srlife: A Fast Tool for High Temperature Receiver Design and Analysis

Applied Materials Division
About Argonne National Laboratory
Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via DOE's SciTech Connect (http://www.osti.gov/scitech/)

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):
U.S. Department of Commerce
National Technical Information
Service 5301 Shawnee Rd
Alexandria, VA 22312
www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703) 605-6000
Fax: (703) 605-6900
Email: orders@ntis.gov

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728
Email: reports@osti.gov

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

Applied Materials Division
Argonne National Laboratory

May 2022

Prepared by

Mark C. Messner, Argonne National Laboratory
Bipul Barua, Argonne National Laboratory
Michael D. McMurtrey, Idaho National Laboratory
ABSTRACT

This report describes a tool for estimating the structural service life of tubular, panel solar receivers operating at high temperatures. A complete version of the tool is available as open source software at [https://github.com/Argonne-National-Laboratory/srlife](https://github.com/Argonne-National-Laboratory/srlife) and can be installed through the PyPi package manager. Given the basic receiver geometry and the thermal loads on the receiver, the tool provides 1D, 2D, or 3D thermal and structural (single tube and simplified system) analysis and creep-fatigue service life prediction for six metallic alloys – 316H, 800H, Alloy 617, Alloy 740H, and Alloy 230. With the exception of Alloy 230 where the material data is preliminary, the software included a detailed set of models for the materials, well-supported by high temperature experimental test data. The tool is designed for easy integration with a software stack, including solar field and thermohydraulic simulations, for optimizing receiver designs to meet service life and economic targets. The report describes several heuristics that can be applied in srlife to reduce the analysis time by several orders of magnitude but with fairly accurate life estimation when compared with full analysis. The report provides several examples demonstrating the utility of srlife in receiver design. Finally, the report discusses high temperature tests on Alloy 282, collected as part of this project, used to develop and support the material model for that alloy.
# TABLE OF CONTENTS

Abstract ............................................................................................................................................ i  
Table of Contents ........................................................................................................................... iii  
List of Figures .................................................................................................................................. v  
List of Tables ................................................................................................................................ vii  

1 Introduction................................................................................................................................. 1  

2 Analysis and Life Estimation Process........................................................................................3  
2.1 Package organization ............................................................................................................ 3  
2.2 User input and output ............................................................................................................ 3  
2.3 Thermal analysis .................................................................................................................... 5  
2.4 Structural analysis .................................................................................................................. 7  
2.5 Life estimation ....................................................................................................................... 12  

3 Design Heuristics ....................................................................................................................... 15  
3.1 Reference receivers .............................................................................................................. 15  
3.2 Reducing number of tubes in analysis ................................................................................ 17  
3.3 Reducing 3D analysis to 2D or 1D ....................................................................................... 19  
3.4 Additional heuristics ............................................................................................................ 23  

4 Material Data ............................................................................................................................. 25  
4.1 Metal thermal properties ....................................................................................................... 25  
4.2 Metal deformation models .................................................................................................... 27  
4.3 Metal damage properties ....................................................................................................... 35  

5 Applications ............................................................................................................................... 49  
5.1 Design and analysis studies ................................................................................................. 49  
5.2 Overview of ANU’s receiver design optimization ............................................................... 53  

6 Alloy 282 Test Results ............................................................................................................... 55  
6.1 Tensile ................................................................................................................................. 57  
6.2 Creep ................................................................................................................................... 60  
6.3 Cyclic testing ......................................................................................................................... 62  

7 Conclusions ................................................................................................................................ 69  

Acknowledgements ......................................................................................................................... 71  
Bibliography .................................................................................................................................. 73  
A srlife Documentation.................................................................................................................... 77
LIST OF FIGURES

Figure 2.1: Outline of the main modules in srlife and the process of going from user input to the receiver life estimation ................................................................. 3

Figure 2.2: Schematic of the required user input. Information can be provided through python data structures or an HDF5 file ................................................................. 4

Figure 2.3: Thermal analysis results comparison between (a) srlife and (b) MOOSE. Results show temperature distribution in the tube at an instance. The tube is subjected to same loads and boundary conditions in both srlife and MOOSE simulations ............................................... 6

Figure 2.4: Thermal analysis results comparison between (a) srlife and (b) MOOSE. Results show temperature vs time at the maximum temperature location in the tube. The tube is subjected to same loads and boundary conditions in both srlife and MOOSE simulations. ................................................................................................................................. 6

Figure 2.5: Schematic of receiver structural connections. Tubes are treated as nonlinear springs through the force-displacement relation derived from the top-surface applied displacement boundary condition. Connections are treated as linear-elastic springs with user defined stiffness. Very stiff interconnection springs produce strong mechanical coupling, low spring coefficients produce weak coupling ....................................................................................... 8

Figure 2.6: Structural analysis results comparison between (a) srlife and (b) MOOSE. Results show zz stress component distribution in the tube at an instance. The tube is subjected to same loads and boundary conditions in both srlife and MOOSE simulations .............................................................................. 9

Figure 2.7: Structural analysis results comparison between (a) srlife and (b) MOOSE. Results show stress components vs time at the maximum temperature location of the tube. The tube is subjected to same loads and boundary conditions in both srlife and MOOSE simulations. ................................................................................................................................................. 10

Figure 2.8: Comparison of analytical (dashed) with srlife structural solver (solid) radial displacement for the four (linear-elastic) test cases using 15 radial, 30 circumferential and 5 axial subdivisions .................................................................................................................................. 11

Figure 2.9: Example creep-fatigue damage envelop and illustration of simulated damage extrapolated to the onset of creep-fatigue damage to estimate life ............................................................................................................................... 13

Figure 3.1: Position of the heliostats and receiver created using SolarPILOT ................................................................................................................................. 16

Figure 3.2: (a) Solar flux on the receiver at noon. (b) Flux at noon on the tube crown along the tube length (shown for only one tube per panel as an example). (c) Flux variation during the day at one location in one tube in a panel. (d) Panel arrangement going from 0° to ±180° of the receiver circumference along the two flow paths. (e) Tube arrangement within a panel. ....................................................................................................................................... 17

Figure 3.3: Illustration of reduced number of tubes per panel and their location in a panel considered in simulation instead of simulating all tubes (100 in reference receivers) in a panel ....... 18

Figure 3.4: Receiver life vs number of tubes considered in 3D analysis of (a) low temperature receiver and (b) high temperature receiver .................................................................................................................. 19

Figure 3.5: Illustration of reduction of 3D analysis to 2D and 1D analyses ................................................................................................................................. 20

Figure 3.6: Simulation time comparison among 1D, 2D, and 3D analyses for simulating one cycle of the high temperature receiver using 12 CPUs (Intel® Xeon(R) Platinum 8270 @ 2.70GHz). Panels are disconnected but tubes are rigidly connected to the manifold. Material is Alloy 740H and material deformation model is elastic-creep ................................................................................................................................. 20

Figure 3.7: Tube crown temperature along the length of the tube for tube#0 and tube#99 in each panel for the high (left) and low (right) temperature receiver. Results are from 3D thermal analysis ............................................................................................................................................. 21

Figure 3.8: Analysis cost vs accuracy ................................................................................................................................. 22
Figure 3.9: Comparison between a transient analysis of a tube (2D cross-section) with the default behavior (a) and the temperature reset heuristic (b). The tube without the heuristic carries over a small thermal gradient to the next cycle. In actuality, the long soak at cold temperatures overnight would eliminate this gradient, as approximated by the heuristic........23
Figure 3.10: Plot comparing three methods of extrapolating creep-fatigue damage beyond the cycles explicitly considered in the analysis.................................................................24
Figure 4.1: Voce hardening model fit to the digitized tensile data for Alloy 282.................................30
Figure 4.2: Voce hardening model fit to the digitized tensile data for Alloy 230.................................31
Figure 4.3: Kocks-Mecking diagram used to construct the creep rate model for different materials........34
Figure 4.4: Larson Miller plots for (a) 316H, (b) Alloy 617, (c) Alloy 282, (d) Alloy 740H, and (e) Alloy 230.........................................................................................................................37
Figure 4.5: Larson-Miller plots and the corresponding stress vs time-to-rupture plots compared between the 2nd order polynomial and the 1st order polynomial considered in Larson-Miller correlation for Alloy 740H........................................................................................................38
Figure 4.6: Stress vs time-to-creep rupture at different temperatures for different alloys................39
Figure 4.7: Example fatigue curve fitting using ASME Section III, Division 5 design fatigue curve. The cutoff strain is used to limit the fatigue equations to interpolation within available data..........................................................40
Figure 4.8: Average fatigue curves for 316H, 800H, Alloy 617, and Alloy 740H....................................41
Figure 4.9: Average fatigue curves for Alloy 230 along with fatigue test data....................................41
Figure 4.10: Average fatigue curves for Alloy 282 along with fatigue test data from different sources. Test data include solution annealed (SA), aged, and cast specimens tested at Haynes, General Electric (GE), Siemens-Westinghouse (SW), and INL........................................43
Figure 4.11: INL’s test data on solution annealed (SA) specimen compared with the average fatigue curves and Haynes’ test data on solution annealed (SA)....................................................................................44
Figure 4.12: Creep-fatigue data plotted on the interaction diagram and two design envelopes: our original recommendation with an intersection at (0.05,0.05) and the final recommendation with an intersection at (0.1,0.1)............................................................47
Figure 5.1: Simple beam model for estimating manifold stiffness......................................................50
Figure 5.2: Manifold stiffness as a function of distance from the support..........................................51
Figure 6.1: Schematic of threaded specimens used for tensile and creep tests. Units are in inches. ......55
Figure 6.2: Schematic of button-head specimens used for fatigue and creep-fatigue testing. Units are in inches.................................................................56
Figure 6.3: Recreation of SA, PA, and OA Haynes 282 from [44]. Fatigue data shown on the left, creep on the right..........................................................57
Figure 6.4: Tensile tests (stress vs strain) for wrought Haynes 282 in the SA condition.........................58
Figure 6.5: Yields stress values for solution annealed Haynes 282 from this work and [44], as well as single-step aged-hardening treatment from [47]..........................................................59
Figure 6.6: Creep curves (strain vs. time) for wrought Haynes 282....................................................61
Figure 6.7: Larson Miller plot for creep test results in this work as well as those from [44]..............61
Figure 6.8: Modification to the grips of the fatigue/creep-fatigue specimens (Figure 6.2) to account for the thinner cast material.................................................................63
Figure 6.9: Stress vs. cycles for all Δε 0.6%, 750 °C tests.................................................................64
Figure 6.10: Stress vs. cycles for all Δε 1.0%, 750 °C tests.................................................................65
Figure 6.11: Hysteresis loops for all 750 °C creep-fatigue tests..........................................................65
Figure 6.12: Stress vs. cycles for all Δε 0.4%, 850 °C tests.................................................................66
Figure 6.13: Stress vs. cycles for all Δε 1.0%, 850 °C tests.................................................................66
Figure 6.14: Hysteresis loops for all 850 °C creep-fatigue tests..........................................................67
# LIST OF TABLES

Table 2.1: Maximum relative error between analytical and \textit{srlife} structural solver for the four (linear-elastic) test cases considered. .................................................................................................. 11

Table 3.1: Details of the reference receiver models. ................................................................................................................... 16

Table 3.2: Receiver life (days) estimation comparison among 3D, 2D, and 1D analyses. Panels are disconnected but tubes are rigidly connected to the manifold. ........................................................................ 21

Table 4.1: Overview of the material data source and the status of current material database of \textit{srlife} package. $^\#$ indicates data collected as part of this project. .................................................................................................. 25

Table 4.2: Sources for thermal conductivity and thermal diffusivity data. ................................................................................................. 26

Table 4.3: Thermal conductivity.......................................................................................................................................................... 26

Table 4.4: Thermal diffusivity.......................................................................................................................................................... 27

Table 4.5: Sources for instantaneous coefficient of thermal expansion (CTE), modulus of elasticity (E), and Poisson’s ratio (\(\nu\)), tensile test data, and minimum creep rate data. .......................... 28

Table 4.6: Instantaneous coefficient of thermal expansion. $^\$\text{ indicates extrapolated values.}$ .......................................................... 28

Table 4.7: Modulus of elasticity. $^\$\text{ indicates extrapolated values.}$ ........................................................................................................... 29

Table 4.8: Poisson’s ratio................................................................................................................................................................. 29

Table 4.9: Voce hardening model parameters for 316H. ...................................................................................................................... 31

Table 4.10: Voce hardening model parameters for 800H. ...................................................................................................................... 31

Table 4.11: Voce hardening model parameters for Alloy 617. ...................................................................................................................... 32

Table 4.12: Voce hardening model parameters for Alloy 740H. ...................................................................................................................... 32

Table 4.13: Voce hardening model parameters for Alloy 230. ...................................................................................................................... 32

Table 4.14: Voce hardening model parameters for Alloy 282. ...................................................................................................................... 32

Table 4.15: Kock-Mecking minimum creep rate model parameters. ........................................................................................................ 33

Table 4.16: Sources for creep rupture, fatigue, and creep-fatigue data. $^\#$ indicates a creep-fatigue interaction diagram is already available, meaning data did not have to be collected for this project. $^\$\text{ indicates data collected at INL as part of this project.}$ ........................................................................ 35

Table 4.17: Time-to-creep rupture equations. ...................................................................................................................................... 36

Table 4.18: Fatigue cycles-to-failure equations. .................................................................................................................................. 42

Table 4.19: Creep-fatigue damage envelope. .................................................................................................................................. 45

Table 4.20: INL-specific fatigue life data used to generate the creep-fatigue diagram. ............................................................................. 46

Table 5.1: Life of the high temperature reference receiver as a function of manifold and system stiffness. Results are from 2D analysis at the maximum temperature location. The material is Alloy 740H and the deformation model is elastic-creep. ...................................................................................................................... 49

Table 5.2: Life of the low temperature reference receiver as a function of manifold and system stiffness. Results are from 2D analysis at the maximum temperature location. The material is Alloy 230 and the deformation model is elastic-creep. ...................................................................................................................... 50

Table 5.3: Manifold design check for the reference receivers. ...................................................................................................................... 51

Table 5.4: Receiver life (days) estimation as a function of material model. Panels are disconnected but tubes are rigidly connected to the manifold. Material model is elastic-creep. Results are from 2D analysis of tubes sliced at maximum temperature location. Panels are disconnected but tubes are rigidly connected to the manifold. ...................................................................................................................................................... 51

Table 5.5: Receiver life (days) estimation for different alloys. Results are based on 2D analysis at the maximum temperature location and using the base (i.e. full inelastic) deformation model for all the alloys. Panels are disconnected but tubes are rigidly connected to the manifold. ...................................................................................................................................................... 52

Table 5.6: Comparison of individual panel life (days) calculated from 2D analysis using the base model. Panels are disconnected but tubes are rigidly connected to the manifold. ...................................................................................................................................................... 52

Table 6.1: Composition of Haynes 282 Heat 2082 8 8433. .................................................................................................................................. 55
Table 6.2: Tensile properties of SA Haynes 282. ....................................................................................... 58
Table 6.3: Average tensile properties of Haynes 282 after the single-step age-hardening treatment, as provided by Haynes International [45]................................................................................................................... 58
Table 6.4: Tensile properties for solution annealed [44] and single-step age-hardening treatment [47] Haynes 282 ................................................................................................................................................. 59
Table 6.5: Summary of creep rupture times. ........................................................................................................ 61
Table 6.6: Cyclic test conditions for wrought 282. Due to time restriction, the 600 minute hold creep-fatigue tests were not performed. ........................................................................................................ 62
Table 6.7: Final fatigue and creep-fatigue results from this Haynes 282 test program. With the exception of the single test labeled Cast-Aged, all testing was performed on wrought plate. Yellow rows indicate tests are on-going at the time of writing this report. .................. 63
1 Introduction

Energy conversion efficiency goals continue to push target temperatures for next-generation concentrating solar power (CSP) systems higher [1]. The design of these high temperature CSP systems will be limited by the service life of the receiver (or, for particle-based designs, the heat exchanger between the particle stream and working fluid). These components will experience the hottest metal temperatures combined with high thermal stresses and, for some designs, high pressures. To recover the capital cost the receiver must operate under diurnal cycling at peak metal temperature of around 800°C for more than 10,000 daily cycles (i.e. 30 years). Designing such a receiver with current high temperature metal alloys will be challenging and will likely require an integrated approach, optimizing the receiver structural and thermohydraulic design simultaneously with the solar field and aiming strategy.

Under these elevated temperature conditions and given the diurnal cycling experienced by the receiver, the life of the receiver will be constrained by creep-fatigue damage accumulation. This failure mechanism will require new design methods and types of material data compared to lower-temperature systems [2]. System optimization requires methods that can quickly predict the expected life of a particular receiver configuration with reasonable accuracy. Current receiver design tools focus on calculating the incident flux from the solar flux [3] and modeling the receiver thermohydraulics [4]. Although some consideration has been given towards the structural design of the receiver [5], current approaches use greatly simplified analysis and design methods.

A complete receiver structural design typically involves full-scale finite element analysis followed by extensive post-processing to evaluate the analysis results against design rules guarding against creep-fatigue, creep, deformation limits, plastic collapse [6]. This process is time-consuming, both computationally and in terms of analyst time, and therefore cannot be easily embedded in a framework for optimizing the design of a receiver. This report describes a fast computational tool for estimating the life of a tubular metal receiver. The main goal of this work is to develop a computational tool that can be embedded in a receiver optimization software stack, including tools for tuning the solar field and receiver thermohydraulics. This complete software stack could then be used to optimize receiver designs to meet key economic targets.

The main goals for the receiver tool are:

1. Provide a best-estimate of the receiver service life given the basic receiver geometry and the thermal loads on the receiver coming from external solar field and thermohydraulics simulations. The tool estimates life based on creep-fatigue damage accumulation in the receiver using nominal, i.e. best estimate and not lower-bound, properties.

2. Provide the estimated life quickly so that the tool can be used to optimize the receiver design automatically, for example using computational optimization methods.

3. Have a simple, clean API so the tool can be integrated into existing receiver optimization software stacks.

The tool does not:

1. Perform thermohydraulics or solar field calculations. The tool relies on external software to provide the thermal boundary conditions imposed on the metallic receiver.
2. Handle geometries beyond tubular, panel receivers. However, some of the modules which make up the tool might be useful for other types of designs.

3. Provide the required material data for a wide range of materials. Adequate creep-fatigue test data is only available for a limited number of materials and so the tool provides properties for a limited number of high temperature metal alloys. The user can add additional material data as it becomes available.

A complete version of the tool, named \textit{srlife}, is available both on github (https://github.com/Argonne-National-Laboratory/srlife) and through the PyPi (https://pypi.org) package manager, making the package installable through the standard python package manager “pip”. Chapter 2 describes the analysis processes \textit{srlife} applies to estimate the life of a receiver along with a description of the required input data from the user.

A complete receiver may contain hundreds of tubes. Performing full 3D thermal and structural analysis of a complete receiver is too expensive. To reduce the computational cost, Chapter 3 assesses several heuristics on two reference receiver models and compares their performances in terms of reduced computation cost and accuracy relative to the full analysis. The chapter provides a guideline for implementing the design heuristics in \textit{srlife}.

\textit{srlife} requires metal thermal, deformation, and damage properties for receiver life estimation. The tool currently contains material data for six high temperature alloys – 316H, 800H, Alloy 617, Alloy 740H, Alloy 282, and Alloy 230. Chapter 4 discusses the construction these material data from different sources.

Chapter 5 then provides some examples of different design and analysis studies on the reference receiver models using \textit{srlife}. The chapter also includes a summary of Australian National University’s (ANU’s) work on optimizing the design of a Gen3 liquids-pathway sodium-cooled receiver using \textit{srlife}.

Chapter 6 summarizes the Alloy 282 fatigue and creep-fatigue testing effort at Idaho National Laboratory as part of this project.

Finally, Chapter 7 summarizes the work presented in this report. An appendix includes the complete documentation for the \textit{srlife} package, detailing the API and providing several example receiver analyses.
2 Analysis and Life Estimation Process

`srlife` is a python module, released under an open-source MIT license. One objective in developing the package is to have it depend on only commonly-available python libraries found in the PyPi package database, to facilitate easy installation. The package itself is written in pure Python to ease cross-platform compatibility. However one dependency, NEML (https://github.com/Argonne-National-Laboratory/neml), requires compilation. However, NEML provides a PyPi package with automatic compilation scripts so that the user can easily install `srlife` from the PyPi database.

2.1 Package organization

Figure 2.1 outlines the main modules in `srlife` along with the steps in the life-estimation analysis. In addition, the package includes modules providing basic utilities such as conversion between tensor and Mandel vector notation, loading material data from files, etc. These basic utility modules are not described here. The subsequent sections in this chapter provides a high-level overview of the life estimation process in `srlife`, discussing all the main modules except the material database. The detailed description of the material database is in Chapter 4.

![Diagram showing the main modules in srlife and the process of going from user input to the receiver life estimation.](image)

2.2 User input and output

The user provides the basic geometry of the tubular receiver as well as thermal and mechanical boundary conditions. Figure 2.2 outlines the hierarchical data structure used to define a complete receiver design. The API requires the user to define the receiver, load the required material properties, and execute the analysis with a short Python script. This approach is sufficient to embed `srlife` in a python-driven software stack. However, the receiver data structure in Figure 2.2 can also be provided as an HDF5 file.
The data structure defines a receiver as a collection of panels, each of which are a collection of individual tubes. Each tube needs thermal boundary conditions – typically we expect these to be a net, incident flux calculated from a solar field/aiming simulation on the tube outside diameter and convective heat transfer between the tube and a fluid on the inner diameter, with the fluid temperatures and flow rates calculated from an external thermohydraulic simulation. However, the full range of flux (Neumann), fixed temperature (Dirichlet), and convective boundary conditions are available.

Each tube also requires mechanical boundary conditions. The user only needs to provide the inner pressure in the tube as a function of time, again from external thermohydraulic simulations. *srlife* automatically provides essential boundary conditions on the tube displacements sufficient to constrain the model against rigid rotation and translation. The user must also specify the initial tube temperature.

In addition to the thermomechanical boundary conditions, the user provides metadata to help the package analyse the system. Specifically, the user must provide the period of an individual daily cycle and the number of daily cycles explicitly represented in the thermomechanical boundary conditions. For example, the user might provide only four cycles of boundary conditions (e.g. conditions for the equinoxes and the solstices) and use these four days to represent the remainder of the cycles in a full year of service.
Finally, Figure 2.2 notes that the user provides two spring stiffnesses – one describing the mechanical connection between the panels in the receiver and one between the tubes in each panel (manifold stiffness). Section 2.4.2 below on the system structural analysis describes these spring stiffnesses in greater detail.

At the end of the analysis, srlife outputs the best estimate life in daily cycles. In addition, the user can output the full thermal, structural, damage results to VTK files which can be visualized with program like ParaView. These results can also be saved in an HDF5 file.

By default, srlife saves the full thermal and structural results from each time step of the analysis for all tubes in the receiver memory. On machines with a large amount of RAM this is not an issue, but smaller machines can run out of memory, particularly considering the large amount of memory required during the linear solve phase of the structural model. To alleviate this problem srlife also includes the option to page the tube results to disk, freeing up the memory for other tasks.

### 2.3 Thermal analysis

The model assumes the tube thermal analyses are decoupled, i.e. each tube is thermally isolated from all the other tubes in the receiver, except through the external thermohydraulic simulation. The srlife thermal analysis module uses a finite difference, cylindrical coordinate, transient heat transfer solver to determine the metal temperature history corresponding to the user-supplied thermal boundary conditions. The solver provides 3D, 2D, and 1D analysis options. This solver is implemented in srlife directly, only using the numpy and scipy libraries to setup the sparse finite-difference equations and solve the resulting system of nonlinear equations. This module provides the metal temperatures as a function of time at each grid point in the cylindrical fixed grid.

