A DOCUMENT DRIVEN APPROACH TO CERTIFYING SCIENTIFIC COMPUTING SOFTWARE

By
NIRMITHA KOOTHOOR, B.ENG.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the degree
Master of Applied Science in Software Engineering

McMaster University

MASTER OF APPLIED SCIENCE (2013) McMaster University
(Software Engineering) Hamilton, Ontario

TITLE: A Document Driven Approach to Certifying Scientific Computing Sofware
AUTHOR: Nirmitha Koothoor, B.Eng.
SUPERVISOR: Dr. Spencer Smith
NUMBER OF PAGES: viii, 142

Abstract

With the general engineering practices being followed for the development of scientific software, scientists are seemingly able to simulate real world problems successfully and generate accurate numerical results. However, scientific software is rarely presented in such a way that an external reviewer would feel comfortable in certifying that the software is fit for its intended use. The documentation of the software development - Requirements, Design and Implementation, is not being given the importance it deserves. Often, the requirements are improperly and insufficiently recorded, which make the design decisions difficult. Similarly, incomplete documentation of design decisions and numerical algorithms make the implementation difficult. Lack of traceability between the requirements, design and the code leads to problems with building confidence in the results.

To study the problems faced during certification, a case study was performed on a legacy software used by a nuclear power generating company in the 1980's for safety analysis in a nuclear reactor. Unlike many other scientific codes of that time, the nuclear power generating company included a full theory manual with their code. Although the theory manual was very helpful, the documentation and development approach still needed significant updating. During the case study, 27 issues were found with the documentation of the theory manual, 2 opportunities to update the design and 6 programming style issues were found in the original FORTRAN code. This shows room for improvement in the documentation techniques in the development of scientific software based on a physical model.

This thesis provides a solution to the certification problem, by introducing software engineering methodologies in the documentation of the scientific software. This work proposes a new template for the Software Requirements Specification (SRS) to clearly and sufficiently state the functional and the non-functional requirements, while satisfying the desired qualities for a good SRS. Furthermore, the proposed template acts as a checklist and helps in systematically and adequately developing the requirements document. For developing the design and implementation, this thesis introduces Literate Programming (LP) as an alternative to traditional structured programming. Literate Programming documents the numerical algorithms, logic behind the development and the code together in the same document, the Literate Programmer's Manual (LPM). The LPM is developed in connection with the SRS. The explicit traceability between the theory, numerical algorithms and implementation (code), simplifies the process of verification and the associated certification.

Acknowledgments

With profound gratitude I would like to acknowledge the support and help of my Supervisor Dr. Spencer Smith. His academic, research expertise, experience, understanding, motivation and encouragement have been invaluable to me in completing my graduation on time. I learnt self confidence, taking initiative and a positive approach towards finding out solutions for many problems I have encountered during this thesis.

I would like to express my sincere thanks to Dr. Ned Nedialkov for his generous support and guidance during Literate Programming, and to Dr. John Luxat for helping me to understand the reactor physics. I would also like to thank Dr. Ned Nedialkov and Dr. Alan Wassyng for their valuable suggestions while reviewing my thesis.

I immensely thank my parents Vasanth Kumar and Rajini and other family members for their un ended support throughout my academic career without which I would not have finished this thesis. I will ever remain grateful for their care, sacrifices, unconditional love and affection.

I would like to acknowledge School of Graduate Studies and McSCert of Computing and Software department of McMaster University for the magnificient financial assistance provided without which this research would not have been possible.

