




FINIX - Fuel behavior model and interface for multiphysics applications - Code documentation for version 1.19.12

Authors: Henri Loukusa, Jussi Peltonen, Ville Valtavirta

Confidentiality: Public

Report's title FINIX - Fuel behavior model and interface for multiphysics applications - Code documentation for version 1.19.12	
Customer, contact person, address VYR/SAFIR2022	Order reference
Project name Interdisciplinary Fuels And Materials	Project number/Short name 122317/INFLAME
Author(s) Henri Loukusa, Jussi Peltonen, Ville Valtavirta	Pages 96/–
Keywords fuel behaviour modelling, FINIX	Report identification code VTT-R-01103-19
<p>Summary</p> <p>The FINIX fuel behavior code has been updated to version 1.19.12. The new version has been validated, for which a separate report has been published. Several small corrections have been made and a new automatic timestepping algorithm implemented for steady-state scenarios.</p> <p>The FINIX code has been designed to provide a fuel behavior module for other calculation codes in multiphysics simulations. The intended use is the improvement of fuel behavior description in neutronics, thermal hydraulics and reactor dynamics codes, without having to employ full-scale fuel performance codes. FINIX couples with the host code on a source code level, and provides an interface of functions that can be used to access the fuel behavior model from the host code. At VTT, FINIX has been integrated into the Monte Carlo reactor physics code Serpent 2, and reactor dynamics codes TRAB-1D, TRAB3D and HEXTRAN.</p> <p>FINIX consists of several interconnected models that describe the thermo-mechanical behavior of the fuel rod. FINIX solves the transient or steady-state heat equation, with couplings to the cladding and pellet mechanical behavior through the gap conductance and pressure. Publicly available experimental correlations are used for the material properties, and simple models for the heat transfer from the cladding to the coolant have been included.</p> <p>FINIX has been verified against the FRAPTRAN and FRAPCON fuel performance codes in RIA and steady-state scenarios, and compared against experimental Halden reactor data. Results indicate good agreement with the FRAPTRAN and FRAPCON codes and experimental measurements. Limitations of the present version in the simulated scenarios have also been identified.</p>	
Confidentiality	Public
Espoo 9.12.2019	
Written by  Henri Loukusa, Research Scientist	Reviewed by  Janne Heikinheimo, Research Scientist
Accepted by  Ville Tulkki, Research Team Leader	
VTT's contact address VTT Technical Research Centre of Finland, P.O. Box 1000, FI-02044 VTT, FINLAND	
Distribution (customer and VTT) SAFIR RG3, VTT	
<i>The use of the name of the VTT Technical Research Centre of Finland (VTT) in advertising or publication in part of this report is only permissible with written authorisation from the VTT Technical Research Centre of Finland.</i>	

Contents

Contents	5
1. Introduction.....	6
2. General model description	8
2.1 Version	8
2.2 Model assumptions and approximations	8
2.2.1 Geometry	8
2.2.2 Fuel pellet	9
2.2.3 Cladding.....	9
2.2.4 Fill gas and FGR.....	9
2.3 Changes from previous versions	9
2.4 FINIX publications.....	10
3. Thermal model	12
3.1 The heat equation.....	12
3.2 Plenum temperature	13
3.3 Gas gap conductance.....	14
3.3.1 Conduction through the gas	14
3.3.2 Radiation across the gap.....	16
3.3.3 Contact between the pellet and cladding	16
4. Mechanical model	17
4.1 Internal pressure.....	17
4.2 Pellet mechanical model.....	17
4.3 Cladding mechanical model	17
4.3.1 Open gap model	18
4.3.2 Closed gap model.....	19
4.3.3 Plastic deformation model	21
4.3.4 Creep model	23
4.3.5 Failure models	24
5. Other models.....	25
5.1 Fission gas release.....	25
5.1.1 Intragranular gas behaviour.....	25
5.1.2 Intergranular gas behaviour and release	26

5.2	Coolant.....	27
5.2.1	Axial coolant temperature rise model.....	27
5.2.2	Cladding-coolant heat transfer coefficient.....	28
5.3	Cladding oxidation	28
5.4	Grain growth	30
5.5	Radial power distribution	30
5.6	Fast neutron flux and fluence	32
6.	Material correlations.....	33
6.1	Fuel properties.....	33
6.1.1	Specific heat	33
6.1.2	Thermal conductivity	33
6.1.3	Thermal strain.....	33
6.1.4	Radial pellet relocation	34
6.1.5	Pellet swelling	34
6.1.6	Pellet densification.....	35
6.2	Cladding properties	35
6.2.1	Specific heat	35
6.2.2	Thermal conductivity	37
6.2.3	Thermal strain.....	37
6.2.4	Meyer's hardness.....	38
6.2.5	Young's modulus.....	39
6.2.6	Poisson's ratio.....	39
6.2.7	Primary creep	39
6.2.8	Secondary creep	40
6.2.9	Yield stress	41
6.3	Cladding oxide properties.....	44
6.3.1	Heat capacity	44
6.3.2	Thermal conductivity	45
6.4	Fission gas release properties	45
6.4.1	Gas atom production rate.....	45
6.4.2	Gas atom diffusion coefficient	45
6.4.3	Intragranular bubble radius.....	45
6.4.4	Volume diffusion coefficient.....	45
6.4.5	Intragranular bubble number density.....	45
6.4.6	Grain boundary vacancy diffusion coefficient	45

6.5	Gas gap and plenum properties	46
6.6	Neutronics	46
6.6.1	Cross sections	46
6.7	Coolant	47
6.7.1	Dittus-Boelter heat transfer coefficient	47
6.7.2	Nucleate boiling heat transfer coefficient	47
6.7.3	Density	47
6.7.4	Isobaric and isochoric heat capacity	48
6.7.5	Dynamic viscosity	48
6.7.6	Thermal conductivity	49
7.	Numerical implementation	52
7.1	General outline of the execution order	52
7.2	Thermal model	52
7.2.1	Transient heat equation	52
7.2.2	Steady-state heat equation	57
7.3	Plenum temperature	62
7.4	Mechanical model	63
7.4.1	Rigid pellet model	63
7.4.2	Cladding model	63
7.5	Oxidation model	63
7.6	Coolant model	64
7.7	Fission gas release	64
7.8	Other numerical features	64
7.8.1	Automatic timestepping	64
7.8.2	Renodalization in restarted calculation	64
8.	Usage instructions	66
8.1	General description	66
8.2	Units	66
8.3	Data structures	67
8.3.1	Main structures	67
8.3.2	Auxiliary structures	67

8.4	Function naming conventions.....	68
8.5	Error message system.....	69
8.6	System setup and simulation.....	70
8.6.1	Initialization.....	70
8.6.2	Transient simulation.....	71
8.6.3	Steady-state simulation.....	71
8.6.4	Updating boundary conditions.....	72
8.6.5	Convergence criterion.....	72
8.6.6	Ending the simulation.....	73
8.7	Input instructions.....	73
8.7.1	Input file finix_options.inp.....	74
8.7.2	Input file finix_rod.inp.....	75
8.7.3	Input file finix_scenario.inp.....	75
8.8	Output files.....	83
8.8.1	finix.sum and finix.z*.....	83
8.8.2	finix_data_structures.dbg.....	83
8.8.3	finix_stripfile.txt.....	86
8.8.4	finix_restart.....	87
9.	Code assessment.....	88
9.1	General performance.....	88
9.2	Verification of the restart feature.....	88
9.3	Solved issues from previous versions.....	89
9.4	Known issues and possible caveats.....	89
10.	Summary.....	90
	References.....	96

1. Introduction

The assessment of the performance of a fuel rod in the reactor core is an integral part of the design, operation, and safety analysis of the nuclear reactor. To study the behavior of the fuel rod, one typically resorts to using a model in one of the two extremes. On one end are the dedicated fuel performance codes, which take into account the multitude of physical phenomena involved in the thermo-mechanical behavior of the fuel rod: diffusion of heat, elastic and plastic deformation of the pellet and the cladding, release of gaseous fission products into the free volume, the interplay of the gas pressure with the mechanical solution of the pellet and the cladding, the feed-back of the deformations and temperatures to the gap heat conductance, the effect of the cladding surface heat flux to the heat transfer into the surrounding coolant, and so on. All of this is done with a complex, interconnected model, where experimental correlations are used to model the dependencies of the material properties on temperature, pressure, burn-up, etc.

On the other end of the spectrum are the models used within, *e.g.*, many thermal-hydraulics or neutronics codes, which are based on simple correlations, non-mechanical thermal elements, or even fixed values of temperature. Although they are quick to understand and efficient to solve, such fuel models may be less-than-realistic in, for instance, transient conditions or, in cases where fuel with extended burn-up should be considered. The relationships between the different reactor core physics descriptions can be seen in the fig. 1.

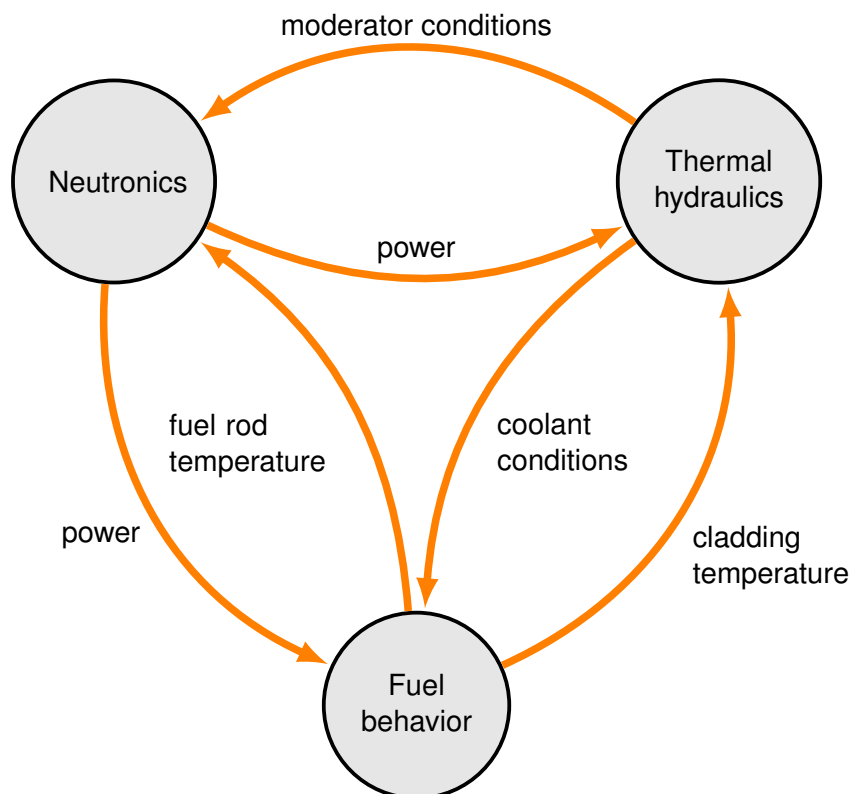


Figure 1. The relationship of different descriptions of nuclear reactor core physics.

The purpose of FINIX is to provide a fuel performance model to be used in a multiphysics context, allowing it to be coupled to existing thermal-hydraulics, reactor dynamics or neutronics codes used at VTT, such as TRAB, HEXTRAN and Serpent. The scope of the FINIX code is somewhere between the full-fledged fuel performance codes and the simple thermal element: although FINIX employs many of the same experimental correlations as the full fuel performance codes, and solves the thermal and mechanical behavior of the rod, several simplifications have been made, both to improve the performance of the code, and to expedite its development. These assumptions and approximations are discussed in Section 2.2.

In the first stage of development, the aim was to develop a model that is capable of solving the transient heat equation, with couplings to the cladding and pellet mechanical behavior through the gap conductance and pressure. In the current second stage of development, models and correlations required for simulation of extended irradiation periods are implemented, and a steady-state heat equation solver has been developed. Experimental correlations are used for the material properties, and simple models for the heat transfer from the cladding to the coolant have been included. The latter can also be easily replaced by the coupling to a thermal-hydraulics code. The physical models, correlations and their numerical implementation is described in Sections 3–7.

The FINIX code has been designed so that it can be coupled on a source-code level, so that passing input and output files between the codes is not necessary. FINIX includes a collection of built-in functions that can be used for basic setup of the system, and for running the actual simulations, using a fairly high-level syntax. In addition, FINIX has an error message system that can be used to detect beyond-normal operation of the code without aborting program execution. The usage of this high-level interface is described in Section 8. Because of the direct coupling on a source-code level, FINIX allows for low-level (detailed) control of its input and output. Since FINIX-0.15.6, input and output files for stand-alone usage have been defined and examples are provided with the code.

Assessment of the FINIX-1.19.1 code without external coupling is presented in a separate report [1], with a summary of the results given in Section 9 of this report. The current version has not been validated completely, and a separate validation report will be produced in the future. FINIX is compared with the FRAPTRAN fuel performance code in several RIA scenarios, and with experimental Halden reactor data. The results show good agreement both with FRAPTRAN and the experiments. Known limitations of FINIX are also discussed in Section 9 and in the validation report [1].

2. General model description

2.1 Version

This document describes version 1.19.12 of the FINIX fuel behavior model. The version number follows the convention where the first number identifies the general stage of development ("0" for early development stage, "1" for first nearly mature version) and the following numbers identify the date of release (year followed by month).

The version history of FINIX and the associated reports are shown collected in table 1.

Table 1. The FINIX version history with references to the associated documentation and validation reports.

Version	Publication	Authors	Report	Validation	Other
1.19.12	Nov. 2019	Loukusa, H., Peltonen, J. and Valtavirta, V.	This report.		
1.19.1	Jan. 2019	Loukusa, H., Peltonen, J. and Valtavirta, V.	[2]	[1]	
1.18.9	Sep. 2018	Loukusa, H. and Valtavirta, V.	[3]	[4]	
0.17.12	Dec. 2017	Loukusa, H. and Valtavirta, V.	[5]		[6]
0.15.12	Dec. 2015	Ikonen, T., Kättö, J. and Loukusa, H.	[7]		
0.15.6	Jun. 2015	Ikonen, T., Kättö, J. and Loukusa, H.	[8]		
0.13.9	Sep. 2013	Ikonen, T.	[9]	[10]	
0.13.1	Jan. 2013	Ikonen, T.	[11]		

2.2 Model assumptions and approximations

2.2.1 Geometry

The FINIX model solves the heat equation and the cladding mechanical behavior in cylindrical geometry. Furthermore, the heat equation is solved in one dimension, with the temperature having dependence only on the radial coordinate r . The azimuthal (θ -) dependence is completely neglected (implying also the assumption that the central axes of the pellet and the cladding are the same), and the axial (z -) dependence is only included by solving the heat equation independently for several axial slices, or nodes. However, there is no heat flux between the neighboring axial nodes. Therefore, the model is only applicable to scenarios where the axial heat transfer is small compared to the radial heat transfer, and where the boundary conditions and the power distribution are symmetric with respect to the azimuthal rotations.

The rod internal pressure is calculated by taking into account the deformations and temperatures of all axial nodes, and is assumed equal throughout the axial length of the rod. Coupled with the one dimensional treatment of the heat equation, this constitutes the so-called $1\frac{1}{2}$ -dimensional model.

2.2.2 Fuel pellet

The fuel pellet is assumed mechanically rigid, so that it has no response to external stresses. The effect of accumulated burn-up to material property correlations is taken into account, where appropriate.

The pellet is assumed to be perfectly cylindrical for the purposes of the solution of the heat equation – specifically, dishing, chamfers or hourglassing is not taken into account. In calculating the axial strain, the mechanical connection between consecutive pellets is assumed to occur at the centerline – a more realistic modeling of the pellet shape remains to be implemented.

FINIX supports externally given fuel swelling, densification and relocation strains. Radial pellet relocation, fuel swelling and densification can also be calculated from correlations.

2.2.3 Cladding

The mechanical model of the cladding is based on the thick cylindrical shell approximation. The model assumes, for example, that the radial differences in stresses and temperatures across the cladding are small. In addition, the model is valid only when the axial curvature of the cladding is small, *i.e.*, when there is very little bowing or bending of the cladding.

The cladding mechanical response is assumed to elasto-plastic. As of version 0.13.9, plastic strains can be externally given to FINIX, and as of version 0.15.12, time-independent plastic deformation is modeled in accordance with the infinitesimal strain theory, and as of version 0.17.12 plastic deformations due to low-temperature creep are modeled. The model is inadequate to model high strain ruptures as only simple failure models are implemented. Ballooning or other large deformations are possibly not correctly modeled, as the implementation of the logarithmic strain framework has not yet been validated. Oxide formation on the cladding is modeled both in steady-state and accidental conditions.

2.2.4 Fill gas and FGR

The amount of fill gas and fractions of individual species can be given as input. Material correlations for helium, argon, krypton, xenon, hydrogen, nitrogen and water vapor are available.

The release of fission gases is modeled. Material changes in the pellet due to accumulating fission products are also taken into account in solid and gaseous fission product swelling, and the evolution of the composition of the fill gas due to fission gas release is modeled.

2.3 Changes from previous versions

Version 1.19.12 incorporates the following changes from version 1.19.1.

A restart functionality has been implemented so that the calculation can be stopped and the complete FINIX state saved to a file. This file can be used to restart the FINIX calculation. The restart functionality is recommended to be used so that the continued simulation is performed with the same number of nodes. However, an interpolation method was also implemented in case the numbers of nodes differ between the restart file and the new simulation.

The logarithmic strain framework has been implemented, and should improve large strain predictions. However, the model remains to be validated. A bug in the cladding creep model was corrected. The previous version did not accumulate creep strain across time steps.

The input files do not have to be located in the running directory anymore, but a specific path to the input files can be specified. An option to specify whether the FINIX input is read in from

the FINIX input files or comes from a host code was implemented for easier coupling between codes.

New general error checking functions were implemented which can be used to check validity of inputs to functions. The error messages produced show the function where the error was produced.

A new option for verbose input reading was implemented. Every parameter that can be input is checked during the input file reading, and now each input value found from the input file is also output. If no value is found for an input value that could be input, a message notifying of this is shown.

Option to "refabricate" a rod was implemented. The refabrication option is meant for use in restarted calculations, where the gas contents of the rod have been changed between base irradiation and transient experiment. The gas contents read in from the restart file are overwritten by the input values in `finix_rod.inp` for the scenario being calculated.

The code is now provided with `CMakeLists.txt` so that the makefile can be generated with `cmake`.

2.4 FINIX publications

In addition to the VTT reports, the following work related to FINIX development has been published:

2020

- Henri Loukusa, Jussi Peltonen, Ville Valtavirta, and Ville Tulkki. Implementation of burnup effects in the multiphysics fuel behavior module FINIX. *Annals of Nuclear Energy*, 136:107022, 2020 Description of FINIX burnup-dependent models for long irradiation periods and related validation efforts. (FINIX-1.19.1)

2019

- V Valtavirta, J Peltonen, U Lauranto, and J Leppänen. SuperFINIX - A flexible-fidelity core level fuel behaviour solver for multi-physics applications. In *NENE2019*, Portoroz, Slovenia, 2019 Description of the SuperFINIX core-level wrapper for FINIX and first results. (FINIX-1.19.1)
- J Peltonen, H Loukusa, and V Tulkki. Validation and verification of the FINIX fuel behavior module. In *TopFuel*, Seattle, WA, USA, 2019 Description of recent FINIX validation and verification. (FINIX-1.19.1)
- Elina Syrjälähti, Timo Ikonen, and Ville Tulkki. Modeling burnup-induced fuel rod deformations and their effect on transient behavior of a VVER-440 reactor core. *Annals of Nuclear Energy*, 125:121–131, 2019 Description of burnup initialization procedure for reactor transient calculations. (FINIX-0.15.6)

2017

- V Valtavirta, J Leppänen, and T Viitanen. Coupled neutronics–fuel behavior calculations in steady state using the Serpent 2 Monte Carlo code. *Annals of Nuclear Energy*, 100:50–64, 2017 (FINIX-0.15.12)

2016

- Timo Ikonen, Elina Syrjälähti, Ville Valtavirta, Henri Loukusa, Jaakko Leppänen, and Ville Tulkki. Multiphysics simulation of fast transients with the FINIX fuel behaviour module. *EPJ Nuclear Sciences & Technologies*, 2:37, 2016 (FINIX-0.15.6)

2015

- E. Syrjälähti, V. Valtavirta, J. Kättö, H. Loukusa, T. Ikonen, J. Leppänen, and V. Tulkki. Multiphysics simulations of fast transients in VVER-1000 and VVER-440 reactors. In *11th International Conference on WWER Fuel Performance, Modelling and Experimental Support, Varna, Bulgaria, 2015*. (FINIX-0.15.6)
- T. Ikonen, E. Syrjälähti, V. Valtavirta, H. Loukusa, J. Leppänen, and V. Tulkki. Multiphysics simulation of fast transients with the FINIX fuel behaviour module. In *TopFuel 2015, Zurich, Switzerland, 2015*. Description of FINIX and applications in coupled-code calculations with Serpent 2, TRAB-1D and TRAB3D/SMABRE. (FINIX-0.13.9)
- T. Ikonen, H. Loukusa, E. Syrjälähti, V. Valtavirta, J. Leppänen, and V. Tulkki. Module for thermomechanical modeling of lwr fuel in multiphysics simulations. *Annals of Nuclear Energy*, 84:111–121, 2015. Description of FINIX and applications in coupled-code calculations with Serpent 2, TRAB-1D and TRAB3D/SMABRE. (FINIX-0.13.9)

2014

- V. Valtavirta, T. Ikonen, T. Viitanen, and J. Leppänen. Simulating fast transients with fuel behavior feedback using the serpent 2 monte carlo code. In *PHYSOR2014, Kyoto, Japan, 2014*. Application of FINIX and Serpent 2 in simulating self-consistently solved temperature and power in a prompt supercritical pin-cell. (FINIX-0.13.9)

2013

- T. Ikonen, V. Tulkki, E. Syrjälähti, V. Valtavirta, and J. Leppänen. FINIX – fuel behavior model and interface for multiphysics applications. In *2013 Fuel Performance Meeting / TopFuel, Charlotte, USA, 2013*. Brief description of the FINIX code, its purpose as a universal fuel behavior module in multiphysics applications and first results. (FINIX-0.13.1)

3. Thermal model

3.1 The heat equation

The conduction of heat is described by the heat equation, where the temperature T in general is a function of time t and all three spacial coordinates. In a cylindrical fuel rod, a convenient choice for the coordinate system are the cylindrical coordinates, r , θ and z . In the present case, however, we make the simplifying assumption that within each axial slice, T has no dependence on z (by assuming the axial heat transfer to be negligible) and no dependence on θ (by assumed symmetry). With these assumptions, the heat equation takes the form

$$C_V(T) \frac{\partial T}{\partial t} - \frac{1}{r} \frac{\partial}{\partial r} \left[\lambda(T) r \frac{\partial T}{\partial r} \right] - s(r) = 0. \quad (3.1)$$

Here C_V is the volumetric heat capacity, λ the thermal conductivity and s the source term (thermal power line density). The temperature is a function of time t and the radial coordinate r , $T = T(r, t)$. The solution of Eq. (3.1) is obtained in the fuel pellet and in the cladding by discretizing the equation with the finite element method (FEM) and by solving the system numerically (see Sec. 7.2.1 for transient heat equation).

The steady-state heat equation simplifies to

$$-\frac{1}{r} \frac{\partial}{\partial r} \left[\lambda[T(r)] r \frac{\partial T}{\partial r} \right] - s(r) = 0. \quad (3.2)$$

The numerical solution of Eq. (3.2) is explained in detail in chapter 7.2.2.

The outer surface of the pellet and the inner surface of the cladding are subject to heat transfer boundary conditions

$$q(R_f) = -\lambda(T(r)) \frac{\partial T}{\partial r} \Big|_{r=R_f} = h[T(R_f) - T(R_{ci})], \quad (3.3)$$

where $q(r)$ is the heat flux, R_f is the fuel outer radius, R_{ci} the cladding inner radius and the notation $|_{r=R_f}$ denotes evaluation of the preceding expression at $r = R_f$. The gap conductance h is calculated using the model described in Sec. 3.3. The gap is assumed to have negligible heat capacity, and thus the conservation of energy implies that the heat flux across the inner surface of the cladding is given by

$$q(R_{ci}) = \frac{R_f}{R_{ci}} q(R_f). \quad (3.4)$$

At the outer surface of the cladding, the boundary condition can be set as constant temperature, constant heat flux or by using a heat transfer coefficient. A more thorough explanation is given in Sec. 7.2.1, where the numerical solution of the heat equation is explained. The remaining boundary condition is the zero heat flux at the inner surface of the pellet (at R_0),

$$q(0) = 0 \Leftrightarrow \frac{\partial T}{\partial r} \Big|_{r=R_0} = 0. \quad (3.5)$$

3.2 Plenum temperature

The model for the plenum gas temperature is derived by assuming that the gas within the plenum is well mixed and described by a single temperature, T_{plen} . The gas exchanges heat with the surrounding walls, whose temperatures are taken as given. Furthermore, it is assumed that the heat capacity of the plenum gas is so small that one can neglect the term with the time derivative of T . One therefore has the steady-state heat equation for the plenum gas,

$$A_p h_p (T_p - T_{\text{plen}}) + A_c h_c (T_c - T_{\text{plen}}) = 0, \quad (3.6)$$

which gives

$$T_{\text{plen}} = \frac{A_p h_p T_p + A_c h_c T_c}{A_p h_p + A_c h_c} \quad (3.7)$$

for the temperature of the plenum. Here A_p (A_c) is the area of the end of the fuel pellet (cladding inner surface) facing the plenum, T_p (T_c) the temperature of the pellet (cladding), and h_p (h_c) the heat transfer coefficient between the pellet (cladding) and the plenum gas. The areas are given by $A_p = \pi R_f^2$ and $A_c = \pi R_{ci}^2 + 2\pi l_p R_{ci}$, where l_p is the (axial) length of the plenum. The temperature of the end of the pellet is calculated as an area-weighted mean temperature (cf. Sec. 7.2). Since the cooling of the end of the pellet due to heat flux into the plenum is not taken into account in the 1D heat equation, this leads to slight over-estimation of the temperature T_p . The temperature of the cladding is assumed to be equal to the coolant temperature, which in turn leads to slight under-estimation of the temperature T_c . The uncertainties introduced in these approximation are not too severe, since the plenum affects the thermo-mechanical solution of the fuel rod only through its coupling to the gap pressure, not affecting the temperatures directly. In addition, solving the surface temperatures accurately would require 2D solution of the heat equations within the plenum, which would lead to undesirable increase in the computational intensity of the model.

The heat transfer coefficients are solved with a similar method as the one used in FRAP-TRAN [23]. For the pellet-plenum heat transfer, the coefficient is

$$h_p = \begin{cases} 0.27 \lambda_{\text{plen}} \frac{(GrPr)^{0.25}}{2R_{ci}}, & \text{for } Gr < 0, \\ 0.54 \lambda_{\text{plen}} \frac{(GrPr)^{0.25}}{2R_{ci}}, & \text{for } 0 \leq Gr \leq 2 \cdot 10^7, \\ 0.14 \lambda_{\text{plen}} \frac{(GrPr)^{0.33}}{2R_{ci}}, & \text{for } Gr > 2 \cdot 10^7, \end{cases} \quad (3.8)$$

where λ_{plen} is the thermal conductivity of the plenum gas, Pr is the Prandtl number and

$$Gr = \frac{g (T_p / T_{\text{plen}} - 1) (2R_{ci})^3}{\nu^2} \quad (3.9)$$

is the Grashof number. In the latter, g is the gravitational acceleration and ν is the kinematic viscosity of the plenum gas.

For the cladding-plenum heat transfer coefficient, the corresponding equations are

$$h_p = \begin{cases} 0.55 \lambda_{\text{plen}} \frac{(GrPr)^{0.25}}{l_p}, & \text{for } Gr \leq 1 \cdot 10^9, \\ 0.021 \lambda_{\text{plen}} \frac{(GrPr)^{0.4}}{l_p}, & \text{for } Gr > 1 \cdot 10^9, \end{cases} \quad (3.10)$$

with

$$Gr = \frac{g (T_c / T_{\text{plen}} - 1) l_p^3}{\nu^2}. \quad (3.11)$$

The correlations for the Prandtl number and the kinematic viscosity are given in Sec. 6.5. The value for the heat conductivity λ_{plen} is not needed, as it is canceled from Eq. (3.7).

3.3 Gas gap conductance

The heat transfer in the gas gap is modelled with Eq. (3.3), with the heat transfer coefficient h given as a sum of three terms:

$$h = h_{\text{cond}} + h_{\text{rad}} + h_{\text{contact}}. \quad (3.12)$$

The first term corresponds to heat conduction across the gap, the second to radiation heat transfer between the pellet surface and the cladding inner surface, and the last one to heat transfer due to solid-solid contact of the pellet and the cladding. When the gas gap remains open, the last term is zero.

3.3.1 Conduction through the gas

The term h_{cond} can be calculated from the heat equation in the gas gap. Assuming the heat capacity of the gas to be small and noting that there is no heat produced in the gap, the equation reduces to $\frac{\partial}{\partial r} \left(\lambda_{\text{gap}}(T) r \frac{\partial T}{\partial r} \right) = 0$. Integrating once with respect to r and applying $\lambda_{\text{gap}} \frac{\partial T}{\partial r} \Big|_{r=R_f} = h_{\text{cond}}(T_{ci} - T_f)$ gives

$$\lambda_{\text{gap}}(T) dT = h_{\text{cond}}(T_{ci} - T_f) \frac{dr}{r}. \quad (3.13)$$

By integrating from T_f to T_{ci} and replacing the temperature-dependent heat conductivity with the average $\bar{\lambda}(T_{ci}, T_f)$ as $\int_{T_f}^{T_{ci}} \lambda_{\text{gap}}(T) dT \equiv \bar{\lambda}(T_{ci}, T_f)(T_{ci} - T_f)$, one obtains

$$h_{\text{cond}} = \frac{\bar{\lambda}(T_{ci}, T_f)}{R_f \ln(1 + d/R_f)}, \quad (3.14)$$

where $d = R_{ci} - R_f$ is the gap width. Since $d \ll R_f$, the term $\ln(1 + d/R_f) \approx d/R_f$, which gives the form used in FRAPTRAN and FRAPCON:

$$h_{\text{cond}} = \frac{\bar{\lambda}(T_{ci}, T_f)}{d}. \quad (3.15)$$

In practice, the average $\bar{\lambda}(T_{ci}, T_f)$ can be approximated by taking the average of the gap temperature, instead of the heat conductivity: $\bar{\lambda}(T_{ci}, T_f) \approx \lambda_{\text{gap}}(T_{\text{gap}})$. The introduced error is of the order of a few percent, at most.