#### 2.3.1 Verification

We performed the verification of the thermal analysis module in srlife by comparing the results from srlife with those from MOOSE (https://mooseframework.inl.gov/), an open source finite element code, developed by Idaho National Laboratory. We considered a single tube subjected to an arbitrary daily heat flux load on the outer surface of the tube. Heat flux on the tube is non uniform along both the length and circumference. Figure 2.3 compares the temperature distribution in the tube at an instance between srlife and MOOSE. Figure 2.4 compares the temperature at the maximum temperature location in the tube as a function of time. These figures indicate the thermal analysis results from srlife are in good agreement with results from MOOSE, verifying the thermal analysis modules of srlife.

In addition, the srlife test-case suite contains several 1D, 2D, and 3D verifications of the thermal solver.
Figure 2.3: Thermal analysis results comparison between (a) *srlife* and (b) MOOSE. Results show temperature distribution in the tube at an instance. The tube is subjected to same loads and boundary conditions in both *srlife* and MOOSE simulations.

Figure 2.4: Thermal analysis results comparison between (a) *srlife* and (b) MOOSE. Results show temperature vs time at the maximum temperature location in the tube. The tube is subjected to same loads and boundary conditions in both *srlife* and MOOSE simulations.
2.4 Structural analysis

2.4.1 Single-tube analysis

The base structural analysis problem is the analysis of single tube considering the thermal strain induced by the metal temperature field, determined by the thermal analysis, and the tube internal pressure, provided by the user. The single tube analysis module imposes no-z displacement boundary conditions on the bottom surface, an imposed displacement in the z-direction on the top surfaces, and sufficient point constraints in the x- and y-directions to prevent rigid rotations and translations. As described in the next section, the top-surface response can be used to impose a wide variety of single-tube responses ranging from completely decoupled, generalized plane strain to having each tube in a panel rigidly connected and all the panels likewise rigidly coupled together.

The single tube analysis module solves the mechanical problem using a quasi-static, nonlinear finite element analysis. Again the module provides 1D, 2D, and 3D options. The scikit-fem library [7] was used to assemble the finite element equations. Unlike the thermal analysis module, the 3D and 2D analyses are in Cartesian coordinates, though the 1D analysis remains in cylindrical coordinates. The module takes the material response, including the coefficient of thermal expansion, from the NEML module as requested by the user from the material property library.

The analysis module takes as input the metal temperatures, the inner tube pressure, and the top-surface displacement and provides complete mechanical results (strains and stresses) at each quadrature point in the finite element calculation, as well as the force on the top surface, required for the system solver.

2.4.2 System analysis

In an actual receiver, individual tubes are mechanically coupled through the stiffness provided by the tube manifold. The panels themselves may be mechanically coupled, for example by the supports connecting each panel to the tower. The amount of coupling provided by these connections can only be determined through detailed finite element analysis, beyond the scope of the package. srlife provides a simplified, parametric approach for modelling this constraint.

Figure 2.5 shows how the package models tube and panel connections as 1D springs. The tubes themselves are treated as nonlinear springs through the force-displacement relation derived from the top-surface applied displacement boundary condition. Connections are treated as linear-elastic springs with a user-defined spring coefficient. Very stiff interconnection springs produce strong mechanical coupling, low spring coefficients produce weak coupling.

The module handles two special cases: a “disconnected” spring with zero stiffness and a “rigid” spring with infinite stiffness. In the former case the spring network splits into two parts at the connections. Each subnetwork can be solved in parallel. In the latter case the module condenses the degrees of freedom on either end of the spring.

This module, in the end, produces a collection of spring networks describing the system. These spring equations are solved, for each time step, using Newton’s method. The tube axial stiffness requires invoking a full finite element simulation of the tube, accounting for the internal pressure and thermal stresses. These finite element updates can proceed in parallel, i.e. all the tubes connected in a spring network can be updated at the same time.
This spring-network representation of the structural system has several advantages:

1. It provides at least a simplified representation of mechanical coupling, which may be important for high temperature systems where colder, less compliant tubes can absorb stress redistributed from hotter, more compliant tubes.

2. It is parametric: in the absence of detailed information on the tube and panel interconnections the user can sample a range of spring stiffnesses, from zero to infinity, to gain an understanding of the effect of mechanical coupling on the predicted life of the receiver.

3. It decouples individual tube analyses. The largest finite element analysis the module runs in that for a single tube. While the tubes are mechanically connected, they are connected indirectly through the spring network. This means the package can solve each individual tube in parallel.

4. Disconnecting all tubes using “disconnect” springs provides a generalized plane strain response for each tube – not free expansion on the top surface. This is a common choice for 2D analysis of receiver tubes.

---

Figure 2.5: Schematic of receiver structural connections. Tubes are treated as nonlinear springs through the force-displacement relation derived from the top-surface applied displacement boundary condition. Connections are treated as linear-elastic springs with user defined stiffness. Very stiff interconnection springs produce strong mechanical coupling, low spring coefficients produce weak coupling.
2.4.3 Verification

We used MOOSE to verify the structural analysis results from srlife using the same tube problem used for verifying thermal analysis module. For structural analysis the tube is subjected to the temperature distribution determined in thermal analysis along with a pressure load acting on the inner surface. We applied the same mechanical boundary condition in both srlife and MOOSE simulations. Figure 2.6 shows the comparison for the distribution of a stress component in the tube at an instance. Figure 2.7 compares all the stress components at the maximum temperature location over the applied loading cycle. These comparisons indicate the structural analysis results from srlife are in good agreement with results from MOOSE.

Figure 2.6: Structural analysis results comparison between (a) srlife and (b) MOOSE. Results show zz stress component distribution in the tube at an instance. The tube is subjected to same loads and boundary conditions in both srlife and MOOSE simulations.
Figure 2.7: Structural analysis results comparison between (a) *srlife* and (b) MOOSE. Results show stress components vs time at the maximum temperature location of the tube. The tube is subjected to same loads and boundary conditions in both *srlife* and MOOSE simulations.

In addition, the *srlife* test-case suite includes several 1D, 2D, and 3D validations of the structural analysis solver. Some of these are analytical validations which includes:

1. Internal pressure under plane strain condition,
2. Internal pressure assuming a cylinder closed at both ends (axial force),
3. Thermal through-wall gradient under plane strain condition, and
4. Thermal through-wall gradient under generalized plane strain condition (zero axial force).

Figure 2.8 shows a comparison of analytical and structural solver radial displacement for the four test cases in 3D. A summary of relative error between analytical and *srlife* radial displacement is given in Table 2.1. In the case of 1D structural analysis the maximum relative error between analytical and solver result is very good for all four cases. As cylinder dimension is increased a general increase in relative error is observed, with pressure boundary conditions leading to more drift in the results.
Figure 2.8: Comparison of analytical (dashed) with srlife structural solver (solid) radial displacement for the four (linear-elastic) test cases using 15 radial, 30 circumferential and 5 axial sub-divisions

<table>
<thead>
<tr>
<th>Model Type</th>
<th>1D</th>
<th>2D</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure, plane strain</td>
<td>6.4e-5</td>
<td>6.7e-3</td>
<td>6.7e-3</td>
</tr>
<tr>
<td>Pressure, out-of-plane</td>
<td>9.3e-5</td>
<td>9.4e-3</td>
<td>4.3e-2</td>
</tr>
<tr>
<td>Axisym. thermal, plane strain</td>
<td>4.9e-4</td>
<td>1.4e-3</td>
<td>1.4e-3</td>
</tr>
<tr>
<td>Axisym. thermal, gen. plane strain</td>
<td>8.7e-4</td>
<td>1.8e-3</td>
<td>1.8e-3</td>
</tr>
</tbody>
</table>

Table 2.1: Maximum relative error between analytical and srlife structural solver for the four (linear-elastic) test cases considered.
2.5 Life estimation

The results of the thermal, single tube, and system structural analysis are fields at each quadrature point, for each tube, giving the time-history of temperature, stress, and strain for the full loading history provided by the user. The damage module uses these time histories to estimate the number of cycles the receiver can undergo before creep-fatigue damage initiates at the worst point of any tube in the receiver.

The damage model applies an “ASME” type approach, based on the creep-fatigue rules in Section III, Division 5 of the ASME Boiler and Pressure Vessel Code [8]. This approach begins by calculating time-fraction creep damage using the formula:

\[ D_c = \frac{\int dt}{t_R(\sigma_e, T)} \]  

(2.1)

where \( \sigma_e \) is the von Mises effective stress, calculated from the individual stress components, \( T \) is the temperature, and \( t_R \) is the rupture time correlation described above. The time integral covers the entire time history provided by the user (which may represent more than one daily cycle). Similarly, the approach requires the fatigue damage calculated as

\[ D_f = \sum \frac{1}{N_f(\Delta \varepsilon, T)} \]  

(2.2)

where \( \Delta \varepsilon \) is an equivalent mechanical strain range, defined in the ASME Code, and the sum proceeds over each daily cycle provided by the user (defined by the cycle period metadata).

The ASME creep-fatigue damage approach uses a creep-fatigue envelope to predict the onset of creep-fatigue damage. Figure 2.9 shows a sample envelope, defined in terms of the creep and fatigue damage fractions. The approach predicts the onset of creep-fatigue damage when the point \( (D_f, D_c) \) falls outside the damage envelope.

The figure also illustrates the extrapolation of the creep-fatigue damage accumulated during the user-defined load cycle history to determine the number of allowable repetitions. The damage module provides several options for extrapolation which are discussed in Chapter 3. Essentially the approach assumes the receiver will continue to accumulate damage based on the option user choses and then finds the largest value of \( N \) for which the point lies just on the damage envelope. The damage calculator repeats this procedure at each point in the receiver to find the worst case, lowest value of \( N \). Because the user provided loading history may span several daily cycles, this value of \( N \) must be multiplied by the number of days in the load history to calculate the final estimated receiver life.
Figure 2.9: Example creep-fatigue damage envelop and illustration of simulated damage extrapolated to the onset of creep-fatigue damage to estimate life.
3 Design Heuristics

A complete receiver may contain hundreds of tubes. Performing full 3D analysis of the receiver using srlife would take several days to complete (without taking the advantage of multi-thread parallel processing). This is far too expensive to embed in an optimization loop and therefore heuristics must be developed to reduce the cost of the analysis. However, reducing the computational cost of the analysis may reduce the accuracy of the estimate of the receiver service life. This chapter, therefore, assesses several heuristics by comparing their performance in terms of reduced computation cost and accuracy relative to the full analysis and provides a guideline for implementing the design heuristics in srlife. These heuristics are:

1. Reducing the number of tubes explicitly represented in the analysis.
2. Reducing the full 3D analysis to either 2D or 1D for either the thermal, structural, or both analyses.
3. Specifying the plane or line to use in reduced 2D or 1D analysis, respectively, at different locations of the tube.

The chapter also discusses two additional heuristics:

1. Daily resetting the tube temperature to their initial values
2. Extrapolation of creep-fatigue damage

3.1 Reference receivers

The first set of heuristics are assessed using two sample receiver designs:

1. A high temperature receiver with chloride salt as the working fluid and
2. A low temperature receiver with nitrite salt as the working fluid.

Table 3.1 provides the details of the receiver models. We used SolarPILOT [3] to determine the heat flux distribution on the receiver through an optimization analysis. Figure 3.1 shows the heliostats configuration for both the receivers. Figure 3.2 provides the heat flux map on the receiver at noon along with the flux variation over time during of the day. Figure 3.2 also provides schematics of the panel arrangements within each of the two flow paths and the tube arrangement within a panel.

The radiation solar heat flux is maximum in the north hemisphere and minimum in the south hemisphere. As the heat flux is symmetric about north-south axis, two flow paths each containing 6 panels are considered. Salt enters the receiver at the north side through panel#0 and leaves the receiver at the south side from panel#5. Each panel consists of 100 vertical tubes with 42.2 mm diameter and 1 mm thickness. The total salt mass flow rates are 2941 kg/s and 1087 kg/s, respectively, for the high and low temperature receivers. The respective design salt pressures are 3 MPa and 0.8 MPa.
<table>
<thead>
<tr>
<th></th>
<th>High temperature receiver</th>
<th>Low temperature receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working fluid</td>
<td>Chloride salt</td>
<td>Nitrite salt</td>
</tr>
<tr>
<td>Design power</td>
<td>500 MWt</td>
<td>500 MWt</td>
</tr>
<tr>
<td>Design DNI</td>
<td>1000 W/m²</td>
<td>1000 W/m²</td>
</tr>
<tr>
<td>Type</td>
<td>360° external tubular</td>
<td>360° external tubular</td>
</tr>
<tr>
<td>Tower height</td>
<td>195 m</td>
<td>195 m</td>
</tr>
<tr>
<td>Receiver height (tube length)</td>
<td>21 m</td>
<td>21 m</td>
</tr>
<tr>
<td>Receiver diameter</td>
<td>17 m</td>
<td>17 m</td>
</tr>
<tr>
<td>Flow paths</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Panels per flow path</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Tubes per panel</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Tube OD</td>
<td>42.2 mm</td>
<td>42.2 mm</td>
</tr>
<tr>
<td>Tube thickness</td>
<td>1 mm</td>
<td>1 mm</td>
</tr>
<tr>
<td>Salt inlet temperature</td>
<td>550°C</td>
<td>290°C</td>
</tr>
<tr>
<td>Salt outlet temperature</td>
<td>720°C</td>
<td>565°C</td>
</tr>
<tr>
<td>Total salt mass flow rate</td>
<td>2941 kg/s</td>
<td>1087 kg/s</td>
</tr>
<tr>
<td>Design salt pressure</td>
<td>3 MPa</td>
<td>0.8 MPa</td>
</tr>
</tbody>
</table>

Table 3.1: Details of the reference receiver models.

Figure 3.1: Position of the heliostats and receiver created using SolarPILOT.
3.2 Reducing number of tubes in analysis

A full analysis of the reference receivers would require analyzing 600 tubes, which is very expensive. A first step to reduce the computational cost would be to reduce the number of tubes in the analysis. A minimum number of tubes per panel should be selected for analysis so that the life estimated from the reduced analysis represents the life of the receiver calculated from full analysis. To investigate this we performed several 3D analyses of the reference receivers by selecting different number of tubes per panel. Figure 3.3 illustrates the different cases of receiver analyses considered with reduced number of tubes per panel. The single tube per panel case considers two simulations – one with the tube at the middle of the panel and one with the hottest tube in the panel. The two tubes per panel simulation considers the hottest and coldest tubes of the panel. The three and five tubes per panel cases also include the hottest and coldest tubes.

Figure 3.4 compares the calculated receiver lives from 3D thermal and structural analyses as a function of tubes per panel considered in the analysis. These simulations consider Alloy 740H for the high temperature receiver and Alloy 230 for the low temperature receiver and an elastic-creep deformation model for both materials. The analyses also considered the panels structurally disconnected but the tubes within a panel rigidly connected, i.e. infinite manifold stiffness.
Figure 3.4 indicates the estimated lives from single tube per panel analyses are much longer for both the receivers when compared with results from analyses with considering higher number of tubes per panel. Each tube within a panel experiences different incident flux and therefore have different metal temperature. This results in different axial thermal strains for the tubes. Since the tubes are structurally connected to each other through the manifold, they cannot expand freely in the axial direction. The axial deformation of a tube depends not only on its temperature but also on the temperature of other tubes in the panel. Simulating only one tube from a panel does not consider this axial constraint imposed by other tubes and let the tube to deform freely in the axial direction and hence results in a longer estimated life.

The two tube per panel case considers the hottest and coldest tubes within a panel. This applies the extreme axial constraint on the tubes and therefore the estimated life is the minimum compared to other cases. Increasing the number of tubes in analysis slightly increases the estimated life of the receiver. However, the increase is not significant when compared with the analysis cost. We therefore recommend analyzing at least two tubes per panel which should include both the hottest and coldest tubes of the panel.

Figure 3.3: Illustration of reduced number of tubes per panel and their location in a panel considered in simulation instead of simulating all tubes (100 in reference receivers) in a panel.
Figure 3.4: Receiver life vs number of tubes considered in 3D analysis of (a) low temperature receiver and (b) high temperature receiver.

### 3.3 Reducing 3D analysis to 2D or 1D

The above discussion significantly reduces the total number of simulated tubes for a receiver by considering only the hottest and coldest tubes in a panel. For instance, life estimation of the reference receivers requires simulating only 12 tubes instead of 600 tubes. However, 3D analysis of 12 tubes will still be expensive, especially for embedding `srlife` in a receiver design optimization loop. Moreover, often structural analysis are run for repeated number of cycles if an inelastic model is used to capture the time dependent deformation of the material for more accurate estimation of life compared to the elastic model. This requires development of additional strategies to reduce the analysis time.

A significant reduction of the analysis time can be achieved by performing 1D or 2D analysis of the tubes instead of a 3D analysis. Figure 3.5 illustrates how a 3D model of a tube can be reduced into 2D by taking a normal slice along the tube axis and further reduced into 1D by taking a though thickness slice at the tube crown temperature location. Figure 3.6 provides a bar chart comparing the computational time among 1D, 2D, and 3D analyses. The bar chart shows the total analysis time as well as the time taken by each of the thermal, structural, and damage calculation modules of `srlife`. The bar chart indicates at least two order of magnitude reduction in analysis time for both the thermal and structural simulation calculations when the 3D model is reduced to 2D. This is a drastic reduction in analysis time. Further reducing the model into 1D also reduces the analysis time but not very significantly.

However, this large reduction in computational cost does come with a cost. Reducing a 3D model into 2D or 1D will always reduce the accuracy in simulation results. The question is how much accuracy is lost and whether it is acceptable considering the gain in computational cost. The following section investigates this by comparing the 2D and 1D analysis results at different tube locations with the 3D analysis results.
Figure 3.5: Illustration of reduction of 3D analysis to 2D and 1D analyses.

Figure 3.6: Simulation time comparison among 1D, 2D, and 3D analyses for simulating one cycle of the high temperature receiver using 12 CPUs (Intel® Xeon(R) Platinum 8270 @ 2.70GHz). Panels are disconnected but tubes are rigidly connected to the manifold. Material is Alloy 740H and material deformation model is elastic-creep.

3.3.1 Analysis results comparison

Figure 3.7 plots the tube crown temperatures along the tube length at noon for both the high and low temperature receivers. The temperatures are calculated from 3D thermal analysis of the receiver by considering the hottest and coldest tubes from each panel. The maximum tube temperature for the high temperature receiver is about 760°C, while it is about 650°C for the low temperature receiver. The tube crown temperature profiles along the tube length indicate that the maximum temperature location is not always at the maximum flux location. Since the working fluid follows a serpentine flow path, the location (along the tube length) of the maximum tube crown temperature also depends on the fluid temperature.
Figure 3.7: Tube crown temperature along the length of the tube for tube#0 and tube#99 in each panel for the high (left) and low (right) temperature receiver. Results are from 3D thermal analysis.

Table 3.2 compares the life of the receivers estimated from 3D, 2D, and 1D analyses. For 2D and 1D analyses, the structural analysis is based on a “slice” at two different locations along the tube length – a) the maximum flux location, and b) the maximum temperature location. As discussed above, the maximum temperature location is not always the maximum flux location and therefore the estimated lives from 2D and 1D analyses at two locations are different. Table 3.2 indicates that the 1D analysis results are highly inaccurate and predict much longer lives for the receiver compared to the 3D analysis. This is expected as the bending stress due to the circumferential temperature difference is not considered in the 1D analysis. On the other hand, the 2D analysis is reasonably accurate, particularly for the maximum temperature slice location.

<table>
<thead>
<tr>
<th>Receiver</th>
<th>Material and model</th>
<th>3D analysis</th>
<th>2D analysis (location)</th>
<th>1D analysis (location)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High temperature</td>
<td>Alloy 740H, elastic-creep</td>
<td>3063</td>
<td>3165 (Max. flux)</td>
<td>558326 (Max. flux)</td>
</tr>
<tr>
<td>Chloride salt</td>
<td></td>
<td></td>
<td>3165 (Max. temp)</td>
<td>558326 (Max. temp)</td>
</tr>
<tr>
<td>Low temperature</td>
<td>Alloy 230, elastic-creep</td>
<td>24737</td>
<td>44666 (Max. flux)</td>
<td>706938 (Max. flux)</td>
</tr>
<tr>
<td>Nitrite salt</td>
<td></td>
<td></td>
<td>24136 (Max. temp)</td>
<td>459047 (Max. temp)</td>
</tr>
</tbody>
</table>

Table 3.2: Receiver life (days) estimation comparison among 3D, 2D, and 1D analyses. Panels are disconnected but tubes are rigidly connected to the manifold.

3.3.2 Analysis cost vs accuracy

Figure 3.8 compares the 1D (location: maximum temperature), 2D (location: maximum temperature), and 3D analyses in terms of cost and accuracy. The clearly show the 2D analysis at the maximum temperature location is fairly accurate while taking a fraction of time the 3D analysis takes.
3.3.3 Procedure for speeding up analysis

Based on the discussion above the following procedure can be applied in the “heuristics” system of *srlife* to speed up the life estimation analysis:

1. Create a reduced 3D model of the receiver by considering at least two tubes from each panel. The tubes should include the hottest and coldest tubes of the panel. The hottest tube is the tube with the maximum tube crown temperature averaged over time and the coldest tube is the tube with the minimum tube crown temperature averaged over time.

2. Perform thermal analysis of the reduced 3D model. If there are more than one cycle type available, run only one repetition of each cycle type.

3. From 3D thermal analysis determine the maximum temperature location along tube length.

4. Reduce 3D model of individual tube to 2D model by slicing at the maximum metal temperature location.

5. Perform thermal and structural analysis of the reduced 2D model of the receiver for complete loadings, i.e. applying all cycles for repeated loadings.

6. Determine life of the receiver based on results from the 2D analysis.
3.4 Additional heuristics

3.4.1 Daily resetting tube temperature

In response to a suggestion from ANU, this heuristic “resets” the tube temperatures to their initial values after each daily thermal cycle. Figure 3.9 illustrates the reason for this heuristic. Typically design thermal cycles will not include the night periods where the receiver temperatures slowly become constant as conduction relieves any remaining thermal gradients. Design analyses applying transient heat transfer theory and not explicitly representing these nightly isothermal holds can restart the next daily cycle from a non-isothermal condition, with (small) thermal gradients still present. These small gradients can introduce a small amount of fictitious ratcheting strain.

`srlife` includes two options to mitigate this problem:

1. An option to use steady-state heat transfer theory
2. An option to reset the tube temperatures to an isothermal state before starting the next daily cycle.

In addition, designers could always explicitly include the nightly soak in the design thermal cycles.

![Figure 3.9: Comparison between a transient analysis of a tube (2D cross-section) with the default behavior (a) and the temperature reset heuristic (b). The tube without the heuristic carries over a small thermal gradient to the next cycle. In actuality, the long soak at cold temperatures overnight would eliminate this gradient, as approximated by the heuristic.](image)

3.4.2 Extrapolating creep-fatigue damage

To determine the best-estimate allowable cycles, `srlife` extrapolate the accumulated creep-fatigue damage from the cycles represented in the analysis out to the point lies just on the damage envelope. A typical extrapolation approach assumes the remaining life of the component repeats the represented cycles over and over, until predicted failure. This extrapolation is reasonable for an elastic analysis or an analysis where the component undergoes only a small amount of stress relaxation. However, it is less appropriate for cases where there is significant stress distribution via creep in the receiver. This stress redistribution changes the local accumulation of damage over time, meaning the damage accumulated in early cycles is significantly different than the damage accumulated in later cycles. Therefore, `srlife` includes two additional damage extrapolation approaches:
1. "last": extrapolate based on the last repetition of the represented duty cycles.

2. "polynomial": extrapolate using a polynomial fit of a plot of the number of cycles versus the damage accumulated per cycle. The user can select the polynomial order of the extrapolation.

Figure 3.10 illustrates all three approaches. To maintain backwards compatibility the tool uses the typical approach (lump) by default, but the user can select the other approaches and, for the polynomial extrapolation method, control the degree of the polynomial.

![Figure 3.10: Plot comparing three methods of extrapolating creep-fatigue damage beyond the cycles explicitly considered in the analysis.](image-url)
4 Material Data

Figure 2.1 illustrates the different material property groups used by thermal, structural, and damage modules in srlife. The package categorizes the material data into four basic material property groups – fluid thermal properties, metal thermal properties, metal deformation models, and metal damage properties. All the material data are provided in the package in XML format which can be easily modified. Users can also add new material data into the package by creating new XML files and following the format of the existing XML files in the package.

