobalVariables = []
addNodeVariables = ["VolHeatGen"]
addElementVariables = []

# copy exodus file and add values to all nodes
exo1 = copyTransfer('OptTest_WBlock.g','OptTest_VolHeatGen.g','ctype',
                    addGlobalVariables,addNodeVariables, addElementVariables)
exo1.put_node_variable_values('VolHeatGen',1,WSteady['Val'])
exo1.put_time(1,1.0)
exo1.close()

#
# Save Pulse DT Values to New Exodus File
#
```

```
addGlobalVariables = []
addNodeVariables = ["PulseDT"]
addElementVariables = []

# copy exodus file and add values to all nodes
exo2 = copyTransfer('OptTest_WBlock.g','OptTest_PulseDT.g','ctype',
                    addGlobalVariables,addNodeVariables, addElementVariables)
exo2.put_node_variable_values('PulseDT',1,WPulse['Val'])
exo2.put_time(1,1.0)
exo2.close()
```

APPENDIX C.2 POINT CLOUD INTERPOLATION PYTHON MODULE

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

def fitData(sourceCoord,sourceVal,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
    #
    outVal = np.multiply(sourceVal[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')
    outerVal = knn.fit(sourceCoord,sourceVal).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))
    finalVal = np.hstack((outVal,outerVal))

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

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

    final['Index'] = finalIndex
```

```
final['ID'] = finalID
final['x'] = finalCoord[:,0]
final['y'] = finalCoord[:,1]
final['z'] = finalCoord[:,2]
final['Val'] = finalVal

final.sort(order='Index')

return final
```

APPENDIX D. Heat Transfer Analysis Scripts

APPENDIX D.1 SIERRA ARIA HEAT TRANSFER ANALYSIS INPUT

```
begin sierra


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

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

    begin parameters for block W_BLOCK_SURF
      material W_Irrad
    end

    begin parameters for block W_BLOCK_INT
      material W_Irrad
    end

  end


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

  begin aria material W_Irrad
    density = constant RHO = {rho}
    heat conduction = Basic
    specific heat = constant CP = {cp}
    thermal conductivity = constant K = {k}
    shell thickness = constant T = 5.0e-7
  end


  ##---------------------------------------------------------------------
  ## MODEL DEFINITION & SOLVER COMMANDS
  ##---------------------------------------------------------------------

  begin trilinos equation solver myLinearSolver
    maximum iterations = 1000
    residual norm tolerance = 1.0e-8
  end

  begin AZTEC equation solver Solve_Temperature_CG
    # NOTE: Conjugate Gradient with Jacobi preconditioning
    #       Good for nonlinear problems, symmetric matrices
    solution method = CG
    preconditioning method = JACOBI
    maximum iterations = 5000 #500
    residual norm tolerance = 1e-10
    residual norm scaling = ANORM # default
  end

  begin AZTEC equation solver Solve_Temperature_GMRES
    # NOTE: GMRES with Jacobi preconditioning
    #       Good for nonlinear problems, nonsymmetric matrices
    solution method = GMRES
    preconditioning method = JACOBI
    maximum iterations = 9000
    residual norm tolerance = 1e-8
    residual norm scaling = None
  end
```

```
##----------------------------------------------------------------------
## LOAD VOLUMETRIC HEAT GENERATION DATA
##----------------------------------------------------------------------
begin finite element model VolHeatGenMesh
  database name = OptTest_VolHeatGen.g
  database type = exodusII
  begin parameters for block W_BLOCK_SURF
    material W_Irrad
  end
  begin parameters for block W_BLOCK_INT
    material W_Irrad
  end
end


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

begin procedure HEATING

  begin solution control description
    use system Main
    begin system Main
      #Simulation Max Global Iterations = 1000
      begin sequential MySolveBlock
        transfer NeutronicsHeating
        advance TargetBlock
      end
    end
  end

  begin transfer NeutronicsHeating
    copy volume nodes from InputOutput to TargetBlock
    send field VolHeatGen state none to VolHeatGenSurf state none
    send field VolHeatGen state none to VolHeatGenInt state none
  end

  begin input_output region InputOutput
    fixed time
    use finite element model VolHeatGenMesh
  end

  begin aria region TargetBlock

    use finite element model TargetBlockMesh

    nonlinear solution strategy = newton
    use dof averaged nonlinear residual
    maximum nonlinear iterations = 20
    nonlinear residual tolerance = 1.0e-6
    use linear solver Solve_Temperature_CG

    EQ Energy for Temperature on W_BLOCK_SURF using Q1 with DIFF SRC
    EQ Energy for Temperature on W_BLOCK_INT using Q1 with DIFF SRC

    IC Const at W_BLOCK_SURF Temperature = 35.0
    IC Const at W_BLOCK_INT Temperature = 35.0

    BC Flux for Energy on W_SURF_1 = Nat_Conv T_REF = 35.0 H = {7000.0*flow}
    BC Flux for Energy on W_SURF_2 = Nat_Conv T_REF = 35.0 H = {4500.0*flow}
    BC Flux for Energy on W_SURF_3 = Nat_Conv T_REF = 35.0 H = {3500.0*flow}
    BC Flux for Energy on W_SURF_4 = Nat_Conv T_REF = 35.0 H = {5500.0*flow}
    BC Flux for Energy on W_SURF_5 = Nat_Conv T_REF = 35.0 H = {7000.0*flow}
    BC Flux for Energy on W_SURF_6 = Nat_Conv T_REF = 35.0 H = {8000.0*flow}
    BC Flux for Energy on W_SURF_7 = Nat_Conv T_REF = 35.0 H = {7000.0*flow}
    BC Flux for Energy on W_SURF_8 = Nat_Conv T_REF = 35.0 H = {9900.0*flow}

    user field real node scalar VolHeatGenSurf on W_BLOCK_SURF
    user field real node scalar VolHeatGenInt on W_BLOCK_INT
```

```
      begin volume heating myVolHeatGenSurf
        add volume W_BLOCK_SURF
        #value = 10000.0
        nodal variable = VolHeatGenSurf
      end

      begin volume heating myVolHeatGenInt
        add volume W_BLOCK_INT
        #value = 10000.0
        nodal variable = VolHeatGenInt
      end

      begin results output
        database name = %B.e
        At Step 1 Increment = 1
        Nodal Variables = Solution->Temperature as TEMP
      end

    end aria region TargetBlock

  end procedure Heating

end sierra
```