In FRAPTRAN and FRAPCON, an effective gap width d_{eff} is used instead of the bare d . The same approach is taken also here. The effective gap width is given by the FRAPCON correlation [24, 25]

$$d_{\text{eff}} = e^{-0.00125 P_{\text{contact}}(\rho_f + \rho_c)} + 1.8(g_f + g_c) - b + d, \quad (3.16)$$

where P_{contact} is the contact pressure between the pellet and the cladding (in kg/cm², see Sec. 4.3.2), $\rho_f(\rho_c)$ is the surface roughness of the pellet (cladding) in meters, $g_f(g_c)$ is the temperature jump distance (in meters) at the pellet (cladding) surface. The constant $b = 1.397 \cdot 10^{-6}$ m. The sum of the temperature jump distances is calculated from

$$g_f + g_c = A \left(\frac{\lambda \sqrt{T}}{P} \right) \left(\frac{1}{\sum_i a_i f_i M_i^{-1/2}} \right), \quad (3.17)$$

where λ , T and P are the thermal conductivity, temperature and pressure of the gas in units of W/mK, K and Pa, respectively, and a_i , f_i and M_i are the thermal accommodation coefficients,

mole fractions and molecular weights of the gas constituents. The coefficient A is 0.7816 in the FRAPCON-3.4 correlation and 0.0137 in the FRAPCON-4.0 correlation.

FINIX also has an option to use the FRAPTRAN correlation for the effective gap width. In this case, the gap width is given by

$$d_{\text{eff}} = e^{-0.00125P_{\text{contact}}(\rho_f + \rho_c) + 0.0316(g_f + g_c) + d} \quad (\text{optional}). \quad (3.18)$$

The heat transfer coefficient h_{cond} is then given as

$$h_{\text{cond}} = \frac{\lambda_{\text{eff}}}{d_{\text{eff}}}, \quad (3.19)$$

where d_{eff} is given by Eq. (3.16) and λ_{eff} is calculated for the gas mixture of n species as [26]

$$\lambda_{\text{eff}} = \sum_i^n \frac{\lambda_i x_i}{x_i + \sum_j (1 - \delta_{ij}) \Psi_{ij} x_j} \quad (3.20)$$

Here λ_i is the heat conductivity and x_i the mole fraction of the species i , δ_{ij} the Kronecker delta and

$$\Psi_{ij} = \psi_{ij} \left(1 + 2.41 \frac{(M_i - M_j)(M_i - 0.142M_j)}{(M_i + M_j)^2} \right), \quad (3.21)$$

$$\psi_{ij} = \frac{[1 + (\lambda_i/\lambda_j)^{1/2}(M_i/M_j)^{1/4}]^2}{2^{3/2} (1 + M_i/M_j)^{1/2}}, \quad (3.22)$$

where M_i is the molecular weight of the species.

The heat conductivity of the gases (with the exception of steam) is of the form

$$\lambda_i = A_i T^{B_i}, \quad (3.23)$$

with the coefficients given in Table 2.

For the water vapour (steam), the heat conductivity is given by [26]

$$\lambda_{H_2O} = 4.44 \cdot 10^{-6} T^{1.45} + 9.45 \cdot 10^{-5} (2.1668P/T)^{1.3} \quad (\text{for } T > 973.15 \text{ K}), \quad (3.24)$$

$$\begin{aligned} \lambda_{H_2O} = & P/T \left(-2.851 \cdot 10^{-8} + 9.424 \cdot 10^{-10} T - 6.005 \cdot 10^{-14} T^2 \right) \\ & + 1.009 (P/T)^2 / (T - 273.15)^{4.2} + 17.6 \cdot 10^{-3} \\ & + 5.87 \cdot 10^{-5} (T - 273.15) + 1.08 \cdot 10^{-7} (T - 273.15)^2 \\ & - 4.5 \cdot 10^{-11} (T - 273.15)^3 \quad (\text{for } T \leq 973.15 \text{ K}). \end{aligned} \quad (3.25)$$

Here T is in Kelvin and P in Pascal.

Table 2. The gas conductivity constants of Eq. (3.23).

Species	A	B
He	$2.531 \cdot 10^{-3}$	0.7146
Ar	$4.092 \cdot 10^{-4}$	0.6748
Kr	$1.966 \cdot 10^{-4}$	0.7006
Xe	$9.825 \cdot 10^{-5}$	0.7334
H ₂	$1.349 \cdot 10^{-4}$	0.8408
N ₂	$2.984 \cdot 10^{-4}$	0.7799

3.3.2 Radiation across the gap

The radiation heat transfer coefficient is given by the gray body radiation formula

$$h_{\text{rad}} = \frac{\sigma_{SB}}{\frac{1}{\epsilon_f} + \frac{R_f}{R_{ci}} \left(\frac{1}{\epsilon_c} - 1 \right)} \frac{T_f^4 - T_{ci}^4}{T_f - T_{ci}}, \quad (3.26)$$

where σ_{SB} is the Stefan-Boltzmann constant and $\epsilon_f(\epsilon_c)$ is the emissivity of the fuel outer surface (cladding inner surface). The emissivities used in FINIX are

$$\epsilon_f = 0.78557 + 1.5263 \cdot 10^{-5} T_f, \quad (3.27)$$

where the temperature is in Kelvin, and

$$\epsilon_c = 0.809. \quad (3.28)$$

The correlation for ϵ_f is the same as in FRAPTRAN, while the value for ϵ_c is the same as in the FEMAXI code [27]. The latter was chosen over the FRAPTRAN correlation, which requires knowledge of the cladding oxide thickness, which was not present at the time of implementation of the model§. In FRAPTRAN, the value $\epsilon_c = 0.809$ would correspond to an effective oxide thickness of approximately 3.9 microns.

3.3.3 Contact between the pellet and cladding

The contact heat transfer coefficient h_{contact} is given by

$$h_{\text{contact}} = \begin{cases} 13.740 \frac{\lambda_m P_{\text{rel}}^{1/2}}{\rho_f^{-0.528} \sqrt{\rho_f^2 + \rho_c^2}}, & \text{for } P_{\text{rel}} \leq 9 \cdot 10^{-6}, \\ 0.041226 \frac{\lambda_m}{\rho_f^{-0.528} \sqrt{\rho_f^2 + \rho_c^2}}, & \text{for } 9 \cdot 10^{-6} < P_{\text{rel}} \leq 0.003, \\ 4579.5 \frac{\lambda_m P_{\text{rel}}^2}{\rho_f^{-0.528} \sqrt{\rho_f^2 + \rho_c^2}}, & \text{for } 0.003 < P_{\text{rel}} \leq 0.0087, \\ 39.846 \frac{\lambda_m P_{\text{rel}}}{\rho_f^{-0.528} \sqrt{\rho_f^2 + \rho_c^2}}, & \text{for } P_{\text{rel}} > 0.0087, \end{cases} \quad (3.29)$$

where $P_{\text{rel}} = P_{\text{contact}}/H$ is the relative ratio of the contact pressure and Meyer's hardness H_M of the cladding (see Sec. 6.2.4). In addition, $\lambda_m = 2\lambda_f\lambda_c/(\lambda_f + \lambda_c)$ is the geometric mean of the fuel thermal conductivity λ_f and the cladding thermal conductivity λ_c . The correlation of Eq. (3.29) is the same as in FRAPTRAN, with the numerical constants merged into one.

4. Mechanical model

4.1 Internal pressure

The internal pressure P of the rod is calculated from the ideal gas equation of state, $PV = nRT$, where V is the gas volume, n the amount of substance of the gas, R the ideal gas constant and T the temperature. Assuming that pressure differences between the gap, plenum and central hole equalize immediately, the internal pressure is given by

$$P = \frac{nR}{V_{\text{plen}}/T_{\text{plen}} + \sum_k (V_{\text{cent},k}/T_{\text{cent},k} + V_{\text{gap},k}/T_{\text{gap},k})}. \quad (4.1)$$

Here the plenum volume is

$$V_{\text{plen}} = \pi R_{ci}^2 l_p, \quad (4.2)$$

the central hole volume

$$V_{\text{cent},k} = \pi R_{0,k}^2 l_{f,k}, \quad (4.3)$$

and the gap volume

$$V_{\text{gap},k} = \pi l_{f,k} (R_{ci,k}^2 - R_{f,k}^2), \quad (4.4)$$

where k is the index of the axial slice and $l_{f,k}$ is the axial length of the fuel and $R_{0,k}$, $R_{f,k}$ and $R_{ci,k}$ are the pellet inner radius, pellet outer radius and cladding inner radius, respectively, of the k :th axial slice.

4.2 Pellet mechanical model

The fuel pellet is assumed to have an infinite elastic modulus and no stress-induced deformations (the so-called rigid pellet model [23]). Thermal strain and radial relocation, swelling and densification of the fuel are taken into account with correlations. The correlation between the thermal strain and temperature is presented in Sec. 6.1.3, pellet relocation correlation in Sec. 6.1.4, pellet solid and gaseous swelling correlations in Sec. 6.1.5 and pellet densification correlation in Sec. 6.1.6. The radial displacement of the pellet outer surface is calculated by integrating the strains over the pellet radius. The axial strain is calculated using the temperature on the center of the pellet for the thermal strain, where the strain is largest. The details of the thermal strain calculation are explained in Sec. 7.4.1. Swelling and densification strains are also included for the axial strain, but pellet relocation strain is neglected.

4.3 Cladding mechanical model

The mechanical model of the cladding is similar to the FRACAS-I model used in both FRAPCON and FRAPTRAN [23, 24]. The cladding mechanical model is further divided into two distinct situations. The first model considers the case where the gap remains open, and the mechanical equilibrium is determined simultaneously with the calculation of the internal pressure. The second model is invoked when the gap is closed, and the equilibrium is determined by a no-slip condition between the pellet and the cladding.

The calculation of the plastic strain increment proceeds similarly in the open gap and closed gap cases. If the effective stress on the cladding exceeds the yield stress, the time-independent plastic strain calculation is commenced. Otherwise the creep model is used.

The oxidation of the cladding decreases the load-carrying cladding thickness, as the cladding oxide is brittle. The oxidation is taken into account when the oxidation model is on by neglecting the oxidation nodes at the cladding inner and outer surfaces in the mechanical solution. In the formulas below, R_{ci} refers to the metal surface under the cladding inner oxide layer, and R_{co} to the metal surface under the cladding outer oxide layer.

FINIX uses engineering strain as the main strain measure, which has a linear relationship between strain and displacement, but for some strain measures this relationship is not linear. At small strains, all strain measures yield equal results, and the selection of the measure bears no significance [28, 29]. To determine the displacement correctly in the case of large strains, a nonlinear strain-displacement relationship must be used [28, 29]. Strain measures differ in which reference geometry the deformation is defined on: the undeformed or deformed geometry [28]. As the reference geometry in FINIX is the undeformed, as-fabricated geometry of the fuel rod due to the choice of engineering strain, the natural choice is to use the logarithmic strain, which defines the deformation on the undeformed geometry. In FINIX, the strains are typically treated as engineering strain, but logarithmic strain is used to obtain the displacement. The conversions to and from logarithmic strain are performed per the relationship

$$\epsilon_{\log} = \ln(\epsilon + 1). \quad (4.5)$$

In the following, the equations where this conversion is used is specifically noted. The conversion to logarithmic strain is typically done before applying a strain-displacement relationship, and backwards after the calculation has been performed.

4.3.1 Open gap model

First, we consider the open gap mechanical model. In this case, the displacement of the cladding inner surface depends on the internal pressure, which in turn is a function of the gap volume. Therefore, the mechanical equilibrium has to be determined simultaneously with the internal pressure calculation. In practice, the solution has to be found iteratively, since no closed form solution of the full set of equations is known. This numerical procedure is explained in Sec. 7.4.2. For the purposes of deriving the equations, we take the rod internal pressure P as given and fixed.

The cladding model is based on the thin wall approximation, which implies constant stress, strain and temperature across the cladding radial direction. The temperature is taken as the average temperature of the cladding from the solution of the heat equation. In addition, the loading and deformation of the cladding is assumed axisymmetric, and bending strains and stresses are neglected.

Given the internal pressure P , the outside (coolant) pressure P_o , and the cladding inner and outer radii, R_{ci} and R_{co} , the hoop stress σ_θ and the axial stress σ_z are obtained as

$$\sigma_\theta = \frac{R_{ci}P - R_{co}P_o}{R_{co} - R_{ci}}, \quad (4.6)$$

$$\sigma_z = \frac{R_{ci}^2P - R_{co}^2P_o}{R_{co}^2 - R_{ci}^2}. \quad (4.7)$$

When the stresses are known, the effective stress and the effective strain can be calculated in order to find out whether the yield stress has been exceeded. The effective stress is calculated as described in chapter 4.3.3. The effective strain is calculated as a sum of viscoplastic, thermal and elastic strains. If the yield stress has been exceeded, a plastic strain increment is calculated as described in chapter 4.3.3.

The hoop, axial and radial strains are connected to the stresses through relations

$$\epsilon_{\theta} = \frac{1}{E}(\sigma_{\theta} - \nu\sigma_z) + \epsilon_{\theta}^{\text{th}} + \epsilon_{\theta}^{\text{pl}} + d\epsilon_{\theta}^{\text{pl}} + \epsilon_{\theta}^{\text{c}} + d\epsilon_{\theta}^{\text{c}}, \quad (4.8)$$

$$\epsilon_z = \frac{1}{E}(\sigma_z - \nu\sigma_{\theta}) + \epsilon_z^{\text{th}} + \epsilon_z^{\text{pl}} + d\epsilon_z^{\text{pl}} + \epsilon_z^{\text{c}} + d\epsilon_z^{\text{c}}, \quad (4.9)$$

$$\epsilon_r = -\frac{\nu}{E}(\sigma_{\theta} + \sigma_z) + \epsilon_r^{\text{th}} + \epsilon_r^{\text{pl}} + d\epsilon_r^{\text{pl}} + \epsilon_r^{\text{c}} + d\epsilon_r^{\text{c}}, \quad (4.10)$$

Here ϵ_i^{th} are the cladding thermal strains (see Sec. 6.2.3), ϵ_i^{pl} the cladding plastic strains, $d\epsilon_i^{\text{pl}}$ the cladding plastic strain increments (zero if yield stress was not exceeded), ϵ_i^{c} the cladding creep strains, $d\epsilon_i^{\text{c}}$ the cladding creep strain increments, E the Young's modulus (Sec. 6.2.5) and ν the Poisson ratio (Sec. 6.2.6). The strains relate the dimensions of the cladding in the hot state to the dimensions in the cold state. The axial strain is essentially decoupled from the hoop and radial strains, so that the axial length of the slice is

$$l_c = (1 + \epsilon_z)l_{c,\text{cold}}, \quad (4.11)$$

where the subscript k identifying the axial slice has been dropped for convenience [cf. Eq. (4.4)]. The corresponding relations for the cladding inner and outer radii can be derived from the change in the radius of the cladding midplane, $(R_{co} + R_{ci})/2 \equiv \bar{R} = (1 + \epsilon_{\theta})\bar{R}_{\text{cold}}$ and the change in the cladding thickness, $R_{co} - R_{ci} = (1 + \epsilon_r)(R_{co,\text{cold}} - R_{ci,\text{cold}})$. The resulting expressions for R_{ci} and R_{co} are

$$R_{ci} = R_{ci,\text{cold}} \left(1 + \frac{1}{2}\epsilon_{\theta} + \frac{1}{2}\epsilon_r \right) + R_{co,\text{cold}} \left(\frac{1}{2}\epsilon_{\theta} - \frac{1}{2}\epsilon_r \right), \quad (4.12)$$

$$R_{co} = R_{ci,\text{cold}} \left(\frac{1}{2}\epsilon_{\theta} - \frac{1}{2}\epsilon_r \right) + R_{co,\text{cold}} \left(1 + \frac{1}{2}\epsilon_{\theta} + \frac{1}{2}\epsilon_r \right). \quad (4.13)$$

Equations (4.12) and (4.13) relate the radii to their cold-state values and the strains, thus completing the model. If so selected in the source code, the logarithmic strain conversion is done when using these relationships to calculate the radial displacement.

The open gap model of Eqs. (4.6)–(4.13) is solved iteratively. This is because the values of R_{ci} and R_{co} are used to determine the stresses σ_{θ} and σ_z , which in turn are used to solve R_{ci} and R_{co} .

4.3.2 Closed gap model

Strong contact. If the open gap model indicates that the inner surface of the cladding is in contact with the pellet, *i.e.*, $R_{ci} \leq R_f + \rho_f + \rho_c$, then the solution of the mechanical equilibrium proceeds with the closed gap model. For the closed gap model, one uses similar relations to the open gap model, albeit with different boundary conditions. Since the gap is closed, the internal gas pressure cannot be used as a boundary condition. Instead, the contact pressure P_{contact} between the pellet and the cladding remains to be determined. However, the inner radius of the cladding is fixed by the contact, so that

$$R_{ci} = R_f + \rho_f + \rho_c. \quad (4.14)$$

In addition, the axial strain of the cladding, ϵ_z , is determined by the no-slip condition at the pellet-cladding boundary. Any axial strain of the pellet that takes place after the gap has closed is added to the cladding axial strain. The axial strain of the cladding is therefore

$$\epsilon_z = \epsilon_{z,0} + \epsilon_z^{\text{fuel}} - \epsilon_{z,0}^{\text{fuel}}, \quad (4.15)$$

where the additional subscript 0 indicates the strain just prior to gap closing.

For the closed gap, the cladding outer radius can be solved explicitly. If so selected in the source code, the logarithmic strain conversion is done before solving this equation. After some algebraic manipulation, one gets

$$R_{co} = \frac{\frac{1}{\nu}(R_{co,cold} + R_{ci,cold}) - 2R_{co,cold}}{\frac{1}{\nu}(R_{co,cold} + R_{ci,cold}) - 2R_{ci,cold}} R_{ci} + \frac{R_{co,cold}^2 - R_{ci,cold}^2}{\frac{1}{\nu}(R_{co,cold} + R_{ci,cold}) - 2R_{ci,cold}} \left[\frac{1}{\nu} - \epsilon_z + \epsilon_\theta^{th} + \epsilon_\theta^{pl} + d\epsilon_\theta^{pl} + \epsilon_\theta^c + d\epsilon_\theta^c + \epsilon_z^{th} + \epsilon_z^{pl} + d\epsilon_z^{pl} + \epsilon_z^c + d\epsilon_z^c + \left(\frac{1}{\nu} - 1 \right) \left(\epsilon_r^{th} + \epsilon_r^{pl} + d\epsilon_r^{pl} + \epsilon_r^c + d\epsilon_r^c \right) \right] \quad (4.16)$$

$$\left(\frac{1}{\nu} - 1 \right) \left(\epsilon_r^{th} + \epsilon_r^{pl} + d\epsilon_r^{pl} + \epsilon_r^c + d\epsilon_r^c \right) \quad (4.17)$$

From R_{ci} and R_{co} one can then solve the hoop and radial strains using

$$\epsilon_\theta = \frac{R_{co} + R_{ci}}{R_{co,cold} + R_{ci,cold}} - 1, \quad (4.18)$$

$$\epsilon_r = \frac{R_{co} - R_{ci}}{R_{co,cold} - R_{ci,cold}} - 1, \quad (4.19)$$

Equations (4.18) and (4.19) are equivalent with Eqs. (4.8) and (4.10). The strains then give the stresses as

$$\sigma_\theta = \frac{E}{1 + \nu} \left(\left(\epsilon_\theta - \epsilon_\theta^{pl} - d\epsilon_\theta^{pl} + \epsilon_\theta^c + d\epsilon_\theta^c \right) - \left(\epsilon_r - \epsilon_r^{pl} - d\epsilon_r^{pl} + \epsilon_r^c + d\epsilon_r^c \right) \right), \quad (4.20)$$

$$\sigma_z = \nu \sigma_\theta + E \left(\epsilon_z - \epsilon_z^{th} - \epsilon_z^{pl} - d\epsilon_z^{pl} + \epsilon_z^c + d\epsilon_z^c \right), \quad (4.21)$$

Finally, the contact pressure is calculated from

$$P_{\text{contact}} = \frac{\sigma_\theta (R_{co} - R_{ci}) + P_o R_{co}}{R_{ci}}. \quad (4.22)$$

If the effective stress (calculated as described in chapter 4.3.3) exceeds the yield stress, a plastic strain increment is calculated as described in chapter 4.3.3. After the plastic strain increment has been calculated, the total strains are calculated from Eqs. (4.18) and (4.19), the stresses and contact pressure updated and R_{co} calculated.

Weak contact. If the solution of the closed gap model with the strong contact assumptions gives an interfacial pressure that is lower than the internal gas pressure, $P_{\text{contact}} < P$, then the gas can push the cladding and the pellet slightly apart, allowing them to slide against each other. In this situation, the no-slip condition in the axial direction no longer holds. Instead, the axial strain of the cladding adjusts until the contact pressure equals the internal rod pressure, and the cladding again becomes axially locked with the pellet. In the weak contact case, no more plastic deformation is calculated.

For the weak contact case, the contact pressure $P_{\text{contact}} = P$. The cladding inner radius R_{ci} is given as for the strong contact case. The outer radius R_{co} can be solved from the implicit

equation,

$$\begin{aligned}
 & R_{co}^3 \frac{1}{R_{co,cold} - R_{ci,cold}} + R_{co}^2 \left(-\frac{R_{ci}}{R_{co,cold} - R_{ci,cold}} - 1 - \epsilon_r^{th} - \epsilon_r^{pl} - d\epsilon_r^{pl} + \epsilon_r^c + d\epsilon_r^c + \frac{2\nu}{E} P_o \right) \\
 & + R_{co} \left[-\frac{R_{ci}^2}{R_{co,cold} - R_{ci,cold}} - \frac{\nu}{E} (P_o - P) \right] \\
 & + \frac{R_{ci}^3}{R_{co,cold} - R_{ci,cold}} + \left(1 + \epsilon_r^{th} + \epsilon_r^{pl} + d\epsilon_r^{pl} + \epsilon_r^c + d\epsilon_r^c + \frac{2\nu}{E} P \right) R_{ci}^2 = 0, \tag{4.23}
 \end{aligned}$$

using Newton-Raphson iteration [30]. If so selected in the source code, the logarithmic strain conversion is done before solving this equation. The stresses and strains can then be solved from

$$\sigma_\theta = \frac{R_{ci} P_{contact} - R_{co} P_o}{R_{co} - R_{ci}}, \tag{4.24}$$

$$\sigma_z = \frac{R_{ci}^2 P_{contact} - R_{co}^2 P_o}{R_{co}^2 - R_{ci}^2}. \tag{4.25}$$

$$\epsilon_\theta = \frac{1}{E} (\sigma_\theta - \nu \sigma_z) + \epsilon_\theta^{th} + \epsilon_\theta^{pl} + d\epsilon_\theta^{pl} + \epsilon_\theta^c + d\epsilon_\theta^c, \tag{4.26}$$

$$\epsilon_z = \frac{1}{E} (\sigma_z - \nu \sigma_\theta) + \epsilon_z^{th} + \epsilon_z^{pl} + d\epsilon_z^{pl} + \epsilon_z^c + d\epsilon_z^c, \tag{4.27}$$

$$\epsilon_r = -\frac{\nu}{E} (\sigma_\theta + \sigma_z) + \epsilon_r^{th} + \epsilon_r^{pl} + d\epsilon_r^{pl} + \epsilon_r^c + d\epsilon_r^c. \tag{4.28}$$

In the closed gap model, since the inner radius R_{ci} is fixed by the boundary condition, there is no need to iterate the solution with the solution of the internal pressure P (irrespective of the strength of the contact).

4.3.3 Plastic deformation model

Time-independent plasticity of the cladding is modeled in FINIX, and only infinitesimal strains are covered by the approach. The calculations of the increment in plastic strain proceed similarly in both the open gap and the closed gap, strong contact cases. The following assumptions are made in FINIX concerning cladding plasticity:

- The cladding behaves isotropically,
- the cladding is incompressible,
- the cladding follows the strain hardening hypothesis,
- the cladding yields according to the von Mises yield criterion,
- the cladding follows the Prandtl-Reuss flow rule.

The following discussion and definitions are based on Refs. [31] and [32]. The onset of yielding can be determined from a yield function. When the yield function has a value below zero, the cladding behaves elastically. If the yield function has a value greater than zero, cladding plastic deformation occurs so that the yield function returns to zero. Positive values of the yield function are unphysical. In FINIX, the von Mises yield function is used. The von Mises yield function, f_{VM} , for a strain-hardening material is as follows:

$$f_{VM} = \sigma_e - \sigma_Y(\epsilon_e). \tag{4.29}$$

The so-called yield surface is determined by $f_{VM} = 0$. In Eq. (4.29), σ_Y is the yield stress, which is a function of effective strain, and σ_e is the (von Mises) effective stress. The calculation of the yield stress is discussed in chapter 6.2.9. The effective stress (and analogously, strain) is defined as

$$\sigma_e = \sqrt{\frac{3}{2} \sum_i s_i s_i}, \quad (4.30)$$

$$\epsilon_e = \sqrt{\frac{2}{3} \sum_i e_i e_i}, \quad (4.31)$$

where s_i is the deviatoric stress and e_i the deviatoric strain, or

$$s_i = \sigma_i - \frac{Tr(\sigma)}{3} = \sigma_i - p_{hs}, \quad (4.32)$$

$$e_i = \epsilon_i - \frac{Tr(\epsilon)}{3}. \quad (4.33)$$

Here, $Tr(\sigma)$ is the trace of the (Cauchy) stress tensor σ , or the sum of the principal values of the tensor. The second term in the deviatoric stress equation is therefore equivalent to the hydrostatic pressure, p_{hs} . Note that in this discussion, only the principal values of stresses and strains are used, so $\sigma_i = \sigma_{ii}$, for example.

Using Hooke's law (see Eqs. (4.8), (4.9) and (4.10)) and the definition of deviatoric stress and strain, the following relation between the deviatoric stress and strain can be found:

$$s_i = 2 \left(\frac{E}{2(1+\nu)} \right) e_i = 2G e_i, \quad (4.34)$$

where we have defined the shear modulus, G . For effective stress and effective strain, a similar relationship can be found using their definitions and Eq. (4.34):

$$\sigma_e = 3G \epsilon_e. \quad (4.35)$$

When stresses resulting in an effective stress greater than the yield stress is calculated to occur with the elastic solution, FINIX solves the increment in plastic strain by the radial return algorithm in order to return the stresses to the yield surface determined by Eq. (4.29). The radial return algorithm is an implicit integration algorithm, so all values of stress and strain refer to stresses and strains at the end of the time step. The effective stress in Eq. (4.29) can be decomposed into an effective trial stress and an effective plastic strain increment, $d\epsilon_e^{pl}$, that returns the stress to the yield surface (see Ref. [32] for the derivation):

$$\sigma_e = \sigma_{e,trial} - 3G d\epsilon_e^{pl}. \quad (4.36)$$

The effective trial stress can be calculated from the trial stress with Eq. (4.30). The trial stress components are calculated using the following equation (see Ref. [32] for the derivation):

$$\sigma_{trial,i} = \sigma_i + 2G d\epsilon_i^{pl}. \quad (4.37)$$

The plastic strain increment components, $d\epsilon_i^{pl}$, are obtained from the Prandtl-Reuss flow rule, Eq. (4.39). Substituting Eq. (4.36) into Eq. (4.29), we can now solve for the effective plastic strain increment that satisfies the yield function:

$$\sigma_{e,trial} - 3Gd\epsilon_e^{pl} - \sigma_Y(\epsilon_e^{pl}) = 0. \quad (4.38)$$

This is called the radial return method, which is widely used in solving the plasticity equations. The effective plastic strain increment is found from Eq. (4.38) by Newton-Raphson iteration. An initial guess of $1 \cdot 10^{-15}$ is used in FINIX. When the effective plastic strain increment is found, the individual components of the plastic strain increment are obtained from the Prandtl-Reuss flow rule:

$$d\epsilon_i^{pl} = \frac{3}{2} \frac{s_i}{\sigma_e} d\epsilon_e^{pl}. \quad (4.39)$$

With the knowledge of the plastic strain increment components, one may calculate the stresses at the end of the time step by solving for σ_i in Eq. (4.37). The new stresses and plastic strain increments are then used to calculate the total strains from Hooke's law. In the open gap case, both the inner and outer radius of the cladding are updated with the new strains, but in the closed gap, strong contact case only the cladding outer radius is updated.

The increments in plastic strain are only added to the plastic strain after the solution for the time step has converged. It must be noted that the effective plastic strain must be treated separately from the plastic strains as the effective plastic strain depends on the past deformation and does not decrease. The effective plastic strain is a sum of all the effective plastic strain increments that have occurred in the past.

4.3.4 Creep model

The creep model of Limbäck and Andersson [33] or a model by Geelhood [34] can be used in FINIX. The models divide creep into primary and secondary creep, and are valid at normal operating temperatures. High-temperature creep is not modeled. In turn, secondary creep is further divided into thermal and irradiation creep. However, the implemented models differ in the selection of the driving force for primary creep, and their treatment of the time on the creep curve (see Eq. (4.40)). The differences are described in detail below.

Primary creep is assumed to be dependent on the secondary creep strain rate, $\dot{\epsilon}_s$. Using the conditions of the current time step, the secondary creep strain rate (see Sec. 6.2.8) is calculated. With this secondary strain rate, the saturated primary creep strain, ϵ_p^{sat} , (see Sec. 6.2.7) is calculated. The primary creep strain is assumed to follow a function

$$\epsilon_{c,p} = \epsilon_p^{sat} (1 - \exp(f(\dot{\epsilon}_s)t)), \quad (4.40)$$

for which the function $f(\dot{\epsilon}_s)$ is described in Sec. 6.2.7.

The primary creep strain increment during a time step is calculated by

$$\Delta\epsilon_{c,p} = a (\epsilon_{c,p}(t_1) - \epsilon_{c,p}(t_0)), \quad (4.41)$$

where a is the direction of creep.