The tool uses the convective heat transfer coefficient (fluid thermal property) to calculate the heat transfer between the metal and the fluid flowing inside the receiver tubes. srlife currently does not calculate this property and relies on the users to enter the values in the material data base for their specific receiver design.

The package contains complete metal thermal, deformation, and damage properties for 316H, 800H, Alloy 617, Alloy 740H, Alloy 282, and Alloy 230. The subsequent section discusses in detail the construction of the material data for these material. Table 4.1 provides an overview of the material data source and status for all six materials. The material data for first three alloys are well supported by the large DOE:NE databases. For Alloy 740, the material data are well supported by past DOE-funded work, collected in a material database, as well as by the tests conducted at INL as part of a previous SETO funded project. Some of Alloy 282 data come from literature and some, specifically creep-fatigue test data were collected at INL as part of this project. Creep-fatigue test data are not available for Alloy 230 to construct a creep-fatigue damage envelop. srlife currently uses the Alloy 617 creep-fatigue damage envelop for Alloy 230.

<table>
<thead>
<tr>
<th>Materials</th>
<th>Source</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>316H</td>
<td>DOE:NE</td>
<td>Well supported</td>
</tr>
<tr>
<td>800H</td>
<td>DOE:NE</td>
<td>Well supported</td>
</tr>
<tr>
<td>Alloy 617</td>
<td>DOE:NE</td>
<td>Well supported</td>
</tr>
<tr>
<td>Alloy 740H</td>
<td>DOE &amp; EERE</td>
<td>Well supported</td>
</tr>
<tr>
<td>Alloy 282</td>
<td>Literature, INL data</td>
<td>Somewhat supported</td>
</tr>
<tr>
<td>Alloy 230</td>
<td>Literature</td>
<td>Very tentative</td>
</tr>
</tbody>
</table>

Table 4.1: Overview of the material data source and the status of current material database of srlife package. # indicates data collected as part of this project.

4.1 Metal thermal properties

The package applies full transient thermal analysis and therefore requires both the thermal conductivity and diffusivity of the receiver material as a function of temperature. As indicated in Table 4.2, these properties come from ASME Code for all the materials except for Alloy 282 for which the data were extracted from the material data sheet [9]. The values of thermal conductivity and thermal diffusivity for all the materials are provided in Table 4.3 and Table 4.4, respectively.
4.1.1 Thermal conductivity

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>316H</th>
<th>800H</th>
<th>Alloy 617</th>
<th>Alloy 740H</th>
<th>Alloy 282</th>
<th>Alloy 230</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>14.1</td>
<td>11.5</td>
<td>10.5</td>
<td>10.2</td>
<td>10.3</td>
<td>9.0</td>
</tr>
<tr>
<td>50</td>
<td>14.6</td>
<td>12.1</td>
<td>11.1</td>
<td>10.8</td>
<td>10.6</td>
<td>9.5</td>
</tr>
<tr>
<td>100</td>
<td>15.4</td>
<td>13.0</td>
<td>12.1</td>
<td>11.7</td>
<td>12.0</td>
<td>10.4</td>
</tr>
<tr>
<td>150</td>
<td>16.1</td>
<td>13.9</td>
<td>13.0</td>
<td>12.4</td>
<td>13.1</td>
<td>11.4</td>
</tr>
<tr>
<td>200</td>
<td>16.8</td>
<td>14.7</td>
<td>14.0</td>
<td>13.0</td>
<td>14.1</td>
<td>12.4</td>
</tr>
<tr>
<td>250</td>
<td>17.6</td>
<td>15.5</td>
<td>14.9</td>
<td>13.8</td>
<td>15.2</td>
<td>13.5</td>
</tr>
<tr>
<td>300</td>
<td>18.3</td>
<td>16.4</td>
<td>15.8</td>
<td>14.5</td>
<td>16.3</td>
<td>14.5</td>
</tr>
<tr>
<td>350</td>
<td>19.0</td>
<td>17.1</td>
<td>16.6</td>
<td>15.1</td>
<td>17.4</td>
<td>15.5</td>
</tr>
<tr>
<td>400</td>
<td>19.7</td>
<td>17.9</td>
<td>17.4</td>
<td>15.7</td>
<td>18.5</td>
<td>16.5</td>
</tr>
<tr>
<td>450</td>
<td>20.5</td>
<td>18.7</td>
<td>18.2</td>
<td>16.4</td>
<td>19.5</td>
<td>17.5</td>
</tr>
<tr>
<td>500</td>
<td>21.2</td>
<td>19.4</td>
<td>18.9</td>
<td>17.1</td>
<td>20.5</td>
<td>18.5</td>
</tr>
<tr>
<td>550</td>
<td>21.9</td>
<td>20.3</td>
<td>19.6</td>
<td>17.8</td>
<td>21.6</td>
<td>19.5</td>
</tr>
<tr>
<td>600</td>
<td>22.6</td>
<td>21.1</td>
<td>23.3</td>
<td>18.4</td>
<td>22.6</td>
<td>20.4</td>
</tr>
<tr>
<td>650</td>
<td>23.2</td>
<td>22.0</td>
<td>25.3</td>
<td>19.3</td>
<td>23.7</td>
<td>21.4</td>
</tr>
<tr>
<td>700</td>
<td>23.9</td>
<td>22.9</td>
<td>25.7</td>
<td>20.2</td>
<td>24.8</td>
<td>22.4</td>
</tr>
<tr>
<td>750</td>
<td>24.6</td>
<td>23.8</td>
<td>25.3</td>
<td>21.2</td>
<td>25.5</td>
<td>23.4</td>
</tr>
<tr>
<td>800</td>
<td>-</td>
<td>-</td>
<td>25.2</td>
<td>22.1</td>
<td>26.1</td>
<td>24.3</td>
</tr>
<tr>
<td>850</td>
<td>-</td>
<td>-</td>
<td>25.5</td>
<td>23.0</td>
<td>26.7</td>
<td>25.1</td>
</tr>
<tr>
<td>900</td>
<td>-</td>
<td>-</td>
<td>26.2</td>
<td>23.8</td>
<td>27.3</td>
<td>25.9</td>
</tr>
</tbody>
</table>

Table 4.3: Thermal conductivity.

4.1.2 Thermal diffusivity

Note thermal diffusivity values for Alloy 740H and Alloy 282 are calculated from thermal conductivity and density values provided in Alloy 740H Code Case [11] and the material data sheet [12], respectively.
Temperature (°C) | 316H | 800H | Alloy 617 | Alloy 740H | Alloy 282 | Alloy 230
---|---|---|---|---|---|---
20 | 3.57 | 3.14 | 2.88 | 2.81 | 2.88 | 2.44
50 | 3.64 | 3.27 | 2.99 | 2.93 | 2.98 | 2.54
100 | 3.75 | 3.46 | 3.17 | 3.05 | 3.15 | 2.70
150 | 3.86 | 3.63 | 3.33 | 3.17 | 3.32 | 2.86
200 | 3.98 | 3.78 | 3.49 | 3.29 | 3.48 | 3.02
250 | 4.11 | 3.92 | 3.64 | 3.46 | 3.65 | 3.17
300 | 4.22 | 4.05 | 3.79 | 3.62 | 3.81 | 3.31
350 | 4.33 | 4.18 | 3.93 | 3.75 | 3.97 | 3.45
400 | 4.44 | 4.30 | 4.08 | 3.87 | 4.13 | 3.58
450 | 4.55 | 4.42 | 4.22 | 4.00 | 4.29 | 3.71
500 | 4.66 | 4.53 | 4.37 | 4.13 | 4.44 | 3.84
550 | 4.78 | 4.63 | 4.53 | 4.26 | 4.59 | 3.97
600 | 4.90 | 4.73 | 4.69 | 4.39 | 4.73 | 4.09
650 | 5.01 | 4.82 | 4.85 | 4.51 | 4.91 | 4.19
700 | 5.12 | 4.91 | 5.03 | 4.62 | 5.09 | 4.29
750 | 5.19 | 5.00 | 4.89 | 4.70 | 4.99 | 4.32
800 | - | - | 4.87 | 4.78 | 4.88 | 4.34
850 | - | - | 4.94 | 4.71 | 4.93 | 4.20
900 | - | - | 5.07 | 4.64 | 4.98 | 4.06

Table 4.4: Thermal diffusivity.

4.2 Metal deformation models

Structural analysis of the receiver requires deformation models describing the high temperature mechanical response of the material – how it deforms as a function of time, temperature, and stress. srlife outsources the actual material models to the NEML library, also developed at Argonne National Laboratory. NEML provides high temperature structural constitutive models, including models that account for creep and cyclic plasticity. srlife provides an interface to load material models for use in the receiver life estimation routines as well as simple, but reasonably accurate, high temperature constitutive models for the materials.

srlife provides three options for the material model selection – elastic model, elastic-creep model, and base model. The base model is a simple, but reasonably accurate, inelastic model accounting for time-independent plastic and time-dependent creep deformations. While the elastic-creep model does not account for time-independent plastic deformation and can be used for materials with very high yield stress such as Alloy 740H and Alloy 282. The elastic model is the simplest but most conservative among three. All the models are based on an additive, history-independent decomposition of the total strain, $\varepsilon$ into

$$
\varepsilon = \begin{cases} 
\varepsilon_{th} + \varepsilon_e + \varepsilon_p + \varepsilon_c; \text{base model} \\
\varepsilon_{th} + \varepsilon_e + \varepsilon_c; \text{elastic – creep model} \\
\varepsilon_{th} + \varepsilon_e; \text{elastic model}
\end{cases}
$$

(4.1)

where $\varepsilon_{th}$ is the thermal strain, $\varepsilon_e$ is the elastic strain, $\varepsilon_p$ is the plastic strain, and $\varepsilon_c$ is the creep strain.
4.2.1 Instantaneous coefficient of thermal expansion

The required material property for calculating $\varepsilon_{th}$ is the instantaneous coefficient of thermal expansion (CTE). Table 4.5 describes the sources and Table 4.6 provides the values of this material property. Note the instantaneous CTE for Alloy 282 are calculated values from mean CTE provided in the material data sheet [12].

<table>
<thead>
<tr>
<th>Material</th>
<th>CTE, E, v</th>
<th>Tensile test data</th>
<th>Minimum creep rate data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alloy 740H</td>
<td>ASME Code Case 2702 [11]</td>
<td></td>
<td>[27]</td>
</tr>
<tr>
<td>Alloy 282</td>
<td>Haynes material data sheet [12]</td>
<td></td>
<td>[19]</td>
</tr>
<tr>
<td>Alloy 230</td>
<td>ASME B&amp;PV Code Section II Part D [9]</td>
<td>[20-21]</td>
<td>[27-29]</td>
</tr>
</tbody>
</table>

Table 4.5: Sources for instantaneous coefficient of thermal expansion (CTE), modulus of elasticity (E), and Poisson’s ratio (v), tensile test data, and minimum creep rate data.

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>Instantaneous CTE (μm/mm/°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>15.3 14.2 12.6 12.4 12.1 12.4</td>
</tr>
<tr>
<td>50</td>
<td>16.0 14.9 12.8 12.4 12.1 12.5</td>
</tr>
<tr>
<td>100</td>
<td>17.0 15.8 13.0 12.4 12.1 12.8</td>
</tr>
<tr>
<td>150</td>
<td>17.8 16.4 13.3 13.0 12.4 13.1</td>
</tr>
<tr>
<td>200</td>
<td>18.4 16.7 13.6 13.6 12.6 13.4</td>
</tr>
<tr>
<td>250</td>
<td>18.8 17.0 14.0 14.0 13.1 13.9</td>
</tr>
<tr>
<td>300</td>
<td>19.1 17.2 14.4 14.3 13.5 14.3</td>
</tr>
<tr>
<td>350</td>
<td>19.3 17.4 14.8 14.7 13.7 14.8</td>
</tr>
<tr>
<td>400</td>
<td>19.5 17.7 15.2 15.1 13.9 15.2</td>
</tr>
<tr>
<td>450</td>
<td>19.8 18.0 15.7 15.4 14.5 15.5</td>
</tr>
<tr>
<td>500</td>
<td>20.2 18.3 16.2 15.6 15.0 15.9</td>
</tr>
<tr>
<td>550</td>
<td>20.6 18.6 16.7 15.8 14.9 15.2</td>
</tr>
<tr>
<td>600</td>
<td>21.1 19.0 17.2 16.0 14.7 16.4</td>
</tr>
<tr>
<td>650</td>
<td>21.6 19.4 17.8 16.9 15.9 16.6</td>
</tr>
<tr>
<td>700</td>
<td>21.7 19.9 18.4 17.7 17.1 16.8</td>
</tr>
<tr>
<td>750</td>
<td>21.2 20.7 19.1 19.1 18.4 17.2$^5$</td>
</tr>
<tr>
<td>800</td>
<td>-    -   19.8 20.4 19.6 17.6$^5$</td>
</tr>
<tr>
<td>850</td>
<td>-    -   20.5 18.5 21.7 18.0$^5$</td>
</tr>
<tr>
<td>900</td>
<td>-    -   21.2 16.5 23.7 18.4$^5$</td>
</tr>
</tbody>
</table>

Table 4.6: Instantaneous coefficient of thermal expansion. $^5$ indicates extrapolated values.
4.2.2 Modulus of elasticity

The elastic strain is calculated using the modulus of elasticity, $E$.

$$\varepsilon_e = \frac{\sigma}{E}$$  \hspace{1cm} (4.2)

The values of $E$ are provided in Table 4.7 while the sources are mentioned in Table 4.5.

<table>
<thead>
<tr>
<th>Temperature ($^\circ$C)</th>
<th>Modulus of elasticity (MPa)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>316H</td>
</tr>
<tr>
<td>20</td>
<td>195300</td>
</tr>
<tr>
<td>50</td>
<td>193000</td>
</tr>
<tr>
<td>100</td>
<td>189000</td>
</tr>
<tr>
<td>150</td>
<td>186000</td>
</tr>
<tr>
<td>200</td>
<td>183000</td>
</tr>
<tr>
<td>250</td>
<td>179000</td>
</tr>
<tr>
<td>300</td>
<td>176000</td>
</tr>
<tr>
<td>350</td>
<td>172000</td>
</tr>
<tr>
<td>400</td>
<td>169000</td>
</tr>
<tr>
<td>450</td>
<td>165000</td>
</tr>
<tr>
<td>500</td>
<td>160000</td>
</tr>
<tr>
<td>550</td>
<td>156000</td>
</tr>
<tr>
<td>600</td>
<td>151000</td>
</tr>
<tr>
<td>650</td>
<td>146000</td>
</tr>
<tr>
<td>700</td>
<td>140000</td>
</tr>
<tr>
<td>750</td>
<td>134000*</td>
</tr>
<tr>
<td>800</td>
<td></td>
</tr>
<tr>
<td>850</td>
<td></td>
</tr>
<tr>
<td>900</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: Modulus of elasticity. * indicates extrapolated values.

4.2.3 Poisson’s ratio

The database also needs to include Poisson’s ratio which is required for constructing the stiffness matrix. The values are provided in Table 4.8 and sources are in Table 4.5 for all six materials.

<table>
<thead>
<tr>
<th>Temperature ($^\circ$C)</th>
<th>Poisson’s ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>316H</td>
</tr>
<tr>
<td>20</td>
<td>0.310</td>
</tr>
<tr>
<td>100</td>
<td>0.310</td>
</tr>
<tr>
<td>200</td>
<td>0.310</td>
</tr>
<tr>
<td>300</td>
<td>0.310</td>
</tr>
<tr>
<td>400</td>
<td>0.310</td>
</tr>
<tr>
<td>500</td>
<td>0.310</td>
</tr>
<tr>
<td>600</td>
<td>0.310</td>
</tr>
<tr>
<td>700</td>
<td>0.310</td>
</tr>
<tr>
<td>800</td>
<td>-</td>
</tr>
<tr>
<td>900</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.8: Poisson’s ratio.
4.2.4 Hardening models

srlife uses Voce hardening model to describe $\varepsilon_p$ for all the materials.

$$\varepsilon_p = \frac{-1}{\delta} \ln \left( 1 - \frac{\sigma - \sigma_1}{\sigma_p - \sigma_1} \right)$$  \hspace{1cm} (4.3)

The coefficients in Equation 4.3 were determined from average tensile flow curves at different temperatures. The tensile test data for 316H and 800H come from the large historical database available in ASME Code background documents and literature. Some of these sources are listed in Table 4.5. Tensile test data for Alloy 740 were collected at INL and reported in [18].

Alloy 282 tensile test data were digitized from figures in a document received from DOE. These data are not in the cleanest form and therefore there is not a very high confidence on the model representing exact material behaviour. Figure 4.1 shows the Voce hardening model fit to the digitized data.

Alloy 230 tensile test data at 800°C and 900°C were digitized from [21]. Data for lower temperatures were digitized from another source [32] which collected the tensile test data only up to 0.6% strain. Figure 4.2 plots the Voce hardening model fit to Alloy 230 data.

Table 4.9 to Table 4.14 list the Voce hardening parameters as functions of temperature for all the alloys.

![Figure 4.1: Voce hardening model fit to the digitized tensile data for Alloy 282.](image)
Figure 4.2: Voce hardening model fit to the digitized tensile data for Alloy 230.

<table>
<thead>
<tr>
<th>T (°C)</th>
<th>$\sigma_1$ (MPa)</th>
<th>$\sigma_p$ (MPa)</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>240.31</td>
<td>477.28</td>
<td>40.08</td>
</tr>
<tr>
<td>93</td>
<td>233.70</td>
<td>449.18</td>
<td>42.86</td>
</tr>
<tr>
<td>204</td>
<td>225.17</td>
<td>408.60</td>
<td>48.51</td>
</tr>
<tr>
<td>316</td>
<td>215.40</td>
<td>368.69</td>
<td>55.54</td>
</tr>
<tr>
<td>427</td>
<td>205.69</td>
<td>331.82</td>
<td>64.46</td>
</tr>
<tr>
<td>538</td>
<td>193.48</td>
<td>293.71</td>
<td>76.31</td>
</tr>
<tr>
<td>593</td>
<td>187.02</td>
<td>275.21</td>
<td>83.82</td>
</tr>
<tr>
<td>649</td>
<td>180.05</td>
<td>256.48</td>
<td>93.11</td>
</tr>
<tr>
<td>704</td>
<td>167.28</td>
<td>232.96</td>
<td>103.50</td>
</tr>
<tr>
<td>760</td>
<td>137.76</td>
<td>203.44</td>
<td>98.51</td>
</tr>
</tbody>
</table>

Table 4.9: Voce hardening model parameters for 316H.

<table>
<thead>
<tr>
<th>T (°C)</th>
<th>$\sigma_1$ (MPa)</th>
<th>$\sigma_p$ (MPa)</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>158.25</td>
<td>510.46</td>
<td>22.83</td>
</tr>
<tr>
<td>93</td>
<td>155.39</td>
<td>506.74</td>
<td>22.47</td>
</tr>
<tr>
<td>204</td>
<td>150.84</td>
<td>500.75</td>
<td>21.90</td>
</tr>
<tr>
<td>316</td>
<td>146.23</td>
<td>494.60</td>
<td>21.33</td>
</tr>
<tr>
<td>427</td>
<td>140.85</td>
<td>487.54</td>
<td>20.64</td>
</tr>
<tr>
<td>538</td>
<td>135.47</td>
<td>480.36</td>
<td>19.96</td>
</tr>
<tr>
<td>593</td>
<td>132.80</td>
<td>476.75</td>
<td>19.62</td>
</tr>
<tr>
<td>649</td>
<td>129.30</td>
<td>379.62</td>
<td>26.50</td>
</tr>
<tr>
<td>704</td>
<td>125.75</td>
<td>243.75</td>
<td>54.46</td>
</tr>
<tr>
<td>760</td>
<td>116.41</td>
<td>175.24</td>
<td>105.58</td>
</tr>
</tbody>
</table>

Table 4.10: Voce hardening model parameters for 800H.
Table 4.11: Voce hardening model parameters for Alloy 617.

<table>
<thead>
<tr>
<th>T (°C)</th>
<th>σ₁ (MPa)</th>
<th>σₚ (MPa)</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>343.28</td>
<td>816.54</td>
<td>6.49</td>
</tr>
<tr>
<td>100</td>
<td>334.57</td>
<td>798.58</td>
<td>6.45</td>
</tr>
<tr>
<td>200</td>
<td>326.00</td>
<td>778.37</td>
<td>6.45</td>
</tr>
<tr>
<td>300</td>
<td>319.07</td>
<td>759.80</td>
<td>6.48</td>
</tr>
<tr>
<td>400</td>
<td>308.93</td>
<td>737.09</td>
<td>6.46</td>
</tr>
<tr>
<td>500</td>
<td>297.08</td>
<td>711.74</td>
<td>6.41</td>
</tr>
<tr>
<td>600</td>
<td>285.17</td>
<td>557.55</td>
<td>9.49</td>
</tr>
<tr>
<td>650</td>
<td>279.80</td>
<td>409.97</td>
<td>19.46</td>
</tr>
<tr>
<td>700</td>
<td>273.09</td>
<td>330.20</td>
<td>43.06</td>
</tr>
<tr>
<td>750</td>
<td>245.85</td>
<td>269.83</td>
<td>100.05</td>
</tr>
</tbody>
</table>

Table 4.12: Voce hardening model parameters for Alloy 740H.

<table>
<thead>
<tr>
<th>T (°C)</th>
<th>σ₁ (MPa)</th>
<th>σₚ (MPa)</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>585.97</td>
<td>748.97</td>
<td>142.37</td>
</tr>
<tr>
<td>700</td>
<td>585.97</td>
<td>748.97</td>
<td>142.37</td>
</tr>
<tr>
<td>750</td>
<td>567.40</td>
<td>717.40</td>
<td>112.45</td>
</tr>
<tr>
<td>800</td>
<td>527.03</td>
<td>622.03</td>
<td>97.97</td>
</tr>
<tr>
<td>850</td>
<td>438.62</td>
<td>468.62</td>
<td>721.90</td>
</tr>
</tbody>
</table>

Table 4.13: Voce hardening model parameters for Alloy 230.

<table>
<thead>
<tr>
<th>T (°C)</th>
<th>σ₁ (MPa)</th>
<th>σₚ (MPa)</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>538</td>
<td>259.69</td>
<td>314.70</td>
<td>107.46</td>
</tr>
<tr>
<td>649</td>
<td>248.08</td>
<td>298.08</td>
<td>136.90</td>
</tr>
<tr>
<td>760</td>
<td>221.10</td>
<td>271.09</td>
<td>136.93</td>
</tr>
<tr>
<td>800</td>
<td>187.45</td>
<td>242.50</td>
<td>95.05</td>
</tr>
<tr>
<td>900</td>
<td>109.68</td>
<td>139.67</td>
<td>133.11</td>
</tr>
</tbody>
</table>

Table 4.14: Voce hardening model parameters for Alloy 282.
4.2.5 Creep models

To model the time-dependent creep strain, $\varepsilon_c$, srlife considers a simple creep model based on the minimum creep rate, $\dot{\varepsilon}_{\text{min}}(T, \sigma)$, essentially adopted from the models of Kocks [30] and Mecking [31].

$$\varepsilon_c = \dot{\varepsilon}_{\text{min}} t = \dot{\varepsilon}_0 e^{B \mu b^3/(AkT)} \left( \frac{\sigma}{\mu} \right)^{-\mu b^3/(AkT)} t$$

(4.4)

where $\mu$ is the material shear stress given as $\mu = \frac{E}{2(1+\nu)}$, $k$ is the Boltzmann constant ($= 1.38 \times 10^{-23}$ J/K), $T$ is the absolute temperature, $b$ is a characteristic Burgers vector, $\dot{\varepsilon}_0$ is a reference strain rate, and $A$ and $B$ are constants.