Contents

Abstract **iii**

Acknowledgments **iv**

1 Introduction **2**

1.1 Research Context 2
1.2 Motivation of this Research 4
1.3 Research Problem and Scope 5
1.4 Organization of Thesis 6

2 Background **8**

2.1 Software Requirements Specification (SRS) 8
2.2 Desirable Qualities for an SRS 9
2.3 Literate Programming 12
2.4 Certification 13

3 Requirements Documentation by SRS Template **15**

3.1 N286.7 Standard's Expected Documentation of Requirements and Theory . 15
3.2 Proposed SRS Template 16
3.2.1 Table of Symbols 19
3.2.2 Goals 20
3.2.3 Assumptions 20
3.2.4 Theoretical Models 21
3.2.5 General Definitions 21
3.2.6 Data Definitions 22
3.3 An Introduction to our Case Study 23
3.4 Evaluation of the Template 26
3.4.1 Issues in Documentation of R_1 (Effective Thermal Resistance Between T_1 and T_2) 26
3.4.1.1 Context from the Original Theory Manual 26
3.4.1.2 Problems 27
3.4.1.3 Quality Improvements by the Proposed Template 28
3.4.1.4 Excerpt 1 Showing the Documentation of R_1 Using the Proposed Template 29
3.4.1.5 Comments on the Excerpt 32
3.4.2 Issues in Documentation of R_2 (Effective Thermal Resistance Between T_B and T_2) 33
3.4.2.1 Context from the Original Theory Manual 33
3.4.2.2 Problems 33
3.4.2.3 Quality Improvements by the Proposed Template 34

3.4.2.4 Excerpt 2 Showing the Documentation of R_2 Using the Proposed Template 35
3.4.2.5 Comments on the Excerpt 39
3.4.3 Issues in Documentation of Numerical Algorithm for Solving T_2 Quadratic Equation 39
3.4.3.1 Context From the Original Theory Manual 39
3.4.3.2 Problems 40
3.4.3.3 Quality Improvements by the Proposed Template 40
3.4.4 Issues in Documentation of Dryout Requirements 41
3.4.4.1 Context 41
3.4.4.2 Problems 41
3.4.4.3 Quality Improvements by the Proposed Template 42
3.4.4.4 Excerpt 3 Showing the Documentation of Dryout and Heat Out (q'_{out}) Using the Proposed Template 42
3.5 List of Issues Uncovered From the Original Theory Manual 44

4 Implementation of the SRS using Literate Programming 45
4.1 N286.7 Standard's Expected Documentation of Design and Implementation 45
4.2 Design and Implementation Using LP 46
4.2.1 Numerical Algorithm for Solving ODEs 48
4.2.2 Overall Algorithm and Function of *fuel_temp_* 48
4.2.3 Naming Convention of Variables for *fuel_temp_* 52
4.3 Evaluation of LP 55
4.4 Design and Implementation of Instance Models 56
4.4.1 Steady State Computation 56
4.4.2 Transient State Computation 60

5 Conclusions 62
5.1 Summary 62
5.2 Future work 64

Bibliography 66

A Software Requirement Specification for FP 69
A.1 Reference Material 70
A.1.1 Table of Symbols 70
A.1.1.1 Quantities related to Thermal Analysis 70
A.1.1.2 Quantities related to Nuclear Physics 71
A.1.2 Abbreviations and Acronyms 71
A.2 Introduction 72
A.2.1 Purpose of Document 72
A.2.2 Scope 72
A.2.3 Organization of Document 72
A.2.4 Intended Audience 72
A.3 General System Description 73
A.3.1 System Context 73
A.3.2 User Characteristics 73
A.3.3 System Constraints 73
A.4 Specific System Description 73
A.4.1 Problem Description 73
A.4.1.1 Background 74
A.4.1.2 Terminology and Definitions 77

A.4.1.3 Physical System Description 78
A.4.1.4 Goal Statements 78
A.4.2 Solution Characteristics Specification 79
A.4.2.1 Assumptions 79
A.4.2.2 Theoretical Models 80
A.4.2.3 General Definitions 80
A.4.2.4 Data Definitions 84
A.4.2.5 Instance Models 98
A.4.2.6 Data Constraints 102
A.4.2.7 System Behavior 104
A.4.3 Nonfunctional Requirements 105
A.4.3.1 Accuracy 105
A.4.3.2 Maintainability 105
A.4.3.3 Solution Validation Strategies 106
A.5 Other System Issues 106
A.5.1 Open Issues 106
A.5.2 Off-the-Shelf Solutions 106
A.5.3 Waiting Room 106
A.6 Traceability Matrix 106
A.7 Auxillary Constants 108