The secondary creep strain increment is calculated from the sum of the thermal and irradiation creep rates as

$$\Delta\epsilon_{c,s} = b (\dot{\epsilon}_{c,s,th} + \dot{\epsilon}_{c,s,irr}) dt, \quad (4.42)$$

where the time step is given in hours and b is the direction of creep. The increments in creep strain are added to the total creep strain only after the solution has converged at the end of a time step.

Limbäck-Andersson model The time and driving forces in the Limbäck-Andersson model as implemented in FINIX are as follows: If the creep curve changes, the virtual time on the creep curve is adjusted according to the strain hardening rule. A new virtual time at the new creep curve, corresponding to the same creep strain at the previous creep curve, is found by the bisection method. From the values of $\epsilon_{c,p}$ calculated with the conditions at the previous and the current time step, the limits of a bisection iteration are set: If the creep strain $\epsilon_{c,p}$ is larger at this time, the new virtual time can be found between 0 and t . If the creep strain $\epsilon_{c,p}$ is smaller at this time, the new virtual time must be greater than t . If the time calculated in this case becomes very large, primary creep strain has saturated. When the hoop stress is tensile, tensile virtual time is advanced and compressive virtual time is decreased, and vice versa when the hoop stress is compressive. Neither time is allowed to decrease below zero. The driving force for both the primary and secondary creep is σ_e , and the direction for both is same:

$$a = b = \frac{|\sigma_h|}{\sigma_e}. \quad (4.43)$$

Geelhood model The time and driving forces in the Geelhood model are as follows [34]: The time t input into Eq. (4.40) is initialized to zero after each significant stress change. When other conditions than effective stress change, the strain hardening rule is applied as described above for the original Limbäck-Andersson model. The significance limit is set at 5 MPa. The direction of primary creep is the direction of the last effective stress (see Eq. (4.30)) change, and the direction of secondary creep is the direction of the hoop stress. The directions a and b in the previous equations are then

$$a = \frac{|\Delta\sigma_e|}{\Delta\sigma_e}, \text{ and} \quad (4.44)$$

$$b = \frac{|\sigma_h|}{\sigma_h}, \quad (4.45)$$

where $\Delta\sigma_e$ is the last significant effective stress change and σ_h the hoop stress. In calculating the secondary creep rate for the primary creep curve, $\Delta\sigma_e$ is used instead of σ_e as in the original Limbäck-Andersson model. For secondary creep, σ_h is used, as in the original model.

4.3.5 Failure models

FINIX includes two simple criteria [35] for rod failure: the plastic instability criterion and the overstrain criterion. The plastic instability criterion predicts rod failure when true effective strain of the cladding exceeds 0.02 and simultaneously the true effective strain rate exceeds 0.0278 s^{-1} (100 h^{-1}). The overstrain criterion predicts rod failure when the true effective strain exceeds 0.5. The plastic instability criterion seems to be conservative, but improves the numerical stability of FINIX in cases of high strains. For numerical stability, the strain rate input into the plastic instability criterion at any given time step is calculated as a weighted average between current and previous strain rates.

5. Other models

5.1 Fission gas release

5.1.1 Intragranular gas behaviour

The intragranular behavior of the fission gases – diffusion in the uranium oxide lattice, formation of intragranular bubbles and the growth and diffusion of the bubbles – is described by the Forsberg-Massih model [36]. The four-term approximation of the integration kernel reported by Hermansson and Massih [37] is used. Resolution from the grain boundary due to irradiation is ignored in the implementation in FINIX.

The Forsberg-Massih model approximately solves the differential equation

$$\frac{\partial C(r, t)}{\partial t} = D_{\text{eff}}(t) \Delta_r C(r, t) + \beta(t) \quad (5.1)$$

with boundary conditions

$$C(r, 0) = C(a, t) = 0, \quad (5.2)$$

where C is the concentration of the gas, β the fission gas production rate (see Sec. 6.4.1) and D_{eff} the effective diffusion coefficient.

The effective diffusion coefficient is calculated from the gas atom diffusion coefficient in trap-free media and the bubble diffusion coefficient as

$$D_{\text{eff}} = \frac{b}{b+g} D_{\text{atom}} + \frac{g}{b+g} D_{\text{bubble}}, \quad (5.3)$$

where b is the rate of irradiation-induced resolution of gas atoms, g the trapping rate of gas atoms into intragranular bubbles, D_{atom} is the gas atom diffusion coefficient (see Sec. 6.4.2) and D_{bubble} is the bubble diffusion coefficient. The bubble diffusion coefficient is as reported by van Uffelen et al. [38]:

$$D_{\text{bubble}} = \frac{3}{4} \frac{V_{\text{gas}} D_{\text{vol}}}{\pi R_b^3}, \quad (5.4)$$

where V_{gas} is the gas atom volume, $8.4756 \cdot 10^{-29} \text{ m}^3$, R_b the intragranular bubble radius (see Sec. 6.4.3) and D_{vol} the volume diffusion coefficient (see Sec. 6.4.4).

The rate of irradiation-induced bubble resolution is calculated as [39]

$$b = 3.03 \pi l \dot{F} (R_b + d)^2, \quad (5.5)$$

where l is the fission fragment range, $6 \mu\text{m}$, \dot{F} the fission rate, R_b the intragranular bubble radius and d the fission fragment damage radius, 1 nm , perpendicular to the fragment propagation line.

The trapping rate of gas atoms into bubbles is calculated as [39]:

$$g = \frac{4 \pi R_b N_b D_{\text{atom}}}{N_A}, \quad (5.6)$$

where R_b is the intragranular bubble radius (see Sec. 6.4.3), N_b the intragranular bubble number density (see Sec. 6.4.5), D_{atom} is the gas atom diffusion coefficient (see Sec. 6.4.2) and N_A is the Avogadro constant.

The approximate solution to Eq. (5.1) is described in detail by Forsberg and Massih [36] and Hermansson and Massih [37]. The solution method is otherwise as reported by the authors, except for the different grain surface boundary condition as discussed before and the introduction of a grain boundary sweeping model from FRAPCON, where a portion of the gas is removed from the grain to the grain boundary due to grain growth. The following correction factor is applied to the intragranular gas concentration as calculated by the Forsberg-Massih algorithm:

$$f_s = \frac{a_i^3 - a_{i-1}^3}{a_{i-1}^3}, \quad (5.7)$$

where a_i is the new grain size and a_{i-1} the grain size on the previous time step.

5.1.2 Intergranular gas behaviour and release

Because resolution of gas from the grain boundary is neglected in FINIX, the intra- and intergranular models can be solved independently. The grain boundary gas concentration from the intragranular model is used by the intergranular model as is, and the evolution of grain boundary bubbles is computed.

Bubble size and density are determined by the influx of gas atoms to the boundary from inside the grain, the diffusion of vacancies at the grain boundary and the coalescence of grain boundary bubbles.

Bubble growth due to gas atom and vacancy absorption and emission is calculated using the flux of gas atoms from the intragranular model and assuming a van der Waals equation of state for the gas in the bubble. The vacancy absorption rate with these assumptions is

$$\frac{dN_v}{dt} = \frac{2\pi D_v \delta_g}{kTS} (p - p_{eq}), \quad (5.8)$$

where N_v is the number of vacancies in a grain boundary bubble, D_v the grain boundary vacancy diffusion coefficient (see Sec. 6.4.6), δ_g the diffusion layer thickness, set as $5 \cdot 10^{-10}$ m [40], k the Boltzmann constant, T the temperature and S a function of fractional coverage, F_c , defined as

$$S = \frac{-(3 - F_c)(1 - F_c) - 2 \ln F_c}{4}. \quad (5.9)$$

The pressure of the bubble is calculated as

$$p = \frac{kT N_g}{\omega_v N_v}, \quad (5.10)$$

where N_g is the number of gas atoms in a grain boundary bubble and ω_v the volume of an atomic vacancy in the bubble, set as $4.09 \cdot 10^{-29}$ m³.

The equilibrium pressure p_{eq} is calculated with the radius of the grain boundary bubbles, R_{gb} , and the surface tension of the bubbles, γ ,

$$p_{eq} = \frac{2\gamma}{R_{gb}}. \quad (5.11)$$

The radius is calculated from the volume of the bubbles, given by

$$V_{gb} = \omega_g N_g + \omega_v N_v, \quad (5.12)$$

where ω_g is the volume of a gas atom. The volume of the bubbles is related to the radius of the bubbles by

$$R_{gb} = \frac{3V}{4\pi\phi}, \quad (5.13)$$

where ϕ is a factor relating the volume of a lenticular bubble to that of a sphere, given as a function of the bubble dihedral angle θ , and the relation is as follows

$$\phi = 1 - \frac{3}{2} \cos \theta + \frac{1}{2} \cos^3 \theta. \quad (5.14)$$

Solving Eqs. (5.8), (5.10), (5.11) and 5.12) gives the number of vacancies. With the number of gas atoms, the change in bubble area due to absorption of gas atoms as vacancies, $dA_{gb,g}$ is obtained. The change in bubble density due to the coalescence of bubbles is given by

$$\frac{dN_{gb}}{dt} = -\frac{6N_{gb}^2}{3 + 4N_{gb}A_{gb}} \left(\frac{dA_{gb,g}}{dt} \right). \quad (5.15)$$

In the coalescence of bubbles, the total number of gas atoms and vacancies in the grain boundary bubbles is assumed to be conserved. The numbers of gas atoms and vacancies per bubble must then be updated as the number of bubbles has changed.

Finally, the bubble volume is calculated through Eq. (5.12) which is used to calculate the area of bubbles. These are used to calculate the fractional coverage, F_c , with the bubble number density:

$$F_c = N_{gb}A_{gb}. \quad (5.16)$$

If the value of F_c exceeds the threshold fractional coverage of 0.5, the excess gas is released so that $F_c = 0.5$ by adjusting the bubble area. In the gas release, the ratio of gas atoms to vacancies in bubbles is maintained.

The fill gas composition is updated after release. The released gases are thought to consist only of xenon and krypton. A Xe/Kr ratio of 6.275 is assumed, which is an average value for the fission of ^{235}U .

5.2 Coolant

FINIX has an implementation of a thermal-hydraulic model for calculation of heat transfer coefficients from the cladding to the coolant, which requires also the calculation of several thermal properties of coolant. The axial temperature distribution of the coolant is also calculated. The coolant inlet temperature and pressure must be given by the user. The density and heat capacity models only work within a temperature region 1, defined in [41]. Therefore it is advisable to use the user-given heat transfer coefficient for BWR calculations.

5.2.1 Axial coolant temperature rise model

The coolant temperature model is based on the method used in FRAPCON-4.0 [25], and is based on the following formula:

$$T_{\text{cool}}(z) = T_{\text{in}} + \int_0^z \frac{\pi D_0 q''(z) dz}{C_p A_f G}, \quad (5.17)$$

where $T_{\text{cool}}(z)$ is the coolant temperature at elevation z , T_{in} the inlet coolant temperature, D_0 the cladding outer diameter, $q''(z)$ the rod surface heat flux at elevation z , C_p the coolant isobaric heat capacity, G the coolant mass flux and A_f the coolant channel flow area.

By splitting the integral to correspond to axial nodes, in which the heat flux remains constant, the coolant temperature can be solved iteratively for each node. This formula can be rewritten as:

$$T_{\text{cool}}(n) = T_{\text{cool}}(n-1) + \frac{\pi D_0 q''(n) \Delta z}{C_p A_f G} \quad (5.18)$$

5.2.2 Cladding-coolant heat transfer coefficient

FINIX 1.19.1 uses a similar method to calculate the heat transfer coefficient than FRAPTRAN-2.0 [42]. The method solves the coolant-cladding heat transfer coefficient by using a Dittus-Boelter film conductance h_{DB} for the forced convection of when the coolant flow is turbulent, and a modified version of that correlation for the laminar flow [43]. The nucleate boiling of the cladding is accounted by another coefficient, h_{NB} . The total coolant-cladding heat transfer coefficient h_{cc} is then the sum of these two coefficients:

$$h_{cc} = h_{DB} + h_{NB} \quad (5.19)$$

The calculation of the Dittus-Boelter heat transfer coefficient requires the knowledge of the coolant thermal conductivity, dynamic viscosity and density. Here we work under assumption that the coolant is pure water. Therefore the heat transfer coefficient model itself isn't valid for scenarios in which coolant is a non-water substance, such as the CABRI REP-Na RIA tests, where sodium was used as the coolant [44]. However, in those scenarios the temperature of the coolant and the outer surface of the cladding are nearly equal due to more efficient heat transfer between the cladding and metallic coolant.

The sum of the coefficients h_{DB} and h_{NB} yields good results for in pressurized water reactors, but in boiling water conditions they result in low heat transfer coefficients. This in turn causes the temperature difference between the cladding and the coolant to be unacceptably high. This isn't particularly surprising, due to the coolant material properties being calculated in the temperature region 1, which isn't suitable for the BWR coolant. In case of the BWRs this can be solved by either using the traditional user-given heat transfer coefficient or other boundary condition.

The correlations used to calculate the heat transfer coefficients and the various material properties of water are described in detail in section 6.7.

5.3 Cladding oxidation

5.3.0.1 Inner surface

A simple model was implemented for the cladding inner surface oxidation: when the pellet-cladding contact occurs, the cladding inner surface oxide thickness is immediately updated to 10 μm . This value is based on data in [45].

5.3.0.2 Outer surface

The oxidation of zirconium is typically divided into pre-transition growth period where the growth rate is cubic, and a post-transition growth period where the growth rate is approximately linear. The pre-transition oxide growth is calculated by Dyce correlation reported in [46]:

$$\frac{\Delta s}{\Delta t} = \frac{A}{s^2} \cdot \exp\left(-\frac{Q}{RT_w}\right), \quad (5.20)$$

where A is an empirical constant having a value of $2.72 \cdot 10^{-13} \text{ m}^3 \cdot \text{s}^{-1}$, s is the thickness of the oxide layer, Q is an activation energy having a value of $135188 \text{ J} \cdot \text{mol}^{-1}$ and R is the universal

gas constant. T is the temperature at the coolant-oxide interface, defined as

$$T = T_w + \phi \frac{S}{k_{co}} \quad (5.21)$$

where T_w is the temperature at the metal-oxide interface, ϕ is the heat flux at the axial node and k_{co} is the thermal conductivity of the oxide.

The transition thickness of the oxide layer can be calculated using the MATPRO correlation [47], or simply estimated as a constant value (such as 2–3 μm). After this oxide thickness has been reached, the post-transition growth begins. The MATPRO correlation [47] has been implemented as:

$$s_{trans} = 7.749 \cdot \exp\left(-\frac{790}{T}\right). \quad (5.22)$$

By integrating the Dyce correlation, we can get the transition time:

$$t_{trans} = \left(\frac{s_{trans}^3 - s^3}{3}\right) \frac{\exp\left(\frac{Q}{RT_w}\right)}{A} \quad (5.23)$$

The post-transition growth of the oxidation layer for a timestep in steady state scenarios is calculated using the correlation by Kättö [48]:

$$\frac{\Delta s}{\Delta t} = nK \cdot \exp\left(-\frac{E}{RT_w}\right) t^{n-1}, \quad (5.24)$$

where n is an exponent with a value of 1.025, K is a pre-exponential factor with a value of $2.74 \cdot 10^{-7} \text{ m} \cdot \text{s}^{-1}$, E/R is the activation temperature with a value of 8645.4 K and t is the simulation time at that timestep.

Post-transition growth of the oxide can also be calculated using the FRAPTRAN-2.0 correlation for the transient scenarios. This is based on the more traditional Baker-Just correlation [49, 42]:

$$s = \sqrt{s_0^2 + A' \cdot \exp\left(-\frac{Q'}{RT}\right) dt}, \quad (5.25)$$

where s_0 is the oxide width at the beginning of the simulation, A' is the pre-exponential factor with a value of $1.126 \cdot 10^{-6} \text{ m}^2 \cdot \text{s}^{-1}$, Q' is the activation energy with a value of $1.502 \cdot 10^5 \text{ J}$ and dt is the length of the timestep.

For VVER cladding, MTA EK best-estimate E110 correlation by Király et al [50] is used:

$$s = 2.967 \cdot \exp\left(\frac{-10103.0}{T}\right) t^{\frac{1}{2}} \quad (5.26)$$

The heat generation rate in the oxidation reaction is calculated as:

$$q''' = \frac{H \Delta n}{V dt} = \frac{H \Delta V \rho}{M_{ZrO_2} V dt} \quad (5.27)$$

where H is the reaction enthalpy, having a value of $1088 \text{ kJ} \cdot \text{mol}^{-1}$, Δn is the difference in the amount of oxide, V is the volume of the whole oxide node, ΔV is the volume difference of the oxide layer between timesteps and M_{ZrO_2} the molar mass of zirconium dioxide, which has a value of $0.123218 \text{ kg} \cdot \text{mol}^{-1}$. The volume and its differential is calculated assuming both initial and final oxide layers to be hollow cylinders.

Zirconium dioxide has a larger molar volume than zirconium metal, which is depicted by the Pilling-Bedworth ratio of zirconium and zirconium dioxide:

$$R_{PB} = \frac{V_{\text{oxide}}}{V_{\text{metal}}} = 1.56. \quad (5.28)$$

Using this ratio, the decrease in cladding thickness can be calculated. In FINIX, the following formula is used for the outer oxide layer:

$$s_{\text{metal}} = \frac{\sqrt{s^2 R_{PB} + 2s R_{PB} r_{mo} + R_{PB}^2 r_{mo}^2} - r_{mo} R_{PB}}{R_{PB}}, \quad (5.29)$$

where s_{metal} is the thickness of the metal layer corresponding to the outer oxide layer of thickness s and r_{mo} is the radius of the metal-oxide interface, and similarly for the inner oxide layer:

$$s_{\text{metal}} = \frac{r_{mo} R_{PB} - \sqrt{s^2 R_{PB} - 2s R_{PB} r_{mo} + R_{PB}^2 r_{mo}^2}}{R_{PB}}, \quad (5.30)$$

5.4 Grain growth

The grain growth model in FINIX is that of Ainscough [51]. Grain growth is assumed to be dependent only on the temperature, and the grain size (diameter) after a time step dt , a_1 is calculated from the grain size (diameter) at the previous time step, a_0 , as

$$a_1 = a_0 + kdt \left(\frac{1}{a_0} - \frac{1}{a_{\text{lim}}} \right) \quad (5.31)$$

where k is the rate constant of grain growth in $m^2 \cdot s^{-1}$, given as

$$k = 1.456 \cdot 10^{-8} \exp \left(\frac{-2.674 \cdot 10^5}{RT} \right) \quad (5.32)$$

and a_{lim} the limiting grain size in meters, given as

$$a_{\text{lim}} = 2.23 \cdot 10^{-3} \exp \left(\frac{-7620}{T} \right) \quad (5.33)$$

The bracketed term in Eq. (5.31) is not allowed to be negative, so grain size can only increase.

The grain size (diameter) calculated by the model is the linear intercept average grain size (diameter), d_{LI} . The value used elsewhere in FINIX is the average grain size, d_{ave} , for which the following relationship has been established [52]:

$$d_{\text{ave}} = 1.558 d_{LI}. \quad (5.34)$$

Conversions between these values are done within the grain growth function.

5.5 Radial power distribution

The Transuranus Burnup Equations (TUBRNP) radial power distribution model by Lassmann *et al.* [53] is implemented in FINIX. The TUBRNP model does not calculate the correct radial power distribution in gadolinia-doped fuel.

The radial power distribution model calculates the boundary condition for the time step. Therefore a call to the function `finix_steady_state_radial_power_distribution()` is required before a call to the main FINIX solver function.

Table 3. Parameters of the radial form function, as in [25].

	LWR	HWR
p_1	3.45	2.21
p_2	3.00	3.00
p_3	0.45	0.45

The local concentrations of actinides are obtained from the following group of differential equations:

$$\frac{N_{U-235}(r)}{dB} = -\sigma_{a,U-235} N_{U-235}(r)A, \quad (5.35)$$

$$\frac{N_{U-238}(r)}{dB} = -\sigma_{a,U-238} N_{ave,U-238} f(r)A, \quad (5.36)$$

$$\frac{N_{Pu-239}(r)}{dB} = -\sigma_{a,Pu-239} N_{Pu-239}(r)A + \sigma_{c,U-238} N_{ave,U-238} f(r)A, \text{ and} \quad (5.37)$$

$$\frac{N_{Pu-j}(r)}{dB} = -\sigma_{a,Pu-j} N_{Pu-j}(r)A + \sigma_{c,Pu-(j-1)} N_{Pu-(j-1)}(r)A, \quad (5.38)$$

where $\sigma_{a,i}$ are absorption cross sections of nuclide i , j is from 240 to 242, $\sigma_{c,i}$ the capture cross sections of nuclide i , B the burnup and A is a normalization factor defined as

$$A = 0.8815 \frac{\rho_{fuel}}{\alpha \sum_k \sigma_{f,k} N_{k,ave}} \quad (5.39)$$

and $f(r)$ a radial shape function defined as

$$f(r) = 1 + p_1 \exp(-p_2 (r_{fo} - r)^{p_3}), \quad (5.40)$$

where r_{fo} is the pellet radius and p_i empirical constants. The radii are input in millimeters. The cross sections used by FINIX are reported in chapter 6.6. The radial shape function is normalized so that

$$2 \frac{\int_{r_{fi}}^{r_{fo}} f(r) r dr}{r_{fo}^2 - r_{fi}^2} = 1, \quad (5.41)$$

and the parameters p_i are given separately for light and heavy water reactors as defined in table 3. The integral in Eq. (5.41) is calculated with Simpson's rule.

The local non-normalized power density is calculated from

$$q'''_{non-norm.}(r) = \sum_k \sigma_{f,k} N_k \phi, \quad (5.42)$$

where ϕ is the neutron flux. Solutions from diffusion theory are used to calculate a non-normalized value of ϕ through the following proportionality:

$$\phi(r) \propto \begin{cases} I_0(\kappa r) & \text{for a solid cylinder, and} \\ I_0(\kappa r) + K_0(\kappa r) \frac{I_1(\kappa r_{f0})}{K_1(\kappa r_{f0})} & \text{for a hollow cylinder,} \end{cases} \quad (5.43)$$

where I_i are the modified Bessel functions of the first kind of order i , and K_i the modified Bessel functions of the second kind of order i . The modified Bessel functions are calculated as per [30]. κ is the inverse diffusion length, defined as

$$\kappa = \sqrt{3\Sigma_s \sum_k \sigma_{a,k} N_{ave,k}}, \quad (5.44)$$

where Σ_s is the macroscopic scattering cross section and $\sigma_{a,k}$ the absorption cross sections of nuclide k . To obtain normalized power factors, the volume-average non-normalized power is calculated with $q'''_{non-norm,r}$, and the power factors calculated by $\frac{q'''_{non-norm,r}}{q'''_{ave}}$.

5.6 Fast neutron flux and fluence

The fast neutron flux and fluence are calculated from the power density of the fuel with a simple correlation, which yields an approximate value. Fast neutron flux is calculated as

$$\phi = a_\phi \frac{P_{ave}}{\rho_{\%td} \rho_{td}}, \quad (5.45)$$

where P_{ave} is the average power density over the pellet in the axial node, $\rho_{\%td}$ the fractional density, ρ_{td} the theoretical density of the fuel and a_ϕ the factor relating the specific power to fast neutron flux. A default value of $2.29 \cdot 10^{19}$ is used for a_ϕ . Fast neutron fluence is calculated from the flux as

$$\Phi = \phi dt, \quad (5.46)$$

where dt is the time step.

6. Material correlations

The material correlations used in FINIX have been mostly adopted from the MATPRO library [26] and the FRAPTRAN fuel performance code [23]. VVER correlations have mostly been adopted from Shestopalov *et al.* [54]. Typically, the correlations are taken 'as is'. Since they have been thoroughly tested within other fuel performance codes, detailed study of their applicability was not considered necessary at this point. Their properties are discussed in, *e.g.*, Refs. [26, 55].

6.1 Fuel properties

6.1.1 Specific heat

The fuel specific heat correlation is the same as in FRAPTRAN and MATPRO [23, 26]. The specific heat c_m is given as

$$c_m = K_1 \frac{\Theta^2 \exp(\Theta/T)}{T^2 [\exp(\Theta/T) - 1]^2} + K_2 T + K_3 \frac{Y E_d}{2RT^2} \exp(-E_d/RT), \quad (6.1)$$

where T is the temperature in Kelvin, R is the ideal gas constant (≈ 8.314 J/molK) and Y is the oxygen-to-metal ratio. For UO_2 , the numerical values of the constants are $K_1 = 296.7$ J/kgK, $K_2 = 0.0243$ J/kgK², $K_3 = 8.745 \cdot 10^7$ J/kg, $\Theta = 535.285$ K and $E_d = 1.577 \cdot 10^5$ J/mol.

6.1.2 Thermal conductivity

The FRAPTRAN correlation (see Ref. [26]) is used for the fuel thermal conductivity. The thermal conductivity λ is given by

$$\lambda = 1.0789 \frac{d}{1 + 0.5(1 - d)} \lambda_{95}, \quad (6.2)$$

where λ_{95} is the thermal conductivity for UO_2 at 95 % of the theoretical density, and d is the as-fabricated density of pellet as a fraction from the theoretical value. The correlation for λ_{95} is

$$\lambda_{95} = \left[A + a \cdot gad + BT + f(Bu) + \left(1 - 0.9e^{-0.04Bu} \right) g(Bu)h(T) \right]^{-1} + \frac{E}{T^2} e^{-F/T}. \quad (6.3)$$

Here gad is the gadolinia weight fraction, Bu is the burnup (GWd/MTU) and T is the temperature (K). The functions introduced in Eq. (6.3) are $f(Bu) = 0.00187Bu$, $g(Bu) = 0.038Bu^{0.28}$ and $h(T) = [1 + 396 \exp(-Q/T)]^{-1}$, with $Q = 6380$ K. The constants in Eq. (6.3) are $A = 0.0452$ mK/W, $a = 1.1599$, $B = 2.46 \cdot 10^{-4}$ m/W, $E = 3.5 \cdot 10^9$ WK/m and $F = 16361$ K.

6.1.3 Thermal strain

The thermal expansion correlation for UO_2 is taken from MATPRO/FRAPTRAN. The correlation is valid in the solid phase (below $T \approx 3110$) of the fuel pellet, and gives zero strain at 300 K. The (linear) thermal strain is

$$\epsilon_{th} = K_1 T - K_2 + K_3 e^{-E_d/k_B T}, \quad (6.4)$$

where k_B is the Boltzmann constant, $K_1 = 9.8 \cdot 10^{-6}$ 1/K, $K_2 = 2.94 \cdot 10^{-3}$, $K_3 = 0.316$ and $E_d = 1.32 \cdot 10^{-19}$ J. Ref. [26] reports the value $K_2 = 2.61 \cdot 10^{-3}$ for the second constant. However, it is easy to verify that $\epsilon_{th}(T = 300 \text{ K}) \approx 0$ is given by $K_2 = 2.94 \cdot 10^{-3}$.

6.1.4 Radial pellet relocation

Cracking and radial relocation of the fuel pellet due to irradiation and thermal stresses is modeled using the FRAPCON correlation for the pellet radial cracking. Relocation is given as the fractional closure of the gap in relation of the as-fabricated gap G as

$$\Delta G/G = \begin{cases} 0.3 + 0.1f(Bu), & \text{for } LHR < 20 \text{ kW/m,} \\ 0.28 + g(LHR) + [0.12 + g(LHR)]f(Bu), & \text{for } 20 \text{ kW/m} < LHR < 40 \text{ kW/m,} \\ 0.32 + 0.18f(Bu), & \text{for } LHR > 40 \text{ kW/m,} \end{cases} \quad (6.5)$$

where $f(Bu) = Bu/5$ for $Bu < 5 \text{ MWd/kgU}$ and $f(Bu) = 1$ otherwise, and $g(LHR) = 0.0025(LHR - 20)$, with the linear hear rate LHR given in kW/m and the burnup Bu in MWd/kgU.

Half of the radial relocation is considered permanent (hard) and half recoverable (soft). In FINIX, the total (soft + hard) relocation is only taken into account in determining the gap conductance, while only the hard relocation is considered in the mechanical model and in determining the reduction in free volume for the pressure calculation.

6.1.5 Pellet swelling

Solid swelling Swelling of the fuel matrix due to solid fission product accumulation is modeled with the MATPRO/FRAPCON correlation. Linear swelling strain is calculated as

$$\frac{\Delta l}{l} = \begin{cases} 2.067 \cdot 10^{-4} Bu, & \text{for } Bu < 80 \text{ MWd} \cdot \text{kgU}^{-1}, \\ 2.867 \cdot 10^{-4} Bu, & \text{for } Bu \geq 80 \text{ MWd} \cdot \text{kgU}^{-1}, \end{cases} \quad (6.6)$$

Gaseous swelling For gaseous swelling, two correlations can be used: The FRAPCON correlation or if the fission gas release model is being used, the Pastore et al. gaseous swelling model.

The temperature-dependence of the linear gaseous swelling strain from the FRAPCON correlation is as follows:

$$\left(\frac{\Delta l}{l}\right)_T = \begin{cases} 4.55 \cdot 10^{-5} T - 4.37 \cdot 10^{-2}, & \text{for } 960^\circ\text{C} \leq T \leq 1370^\circ\text{C,} \\ -4.05 \cdot 10^{-5} T + 7.40 \cdot 10^{-2}, & \text{for } 1370^\circ\text{C} < T \leq 1832^\circ\text{C.} \end{cases} \quad (6.7)$$

The value of $\frac{\Delta l}{l}$ also depends on burnup as follows:

$$\frac{\Delta l}{l} = \begin{cases} 0.0, & \text{for } Bu < 40 \text{ MWd} \cdot \text{kgU}^{-1}, \\ \left(\frac{\Delta l}{l}\right)_T \frac{Bu-40.0}{10.0}, & \text{for } 40 \leq Bu \leq 50 \text{ MWd} \cdot \text{kgU}^{-1}, \\ \left(\frac{\Delta l}{l}\right)_T, & \text{for } Bu > 50 \text{ MWd} \cdot \text{kgU}^{-1}, \end{cases} \quad (6.8)$$

The Pastore et al. gaseous swelling model [40] requires some parameters calculated by the fission gas release model, so it can only be used when the fission gas release model is on. The linear intragranular gaseous swelling is calculated as

$$\frac{\Delta l}{l} = \frac{4}{9} \pi N_{ig} R_{ig}^3, \quad (6.9)$$

where N_{ig} is the intragranular bubble number density, calculated using a correlation by White and Tucker [56]

$$N_{ig} = \frac{1.52 \cdot 10^{27}}{T} - 3.3 \cdot 10^{23}, \quad (6.10)$$

and R_{ig} the intragranular bubble radius, which is calculated with a correlation by Massih and Forsberg [39]

$$R_{ig} = 1.453 \cdot 10^{-10} \exp\left(1.023 \cdot 10^{-3} T\right). \quad (6.11)$$

The linear intergranular gaseous swelling strain in the Pastore et al. model is calculated from

$$\frac{\Delta l}{l} = \frac{1}{6} N_{gf} R_{SAV} \frac{4}{3} \pi \phi R_{gf}^3, \quad (6.12)$$

where N_{gf} is the grain face bubble number density, R_{SAV} the surface-area-to-volume ratio, ϕ a geometric factor relating the volume of a lenticular bubble to that of a sphere (see Eq. (5.14)), and R_{gf} is the radius of a grain face bubble. N_{gf} and R_{gf} are calculated by the fission gas release model.