APPENDIX D.2 SIERRA ARIA HEAT TRANSFER ANALYSIS EXECUTION SCRIPT

```
#!/bin/sh
#PBS -V
#PBS -j oe
#PBS -m abe
#PBS -l nodes=6:ppn=8
#PBS -N Sierra_Aria_OptTest
# #PBS -M tvj@ornl.gov
#PBS -q all
# #PBS -W depend=afterany:6923

cd $PBS_O_WORKDIR

# Define particulars of this run:
INPUT_FILENAME=OptTest_Thermal.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 aria -i $INPUT_FILENAME

echo Starting sierra pre aria

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

echo Starting sierra run aria

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

echo Starting sierra post aria

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

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

exit
```

APPENDIX E. Structural Analysis Scripts

APPENDIX E.1 SIERRA TEMPERATURE FIELD COMPILATION SCRIPT

```
#=======================================================================
# CREATE TEMPERATURE FIELDS FOR SIERRA DYNAMIC PULSE ANALYSIS
#=======================================================================
import numpy as np
from exodus import exodus, copyTransfer


#
# Load Mesh with Steady-State Temperatures
#
mesh = exodus('OptTest_Thermal.e',mode='r',array_type='numpy')

#mesh.get_node_id_map()
# By inspection, it seems that nodal values in a results file
# are sorted sequentially.  The mesh files, however, have a
# different sorting.

TempSS = mesh.get_node_variable_values('TEMP',1)

mesh.close()


#
# Load Mesh with Beam Pulse Temperatures
#
mesh = exodus('OptTest_PulseDT.g',mode='r',array_type='numpy')

MeshMap = mesh.get_node_id_map()

TempPulse = mesh.get_node_variable_values('PulseDT',1)

mesh.close()


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

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

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

temps[:,0] = 30.0  # At end of life, HIP prestress is assumed relaxed.
temps[:,1] = 30.0
temps[:,2] = TempSS[MeshMap-1]
temps[:,3] = TempSS[MeshMap-1] + TempPulse
temps[:,4] = TempSS[MeshMap-1] + TempPulse

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()
```