B LP Manual **109**
B.1 Overview 109
B.2 Numerical Algorithm 110
B.3 Algorithm 111
B.4 Overall function 111
B.5 Naming Conventions 112
B.6 Initialization section 117
B.6.1 Computing q'_N, T_2 and k_c 117
B.6.2 Computing h_c, h_g and T_S 119
B.6.3 Computing T_{CL} and k_{AV} 120
B.6.4 Computing T_1 122
B.6.5 Computing $\Delta H(T_{\text{abs}})$ 123
B.6.6 Computing C_1, $c_{p,1}$ 123
B.6.7 Computing C_2, $c_{p,2}$ 124
B.6.8 Computing C_{CL}, $c_{p,3}$ 125
B.6.9 Computing δ_{ox}, R_{ox} and q'_{MWR} 125
B.6.10 Computing q'_{out} and initializing f_p 127
B.7 Dynamic section 127
B.7.1 Checking for Dryout 128
B.7.2 Computing $q'_{N,k+1}$ and $k_{c,k+1}$ 128
B.7.3 Computing $h_{c,k+1}$ and $h_{g,k+1}$ 129
B.7.4 Computing $R_{1,k+1}$ and $R_{2,k+1}$ 129
B.7.5 Computing $T_{2,k+1}$ 130
B.7.6 Computing $T_{1,k+1}$ 130
B.7.7 Computing $T_{\text{CL},k+1}$ 131
B.7.8 Computing $T_{S,k+1}$ 131
B.7.9 Computing $\Delta H(T_{\mathbf{abs}})$ **and** $P_{F,SUM}$ 132
B.7.10 Computing $C_1, C_2, C_3, c_{p,1}, c_{p,2}, c_{p,3}$ 132
B.7.11 Computing k_{AV} 133
B.7.12 Computing q'_{out} 134

B.7.13 Computing **rate of oxidation** 134
B.7.14 Computing **metal water reaction heat** (q_{MWR}) 135
B.7.15 Computing **oxidation layer thickness** 135
B.7.16 Computing **Integrated metal water reaction heat** (q'_{MWRI}) 135

C Checklist **137**

D makefile **138**

Chapter 1

Introduction

The scientific computing software used for the analysis of the safety systems such as those for a nuclear reactor, should be certified before being used, especially given the importance of having confidence in the computed results. However, in the scientific computing field, due to a lack of proper documentation of theory and improper structuring of requirements, certification relevant activities, such as verifiability can be problematic. In particular, there is often a disconnect between the theory and the software code for the researchers, which leads to problems with building confidence in the results. This thesis attempts to address this issue by presenting a document driven approach to certifying scientific computing software. The new approach is supported through a case study on presenting the documentation of requirements, design and the code of a software program for thermal analysis of a nuclear fuel rod.

The organization of this chapter is as follows: Section 1.1 points out the problems arising from the general development of scientific software, explains the need for improving the documentation techniques, gives the role of safety analysis software in the nuclear power plants and then gives a brief introduction to the software that this research uses for the case study. Section 1.2 presents the motivation of this research while Section 1.3 clarifies the research problems and scope. The organization of the thesis is given in Section 1.4.

1.1 Research Context

Scientific computation consists of using computer tools to analyze or simulate continuous mathematical models of real world systems of engineering or scientific importance, so that we can better understand and predict the system's behavior. Scientific computation is an option when the problems are too complex, large, dangerous or expensive to explore by traditional forms, like theoretical studies and laboratory experiments [28, 22]. Examples of scientific computing problems include the following: weather prediction, understanding natural disasters like tsunamis, earthquakes etc. analyzing the flow of blood in the body and other problems relating to the areas of health, safety, manufacturing and the environment. To tackle these kinds of problems, where there is no possibility of validating the numerically calculated results against reality, scientists construct mathematical models and develop quantitative analysis techniques. They then use computers to simulate the models and solve the analysis problems. The scientists run the simulations and analysis problems with various sets of input parameters to understand and predict the system's behavior [31].