6.1.6 Pellet densification

The densification is calculated from a model from MATPRO. The model is based on measured resintering density change, but in FINIX a default value for the resintering density change is used, $98.82 \text{ kg}\cdot\text{m}^{-3}$. First, the maximum possible density change is calculated as

$$\left(\frac{\Delta l}{l}\right)_{\max} = \begin{cases} -0.148230, & \text{for } T < 1000 \text{ K,} \\ -0.281637, & \text{for } T \geq 1000 \text{ K,} \end{cases} \quad (6.13)$$

and then the linear densification strain is calculated from

$$\frac{\Delta l}{l} = \left(\frac{\Delta l}{l}\right)_{\max} + \exp(-3.0 (\text{Bu} + c)) + 2.0 \exp(-35.0 (\text{Bu} + c)). \quad (6.14)$$

The parameter c is iterated to yield a densification strain of zero for fresh fuel.

6.2 Cladding properties

6.2.1 Specific heat

Zircaloy cladding specific heat (in $\text{J}\cdot\text{kg}^{-1}\text{K}^{-1}$) is given as a function of temperature in a tabulated form in Table 4. The data is adopted from Ref. [26]. Between 300 K and 1248 K, the temperature is calculated by linear interpolation from the values of Table 4, while for temperatures lower than 300 K and higher than 1248 K, the specific heat is assumed constant.

For Zr1%Nb cladding, the data is presented similarly in a tabulated form in Table 5. The data is adopted from Ref. [54]. Between 393 K and 1400 K, the temperature is calculated by linear interpolation from the values of Table 5, while for temperatures lower than 393 K and higher than 1400 K, the specific heat is assumed constant.

Table 4. The specific heat capacity of the Zircaloy cladding [26]. The specific heat is assumed to have a constant value both below $T = 300$ K and above $T = 1248$ K.

Temperature (K)	Specific heat ($\text{J}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$)
300	281
400	302
640	331
1090	375
1093	502
1113	590
1133	615
1153	719
1173	816
1193	770
1213	619
1233	469
1248	356

Table 5. The specific heat capacity of the Zr1%Nb cladding [54]. The specific heat is assumed to have a constant value both below $T = 393$ K and above $T = 1400$ K.

Temperature (K)	Specific heat ($\text{J}\cdot\text{kg}^{-1}\cdot\text{K}^{-1}$)
393	345
473	360
573	370
673	380
773	383
873	385
883	448
973	680
1025	816
1073	770
1153	400
1173	392
1200	392
1300	393
1400	393

6.2.2 Thermal conductivity

The thermal conductivity of Zircaloy below 2098 K is given by the FRAPTRAN correlation

$$\lambda = 7.51 + 2.09 \cdot 10^{-2} T - 1.45 \cdot 10^{-5} T^2 + 7.67 \cdot 10^{-9} T^3, \quad (6.15)$$

where the temperature T is given in Kelvin and the conductivity λ in $\text{W}\cdot\text{m}^{-1}\text{K}^{-1}$ [26].

According to Shestopalov *et al.* [54], the thermal conductivity of Zr1%Nb cladding below 2133 K is given by

$$\lambda = 15.0636e^{0.4618 \cdot 10^{-3} T}. \quad (6.16)$$

For both Zircaloy cladding at temperatures higher than 2098 K and Zr1%Nb cladding at temperatures higher than 2133 K, the conductivity (in $\text{W}\cdot\text{m}^{-1}\text{K}^{-1}$) is

$$\lambda = 36.0. \quad (6.17)$$

6.2.3 Thermal strain

The correlation for Zircaloy cladding thermal strain is based on the correlation used in FRAPTRAN. The FRAPTRAN formulae give non-zero strain at $T = 300$ K, which we assume be the cold reference state of the system. Thus, the constant term is adjusted to give zero strain at 300 K. The difference w.r.t. the FRAP correlations is less than 10^{-4} .

The correlation is given separately for the axial and diametral strains (the strain is assumed to be isotropic on the plane perpendicular to the axial direction, hence the diametral strain is used for both the radial and hoop thermal strain). The correlation is given separately for the α and β phases of zirconium and interpolated in the intermediate regime.

For $T \leq 1073$ K, the strains are

$$\epsilon_{\text{axial}}^{\alpha} = -1.192 \cdot 10^{-4} + (T - 273.15) \cdot 4.441 \cdot 10^{-6}, \quad (6.18)$$

$$\epsilon_{\text{diametral}}^{\alpha} = -1.80459 \cdot 10^{-4} + (T - 273.15) \cdot 6.721 \cdot 10^{-6}, \quad (6.19)$$

and for $T \geq 1273$ K

$$\epsilon_{\text{axial}}^{\beta} = -8.3942 \cdot 10^{-3} + (T - 273.15) \cdot 9.70 \cdot 10^{-6}, \quad (6.20)$$

$$\epsilon_{\text{diametral}}^{\beta} = -6.7432 \cdot 10^{-3} + (T - 273.15) \cdot 9.70 \cdot 10^{-6}. \quad (6.21)$$

For the intermediate temperatures the strains are interpolated from the α and β phase values so that for $1073 \text{ K} < T < 1273 \text{ K}$

$$\epsilon_{\text{axial}}^{\alpha-\beta} = \frac{1273\text{K} - T}{200\text{K}} \epsilon_{\text{axial}}^{\alpha} + \frac{T - 1073\text{K}}{200\text{K}} \epsilon_{\text{axial}}^{\beta}, \quad (6.22)$$

$$\epsilon_{\text{diametral}}^{\alpha-\beta} = \frac{1273\text{K} - T}{200\text{K}} \epsilon_{\text{diametral}}^{\alpha} + \frac{T - 1073\text{K}}{200\text{K}} \epsilon_{\text{diametral}}^{\beta}. \quad (6.23)$$

The correlations for Zr1%Nb cladding thermal strains are provided by Shestopalov *et al.* [54]. The correlations give zero thermal strain at 300 K. In the correlations below, $\Delta T = (T - 883.0)$. The correlations are given for five temperature intervals. Below 573 K, the strains are given by

$$\epsilon_{\text{axial}} = -0.128 \cdot 10^{-2} + 3.859 \cdot 10^{-6} T + 0.134 \cdot 10^{-8} T^2, \quad (6.24)$$

$$\epsilon_{\text{diametral}} = -0.200 \cdot 10^{-2} + 5.654 \cdot 10^{-6} T + 0.334 \cdot 10^{-8} T^2, \quad (6.25)$$

at temperatures above 573 K but below 883 K by

$$\epsilon_{\text{axial}} = 0.137 \cdot 10^{-2} + 5.4 \cdot 10^{-6} (T - 573.0), \quad (6.26)$$

$$\epsilon_{\text{diametral}} = 0.334 \cdot 10^{-8} T^2 + 5.654 \cdot 10^{-6} T - 0.200 \cdot 10^{-2}, \quad (6.27)$$

at temperatures above 883 K but below 1153 K by

$$\epsilon_{\text{axial}} = 3.047 \cdot 10^{-3} + 2.312 \cdot 10^{-8} \Delta T - 7.358 \cdot 10^{-8} \Delta T^2 + 1.721 \cdot 10^{-10} \Delta T^3, \quad (6.28)$$

$$\epsilon_{\text{diametral}} = 5.598 \cdot 10^{-3} + 2.312 \cdot 10^{-8} \Delta T - 7.358 \cdot 10^{-8} \Delta T^2 + 1.721 \cdot 10^{-10} \Delta T^3, \quad (6.29)$$

at temperatures above 1153 K but below 2133 K by

$$\epsilon_{\text{axial}} = 1.0765 \cdot 10^{-3} + 9.7 \cdot 10^{-6} (T - 1153.0), \quad (6.30)$$

$$\epsilon_{\text{diametral}} = 3.628 \cdot 10^{-3} + 9.7 \cdot 10^{-6} (T - 1153.0), \quad (6.31)$$

and finally at temperatures higher than 2133 K by

$$\epsilon_{\text{axial}} = 1.0582 \cdot 10^{-2}, \quad (6.32)$$

$$\epsilon_{\text{diametral}} = 1.313 \cdot 10^{-2}. \quad (6.33)$$

6.2.4 Meyer's hardness

Meyer's hardness H_M is used in the gap conductance model to determine the magnitude of the contact heat transfer coefficient. The following correlation is used for H_M of Zircaloy cladding (note the error in sign of last term in [26]):

$$H_M = \exp \left[26.034 + T(-0.026394 + T(4.3502 \cdot 10^{-5} - 2.5621 \cdot 10^{-8} T)) \right], \quad (6.34)$$

where the dimension of H_M is ($\text{N} \cdot \text{m}^{-2}$). In addition, the lower limit of H_M of Zircaloy is set as $1.94 \cdot 10^8 \text{ N} \cdot \text{m}^{-2}$.

Meyer's hardness for Zr1%Nb cladding is given by Shestopalov *et al.* [54] for temperatures below 800 K as

$$H_M = 2172.1 \cdot 10^6 - 10.7055 \cdot 10^6 T + 27650 T^2 - 32.78 T^3 + 0.01423 T^4, \quad (6.35)$$

and for temperatures over 800 K as

$$H_M = \exp \left(26.034 - 2.6394 \cdot 10^{-2} T + 4.3502 \cdot 10^{-5} T^2 - 2.5621 \cdot 10^{-8} T^3 \right), \quad (6.36)$$

where the dimensions of H_M is ($\text{N} \cdot \text{m}^{-2}$). Additionally, a minimum hardness of $1 \cdot 10^5 \text{ N} \cdot \text{m}^{-2}$ is specified for Zr1%Nb cladding.

6.2.5 Young's modulus

The Young's modulus E of Zircaloy cladding is given separately for the α and β phases [26]. Below 1094 K, the correlation used is

$$E^\alpha = (1.088 \cdot 10^{11} - 5.475 \cdot 10^7 T + K_1 + K_2)/K_3, \quad (6.37)$$

with T in Kelvin and E in $\text{N}\cdot\text{m}^{-2}$. The coefficient K_1 , K_2 and K_3 are calculated from

$$K_1 = (6.61 \cdot 10^{11} + 5.912 \cdot 10^8 T)\Delta, \quad (6.38)$$

$$K_2 = -2.6 \cdot 10^{10} C, \quad (6.39)$$

$$K_3 = 0.88 + 0.12e^{-\Phi/10^{25}}. \quad (6.40)$$

Here Δ is the average oxygen concentration minus the oxygen concentration of as-received cladding (kg oxygen/kg Zircaloy), C is the cold work (dimensionless ratio of areas) and Φ is the fast neutron fluence ($\text{n}\cdot\text{m}^{-2}$).

From 1239 K upwards the correlation is

$$E^\beta = 9.21 \cdot 10^{10} - 4.05 \cdot 10^7 T. \quad (6.41)$$

In the intermediate range ($1094 \text{ K} < T < 1239 \text{ K}$), the value is interpolated as

$$E^{\alpha-\beta} = \frac{1239\text{K} - T}{(1239 - 1094)\text{K}} E^\alpha + \frac{T - 1094\text{K}}{(1239 - 1094)\text{K}} E^\beta. \quad (6.42)$$

The Young's modulus correlation for Zr1%Nb cladding is as in FRAPTRAN-1.4 [26]. At temperatures below 1073 K, it is given by

$$E = 1.21 \cdot 10^{11} - 6.438 \cdot 10^7 T + 3.021 \cdot 10^{12} x_O, \quad (6.43)$$

where x_O is the mass fraction of oxygen in the cladding. At temperatures above 1073 K, the Young's modulus is given by

$$E = 9.129 \cdot 10^{10} - 4.5 \cdot 10^7 T. \quad (6.44)$$

A minimum of 1 Pa is specified.

6.2.6 Poisson's ratio

The correlation for the Poisson's ratio ν is taken from FRAPTRAN [26]:

$$\nu = 0.42628 - 5.556 \cdot 10^{-5} T. \quad (6.45)$$

The temperature T is given in Kelvin, and ν is dimensionless. The same correlation is used for both Zircaloy and Zr1%Nb claddings.

6.2.7 Primary creep

Primary creep in the model used in FINIX is assumed to be dependent on the secondary creep strain rate. The correlation for saturated primary creep strain is calculated as [33]:

$$\epsilon_p^{\text{sat}} = 0.0216 \dot{\epsilon}_s (2 - \tanh(35500 \dot{\epsilon}_s))^{-2.05}, \quad (6.46)$$

where $\dot{\epsilon}_s$ is the secondary creep strain rate. Primary creep strain at time t is then calculated from

$$\epsilon_p = \epsilon_p^{\text{sat}} \left(1 - \exp\left(-52 \sqrt{\dot{\epsilon}_s t}\right) \right), \quad (6.47)$$

Table 6. Parameters of the Limbäck-Andersson creep rate correlation for stress relief annealed (SRA) and recrystallized annealed (RXA) claddings. Some parameters are the same for both types of cladding. Φ is the fast neutron fluence.

Parameter	SRA	RXA	Validity
A ($\text{K}\cdot\text{MPa}^{-1}\text{h}^{-1}$)	$1.08 \cdot 10^9$	$5.47 \cdot 10^8$	
E (MPa)	$1.149 - 59.9T$		
a_i (MPa^{-1})	$650.0 (1.0 - 0.56 (1.0 - \exp(-1.4 \cdot 10^{-27} \Phi^{1.3})))$		
n	2.0	3.5	
Q ($\text{J}\cdot\text{mol}^{-1}$)	201000.0		
C_0	$4.10 \cdot 10^{-24}$	$1.87 \cdot 10^{-24}$	
C_1	0.85		
C_2	1.0		
f	0.7283	0.7994	$T < 570 \text{ K}$
	$-7.0237 + 0.0136T$	$-3.18562 + 0.00699132T$	$570 \leq T < 625 \text{ K}$
	1.4763	1.184	$T \geq 625 \text{ K}$

6.2.8 Secondary creep

In the creep model used in FINIX, secondary creep is subdivided into thermal and irradiation creep. The thermal creep rate is given by [33]:

$$\dot{\epsilon}_{th} = \frac{AE}{T} \left(\sinh \frac{a_i \sigma_{eff}}{E} \right)^n \exp \left(\frac{-Q}{RT} \right), \quad (6.48)$$

where A , E , a_i , n and Q are parameters of the creep equation. The irradiation creep rate is given by

$$\dot{\epsilon}_{irr} = C_0 \phi^{C_1} \sigma_{eff}^{C_2} f, \quad (6.49)$$

where C_i and f are parameters of the creep correlation and ϕ is the fast neutron flux.

The parameters of the model are those reported by PNNL for stress relief annealed (SRA) and recrystallized annealed (RXA) Zircaloy claddings. The selection of the parameters is made based on the user-given cold work parameter: if it is nonzero, SRA parameters are used, and RXA otherwise. The parameters are reported in table 6. There are no creep correlation parameters for Zr1%Nb cladding implemented in FINIX.

6.2.9 Yield stress

The PNNL stress-strain correlation [57] is used in the calculation of the yield stress for Zircaloy and Zr1%Nb claddings. Parameters for the correlation are reported by Geelhood *et al.* [57] for Zircaloy and by Shestopalov *et al.* [54] for Zr1%Nb cladding. The yield stress, σ_Y , correlation has the following form:

$$\sigma_Y = K \epsilon^n \left(\frac{\dot{\epsilon}}{10^{-3}} \right)^m \quad (6.50)$$

where K , n and m are the strength coefficient, strain hardening exponent and strain rate exponent, parameters fitted to experimental data, and $\dot{\epsilon}$ the strain rate. To find the yield stress after an amount of plastic strain has been accumulated, one must take into account that the yield stress then occurs at the intersection of the yield stress curve, described by Eq. (6.50), and the elastic stress-strain line:

$$\sigma = E (\epsilon - \epsilon^{pl}) \quad (6.51)$$

First setting $\sigma = \sigma_Y$ and solving for ϵ in Eq. (6.51) and then substituting ϵ in Eq. (6.50) yields an implicit equation for the yield stress, so the yield stress must be solved iteratively. A value of 34.5 MPa is used as an initial guess. Currently, a fixed value of $1 \cdot 10^{-5}$ is used for the strain rate when determining whether the yield stress has been exceeded and a plastic strain increment should be calculated. However, the strain rate is reported to have a small effect on the calculated yield stress [57].

The strain, ϵ , in the correlation is logarithmic strain, and the yield stress is logarithmic stress. In FINIX, engineering stress and strain are used, so before using this correlation the values of stress and strain are converted into logarithmic stress and strain. The relation between logarithmic and engineering (or nominal) strain is as in Eq. (4.5) and the relationship for stress is

$$\sigma_{\log} = \sigma (\epsilon + 1). \quad (6.52)$$

A major difference between engineering and logarithmic strain is that true strain is additive, whereas engineering strain is not. However, with very small strains, the error from using engineering strains additively is small. In the plastic strain calculation, the calculation of the plastic strain increment is done using logarithmic strain and stress, and the results are returned to FINIX as engineering strain and stress. Also, care is taken to only compare logarithmic yield stress with the logarithmic effective stress or engineering yield stress with the engineering effective stress.

Strength coefficient. For Zircaloy, the strength coefficient, K , in Eq. (6.50), is subdivided into terms that are functions of temperature of the cladding, T , fast neutron fluence, Φ , cold work, CW , and cladding type. The strength coefficient for Zircaloy is calculated as follows:

$$K = \frac{K_T (1 + K_{CW} + K_\Phi)}{K_{clad}}. \quad (6.53)$$

K_{clad} depends on the type of Zircaloy, and for Zircaloy-2 it is 1.305 and for Zircaloy-4 it is 1.0. The temperature-dependent term K_T is reported for four temperature intervals. At temperatures below 750 K, it is given by

$$K_T = 1.176 \cdot 10^9 + 4.549 \cdot 10^5 T - 3.282 \cdot 10^3 T^2 + 1.728 T^3, \quad (6.54)$$

at temperatures over 750 K but below 1090 K by

$$K_T = 2.523 \cdot 10^6 \exp\left(\frac{2.850 \cdot 10^6}{T^2}\right), \quad (6.55)$$

at temperatures over 1090 K but below 1255 K by

$$K_T = 1.841 \cdot 10^8 - 1.435 \cdot 10^5 T, \quad (6.56)$$

and finally at temperatures over 1255 K but below 2100 K by

$$K_T = 4.33 \cdot 10^7 - 6.685 \cdot 10^4 T + 37.579 T^2 - 7.330 \cdot 10^{-3} T^3. \quad (6.57)$$

The fast neutron fluence dependent term is reported for three fluence intervals. At fluences below $0.1 \cdot 10^{25} \frac{n}{m^2}$ it is given by

$$K_\Phi = \left(-0.1464 + 1.464 \cdot 10^{-25} \Phi\right) \cdot \left(2.25 e^{-20 CW} \min\left(1, e^{\frac{T-550}{10}}\right) + 1\right), \quad (6.58)$$

at fluences above $0.1 \cdot 10^{25} \frac{n}{m^2}$ but below $2 \cdot 10^{25} \frac{n}{m^2}$ by

$$K_\Phi = 2.928 \cdot 10^{-26} \Phi, \quad (6.59)$$

and finally for fluences above $2 \cdot 10^{25} \frac{n}{m^2}$ but below $12 \cdot 10^{25} \frac{n}{m^2}$ by

$$K_\Phi = 0.532 + 2.662 \cdot 10^{-27} \Phi. \quad (6.60)$$

The cold work dependent term is simply

$$K_{CW} = 0.546 CW, \text{ when } 0 < CW < 0.75. \quad (6.61)$$

For Zr1%Nb cladding, the strength coefficient is reported separately for irradiated and unirradiated cladding. In FINIX, if the fast neutron fluence is greater than zero, the irradiated cladding correlation is used. For unirradiated cladding, the strength coefficient is given for two temperature intervals. For the temperature range $293K < T < 797.9K$ the strength coefficient is

$$K = 898.371 \cdot 10^6 - 1.912 \cdot 10^6 + 2.025 \cdot 10^3 T^2 - 0.963 T^3, \quad (6.62)$$

and for the temperature range $797.9K < T < 1223K$ it is

$$K = 1.518 \cdot 10^{10} e^{-0.00560 T}. \quad (6.63)$$

For irradiated cladding, the strength coefficient of Zr1%Nb is reported for three temperature intervals. For temperatures above 293 K but below 763 K, it is given by

$$K = 916.855 \cdot 10^6 - 0.605 \cdot 10^6 T - 247.482 T^2, \quad (6.64)$$

at temperatures above 763 K but below 859.4 K by

$$K = 4.913 \cdot 10^{11} e^{-0.00965 T}, \quad (6.65)$$

and at temperatures above 859.4 K but below 1223 K by

$$K = 1.518 \cdot 10^{10} e^{-0.00561 T}. \quad (6.66)$$

Strain hardening exponent. For Zircaloy, the strain hardening exponent, n , in Eq. (6.50), is subdivided into terms that are functions of temperature, fast neutron fluence and the cladding type. The strain hardening exponent is calculated as follows for Zircaloy:

$$n = \frac{n_T n_\phi}{n_{clad}}. \quad (6.67)$$

n_{clad} depends on the type of Zircaloy, and for Zircaloy-2 it is 1.6 and for Zircaloy-4 it is 1.0. The temperature dependent term, n_T , is given for four temperature intervals. At temperatures below 419.4 K it is

$$n_T = 0.11405. \quad (6.68)$$

At temperatures above 419.4 K but below 1099.0772 K, it is given by

$$n_T = -9.490 \cdot 10^{-2} + 1.165 \cdot 10^{-3} T - 1.992 \cdot 10^{-6} T^2 + 9.588 \cdot 10^{-10} T^3, \quad (6.69)$$

and at temperatures above 1099.0772 K but below 1600 K by

$$n_T = -0.227 + 2.5 \cdot 10^{-4} T. \quad (6.70)$$

At higher temperatures than 1600 K, it is

$$n_T = 0.173. \quad (6.71)$$

The fast neutron fluence dependent term, n_ϕ , is given for three fluence intervals. For fluence below $0.1 \cdot 10^{25} \frac{n}{m^2}$, the term is given by

$$n_\phi = 1.321 + 0.48 \cdot 10^{-25} \phi, \quad (6.72)$$

while above $0.1 \cdot 10^{25} \frac{n}{m^2}$ but below $2 \cdot 10^{25} \frac{n}{m^2}$ it is given by

$$n_\phi = 1.369 + 0.096 \cdot 10^{-25} \phi, \quad (6.73)$$

and above $2 \cdot 10^{25} \frac{n}{m^2}$ but below $7.5 \cdot 10^{25} \frac{n}{m^2}$ by

$$n_\phi = 1.544 + 0.00873 \cdot 10^{-25} \phi. \quad (6.74)$$

At fluences higher than $7.5 \cdot 10^{25} \frac{n}{m^2}$, the fluence dependent term is

$$n_\phi = 1.609. \quad (6.75)$$

For Zr1%Nb cladding, the strain hardening exponent is reported separately for irradiated and unirradiated cladding. In FINIX, if the fast neutron fluence is greater than zero, the irradiated cladding correlation is used. For unirradiated cladding, the strain hardening exponent is given by:

$$n = 0.0463 + 0.000198 T - 3.315 \cdot 10^{-7} T^2 + 1.391 \cdot 10^{-10} T^3. \quad (6.76)$$

The lower limit for validity of the unirradiated cladding correlation is 293 K, and the upper limit is 1223 K. For irradiated cladding the strain hardening exponent is given for three temperature intervals. At temperatures above 293 K but over 759 K, it is given by

$$n = -0.126 + 0.00135 T - 3.537 \cdot 10^{-6} T^2 + 3.735 \cdot 10^{-9} T^3, \quad (6.77)$$

at temperatures above 759 K but below 879 K by

$$n = -0.240 + 0.00284 T - 8.226 \cdot 10^{-6} T^2 + 9.277 \cdot 10^{-9} T^3 - 3.588 \cdot 10^{-12} T, \quad (6.78)$$

and at temperatures above 879 K but below 1223 K by

$$n = 0.0463 + 0.000198 T - 3.315 \cdot 10^{-7} T^2 + 1.391 \cdot 10^{-10} T^3. \quad (6.79)$$

Strain rate exponent. For both Zircaloy and Zr1%Nb claddings, the strain rate exponent, m , in Eq. (6.50) is only dependent on the temperature of the cladding. For Zircaloy, the strain rate exponent is given for three temperature intervals. Below 750 K, it is

$$m = 0.015. \quad (6.80)$$

Between 750 K and 800 K, the strain rate exponent is given by

$$m = -0.544 + 7.458 \cdot 10^{-4} T, \quad (6.81)$$

and at temperatures over 800 K, it is given by

$$m = -0.207 - 3.241 \cdot 10^{-4} T. \quad (6.82)$$

For Zr1%Nb cladding the strain rate exponent is reported for three temperature intervals. Below 752.2 K but over 293 K it is given by

$$m = 0.0228 - 3.448 \cdot 10^{-7} T, \quad (6.83)$$

at temperatures over 752.2 K but below 902.1 K by

$$m = -2.535 + 0.00663T - 5.303 \cdot 10^{-6} T^2 + 1.347 \cdot 10^{-9} T^3, \quad (6.84)$$

and finally at temperatures over 902.1 K but below 1223 K by

$$m = -0.162 + 3.080 \cdot 10^{-4} T. \quad (6.85)$$

6.3 Cladding oxide properties

6.3.1 Heat capacity

The solid phase heat capacity of the cladding oxide is calculated with the Shomate equation based on the NIST-JANAF Thermochemical tables [58]. The information was accessed from the NIST Chemistry WebBook.

$$C_P = 8.12 \left(A + Bt + Ct^2 + Dt^3 + \frac{E}{t^2} \right) \quad (6.86)$$

where C_P is the heat capacity of the zirconium dioxide in units $\text{J} \cdot \text{kg}^{-1} \cdot \text{K}^{-1}$, $t = \frac{T}{1000}$, and constants A, B, C, D and E are seen in the table 7.

Table 7. The coefficients of the Shomate equation for zirconium oxide.

Temperature (K)	298 - 1478	1478 - 2950
A	69.20	74.48
B	8.55	0
C	-0.86	0
D	0.25	0
E	-1.38	0

6.3.2 Thermal conductivity

Thermal conductivity of the cladding oxide is based on the MATPRO correlation for Zircalloys, Geelhood and Luscher [26]:

$$\lambda_{ox} = 0.835 + 1.810 \cdot 10^{-4} \cdot T \quad (6.87)$$

where λ_{ox} is the thermal conductivity of the oxide in units W/(m·K).

6.4 Fission gas release properties

6.4.1 Gas atom production rate

The gas atom production rate (in moles) in FINIX is calculated as per Massih and Forsberg [39]:

$$\beta = 0.3017 \frac{\dot{F}}{N_A}, \quad (6.88)$$

where \dot{F} is the fission rate in units of $m^{-3}s^{-1}$ and N_A the Avogadro constant.

6.4.2 Gas atom diffusion coefficient

The gas atom diffusion coefficient in FINIX is as per Turnbull [59, 60]:

$$D_{atom} = 7.6 \cdot 10^{-10} \exp\left(\frac{-3500.0}{T}\right) + 5.64 \cdot 10^{-25} \sqrt{\dot{F}} \exp\left(\frac{-13800.0}{T}\right) + 8.0 \cdot 10^{-40} \dot{F}, \quad (6.89)$$

where \dot{F} is the fission rate in units of $m^{-3}s^{-1}$.

6.4.3 Intragranular bubble radius

The intragranular bubble radius is calculated in FINIX as per White and Tucker [56]

$$R_b = 5.0 \cdot 10^{-10} \left(1 + 106 \exp\left(\frac{-8691}{T}\right) \right), \quad (6.90)$$

where T is the temperature and R_b is in meters.

6.4.4 Volume diffusion coefficient

The volume diffusion coefficient is calculated in FINIX as per van Uffelen et al. [38]:

$$D_{vol} = 3.0 \cdot 10^{-5} \exp\left(\frac{-7.22 \cdot 10^{-19}}{kT}\right). \quad (6.91)$$

where T is the temperature and, k the Boltzmann constant.

6.4.5 Intragranular bubble number density

The intragranular bubble number density is calculated in FINIX as per White and Tucker [56]:

$$N_b = \frac{1.52 \cdot 10^{27}}{T} - 3.3 \cdot 10^{23}. \quad (6.92)$$

6.4.6 Grain boundary vacancy diffusion coefficient

The grain boundary vacancy diffusion coefficient is calculated in FINIX as [40]:

$$D_v = 6.9 \cdot 10^{-4} \exp\left(\frac{-5.35 \cdot 10^{-19}}{kT}\right). \quad (6.93)$$

6.5 Gas gap and plenum properties

The gas conductivity model and correlations are discussed in chapter 3.3.

The plenum gas model in FINIX has not been updated to treat gases other than helium. The plenum gas is thus assumed to consist solely of helium. While this is a crude approximation, it only affects the plenum temperature through the heat transfer coefficients calculated for the pellet surface and the cladding facing the plenum. Since both are in any case in contact with the same gas and thermal equilibrium in the plenum is assumed, the error resulting from the approximation is manageable until correlations for the other species can be introduced. The correlations for helium are taken from Ref. [61]. Compared to the FRAPTRAN correlations, the numerical values are very similar.