Note the use of minimum creep rate will provide conservative estimation of the creep-fatigue life of the receiver tubes using the time-fraction creep damage approach adopted for the package, as it will produce slower stress relaxation when compared to the actual material response or a model including the details of primary creep. Table 4.5 lists the sources for the minimum creep rate data for all the materials. The values of the Kock-Mecking model parameters are provided in Table 4.15. Figure 4.3 shows the verification of the minimum creep rate data obeying the Kocks-Mecking form of the creep model for all the materials. However, as indicated by the figure the amount of minimum creep rate data for Alloy 230 is very limited and therefore the model parameters provided in Table 4.5 should be considered as very tentative for Alloy 230.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>316</th>
<th>800H</th>
<th>Alloy 617</th>
<th>Alloy 740H</th>
<th>Alloy 282</th>
<th>Alloy 230</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon_0$ (hr$^{-1}$)</td>
<td>1.5x10$^{11}$</td>
<td>3.0x10$^{18}$</td>
<td>5.0x10$^{15}$</td>
<td>1.0x10$^{13}$</td>
<td>5.0x10$^{8}$</td>
<td>1.0x10$^{13}$</td>
</tr>
<tr>
<td>B (mm)</td>
<td>2.02x10$^{-7}$</td>
<td>2.474x10$^{-7}$</td>
<td>2.474x10$^{-7}$</td>
<td>2.53x10$^{-7}$</td>
<td>2.53x10$^{-7}$</td>
<td>2.53x10$^{-7}$</td>
</tr>
<tr>
<td>B</td>
<td>-1.9812</td>
<td>-0.6457</td>
<td>-1.1903</td>
<td>-0.1470</td>
<td>0.0115</td>
<td>-0.3520</td>
</tr>
</tbody>
</table>

Table 4.15: Kock-Mecking minimum creep rate model parameters.
Figure 4.3: Kocks-Mecking diagram used to construct the creep rate model for different materials.
4.3 Metal damage properties

`srlife` uses an “ASME-style” damage model to predict the onset of creep-fatigue damage. This approach calculates the fatigue damage using strain-based fatigue diagrams and Miner’s rule, calculates the creep damage using a time-fraction approach, an effective stress (here assumed to be the von Mises stress), and stress rupture data, and then assess the combination of creep and fatigue damage using a creep-fatigue interaction diagram, based on creep-fatigue test data. The required material data are then time-to-creep rupture correlations, fatigue cycles-to-failure as a function of strain, and creep-fatigue damage envelope. Conventionally, the creep-fatigue interaction diagram is assumed to be temperature-independent. Table 4.16 lists the sources for all the required material data.

<table>
<thead>
<tr>
<th>Materials</th>
<th>Creep rupture</th>
<th>Fatigue</th>
<th>Creep-fatigue</th>
</tr>
</thead>
<tbody>
<tr>
<td>316H</td>
<td>Code background documents, [13, 32-34]</td>
<td>[8]</td>
<td>$</td>
</tr>
<tr>
<td>800H</td>
<td>Code background documents, [23, 43]</td>
<td>[8]</td>
<td>$</td>
</tr>
<tr>
<td>Alloy 617</td>
<td>[35-40], Code background documents</td>
<td>[10]</td>
<td>$</td>
</tr>
<tr>
<td>Alloy 740H</td>
<td>[18]</td>
<td>[18]</td>
<td>$</td>
</tr>
<tr>
<td>Alloy 282</td>
<td>[19], [12, 44], INL data</td>
<td>[12]</td>
<td>$</td>
</tr>
<tr>
<td>Alloy 230</td>
<td>[27, 41]</td>
<td>Not available</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.16: Sources for creep rupture, fatigue, and creep-fatigue data. $ indicates a creep-fatigue interaction diagram is already available, meaning data did not have to be collected for this project. # indicates data collected at INL as part of this project.

4.3.1 Time-to-creep rupture

A Larson-Miller correlation [42] is used interpolate/extrapolate creep rupture test data to determine the time-to-creep rupture equations. The Larson-Miller correlation is:

\[
LMP = (T + 273.15) \times (C + \log_{10} t_r) = a_0 (\log_{10} \sigma)^n + a_1 (\log_{10} \sigma)^{n-1} + \cdots + a_n
\]  (4.5)

with \( t_r \) being the time-to-rupture, \( \sigma \) is the stress, \( T \) is the temperature, \( C \) is the Larson-Miller constant, and \( a_i \)'s are the model coefficients, fit to data. The `srlife` package provides user with two options for time-to-creep rupture calculation – average rupture time based on the Larson-Miller fit to the experimental data and rupture time based on the 95% confidence lower bound prediction interval on the data. The latter is current ASME practice, but best suited for design and not best-estimates of the receiver life.

Creep rupture data of 316H, 800H, and Alloy 617 come from the large historical database available in ASME Code background documents and literature. Some of these sources are provided in Table 4.16. The equations and corresponding coefficients for time-to-creep rupture are provided in Table 4.17. These equations and coefficients were generated by finding the optimal value of \( C \) and the corresponding polynomial correlation between the LMP and the log of the stress that best fits the experimental data. The Larson-Miller correlation for 800H was directly taken from [43]. Figure 4.4 plots the Larson-Miller fits to the experimental data for 316H, Alloy 617, Alloy 740, Alloy 282, and Alloy 230.
Note the right side of Equation 4.5 uses a polynomial of the logarithm of stress. Higher order polynomials are sometimes used to achieve a good fit to the test data. However, higher order correlations can also cause errors when extrapolating outside the data set. As indicated by Figure 4.5, the 2nd order polynomial provides better, though not significantly, fit to the test data than the 1st order polynomial for Alloy 740H. However, since the lowest available stress data is 100 MPa, the extrapolation of the 2nd order polynomial fit to stress values lower than 10 MPa, far outside the test data, results in decreasingly shorter time-to-rupture with reducing stress at all temperatures. This issue can be avoided by using a 1st order polynomial, however, for some materials a linear relation clearly does not fit that data. If a linear correlation is inadequate, then different order of polynomial should be tried. For Alloy 740H, a 1st order polynomial provides good fit to the test data and also avoids the decreasing time-to-rupture estimation with decreasing stress below 10 MPa. Similarly, this extrapolation issue was checked for other materials before finalizing the time-to-rupture correlations. Figure 4.6 plots the time-to-rupture vs stress at different temperatures for other alloys using the equations in Table 4.17. The figure indicates no issue with extrapolation.

<table>
<thead>
<tr>
<th>Materials</th>
<th>Equations: $t_r = f(T, \sigma)$</th>
<th>Coefficients</th>
<th>Average rupture</th>
<th>95% prediction lower bound rupture</th>
</tr>
</thead>
<tbody>
<tr>
<td>316H</td>
<td>$10^{\left(\frac{a_3(\log_{10}\sigma)^3 + a_2(\log_{10}\sigma)^2 + a_1(\log_{10}\sigma) + a_0}{7 + 273.15} - C\right)}$</td>
<td>$C = 17.16$</td>
<td>$C = 17.16$</td>
<td>$a_3 = -1475.24$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2 = 7289.42$</td>
<td>$a_2 = 6958.39$</td>
<td>$a_1 = -16642.65$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_0 = 35684.60$</td>
<td>$a_0 = 34564.80$</td>
<td></td>
</tr>
<tr>
<td>800H</td>
<td>$10^{\left(\frac{a_2(\log_{10}\sigma)^2 + a_1(\log_{10}\sigma) + a_0}{7 + 273.15} - C\right)}$</td>
<td>$C = 15.48$</td>
<td>$C = 15.48$</td>
<td>$a_2 = -268.31$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1 = -4847.72$</td>
<td>$a_1 = -4774.89$</td>
<td>$a_0 = 28978.25$</td>
</tr>
<tr>
<td>Alloy 617</td>
<td>$10^{\left(\frac{a_1(\log_{10}\sigma) + a_0}{7 + 273.15} - C\right)}$</td>
<td>$C = 13.22$</td>
<td>$C = 13.22$</td>
<td>$a_1 = -4676.95$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_0 = 26779.12$</td>
<td>$a_0 = 25701.18$</td>
<td></td>
</tr>
<tr>
<td>Alloy 740H</td>
<td>$10^{\left(\frac{a_1(\log_{10}\sigma) + a_0}{7 + 273.15} - C\right)}$</td>
<td>$C = 18.29$</td>
<td>$C = 18.29$</td>
<td>$a_1 = -5884.39$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_0 = 36280.37$</td>
<td>$a_0 = 35942.33$</td>
<td></td>
</tr>
<tr>
<td>Alloy 282</td>
<td>$10^{\left(\frac{a_3(\log_{10}\sigma)^3 + a_2(\log_{10}\sigma)^2 + a_1(\log_{10}\sigma) + a_0}{7 + 273.15} - C\right)}$</td>
<td>$C = 16.07$</td>
<td>$C = 16.07$</td>
<td>$a_3 = -2101.90$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_2 = 11504.89$</td>
<td>$a_2 = 11359.31$</td>
<td>$a_1 = -24360.85$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_0 = 41461.36$</td>
<td>$a_0 = 40726.06$</td>
<td></td>
</tr>
<tr>
<td>Alloy 230</td>
<td>$10^{\left(\frac{a_1(\log_{10}\sigma) + a_0}{7 + 273.15} - C\right)}$</td>
<td>$C = 11.28$</td>
<td>$C = 11.28$</td>
<td>$a_1 = -4208.21$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_0 = 23255.67$</td>
<td>$a_0 = 22130.61$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.17: Time-to-creep rupture equations.
Figure 4.4: Larson Miller plots for (a) 316H, (b) Alloy 617, (c) Alloy 282, (d) Alloy 740H, and (e) Alloy 230.
Figure 4.5: Larson-Miller plots and the corresponding stress vs time-to-rupture plots compared between the 2nd order polynomial and the 1st order polynomial considered in Larson-Miller correlation for Alloy 740H.
4.3.2 Fatigue cycles-to-failure

Similar to the time-to-creep rupture the *srlife* package also uses equations calibrated to fatigue test data to determine the fatigue cycles-to-failure for a given strain range and temperature. The fatigue test data were fitted using polylogarithmic functions.
\[ N = \begin{cases} 10^{(a_n(\log_{10}\Delta\varepsilon)^n + a_{n-1}(\log_{10}\Delta\varepsilon)^{n-1} + \ldots + a_0)} & \text{if } \Delta\varepsilon > \Delta\varepsilon_{\text{cutoff}} \\ 10^{(a_n(\log_{10}\Delta\varepsilon_{\text{cutoff}})^n + a_{n-1}(\log_{10}\Delta\varepsilon_{\text{cutoff}})^{n-1} + \ldots + a_0)} & \text{if } \Delta\varepsilon \leq \Delta\varepsilon_{\text{cutoff}} \end{cases} \]  

(4.6)

Table 4.16 lists the sources of fatigue data for all the alloys. For 316H, Alloy 800H, and Alloy 617, the strain range versus allowable cycles data of the design fatigue curves provided in the Section III Division 5 of ASME B&PV Code [8] and Alloy 617 Code Case [10] were used to back-calculate the average fatigue curves. The average fatigue cycles-to-failure, \( N \) at a strain range, \( \Delta\varepsilon \)

\[ N = \max\{N_{\text{design}}\left(\frac{\Delta\varepsilon}{2}\right), N_{\text{design}}(\Delta\varepsilon) \times 20\} \]  

(4.7)

with \( N_{\text{design}} \) is being the design allowable cycles at \( \Delta\varepsilon \). The 2 and 20, respectively, are the margins on strain range and cycles originally applied to the average fatigue data. An example of this process is shown in Figure 4.7.

For Alloy 740H, original fatigue experimental data from [18] were used. Fatigue curves for 316H, 800H, Alloy 617 and Alloy 740H are plotted in Figure 4.8. The fatigue data for Alloy 230 were extracted from its material data sheet [12] and plotted along with the fatigue curves in Figure 4.9. A detailed discussion on the processing of the literature data and the INL test data, presented in Chapter 6, to determine the fatigue curves for Alloy 282 is discussed in the next section. Table 4.18 lists the fatigue cycles-to-failure equations for all the materials. The cutoff strain in these equations, also shown as an example in Figure 4.7, is to ensure interpolation within the available data.

Figure 4.7: Example fatigue curve fitting using ASME Section III, Division 5 design fatigue curve. The cutoff strain is used to limit the fatigue equations to interpolation within available data.
Figure 4.8: Average fatigue curves for 316H, 800H, Alloy 617, and Alloy 740H.

Figure 4.9: Average fatigue curves for Alloy 230 along with fatigue test data.
### Materials Equations

\[
N = \begin{cases} 
    f(T, \Delta \varepsilon) ; \text{if } \Delta \varepsilon > \text{cutoff strain} \\
    f(T, \text{cutoff strain}) ; \text{if } \Delta \varepsilon \leq \text{cutoff strain}
\end{cases}
\]

<table>
<thead>
<tr>
<th>Materials</th>
<th>Cutoff strain</th>
<th>Temperature ( T ) (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>316H</td>
<td>2.0 ( \times ) 10^{-3}</td>
<td>( T \leq 313 )</td>
</tr>
<tr>
<td></td>
<td>1.8 ( \times ) 10^{-3}</td>
<td>313 &lt; ( T \leq 698 )</td>
</tr>
<tr>
<td></td>
<td>1.4 ( \times ) 10^{-3}</td>
<td>698 &lt; ( T \leq 753 )</td>
</tr>
<tr>
<td></td>
<td>1.2 ( \times ) 10^{-3}</td>
<td>753 &lt; ( T \leq 923 )</td>
</tr>
<tr>
<td></td>
<td>1.0 ( \times ) 10^{-3}</td>
<td>923 &lt; ( T \leq 978 )</td>
</tr>
<tr>
<td>800H</td>
<td>2.5 ( \times ) 10^{-3}</td>
<td>( T \leq 698 )</td>
</tr>
<tr>
<td></td>
<td>2.8 ( \times ) 10^{-3}</td>
<td>698 &lt; ( T \leq 813 )</td>
</tr>
<tr>
<td></td>
<td>2.0 ( \times ) 10^{-3}</td>
<td>813 &lt; ( T \leq 923 )</td>
</tr>
<tr>
<td></td>
<td>1.8 ( \times ) 10^{-3}</td>
<td>923 &lt; ( T \leq 1033 )</td>
</tr>
<tr>
<td>Alloy 617</td>
<td>2.5 ( \times ) 10^{-3}</td>
<td>( T \leq 698 )</td>
</tr>
<tr>
<td></td>
<td>1.8 ( \times ) 10^{-3}</td>
<td>698 &lt; ( T \leq 977 )</td>
</tr>
<tr>
<td></td>
<td>1.4 ( \times ) 10^{-3}</td>
<td>977 &lt; ( T \leq 1144 )</td>
</tr>
<tr>
<td></td>
<td>1.1 ( \times ) 10^{-3}</td>
<td>1144 &lt; ( T \leq 1223 )</td>
</tr>
<tr>
<td>Alloy 740H</td>
<td>2.0 ( \times ) 10^{-3}</td>
<td>( T \leq 973 )</td>
</tr>
<tr>
<td></td>
<td>1.0 ( \times ) 10^{-3}</td>
<td>973 &lt; ( T \leq 1123 )</td>
</tr>
<tr>
<td>Alloy 282</td>
<td>4.4 ( \times ) 10^{-3}</td>
<td>( T \leq 922 )</td>
</tr>
<tr>
<td></td>
<td>3.8 ( \times ) 10^{-3}</td>
<td>922 &lt; ( T \leq 1033 )</td>
</tr>
<tr>
<td></td>
<td>3.6 ( \times ) 10^{-3}</td>
<td>1033 &lt; ( T \leq 1089 )</td>
</tr>
<tr>
<td></td>
<td>2.5 ( \times ) 10^{-3}</td>
<td>1089 &lt; ( T \leq 1144 )</td>
</tr>
<tr>
<td>Alloy 230</td>
<td>4.8 ( \times ) 10^{-3}</td>
<td>( T \leq 700 )</td>
</tr>
<tr>
<td></td>
<td>4.5 ( \times ) 10^{-3}</td>
<td>700 &lt; ( T \leq 811 )</td>
</tr>
<tr>
<td></td>
<td>4.2 ( \times ) 10^{-3}</td>
<td>811 &lt; ( T \leq 922 )</td>
</tr>
<tr>
<td></td>
<td>3.2 ( \times ) 10^{-3}</td>
<td>922 &lt; ( T \leq 1033 )</td>
</tr>
<tr>
<td></td>
<td>2.4 ( \times ) 10^{-3}</td>
<td>1033 &lt; ( T \leq 1144 )</td>
</tr>
<tr>
<td></td>
<td>2.1 ( \times ) 10^{-3}</td>
<td>1144 &lt; ( T \leq 1255 )</td>
</tr>
</tbody>
</table>

**Table 4.18: Fatigue cycles-to-failure equations.**

### 4.3.2.1 Alloy 282 fatigue curve

Figure 4.10 plots all the fatigue data gathered from literature along with the INL test data collected as part of this project. The data includes test conducted by different organizations – Haynes, General Electric (GE), Siemens-Westinghouse (SW) and INL – on specimens with different heat treatments – solution annealed (SA), aged, and cast – at different test temperatures. However, there is only one test data for cast specimen, conducted at INL.
Overall the aged specimens show longer fatigue life than the SA specimens when compared among the test data from same organization. However, the GE specimens show significantly shorter fatigue life than the specimens tested at other places. The test data from Haynes are only on SA specimen and at temperatures 649°C, 760°C, 816°C, and 871°C. These data show a consistent temperature trend. As Haynes is the manufacturer of Alloy 282 and SA specimen show shorter fatigue life than aged specimens, which in turn will provide a conservative estimation of fatigue damage for the receiver, we used the Haynes data for constructing the fatigue curves for Alloy 282. Figure 4.10 includes the fatigue curves fitted to Haynes data at different temperatures.

Figure 4.10: Average fatigue curves for Alloy 282 along with fatigue test data from different sources. Test data include solution annealed (SA), aged, and cast specimens tested at Haynes, General Electric (GE), Siemens-Westinghouse (SW), and INL.

To assess how close the SA test data from INL to the fatigue curves, constructed using Haynes SA data, Figure 4.11 plots them separately for better visualization. The figure indicates all the INL’s 750°C test data are close to the 760°C fatigue curve. INL’s 850°C data are in between the 816°C and 870°C curves, although they seem to be closer to 816°C curves than a linear interpolation between 816°C and 870°C curves would predict. These could be due to significant fatigue strength reduction for the material above 850°C. For this reason and considering the variation in fatigue data from different sources, the calculation for determining the creep-fatigue damage envelop using INL creep-fatigue test data, as discussed below, uses INL specific fatigue test data instead of interpolating the fatigue curves constructed here.
4.3.3 Creep-fatigue damage interaction

Finally, a creep-fatigue damage envelop is used to determine the creep-fatigue life given the fatigue damage fraction and creep damage fractions. Table 4.19 lists the intersection point of the damage envelope for all the materials and their sources. The intersection points for 316H and Alloy 800H are from Section III Division 5 of ASME Code. For Alloy 617 and Alloy 740H, the intersection points are from Alloy 617 Code Case and [2], respectively.

As part of this project, INL collected creep-fatigue test data for Alloy 282. The section below discusses the construction of Alloy 282 creep-fatigue damage envelop using INL test data.

There is no available creep-fatigue data for Alloy 230 and therefore *srlife* package uses Alloy 617 damage envelope for Alloy 230, as they exhibit similarity in other mechanical properties. However, unlike Alloy 282 there is no plan to update the damage envelope for Alloy 230, which will remain approximate and not suitable for detailed design work.
Alloys | Intersection \((D_f, D_c)\) | Source / Notes |
---|---|---|
316H | \((0.3, 0.3)\) | [8] |
800H | \((0.1, 0.1)\) | [8] |
Alloy 617 | \((0.1, 0.1)\) | [10] |
Alloy 740H | \((0.05, 0.05)\) | [2] |
Alloy 282 | \((0.1, 0.1)\) | Based on INL test data (see Section 4.3.3.1 for details) |
Alloy 230 | \((0.1, 0.1)\) | Creep fatigue data is not available, srlife currently uses Alloy 617 envelope |

Table 4.19: Creep-fatigue damage envelope.

### 4.3.3.1 Alloy 282 creep-fatigue damage envelop

We processed the INL creep-fatigue test data on Alloy 282 to determine an approximate creep-fatigue interaction diagram. Creep-fatigue interaction is the experimental observation, evident in the INL test data, that tests with a combination of strain-controlled fatigue cycling with holds at one or both ends of the cycle fail sooner than a corresponding pure fatigue test without the holds. The creep-fatigue interaction diagram is a design tool providing an acceptance envelope for designers to assess components against creep-fatigue failure. The designer calculates the individual creep and fatigue damage fractions for each location in the component, as discussed previously, and plots that point as the pair \((D_f, D_c)\) on the diagram. If the point falls under the interaction envelope than it is acceptable, indicating a reasonable assurance that the component will not initiated a creep-fatigue crack at that location.

Forming the creep-fatigue diagram requires three sources of data, two of which we discussed previously:

1. Creep rupture data correlated to a model providing the average expected time to failure for a given combination of temperature and stress. We use the Larson-Miller correlation for Alloy 282, discussed previously, for this requirement here.

2. Pure, strain-based fatigue data, correlated to model providing the average expected number of cycles to failure for a given combination of temperature and strain range. Again, we discussed temperature-dependent fatigue curves for Alloy 282 earlier.

3. Creep-fatigue data from which to generate the diagram.

Given these prerequisites, forming the diagram involves calculating a test-specific value of the fatigue and creep damage, plotting these \((D_f, D_c)\) points on a diagram, and drawing an envelope that bounds most of the test data. Specifically, for each test this process is:

1. The experiment controls the strain range and temperature. From this data, calculate the number of cycles to failure, \(N_f\), from the fatigue correlation. Calculate the fatigue damage as the ratio of the actual number of cycles to failure in the test, \(N\), to the expected number of cycles to failure, i.e. \(D_f = N / N_f\).
2. Extract the experimental stress history for the mid-life cycle (i.e., cycle $N/2$). Call this stress history $\sigma(t)$, i.e., the time-dependent uniaxial stress in the sample as a function of time. Calculate the creep damage using the time-fraction rule and by assuming that this single cycle is representative of the entire specimen history. This gives the creep damage as $D_c = N \int_0^{t_{cycle}} \frac{dt}{t_R(T, \sigma)}$ where $t_R$ indicates the expected time to rupture given the temperature and stress, from the Larson-Miller correlation.

This process relies on using average, best-estimate values for the material creep and fatigue strength, not design lower-bound correlations.

We applied this process to the INL creep-fatigue data with one variation. Rather than use the fatigue curves described above, which use several sources of data spanning several heats of material, we used an INL-specific fatigue correlation based on the pure fatigue tests corresponding to each INL creep-fatigue test. That is, rather than use $N_f$ from the all-heat fatigue curves, we took $N_f$ for each creep-fatigue test as the number of cycles to failure from the INL test with the same strain range and temperature. If there were multiple repeats at the same condition in the test database we took the average of the repeated tests. Table 4.20 shows the resulting $N_f$ values for the test conditions required to process the creep-fatigue data.

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Strain range</th>
<th>$N_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>750°C</td>
<td>0.6%</td>
<td>36,044</td>
</tr>
<tr>
<td>750°C</td>
<td>1.0%</td>
<td>1,058</td>
</tr>
<tr>
<td>850°C</td>
<td>0.4%</td>
<td>65,788</td>
</tr>
<tr>
<td>850°C</td>
<td>1.0%</td>
<td>706</td>
</tr>
</tbody>
</table>

Table 4.20: INL-specific fatigue life data used to generate the creep-fatigue diagram.

Figure 4.12 shows the resulting creep-fatigue data, plotted on the interaction diagram. This plot shows two design creep-fatigue envelopes: one with an intersection of (0.05, 0.05) which we originally used as an approximation for Alloy 282 in preliminary designs and our final recommendation with an intersection of (0.1, 0.1).