APPENDIX E.2 SIERRA STRUCTURAL ANALYSIS INPUT

```
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 - Tungsten
  ##--------------------------------------------------------------------

  begin function CTE_W
    type is analytic
    # Initial Temp is the Reference Temp
    evaluate expression is "{CTE} * x"
  end

  begin property specification for material W_Elastic
    density = {rho}
    THERMAL LOG STRAIN FUNCTION = CTE_W
    begin parameters for model ELASTIC
      youngs modulus = {E}
      poissons ratio = {poisson}
    end
  end


  ##--------------------------------------------------------------------
  ## MESH DEFINITION
  ##--------------------------------------------------------------------
  begin shell section SHELL_1
    thickness = 5.0e-7
  end

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

    begin parameters for block W_BLOCK_SURF
      material W_Elastic
      solid mechanics use model ELASTIC
      section = SHELL_1
    end

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

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

    begin parameters for block W_BLOCK_SURF
      material W_Elastic
      solid mechanics use model ELASTIC
      section = SHELL_1
    end

    begin parameters for block W_BLOCK_INT
      material W_Elastic
      solid mechanics use model ELASTIC
    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 = 0.0104007
    end

    begin presto region TargetBlock
      use finite element model TargetBlockMesh

      begin results output
        database name = %B_HIP_SS.e
        at time 0.0 increment = 0.001
        termination time = 0.01
        Nodal Variables = Temperature as TEMP
        Element Variables = Stress as Stress on W_BLOCK_INT
        Element Variables = max_principal_stress as MaxPrin on W_BLOCK_INT
        Element Variables = memb_stress as MembStress on W_BLOCK_SURF
        Element Variables = bottom_stress as BotStress on W_BLOCK_SURF
        Element Variables = top_stress as TopStress on W_BLOCK_SURF
        Element Variables = unrotated_stress as UnRotStress on W_BLOCK_SURF
        Element Variables = transform_shell_stress as TransStress on W_BLOCK_SURF
      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
        Element Variables = transform_shell_stress as TransStress
      end

      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_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
```

APPENDIX E.3 SIERRA STRUCTURAL ANALYSIS EXECUTION SCRIPT

```
#!/bin/sh
#PBS -V
#PBS -j oe
#PBS -m abe
#PBS -l nodes=6:ppn=8
#PBS -N Sierra_OptTest
# #PBS -M tvj@ornl.gov
#PBS -q all
# #PBS -W depend=afterany:6923

cd $PBS_O_WORKDIR

# Define particulars of this run:
INPUT_FILENAME=OptTest_Dynamic.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

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

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

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

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

exit
```