The dynamic viscosity μ , density ρ and Prandtl number Pr are given by

$$\mu = 3.674 \cdot 10^{-7} T^{0.7} \text{ (kg/ms)}, \quad (6.94)$$

$$\rho = 48.14 \cdot 10^{-5} \frac{P}{T} \left[1 + 0.4446 \cdot 10^{-5} \frac{P}{T^{1.2}} \right] \text{ (kg/m}^3\text{)}, \quad (6.95)$$

$$Pr = \frac{0.717}{1 + 1.123 \cdot 10^{-8} P} T^{-(0.01 - 1.42 \cdot 10^{-9} P)} \text{ (dimensionless)}. \quad (6.96)$$

In the above, T is given in Kelvin and P in $\text{N}\cdot\text{m}^{-2}$.

The kinematic viscosity is given by $\nu = \frac{\mu}{\rho}$.

6.6 Neutronics

6.6.1 Cross sections

The radial power distribution model uses microscopic fission and capture cross sections for some actinides and a macroscopic scattering cross chapter. The current radial power distribution is that of Lassmann et al. [53], and it covers the actinides ^{235}U , ^{238}U , ^{239}Pu , ^{240}Pu , ^{241}Pu and ^{242}Pu .

The macroscopic scattering cross chapter is set to be 300 barns (10^{-24} cm^2).

The microscopic cross sections used by FINIX are those used also in FRAPCON, and are reported in table 8. Cross sections are given for light water and heavy water reactors separately.

Table 8. Cross sections used by the radial power distribution model. Subscript f denotes fission cross chapter and c the capture cross chapter. Cross sections are reported in barns (10^{-24} cm^2).

Nuclide	σ_f		σ_c	
	LWR	HWR	LWR	HWR
^{235}U	41.5	107.9	9.7	22.3
^{238}U	0	0	0.78	1.16
^{239}Pu	105	239.18	58.6	125.36
^{240}Pu	0.584	0.304	100	127.26
^{241}Pu	120	296.95	50	122.41
^{242}Pu	0.458	0.191	80	91.3

6.7 Coolant

6.7.1 Dittus-Boelter heat transfer coefficient

The solution of the Dittus-Boelter heat transfer coefficient depends on whether the flow of coolant is laminar or turbulent. This is determined by the Reynolds number, defined as follows:

$$Re = \frac{\phi_{cool} D_e}{\mu} \quad (6.97)$$

where ϕ_{cool} is the coolant mass flux, D_e is the effective hydraulic diameter of the flow channel and μ is the dynamic viscosity of the coolant. The Reynolds number is used in the calculation of the Dittus-Boelter heat transfer coefficient, as well as the determination of the correlation to be used for h_{DB} . If the Reynolds number is larger than 2000, the coolant flow is turbulent and the Dittus-Boelter heat transfer coefficient is calculated as:

$$h_{DB} = \frac{0.023 \lambda_{cool}}{D_e} Re^{0.8} Pr^{0.4} \quad (6.98)$$

while in case of the laminar flow, $Re < 2000$ and h_{DB} is calculated as:

$$h_{DB} = 7.86 \frac{\lambda_{cool}}{D_e} \quad (6.99)$$

where λ_{cool} is the thermal conductivity of the coolant, D_e is the hydraulic diameter of the fuel rod, Re is the Reynolds number, and Pr is the Prandtl number. The Prandtl number is defined as the ratio of momentum diffusivity to thermal diffusivity, and can be calculated from:

$$Pr = C_p \frac{\mu}{\lambda}, \quad (6.100)$$

and the hydraulic diameter is defined as:

$$D_e = \frac{4A_{flow}}{P_{wet}} \quad (6.101)$$

where A_{flow} is the cross-sectional area of the coolant flow, and P_{wet} is the perimeter of the wetted area. These can be solved when the assembly geometry and the rod pitch and radius are known.

6.7.2 Nucleate boiling heat transfer coefficient

The nucleate boiling heat transfer coefficient h_{NB} is based on the Jens-Lottes formulation [62]. The original formulation is written in imperial units, but by converting the formula into SI-units, we get:

$$h_{NB} = 1.264 \cdot \exp\left(\frac{P}{6.205 \cdot 10^6}\right) (q'')^{0.25} \quad (6.102)$$

6.7.3 Density

The coolant density is calculated using the IAWPS Industrial Formulation [41], in which the thermal properties of liquid water are governed by the Gibbs free energy formula:

$$\frac{g(P, T)}{RT} = \gamma(\pi, \tau) = \sum_{i=1}^{34} n_i (7.1 - \pi)^{l_i} (\tau - 1.222)^{j_i}, \quad (6.103)$$

where $g(P, T)$ is the Gibbs free energy, P is the coolant pressure, R the specific gas constant of the water with a value of $0.461526 \text{ kJ} \cdot \text{kg}^{-1} \cdot \text{K}^{-1}$ and T the coolant temperature. π is the reduced pressure, defined as

$$\pi = \frac{P}{16.53 \cdot 10^6 \text{ Pa}} \quad (6.104)$$

and τ the reduced temperature, defined as

$$\tau = \frac{1386K}{T} \quad (6.105)$$

The symbol π is used for the reduced pressure only in this section as the same convention is used in [41]. The indexed factors n_i , l_i and J_i can be found from the IAWPS Industrial Formulation [41] and are implemented in the FINIX source code. The factor are the same in each of the functions γ_i that follow.

By differentiating the Gibbs free energy formula, we can get the definition of density:

$$\rho = \frac{1}{(\partial g / \partial P)_T} = \frac{P}{RT_\pi \gamma_\pi}, \quad (6.106)$$

where

$$\gamma_\pi = \left(\frac{\partial \gamma}{\partial \pi} \right)_\tau = \sum_{i=1}^{34} -n_i l_i (7.1 - \pi)^{l_i - 1} (\tau - 1.222)^{J_i} \quad (6.107)$$

6.7.4 Isobaric and isochoric heat capacity

The correlations for the coolant isobaric and isochoric heat capacities are based on the IAPWS-95 Industrial Formulation [41]. While only the isobaric heat capacity is used in the axial temperature distribution calculations, the isochoric heat capacity must be known in order to solve the thermal conductivity of the coolant. The specific isobaric heat capacity is calculated from the Gibbs energy formula as:

$$C_p(T) = \left(\frac{\partial h}{\partial T} \right)_P = \frac{\partial}{\partial T} \left(g - T \left(\frac{\partial g}{\partial T} \right)_P \right)_P = -R\tau^2 \gamma_{\tau\tau}, \quad (6.108)$$

where h is the enthalpy of the coolant and $\gamma_{\tau\tau}$ is calculated, using the indexed factors n_i , l_i and J_i from the IAWPS Industrial Formulation [41], as:

$$\gamma_{\tau\tau} = \left(\frac{\partial^2 \gamma}{\partial \tau^2} \right)_\pi = \sum_{i=1}^{34} n_i (7.1 - \pi)^{l_i} J_i (J_i - 1) (\tau - 1.222)^{J_i - 2}, \quad (6.109)$$

Similarly, the specific isochoric heat capacity can be derived from the Gibbs free energy as:

$$C_v(T) = \left(\frac{\partial u}{\partial T} \right)_v = \frac{\partial}{\partial T} \left(g - T \left(\frac{\partial g}{\partial T} \right)_P - P \left(\frac{\partial g}{\partial P} \right)_T \right)_P = -R\tau^2 \gamma_{\tau\tau} + R \frac{(\gamma_\pi - \tau \gamma_{\pi\tau})^2}{\gamma_{\pi\pi}}, \quad (6.110)$$

where u is the specific internal energy and $\gamma_{\pi\tau}$ is defined as:

$$\gamma_{\pi\tau} = \left(\frac{\partial^2 \gamma}{\partial \pi \partial \tau} \right)_\pi = \sum_{i=1}^{34} n_i l_i (7.1 - \pi)^{l_i - 1} J_i (\tau - 1.222)^{J_i - 1} \quad (6.111)$$

6.7.5 Dynamic viscosity

In order to solve the heat transfer coefficient, the viscosity must be known for the calculation of the Reynolds and Prandtl numbers. The viscosity calculation is based on the 2008 IAPWS Formulation for the Viscosity of Ordinary Water Substance [63]. It holds for fluid states up to 1173K and 1000 MPa, thus covering all thermophysical states of reactor coolant behaviour.

The correlation includes three viscosity factors: $\bar{\mu}_0, \bar{\mu}_1, \bar{\mu}_2$. The first one corresponds to the viscosity in the limit of zero density, whereas the second one is the residual viscosity that represents the contribution of increasing density. The third factor is the critical region viscosity factor, which is usually left out at industrial applications as suggested by the IAPWS formulation, thus simplifying the viscosity formula as:

$$\bar{\mu} = \bar{\mu}_0(\bar{T}) \times \bar{\mu}_1(\bar{T}, \bar{\rho}) \times \bar{\mu}_2(\bar{T}, \bar{\rho}) \approx \bar{\mu}_0(\bar{T}) \times \bar{\mu}_1(\bar{T}, \bar{\rho}) \quad (6.112)$$

The first viscosity factor $\bar{\mu}_0$ is a dimensionless reduced quantity, which is a function of temperature:

$$\bar{\mu}_0 = \frac{100\sqrt{\bar{T}}}{1.67752 + 2.20462\bar{T}^{-1} + 0.636656\bar{T}^{-2} - 0.241605\bar{T}^{-3}}, \quad (6.113)$$

where \bar{T} is the reduced temperature, defined as:

$$\bar{T} = \frac{T}{647.096K} \quad (6.114)$$

The second factor $\bar{\mu}_1$ is a function of \bar{T} and reduced density $\bar{\rho}$:

$$\bar{\mu}_1 = \exp(\bar{\rho} \sum_{i=0}^5 (\frac{1}{\bar{T}} - 1)^i \sum_{j=0}^6 H_{ij}(\bar{\rho} - 1)^j), \quad (6.115)$$

where the correlation matrix H is based on the regression fit of the experimental data [63]:

$$H = \begin{bmatrix} H_{00} & H_{01} & H_{02} & H_{03} & H_{04} & H_{05} & H_{06} \\ H_{10} & H_{11} & H_{12} & H_{13} & H_{14} & H_{15} & H_{16} \\ H_{20} & H_{21} & H_{22} & H_{23} & H_{24} & H_{25} & H_{26} \\ H_{30} & H_{31} & H_{32} & H_{33} & H_{34} & H_{35} & H_{36} \\ H_{40} & H_{41} & H_{42} & H_{43} & H_{44} & H_{45} & H_{46} \\ H_{50} & H_{51} & H_{52} & H_{53} & H_{54} & H_{55} & H_{56} \end{bmatrix} = \begin{bmatrix} 0.520 & 0.223 & -0.281 & 0.162 & -0.033 & 0 & 0 \\ 0.085 & 0.999 & -0.907 & 0.257 & 0 & 0 & 0 \\ -1.084 & 1.888 & -0.772 & 0 & 0 & 0 & 0 \\ -0.290 & 1.266 & -0.490 & 0 & 0.070 & 0 & -0.0044 \\ 0 & 0 & -0.257 & 0 & 0 & 0.008702 & 0 \\ 0 & 0.121 & 0 & 0 & 0 & 0 & -0.00059 \end{bmatrix}$$

The dimensional viscosity μ can then be calculated by multiplying the reduced viscosity with the critical viscosity $\mu^* = 10^{-6} Pa$:

$$\mu = \bar{\mu} \cdot \mu^* \quad (6.116)$$

6.7.6 Thermal conductivity

The IAPWS Formulation 2011 for the Thermal Conductivity of Ordinary Water Substance [64] reports a correlation for the thermal conductivity of water, which is formulated similarly to the viscosity model presented before. Using the industrial formulation of the correlation, we get the following formula for the reduced thermal conductivity $\bar{\lambda}$:

$$\bar{\lambda} = \bar{\lambda}_0(\bar{T}) \cdot \bar{\lambda}_1(\bar{T}, \bar{\rho}) + \bar{\lambda}_2(\bar{T}, \bar{\rho}) \quad (6.117)$$

Again, the first factor $\bar{\lambda}_0$ represents the reduced thermal conductivity at zero density, the second factor $\bar{\lambda}_1$ is the residual contribution due to the increasing density, and the third factor $\bar{\lambda}_2$ corresponds to the reduced thermal conductivity in the critical region of the fluid. In this correlation, however, the critical region factor must be included due to its high impact in BWR scenarios where the coolant has a low pressure.

The zero density factor $\bar{\lambda}_0$ can be solved as a function of reduced temperature \bar{T} as:

$$\bar{\lambda}_0 = \frac{\sqrt{\bar{T}}}{2.443221 \cdot 10^{-3} + 1.32 \cdot 10^{-2} \bar{T}^{-1} + 6.77 \cdot 10^{-3} \bar{T}^{-2} + 3.46 \cdot 10^{-3} \bar{T}^{-3} + 4.10 \cdot 10^{-4} \bar{T}^{-4}}, \quad (6.118)$$

and the residual contribution factor $\bar{\lambda}_1$ can be calculated as a function of reduced temperature \bar{T} and density $\bar{\rho}$ as:

$$\bar{\lambda}_1 = \exp(\bar{\rho} \sum_{i=0}^4 (\frac{1}{\bar{T}} - 1)^i \sum_{j=0}^5 L_{ij} (\bar{\rho} - 1)^j), \quad (6.119)$$

where the correlation matrix L is based on the regression fit of the experimental data [64]:

$$L = \begin{bmatrix} L_{00} & L_{01} & L_{02} & L_{03} & L_{04} & L_{05} \\ L_{10} & L_{11} & L_{12} & L_{13} & L_{14} & L_{15} \\ L_{20} & L_{21} & L_{22} & L_{23} & L_{24} & L_{25} \\ L_{30} & L_{31} & L_{32} & L_{33} & L_{34} & L_{35} \\ L_{40} & L_{41} & L_{42} & L_{43} & L_{44} & L_{45} \end{bmatrix} \quad (6.120)$$

$$= \begin{bmatrix} 1.604 & -0.646 & 0.111 & 0.103 & -0.050 & 0.0061 \\ 2.338 & -2.788 & 1.536 & -0.463 & 0.083 & -0.007 \\ 2.197 & -4.546 & 3.558 & -1.409 & 0.275 & -0.021 \\ -1.211 & 1.608 & -0.621 & 0.072 & 0 & 0 \\ -2.720 & 4.576 & -3.183 & 1.117 & -0.193 & 0.013 \end{bmatrix}$$

The third factor $\bar{\lambda}_2$ has both a regular and an industrial form in [64]. The industrial form is preferred due to better computational performance and simplicity, being formulated as:

$$\bar{\lambda}_2 = \frac{\Lambda \bar{\rho} \bar{c}_p \bar{T}}{\bar{\mu}} Z(\bar{q}_D \xi) \quad (6.121)$$

Here $\Lambda = 177.85$, \bar{q}_D the cutoff wave number with a value of $2.5 \cdot 10^9 \text{ m}^{-1}$, and \bar{c}_p the reduced isobaric specific heat capacity, defined as:

$$\bar{c}_p = \frac{c_p}{R}, \quad (6.122)$$

where R is the specific gas constant of the water with a value of $0.461526 \text{ kJ} \cdot \text{kg}^{-1} \cdot \text{K}^{-1}$. The correlation length ξ is defined as:

$$\xi = \xi_0 \left(\frac{\Delta \bar{x}}{\Gamma_0} \right)^{\frac{\nu}{\gamma}}, \quad (6.123)$$

where the critical amplitudes are $\xi_0 = 0.13 \cdot 10^{-9} \text{ m}$ and $\Gamma_0 = 0.06$, the critical exponents are $\nu = 0.630$ and $\gamma = 1.239$ and the $\Delta \bar{x}$ is defined as:

$$\Delta \bar{x}(\bar{T}, \bar{\rho}) = \bar{\rho} [\zeta(\bar{T}, \bar{\rho}) - \zeta(\bar{T}_R, \bar{\rho}) \frac{\bar{T}_R}{\bar{T}}] \quad (6.124)$$

where \bar{T}_R is the reduced reference temperature and function $\zeta(\bar{T}, \bar{\rho})$ is a function of density derivatives. The first term is calculated by calculating the numeric differentials of the reduced density and pressure:

$$\zeta(\bar{T}, \bar{\rho}) = \left(\frac{\partial \bar{\rho}}{\partial \bar{P}} \right)_T = -\frac{\gamma_{\pi\pi}}{RT\gamma_{\pi}^2}, \quad (6.125)$$

with $\gamma_{\pi\pi}$ being calculated as:

$$\gamma_{\pi\pi} = \left(\frac{\partial^2 \gamma}{\partial \pi^2} \right)_T = \sum_{i=1}^{34} n_i l_i (l_i - 1) (7.1 - \pi)^{l_i - 2} (\tau - 1.222)^{j_i}, \quad (6.126)$$

whereas the latter derivative can be solved in the industrial applications as:

$$\zeta(\bar{T}_R, \bar{\rho}) = \frac{1}{\sum_{i=0}^5 A_{ij} \bar{\rho}^{-i}} \quad (6.127)$$

Here the matrix A is defined as:

$$A = \begin{bmatrix} A_{00} & A_{01} & A_{02} & A_{03} & A_{04} \\ A_{10} & A_{11} & A_{12} & A_{13} & A_{14} \\ A_{20} & A_{21} & A_{22} & A_{23} & A_{24} \\ A_{30} & A_{31} & A_{32} & A_{33} & A_{34} \\ A_{40} & A_{41} & A_{42} & A_{43} & A_{44} \\ A_{50} & A_{51} & A_{52} & A_{53} & A_{54} \end{bmatrix} \quad (6.128)$$

$$= \begin{bmatrix} 6.538 & 6.527 & 5.355 & 1.552 & 1.120 \\ -5.611 & -6.308 & -3.964 & 0.465 & 0.596 \\ 3.396 & 8.084 & 8.920 & 8.932 & 9.890 \\ -2.275 & -9.822 & -12.03 & -11.03 & -10.33 \\ 10.26 & 12.14 & 9.195 & 6.168 & 4.669 \\ 1.978 & -5.543 & -2.169 & -0.965 & -0.503 \end{bmatrix},$$

and the value of j is determined by the reduced pressure of the coolant as:

$$j = 0 : \bar{\rho} \leq 0.310559 \quad (6.129)$$

$$j = 1 : 0.310559 < \bar{\rho} \leq 0.776398$$

$$j = 2 : 0.776398 < \bar{\rho} \leq 1.242236$$

$$j = 3 : 1.242236 < \bar{\rho} \leq 1.863354$$

$$j = 4 : 1.863354 < \bar{\rho}$$

7. Numerical implementation

7.1 General outline of the execution order

The FINIX calculation of the thermal and mechanical time evolution of the fuel rod proceeds in discrete time steps δt . For each time step, the thermal and mechanical solutions are found by numerical iteration. The iteration process is schematically presented in Fig. 2. The iteration consists of two main loops. The outer loop consists of solving the thermal properties of the fuel and the cladding, the gap conductance and the heat equation and plenum temperature. The second loop consists of solving the internal pressure and pellet and cladding deformations, and is situated within the outer thermal iteration loop. For the mechanical solution, the internal pressure is used as a convergence criterion, while for the outer loop, convergence of the gap conductivities for all axial nodes is required. In between, fission gas behavior is solved. On the algorithm level, the iteration is performed using the secant method, which is a method similar to the Newton-Raphson method, but where the function derivative is evaluated numerically instead of analytically. The method is described in detail in, e.g., Ref. [30].

The numerical methods used to solve the individual modules are described in the following Sections.

7.2 Thermal model

7.2.1 Transient heat equation

FEM discretization of the 1D transient heat equation

As was discussed in chapter 3.1, the temperature in the pellet and cladding is solved in axial slices, in each of which the temperature T is assumed independent of the axial and azimuthal coordinates z and θ . The heat equation then takes the form

$$C_V[T(r)] \frac{\partial T}{\partial t} - \frac{1}{r} \frac{\partial}{\partial r} \left[\lambda[T(r)] r \frac{\partial T}{\partial r} \right] - s(r) = 0, \quad (7.1)$$

with C_V denoting the volumetric heat capacity and λ the conductivity. Note that neither the heat capacity nor conductivity is assumed constant w.r.t. the coordinate r .

The heat equation (7.1) is discretized with the Finite Element Method (FEM) [65]. The pellet is divided into $n_f - 1$ radial *elements*, with the i :th element comprising the volume between the *nodes* at $r = r_i$ and $r = r_{i+1}$. The first node is located at the inner surface of the pellet ($r_1 = R_0$), while the last node is at the pellet outer surface ($r_{n_f} = R_f$). The cladding is similarly divided into $n_c - 1$ elements and n_c nodes, with the first cladding node at the cladding inner surface ($r_{n_f+1} = R_{ci}$) and the last at the outer surface ($r_{n_f+n_c} = R_{co}$).

The gas gap element (between nodes n_f and $n_f + 1$) is handled through the gap conductance boundary conditions, as will be discussed below. In what follows, an element will refer to a general element, either within the pellet or in the cladding, unless otherwise indicated. For the pellet elements, the material parameters (conductivity, heat capacity) are given by the correlations of Sec. 6.1, and for the cladding elements by the correlations of Sec. 6.2.

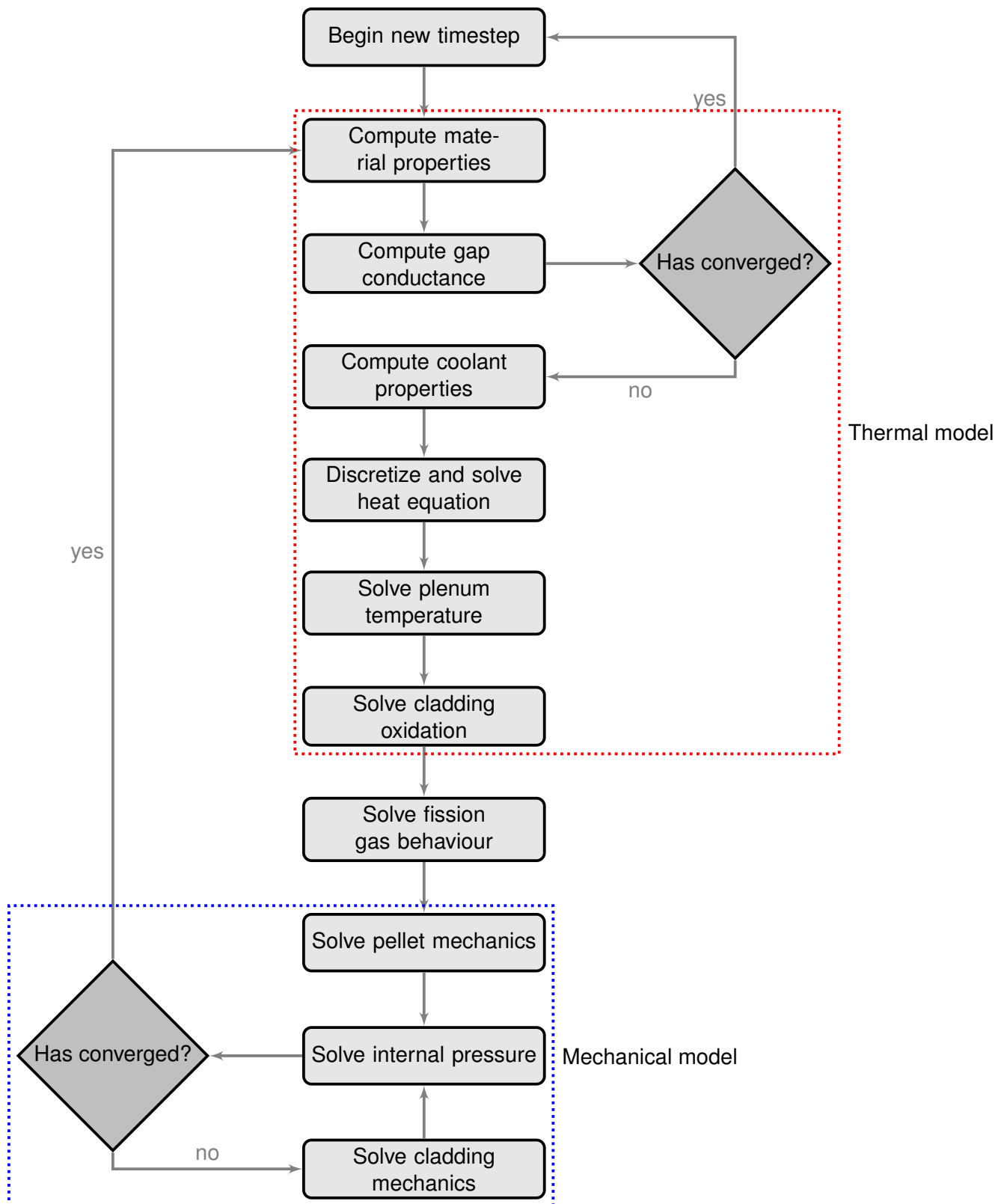


Figure 2. Description of the FINIX solution algorithm. Computation of a new time step begins from the top and proceeds through the modules as indicated by the arrows. Convergence checks are made for the gap conductance in the thermal model, and for the internal pressure in the mechanical model. On the first round of iteration, the convergence is always considered to fail, so that the full thermo-mechanical model is executed at least once.

Discretization of one element

In the finite element method, the continuous equations are discretized first for each element. The global discretization of the whole system is then assembled from the discretized individual elements. For each element, the numerical solution provides the value of the temperature only at the location of the nodes. Within the element, the solution is approximated by *shape functions*, or *basis functions*. In the simplest case, the basis functions are linear, so that within the element the temperature T is assumed to behave linearly as a function of r . The actual finite element equations are then derived by minimizing the residual (discretization error) between the original equations and the discretized equations. The minimization can be done with several different methods. A common choice is the Galerkin method, where the minimization is done by weighting the residual with the basis functions [65]. This is also the method we will use.

The temperature inside the i :th element is approximated with linear basis functions N_i and N_{i+1} so that

$$T(r) \approx [N_i(r) \ N_{i+1}(r)] \begin{bmatrix} T_i \\ T_{i+1} \end{bmatrix}, \quad (7.2)$$

where the square brackets indicate row and column vectors, T_i is the temperature at the i :th node, and

$$N_i(r) = \frac{r_{i+1} - r}{r_{i+1} - r_i}, \quad (7.3)$$

$$N_{i+1}(r) = \frac{r - r_i}{r_{i+1} - r_i}. \quad (7.4)$$

For Eq. (7.1), the Galerkin method results in the matrix equation

$$\begin{aligned} & \int \begin{bmatrix} N_i(r) \\ N_{i+1}(r) \end{bmatrix} C_V(r) [N_i(r) N_{i+1}(r)] dV \frac{\partial}{\partial t} \begin{bmatrix} T_i \\ T_{i+1} \end{bmatrix} \\ & - \begin{bmatrix} \lambda_{i,j} & \lambda_{i,j+1} \\ \lambda_{i+1,i} & \lambda_{i+1,i+1} \end{bmatrix} \begin{bmatrix} T_i \\ T_{i+1} \end{bmatrix} - \int \begin{bmatrix} N_i(r) \\ N_{i+1}(r) \end{bmatrix} s(r) dV = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \end{aligned} \quad (7.5)$$

where the matrix elements $\lambda_{i,j}$ are

$$\lambda_{i,j} = -2\pi\Delta z \int_{r_i}^{r_j} r\lambda(r) \frac{\partial N_i(r)}{\partial r} \frac{\partial N_j(r)}{\partial r} dr + 2\pi\Delta z \left[r\lambda(r) N_i(r) \frac{\partial N_j(r)}{\partial r} \right]_{r_i}^{r_j}, \quad (7.6)$$

and the integral operator can be written as $\int dV = 2\pi\Delta z \int_{r_i}^{r_{i+1}} r dr$, since the integrands have no axial or azimuthal dependence in the slice of thickness Δz . We also linearize the heat capacity C_V , conductivity λ and the source term s within the element, so that, given the values at the nodes i and $i+1$ (denoted by subscripts), we have

$$C_V(r) \approx C_i + (C_{i+1} - C_i) \frac{r - r_i}{r_{i+1} - r_i}, \quad (7.7)$$

$$\lambda(r) \approx \lambda_i + (\lambda_{i+1} - \lambda_i) \frac{r - r_i}{r_{i+1} - r_i}, \quad (7.8)$$

$$s(r) \approx s_i + (s_{i+1} - s_i) \frac{r - r_i}{r_{i+1} - r_i}, \quad (7.9)$$

for $r_i \leq r \leq r_{i+1}$.

Integration over the element gives the matrix equation

$$(\mathbf{K}_i + \mathbf{C}_i) \begin{bmatrix} T_i^{k+1} \\ T_{i+1}^{k+1} \end{bmatrix} = \mathbf{C}_i \begin{bmatrix} T_i^k \\ T_{i+1}^k \end{bmatrix} + \mathbf{f}_i, \quad (7.10)$$

where time derivative has also been discretized with the implicit Euler method. The superscript of T indicates the time step of the time-discretized temperature, so that $T_i^k \equiv T(r = r_i, t = k\delta t)$, where δt is the time step. The implicit Euler method remains unconditionally stable with all values of δt [65, 66]. The matrices in Eq. (7.10) are defined as follows:

$$\mathbf{K}_i = \frac{\lambda_i(2r_i + r_{i+1}) + \lambda_{i+1}(r_i + 2r_{i+1})}{6(r_{i+1} + r_i)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (7.11)$$

$$\mathbf{C}_i = \frac{r_{i+1} - r_i}{60\delta t} \begin{bmatrix} c_i(12r_i + 3r_{i+1}) + c_{i+1}(3r_i + 2r_{i+1}) & c_i(3r_i + 2r_{i+1}) + c_{i+1}(2r_i + 3r_{i+1}) \\ c_i(3r_i + 2r_{i+1}) + c_{i+1}(2r_i + 3r_{i+1}) & c_i(2r_i + 3r_{i+1}) + c_{i+1}(3r_i + 12r_{i+1}) \end{bmatrix}, \quad (7.12)$$

$$\mathbf{f}_i = \frac{r_{i+1} - r_i}{12} \begin{bmatrix} s_i(3r_i + r_{i+1}) + s_{i+1}(r_i + r_{i+1}) \\ s_i(r_i + r_{i+1}) + s_{i+1}(r_i + 3r_{i+1}) \end{bmatrix} + \begin{bmatrix} -r_i q_i \\ r_{i+1} q_{i+1} \end{bmatrix}. \quad (7.13)$$

The second term in the vector \mathbf{f}_i is determined by the boundary conditions of the element, with q_i the heat flux over the surface at $r = r_i$. The boundary conditions will be discussed in more detail below.