The original, preliminary guess for the location of the intersection point for Alloy 282 was based on our recommendation for Alloy 740H [2], in turn based on creep-fatigue data generated through a previous project. Alloy 740H is a similar Ni-based superalloy, comparable to Alloy 282. However, the test data for Alloy 282 generated in this project suggests that 282 may have somewhat superior creep-fatigue strength compared to 740H, while retaining comparable pure fatigue and pure creep rupture strength. However, the current project generated comparatively few creep-fatigue tests to support the D-diagram and the data is somewhat conflicted with the aging issue, discussed in Chapter 6. However, there is sufficient data to recommend the design envelope in Figure 4.12 for use in CSP component design.
Figure 4.12: Creep-fatigue data plotted on the interaction diagram and two design envelopes: our original recommendation with an intersection at (0.05,0.05) and the final recommendation with an intersection at (0.1,0.1).
5 Applications

This chapter performs several example design and analysis studies on the reference receiver designs discussed in Chapter 3. These design and analysis studies demonstrate the utility of *srlife* not only in optimizing the design of a receiver but also selecting the right material for the receiver. These design studies also assess the effect of material model used in analysis on the life estimation and how the estimation of individual panel lives can be used to increase overall receiver life by replacing some of the panels.

The chapter also provides an overview of the design optimization studies conducted by ANU for a Gen3 sodium-cooled receiver utilizing *srlife*.

5.1 Design and analysis studies

5.1.1 Receiver life vs manifold and system stiffnesses

As discussed in Chapter 2, *srlife* provides a simplified parametric approach for modelling mechanical constraint on the tubes provided by the stiffness of the tube manifold and the mechanical constraint on the panels provided by the supports connecting each panel to the tower. This section investigates how the manifold and system stiffnesses affect the life of the reference receiver designs.

Table 5.1 and Table 5.2 list the lives of the receivers as a function of the manifold and system stiffnesses for the high and low temperature receivers, respectively. The lives are calculated from 2D analysis at the maximum temperature location considering an elastic-creep deformation model for the materials. For both the receivers, there is a significant influence of system stiffness on the receiver life. The receiver life decreases drastically when panels are mechanically coupled. On the other hand, the effect of manifold stiffness on receiver life is not as severe as the system stiffness. This can be explained by considering the temperature difference within a group of mechanically coupled tubes. The temperature difference between the hottest and coldest tubes within a panel is much less than the difference between the hottest and coldest tubes in a whole receiver. When all the panels are mechanically coupled the higher temperature difference among all the tubes in the receiver apply much higher axial constraint on the tube resulting in higher stresses and therefore higher creep-fatigue damage and eventually shorter life. This suggests that the panels must be mechanically disconnected for achieving longer service lives. In actual receiver designs the panels are likely only loosely coupled, so no changes to design practice is needed.

<table>
<thead>
<tr>
<th>Stiffness</th>
<th>System (panel k)</th>
<th>Disconnect</th>
<th>$10^3$</th>
<th>$10^6$</th>
<th>Rigid</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Manifold (tube k)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconnect</td>
<td>3706</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$10^3$</td>
<td>3445</td>
<td>3179</td>
<td>1486</td>
<td>1482</td>
<td></td>
</tr>
<tr>
<td>$10^6$</td>
<td>3181</td>
<td>2578</td>
<td>216</td>
<td>214</td>
<td></td>
</tr>
<tr>
<td>Rigid</td>
<td>3165</td>
<td>2577</td>
<td>215</td>
<td>214</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Life of the high temperature reference receiver as a function of manifold and system stiffness. Results are from 2D analysis at the maximum temperature location. The material is Alloy 740H and the deformation model is elastic-creep.
While designing receivers with mechanically disconnected panels is easy, completely decoupling of the tubes within a panel is almost impossible. Here we attempt to estimate the stiffness of the manifold by considering the manifold as a fixed beam under uniformly distributed load imposed by the tubes. Figure 5.1 illustrates the model. Table 5.3 provides the material and geometry information of the manifold along with the design checks. The closed form solution for the deflection, \( \delta \) of a rigid beam under uniformly distributed load is a function of the distance from the beam end, \( x \):

\[
\delta = -\frac{wx^2}{24EI} (L - x)^2
\]  

(5.1)

where E is the elastic modulus, I is the area moment of inertia, L is the length of the manifold, and w is the load per unit length applied on the manifold by the tubes. Based on this equation the manifold stiffness varies from infinite at the fixed supports to about 20 N/mm at the center for both receivers. Figure 5.2 plots an example of how stiffness may change from the support to the center of the manifold. The plot suggests keeping some space between the manifold supports and the tubes will be beneficial for longer life of the tubes.

The simple beam model gives an idea of how stiffness may change along the manifold length and what maximum and minimum values could be. An accurate estimation of the manifold stiffness would require performing detailed finite element analysis. However, the good news is that the life of the receiver is not significantly affected by the manifold stiffness. Therefore, a conservative approach would be considering infinite stiffness, i.e. “rigid” spring, for the manifold.
### Table 5.3: Manifold design check for the reference receivers.

<table>
<thead>
<tr>
<th></th>
<th>Low temperature receiver</th>
<th>High temperature receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiver diameter, $D$</td>
<td>17 m</td>
<td>17 m</td>
</tr>
<tr>
<td>Manifold length, $L = \pi D/(2*6)$</td>
<td>4.45 m</td>
<td>4.45 m</td>
</tr>
<tr>
<td>Manifold design pressure</td>
<td>1 MPa</td>
<td>3 MPa</td>
</tr>
<tr>
<td>Manifold design temperature</td>
<td>600°C</td>
<td>750°C</td>
</tr>
<tr>
<td>Manifold pipe outer diameter</td>
<td>100.58 mm</td>
<td>100.58 mm</td>
</tr>
<tr>
<td>Manifold pipe thickness</td>
<td>10.16 mm</td>
<td>10.16 mm</td>
</tr>
<tr>
<td>Manifold material</td>
<td>316H</td>
<td>316H</td>
</tr>
<tr>
<td>$S_o$</td>
<td>76 MPa (316H, at 600°C)</td>
<td>18 MPa (316H, at 750°C)</td>
</tr>
<tr>
<td>Stress intensity due to pressure, $P_m$</td>
<td>$4.45 MPa &lt; S_o$</td>
<td>$13.35 MPa &lt; S_o$</td>
</tr>
</tbody>
</table>

Table 5.4 compares the lives of the reference receivers for the different deformation models available in srlife. As expected the elastic model provides the most conservative estimation of the life, followed by the elastic-creep model. The base model which accounts for both the time-independent plastic deformation and time-dependent creep deformation provides the most accurate estimation of the receiver life.

### Table 5.4: Receiver life (days) estimation as a function of material model.

<table>
<thead>
<tr>
<th>Model</th>
<th>High temperature receiver Material: Alloy 740H</th>
<th>Low temperature receiver Material: Alloy 230</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base (Full inelastic)</td>
<td>3197</td>
<td>24380</td>
</tr>
<tr>
<td>Elastic-creep</td>
<td>3165</td>
<td>24136</td>
</tr>
<tr>
<td>Elastic</td>
<td>3116</td>
<td>24133</td>
</tr>
</tbody>
</table>

Table 5.4: Receiver life (days) estimation as a function of material model. Panels are disconnected but tubes are rigidly connected to the manifold. Material model is elastic-creep. Results are from 2D analysis of tubes sliced at maximum temperature location. Panels are disconnected but tubes are rigidly connected to the manifold.
5.1.3 Receiver life vs materials

Table 5.5 compares the life of the reference receivers for different structural alloys. As expected, the life of the low temperature receiver is very long when high temperature nickel based alloys are used. On the other hand, the high temperature receiver has zero life when austenitic steel, 316H is used.

Although both are considered to be similar materials, the life of the high temperature receiver is found to be much longer for Alloy 282 than Alloy 740H. The comparison between the creep-fatigue test data between two alloys also suggests superior creep-fatigue performance for Alloy 282.

<table>
<thead>
<tr>
<th>Materials</th>
<th>Low temperature receiver</th>
<th>High temperature receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alloy 740H</td>
<td>Infinite</td>
<td>3197</td>
</tr>
<tr>
<td>Alloy 282</td>
<td>Infinite</td>
<td>7251</td>
</tr>
<tr>
<td>Alloy 617</td>
<td>394828</td>
<td>361</td>
</tr>
<tr>
<td>Alloy 230</td>
<td>24380</td>
<td>246</td>
</tr>
<tr>
<td>800H</td>
<td>51615</td>
<td>119</td>
</tr>
<tr>
<td>316H</td>
<td>23994</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.5: Receiver life (days) estimation for different alloys. Results are based on 2D analysis at the maximum temperature location and using the base (i.e. full inelastic) deformation model for all the alloys. Panels are disconnected but tubes are rigidly connected to the manifold.

5.1.4 Individual panel lives

Tubes in all the panels in a receiver will not fail at the same time as they experience very different temperature and stresses. Table 5.6 lists the life of individual panels in the reference receivers. The table indicates some of the panels will last much longer than the other panels. Therefore, by replacing the panels that are close to the end of their service life the overall life of the receiver can be greatly increased. For instance, if the panels with ID 1, 2, and 3 in the high temperature receiver are replaced with new panels after 8.75 years the overall life of the receiver will increase to 17.5 years.

<table>
<thead>
<tr>
<th>Panel ID</th>
<th>High temperature Chloride salt receiver: Material: Alloy 740H</th>
<th>Low temperature Nitrite salt receiver: Material: Alloy 230</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9969</td>
<td>109263</td>
</tr>
<tr>
<td>1</td>
<td>3408</td>
<td>62300</td>
</tr>
<tr>
<td>2</td>
<td>3197</td>
<td>31563</td>
</tr>
<tr>
<td>3</td>
<td>3458</td>
<td>24380</td>
</tr>
<tr>
<td>4</td>
<td>6768</td>
<td>48517</td>
</tr>
<tr>
<td>5</td>
<td>23239</td>
<td>69546</td>
</tr>
</tbody>
</table>

Table 5.6: Comparison of individual panel life (days) calculated from 2D analysis using the base model. Panels are disconnected but tubes are rigidly connected to the manifold.
5.2 Overview of ANU’s receiver design optimization

As part of this project, the Australian National University integrated beta versions of \textit{srlife} into their receiver design and optimization process as they worked on the design of a sodium CSP system. The purpose, at least from the perspective of this project, was to test the package during development in an actual CSP design environment. ANU issued a separate technical report entitled "The Gen3 Liquid Sodium Receiver: An Evaluation of \textit{srlife}.” This report fully-documented ANU”s work on this effort and so this report does not include a complete repeat of the information contained in this companion report. However, a brief summary of their conclusions follows here:

- ANU validated the accuracy of \textit{srlife} versus detailed calculations using code\_ASTER, providing a second source of detailed verification for the package, in addition to the comparisons to MOOSE calculations discussed above.

- ANU provided suggestions on creep-fatigue damage extrapolation, which ultimately led to the options discussed above.

- The ANU work notes that connecting 2D generalized plane strain models of tubes together in panels using stiff springs tends to overestimate the stiffness of the manifold connection (because the 2D analysis misses the axial compliance of the rest of the tube). However, this approximation is conservative. A full 3D analysis of each tube, possible in \textit{srlife}, would not include this inaccuracy and so it is bounded in the 1D/2D/3D design comparison reported above.

- In the process of working with the beta versions of \textit{srlife} ANU discovered several bugs and errors which were corrected in the course of developing the final package.
6  Alloy 282 Test Results

The design tool developed within this project was to include Haynes 282, however, additional material data was required and collected as a part of this project. This largely focused on creep-fatigue testing, where significant gaps in the data existed. Other data (creep, tensile) existed through testing within other programs. A small amount of creep and tensile testing was performed within this project, to confirm the plate properties and fill in portions of the data set required for the design tool. This work focuses primarily on testing of Haynes 282 wrought plate. Cast material was considered for inclusion within the testing matrix, however, the limited scope of this project did not allow for significant testing on cast material.

The material tested in this study was Heat 2082 8 8433, provided by Haynes International. The composition of this plate is shown in Table 6.1 below. Specimens were machined in accordance with ASTM E8, ASTM E21, ASTM E139, ASTM E606, and ASTM E2714 standards. The schematic of the tensile/creep specimens is shown in Figure 6.1 and Figure 6.2 shows the schematic of the specimen used for fatigue and creep-fatigue testing. The longitudinal axis of the specimens was machined in the same axis as the rolling direction. Specimens were not taken from within 0.25 inches of the plate edges. The testing performed within this project is derived from the American Society of Mechanical Engineers (ASME) Boiler and Pressure Vessel Code (BPVC) which does not consider environment, so all testing was performed in lab air.

| Heat Al B C Co Cr Cu Fe Mn Mo Ni P S Si Ti W Cb(Nb) Ta Zr |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 2082 8 8433 | 1.54     | 0.005    | 0.06     | 10.2     | 19.54    | 0.01     | 0.82     | 0.1      | 0.49     | Bal      | 0.003    | 0.002    | 0.06     | 2.07     | 0.13     | <0.01    | 0.011   |

Table 6.1: Composition of Haynes 282 Heat 2082 8 8433.

![Figure 6.1: Schematic of threaded specimens used for tensile and creep tests. Units are in inches.](image-url)
Testing was performed on material in both the solution annealed state and single-step aged condition. All creep and tensile specimens tested were in the solution annealed state. Fatigue and creep-fatigue specimens in both states were tested, and the state of the specimens are indicated in the results within those sections. For aged specimens, a single step aging practice was followed, with the specimens held at 800 °C for four hours followed by air cooling to room temperature.

Previous work has been performed looking at the effects of the state of Haynes 282, particularly as part of the ultrasupercritical program within the DOE fossil energy office [44]. Three states of Haynes 282 were examined within the fossil energy work: Solution annealed (SA), peak aged-hardened (PA), and overaged (OA). The aging treatment used for peak aged-hardened was a two step process, with holds at 1010 °C and 788 °C for two and eight hours respectively. Overaged used that same heat treatment with an additional 250 hours at 774 °C. Starting around 760 °C, the difference in tensile properties decreased significantly between the three states of Haynes 282. A number of fatigue and creep tests were run as part of the fossil energy program. The results are recreated in Figure 6.3. The fatigue tests were stopped at approximately 100,000 hours and considered runoff tests. The results show similar properties between the three states tested. The SA having somewhat poorer fatigue strength, which would result in more conservative design models if SA was used rather than PA. For creep properties, the SA and OA appear in line with each other in the Larson Miller Plot, and slightly better than the PA, but it is difficult to tell if this is significant, given the usual scatter in creep data. The fossil energy program noted self-ageing occurring during slow (not tensile) high temperature tests, and even in large components after hot forging/extrusion.
While room temperature and lower temperature tensile tests would see significant differences between the aged and solution annealed material states, the differences are less pronounced for high temperature tests. For most of the testing within this work, with the notable exception of the room temperature tensile test, it is expected that there will be almost no difference between the solution annealed tests and the aged tests. With all high temperature testing within this work, prior to the start of tests specimens were held at the test temperature for a three hour soak to ensure temperature stability. Most test temperatures within this work are near the aging temperature, and as such, even the solution annealed specimen will be more similar to the aged state than a true solution annealed state. As such, it is expected that testing within this work will see smaller differences between the solution annealed and aged state than what was observed in the fossil energy program, which already did not see very large differences.

The following sections will cover the results of the Haynes 282 testing within this program. As noted, all tensile and creep testing was performed in the solution annealed state. The fatigue and creep-fatigue testing covered both the solution annealed and aged state, using the one step aging treatment (4 hours hold at 800 °C).

6.1 Tensile

The results of the tensile tests are shown in Figure 6.4 and Table 6.2. The jump in yield stress at 700 °C is expected, and is known to be above the room temperature yield stress of solution annealed Haynes 282 [44]. For comparison, the published [45] average tensile properties are shown in Table 6.3. The values in Table 6.3 are for Haynes 282 that has undergone the single-step age-hardening treatment, and so the differences between the values in Table 6.2 and Table 6.3 are expected, particularly the large differences for the room temperature test. Actual test values from solution annealed [44] and single-step age-hardened treatment [47] Haynes 282 are shown in Table 6.4, and the yield stresses from this data is shown graphically with the measured yield stress from this current work in Figure 6.5. From these results, it is clear that the room temperature tensile properties from this current work matches well with the expected values. However, the high temperature values fall more in line with what is expected for the single-step age-hardening values, likely due to the soak time performed prior to test initiation. It is not clear that the testing performed on the solution annealed state by Viswanathan, et al. [44] underwent the same soaking step, as their goal was to test the true solution annealed state.
Figure 6.4: Tensile tests (stress vs strain) for wrought Haynes 282 in the SA condition.

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>0.2% Yield (Mpa)</th>
<th>UTS (Mpa)</th>
<th>Elongation (%)</th>
<th>Reduction of area (%)</th>
<th>Modulus (Gpa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT</td>
<td>571.9</td>
<td>978.3</td>
<td>57.2</td>
<td>46.6</td>
<td>225</td>
</tr>
<tr>
<td>700</td>
<td>641.4</td>
<td>820.3</td>
<td>21.3</td>
<td>17.7</td>
<td>161</td>
</tr>
<tr>
<td>750</td>
<td>604.7</td>
<td>726.2</td>
<td>13.8</td>
<td>15.1</td>
<td>163</td>
</tr>
<tr>
<td>800</td>
<td>606.4</td>
<td>678.5</td>
<td>21.3</td>
<td>26.2</td>
<td>137</td>
</tr>
<tr>
<td>850</td>
<td>511.3</td>
<td>515.8</td>
<td>43.9</td>
<td>53.2</td>
<td>140</td>
</tr>
</tbody>
</table>

Table 6.2: Tensile properties of SA Haynes 282.

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>0.2% Yield (Mpa)</th>
<th>UTS (Mpa)</th>
<th>Elongation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT</td>
<td>733</td>
<td>1170</td>
<td>34</td>
</tr>
<tr>
<td>100</td>
<td>684</td>
<td>1123</td>
<td>34</td>
</tr>
<tr>
<td>200</td>
<td>655</td>
<td>1095</td>
<td>35</td>
</tr>
<tr>
<td>300</td>
<td>640</td>
<td>1060</td>
<td>35</td>
</tr>
<tr>
<td>400</td>
<td>639</td>
<td>1021</td>
<td>36</td>
</tr>
<tr>
<td>500</td>
<td>628</td>
<td>998</td>
<td>36</td>
</tr>
<tr>
<td>600</td>
<td>626</td>
<td>1002</td>
<td>32</td>
</tr>
<tr>
<td>700</td>
<td>620</td>
<td>952</td>
<td>23</td>
</tr>
<tr>
<td>800</td>
<td>577</td>
<td>745</td>
<td>16</td>
</tr>
<tr>
<td>900</td>
<td>406</td>
<td>480</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 6.3: Average tensile properties of Haynes 282 after the single-step age-hardening treatment, as provided by Haynes International [45].
<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>SA (SA) or Aged</th>
<th>0.2% Yield (Mpa)</th>
<th>UTS (Mpa)</th>
<th>Elongation (%)</th>
<th>Reduction in area (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>SA</td>
<td>476.4</td>
<td>859.1</td>
<td>55.6</td>
<td>47.4</td>
</tr>
<tr>
<td>427</td>
<td>SA</td>
<td>473.0</td>
<td>800.5</td>
<td>60.5</td>
<td>48.3</td>
</tr>
<tr>
<td>593</td>
<td>Aged</td>
<td>628.3</td>
<td>998.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>621</td>
<td>Aged</td>
<td>618.8</td>
<td>1007.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>649</td>
<td>SA</td>
<td>439.9</td>
<td>717.7</td>
<td>43.4</td>
<td>47.7</td>
</tr>
<tr>
<td>649</td>
<td>Aged</td>
<td>616.4</td>
<td>1016.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>677</td>
<td>Aged</td>
<td>618.3</td>
<td>980.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>704</td>
<td>SA</td>
<td>579.8</td>
<td>786.0</td>
<td>29.5</td>
<td>25.2</td>
</tr>
<tr>
<td>704</td>
<td>Aged</td>
<td>620.2</td>
<td>946.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>732</td>
<td>Aged</td>
<td>620.4</td>
<td>892.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>760</td>
<td>SA</td>
<td>475.7</td>
<td>664.0</td>
<td>24</td>
<td>26.3</td>
</tr>
<tr>
<td>760</td>
<td>Aged</td>
<td>620.6</td>
<td>837.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>788</td>
<td>Aged</td>
<td>591.1</td>
<td>773.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>816</td>
<td>SA</td>
<td>552.3</td>
<td>701.2</td>
<td>18.5</td>
<td>20.7</td>
</tr>
<tr>
<td>816</td>
<td>Aged</td>
<td>561.5</td>
<td>709.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>843</td>
<td>Aged</td>
<td>530.9</td>
<td>648.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>871</td>
<td>Aged</td>
<td>499.1</td>
<td>585.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>899</td>
<td>Aged</td>
<td>410.1</td>
<td>482.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>927</td>
<td>Aged</td>
<td>321.0</td>
<td>379.9</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

6.2 Creep

The primary goal for creep testing within this project was to ensure that the plate of Haynes 282 tested within this project fell within the expected values so that the larger creep databases that exist from other work, particularly the ASME BPVC Code Case efforts, could be used. The results of the creep testing are shown in Figure 6.6 and Table 6.5. A comparison with the data from Viswanathan, et al. [44], original shown in Figure 6.3, was combined with our own data and shown in Figure 6.7. From these results, it is apparent that our plate behaves as expected and has similar creep properties as reported by others [44, 47].
Figure 6.6: Creep curves (strain vs. time) for wrought Haynes 282.

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>Stress (MPa)</th>
<th>Rupture time (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>750</td>
<td>320</td>
<td>1424.0</td>
</tr>
<tr>
<td>750</td>
<td>320</td>
<td>1485.0</td>
</tr>
<tr>
<td>800</td>
<td>200</td>
<td>1585.5</td>
</tr>
<tr>
<td>850</td>
<td>100</td>
<td>3639.2</td>
</tr>
</tbody>
</table>

Table 6.5: Summary of creep rupture times.

Figure 6.7: Larson Miller plot for creep test results in this work as well as those from [44].
6.3 Cyclic testing

Cyclic testing, including fatigue and creep-fatigue, was performed on MTS servo-hydraulic test frames, with three zone furnaces capable of heating specimens up to 1,200 °C. Fatigue and creep-fatigue tests were performed in accordance with ASTM E606 (Standard Test Method for Strain-Controlled Fatigue Testing) and ASTM 2714 (Standard Test Method for Creep-Fatigue Testing), respectively. Testing was performed at temperatures of 750 and 850°C. Strain was controlled with strain ranges (\(\Delta \varepsilon\)) of 0.4% and 1% total strain. The tests were fully reversed, so the \(\Delta \varepsilon\) of 1% tests were performed from -0.5% up to a 0.5% strain. Tests were typically stopped prior to specimen fracture to protect the fracture surface and the extensometer. To consistently determine fatigue life, \(N_{25}\) was used, which represents fatigue life as the cycle that is 25% lower in stress from the point where the stress vs cycle plot deviates from linearity (the cycle where this deviation occurs is referred to as \(N_0\)). \(N_0\) is often assumed to be the point where a fatigue crack initiates, which ultimately leads to failure. Creep-fatigue testing involves cyclic testing with a hold at either the peak tensile or peak compressive stress (in this case, peak tensile stress will be used, as it is likely the more deleterious hold condition). This allows for creep/stress relaxation to occur between cycles. Test conditions were chosen to match those used for Alloy 740H in a prior project [46].

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>Strain range</th>
<th>Hold time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>750</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>750</td>
<td>0.6</td>
<td>10</td>
</tr>
<tr>
<td>750</td>
<td>0.6</td>
<td>60</td>
</tr>
<tr>
<td>750</td>
<td>0.6</td>
<td>120</td>
</tr>
<tr>
<td>750</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>750</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>750</td>
<td>1</td>
<td>640</td>
</tr>
<tr>
<td>750</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>850</td>
<td>0.4</td>
<td>0</td>
</tr>
<tr>
<td>850</td>
<td>0.4</td>
<td>10</td>
</tr>
<tr>
<td>850</td>
<td>0.4</td>
<td>60</td>
</tr>
<tr>
<td>850</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>850</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>850</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>850</td>
<td>1</td>
<td>600</td>
</tr>
</tbody>
</table>

Table 6.6: Cyclic test conditions for wrought 282. Due to time restriction, the 600 minute hold creep-fatigue tests were not performed.