APPENDIX F. Fatigue Analysis Scripts

APPENDIX F.1 PYTHON FATIGUE ANALYSIS SCRIPT

```
import numpy as np
from exodus import exodus, copyTransfer
import exodusCalcs
import matplotlib.pyplot as plt



SultT = {SultT}   # Ultimate tensile strength of irradiated tungsten [Pa]
SultC = 1000.E+6  # Ultimate compressive strength of irradiated tungsten [Pa]

#---------------------------------------------------------------
# Calculate fatigue strength of irradiated tungsten
# See <tungstenFatigue-Irradiated-Rev2.ipynb>
#---------------------------------------------------------------
Life = 10.0 # years
Year_to_Sec = 365.0*24.0*3600.0 # sec in a year
Uptime = 5000.0 / (365.0*24.0) # hours operation / year
PulseFreq = 15.0 # pulses / sec
NumTarget = 21.0
Pulses = Life * Year_to_Sec * Uptime * PulseFreq / NumTarget

Sfprime = 568.592632E+6
b = -0.04150804

K_irr = SultT / Sfprime
K_95 = {K_95}

Slife = K_95 * K_irr * Sfprime * (2 * Pulses)**b
#---------------------------------------------------------------



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


#confirm block ID and name
ei.get_elem_blk_ids()
ei.get_elem_blk_names()
blkID = 1


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

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


"""
CALCULATE STATIC PRINCIPAL STRESSES
===============================================================================
"""
jj = 0
IP = 1
S_xx = ei.get_element_variable_values(blkID, 'TransStress_xx_'+str(IP), jj+1)
S_yy = ei.get_element_variable_values(blkID, 'TransStress_yy_'+str(IP), jj+1)
S_xy = ei.get_element_variable_values(blkID, 'TransStress_xy_'+str(IP), jj+1)
maxPrin = (S_xx + S_yy)/2.0 + np.sqrt(((S_xx - S_yy)/2.0)**2.0 + S_xy**2.0)
minPrin = (S_xx + S_yy)/2.0 - np.sqrt(((S_xx - S_yy)/2.0)**2.0 + S_xy**2.0)
StressStatic = np.where(np.absolute(maxPrin) >= np.absolute(minPrin), maxPrin, minPrin)
```

```
"""
CALCULATE FOS BASED ON CRITICAL NORMAL PLANE THEORY
================================================================================
"""
# initialize stress vectors
CritPlaneDeg = np.ones([len(eleIDs)])*1.0e+100
CritPlaneFOS = np.ones([len(eleIDs)])*1.0e+100
CritPlaneMean = np.ones([len(eleIDs)])*1.0e+100
CritPlaneAmp = np.ones([len(eleIDs)])*1.0e+100

IP = 1

for Plane in range(0, 180, 10):

  # initialize stress vectors
  StressDynamicMin = np.ones([len(eleIDs)])*1.0e+100
  StressDynamicMax = np.ones([len(eleIDs)])*-1.0e+100

  for jj in range(1,numTimes+1):
    S_xx = ei.get_element_variable_values(blkID, 'TransStress_xx_'+str(IP), jj)
    S_yy = ei.get_element_variable_values(blkID, 'TransStress_yy_'+str(IP), jj)
    S_xy = ei.get_element_variable_values(blkID, 'TransStress_xy_'+str(IP), jj)
    S_n = (S_xx + S_yy)/2.0 \
           + (S_xx - S_yy)/2.0 * np.cos(2.0*Plane*np.pi/180.0) \
           + S_xy * np.sin(2.0*Plane*np.pi/180.0)
    StressDynamicMax = np.maximum(StressDynamicMax,S_n)
    StressDynamicMin = np.minimum(StressDynamicMin,S_n)

  Smean = (StressDynamicMax + StressDynamicMin) / 2.0
  Samp = (StressDynamicMax - StressDynamicMin) / 2.0
  FOS = 1.0 / (Samp/Slife + abs(Smean)/SultT) # Assumes 1st quadrant

  CritPlaneDeg = np.where(FOS < CritPlaneFOS, Plane, CritPlaneDeg)
  CritPlaneMean = np.where(FOS < CritPlaneFOS, Smean, CritPlaneMean)
  CritPlaneAmp = np.where(FOS < CritPlaneFOS, Samp, CritPlaneAmp)
  # finally, overwrite my test condition
  CritPlaneFOS = np.where(FOS < CritPlaneFOS, FOS, CritPlaneFOS)

ei.close()




"""
REPORT FACTORS OF SAFETY
================================================================================
"""
# Get the smallest, overall FOS
Goodman_FOS = CritPlaneFOS.min()
#print("The minimum fatigue factor of safety is:")
#print(Goodman_FOS)

file = open("Goodman_FOS.out", "w")
file.write(str(Goodman_FOS))
file.close()

file = open("Post_minFOS.out", "w")
file.write("Element ID = " + str(eleIDs[np.argmin(CritPlaneFOS)]) + "\n")
file.write("FOS = " + str(CritPlaneFOS[np.argmin(CritPlaneFOS)]) + "\n")
file.write("Sm [MPa] = " + str(CritPlaneMean[np.argmin(CritPlaneFOS)]/1.e+6) + "\n")
file.write("Samp [MPa] = " + str(CritPlaneAmp[np.argmin(CritPlaneFOS)]/1.e+6) + "\n")
file.write("Plane [deg] = " + str(CritPlaneDeg[np.argmin(CritPlaneFOS)]) + "\n")
file.close()
```