The gas gap element

For $i = \{1, 2, \dots, n_f - 1, n_f + 1, n_f + 2, \dots, n_f + n_c\}$, the element discretization is given by the 2×2 matrices derived in the previous chapter. For the gas gap, *i.e.*, the n_f :th element, the matrices are derived from the gas gap conductance model described in Sec. 3.3. Given the heat transfer coefficient h , the matrix \mathbf{K}_{n_f} is

$$\mathbf{K}_{n_f} = \begin{bmatrix} hr_{n_f} & -hr_{n_f} \\ -hr_{n_f} & hr_{n_f} \end{bmatrix}. \quad (7.14)$$

There is no power generated in the gap, and the specific heat of the gas is assumed negligible. Hence, $\mathbf{C}_{n_f} = \mathbf{0}$ and $\mathbf{f}_{n_f} = \mathbf{0}$.

Global matrices

The global matrices for the whole system are assembled from the 2×2 element matrices by taking the sum node by node. The result is an $(n_f + n_c) \times (n_f + n_c)$ tridiagonal matrix. For brevity, we introduce a shorthand notation of the element matrices:

$$\mathbf{K}_i \equiv \begin{bmatrix} K_i^{(11)} & K_i^{(12)} \\ K_i^{(21)} & K_i^{(22)} \end{bmatrix}, \quad \mathbf{C}_i \equiv \begin{bmatrix} C_i^{(11)} & C_i^{(12)} \\ C_i^{(21)} & C_i^{(22)} \end{bmatrix}. \quad (7.15)$$

Then, the global matrices are of the tridiagonal form

$$\mathbf{K} = \begin{bmatrix} D_1 & U_1 & & & & \mathbf{0} \\ L_1 & D_2 & U_2 & & & \\ & L_2 & D_3 & U_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & L_{n-2} & D_{n-1} & U_{n-1} \\ \mathbf{0} & & & & L_{n-1} & D_n \end{bmatrix}, \quad (7.16)$$

where $n = n_f + n_c$ is the total number of nodes. The diagonal (D), upper diagonal (U) and lower diagonal (L) elements are given as

$$D_i = K_i^{(11)} + K_{i-1}^{(22)} \quad (\text{for } 2 \leq i \leq n - 1); \quad D_1 = K_1^{(11)}; \quad D_n = K_{n-1}^{(22)}, \quad (7.17)$$

$$U_i = K_i^{(12)} \quad (\text{for } 1 \leq i \leq n - 1), \quad (7.18)$$

$$L_i = K_i^{(21)} \quad (\text{for } 1 \leq i \leq n - 1). \quad (7.19)$$

The matrix \mathbf{C} is assembled in a similar fashion, with the elements of the matrices \mathbf{K}_i replaced with the elements of \mathbf{C}_i .

The load vector is given as

$$\mathbf{f} = \begin{bmatrix} f_1^{(1)} \\ f_1^{(2)} + f_2^{(1)} \\ f_2^{(2)} + f_3^{(1)} \\ \vdots \\ f_{n-2}^{(2)} + f_{n-1}^{(1)} \\ f_{n-1}^{(2)} \end{bmatrix}, \quad (7.20)$$

where $f_i^{(1)}$ and $f_i^{(2)}$ are the two components of \mathbf{f}_i :

$$\mathbf{f}_i \equiv \begin{bmatrix} f_i^{(1)} \\ f_i^{(2)} \end{bmatrix}. \quad (7.21)$$

Finally, the global matrix equation for the complete system is

$$(\mathbf{K} + \mathbf{C})\mathbf{T}^{k+1} = \mathbf{C}\mathbf{T}^k + \mathbf{f}, \quad (7.22)$$

where the vector \mathbf{T}^k contains the temperatures at the k :th time step:

$$\mathbf{T}^k \equiv \begin{bmatrix} T_1^k \\ T_2^k \\ \vdots \\ T_n^k \end{bmatrix}. \quad (7.23)$$

Time discretization

Implicit time discretization of the heat equation is implied in Eq. (7.22). The time discretization follows the standard implicit finite difference discretization (see, *e.g.*, Ref. [66]), which remains unconditionally stable for all time steps δt . The implicit formulation means that to calculate the temperature at time $t = (k + 1)\delta t$, the temperature-dependent terms of the heat equation are evaluated at the same point in time, at $t = (k + 1)\delta t$. For constant (non-temperature-dependent) material properties, it is possible to solve the one-dimensional implicit time-discretized matrix equation, Eq. (7.22), without iteration. However, since the material correlations depend on temperature, and cannot in general be linearized, it is necessary to iterate the solution of \mathbf{T}^{k+1} , until the temperature converges. This is part of the iteration procedure described above in Sec. 7.1.

Boundary conditions

The boundary conditions affect the discretization of the elements at the center of the pellet and at the outer surface of the cladding. For the first element, the flux through the first node is set to zero, *i.e.*, $q_1 = 0$ in Eq. (7.13). For the cladding outer surface, several alternative boundary conditions can be used.

Dirichlet boundary condition. For a fixed surface temperature, $T(R_{co}) = T_{\text{surf}}$, the appropriate boundary condition is enforced by setting $L_n = D_n = 0$ for \mathbf{C} , $L_n = 0$ and $D_n = 1$ for \mathbf{K} , and $f_{n-1}^{(2)} = T_{\text{surf}}$ in the load vector \mathbf{f} . This is equivalent with the the equation $T_n^{k+1} = T_{\text{surf}}$.

Neumann boundary condition. If the heat flux across the cladding outer surface is fixed, then one only needs to assign q_n the desired value in \mathbf{f} .

Robin boundary condition. The heat flux can also be given as a function of the bulk temperature of the coolant, T_{coolant} , and the heat transfer coefficient h_{co} so that $q_n = h_{co} [T_n - T_{\text{coolant}}]$. In this case, the last element of \mathbf{f} is $f_{n-1}^{(2)} = r_n h_{co} T_{\text{coolant}}$, and the remaining $r_n h_{co}$ is added to the last diagonal element D_n in the matrix \mathbf{K} .

In the case of the Robin boundary condition, the heat transfer coefficients can be computed from internal correlations (see Sec. 6.7) or, they be given by the user.

Solution of the matrix equations

The resulting matrix equation for the vector \mathbf{T}^{k+1} , Eq. (7.22), can be solved non-iteratively with the tridiagonal matrix algorithm, which a variant of the standard Gaussian elimination method. The algorithm is a standard numerical method, and its details are not explained here. The interested reader is referred to Chapter 2 of Ref. [30].

Although the heat equation itself can be solved non-iteratively, the dependence of the material properties, gap conductance and the pellet and cladding mechanical solution on the temperature requires iteration of the full thermo-mechanical solution. The scheme is described in more detail above, in Sec. 7.1.

7.2.2 Steady-state heat equation

Discretization of the 1D steady-state heat equation

For the steady state solver, we shall linearize the thermal conductivity and the power density inside the elements as is done with the transient solution in order to model the same distributions with both solvers.

If we consider the steady state form of the 1D heat equation

$$-\frac{1}{r} \frac{\partial}{\partial r} \left[\lambda [T(r)] r \frac{\partial T}{\partial r} \right] - s(r) = 0, \quad (7.24)$$

and write the linearization of the thermal conductivity and heat source as

$$a_q r + b_q = q_i + (q_{i+1} - q_i) \frac{r - r_i}{r_{i+1} - r_i} \quad (7.25)$$

with

$$a_q = \frac{q_{i+1} - q_i}{r_{i+1} - r_i} \quad (7.26)$$

$$b_q = q_i - (q_{i+1} - q_i) \frac{r_i}{r_{i+1} - r_i}. \quad (7.27)$$

we can write the equation in a single element as

$$-\frac{1}{r} \frac{\partial}{\partial r} \left[(a_\lambda r + b_\lambda) r \frac{\partial T}{\partial r} \right] = a_s r + b_s, \quad (7.28)$$

Now we can work towards an analytic solution for the element inner node temperature T_j .

Nonzero a and b

We'll first assume that $a_\lambda \neq 0$ and $b_\lambda \neq 0$ and handle the exceptions later:

$$-\frac{\partial}{\partial r} \left[(a_\lambda r + b_\lambda) r \frac{\partial T}{\partial r} \right] = a_s r^2 + b_s r \quad (7.29)$$

$$-(a_\lambda r + b_\lambda) r \frac{\partial T}{\partial r} = \frac{a_s}{3} r^3 + \frac{b_s}{2} r^2 + C_0 \quad (7.30)$$

$$-\frac{\partial T}{\partial r} = \frac{\frac{a_s}{3} r^3 + \frac{b_s}{2} r^2 + C_0}{a_\lambda r^2 + b_\lambda r} \quad (7.31)$$

$$\begin{aligned} -\frac{\partial T}{\partial r} &= \frac{a_s}{3a_\lambda} r + \frac{3b_s a_\lambda - 2b_\lambda a_s}{6a_\lambda^2} - \frac{3b_\lambda b_s a_\lambda - 2b_\lambda^2 a_s}{6a_\lambda^2 (a_\lambda r + b_\lambda)} \\ &\quad + \frac{C_0}{b_\lambda r} - \frac{C_0 a_\lambda}{b_\lambda (a_\lambda r + b_\lambda)} \end{aligned} \quad (7.32)$$

Now we can integrate from r_i to r_{i+1} and from T_i to T_{i+1}

$$\begin{aligned} -\int_{T_i}^{T_{i+1}} dT &= \int_{r_i}^{r_{i+1}} \left[\frac{a_s}{3a_\lambda} r + \frac{3b_s a_\lambda - 2b_\lambda a_s}{6a_\lambda^2} - \frac{3b_\lambda b_s a_\lambda - 2b_\lambda^2 a_s}{6a_\lambda^2 (a_\lambda r + b_\lambda)} \right. \\ &\quad \left. + \frac{C_0}{b_\lambda r} - \frac{C_0 a_\lambda}{b_\lambda (a_\lambda r + b_\lambda)} \right] dr \end{aligned} \quad (7.33)$$

We know that the thermal conductivity is positive throughout the integration interval, i.e.,

$$a_\lambda r + b_\lambda > 0,$$

giving

$$\begin{aligned} T_i - T_{i+1} &= \frac{a_s}{6a_\lambda} (r_{i+1}^2 - r_i^2) + \frac{3b_s a_\lambda - 2b_\lambda a_s}{6a_\lambda^2} (r_{i+1} - r_i) \\ &\quad - \frac{3b_\lambda b_s a_\lambda - 2b_\lambda^2 a_s}{6a_\lambda^3} \log \frac{a_\lambda r_{i+1} + b_\lambda}{a_\lambda r_i + b_\lambda} \\ &\quad + \frac{C_0}{b_\lambda} \log \frac{r_{i+1}}{r_i} - \frac{C_0}{b_\lambda} \log \frac{a_\lambda r_{i+1} + b_\lambda}{a_\lambda r_i + b_\lambda} \end{aligned} \quad (7.34)$$

We can simplify the expression by substituting in the nodal thermal conductivity

$$a_\lambda r_n + b_\lambda = \lambda_n$$

to get

$$\begin{aligned} T_i - T_{i+1} &= \frac{a_s}{6a_\lambda} (r_{i+1}^2 - r_i^2) + \frac{3b_s a_\lambda - 2b_\lambda a_s}{6a_\lambda^2} (r_{i+1} - r_i) \\ &\quad - \frac{3b_\lambda b_s a_\lambda - 2b_\lambda^2 a_s}{6a_\lambda^3} \log \frac{\lambda_{i+1}}{\lambda_i} \\ &\quad + \frac{C_0}{b_\lambda} \log \frac{r_{i+1}}{r_i} - \frac{C_0}{b_\lambda} \log \frac{\lambda_{i+1}}{\lambda_i} \end{aligned} \quad (7.35)$$

We see that if $r_i = 0$ we must have $C_0 = 0$. In other cases C_0 can be determined using the heat flux equation written at $r = r_i$

$$q''(r_i) = (-\lambda(r) \nabla T(r))_{r=r_i} = \frac{P(r < r_i)}{2\pi r_i \Delta Z} \hat{e}_r \quad (7.36)$$

The equation states that the heat flux q'' at r_i is equal to the power generated inside the cylindrical surface at $r = r_i$ divided by the surface area. This assumes steady state in heat transfer and generation as well as no axial heat transfer in the fuel. Taking the gradient of the one dimensional temperature field gives

$$\left(-\lambda(r)\frac{\partial T}{\partial r}\right)_{r=r_i} \hat{e}_r = \frac{P(r < r_i)}{2\pi r_i \Delta z} \hat{e}_r. \quad (7.37)$$

We can substitute the derivative of the temperature using Eq. 7.31 to get

$$\lambda_i \frac{\frac{a_s}{3} r_i^3 + \frac{b_s}{2} r_i^2 + C_0}{r_i (a_\lambda r_i + b_\lambda)} = \frac{P(r < r_i)}{2\pi r_i \Delta z} \quad (7.38)$$

$$\lambda_i \frac{\frac{a_s}{3} r_i^3 + \frac{b_s}{2} r_i^2 + C_0}{r_i \lambda_i} = \frac{P(r < r_i)}{2\pi r_i \Delta z} \quad (7.39)$$

$$\frac{\frac{a_s}{3} r_i^3 + \frac{b_s}{2} r_i^2 + C_0}{r_i} = \frac{P(r < r_i)}{2\pi r_i \Delta z} \quad (7.40)$$

Solving for C_0 yields

$$C_0 = \frac{P(r < r_i)}{2\pi \Delta z} - \frac{a_s}{3} r_i^3 - \frac{b_s}{2} r_i^2, \quad (7.41)$$

where the power term can be written as a multiple of the average linear power at regions $r < r_i$ and the height of the axial segment

$$P(r < r_i) = q'(r < r_i) \Delta z = \sum_{j=0}^{j=i-1} \int_{r_j}^{r_{j+1}} a_s^j r + b_s^j dA \Delta z. \quad (7.42)$$

$$P(r < r_i) = \sum_{j=0}^{j=i-1} \int_{r_j}^{r_{j+1}} (a_s^j r + b_s^j) 2\pi r dr \Delta z. \quad (7.43)$$

Finally we'll obtain an expression for C_0

$$C_0 = \sum_{j=0}^{j=i-1} \left[\frac{a_s^j}{3} (r_{j+1}^3 - r_j^3) + \frac{b_s^j}{2} (r_{j+1}^2 - r_j^2) \right] - \frac{a_s}{3} r_i^3 - \frac{b_s}{2} r_i^2, (r_i \neq 0) \quad (7.44)$$

The value of C_0 depends only on the power distribution and the node radial coordinates.

Nonzero b, zero a

For the case where $a_\lambda = 0$ we have a constant heat conductivity in the element. The solution of the heat equation is somewhat simplified:

$$-\frac{\partial}{\partial r} \left[b_\lambda r \frac{\partial T}{\partial r} \right] = a_s r^2 + b_s r \quad (7.45)$$

$$-b_\lambda r \frac{\partial T}{\partial r} = \frac{a_s}{3} r^3 + \frac{b_s}{2} r^2 + C_0 \quad (7.46)$$

$$-\frac{\partial T}{\partial r} = \frac{a_s}{3b_\lambda} r^2 + \frac{b_s}{2b_\lambda} r + \frac{C_0}{b_\lambda r} \quad (7.47)$$

$$T_i - T_{i+1} = \frac{a_s}{9b_\lambda} (r_{i+1}^3 - r_i^3) + \frac{b_s}{4b_\lambda} (r_{i+1}^2 - r_i^2) + \frac{C_0}{b_\lambda} \log \frac{r_{i+1}}{r_i} \quad (7.48)$$

The constant C_0 has the same value as in the case with nonzero a_λ and b_λ .

Nonzero a, zero b

For the case where $b_\lambda = 0$, the solution of the heat equation is again simplified:

$$-\frac{\partial}{\partial r} \left[a_\lambda r^2 \frac{\partial T}{\partial r} \right] = a_s r^2 + b_s r \quad (7.49)$$

$$-a_\lambda r^2 \frac{\partial T}{\partial r} = \frac{a_s}{3} r^3 + \frac{b_s}{2} r^2 + C_0 \quad (7.50)$$

$$-\frac{\partial T}{\partial r} = \frac{a_s}{3a_\lambda} r + \frac{b_s}{2a_\lambda} + \frac{C_0}{a_\lambda r^2} \quad (7.51)$$

$$T_i - T_{i+1} = \frac{a_s}{6a_\lambda} (r_{i+1}^2 - r_i^2) + \frac{b_s}{2a_\lambda} (r_{i+1} - r_i) - \frac{C_0}{a_\lambda} \left(\frac{1}{r_{i+1}} - \frac{1}{r_i} \right) \quad (7.52)$$

Again, the constant C_0 is determined with Eq. 7.44.

Gas gap and cladding outer surface

Now we can use equations 7.35, 7.48 and 7.52 to determine the inner temperature for each element based on the outer temperature of the element, the linearized thermal conductivity in the element and the linearized power density distribution in the pellet. Since there are two elements whose outer temperature is not the inner temperature of the following element, namely the pellet surface element and the cladding outer surface element, we'll need additional relations to obtain the outer temperatures for these two elements.

In the gas gap we'll use the gas gap heat transfer coefficient (h_{gap}) calculated by FINIX to obtain the temperature difference across the gap:

$$(T_{\text{fo}} - T_{\text{ci}})h_{\text{gap}} = q''_{\text{fo}} \quad (7.53)$$

The heat flux at the pellet outer surface is calculated simply as

$$q''_{\text{fo}} = \frac{q'}{2\pi r_{\text{fo}}}, \quad (7.54)$$

where q' is the axial zone linear heat rate and r_{co} is the pellet outer radius. The pellet surface temperature is then simply

$$T_{\text{fo}} = T_{\text{ci}} + \frac{q'}{2\pi r_{\text{fo}} h_{\text{gap}}} \quad (7.55)$$

The cladding outer surface may be given as a boundary condition. In the case, where the cladding outer surface temperature is not given as a boundary condition, we'll utilize the (user given) bulk coolant temperature (T_{cool}) and the heat transfer coefficient between the cladding and the coolant (h_{cool}) either given by user or calculated by FINIX:

$$(T_{\text{co}} - T_{\text{cool}})h_{\text{cool}} = q''_{\text{co}} \quad (7.56)$$

The heat flux at the cladding outer surface is calculated simply as

$$q''_{\text{co}} = \frac{q'}{2\pi r_{\text{co}}}, \quad (7.57)$$

where q' is the axial zone linear heat rate and r_{co} is the cladding outer radius. The cladding surface temperature is then simply

$$T_{\text{co}} = T_{\text{cool}} + \frac{q'}{2\pi r_{\text{co}} h_{\text{cool}}}. \quad (7.58)$$

Assembling the matrix equation

Based on the derivations done in the previous chapter we can assemble a matrix equation for the temperatures at the nodes. The coefficient matrix will have ones on the diagonal and minus ones on the first superdiagonal:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{n_f+n_c-1} \\ T_{n_f+n_c} \end{bmatrix} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_{n_f+n_c-1} \\ l_{n_f+n_c} \end{bmatrix} \quad (7.59)$$

The matrix equation can be solved by calculating the constants on the RHS and using backwards substitution.

The constants on the RHS are as follows:

Elements in fuel pellet. For $l_i, i \in (0, \dots, n_f - 1)$, we'll use the solutions from equations 7.35, 7.48 and 7.52. In the case with $a_\lambda \neq 0$ and $b_\lambda \neq 0$ we have

$$\begin{aligned} l_i &= \frac{a_s}{6a_\lambda} (r_{i+1}^2 - r_i^2) + \frac{3b_s a_\lambda - 2b_\lambda a_s}{6a_\lambda^2} (r_{i+1} - r_i) \\ &\quad - \frac{3b_\lambda b_s a_\lambda - 2b_\lambda^2 a_s}{6a_\lambda^3} \log \frac{\lambda_{i+1}}{\lambda_i} \\ &\quad + \frac{C_0}{b_\lambda} \log \frac{r_{i+1}}{r_i} - \frac{C_0}{b_\lambda} \log \frac{\lambda_{i+1}}{\lambda_i}. \end{aligned} \quad (7.60)$$

For the case with $a_\lambda = 0$ and $b_\lambda \neq 0$ we have

$$l_i = \frac{a_s}{9b_\lambda} (r_{i+1}^3 - r_i^3) + \frac{b_s}{4b_\lambda} (r_{i+1}^2 - r_i^2) + \frac{C_0}{b_\lambda} \log \frac{r_{i+1}}{r_i} \quad (7.61)$$

and for the case with $a_\lambda \neq 0$ and $b_\lambda = 0$ we have

$$l_i = \frac{a_s}{6a_\lambda} (r_{i+1}^2 - r_i^2) + \frac{b_s}{2a_\lambda} (r_{i+1} - r_i) - \frac{C_0}{a_\lambda} \left(\frac{1}{r_{i+1}} - \frac{1}{r_i} \right). \quad (7.62)$$

In all of the cases C_0 is either calculated using Eq. 7.44 or set to zero in the case of $r_i = 0$. One should note that C_0 has to be calculated separately for each element.

Gas gap element. For d_{n_f} we'll use the equation 7.55 to get

$$l_{n_f} = \frac{q'}{2\pi r_{fo} h_{gap}}. \quad (7.63)$$

Elements in cladding. For the cladding elements (i.e., $l_i, i \in (n_f + 1, \dots, n_f + n_c)$), the constants on the RHS are calculated similar to pellet elements, barring the fact that the power distribution is zero in the element. Equations 7.60–7.62 can be simplified. In the case with $a_\lambda \neq 0$ and $b_\lambda \neq 0$

$$l_i = \frac{C_0}{b_\lambda} \log \frac{r_{i+1}}{r_i} - \frac{C_0}{b_\lambda} \log \frac{\lambda_{i+1}}{\lambda_i}. \quad (7.64)$$

For the case with $a_\lambda = 0$ and $b_\lambda \neq 0$ we have

$$l_i = \frac{C_0}{b_\lambda} \log \frac{r_{i+1}}{r_i} \quad (7.65)$$

and for the case with $a_\lambda \neq 0$ and $b_\lambda = 0$ we have

$$l_i = -\frac{C_0}{a_\lambda} \left(\frac{1}{r_{i+1}} - \frac{1}{r_i} \right). \quad (7.66)$$

Again, C_0 is calculated for each element using Eq. 7.44.

Cladding outer surface node. For the cladding outer surface we have, depending on the boundary condition, either

$$l_{n_f+n_c} = T_{co} \quad (7.67)$$

if the cladding outer surface T_{co} is given as a boundary condition or

$$l_{n_f+n_c} = T_{cool} + \frac{q'}{2\pi r_{co} h_{cool}} \quad (7.68)$$

in the case of other boundary conditions.

7.3 Plenum temperature

The plenum temperature model described in Sec. 6.5 can be solved self-consistently for fixed plenum pressure and cladding and pellet surface temperatures. However, since the properties of the fill gas and thereby the heat transfer coefficients depend on the temperature, the equations do not have a closed form solution. Instead, the solution is found iteratively with the secant method [30]. The area-averaged temperature $T_0 = (A_p T_p + A_c T_c) / (A_p + A_c)$ is used as the initial guess. Typically, the iteration requires two steps or less to converge within the numerical tolerance.

7.4 Mechanical model

7.4.1 Rigid pellet model

The primary assumption of the rigid pellet approximation is that the pellet is rigid (hard) enough to be completely non-deformable under external stresses. Thus, the displacement of the pellet nodes and the outer surface can be calculated directly from the strain correlations, without iteration.

7.4.2 Cladding model

The employed cladding mechanical model depends on the type of contact between the pellet and the cladding. In principle, iteration of the mechanical solution with internal pressure is only necessary when the gap remains open; if the gap is closed, the gap volume is fixed and therefore does not affect the pressure. However, the model typically consists of several axial nodes, and the pressure is function of the displacements in all of those nodes. In some of the nodes, the gap may remain open while in others it is closed (due to, *e.g.*, nonuniform axial power distribution). If the gap remains open even in some of the nodes, the internal pressure has to be solved iteratively. This iteration process then changes the boundary conditions in the closed gap model, which has to be re-solved. Therefore, even the closed gap models are solved several times because of the iteration of the mechanical solution of the full rod. Note that the plastic strain increments calculated on each iteration are only saved to the total plastic strains after the whole solution has converged (at the end of the time step).

In practice, the iteration is performed by first calculating an initial guess for the pressure (using the displacements from, *e.g.*, previous time step), then solving the mechanical model for each axial node independently with fixed pressure, and finally re-calculating the pressure with updated displacements. The process is repeated, using the secant method to predict the pressure, until the internal pressure converges. Optionally, one can also use either the bisection or false position methods for the internal pressure prediction through an option implemented in the source code.

7.5 Oxidation model

The cladding oxidation model calculates the cladding outer surface oxide thickness on every timestep, as long as the rod doesn't reach a critical oxidation level at which it stops. In a LOCA it is plausible that the cladding entirely oxidizes. However, in this case the FINIX assumptions would fail in any case, so a critical oxidation thickness of the cladding is set for the oxidation model to stop calculating further oxidation. This implementation was selected because of numerical reasons.

The oxide layer is treated as a separate node, which doesn't participate in the mechanical solution. The oxide nodes are otherwise similar to cladding nodes, except they have different correlations for their thermal properties. As the thickness of the oxide layer is updated, the change in the oxide volume is calculated, and the enthalpy of this chemical reaction is solved, considering it as an additional source of heat for the oxidation node.

Both the cladding inner and outer surface oxide thicknesses are initially solved either from the input, or by estimating an initial thickness of $2 \cdot 10^{-8}$ m. The initial growth rate might be faster, if the oxide thickness is below the transition thickness. Thus there are two growth modes for the oxidation: pre-transition and post-transition growth. Both of these are mainly dependent on the temperature of the cladding.

7.6 Coolant model

Coolant model was updated in FINIX-1.19.1. Although the model isn't important when FINIX is coupled with a thermohydraulics codes, it is required for the stand-alone performance of FINIX and when coupled to neutronics codes. In the current version, the inlet temperature and pressure of the coolant are given by the user. At each timestep, these are used to calculate the density and heat capacity of the coolant, which further help to solve the viscosity and thermal conductivity of the coolant. After this, the coolant-cladding heat transfer coefficient is calculated. This heat transfer coefficient simulates the flow of the heat out of the rod, thus impacting the temperatures in the pellet and cladding as well. Finally, the temperature distribution of the coolant is calculated utilizing the inlet temperature and the coolant heat capacity. This produces the axially growing temperature distribution in the coolant, which in turn leads into better oxidation results.

7.7 Fission gas release

The fission gas release model is connected to the internal pressure calculation through the gas mole amount and the thermal model through the fill gas composition.

On each iteration, a gas mole amount increment is calculated, and it is added to the gas mole amount only on convergence. During the iteration, the gas mole amount on each iteration is thought to consist of the gas mole amount plus its increment, which varies between iterations.

The fill gas composition is calculated for the sum of the gas mole amount and its increment on each iteration, and the composition on the previous time step is saved in FINIX internal data structures between time steps.

7.8 Other numerical features

7.8.1 Automatic timestepping

Automatic timestepping in steady-state cases was implemented in version 1.18.9. The timestepping algorithm is very simple, and uses the power history to calculate time steps according to the power history steps given. If the power history time step is larger than a set limit, the time step is halved until the value is smaller than the maximum allowed time step. This allows for the user to not give the time step history twice, first in the power history, and then as a separate time step history as was the case in the previous versions.

Additionally, the steady-state solver halves the time step if a solution is not reached in the set amount of iteration steps. This makes the calculation numerically more reliable, as the calculation does not fail even if the given time step is too large.

7.8.2 Renodalization in restarted calculation

If the amount of axial and radial nodes remains unchanged before and after restart, the restart-file will be simply read in without further processing. This is recommended in situations where restart is simply used to continue an unfinished simulation with the original inputs (such as power histories). However, should the nodalization change, the data in the restart-file will be interpolated to provide approximate values for the new nodalization.

The variable fields of the data structures are present in three forms. Single values, such as gas fractions, will apply to the rod similarly regardless of the nodalization. 1D arrays, such as cladding and pellet strains, are mostly lists of variable values at different axial or radial coordinates. These will be interpolated linearly, with the value of the first and the last node remaining same in both nodalizations. 2D arrays, such as temperature and burnup, are lists of

values that depend on both axial and radial coordinate. These have to be interpolated bilinearly. FINIX calculates bilinear interpolation using the following method:

$$f(r, z) \approx \frac{1}{(r_2 - r_1)(z_2 - z_1)} (f(Q_{11})(r_2 - r)(z_2 - z) + f(Q_{21})(r - r_1)(z_2 - z) + f(Q_{12})(r_2 - r)(z - z_1) + f(Q_{22})(r - r_1)(z - z_1)), \quad (7.69)$$

where Q_{ij} are the points (z_i, r_i) of the known values. After renodalization, matrix `bilinear_array` is used to store two integer values for each of the new nodes. These values correspond to the nodal coordinates of the bottom-left node in the rectangle composed of four old nodes that contains the new node. This is visualized in figure 3 where the dotted line points out the old node which is used to form the rectangle. Essentially, this information is used to determine which four nodes are used to determine the bilinearly interpolated value for each 2D quantity of each new node.

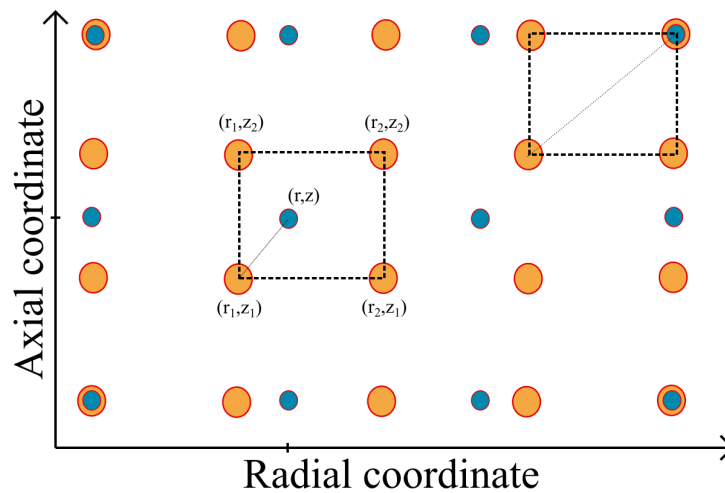


Figure 3. Bilinear interpolation after renodalization. Orange spheres represent the old nodalization, whereas the blue spheres represent the new nodalization.