Replicate testing was performed at several conditions. The scope of the Haynes 282 testing was small compared to the large test matrix needed for formally creating design models through ASME, however, the results can be used to create a preliminary creep-fatigue interaction assessment to understand how the material likely behaves and what the design models may look like.
In addition to plate form, cast Haynes 282 was received from Oak Ridge National Laboratory (ORNL) for testing. This was not the primary task of this work, and so testing was limited, with only a single fatigue test ultimately being completed. The cast material was heat treated (two-step aging treatment) and machined at ORNL. As the cast material is not thick enough for the standard fatigue/creep-fatigue specimens, a modification to the design of the grips was made, shown in Figure 6.8. This is not expected to negatively affect the testing of the material at all.

![Modified button head grip](image)

**Figure 6.8:** Modification to the grips of the fatigue/creep-fatigue specimens (Figure 6.2) to account for the thinner cast material.

A summary of the data collected from all cyclic testing is shown in Table 6.7. As of the writing of this report, three tests have not been completed, but are going to be continued after the end of this project. They are highlighted yellow in Table 6.7, and a revision to this report will be issued upon completion of all testing.

<table>
<thead>
<tr>
<th>Temp (°C)</th>
<th>Strain Rate (%)</th>
<th>t₀ (min)</th>
<th>Heat Treatment</th>
<th>Δε₀ (%)</th>
<th>σ₀, Start (MPa)</th>
<th>σ₀, Midlife (MPa)</th>
<th>σ₀, Failure (MPa)</th>
<th>Cycles to Initiation</th>
<th>Cycles to Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>750</td>
<td>0.001</td>
<td>0</td>
<td>SA</td>
<td>0.6</td>
<td>450</td>
<td>443</td>
<td>501</td>
<td>5015</td>
<td>5608</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>0</td>
<td>Aged</td>
<td>0.6</td>
<td>460</td>
<td>409</td>
<td>452</td>
<td>59015</td>
<td>59148</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>10</td>
<td>SA</td>
<td>0.6</td>
<td>434</td>
<td>412</td>
<td>416</td>
<td>42922</td>
<td>43377</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>60</td>
<td>SA</td>
<td>0.6</td>
<td>412</td>
<td>361</td>
<td>313</td>
<td>4335</td>
<td>4494</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>0</td>
<td>Aged</td>
<td>0.6</td>
<td>453</td>
<td>483</td>
<td>500</td>
<td>770</td>
<td>1089</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>0</td>
<td>SA</td>
<td>1.0</td>
<td>506</td>
<td>464</td>
<td>487</td>
<td>782</td>
<td>1411</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>0</td>
<td>Cast-Aged</td>
<td>1.0</td>
<td>535</td>
<td>514</td>
<td>502</td>
<td>424</td>
<td>674</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>0</td>
<td>Aged</td>
<td>1.0</td>
<td>604</td>
<td>493</td>
<td>493</td>
<td>725</td>
<td>1084</td>
</tr>
<tr>
<td>750</td>
<td>0.001</td>
<td>60</td>
<td>SA</td>
<td>1.0</td>
<td>582</td>
<td>501</td>
<td>693</td>
<td>501</td>
<td>216</td>
</tr>
<tr>
<td>850</td>
<td>0.001</td>
<td>0</td>
<td>Aged</td>
<td>0.4</td>
<td>298</td>
<td>313</td>
<td>313</td>
<td>64594</td>
<td>65788</td>
</tr>
<tr>
<td>850</td>
<td>0.001</td>
<td>10</td>
<td>Aged</td>
<td>0.4</td>
<td>223</td>
<td>-</td>
<td>-</td>
<td>77</td>
<td>3913</td>
</tr>
<tr>
<td>850</td>
<td>0.001</td>
<td>60</td>
<td>Aged</td>
<td>0.4</td>
<td>223</td>
<td>194</td>
<td>188</td>
<td>77</td>
<td>3913</td>
</tr>
</tbody>
</table>

Table 6.7: Final fatigue and creep-fatigue results from this Haynes 282 test program. With the exception of the single test labeled Cast-Aged, all testing was performed on wrought plate. Yellow rows indicate tests are on-going at the time of writing this report.
Detailed results of the cyclic testing can be found in Figure 6.9 to Figure 6.14. Much of this data is used to create the creep-fatigue interaction diagram. There were significant issues with two tests in particular, which resulted in the recommendation that they are not taken into consideration for the design models. The SA fatigue test at 750 °C, Δε 0.6% failed unusually early. It is believed there must have been a defect, such as a machining defect, that caused an early crack initiation. Figure 6.9 shows that the peak stresses were very similar to the two aged fatigue tests at the same condition, the crack just initiated very early (roughly the same cycle count as the SA creep-fatigue tests). The sample was examined under an optical microscope to look for obvious flaws, but nothing was observed at that level. Detailed fractography is outside the scope of this work. The second test of concern is the first of the SA 850 °C, Δε 1.0%. This test was interrupted due to a power outage with the frame, and upon restarting, the peak stresses were significantly higher than expected, suggesting that at the unexpected stop, the specimen likely was loaded in an uncontrolled manner.

Figure 6.9: Stress vs. cycles for all Δε 0.6%, 750 °C tests.
Figure 6.10: Stress vs. cycles for all $\Delta \varepsilon$ 1.0%, 750 °C tests.

Figure 6.11: Hysteresis loops for all 750 °C creep-fatigue tests.
Figure 6.12: Stress vs. cycles for all Δε 0.4%, 850 °C tests.

Figure 6.13: Stress vs. cycles for all Δε 1.0%, 850 °C tests.
In comparison of the aged vs. SA, the most significant differences are expected for the 750 °C tests. At 850 °C, there is not a significant difference expected in testing the aged vs. SA, as the cyclic specimens are always held at temperature for 3 hours prior to testing to ensure uniform temperatures, and the 850 °C hold is expected to result in a similar microstructure whether the initial condition was SA or aged. Duplicate tests performed on the material in the aged and SA conditions allow for a direct comparison. In particular, the fatigue tests at 750 °C, Δε 1% and the long-hold creep-fatigue tests at 750 °C, Δε 0.6% show good comparisons. There are differences apparent in the initial portion of the fatigue test (approximately the first 300 cycles) where the peak stresses are higher for the aged material, before settling into the same peak stresses beyond that point. This is likely due to the precipitate structure in the aged specimen causing it to be harder than the solution annealed, but it is expected that a similar microstructure exists beyond the initial portion of the test, resulting in a similar looking curve beyond that point. For the creep-fatigue tests (60 min. hold SA vs. 120 min. hold Aged), the curves look very similar (except that the 120 min. hold test has not completed). It is not uncommon for long hold time tests to be similar, as the stress relaxation has reached a point where additional creep does not occur to a significant degree due to the low stresses that were reached. Additional testing on 850 °C aged material is expected to show that there is no significant differences for this test temperature.
7 Conclusions

This report describes the development of srlife, an open-source python module, for structural service life estimation of high temperature solar receivers. Given the time-dependent thermal and mechanical boundary conditions, representing representative conditions on one or more thermal days, applied to the individual tubes, the package estimates the creep-fatigue service life of the receiver, providing this estimate as a number of repetitions of the user-provided daily cycle(s). In addition, the user can output the full thermal, structural, and damage results to VTK files which can be visualized and postprocessed using software like ParaView. srlife can be embedded in a python-driven software stacks, including incident solar flux and thermohydraulic simulations and levelized cost analysis, for receiver design optimizations.

The report discusses the construction of the required material data included in srlife material database for six metallic alloys – 316H, 800H, Alloy 617, Alloy 740H, Alloy 282, and Alloy 230. For first four alloys the material data are well supported through large DOE:NE and DOE:EERE database. For Alloy 282, the available creep-fatigue test data, collected as part of this project, is comparatively less but sufficient enough for designing CSP components. No creep-fatigue data is found in literature for Alloy 230. The tool currently uses the Alloy 617 creep-fatigue damage envelop for this and therefore Alloy 230 data in srlife should not be used in designing actual CSP components. The tool provides three options for modelling the constitutive response of the material – elastic, elastic-creep, and base (full inelastic model accounting for both time-independent plasticity and creep). The elastic model provides the most conservative estimation of service life, followed by elastic-creep and base models.

The report assesses several heuristics for fast estimation of receiver structural life using srlife. The heuristics include reducing the number of tubes explicitly represented in the analysis, reducing the full 3D analysis to either 2D or 1D, and specifying the location in the tube for 2D or 1D analysis. The assessment of the heuristics for two reference receiver designs shows that the analysis time can be reduced by several orders of magnitude while retaining adequate accuracy by considering only the hottest and coldest tubes in a panel and analyzing the 2D slices of the tubes at the maximum metal temperature location.

Two additional heuristics are also discussed – 1) daily resetting the tube temperature to avoid simulating the nightly isothermal hold and 2) different approaches for extrapolating creep-fatigue damage to avoid simulation for the whole service life.

The report provides several examples demonstrating the use of srlife in simplified system analysis for a complete receiver design, receiver structural material selection, and extending receiver life by changing a few panels that fail earlier than other panels.
Acknowledgements

This work was sponsored by the U.S. Department of Energy, under Contract No. DE-AC02-06CH11357 with Argonne National Laboratory, managed and operated by UChicago Argonne LLC. Funding was provided by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Solar Energy Technologies Office, Concentrating Solar Power Program, under award #36489.

Discussions with William Logie, Joseph Coventry, and Charles-Alexis Asselineau of The Australian National University are acknowledged.
Bibliography


[19] Alloy 282 Code Case data from ORNL, provided by DOE.


A \textit{srlife} Documentation
srlife is a python package aimed at assessing the expected life of high temperature metallic solar receivers and other components. The An overview of srlife section describes the package objectives, organizations, and assumptions, the Installing srlife section provides installation instructions, and Tutorial: how to use the package provides a tutorial on how to run the package for a sample analysis.
srlife is a package for estimating the life of high temperature solar receivers. The package currently focuses on metallic, tubular, panel receivers somewhat skewed towards new designs using molten chloride salt as the working fluid, though the package includes data for evaluating different types of working fluids and receiver materials.

The package provides a complete assessment of a receiver design starting from the thermal and mechanical boundary conditions applied to the individual tubes in the receiver. This means that the package requires input from additional simulations in order to generate the boundary conditions. Specifically, srlife would usually sit on top of a simulation of the heliostat to calculate the incident solar flux, simulations or measurements of the effective absorption of the receiver tubes, and system-level and detailed thermohydraulic simulations to determine the local thermomechanical boundary conditions on each tube.

Given this information, presented as time-dependent boundary conditions representing representative conditions on one or more thermal days, the package estimates the structural life of the receiver, providing this estimate as a number of repetitions of the user-provided daily cycle(s).

The package includes material information for a variety of receiver structural materials and working fluids. The user can add additional material models by manipulating a fixed XML file format – they do not need to alter the package source code to add new materials or provide variant material models for the materials already included in the base release.

srlife provides modules to:

1. Define the receiver geometry and topology – how panels are connected to each other and how tubes are connected within a panel.

2. **Provide thermomechanical boundary conditions, specifically options for:**
   a. Inner or outer diameter incident heat flux
   b. Inner or outer diameter fixed temperature
   c. Inner or outer diameter convective heat transfer
   d. Inner pressure

3. Finite difference, transient heat transfer solvers to convert the thermal boundary conditions into the receiver tube temperature fields.

4. A full-scale finite element solver to take the tube temperatures and mechanical boundary conditions to the tube stress/strain/displacement fields.

5. Connections to a extensive nonlinear material model library neml to provide accurate inelastic constitutive models.

6. A receiver system solver that can account for connections between tubes in a panel and panels in a receiver, to accurately model the effect of structural connections with an abstract, numerically inexpensive representation.

7. Damage solvers to estimate the level of creep-fatigue damage in a tube given the structural and thermal results.
8. An extensive material property library covering common high temperature metallic receiver materials.

1.1 Conventions

With user-defined material models the end-user can apply any unit system they want. However, the built-in material library uses the following unit conventions:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>mm</td>
</tr>
<tr>
<td>angle</td>
<td>radians</td>
</tr>
<tr>
<td>stress</td>
<td>MPa</td>
</tr>
<tr>
<td>time</td>
<td>hr</td>
</tr>
<tr>
<td>strain</td>
<td>mm/mm</td>
</tr>
<tr>
<td>temperature</td>
<td>K</td>
</tr>
<tr>
<td>flux</td>
<td>W/mm$^2$</td>
</tr>
<tr>
<td>conductivity</td>
<td>W/mm$^2$-K</td>
</tr>
<tr>
<td>diffusivity</td>
<td>mm$^2$/hr</td>
</tr>
<tr>
<td>film coefficient</td>
<td>W/mm$^2$-K</td>
</tr>
</tbody>
</table>

**Warning:** The built-in material model library aims for *average* life estimation, approximating the average time to failure for a particular component. This type of estimation is not suitable for a full design calculation, where lower-bound properties with adequate design margin must be used.
srlife is available in the pypi package repository and can be installed with pip. srlife uses python3 and requires several additional python packages, all of which are available pypi.
srlife is compatible with python3 only, specifically python versions 3.6, 3.7, 3.8, and 3.9

### 2.1 Install using the pip package manager

The easiest way to install the package is to use the pip package manager, installing srlife from pypi automatically.

#### 2.1.1 Ubuntu Linux 18.04

```bash
pip install srlife
```

#### 2.1.2 MacOS Sierra 10.14 Mojave

It is easiest to install srlife using a homebrew version of python, not the default system python.
Go to [brew.sh](https://brew.sh) and follow the directions to install homebrew.

Open up a terminal and run:

```bash
brew install python
pip3 install srlife
```

`srlife` will then be available as a package through the homebrew version of python (often available as `python3` instead of `python`).

### 2.2 Install from the github repository directly

If you want to use the current development version of srlife or if you want to also obtain the tutorial, example, and test files you can install the package directly from [github](https://github). In addition to the, cmake, BLAS, and LAPACK requirements you will need git and, optionally, the nose package to automatically run the tests.
2.2.1 Ubuntu Linux 18.04

The following installs the prerequisites, downloads srlife, sets up the python package, and runs the automated test suite.

```bash
sudo apt install build-essential cmake libblas-dev liblapack-dev python3-dev python3-setuptools python3-pip python3-nose
git clone https://github.com/Argonne-National-Laboratory/srlife.git
cd srlife
pip3 install --user wheel
pip3 install --user -r requirements.txt
nositests3
```

Note the package is installed wherever the user executed the `git clone` command. Using the package outside this directory requires adding it to the `PYTHONPATH` environment variable.
CHAPTER THREE

TUTORIAL: HOW TO USE THE PACKAGE

This tutorial guides you through setting up a very simple receiver model using the python interface, saving the model to an HDF5 file for archiving and as an example of how to use the file interface to the package, setting up an analysis, running the analysis, and examining detailed results. The tutorial assumes you have installed srlife, have a working text editor, and can run python programs from the command line. The example commands here are for Ubuntu Linux, but the step-by-step directions are similar for any operating system.

The files produced by this tutorial are available in the full srlife source package, which can be obtained via git:

```
git clone https://github.com/Argonne-National-Laboratory/srlife.git
cd srlife
```

The files are in the `examples/tutorial` directory. Obtaining the files is not necessary as this tutorial will walk you through creating the files and output data available in that source directory.

In the following sections python code can either be entered directly into the python interpreter (recall srlife uses `python3`) or entered into a script and run to see the results. The example assumes the user creates two python scripts:

1. `setup_problem.py` to build the model and write it to an HDF5 file.
2. `run_problem.py` to actually run the analysis.

### 3.1 Receiver geometry, loading, and materials

The following image describes the very simple receiver used in this example:
The receiver has two panels each with two tubes. This is (of course) not a realistic configuration, but setup to ensure the life estimation only takes a short period of time, even for a full 3D analysis.

Tube are all geometrically the same with an outer radius of \( r_o = 12.7 \) mm, a thickness of \( s = 1.0 \) mm, and a height of \( h = 5000 \) mm. The problem applies a fairly coarse discretization of 12 radial divisions, 20 circumferential divisions, and 10 axial divisions. Each structural finite element problem then has around 6000 degrees of freedom, which is not a trivially small amount for full incremental 3D analysis.

The two tubes in each individual panel are rigidly connected through their top-surface displacements. The two panels are completely structurally disconnected.

The thermal boundary conditions are an incident flux on the tube outer diameter and convective heat transfer on the inner diameter. The tubes also experience a time varying inner pressure.

The analysis encompasses a single, representative day of 24 hours. The boundary conditions are built up from simpler functions mathematical functions. In a real analysis this information would come from other simulation software, rather than an exact mathematical description.
The boundary conditions use the function

\[ A(t) = \sin\left(\frac{\pi t}{12}\right) \]

\[ O(t) = \frac{1 + \text{sign}(A(t))}{2} A(t) \]

to describe when the receiver is operating (for 12 hours) or in a standby condition overnight. This function looks like this

which describes a gradual ramp up to and down from the peak solar conditions followed by 12 hours at some standby condition.

The incident flux on the tube outside diameter is then described by the composite function

\[ h_{\theta}(\theta) = \cos \theta \]

\[ h_z(z) = \frac{1 + \sin(\pi z/h)}{2} \]

\[ h(t, \theta, z) = h_{\text{max}} O(t) h_{\theta}(\theta) h_z(z) h_{\text{tube}} \]

This example uses \( h_{\text{max}} = 0.6 \text{ W/mm}^2 \) and \( h_0 = 1.0, h_1 = 0.8, h_2 = 0.6, h_3 = 0.4 \) for each of the four tubes. The figure below plots the \( \theta = [-\pi/2, \pi/2] \) interval of the incident flux for tube 0 at the peak flux (\( t = 6 \text{ hours} \)).
For the internal convective heat transfer the four tubes all have the same fluid temperature distribution, given by

\[ T_{\text{fluid}}(t, z) = \Delta T O(t) \frac{z}{h} + T_{\text{start}} \]

with \( \Delta T = 50 \text{ K} \) and \( T_{\text{start}} = 823 \text{ K} \) in this example. The plot below shows the fluid temperature gradient in each tube at several different times throughout the daily cycle:
Finally, the internal pressure in all four tubes is the same and given by

\[ p(t) = p_{\text{max}} \Omega(t) \]

with \( p_{\text{max}} = 1 \) MPa.

In the example the tube material is 316H stainless steel and the working fluid is molten chloride salt.

### 3.2 Defining the receiver geometry and loading conditions

The tutorial assumes the following python code is entered into a script called `setup_problem.py`.

The first requirement is to import the required modules from the srlife package along with numpy:

```python
import numpy as np
from srlife import receiver
```

The first step is to setup the `srlife.receiver.Receiver` object which contains all the information about the receiver loading, geometry, and analysis cycle. This information can be written to an HDF5 file for archiving and then read back in to complete an analysis. The receiver geometry and loading is independent of the tube material, which can then be easily changed to explore different receiver materials.

First define some basic information about the receiver loading cycle, setup currently-empty receiver and panel objects to receive the individual tube information, and define the structural connections between panels in the receiver and tubes in a panel:
# Setup the base receiver
period = 24.0  # Loading cycle period, hours
days = 1  # Number of cycles represented in the problem
panel_stiffness = "disconnect"  # Panels are disconnected from one another

model = receiver.Receiver(period, days, panel_stiffness)

# Setup each of the two panels
tube_stiffness = "rigid"
panel_0 = receiver.Panel(tube_stiffness)
panel_1 = receiver.Panel(tube_stiffness)

The following code then defines the variables needed to specify the receiver geometry and discretization as well as
the Python implementation of the mathematical functions describing the receiver boundary conditions, defined in the
previous section

# Basic receiver geometry
r_outer = 12.7  # mm
thickness = 1.0  # mm
height = 5000.0  # mm

# Tube discretization
nr = 12
nt = 20
nz = 10

# Mathematical definition of the tube boundary conditions
# Function used to define daily operating cycle
onoff_base = lambda t: np.sin(np.pi*t/12.0)
onoff = lambda t: (1+np.sign(onoff_base(t)))/2 * onoff_base(t)

# Max flux
h_max = 0.6  # W/mm² (which is also MW/m²)
# Flux circumferential component
h_circ = lambda theta: np.cos(theta)
# Flux axial component
h_axial = lambda z: (1+np.sin(np.pi*z/height))/2
# Total flux function
h_flux = lambda time, theta, z: onoff(time) * h_max * h_circ(theta) * h_axial(z)

# Flux multipliers for each tube
h_tube_0 = 1.0
h_tube_1 = 0.8
h_tube_2 = 0.6
h_tube_3 = 0.4

# ID fluid temperature histories for each tube
delta_T = 50  # K
T_base = 550 + 273.15  # K
fluid_temp = lambda t, z: delta_T * onoff(t) * z/height + T_base

# ID pressure history
p_max = 1.0  # MPa
pressure = lambda t: p_max * onoff(t)
srlife takes boundary condition information at *discrete* time intervals. This allows the software to interface with system thermohydraulic codes that may produce discrete temperature and pressure information. In this example we’ll provide the boundary condition information at uniformly-spaced times throughout the 24 hour cycle:

```python
# Time increments throughout the 24 hour day
times = np.linspace(0, 24, 24*2+1)
```

Similarly, the spatial information about the flux and convective boundary conditions must be defined over discrete grid points in cylindrical coordinates. srlife uses the "ij" indexing scheme defined in *numpy*, where the individual coordinate arrays are indexed with a matrix scheme:

```python
# Various meshes needed to define the boundary conditions
# 1) A mesh over the times and height (for the fluid temperatures)
time_h, z_h = np.meshgrid(times, np.linspace(0, height, nz), indexing='ij')
# 2) A surface mesh over the outer surface (for the flux)
time_s, theta_s, z_s = np.meshgrid(times, np.linspace(0, 2*np.pi, nt+1)[:nt], np.linspace(0, height, nz), indexing = 'ij')
```

Each individual tube can be defined in terms of its geometry, discretization, and boundary conditions. The first tube is defined like this:

```python
# Setup each tube in turn and assign it to the correct panel
# Tube 0
tube_0 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_0.set_times(times)
tube_0.set_bc(receiver.ConvectiveBC(r_outer-thickness, height, nt, times, fluid_temp(time_h,z_h)), "inner")
tube_0.set_bc(receiver.HeatFluxBC(r_outer, height, nt, nz, times, h_flux(time_s, theta_s, z_s) * h_tube_0), "outer")
tube_0.set_pressure_bc(receiver.PressureBC(times, pressure(times)))
```

The remainder of the tubes are defined similarly:

```python
# Tube 1
tube_1 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_1.set_times(times)
tube_1.set_bc(receiver.ConvectiveBC(r_outer-thickness, height, nz, times, fluid_temp(time_h,z_h)), "inner")
tube_1.set_bc(receiver.HeatFluxBC(r_outer, height, nt, nz, times, h_flux(time_s, theta_s, z_s) * h_tube_1), "outer")
tube_1.set_pressure_bc(receiver.PressureBC(times, pressure(times)))

# Tube 2
tube_2 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_2.set_times(times)
tube_2.set_bc(receiver.ConvectiveBC(r_outer-thickness, height, nz, times, fluid_temp(time_h,z_h)), "inner")
tube_2.set_bc(receiver.HeatFluxBC(r_outer, height, nt, nz, times, h_flux(time_s, theta_s, z_s) * h_tube_2), "outer")
tube_2.set_pressure_bc(receiver.PressureBC(times, pressure(times)))

# Tube 3
tube_3 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_3.set_times(times)
```

(continues on next page)
Finally, each tube must be added to the relevant panel and the panels to the receiver:

```python
# Assign to panel 0
panel_0.add_tube(tube_0, "tube0")
panel_0.add_tube(tube_1, "tube1")

# Assign to panel 1
panel_1.add_tube(tube_2, "tube2")
panel_1.add_tube(tube_3, "tube3")

# Assign the panels to the receiver
model.add_panel(panel_0, "panel0")
model.add_panel(panel_1, "panel1")
```

At this point the `srlife.receiver.Receiver` object is fully-defined and ready to be used in a life assessment. However, for this tutorial instead save the receiver to disk for later use:

```python
# Save the receiver to an HDF5 file
model.save("model.hdf5")
```

Assuming you save this script as `setup_problem.py` running it with

```
python setup_problem.py
```

will produce a file called `model.hdf5` which saves all the information described above for later use.

The key point of this rather lengthy script is that actual users of the srlife should never have to write a script like this explicitly defining the receiver boundary conditions. Instead this information should be obtained from some upstream thermohydraulic and (ultimately) heliostat analysis system. The user would then write interface code either directly in Python or using the HDF5 file format as an intermediary to transfer the information into srlife.

### 3.3 Defining the analysis material models and analysis parameters

The remainder of the information needed to estimate the life of the receiver defined above is:

1. What material the receiver is constructed from to define suitable thermal, structural, and damage models.
2. The working fluid properties to simulate convective heat transfer.
3. How to solve the thermal, structural, and damage analysis simulations required to estimate the life of the component.

The following assumes the user makes a new Python script called `run_problem.py` to define this additional information and run the analysis.

First load the required modules from srlife and, again, the numpy library to help with some mathematics:
import numpy as np
from srlife import receiver, solverparams, library, thermal, structural, system, damage, managers

The script must first load the receiver, defined in the previous section, from the HDF5 file for reuse:

```python
# Load the receiver we previously saved
model = receiver.Receiver.load("model.hdf5")
```

srlife maintains a library of material models for several metallic receivers and working fluids. Standard models can be loaded from the this library for use in the analysis:

```python
# Choose the material models
fluid_mat = library.load_fluid("salt", "base") # Generic chloride salt model
# Base 316H thermal and damage models, a simplified deformation model to
# cut down on the run time of the 3D analysis
thermal_mat, deformation_mat, damage_mat = library.load_material("316H", "base",
"elastic_creep", "base")
```

This example uses the “base” representations of the chloride salt and 316H models, except for a simplified structural model to help cut down on analysis time.

The user must then tell srlife how to solve the problem. This might involve some simplification of the full 3D problem and requires defining thermal, single-tube structural, system structural, and damage solvers as well as a set of solution parameters.

The tutorial reduces the analysis (thermal and structural) to one dimension to make the resulting thermal/structural analysis run essentially instantaneously:

```python
# Cut down on run time for now by making the tube analyses 1D
# This is not recommended for actual design evaluation
for panel in model.panels.values():
    for tube in panel.tubes.values():
        tube.make_1D(tube.h/2, 0)
```

For now, srlife only provides a single solver of each type and all the solution parameters have sensible default values. However, the following code could be changed to use a custom thermal, structural, or damage solver or to change how the module solves the thermal and structural subproblems:

```python
# Setup some solver parameters
params = solverparams.ParameterSet()
params['progress_bars'] = True # Print a progress bar to the screen as we solve
params['nthreads'] = 1 # Solve will run in multithreaded mode, set to number of available cores
params['system']['atol'] = 1.0e-4 # During the standby very little happens, lower the atol to accept this result

# Choose the solvers, i.e. how we are going to solve the thermal, # single tube, structural system, and damage calculation problems. # Right now there is only one option for each
thermal_solver = thermal.FiniteDifferenceImplicitThermalSolver(    params["thermal"])
# Define the structural solver to use in solving the individual tube problems
```
structural_solver = structural.PythonTubeSolver(params["structural"])  
# Define the system solver to use in solving the coupled structural system  
system_solver = system.SpringSystemSolver(params["system"])  
# Damage model to use in calculating life  
damage_model = damage.TimeFractionInteractionDamage(params["damage"])  

The user might consider changing the params[‘nthreads’] parameter to match the number of cores on their machine, to speed up the analysis.

### 3.4 Running the life estimation analysis

With the problem fully defined a solution manager can be setup to manage the thermal, structural, and damage solves and to complete the life estimation. Finally, the analysis can be run and the life of the receiver estimated:

```python
# The solution manager
solver = managers.SolutionManager(model, thermal_solver, thermal_mat, fluid_mat,  
structural_solver, deformation_mat, damage_mat, system_solver, damage_model, pset =  
params)

# Actually solve for life
life = solver.solve_life()
print("Best estimate life: %f daily cycles" % life)
```

This solution will take a long time (up to an hour on some machines) to complete because it is using a full 3D analysis of each tube. Running the problem with multiple threads will help decrease the required solution time.

If the params[‘progress_bars’] parameter is kept as True then the program will print a status bar representing its progress along each individual step (thermal, structural, and damage). Finally, the program should output:

```
Best estimate life: 9062.849331 daily cycles
```

indicating that the module predicts this receiver to have a structural life of 9062 repetitions of the daily cycle, or about 25 years.

### 3.5 Visualizing tube results

Optionally, the user can output the full, temporal and spatial tube results to a VTK file for additional postprocessing:

```python
# Save the tube data out for additional visualization
for pi, panel in model.panels.items():  
    for ti, tube in panel.tubes.items():  
        tube.write_vtk("tube-%s-%s" % (pi, ti))
```

This command produces a series of VTK files (one per tube per time step) containing the full thermal, structural, and damage results. These files can be visualized with a program like ParaView.
### 3.6 Complete example scripts

*setup_problem.py*

```python
import numpy as np

from srlife import receiver

# Setup the base receiver
period = 24.0  # Loading cycle period, hours
days = 1  # Number of cycles represented in the problem
panel_stiffness = "disconnect"  # Panels are disconnected from one another

model = receiver.Receiver(period, days, panel_stiffness)

# Setup each of the two panels
tube_stiffness = "rigid"
panel_0 = receiver.Panel(tube_stiffness)
panel_1 = receiver.Panel(tube_stiffness)

# Basic receiver geometry
r_outer = 12.7  # mm
thickness = 1.0  # mm
height = 5000.0  # mm

# Tube discretization
nr = 12
nt = 20
nz = 10

# Mathematical definition of the tube boundary conditions
# Function used to define daily operating cycle
onoff_base = lambda t: np.sin(np.pi*t/12.0)
onoff = lambda t: (1+np.sign(onoff_base(t)))/2 * onoff_base(t)

# Max flux
h_max = 0.6  # W/mm^2 (which is also MW/m^2)
# Flux circumferential component
h_circ = lambda theta: np.cos(theta)
# Flux axial component
h_axial = lambda z: (1+np.sin(np.pi*z/height))/2
# Total flux function
h_flux = lambda time, theta, z: onoff(time) * h_max * h_circ(theta) * h_axial(z)

# Flux multipliers for each tube
h_tube_0 = 1.0
h_tube_1 = 0.8
h_tube_2 = 0.6
h_tube_3 = 0.4

# ID fluid temperature histories for each tube
delta_T = 50  # K
T_base = 550 + 273.15  # K
fluid_temp = lambda t, z: delta_T * onoff(t) * z/height + T_base
```

(continues on next page)
# ID pressure history

```python
p_max = 1.0  # MPa
pressure = lambda t: p_max * onoff(t)
```

# Time increments throughout the 24 hour day

```python
times = np.linspace(0, 24, 24*2+1)
```

# Various meshes needed to define the boundary conditions

1) A mesh over the times and height (for the fluid temperatures)

```python
time_h, z_h = np.meshgrid(times, np.linspace(0, height, nz), indexing='ij')
```

2) A surface mesh over the outer surface (for the flux)

```python
time_s, theta_s, z_s = np.meshgrid(times, np.linspace(0, 2*np.pi, nt+1)[:nt],
                                   np.linspace(0, height, nz), indexing='ij')
```

# Setup each tube in turn and assign it to the correct panel

# Tube 0

```python
tube_0 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_0.set_times(times)
tube_0.set_bc(receiver.ConvectiveBC(r_outer-thickness,
                                      height, nz, times, fluid_temp(time_h,z_h)), "inner")
tube_0.set_bc(receiver.HeatFluxBC(r_outer, height,
                                   nt, nz, times, h_flux(time_s, theta_s, z_s) * h_tube_0), "outer")
tube_0.set_pressure_bc(receiver.PressureBC(times, pressure(times)))
```

# Tube 1

```python
tube_1 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_1.set_times(times)
tube_1.set_bc(receiver.ConvectiveBC(r_outer-thickness,
                                      height, nz, times, fluid_temp(time_h,z_h)), "inner")
tube_1.set_bc(receiver.HeatFluxBC(r_outer, height,
                                   nt, nz, times, h_flux(time_s, theta_s, z_s) * h_tube_1), "outer")
tube_1.set_pressure_bc(receiver.PressureBC(times, pressure(times)))
```

# Tube 2

```python
tube_2 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_2.set_times(times)
tube_2.set_bc(receiver.ConvectiveBC(r_outer-thickness,
                                      height, nz, times, fluid_temp(time_h,z_h)), "inner")
tube_2.set_bc(receiver.HeatFluxBC(r_outer, height,
                                   nt, nz, times, h_flux(time_s, theta_s, z_s) * h_tube_2), "outer")
tube_2.set_pressure_bc(receiver.PressureBC(times, pressure(times)))
```

# Tube 3

```python
tube_3 = receiver.Tube(r_outer, thickness, height, nr, nt, nz, T0 = T_base)
tube_3.set_times(times)
tube_3.set_bc(receiver.ConvectiveBC(r_outer-thickness,
                                      height, nz, times, fluid_temp(time_h,z_h)), "inner")
tube_3.set_bc(receiver.HeatFluxBC(r_outer, height,
                                   nt, nz, times, h_flux(time_s, theta_s, z_s) * h_tube_3), "outer")
tube_3.set_pressure_bc(receiver.PressureBC(times, pressure(times)))
```
# Assign to panel 0
panel_0.add_tube(tube_0, "tube0")
panel_0.add_tube(tube_1, "tube1")

# Assign to panel 1
panel_1.add_tube(tube_2, "tube2")
panel_1.add_tube(tube_3, "tube3")

# Assign the panels to the receiver
model.add_panel(panel_0, "panel0")
model.add_panel(panel_1, "panel1")

# Save the receiver to an HDF5 file
model.save("model.hdf5")

run_problem.py

```python
import numpy as np
from srlife import receiver, solverparams, library, thermal, structural, system, damage, managers

# Load the receiver we previously saved
model = receiver.Receiver.load("model.hdf5")

# Choose the material models
fluid_mat = library.load_fluid("salt", "base")  # Generic chloride salt model
# Base 316H thermal and damage models, a simplified deformation model to
# cut down on the run time of the 3D analysis
thermal_mat, deformation_mat, damage_mat = library.load_material("316H", "base", "base", "base")

# Cut down on run time for now by making the tube analyses 1D
# This is not recommended for actual design evaluation
for panel in model.panels.values:
    for tube in panel.tubes.values():
        tube.make_1D(tube.h/2, 0)

# Setup some solver parameters
params = solverparams.ParameterSet()
params['progress_bars'] = True  # Print a progress bar to the screen as we solve
params['nthreads'] = 4  # Solve will run in multithreaded mode, set to number of available cores
params['system']['atol'] = 1.0e-4  # During the standby very little happens, lower the atol to accept this result

# Choose the solvers, i.e. how we are going to solve the thermal, single tube, structural system, and damage calculation problems.
# Right now there is only one option for each
# Define the thermal solver to use in solving the heat transfer problem
thermal_solver = thermal.FiniteDifferenceImplicitThermalSolver(params['thermal'])
```

(continues on next page)
# Define the structural solver to use in solving the individual tube problems
structural_solver = structural.PythonTubeSolver(params["structural"])

# Define the system solver to use in solving the coupled structural system
system_solver = system.SpringSystemSolver(params["system"])

# Damage model to use in calculating life
damage_model = damage.TimeFractionInteractionDamage(params["damage"])

# The solution manager
solver = managers.SolutionManager(model, thermal_solver, thermal_mat, fluid_mat,
structural_solver, deformation_mat, damage_mat,
system_solver, damage_model, pset = params)

# Actually solve for life
life = solver.solve_life()
print("Best estimate life: %f daily cycles" % life)

# Save the tube data out for additional visualization
for pi, panel in model.panels.items():
    for ti, tube in panel.tubes.items():
        tube.write_vtk("tube-%s-%s" % (pi, ti))
MANAGERS: RUNNING AN ANALYSIS

The `srlife.managers.SolutionManager` class manages an analysis, taking the basic input information:

1. The fully-populated `srlife.receiver.Receiver` class.
2. The thermal, fluid, deformation, and damage material models.
3. The thermal, structural, and system solvers.
4. Optionally, a `srlife.solverparams.ParameterSet` class defining solution parameters, including the number of parallel threads to use in the analysis and providing the estimated life of the receiver as a number of repetitions of the daily cycle.

Once the manager class is constructed the user only needs to call the

```python
life = manager.solve_life()
```

method, which completes the full analysis and returns the estimated life in terms of the number of expected single-day repetitions. The calculation scales the results appropriately given the number of explicitly-defined `days` provided to the `srlife.receiver.Receiver`.

### 4.1 SolutionManager description

The `srlife.managers.SolutionManager` is a wrapper around several internal srlife subclasses. Specifically, the manager handles the process of:

1. Solving for the tube temperatures given the thermal boundary conditions.
2. Solving for the stress/strain deformation response of each tube. Depending on the interconnect stiffnesses provided by the user these tube problems may be coupled in a 1D sense through the tube top-surface displacements.
3. Using the temperature and stress/strain information to solve for the damage in each tube.
4. Finding the worst-case tube and calculating the estimated life.

```python
class srlife.managers.SolutionManager(receiver, thermal_solver, thermal_material, fluid_material, structural_solver, deformation_material, damage_material, system_solver, damage_model, pset={})
```

Solution manager

High level solution manager walking through the thermal, structural, and damage calculations.

**Parameters**

- `receiver (receiver.Receiver)` – receiver object to solve
- `thermal_solver (thermal.ThermalSolver)` – how to solve the heat transport
• **thermal_material** *(materials.ThermalMaterial)* – solid thermal properties

• **fluid_material** *(materials.FluidMaterial)* – fluid thermal properties

• **structural_solver** *(structural.TubeSolver)* – how to solve the mechanics problem

• **deformation_material** *(materials.DeformationMaterial)* – how things deform with time

• **damage_material** *(materials.StructuralMaterial)* – how to calculate creep damage

• **system_solver** *(system.SystemSolver)* – how to tie tubes into the structural system

• **damage_model** *(damage.DamageCalculator)* – how to calculate damage from the results

• **pset** *(Optional[solverparams.ParameterSet])* – optional set of solver parameters

**add_heuristic** *(heuristic)*

Add a heuristic to use during the solve

Parameters

- **heuristics** *(list)* – new heuristic to add

**calculate_damage** *

Calculate damage from the results

Returns

- Number of allowed daily cycles

Return type

- float

**property ntubes**

Pass through to get the number of tubes that need to be analyzed

Returns

- total number of tubes in entire receiver

Return type

- int

**progress_decorator** *(base, ntotal)*

Either wrap with a progress bar decorator or return a dummy

Parameters

- **base** *(function)* – base function to wrap

- **ntotal** *(int)* – total number in iterator, needed for wrapping iterators

Returns

- either function wrapped with decorator or base function

**solve_heat_transfer** *

Solve the heat transfer problem for each tube

Adds the thermal results to each receiver.Tube object

**solve_life** *

User interface: solve everything and return receiver life

The trigger for everything: solve the complete problem and report the best-estimate life.

Returns

- Number of allowed daily cycles

Return type

- float

**solve_structural** *

Solve the structural problem for the complete system

Adds the structural results to each tube.
property tubes
    Direct iterator over tubes
    Returns iterator over tubes

4.2 Heuristics

The base assumption in srlife is that the thermal, structural, and damage analyses will use full 3D theories, consider every tube in every receiver, and follow the user provided input (thermal history, structural/spring boundary conditions, etc.) exactly. Solver heuristics modify these base assumptions, with the goal of reducing the time required to complete the analysis at the expense of some accuracy. Heuristics can trigger some action at any point throughout the analysis, in the setup, thermal, structural, or damage phases.

All heuristics inherit from a common base class, `srlife.managers.Heuristic`.

class srlife.managers.Heuristic
    Solution heuristic superclass
    Class that defines a heuristic to modify the basic 3D solve process in some way.
    To implement a specific heuristic override the appropriate pure virtual methods.
    
    args_for_tube_thermal_solver(receiver, tube)
    Add tube solver args
    
    Parameters
    • receiver (receiver.receiver) – receiver object affected
    • tube (receiver.tube) – tube object affected

4.2.1 Cycle reset heuristic

The cycle reset heuristic returns each tube to its initial temperature at the end of every thermal cycle. This heuristic represents the effects of a long hold at lower temperature, often omitted in the analysis. When using transient heat transfer not including this heuristic may mean the tube begin to accumulate a small amount of residual stress, related to any unrelaxed thermal gradient still present at the end of each day.

class srlife.managers.CycleResetHeuristic
    Reset the tube temperatures each cycle to the initial values
    
    args_for_tube_thermal_solver(receiver, tube)
    Modify the tube solver before the solve
    
    Parameters
    • receiver (receiver.receiver) – receiver object affected
    • tube (receiver.tube) – tube object affected
4.3 ParameterSet description

A `srlife.solverparams.ParameterSet` is a hierarchical dictionary. The top level dictionary contains global parameters that apply to all solvers, for example

```python
params = solverparams.ParameterSet()
params["nthreads"] = 4
```

specifies that all solvers can use up to 4 parallel threads. In addition, the top level object contains subdictionaries describing the parameters for the thermal, structural, and receiver system solvers. For example, the code

```python
params["thermal"]["rtol"] = 1.0e-6
```

sets the relative tolerance of the thermal solver. The tables below provide the options currently available at each level.

### 4.3.1 Global options

<table>
<thead>
<tr>
<th>Option</th>
<th>Data type</th>
<th>Default</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>nthreads</td>
<td>int</td>
<td>1</td>
<td>Number of parallel threads to use in solves.</td>
</tr>
<tr>
<td>progress</td>
<td>bool</td>
<td>False</td>
<td>Provide progress bar in the command line</td>
</tr>
</tbody>
</table>

### 4.3.2 Thermal solver options

<table>
<thead>
<tr>
<th>Option</th>
<th>Data type</th>
<th>Default</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtol</td>
<td>float</td>
<td>1.0e-6</td>
<td>Nonlinear solver relative tolerance</td>
</tr>
<tr>
<td>atol</td>
<td>float</td>
<td>1.0e-2</td>
<td>Nonlinear solver absolute tolerance</td>
</tr>
<tr>
<td>miter</td>
<td>int</td>
<td>100</td>
<td>Maximum nonlinear solver iterations</td>
</tr>
<tr>
<td>substep</td>
<td>int</td>
<td>1</td>
<td>Divide each higher-level timestep into substep smaller steps</td>
</tr>
<tr>
<td>verbose</td>
<td>bool</td>
<td>False</td>
<td>Print debug information to the terminal</td>
</tr>
<tr>
<td>steady</td>
<td>bool</td>
<td>False</td>
<td>Use steady state heat transfer, i.e. conduction only</td>
</tr>
</tbody>
</table>

### 4.3.3 Tube solver options

<table>
<thead>
<tr>
<th>Option</th>
<th>Data type</th>
<th>Default</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtol</td>
<td>float</td>
<td>1.0e-6</td>
<td>Nonlinear solver relative tolerance</td>
</tr>
<tr>
<td>atol</td>
<td>float</td>
<td>1.0e-8</td>
<td>Nonlinear solver absolute tolerance</td>
</tr>
<tr>
<td>miter</td>
<td>int</td>
<td>10</td>
<td>Maximum nonlinear solver iterations</td>
</tr>
<tr>
<td>qorder</td>
<td>int</td>
<td>1</td>
<td>Quadrature order for the finite element method</td>
</tr>
<tr>
<td>verbose</td>
<td>bool</td>
<td>False</td>
<td>Print debug information to the terminal</td>
</tr>
<tr>
<td>dof_tol</td>
<td>float</td>
<td>1.0e-6</td>
<td>Geometric tolerance for finding nodes on planes</td>
</tr>
</tbody>
</table>
4.3.4 Structural solver options

<table>
<thead>
<tr>
<th>Option</th>
<th>Data type</th>
<th>Default</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtol</td>
<td>float</td>
<td>1.0e-6</td>
<td>Nonlinear solver relative tolerance</td>
</tr>
<tr>
<td>atol</td>
<td>float</td>
<td>1.0e-4</td>
<td>Nonlinear solver absolute tolerance</td>
</tr>
<tr>
<td>miter</td>
<td>int</td>
<td>25</td>
<td>Maximum nonlinear solver iterations</td>
</tr>
<tr>
<td>verbose</td>
<td>bool</td>
<td>False</td>
<td>Print debug information to the terminal</td>
</tr>
</tbody>
</table>

4.3.5 Damage model options

<table>
<thead>
<tr>
<th>Option</th>
<th>Data type</th>
<th>Default</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>extrapolate</td>
<td>string</td>
<td>“lump”</td>
<td>How to extrapolate damage, options are “lump”, “last”, and “poly”</td>
</tr>
<tr>
<td>order</td>
<td>int</td>
<td>1</td>
<td>Polynomial order to use in conjunction with the “poly” option</td>
</tr>
</tbody>
</table>

4.3.6 Class description

```python
class srlife.solverparams.ParameterSet(**kwargs)

Recursive default dictionary that can be pickled

Parameters **kwargs -- Arbitrary keyword arguments to be added to dictionary

get_default(key, value)

Get a key, if not present return the default value
```
5.1 Receiver system

As described in the overview, the user must provide the geometric definition of the tubular receiver and the mechanical and thermal boundary conditions as part of the required package input. The \texttt{srlife.receiver} module defines data structures for providing this information and some simple methods for manipulating the data.

The python class \texttt{srlife.receiver.Receiver} provides the top level interface and uses a hierarchy of objects to describe the receiver geometry and boundary conditions. The figure below describes the structure.

The overall data structure is hierarchical, descending from the single \texttt{srlife.receiver.Receiver} object describing the entire receiver. The receiver has one or more \texttt{srlife.receiver.Panel} objects each of which have one ore more \texttt{srlife.receiver.Tube} objects. Structurally, each tube in a panel is connected through its top-surface displacement through a spring. Similarly, each panel is connected to the other panels through a spring. The spring stiffness defines the type of connection, options are:
1. A string “disconnect” meaning the spring has zero stiffness and the member floats freely, not connected to the result of the objects of its type.

2. A string “rigid” which means the object is rigidly connected to the others of its type with an infinitely-rigid spring.

3. A float giving the discrete spring stiffness.

The `srlife.receiver.Receiver` structure itself stores additional metadata:

1. The cycle period. Typically this will be 24 hours, but could be a reduced representation, for example cutting out a period of the hold in the cold nighttime condition.

2. The number of days (cycle repetitions) actually represented in the structural and thermal boundary conditions. This could, and almost always will be, not the full receiver life in days as the prescribed loads could represent a larger number of actual repetitions in service.

The Tube object holds basic geometric information describing a cylindrical tube, the tube discretization which is a fixed grid in cylindrical coordinates given by a fixed number of radial, circumferential, and axial subdivisions, the initial tube temperature, the discrete times at which the object stores boundary condition information and results, zero or more nodal results fields, zero or more quadrature result fields, an optional inner pressure boundary condition, an optional inner diameter thermal boundary condition, and an optional outer diameter thermal boundary condition. In addition, the tube carries a flag indicating whether the tube should be treated with a full 3D analysis or a reduced 2D or 1D analysis. For the reduced analysis the tube retains the height of the slice (2D and 1D) and the angle of the slice (1D), with respect to the full 3D cylindrical coordinate system.

The tube result and boundary condition times are a 1D array of time values, starting at zero. This array has length $n_{\text{time}}$ and must be consistent for all the nodal results, quadrature results, pressure boundary conditions, and thermal boundary conditions.

Nodal results are stored on a fixed grid defined by an array of size $n_{\text{time}} \times n_r \times n_\theta \times n_z$

where $n_r$ is the number of radial grid points, $n_\theta$ is the number of circumferential grid points, and $n_z$ is the number of axial grid points. A 2D representation leaves off the last axis and a 1D representation leaves off the last two axes.

The inner pressure boundary condition `srlife.receiver.PressureBC` holds a 1D array of pressures defining the pressure at discrete times. The size of this 1D array must be $n_{\text{time}}$ and match the tube object.

The user has three options for thermal boundary conditions, which can be specified on the inner diameter, outer diameter, or both:


The data for the flux and temperature boundary conditions is an array of flux or temperature values of size $n_{\text{time}} \times n_\theta \times n_z$ representing a fixed grid of points on either the tube inner or outer diameter. The user must provide the full 3D data. Similarly, the data for the convective heat transfer boundary condition is an array of fluid temperature data of size $n_{\text{time}} \times n_z$.