8. Usage instructions

8.1 General description

The FINIX fuel behavior model and interface is designed to work primarily as a subprogram within a larger simulation code, to provide or replace the existing fuel performance model or subroutines. In this document, this larger code is referred to as the *host code*.

The purpose of FINIX is to provide the host code all the required fuel behavior subroutines via an interface that integrates directly with the host code on the source code level, and only requires a limited number of function calls between FINIX and the host. All the data between the host code and FINIX is passed as arguments of the function calls; no input or output files for the data exchange are needed. This makes execution of the subroutines faster, because disk access is minimized, and also allows the user of the host code to specify which output from FINIX (if any) is saved as a file.

Although the communication between FINIX and the host code is handled without input/output files, since FINIX-0.13.9 it has been possible to initialize FINIX for accumulated burnup using FRAPCON-generated FRAPTRAN restart files, and since version 0.15.6 it has been possible to initialize FINIX data structures by using input files.

FINIX has an error message system, which is designed to provide the host code run-time errors and warnings. For example, convergence failures or invalid correlation parameters would be passed as an error message to the host code. A more detailed description of the error message system is given below.

The setup and usage of the FINIX model is done via built-in functions, which can be used to provide default values for system parameters, give the spatial discretization and, to solve the thermo-mechanical model. A more thorough walkthrough of the procedure is given below, in Sec. 8.6. The source code also includes an example file, *host.c*, which provides FINIX with the necessary run-time data and shows how the model is initialized and run.

A more detailed documentation of the source code, with function descriptions and dependencies, is also given as a separate document automatically generated by Doxygen.

8.2 Units

FINIX functions always use the base SI units in both input arguments and in output. For example, distances are given in meters (m), temperatures are given in Kelvin (K) and power, linear power, and power density are given in W, $\text{W}\cdot\text{m}^{-1}$ and $\text{W}\cdot\text{m}^{-3}$, respectively. The user is responsible for writing any unit conversion functions between the host code and FINIX.

In general, the data in FINIX input files should be given in SI units as well. However, time-related data can be given in seconds, hours or days. If the input will be read from a FRAPTRAN input file, it is possible to choose either British or SI units.

In FINIX, the term "power" refers primarily to thermal power, as opposed to fission power. Currently the code makes no distinction between the two. However, in principle the user should

always supply FINIX with the *thermal* power history, including the decay heat of the fission products.

The cold state in FINIX specifies the reference temperature for the fill gas properties and thermal strains. The cold state for thermal strains is defined as 300 K, and the cold state for fill gas properties can be set in the input.

8.3 Data structures

The main purpose of FINIX is to provide the host code the temperature distribution and spatial deformations of the pellet and cladding by using the power history and coolant conditions provided by the host code as input. In addition to these primary data, there are data that are either required by FINIX as parameters, or are generated by solution of the thermo-mechanical model. Although much of this may be irrelevant to the host code, this data is required to continue the FINIX simulation for several consecutive time steps. All the data that FINIX uses or produces is stored in six structures. These structures are described below.

8.3.1 Main structures

The structure called **Rod** contains the data that describes the physical properties of the fuel rod. These properties include rod dimensions, pellet properties, cladding properties, gas properties and bundle dimensions. This data will not change during simulation. The contents of the Rod structure can be read from the corresponding input file.

The **Boundary_conditions** structure is used for storing the boundary condition data needed by FINIX. This data includes variables that specify the simulation time, power data, and the data related to cladding and coolant conditions. For each FINIX calculation time step, the Boundary_conditions structure needs to be updated either by the host code, or in the FINIX main program (in *host.c*).

The **Scenario** structure contains power profile and irradiation history data that will be read from an input file if needed. Once this data has been read from an input file, it will not change during simulation. The history data from the Scenario structure should be transferred to Boundary_conditions structure at each time step. If the host code provides the boundary condition data, the Scenario structure is not needed. The contents of the Scenario structure can be read from the corresponding input file.

The **Results** structure contains the simulation results calculated by FINIX that are relevant to the end user.

The **Options** structure contains the variables that specify the nodalization and the calculation options. This data will not change during simulation. The contents of the Options structure can be read from the corresponding input file.

The **Workspace** structure contains internal FINIX arrays that FINIX uses for internal calculations and variables that FINIX calculates but are not that relevant for the end user to know. Some of the values are saved between time steps so that time derivatives for various models can be calculated, and some values are cumulative. The structure also allows for efficient memory allocation in FINIX.

8.3.2 Auxiliary structures

Some structures are used to store similar data across FINIX. These are mostly members of the **Results** structure, but are also found elsewhere.

The **Cylindrical** structure is used in storing values of results in each component in cylindrical coordinates, that is, values for strains and stresses in the mechanical model. Some members of the **Results** structure are Cylindrical structures. The **Cylindrical** structure contains members hoop, axial and radial, and functions for easy manipulation of these structures are implemented. With these functions one can, for example, copy the values of structures into another or perform simple calculations with them.

The **GasComposition** structure is used to store values related to fill gas components. Currently, it stores values for the following components: helium, argon, krypton, xenon, hydrogen, nitrogen and water vapor.

The **GasConcentration** structure is used to store the concentrations of fission gases at different microstructural locations. The stored values are the total produced fission gas, fission gas at the grain boundary, fission gas within the grains and the released fission gas. The local FGR at any node can be calculated as the ratio of the total produced and released fission gases.

The **Actinides** structure is used to store values related to the following actinide nuclides: ^{235}U , ^{238}U , ^{239}Pu , ^{240}Pu , ^{241}Pu and ^{242}Pu . The structure is used in members of the **Workspace** structure and in the radial power distribution model.

8.4 Function naming conventions

As of FINIX-0.17.12, the function naming scheme has been unified across the code. All function names begin with the prefix `finix_` to identify them as FINIX functions. Next is the source code file identifier, for example, `transient_` for `transient.c`. Finally, the description of the function is given. The adopted prefixes for each source code file are given in table 9. From this list, the source code file of each function can be easily found.

Table 9. Function name prefixes in different FINIX source code files.

Prefix	Source code file
<code>aux_</code>	<code>aux_functions.c</code>
<code>clmech_</code>	<code>clmech.c</code> , <code>clmechaux.c</code> , <code>clmechprop.c</code>
<code>clth_</code>	<code>clthprop.c</code>
<code>clox_</code>	<code>clox.c</code>
<code>coolant_</code>	<code>coolant.c</code>
<code>fgr_</code>	<code>fgr.c</code> , <code>fgrprop.c</code>
<code>calculate_</code>	<code>finix_calculate.c</code>
<code>data_</code>	<code>finix_data.c</code>
<code>init_</code>	<code>finix_initialization.c</code>
<code>output_</code>	<code>finix_output.c</code>
<code>fumech_</code>	<code>fumech.c</code> , <code>fumechprop.c</code>
<code>futh_</code>	<code>futhprop.c</code>
<code>gap_</code>	<code>gap.c</code>
<code>heateq_</code>	<code>heateq1d.c</code>
<code>steady_state_</code>	<code>steadystate.c</code>
<code>transient_</code>	<code>transient.c</code>
Special prefixes	
<code>construct_</code> , <code>destruct_</code> , <code>allocate_</code> , <code>append_</code> , <code>create_</code> , <code>free_</code>	

8.5 Error message system

FINIX uses an error message system to inform the user of issues such as convergence failures, parameters exceeding correlations' range of validity, etc. The function that encounters the issue returns the pointer to the error message. The message is then passed down, and further error messages are appended to it, until the message reaches the host code. It is then the responsibility of the host code to act on the error message by printing it on screen, writing it on disc, aborting the program execution, or otherwise. FINIX will not terminate the execution, if an issue is encountered. Only the error message will be returned.

The format of the error message is an array of text strings, formally of type `char**`. If no errors have occurred, then the top-level pointer will have a value of `NULL`. This can be used to distinguish between error-free and faulty operation of the FINIX functions. If the value is different from `NULL`, then the return value contains an error message. The returned array then contains pointers to the error message strings (of type `char*`). The last message is followed by a `NULL` pointer, which is used as a marker to terminate the message.

In practice, the user can use built-in functions to deal with FINIX the error messages. One merely needs to declare the pointer in the host code. For example, the following lines will declare the pointer, then call the FINIX transient solver function, catch the error message, print it on screen if it is not empty (*i.e.*, non-`NULL`), and finally free the memory allocated to the error message.

```
char **err=NULL;

err=finix_transient_solve(bc->dt, rod, bc, results, options, workspace);
if(err!=NULL){ finix_printf_err(err);}
finix_free_err(&err);
```

Two general functions can be used to check errors in values within functions. These are implemented for scalars and Cylindrical structures (`finix_output_check_scalar_inf_nan_sign()` and `finix_output_check_cylindrical_inf_nan()`). The function for Cylindrical structures checks for NaNs and infinities in the values. The function for scalars is more versatile: it automatically checks for infinities and NaNs, but it can also be used to check if the parameter is in a certain range. Table 10 shows the specific instructions for use of the scalar checking function.

Table 10. Instructions for use of the error checking function `finix_output_check_scalar_inf_nan_sign()`.

Input parameter	Expected content	Format
<code>param_name</code>	Name of the parameter	<code>char*</code>
<code>func_name</code>	Name of the function within which the error checking function is called	<code>char*</code>
<code>param</code>	The parameter to be checked	double
<code>range</code>	The range of the parameter to be checked, P = parameter must be positive, N = parameter must be negative, L = limits given separately	<code>char</code>
<code>llimit</code>	The lower limit of the range to be checked when range = L	double
<code>ulimit</code>	The upper limit of the range to be checked when range = L	double

8.6 System setup and simulation

8.6.1 Initialization

To use the FINIX code in a coupled system, the user must declare the necessary data structures in the host code, initialize the FINIX structures, and calculate the initial state of the simulation by using FINIX functions. After this the model can be run for as many time steps as needed. FINIX comes with an example host code file, *host.c*, which should be replaced in its entirety by the host code. The *host.c* file can be used as an example on how to setup the model. The minimal necessary steps to get FINIX up and running is described in this chapter.

In a coupled system the user is responsible for using an appropriate time step. Although the FINIX algorithms remain stable for very long time steps (longer than several days), discretization error can not be avoided. For relatively slow transients, a time step of the order of 1 millisecond should give very small discretization error during the transient, although for a fast RIA, a considerably smaller time step ($\delta t \approx 10^{-5}$ s) may be needed.

The declaration and the definition of the variables can be done as follows:

```
// Declare and construct FINIX data structures
Rod *rod = finix_construct_rod();
Boundary_conditions *bc = finix_construct_bc();
Scenario *scenario = finix_construct_scenario();
Results *results = finix_construct_results();
Options *options = finix_construct_options();
Workspace *workspace = finix_construct_workspace();

// Declare FINIX error message strings
char **finix_err=NULL, **new_err=NULL;
```

The next step is to initialize the system. The FINIX data structures can be initialized by calling the function *finix_init_data_structures()*. The function reads the input files given by the user, and transfers the data from the input files to the data structures. If no input files are given or some of the input is missing, the above constructor functions initialize the missing data with default values. As default, the data structure initialization function selects the rod, scenario and options to be used based on what identifier is given in the input file. Alternatively, one may select the rod, scenario and options to be used from within the host code with the function *finix_init_choose_input()*. When using this function, the keyword *from_host* must be used in the input files. The choices are given as follows, before the call to function *finix_init_data_structures()*:

```
// Choose the input file segment to use
finix_err = finix_init_choose_input(rod, "rodID", scenario, "scenID",
options, "optionsID");
if (finix_err != NULL) finix_printf_err(finix_err);
finix_free_err(&finix_err);
```

Before the actual simulation, the steady-state is solved so that FINIX internal structures for many of the parameters are initialized with more realistic values. For this purpose FINIX has the function *finix_steady_state_solve_initial()*, which solves the steady state heat equation and the corresponding mechanical equilibrium for the cladding. This function is a wrapper for the

actual steady state solver function, but in which some models are turned off and cumulative values are not updated.

FINIX can be initialized as follows:

```
// Initialize FINIX data structures
finix_err = finix_init_data_structures(rod, bc, scenario, results,
options, workspace);
if (finix_err != NULL) finix_printf_err(finix_err);
finix_free_err(&finix_err);

// Solve initial steady state
finix_err = finix_steady_state_solve_initial(rod, bc, results, options,
workspace);
if(finix_err != NULL) finix_printf_err(finix_err);
finix_free_err(&finix_err);
```

8.6.2 Transient simulation

The transient is solved in distinct time steps defined in the host code, by calling the *finix_transient_solve()* function. Before each function call, the power density, boundary conditions and other parameters and options can be changed. However, for one time step, *i.e.*, for one function call, they are constant. After each function call, the output of the FINIX model may be saved. For example, one can write FINIX summary files and node-specific files by calling a function *finix_output_print()*.

The transient can be solved for each time step as follows:

```
// Solve time step with finix_transient_solve()
finix_err=finix_transient_solve(bc->dt, rod, bc, results, options,
workspace);
if (finix_err != NULL) finix_printf_err(finix_err);
finix_free_err(&finix_err);

// Update the cumulative simulation time
bc->time += bc->dt;
```

8.6.3 Steady-state simulation

A steady-state simulation is performed similarly to the transient simulation. The steady-state is solved in distinct time steps defined in the host code, by calling the *finix_steady_state_solve()* function, as follows:

```
finix_err = finix_steady_state_solve(rod, bc, results, options, workspace);
if(finix_err != NULL) finix_printf_err(finix_err);
finix_free_err(&finix_err);

// Update the cumulative simulation time
bc->time += bc->dt;
```


In contrast with the transient simulation, the densities of fuel and cladding must be updated on each time step. The burnup must also be calculated if it is not provided by the host code. These are performed as follows:

```
// Calculate density
finix_err = finix_calculate_density(options, rod, results);
if (finix_err != NULL){ finix_printf_err(finix_err); }
finix_free_err(&finix_err);

// Calculate burnup
finix_err=finix_steady_state_calculate_burnup(bc->dt, options, rod,
results, bc, workspace);
if(finix_err!=NULL){ finix_printf_err(finix_err);}
finix_free_err(&finix_err);
```

8.6.4 Updating boundary conditions

As mentioned before, the above lines of code are the minimal setup that is needed to run FINIX in a coupled system. However, if the host code cannot provide all the necessary boundary conditions, the missing boundary conditions must be given in FINIX input file `finix_scenario.inp` or otherwise. If given through the input file, the boundary conditions must be updated for each time step by calling the function `finix_data_update_bc()` as follows:

```
// Update boundary conditions before calling finix_transient_solve()
finix_err = finix_data_update_bc(bc, scenario, options, rod, results,
workspace);
if(finix_err != NULL) finix_printf_err(finix_err);
finix_free_err(&finix_err);
```

When using the FINIX internal radial power distribution model, a call to the function `finix_steady_state_radial_power_distribution()` is required before a call to the solver functions. The radial power distribution model modifies the boundary conditions for the time step. The call is performed as follows:

```
finix_err = finix_steady_state_radial_power_distribution(rod, results,
bc, options, workspace);
if (finix_err != NULL){ finix_printf_err(finix_err); }
finix_free_err(&finix_err);
```

Note that the calculated actinide concentrations are only summed with the previous timestep values inside the solver functions.

8.6.5 Convergence criterion

The general convergence criterion used by FINIX is the variable `dhmax` in the solver functions `finix_transient_solve()` and `finix_steady_state_solve()`. `dhmax` is calculated as the maximum value in any axial node of the magnitude of the relative difference in gap conductance between iterations, or the maximum of the following for any z :

$$\Delta h_z = \left\| \frac{h_{1,z} - h_{0,z}}{h_{0,z}} \right\|, \quad (8.1)$$

where $h_{0,z}$ is the gap conductance at the previous iteration at axial node z , and $h_{1,z}$ the same for the current iteration.

The tolerance is set inside both functions as the variable `ITERTOL`. In both functions there are two similar possible exit points from the iteration loop:

```
if (dhmax < ITERTOL) {  
    // solution has converged  
    break;  
}
```

8.6.6 Ending the simulation

When the simulation has been completed and FINIX is no longer needed, the memory allocated for FINIX structures must be free'd. This can be done by using the function `finix_destruct_data_structures()`.

```
// Free allocated memory  
finix_destruct_data_structures(rod, bc, scenario, results, options,  
workspace);
```

8.7 Input instructions

FINIX uses three input files called `finix_options.inp`, `finix_rod.inp`, and `finix_scenario.inp`. These input files can be used to control the simulation. Each input file contains data for several fuel rods. When a new input is created for a new fuel rod, the input should be appended at the end of the files. An alternative way to control the simulation is to use FRAPTRAN input files.

In a coupled system, where all boundary conditions are given by the host code, no input files are required. If no input files are used, default values for simulation options and rod properties will be used. However, it is advisable to specify simulation options and rod properties in files `finix_options.inp` and `finix_rod.inp` (or otherwise in the host code) to get more realistic results. If the host code cannot provide all the necessary boundary conditions, i.e., the contents of the `Boundary_conditions` structure, the `finix_scenario.inp` file must be used to specify the missing boundary conditions.

If a stand-alone version of FINIX is used, the scenario boundary conditions must be given in file `finix_scenario.inp`. Again, it is advisable but not mandatory to give the simulation options and rod properties in files `finix_options.inp` and `finix_rod.inp`. Alternatively, input can be given in a FRAPTRAN input file. However, even though the rod and scenario parameters are given in a FRAPTRAN input file, it is advisable to specify FINIX simulation options in `finix_options.inp` because FINIX simulation options cannot be set in a FRAPTRAN input file. All the data given in FRAPTRAN input file overwrites the default data and the data read from FINIX input files.

The contents of FINIX input files is organized in data blocks. Each block of data describes one fuel rod. Therefore FINIX needs to know which block of data to read from a file. The data blocks are of the following form:

```
begin rod_name1
data_1 = x
data_2 = y
data_n = z
end rod_name1
```

As can be seen above, every data block begins with "begin" statement and ends with "end" statement. These statements are followed by the name of the rod. Note that the rod name must not include white spaces. As the data in input files has been grouped into blocks that describe each fuel rod, one should tell FINIX from which data block to read the data. This can be done with the "USE" statement as follows:

```
USE rod_name1
```

```
begin rod_name1
data_1 = x
data_2 = y
data_n = z
end rod_name1
```

```
begin rod_name2
data_1 = x
data_2 = y
end rod_name2
```

The "USE" statement tells FINIX that it should read the data from a data block that begins with "begin rod_name1" and ends with "end rod_name1". Every FINIX input file should begin with the "USE" statement.

If the data block is to be selected by the host code, the statement "USE from_host" must be used. When this statement is used, one may give the data block identifiers from within the host code with the function *finix_init_choose_input()*.

The data in input files can be arranged in any order. After a keyword, there should be an "=" sign followed by the data that corresponds to the keyword. If the data consists of several values, they should be separated with commas. The lines in input files that begin with "!", "*", or "/" are commented lines, and FINIX will ignore them. The contents of the three input files is described in the following chapters.

8.7.1 Input file finix_options.inp

Input file finix_options.inp can be used to set FINIX nodalization and to select the models FINIX uses during simulation. Table 11 shows the data that can be given in finix_options.inp. If default value will be used, it is not necessary to add that data in the input file. The contents of a file could look like this:

```
USE rod_123
```

```
begin rod_123
axial_nodes = 10
pellet_radial_nodes = 15
boundary_option = 0
end rod_123
```

Now the rod will be divided to 10 axial nodes and the fuel pellet will be divided to 15 radial nodes. The rod outer surface temperature will be given by the user. For the rest of the keywords default values shown in Table 11 will be used.

Input file `finix_options.inp` has also a special purpose. It can be used to tell FINIX that the input will be read from a FRAPTRAN input file. In this case the the first line of the file should read "USE FRAPTRAN". When FINIX simulation begins, it asks the user the name of the FRAPTRAN input file and the name of the FRAPCON generated restart file. If FRAPTRAN input file will be read, the contents of the `finix_options.inp` could look like this:

```
USE FRAPTRAN
```

or like this:

```
USE FRAPTRAN
```

```
begin FRAPTRAN
axial_nodes = 10
pellet_radial_nodes = 15
boundary_option = 0;
end FRAPTRAN
```

In the first case all the data will be read from a FRAPTRAN input file. In the second case nodalization and boundary option will be given in `finix_options.inp`. If nodalization is also given in FRAPTRAN input file, the data given in FRAPTRAN input file overwrites the data given in `finix_options.inp`. Note that the variables that specify FINIX model selections should always be given in `finix_options.inp`. If the input will be read from the FRAPTRAN input file, input files `finix_rod.inp` and `finix_scenario.inp` will be ignored.

8.7.2 Input file `finix_rod.inp`

Input file `finix_rod.inp` can be used to determine the fuel rod properties. Table 12 shows the data that can be given in `finix_rod.inp`. The contents of a file could look like this:

```
USE rod_123
```

```
begin rod_123
fuel_length = 4.0
clad_length = 4.0
plenum_length = 0.4
end rod_123
```

In this case default values will be used for all the parameters except for fuel, clad, and plenum length.

8.7.3 Input file `finix_scenario.inp`

Input file `finix_scenario.inp` can be used to determine the rod conditions during an irradiation or experiment period. As mentioned before, this data must be given if FINIX is used as a stand-alone version, or the host code cannot provide all the necessary boundary conditions. Table 13 shows the data that can be given in `finix_scenario.inp`. Note that it is not necessary to give all the data presented in Table 13 to fully determine the boundary conditions. For example, if the user has chosen to use coolant bulk temperature as boundary condition (`boundary_option = 3`), it is not necessary to determine the clad temperature history in `finix_scenario.inp`. The contents of the file `finix_scenario.inp` could look like this:

```
USE rod_xyz

begin rod_xyz
restartfile = imprestart.rod_xyz
end_time = 1.0

time_step_history =
0.0001, 0.0,
0.00001, 0.0702,
0.0001, 0.0815,
0.001, 0.210

average_power_history =
0.0, 0.0,
524934.384, 0.066,
23622047.28, 0.079,
5511811.032, 0.087,
262467.192, 0.095,
0.0, 0.1

axial_power_profile =
0.728, 0.0,
0.975, 0.099,
1.156, 0.259,
1.053, 0.420,
0.668, 0.569

coolant_temperature_history_zones = 0.373, 0.569
coolant_temperature_history(1) =
553.15, 0.0,
1023.15, 0.2,
1048.15, 0.4,
753.15, 1.0
coolant_temperature_history(2) =
553.15, 0.0,
878.15, 0.2,
923.15, 0.4,
948.15, 1.0

heat_transfer_coefficient_history_zones = 0.569
heat_transfer_coefficient_history(1) = 2000000.0, 0.0

coolant_pressure_history = 0.5e6, 0.0
end rod_xyz
```

As can be seen in the example above, the history data is always given in data pairs. For example, the time step history is given so that the first value of the data pair is the length of the time step, and the second value is the time at which this time step length takes effect. This time step length is used until a new data pair is given. Similar logic applies to other history data as well.

All time-related FINIX input can be given in seconds, hours or days. The units of the time can be selected by modifying the value of `time_unit` in `finix_scenario.inp`. Note that all the time-related data in Table 13 is expressed in seconds (default) only for clarity. The time units of the data shown in Table 13 depends on the value of `time_unit`. The value of `time_unit` also determines the units of the time in output files `finix.z*` and `finix.sum`.

In the example above one can also see that some of the history data, such as the coolant temperature history, can be given separately for different axial zones. In this case the coolant temperature history has been given for axial zones 1 and 2. The number of zones does not have to match the number of axial nodes given in `finix_options.inp`. FINIX will automatically calculate history data for each axial node based on the data given for each zone. Note that one should also determine the top elevations of each zone from the rod bottom. In the example above, the top of the first coolant temperature history zone is 0.373 m above the rod bottom, while the top of the second zone is 0.569 m above the rod bottom.

Table 11. Data that can be given in *finix_options.inp*

Keyword	Default	Description
axial_nodes	11	Number of axial nodes.
pellet_radial_nodes	17	Number of radial nodes in fuel pellet.
clad_radial_nodes	5	Number of radial node in the cladding.
boundary_option	3	Specifies the type of the boundary conditions. 0 = user-given rod outer surface temperature, 1 = user-given heat flux between rod outer surface and coolant, 2 = user-given heat transfer coefficient and coolant bulk temperature, 3 = user-given bulk temperature and calculated heat transfer coefficient from internal correlations (needs inlet mass flux).
temperature_iteration	1	Temperature iteration. 0 = off, 1 = on.
gap_conductance_model	3	Gap conductance model. 1 = FRAPCON-3.4 model, 2 = FRAPTRAN model (recommended), 3 = FRAPCON-4.0 model, -n = negative value switches off contact conductance
clad_elasticity_model	1	Clad elasticity model. 0 = off, 1 = on.
plenum_model	1	Plenum model. 0 = off, 1 = on.
pellet_relocation_model	1	Pellet relocation model. 0 = off, 1 = on.
clad_plasticity_model	1	Cladding (time-independent) plastic deformation model. 0 = off, 1 = on.
clad_creep_model	2	Cladding creep model. 0 = off, 1 = Limbäck-Andersson, 2 = Geelhood.
FGR_model	1	Fission gas release model. 0 = off, 1 = on.
swelling_model	1	Pellet swelling model. 0 = off, 1 = MATPRO solid and gaseous, 2 = MATPRO solid, Pastore et al. gaseous
densification_model	1	Pellet densification model. 0 = off, 1 = on.
radial_power_distribution_model	1	Radial power distribution model. 0 = off, 1 = on.
failure_model	1	Cladding failure model. 0 = off, 1 = on.
oxidation_model	1	Cladding oxidation model. 0 = off, 1 = on.
restart_print	0	Option to print restart file. 0 = off, N = every Nth time step.
refabrication	0	Option to change rod gas contents after restart, 0 = off, 1 = on, read gas contents from <i>finix_rod.inp</i> .
verbose_input	1	Option to print out input file contents, 0 = off, 1 = on.

Table 12. Data that can be given in finix_rod.inp

Keyword	Units	Default	Description
rodID	-	-	An identifier for the rod for use by the host code. Length limit is 100 characters.
pellet_inner_radius	m	0.0	Radius of fuel pellet center hole.
pellet_outer_radius	m	0.0047	Pellet outer radius.
clad_inner_radius	m	0.00479	Fuel cladding inner radius.
clad_outer_radius	m	0.00546	Fuel cladding outer radius.
fuel_length	m	3.6576	Length of rod fuel column.
clad_length	m	3.6576	Length of fuel cladding wall in active area.
plenum_length	m	0.2	Plenum length.
pellet_roughness	m	$2.0 \cdot 10^{-6}$	Arithmetic mean roughness of fuel pellet surface.
fractional_density	-	0.938	Fractional theoretical density of fuel pellet.
grain_size	m	$10.0 \cdot 10^{-6}$	Average 3D grain size (diameter) of fuel pellet.
enrichment	-	0.03	Fraction of uranium in the fuel of the isotope ^{235}U .
gadolinia_weight_fraction	-	0.0	Weight fraction of gadolinia (Gd_2O_3) in fuel pellets.
clad_roughness	m	$0.5 \cdot 10^{-6}$	Arithmetic mean roughness of cladding inner surface.
coldwork	-	0.5	Reduction of cross-sectional area of cladding by cold working process.
clad_oxygen_concentration	-	0.0012	Cladding average oxygen concentration.
fast_neutron_fluence	$\text{n}\cdot\text{m}^{-2}$	0.0	Fast neutron fluence that the cladding was exposed to during lifetime.
clad_type	-	0	Cladding material identifier. 0 = Zircaloy, 1 = Zr1%Nb.
fill_gas_pressure	$\text{N}\cdot\text{m}^{-2}$	$1.207 \cdot 10^6$	As-fabricated fill gas pressure.
fill_gas_temperature	K	300.0	As-fabricated fill gas temperature.
gas_fraction_X	-	1.0, X = He	Fraction of gas that is X, where X = He for helium, Ar for argon, Kr for krypton, Xe for xenon, H2 for hydrogen, N2 for nitrogen or H2O for steam.
pitch	m	$14.43 \cdot 10^{-3}$	Center-to-center spacing of fuel rods.
rods_in_unit_cell	-	1.0	Number of rods in one unit cell (1.0 for square lattice, 0.5 for triangular lattice).

Table 13. Data that can be given in finix_scenario.inp

Keyword	Units	Default	Description
restartfile	-	-	Name of the Frapcon generated restart file (omit if there is no need to read a restart file).
steady_state_simulation	-	0	Type of the scenario. 0 = transient, 1 = steady-state.
time_unit	-	0	Units of all time-related FINIX input. Determines also the units of time in output files finix.z* and finix.sum. 0 = seconds, 1 = hours, 2 = days.
end_time	s	0.15	Simulation end time.
time_step_history	s, s	0.0001, 0.0	Time step history. The first value is the size of the timestep. The second value is the time at which this time step size takes effect. Each time step size is used until a new data pair is given.
power_factor	-	1.0	Multiplier for linear power.
average_power_history	W·m ⁻¹ , s	0.1e6, 0.0	Rod average linear heat generation rate history. Each linear heat generation rate is used until a new data pair is given.
power_history_zones	m	-	Top elevation of each power history zone. Enter as many values as there will be power history zones.
power_history(n)	W·m ⁻¹ , s	0.1e6, 0.0	Power and time data pairs for each power history zone. Each linear power will be used until a new data pair is given. Enter as many data sets as there are power history zones. power_history(1) starts input for zone 1, power_history(n) starts input for zone n.
axial_power_profile	-, m	-	Axial power profile. The first value is the axial power factor normalized to rod-average. The second value is the node top elevation beginning from the rod bottom. Begin inserting data pairs from the rod bottom towards the top, until the the axial power profile is fully defined.
radial_power_profile	-, m	-	Radial power profile for all axial nodes. The first value is the radial power factor. The second value is the distance from the fuel centerline to the radial node periphery. Begin inserting data pairs from fuel centerline to the edge.
coolant_temperature_history_zones	m	-	Top elevation of each coolant temperature history zone. Enter as many values as there will be coolant temperature history zones.