Again, the user must always provide the full 3D information.

The user can provide the required input data in two ways:
1. The *Python objects*

2. An *HDF5 file*

Both methods provide options for linking into external programs.

## 5.2 Python objects

### 5.2.1 Receiver geometry

The Receiver object contains panels and the interconnect spring stiffness between the panels.

```python
class srlife.receiver.Receiver(period, days, panel_stiffness)
```

Basic definition of the tubular receiver geometry.

A receiver is a collection of panels linked together by an elastic spring stiffness. This stiffness can be a real number, “rigid” or “disconnect”

Panels can be labeled by strings. By default the names are sequential numbers.

In addition this object stores some required metadata:

1) The daily cycle period (which can be less than 24 hours if the analysis neglects some of the night period)

2) The number of days (see #1) explicitly represented in the analysis results.

**Parameters**

- `period (float)` – single daily cycle period
- `days (int)` – number of daily cycles explicitly represented
- `panel_stiffness (float or string)` – panel stiffness (float) or “rigid” or “disconnect”

```python
add_panel(panel, name=None)
```

Add a panel object to the receiver

**Parameters**

- `panel (Panel)` – panel object
- `name (Optional[str])` – panel name, by default follows fixed scheme

```python
close(other)
```

Check to see if two objects are nearly equal.

Primarily used for testing

**Parameters**

- `other (Receiver)` – the object to compare against

**Returns** True if the receivers are similar.

**Return type** bool

```python
classmethod load(fobj)
```

Load a Receiver from an HDF5 file

A full description of the HDF format is included in the module documentation

**Parameters**

- `fobj (string)` – either a h5py file object or a filename

**Returns** The constructed receiver object.
Return type *Receiver*

**property npanels**
Number of panels in the receiver

**Returns** Number of panels

**Return type** int

**property ntubes**
Shortcut for total number of tubes

**Returns** Number of tubes in all panels

**Return type** int

**save** *(fobj)*
Save to an HDF5 file

This saves a Receiver object to the HDF5 format.

**Parameters**

- **fobj** *(str)* – either a h5py file object or a filename

**set_paging** *(page)*
Tell tubes to store results on or off disk

**Parameters**

- **page** *(bool)* – if true, page results to disk

**property tubes**
Shortcut iterator over all tubes

**Returns** iterator over panels

**write_vtk** *(basename)*
Write out the receiver as individual panels with names basename_panelname

The VTK format is mostly used for additional postprocessing. The VTK format cannot be used for input.

**Parameters**

- **basename** *(str)* – base file name

The Panel object contains tubes and the interconnect spring stiffness between each tube in the panel

**class** *srlifereceiver.Panel*(stiffness)

Basic definition of a panel in a tubular receiver.

A panel is a collection of Tube object linked together by an elastic spring stiffness. This stiffness can be a real number, a string “disconnect” or a string “rigid”

Tubes in the panel can be labeled by strings. By default the names are sequential numbers.

**Parameters**

- **stiffness** – manifold spring stiffness

**add_tube**(tube, name=None)
Add a tube object to the panel

**Parameters**

- **tube** *(Tube)* – tube object
- **name** *(Optional[str])* – Tube name, defaults to fixed scheme.

**close**(other)
Check to see if two objects are nearly equal.

Primarily used for testing
**Parameters**

other (Panel) – the object to compare against

**Returns**

true if the panels are sufficiently similar

**Return type**

`bool`

### classmethod `load(fobj)`

Load from an HDF5 file

**Parameters**

fobj (h5py.Group) – h5py group containing the panel

### property `ntubes`

Number of tubes in the panel

**Returns**

number of tubes in the panel

**Return type**

`int`

### `save(fobj)`

Save to an HDF5 file

**Parameters**

fobj (h5py.Group) – h5py group

### `write_vtk(basename)`

Write out the panels as individual tubes with names basename_tubename

**Parameters**

basename (string) – base file name

---

The Tube object contains the majority of the analysis information and results of the analysis. The structure holds the basic tube geometry, the tube discretization, described as intervals in a cylindrical coordinate system, and initial tube temperature, the thermal and mechanical boundary conditions, and result fields stored at discrete times and either at node points or quadrature points.

```python
class srlife.receiver.Tube(outer_radius, thickness, height, nr, nt, nz, T0=0.0, page=False)
```

Geometry, boundary conditions, and results for a single tube.

The basic tube geometry is defined by an outer radius, thickness, and height.

Results are given at fixed times and on a regular polar grid defined by a number of r, theta, and z increments. The grid points are then deduced by linear subdivision in r between the outer radius and the outer radius - t, 0 to 2 pi, and 0 to the tube height.

Result fields are general and provided by a list of names. The receiver package uses the metal temperatures, stresses, mechanical strains, and inelastic strains.

Analysis results can be provided over the full 3D grid (default), a single 2D plane (identified by a height), or a single 1D line (identified by a height and a theta position)

Boundary conditions may be provided in two ways, either as fluid conditions or net heat fluxes. These are defined in the HeatFluxBC or ConvectionBC objects below.

**Parameters**

- `outer_radius (float)` – tube outer radius
- `thickness (float)` – tube thickness
- `height (float)` – tube height
- `nr (int)` – number of radial increments
- `nt (int)` – number of circumferential increments
- `nz (int)` – number of axial increments
- `T0 (Optional[float])` – initial temperature
• page (Optional[bool]) – store results on disk if True

add_blank_quadrature_results(name, shape)
Add a blank quadrature point result field

Parameters
• name (str) – parameter set name
• shape (tuple) – required shape

add_blank_results(name, shape)
Add a blank node point result field

Parameters
• name (str) – parameter set name
• shape (tuple) – required shape

add_quadrature_results(name, data)
Add a result at the quadrature points

Parameters
• name (str) – parameter set name
• data (np.array) – actual results data

add_results(name, data)
Add a node point result field

Parameters
• name (str) – parameter set name
• data (np.array) – actual results data

close(other)
Check to see if two objects are nearly equal.
Primarily used for testing

Parameters other (Tube) – the object to compare against

Returns true if the tubes are similar
Return type bool

copy_results(other)
Copy the results fields from one tube to another

Parameters other – other tube object

property dim
Actual problem discretization

Returns tuple giving the fixed grid discretization
Return type tuple(int)

element_volumes()
Calculate the element volumes

Returns np.array with each element volume
class method load(fobj)
Load from an HDF5 file

Parameters fobj (h5py.Group) – h5py to load from

make_1D(height, angle)
Abstract the tube as 1D

Reduce to a 1D abstraction along a ray given by the provided height and angle.

Parameters

• height (float) – the height of the ray
• angle (float) – the angle, in radians

make_2D(height)
Abstract the tube as 2D

Reduce to a 2D abstraction by slicing the tube at the indicated height

Parameters height (float) – the height at which to slice

property mesh
Calculate the problem mesh (should only be needed for I/O)

Returns Results of np.meshgrid over the problem discretization

property ndim
Number of problem dimensions

Returns tube dimension

Return type int

property ntime
Number of time steps

Returns number of time steps

Return type int

save(fobj)
Save to an HDF5 file

Parameters fobj (h5py.Group) – h5py group to save to

set_bc(bc, loc)
Set the inner or outer heat flux BC

Parameters

• bc (ThermalBC) – boundary condition object
• loc (string) – location – either “inner” or “outer” wall

set_paging(page, i)
Set the value of the page parameter

Parameters

• page – if true store results on disk
• i – tube number to use

5.2. Python objects
**5.2.2 Boundary conditions**

The only mechanical boundary condition the user needs to specify is the Tube internal pressure.

```python
class srlife.receiver.PressureBC(times, data)
    Stores information about the tube internal pressure
    Simple class to store tube pressure, assumed to be constant in space and just vary with time.
    Parameters
        • times (np.array) – times throughout load cycle
        • data (np.array) – pressure values
    close(other)
        Test method for comparing BCs
        Parameters other (PressureBC) – other object
        Returns true if the objects are sufficiently similar
        Return type bool
   classmethod load(fobj)
        Load from an HDF5 file
        Parameters fobj (h5py.Group) – h5py group to load from
    property ntime
        Number of time steps
        Returns number of time steps in the pressure history definition
        Return type int
    pressure(t)
        Return the pressure as a function of time
        Parameters t (float) – time
        Returns internal pressure at that time
        Return type float
```
save(obj)

Save to an HDF5 file

Parameters:

- **fobj** (*h5py.Group*) – h5py group to save to

Superclass handing HDF storage dispatch for thermal boundary conditions

class srlife.receiver.ThermalBC

Superclass for thermal boundary conditions.

Currently just here to handle dispatch for HDF files

Thermal boundary conditions options are fixed flux (HeatFluxBC), fixed temperature (FixedTempBC), and convection (ConvectiveBC).

class srlife.receiver.HeatFluxBC(radius, height, nt, nz, times, data)

A net heat flux on the radius of a tube. Positive is heat input, negative is heat output.

These conditions are defined on the surface of a tube at fixed times given by a radius and a height. The radius is not used in defining the BC but is used to ensure the BC is consistent with the Tube object.

The heat flux is given on a regular grid of theta, z points each defined but a number of increments. This grid need not agree with the Tube solid grid.

Parameters:

- **radius** (*float*) – boundary condition application radius
- **height** (*float*) – tube height
- **nt** (*int*) – number of circumferential increments
- **nz** (*int*) – number of axial increments
- **times** (*np.array*) – heat flux times
- **data** (*np.array*) – heat flux data

class srlife.receiver.FixedTempBC(radius, height, nt, nz, times, data)

Fixed temperature BC.

These conditions are defined on the surface of a tube at fixed times given by a radius and a height. The radius is not used in defining the BC but is used to ensure the BC is consistent with the Tube object.

The heat flux is given on a regular grid of theta, z points each defined but a number of increments. This grid need not agree with the Tube solid grid.

Parameters:

- **radius** (*float*) – boundary condition application radius
- **height** (*float*) – tube height
- **nt** (*int*) – number of circumferential increments
- **nz** (*int*) – number of axial increments
- **times** (*np.array*) – fixed temperature times
- **data** (*np.array*) – fixed temperature data

class srlife.receiver.ConvectiveBC(radius, height, nz, times, data)

A convective BC on the surface of a tube defined by a radius and height.

The radius is not used in defining the BC, but is used to check consistency with the Tube object.

5.2. Python objects
This condition is defined axially by a fluid temperature at fixed times on a fixed grid of z points defined by a number of increments.

**Parameters**

- **radius** (*float*) – radius of application
- **height** (*float*) – height of fluid temperature info
- **nz** (*int*) – number of axial increments
- **times** (*np.array*) – data times
- **data** (*np.array*) – actual fluid temperature data

### 5.3 HDF5 file

HDF5 provides a hierarchical data structure aimed at storing and transferring scientific data. srlife uses the format, through the h5py package, to serialize and store the definition of a complete receiver to file. A user could write an HDF5 file with the appropriate format to interface with srlife from an external program.

The interface uses HDF5 attributes, datasets, and groups to store the data. An attributes is metadata, a single string, bool, float, integer, etc. A dataset is a fixed-size array, essentially in the h5py package a numpy array. A group is a container that can hold attributes, datasets, or additional groups.

The format adopted here is then an exact mirror of the type structure defined above. The `srlife.receiver.Receiver` object’s metadata and data sits at the top of the HDF5 file (i.e. not contained in a group), and stores the `srlife.receiver.Panel`, `srlife.receiver.Tube`, `srlife.receiver.PressureBC`, and `srlife.receiver.ThermalBC` as groups and subgroups. The HDF5 file maintains the hierarchical structure of the python data structures, so the receiver stores the panels in a group, each panel stores its tubes in a group, and the thermal and pressure BCs are subgroups of the tubes.

The following summarizes the HDF5 format. Example HDF5 files are contained in the `srlife/examples` directory.

#### 5.3.1 Receiver

This is the top-level of the HDF5 file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Data type</th>
<th>Explanation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>period</td>
<td>attribute</td>
<td>float</td>
<td>Daily cycle period</td>
<td></td>
</tr>
<tr>
<td>days</td>
<td>attribute</td>
<td>int</td>
<td>Number of cycle repetitions</td>
<td></td>
</tr>
<tr>
<td>stiffness</td>
<td>attribute</td>
<td>float/string</td>
<td>Panel interconnect stiffness</td>
<td>&quot;disconnect&quot;, &quot;rigid&quot;, or value of stiffness</td>
</tr>
<tr>
<td>panels</td>
<td>group</td>
<td>n/a</td>
<td>Each panel is a subgroup of this group</td>
<td>Default naming scheme: 0, 1, 2,...</td>
</tr>
</tbody>
</table>
### 5.3.2 Panel

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Data Type</th>
<th>Explanation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>stiffness</td>
<td>attribute</td>
<td>float/string</td>
<td>Panel interconnect stiffness</td>
<td>“disconnect”, “rigid”, or value of stiffness</td>
</tr>
<tr>
<td>tubes</td>
<td>group</td>
<td>n/a</td>
<td>Each tube is a subgroup of this group</td>
<td>Default naming scheme: 0, 1, 2, …</td>
</tr>
</tbody>
</table>

### 5.3.3 Tube

When providing an HDF5 file as input the results and quadrature_results groups must exist but the user does not need to provide any pre-populated results fields (srlife will add these during the analysis).

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Data Type</th>
<th>Explanation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>attribute</td>
<td>float</td>
<td>Tube inner radius</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>attribute</td>
<td>float</td>
<td>Tube thickness</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>attribute</td>
<td>float</td>
<td>Tube height</td>
<td></td>
</tr>
<tr>
<td>nr</td>
<td>attribute</td>
<td>int</td>
<td>Number of radial nodes</td>
<td></td>
</tr>
<tr>
<td>nt</td>
<td>attribute</td>
<td>int</td>
<td>Number of circumferential nodes</td>
<td></td>
</tr>
<tr>
<td>nz</td>
<td>attribute</td>
<td>int</td>
<td>Number of axial nodes</td>
<td></td>
</tr>
<tr>
<td>abstraction</td>
<td>attribute</td>
<td>string</td>
<td>Tube dimension</td>
<td>“1D”, “2D”, or “3D”</td>
</tr>
<tr>
<td>times</td>
<td>dataset</td>
<td>dim: (ntime,)</td>
<td>Discrete time points</td>
<td></td>
</tr>
<tr>
<td>results</td>
<td>group</td>
<td>n/a</td>
<td>Node point results</td>
<td>Each result field is a dataset</td>
</tr>
<tr>
<td>quadrature_results</td>
<td>group</td>
<td>n/a</td>
<td>Quadrature point results</td>
<td>Each result field is a dataset</td>
</tr>
<tr>
<td>outer_bc</td>
<td>group</td>
<td>n/a</td>
<td>Outer diameter thermal BC</td>
<td>See below</td>
</tr>
<tr>
<td>inner_bc</td>
<td>group</td>
<td>n/a</td>
<td>Inner diameter thermal BC</td>
<td>See below</td>
</tr>
<tr>
<td>pressure_bc</td>
<td>group</td>
<td>n/a</td>
<td>Internal pressure</td>
<td>See below</td>
</tr>
</tbody>
</table>

### 5.3.4 HeatFluxBC

The user must always provide the full flux information (i.e. over the full 3D tube).

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Data Type</th>
<th>Explanation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>attribute</td>
<td>string</td>
<td>Thermal BC type</td>
<td>Must be “HeatFlux”</td>
</tr>
<tr>
<td>r</td>
<td>attribute</td>
<td>float</td>
<td>Radius of application</td>
<td>Must match tube inner or outer radius</td>
</tr>
<tr>
<td>h</td>
<td>attribute</td>
<td>float</td>
<td>Tube height</td>
<td>Must match tube height</td>
</tr>
<tr>
<td>nt</td>
<td>attribute</td>
<td>int</td>
<td>Number of circumferential nodes</td>
<td></td>
</tr>
<tr>
<td>nz</td>
<td>attribute</td>
<td>int</td>
<td>Number of axial nodes</td>
<td></td>
</tr>
<tr>
<td>times</td>
<td>dataset</td>
<td>dim: (ntime,)</td>
<td>Discrete times points</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>dataset</td>
<td>dim: (ntime, nt, nz)</td>
<td>Discrete flux data</td>
<td>Fixed array over tube inner/outer surface</td>
</tr>
</tbody>
</table>
### 5.3.5 FixedTempBC

The user must always provide the full temperature information (i.e. over the full 3D tube).

<table>
<thead>
<tr>
<th>Field</th>
<th>Type attribute</th>
<th>Data type</th>
<th>Explanation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>attribute</td>
<td>string</td>
<td>Thermal BC type</td>
<td>Must be “FixedTemp”</td>
</tr>
<tr>
<td>r</td>
<td>attribute</td>
<td>float</td>
<td>Radius of application</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>attribute</td>
<td>float</td>
<td>Tube height</td>
<td></td>
</tr>
<tr>
<td>nt</td>
<td>attribute</td>
<td>int</td>
<td>Number of circumferential nodes</td>
<td></td>
</tr>
<tr>
<td>nz</td>
<td>attribute</td>
<td>int</td>
<td>Number of axial nodes</td>
<td></td>
</tr>
<tr>
<td>times</td>
<td>dataset</td>
<td>dim: (ntime,)</td>
<td>Discrete times points</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>dataset</td>
<td>dim: (ntime, nt, nz)</td>
<td>Discrete temperature data</td>
<td>Fixed array over tube inner/outer surface</td>
</tr>
</tbody>
</table>

### 5.3.6 ConvectiveBC

The user must always provide the full fluid temperature information (i.e. over the full 3D tube).

<table>
<thead>
<tr>
<th>Field</th>
<th>Type attribute</th>
<th>Data type</th>
<th>Explanation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>attribute</td>
<td>string</td>
<td>Thermal BC type</td>
<td>Must be “Convective”</td>
</tr>
<tr>
<td>r</td>
<td>attribute</td>
<td>float</td>
<td>Radius of application</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>attribute</td>
<td>float</td>
<td>Tube height</td>
<td></td>
</tr>
<tr>
<td>nz</td>
<td>attribute</td>
<td>int</td>
<td>Number of axial nodes</td>
<td></td>
</tr>
<tr>
<td>times</td>
<td>dataset</td>
<td>dim: (ntime,)</td>
<td>Discrete times points</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>dataset</td>
<td>dim: (ntime, nz)</td>
<td>Discrete fluid temperature data</td>
<td>Fixed array over tube height</td>
</tr>
</tbody>
</table>

### 5.3.7 PressureBC

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Data type</th>
<th>Explanation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>times</td>
<td>dataset</td>
<td>dim: (ntime,)</td>
<td>Discrete times points</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>dataset</td>
<td>dim: (ntime,)</td>
<td>Discrete pressure data</td>
<td></td>
</tr>
</tbody>
</table>
MATERIALS: THERMAL AND STRUCTURAL MATERIAL MODELS

srlife includes a library of material models to provide the data required to solve the thermal, structural, and damage analysis calculations. These models represent a best estimate of average material properties, meaning the analysis results and the final life estimate represent average and not lower bound design estimates.

The material system references the tube metallic material. The table below lists the available metallic material types along with a comment on the source of data and reliability of the current model.

<table>
<thead>
<tr>
<th>Material</th>
<th>String ID</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>316H stainless steel</td>
<td>“316H”</td>
<td>Large US DOE nuclear energy database</td>
</tr>
<tr>
<td>800H high alloy steel</td>
<td>“800H”</td>
<td>Large US DOE nuclear energy database</td>
</tr>
<tr>
<td>Alloy 230 Ni-based alloy</td>
<td>“A230”</td>
<td>Extremely limited literature data, creep-fatigue properties questionable</td>
</tr>
<tr>
<td>Alloy 617 Ni-based alloy</td>
<td>“A617”</td>
<td>Large US DOE nuclear energy database</td>
</tr>
<tr>
<td>Alloy 740H Ni-based alloy</td>
<td>“740H”</td>
<td>Limited creep-fatigue database on single heat</td>
</tr>
<tr>
<td>Alloy 282 Ni-based alloy</td>
<td>“A282”</td>
<td>Limited literature data, creep-fatigue properties preliminary</td>
</tr>
</tbody>
</table>

The material system accommodates variants within each model type. The srlife package provides a “base” variant for all the materials. The user can add additional material variants to explore alternate model forms or models calibrated against different datasets. The comments in the table above only apply to the “base” material variant.

6.1 Material model descriptions

6.1.1 Thermal materials

The thermal material model provides the metal’s conductivity and diffusivity as a function of temperature.

6.1.2 Deformation materials

The deformation model system is a thin wrapper around the neml – a nonlinear constitutive model system focused on high temperature materials developed by Argonne National Laboratory. The “base” models for each material are decoupled creep-plasticity models representing both rate independent plasticity and rate dependent plasticity. These models are not as accurate as fully-coupled viscoplastic models, but are easily calibrated against commonly-available experimental data.
6.1.3 Damage (structural) materials

The current version of srlife includes only one damage model: an ASME-type approach that uses Miner’s rule, time-fraction creep damage, and a creep-fatigue interaction diagram to determine creep-fatigue failure. The damage material model provides the required data, including nominal strain-based fatigue curves, a creep rupture correlation, and the interaction diagram.

6.1.4 Fluid materials

The fluid material system is different than the other three materials. The model describes the convective heat transfer coefficient between the base metal and a given fluid as a function of temperature. Note that this neglects the influence of flow rate, which may be incorporated into future versions. Instead of being indexed against the tube material the fluid material systems indexes the models first against the coolant type. The table below describes the options currently embedded in NEML.

<table>
<thead>
<tr>
<th>Fluid</th>
<th>String ID</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Molten chloride salt</td>
<td>“salt”</td>
<td></td>
</tr>
</tbody>
</table>

As with the metallic material models, the fluid material system also subdivides models with a variant specification. Again, srlife provides a “base” variant and users could expand the system to other models.

Convective heat transfer properties could vary both with the working fluid type and with the tube material. The fluid material system provides a default option suitable for most metallic tube material and, where data is available, also specializes the fluid model to specific materials.

6.2 Loading material models

The user only needs to interact with two functions to load in material models. The first loads in the thermal, deformation, and structural models for a particular tube material:
srlife.library.load_material(name, thermal_model, deformation_model, damage_model)

Load solid material properties

Parameters

- name (str) – name of the material (title of xml file)
- thermal_model (str) – which thermal model variant to use
- deformation_model (str) – which deformation model variant to use
- damage_model (str) – which damage model variant to use

Returns thermal material model materials.DeformationMaterial: deformation material model materials.StructuralMaterial: damage material model

Return type materials.ThermalMaterial

The second loads in data for a particular working fluid:
srlife.library.load_fluid(name, model)

Load fluid material properties:

Parameters

- name (str) – name of the fluid (title of xml file)
• **model (str)** – particular convection model to use

**Returns**  fluid material model

**Return type**  material.FluidMaterial
CHAPTER
SEVEN

SOLVERS: REUSABLE NONLINEAR SOLVERS

7.1 newton description

`srlife.solvers.newton` is a reusable Newton-Raphson solver with backtracking line search that can be used wherever we need to solve a nonlinear system of equations.

`srlife.solvers.newton(RJ, x0, rel_tol=-5.0, abs_tol=1e-08, mites=20, linear_solver=<function solve>, verbose=True, return_extra=False, linesearch=True, max_search=10)`

Simple newton-raphson solver

**Parameters**

- **RJ** – function that gives the residual and jacobian values
- **x0** – initial guess
- **rel_tol** *(Optional*[1.0e-6]) – relative convergence tolerance
- **abs_tol** *(Optional*[1.0e-8]) – absolute convergence tolerance
- **mites** *(Optional*[20]) – maximum number of iterations
- **linear_solver** *(Optional*[numpy.linalg.solve]) – function that solves the linear system A x = b
- **verbose** *(Optional*[True]) – if true, print debug info,
- **return_extra** *(Optional*[False]) – if true also return the final residual vector and Jacobian
- **linesearch** *(Optional*[True]) – if true do backtracking linesearch
- **max_search** *(Optional*[10]) – max number of backtracking steps