Table 13 (continued). Data that can be given in *finix_scenario.inp*

Keyword	Units	Default	Description
coolant_temperature_history(n)	his- K, s	561.0, 0.0	Coolant temperature and time data pairs for each coolant temperature history zone. Each temperature will be used until a new data pair is given. Enter as many data sets as there are coolant temperature history zones. coolant_temperature_history(1) starts input for zone 1, coolant_temperature_history(n) starts input for zone n.
clad_temperature_history_zones	m	-	Top elevation of each clad temperature history zone. Enter as many values as there will be clad temperature history zones.
clad_temperature_history(n)	K, s	561.0, 0.0	Clad temperature and time data pairs for each clad temperature history zone. Each temperature will be used until a new data pair is given. Enter as many data sets as there are clad temperature history zones. clad_temperature_history(1) starts input for zone 1, clad_temperature_history(n) starts input for zone n.
heat_transfer_coefficient_history_zones	m	-	Top elevation of each heat transfer coefficient history zone. Enter as many values as there will be heat transfer coefficient history zones.
heat_transfer_coefficient_history(n)	W·m ⁻² ·K ⁻¹ , s	2e4, 0.0	Heat transfer coefficient and time data pairs for each heat transfer coefficient history zone. Each heat transfer coefficient will be used until a new data pair is given. Enter as many data sets as there are heat transfer coefficient history zones. heat_transfer_coefficient_history(1) starts input for zone 1, heat_transfer_coefficient_history(n) starts input for zone n.

Table 13 (continued). Data that can be given in *finix_scenario.inp*

Keyword	Units	Default	Description
heat_flux_history_zones	m	-	Top elevation of each heat flux history zone. Enter as many values as there will be heat flux history zones.
heat_flux_history(n)	W·m ⁻² , s	0.0, 0.0	Heat flux (between rod outer surface and coolant) and time data pairs for each heat flux history zone. Each heat flux will be used until a new data pair is given. Enter as many data sets as there are heat flux history zones. heat_flux_history(1) starts input for zone 1, heat_flux_history(n) starts input for zone n.
coolant_pressure_history	N·m ⁻² , s	15.51e6, 0.0	Enter coolant pressure and time data pairs. Each value of pressure is used until a new data pair is given.
coolant_mass_flux_history	kg·m ⁻² ·s ⁻¹ , s	3460.0, 0.0	Enter coolant mass flux and time data pairs. Each value of mass flux is used until a new data pair is given.

8.8 Output files

FINIX includes functions that can be used to print output files for many purposes. One aim of this chapter is to describe the contents of these output files. In addition, this chapter shows how to call the output writing functions from the host code.

8.8.1 finix.sum and finix.z*

The data describing the behaviour of a fuel rod as a function of time is presented in output files finix.sum and finix.z*. The contents of these files is presented in Tables 14 and 15. Output file finix.sum contains rod summary data, while output files finix.z* contain node-specific data. Node-specific output files are written for each axial node. Asterisk (*) in the aforementioned file names represents the node number.

By default, the cumulative simulation time in files finix.sum and finix.z* is given in seconds. However, the simulation time in the aforementioned files will be given in hours or days if a non-default value is given for keyword "time_unit" in finix_scenario.inp. In other words, the same units of time are used in files finix_scenario.inp, finix.sum and finix.z*.

To print these output files, the following additional lines of code must be included in the host code:

```
// Declare a structure containing file pointers to output files
Output *files ;
```

```
// Initialize output writing to finix.sum and finix.z files
files = finix_output_initialize(options);
```

In the initialization call above the type of the function parameter is Options*. After declarations and initializations the output writing function can be called for each time step as follows:

```
finix_output_print(files , rod , bc , options , results);
```

Here the types of the function parameters are Output*, Rod*, Boundary_conditions*, Options*, and Results*. After the output files have been printed, they must be closed and the memory must be free'd. This can be done as follows:

```
// Close files and free allocated memory
finix_output_close(files , options);
```

8.8.2 finix_data_structures.dbg

Output file finix_data_structures.dbg shows all the data stored in FINIX data structures at the time of the output writing call. The function is called during FINIX initialization by default, but it can be called later again if needed. The output file is especially useful for checking that the input files have been read correctly. The output file finix_data_structures.dbg can be printed by calling a function

```
finix_output_print_data_structures(rod , bc , scenario , results , options);
```

Table 14. Contents of the output file finix.sum

Column name	Units	Description
Step	-	Time step number.
Time	s or h or d	Cumulative simulation time.
Buav	MWd·kg _U ⁻¹	Rod average burn-up.
Qav	W·m ⁻¹	Average linear power.
FGR%av	%	Percentage released fission gas.
Intpr	N·m ⁻²	Rod fill gas pressure pressure.
Coolpr	N·m ⁻²	Coolant pressure.
Fuext	m	Fuel axial elongation.
Clect	m	Cladding axial elongation.
Tplen	K	Fill gas temperature.
Tzon	-	Axial node where maximum temperature occurs.
Qlo	W·m ⁻¹	Maximum local linear power.
BUlo	MWd·kg _U ⁻¹	Maximum local burn-up.
Tmax	K	Maximum temperature.
FGR%lo	%	Maximum local fission gas release.
Gap	m	Average pellet-cladding gap width.
Gapcon	W·m ⁻² ·K ⁻¹	Gap average conductance.
Tclav	K	Clad average temperature.
Tcool	K	Coolant average temperature.
Tcentav	K	Fuel centreline average temperature.
tfail	s	Time of cladding failure.

Table 15. Contents of the output file *finix.z**

Column name	Units	Description
Step	-	Time step number.
Time	s or h or d	Cumulative simulation time.
Burnup	MWd·kg _U ⁻¹	Average fuel burn-up in node.
Linrat	W·m ⁻¹	Node linear power.
Tcool	K	Coolant temperature at the node elevation.
Tclou	K	Temperature at the cladding outer surface.
Tclav	K	Cladding average temperature.
Tclin	K	Temperature at the cladding inner surface.
Tfout	K	Temperature at the pellet surface.
Tfav	K	Pellet average temperature.
Tcent	K	Fuel centerline temperature.
FGR%lo	%	Fission gas release.
Gap	m	Pellet-cladding gap width in node.
DTgap	K	Temperature difference over gap.
Gapcon	W·m ⁻² ·K ⁻¹	Gap conductance.
Conpr	N·m ⁻²	Pellet-cladding contact pressure.
Hoopstrs	N·m ⁻²	Clad hoop stress.
Dradcl	m	Change in cladding radius.
Buav	MWd·kg _U ⁻¹	Rod average burn-up.

where the function parameter types are Rod*, Boundary_conditions*, Scenario*, Results*, and Options*.

8.8.3 finix_stripfile.txt

Output file finix_stripfile.txt contains data in FRAPTRAN stripfile format. The file describes the state of the fuel rod at various time steps.

To print this output file, the following additional lines of code must be included in the host code:

```
// Declare a variable for printing the results
FILE *writefile ;

// Open the file
writefile = fopen(" finix_stripfile .txt ","w");

// Print the header lines
finix_output_fprintf_stripfile(0, writefile , bc->time , results , rod , options , bc
```

After these steps the output writing function can be called for each time step as follows:

```
finix_output_fprintf_stripfile(1, writefile , bc->time , results , rod , options , bc
```

After the file has been printed, it must be closed:

```
fclose ( writefile );
```

8.8.4 finix_restart

A restart feature was implemented in FINIX-1.19.12. The feature enables the user to save all the current values in Options, Results and Workspace structures into a file. This file can be used to continue the simulation from the time step at which the restart file was written. Furthermore, this feature allows the user to run both the base irradiation and transient phases of a rod using FINIX, whereas FRAPCON had to be used previously to provide the state of the fuel at end of base irradiation for the transient phase.

Printing the restart file takes a considerable amount of time in the scale of a fuel simulation, and therefore it shouldn't be done at each and every time step. A new option `restart_print` allows the user to give an integer n ($n > 0$) in the `finix_options.inp` input file, setting FINIX to print out the restart file at every n th time step.

To use the restart file in the FINIX initialization, the following lines must be included in the `finix_scenario.inp` input file:

```
restartfile = finix_restart  
restarttype = 1
```

where the first line determines the name of the restart file (in this case `finix_restart`) and the second line determines which type of restart will be used: restart type of 1 sets FINIX to read in the FINIX format restart file, whereas any other value instructs FINIX to read in the FRAPCON format restart file.

9. Code assessment

9.1 General performance

The validation of FINIX-1.19.1 is presented in a separate validation report [1], where the detailed results can be found. The current version FINIX-1.19.12 has not yet been validated, but will be performed in the near future.

FINIX-1.19.1 includes several models important for the accurate description of long irradiations. Some limitations are still present. FINIX-1.19.1 has been compared against experimental centerline temperature data from Halden steady state irradiation experiments IFA-429, IFA-432, IFA-515, IFA-677 and IFA-681. The agreement between simulated and experimental results is good. The FINIX calculated values and the experimental values typically agree within roughly 10 %, with FINIX having a slight tendency to underestimate the centerline temperature.

FINIX-1.19.1 was also validated against FRAPTRAN simulations of selected reactivity initiated accidents and a limited amount of loss-of-coolant accidents. The results for FRAPTRAN are described in the FRAPTRAN code assessment document [67], while the FINIX results and the comparison are discussed in the FINIX-1.19.1 validation report [1]. The RIA cases consisted of western and VVER type fuel rods, some of which had failed during the experiment, and some had not. In some of scenarios significant plastic deformation of the cladding was indicated by FRAPTRAN, while in some very little permanent deformation occurred. All the cases were initialized for non-fresh fuel using FRAPCON for steady state irradiation. The comparison between FINIX and FRAPTRAN shows good agreement between the codes. In almost all the cases, the fuel and cladding temperatures are very closely reproduced.

9.2 Verification of the restart feature

Ideally, the data structures in FINIX would be identical before the print of a restart file in the first run and after the restart file has been read in the second run. This would result in the output of a simulation being consistent whether it was run in one run, or in a an interrupted run that was finished with a restart. Due to the limitations presented by double precision and other limiting factors, the outputs of these two methods have slight differences.

A steady-state scenario IFA-432 rod 1 was run first in one run, and its results were compared to two runs that were interrupted at simulation time of 281.6 days and restarted to finish. In one of these restarted runs the nodalization remained unchanged, whereas in the other one the amounts of radial and axial nodes were changed. All simulations had the same time steps and input values, and consequently had identical results prior to the interruption.

From the moment of restart onwards the simulation results were compared quantitatively, calculating the average relative difference for each non-input quantity represented in the .sum-file. These results are presented in table 16. Due to the precision of the .sum-output being only four decimals, these differences are mostly present at the end of the simulation where the initial differences of less than 0.01 % have had an observable impact on. The restarted run with a change in the number of nodes yields higher differences, due to the nodalization having a large impact on the predicted values of most quantities.

Table 16. Relative differences caused by FINIX restart with no change in the nodalization and with renodalization during the restart.

Quantity	Average relative absolute difference (%)	
	Constant nodes	With renodalization
Buav	$1.79 \cdot 10^{-6}$	5.15
FGR%av	$2.11 \cdot 10^{-2}$	10.9
Intpr	$7.46 \cdot 10^{-2}$	7.29
Fuext	$1.36 \cdot 10^{-4}$	1.17
Clect	$9.06 \cdot 10^{-4}$	$4.79 \cdot 10^{-2}$
Tplen	$6.06 \cdot 10^{-5}$	0.61
Bulo	$1.66 \cdot 10^{-6}$	0.71
Tmax	$1.26 \cdot 10^{-4}$	1.56
FGR%lo	$1.68 \cdot 10^{-2}$	9.54
Gap	$1.81 \cdot 10^{-3}$	11.1
Gapcon	$1.46 \cdot 10^{-3}$	9.72
Tclav	$1.15 \cdot 10^{-6}$	$1.02 \cdot 10^{-2}$
Tcentav	$1.81 \cdot 10^{-6}$	1.45

9.3 Solved issues from previous versions

The calculated cladding creep strain did not accumulate across time steps in the version 1.19.1, but this is now fixed and the creep strain accumulates correctly.

Some finite strain capability has been added to FINIX in the form of the logarithmic strain framework. The finite strain calculation accuracy has not yet been validated, but will be performed in the near future.

9.4 Known issues and possible caveats

As shown in [1], the performance is very good for temperature calculations. However, a number of issues remain to be solved. These are:

- The fuel centerline temperature is often overestimated in the beginning of the irradiation of a fuel rod.
- The cladding-coolant heat transfer coefficient model yields too high heat transfer coefficients for BWR conditions.
- The pressure calculated by FINIX typically differs somewhat from that calculated by FRAPTRAN. The most probable cause for this is a difference in the rod free volume, possibly in the volume of the plenum. The plenum temperature is also calculated in a different way, but the difference exists already for zero power. The plenum temperature and fill gas pressure are both predicted to be higher than in FRAPTRAN.
- The plastic strains read from a FRAPCON restart file are treated differently by FINIX and FRAPTRAN. It is not exactly clear how FRAPTRAN treats the plastic deformations, but it seems that this is a source of some of the discrepancies between FINIX and FRAPTRAN.
- The fission gas release solution seems to be dependent on the time step. With smaller

time steps, higher fission gas releases are calculated, and with larger time steps, lower fission gas releases are calculated. In the FINIX FGR validation cases, the time step of the power history was directly used.

- The logarithmic strain framework implementation has not been validated for large strains.

In addition to the technical issues, one should keep in mind the limitations of the FINIX models:

- In many cases, when the range of validity of a model is exceeded, FINIX will not crash or abort execution. Instead, the solver will do its task and pass an error message. It is the responsibility of the user to catch the message and act accordingly. **Calls to FINIX functions should always be accompanied by error message checking.**
- The coolant model of FINIX is very limited. The model is not reliable beyond nucleate boiling.
- The FINIX creep model is the same model that is used in traditional fuel performance codes, but such models have been shown to be valid only with increasing stress on the cladding. Therefore the creep strains calculated by FINIX may be erroneous when modeling fuel rods where stress reversal on the cladding occurs (compressive stress turns into tensile or vice versa).
- The fission gas release model in FINIX slightly underestimates the fission gas release, and additional models should be implemented.
- FINIX rod failure criterion is inaccurate, but is implemented for numerical stability.
- FINIX contains only a low-temperature creep model, and high-temperature creep such as that in a LOCA scenario is not yet modeled.
- The burnup calculation of FINIX does not differentiate between the thermal and fission power. The results are indicative, and generally accurate within roughly 5–10 % of the experimentally determined values. When using FINIX with a host code which calculates a more accurate burnup, it is advised to input this burnup into FINIX.
- The FINIX radial power distribution model does not take into account the effect of burnable poisons.
- The cladding plasticity calculation requires a sufficiently small time step so that the calculation converges. Currently there is no general automatic substepping algorithm in FINIX, so a sufficiently small time step must be used if gap conductance or the radial return algorithm in the plasticity calculation do not converge.

10. Summary

The FINIX fuel behavior code has been updated to version 1.19.12. In this version a new solver of the steady-state heat equation is implemented along with several models important in modeling long irradiations: fission gas release, pellet swelling and densification and cladding creep.

Validation of the stand-alone FINIX-1.19.1 has been done in a separate report [1]. Results show good performance in RIA and steady state scenarios. Especially the temperature distributions are reliably calculated. Limitations have been discussed in chapter 9. The validation of FINIX-1.19.12 is to be performed in the near future.

The primary purpose of the FINIX code is to provide a fuel behavior module for other simulation codes in multiphysics simulations. The intended use is the improvement of fuel behavior description in neutronics, thermal hydraulics and reactor dynamics codes, without having to employ the available full-scale fuel performance codes. FINIX couples with the host code on a source code level, and provides an interface of functions that can be used to access the fuel behavior model from the host code. The required knowledge on the correlation-level details and rod parameters has been minimized by defining default templates that can be used without having information on all model-specific details.

Currently FINIX has been integrated into the Monte Carlo reactor physics code Serpent 2, where FINIX serves as the default fuel behavior module. In addition to Serpent, FINIX has been integrated into VTT's reactor dynamics codes TRAB-1D, TRAB3D and HEXTRAN. Results have been reported, for example, in Refs. [22, 20, 21, 16, 15].

Bibliography

- [1] J. Peltonen. Finix fuel behavior model and interface for multiphysics applications. Validation of version 1.19.1. Technical Report VTT-R-00135-19, VTT Technical Research Centre of Finland, 2019.
- [2] H Loukusa, J Peltonen, and V Valtavirta. FINIX – Fuel Behavior Model and Interface for Multiphysics Applications - Code Documentation for version 1.19.1. Technical Report VTT-R-00052-19, 2019.
- [3] H. Loukusa and V. Valtavirta. Finix fuel behavior model and interface for multiphysics applications. Code documentation for version 1.18.9. Technical Report VTT-R-06824-18, VTT Technical Research Centre of Finland, 2018.
- [4] J. Peltonen. Finix fuel behavior model and interface for multiphysics applications. Validation of version 1.18.9. Technical Report VTT-R-06823-18, VTT Technical Research Centre of Finland, 2018.
- [5] H. Loukusa and V. Valtavirta. Finix fuel behavior model and interface for multiphysics applications. Code documentation for version 0.17.12. Technical Report VTT-R-06793-17, VTT Technical Research Centre of Finland, 2018.
- [6] V Valtavirta. Implementing a steady state temperature solver for FINIX. Technical Report VTT-R-00518-17, VTT Technical Research Centre of Finland Ltd., Espoo, Finland, 2017.
- [7] T. Ikonen, J. Kättö, and H. Loukusa. Finix fuel behavior model and interface for multiphysics applications. Code documentation for version 0.15.12. Technical Report VTT-R-05887-15, VTT Technical Research Centre of Finland, 2015.
- [8] T. Ikonen, J. Kättö, and H. Loukusa. Finix fuel behavior model and interface for multiphysics applications. Code documentation for version 0.15.6. Technical Report VTT-R-02988-15, VTT Technical Research Centre of Finland, 2015.
- [9] T. Ikonen. Finix fuel behavior model and interface for multiphysics applications. Code documentation for version 0.13.9. Technical Report VTT-R-06563-13, VTT Technical Research Centre of Finland, 2013.
- [10] H. Loukusa. Finix fuel behavior model and interface for multiphysics applications. Validation of version 0.13.9. Technical Report VTT-R-06565-13, VTT Technical Research Centre of Finland, 2013.
- [11] T. Ikonen. Finix fuel behavior model and interface for multiphysics applications. Code documentation for version 0.13.1. Technical Report VTT-R-00730-13, VTT Technical Research Centre of Finland, 2013.
- [12] Henri Loukusa, Jussi Peltonen, Ville Valtavirta, and Ville Tulkki. Implementation of burnup effects in the multiphysics fuel behavior module FINIX. *Annals of Nuclear Energy*, 136:107022, 2020.
- [13] V Valtavirta, J Peltonen, U Lauranto, and J Leppänen. SuperFINIX - A flexible-fidelity core level fuel behaviour solver for multi-physics applications. In *NENE2019*, Portoroz, Slovenia, 2019.

- [14] J Peltonen, H Loukusa, and V Tulkki. Validation and verification of the FINIX fuel behavior module. In *TopFuel*, Seattle, WA, USA, 2019.
- [15] Elina Syrjälähti, Timo Ikonen, and Ville Tulkki. Modeling burnup-induced fuel rod deformations and their effect on transient behavior of a VVER-440 reactor core. *Annals of Nuclear Energy*, 125:121–131, 2019.
- [16] V Valtavirta, J Leppänen, and T Viitanen. Coupled neutronics–fuel behavior calculations in steady state using the Serpent 2 Monte Carlo code. *Annals of Nuclear Energy*, 100:50–64, 2017.
- [17] Timo Ikonen, Elina Syrjälähti, Ville Valtavirta, Henri Loukusa, Jaakko Leppänen, and Ville Tulkki. Multiphysics simulation of fast transients with the FINIX fuel behaviour module. *EPJ Nuclear Sciences & Technologies*, 2:37, 2016.
- [18] E. Syrjälähti, V. Valtavirta, J. Kättö, H. Loukusa, T. Ikonen, J. Leppänen, and V. Tulkki. Multiphysics simulations of fast transients in VVER-1000 and VVER-440 reactors. In *11th International Conference on WWER Fuel Performance, Modelling and Experimental Support*, Varna, Bulgaria, 2015.
- [19] T. Ikonen, E. Syrjälähti, V. Valtavirta, H. Loukusa, J. Leppänen, and V. Tulkki. Multiphysics simulation of fast transients with the FINIX fuel behaviour module. In *TopFuel 2015, Zurich, Switzerland*, 2015.
- [20] T. Ikonen, H. Loukusa, E. Syrjälähti, V. Valtavirta, J. Leppänen, and V. Tulkki. Module for thermomechanical modeling of lwr fuel in multiphysics simulations. *Annals of Nuclear Energy*, 84:111–121, 2015.
- [21] V. Valtavirta, T. Ikonen, T. Viitanen, and J. Leppänen. Simulating fast transients with fuel behavior feedback using the serpent 2 monte carlo code. In *PHYSOR2014, Kyoto, Japan*, 2014.
- [22] T. Ikonen, V. Tulkki, E. Syrjälähti, V. Valtavirta, and J. Leppänen. FINIX – fuel behavior model and interface for multiphysics applications. In *2013 Fuel Performance Meeting / TopFuel, Charlotte, USA*, 2013.
- [23] K.J. Geelhood, W.G. Luscher, C.E. Beyer, and J.M. Cuta. Fraptran 1.4: A computer code for the transient analysis of oxide fuel rods. Technical Report NUREG-CR-7023, Vol. 1, Pacific Northwest National Laboratory, 2011.
- [24] K.J. Geelhood, W.G. Luscher, and C.E. Beyer. Frapcon-3.4: A computer code for the calculation of steady-state thermal-mechanical behavior of oxide fuel rods for high burnup. Technical Report NUREG-CR-7022, Vol. 1, Pacific Northwest National Laboratory, 2011.
- [25] K J Geelhood, W G Luscher, P A Raynaud, and I E Porter. FRAPCON-4.0: A computer code for the calculation of steady-state, thermal-mechanical behavior of oxide fuel rods for high burnup. Technical Report PNNL-19418, Pacific Northwest National Laboratory, 2015.
- [26] W.G. Luscher and K.J. Geelhood. Material property correlations: Comparisons between FRAPCON-3.4, FRAPTRAN 1.4, and MATPRO. Technical Report NUREG-CR-7024, Pacific Northwest National Laboratory, 2011.
- [27] M. Suzuki and H. Saitou. *Light Water Reactor Fuel Analysis Code FEMAXI-6 (Ver.1) - Detailed Structure and User's Manual*. Japan Atomic Energy Agency, 2006.

- [28] Valentino Di Marcello, Arndt Schubert, Jacques Van De Laar, and Paul Van Uffelen. The TRANSURANUS mechanical model for large strain analysis. *Nuclear Engineering and Design*, 276:19–29, 2014.
- [29] Thomas Helfer. Extension of monodimensional fuel performance codes to finite strain analysis using a Lagrangian logarithmic strain framework. *Nuclear Engineering and Design*, 288:75–81, 2015.
- [30] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, 2nd Edition*. Cambridge University Press, 2002.
- [31] A. S. Khan and S. Huang. *Continuum theory of plasticity*. John Wiley & Sons, Inc., 1995.
- [32] F. Dunne and N. Petrinic. *Introduction to computational plasticity*. Oxford University Press, 2005.
- [33] M Limbäck and T Andersson. A Model for Analysis of the Effect of Final Annealing on the In- and Out-of-Reactor Creep Behavior of Zircaloy Cladding. In *Zirconium in the Nuclear Industry (11th)*, pages 448–468. ASTM, 1996.
- [34] Kenneth Geelhood. Implementing primary creep calculations during stress changes and reversals in the fuel performance code FRAPCON-3. In *TopFuel*, pages 188–194, Charlotte, NC, USA, 2013. ANS.
- [35] P van Uffelen, C Györi, A Schubert, J Van De Laar, Z Hózer, and G Spykman. Extending the application range of a fuel performance code from normal operating to design basis accident conditions. *Journal of Nuclear Materials*, 383:137–143, 2008.
- [36] K Forsberg and AR Massih. Diffusion theory of fission gas migration in irradiated nuclear fuel UO₂. *Journal of Nuclear Materials*, 135:140–148, 1985.
- [37] P Hermansson and A R Massih. An effective method for calculation of diffusive flow in spherical grains. *Journal of Nuclear Materials*, 304:204–211, 2002.
- [38] P. Van Uffelen, G. Pastore, V. di Marcello, and L. Luzzi. Multiscale modelling for the fission gas behaviour in the transuranus code. *Nuclear Engineering and Technology*, 43:477–488, 2011.
- [39] A R Massih and K Forsberg. Calculation of grain boundary gaseous swelling in UO₂. *Journal of Nuclear Materials*, 377:406–408, 2008.
- [40] G. Pastore, L. Luzzi, V. Di Marcello, and P. Van Uffelen. Physics-based modelling of fission gas swelling and release in UO₂ applied to integral fuel rod analysis. *Nuclear Engineering and Design*, 256:75–86, 2013.
- [41] IAPWS. Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic properties of Water and Steam. Technical Report R7-97(2012), IAPWS, 2012.
- [42] K J Geelhood, W G Luscher, J M Cuta, and I A Porter. FRAPTRAN-2.0: A Computer Code for the Transient Analysis of Oxide Fuel Rods. Technical Report PNNL-19400, Vol. 1 Rev. 2, PNNL, Richland, WA, USA, 2016.
- [43] L.M.K Boelter F.W Dittus. Heat Transfer in Automobile Radiators of the Tubular Type. *University of California Publications in Engineering*, 2(13):443–461, 1930.
- [44] Gonner C. Schmitz F., Papin J. RIA tests in CABRI with MOX fuel (IAEA-CSP-3/P). Technical report, IAEA, 2000.

- [45] L Desgranges. Internal corrosion layer in PWR fuel. In *Seminar proceedings of Thermal performance of high burn-up {LWR} fuel*, pages 187–196, 1998.
- [46] A.R. Massih K. Forsberg, M. Limbick. A model for uniform Zircaloy clad corrosion in pressurized water reactors. *Nuclear Engineering and Design*, 154:157–168, 1995.
- [47] A. Reymann D. Hagrman. MATPRO - Version 11, A Handbook of Materials Properties for Use in the Analysis of Light Water Reactor Fuel Rod Behaviour. Technical Report NUREG/CR-0497, TREE-1280, EG&G Idaho, Inc., 1979.
- [48] J. Kättö. Corrosion and its modeling in nuclear reactor fuel cladding. Master's thesis, Aalto University Department of Energy Technology, 2013.
- [49] L.C. Just L. Baker Jr. Studies of metal-water reactions at high temperatures iii. experimental and theoretical studies of the zirconium-water reaction. Technical Report ANL-6548, Argonne National Laboratory, 1962.
- [50] Márton Király, Katalin Kulacsy, Zoltán Hózer, Erzsébet Perez-Feró, and Tamás Novotny. High-temperature steam oxidation kinetics of the e110g cladding alloy. *Journal of Nuclear Materials*, 475:27 – 36, 2016.
- [51] J B Ainscough and J O Ware. Isothermal grain growth kinetics in sintered UO₂ pellets. *Journal of Nuclear Materials*, 49:117–128, 1973.
- [52] M I Mendelson. Average grain size in polycrystalline ceramics. *Journal of the American Ceramic Society*, 52:443–446, 1969.
- [53] K. Lassmann, C. O'Carroll, J. van de Laar, and C. T. Walker. The radial distribution of plutonium in high burnup UO₂ fuels. *Journal of Nuclear Materials*, 208:223–231, 1994.
- [54] A. Shestopalov, K. Lioutov, and Yegorova. Adaptation of USNRC's FRAPTRAN and IRSN's SCANAIRtransient codes and updating of MATPRO package for modeling of LOCA and RIA validation cases with Zr-1%Nb (VVER type) cladding. Technical Report NUREG/IA-0209, US NRC, 2003.
- [55] V. Valtavirta. Designing and implementing a temperature solver routine for Serpent. Master's thesis, Aalto University, 2012.
- [56] R.J. White and M.O. Tucker. A new fission-gas release model. *Journal of Nuclear Materials*, 256:1–38, 1983.
- [57] K. J. Geelhood, C. E. Beyer, and W. G. Luscher. PNNL stress/strain correlation for Zircaloy. Technical Report PNNL-17700, Pacific Northwest National Laboratory, 2008.
- [58] Chase M.W. Jr. NIST-JANAF Thermochemical Tables, Fourth Edition, J. Phys. Chem. Ref. Data, Monograph 9. Technical report, NIST, 1998.
- [59] J.A. Turnbull, C. Friskney, Findlay. J., F. Johnson, and A.J. Walter. The diffusion coefficients of gaseous and volatile species during the irradiation of uranium dioxide. *Journal of Nuclear Materials*, 107:168–184, 1982.
- [60] J A Turnbull, R J White, and C Wise. The diffusion coefficient for fission gas atoms in uranium dioxide. In *Water Reactor Fuel Element Computer Modelling in Steady State, Transient and Accident Conditions IWGFPT/32*, pages 174–181, Preston, UK, 1988. IAEA.
- [61] H. Petersen. The properties of helium: density, specific heats, viscosity, and thermal conductivity at pressures from 1 to 100 bar and from room temperature to about 1800 K.

Technical Report RISO-224, Danish Atomic energy Commission Research Establishment Risö, 1970.

- [62] P.A. Jens W.H., Lottes. Analysis of Heat Transfer, Burnout, Pressure Drop and Density Data for High-Pressure Water . Technical Report ANL-4627, Argonne National Laboratory, 1951.
- [63] M. L. Huber, R. A. Perkins, A. Laesecke, D. G. Friend, J. V. Sengers, M. J. Assael, I. N. Metaxa, E. Vogel, R. Mareš, and K. Miyagawa. New international formulation for the viscosity of H₂ O. *Journal of Physical and Chemical Reference Data*, 38(2):101–125, 2009.
- [64] R A Huber, M L Perkins, D. G. Friend, M. L. Huber, R. a. Perkins, D. G. Friend, J. V. Sengers, M. J. Assael, I. N. Metaxa, K. Miyagawa, R. Hellmann, and E. Vogel. New International Formulation for the Thermal Conductivity of H₂O. *Journal of Physical and Chemical Reference Data*, 41(3), 2012.
- [65] R.W. Lewis, P. Nithiarasu, and K.N. Seetharamu. *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. John Wiley & Sons, 2004.
- [66] J.C. Tannehill, D.A. Anderson, and R.H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Taylor and Francis, 1997.
- [67] K.J. Geelhood, W.G. Luscher, and C.E. Beyer. Fraptran 1.4: Integral assessment. Technical Report NUREG-CR-7023, Vol. 2, Pacific Northwest National Laboratory, 2011.