With the general development approaches common in scientific computing software, the certification process is difficult for the following reasons:

- The documentation of requirements and theory is often given lower priority than other development activities, with most of the time and efforts dedicated to the implementation. The documentation of all the necessary functional and non functional requirements of the software is often neglected. This makes the achievement of qualities like completeness, correctness and verifiability difficult.
- Many scientific computing softwares based on a physical model are documented in a "reverse engineering manner". That is, first the code is developed and then the theory behind it is documented. Starting from computer code, it is a difficult task to develop the manual, since proper documentation requires understanding the assumptions made and the way the mathematical models have been manipulated to best fit into the code.
- Due to the frequent lack of traceability between theory and implementation, certification can become time consuming and complex.
- Any inconsistencies in the documentation confuse the reader, making verification and maintainability difficult.
- If traceability between the requirements and the implementation is not complete, then a conservative recertification process needs to reproduce most of the work of the initial certification exercise. This means that without proper documentation, recertification can be very expensive and time consuming, as every line of the code has to be verified right from the beginning along with the requirements and the numerical algorithms.
- The justification of the choice of numerical algorithms and explanation of their implementation are often lacking, which makes the task of verification challenging.

The above mentioned problems emphasize the importance of documentation, and suggests that there is room for improvement in the general documentation practices. We need the requirements, design and the code to be documented in a proper manner. The theme of this research is to simplify the task of certification by introducing a document driven approach for developing scientific computing software.

Due to the vastness of scientific computation field, we restrict ourselves to computational physics problems and in our case study, we further narrow our domain to nuclear physics. We chose certification of safety analysis software that is used at nuclear power plants as our research context. At nuclear power plants, safety analysis software is developed to justify the design and analysis of safety-related equipment, systems, structures and components [3]. The regulators make their decision to approve the reactor design based on the safety analysis conducted by this software. So, it is necessary that the safety analysis software is certified to provide adequate confidence in its correctness and to achieve the essential qualities of verifiability, completeness, correctness, maintainability and dependability. These qualities are discussed in detail in Section 2.2.

In this research, we do a case study on a legacy nuclear safety analysis software that was used by a nuclear power generating company in the 1980's. For the purpose of this thesis, the software under study is referred to by the name FP. Although FP is not being used anymore, it was developed to perform thermal analysis of a single fuelpin in the reactor. FP is used to simulate simplified reactor physics, fuel management calculations and simulations that can be used for reliability studies at the nuclear power generating stations. The FP software was provided to us with a developed code and a theory manual, which includes the requirements, numerical algorithms, assumptions, constraints and the mathematical models as developed by the nuclear power generating company. The inclusion of a theory manual with the code represents a commitment to documentation that would have been rare at the time of development.

Our goal is to rebuild the FUELTE subroutine of FP software using the documentation techniques introduced by this thesis. The design and development of the documentation was done to be consistent with clause 11.2 of N286.7, where N286.7 forms the standard set for the quality assurance of analytical, scientific, and design computer programs for nuclear power plants [3]. These standards apply for the design, development, maintenance, modification and use of the software at nuclear power plants.

1.2 Motivation of this Research

In many cases, scientific software developers are the scientists themselves. Developers of this kind are called the professional end user developers. Typically they have little or no education or training in software engineering [26]. Due to the complex science and the theory behind the software, an average developer does not understand the application domain and hence the scientists prefer to become the developers. With this group of developers, especially given the importance of the software they are writing, it is desirable to provide explicit instructions and guidelines for the process, including documentation standards.

The importance of documentation is first apparent when considering requirements. Often scientists do not know exactly what they are building while developing the software. They may lack a proper idea of the requirements of the final product and they may have an incomplete understanding of what they want to achieve. By running simulations they continue to learn and refine the requirements, as their understanding of the domain progresses. Therefore, they can sometimes fail to structure the requirements properly. This triggers problems during the development and maintenance phases [20]. Failure to document accurate, precise and adequate requirement specifications leads to the development of unreliable, undesirable products, delayed completion of the project and many other misunderstandings [19]. This motivates us to adopt existing software methodologies for requirements documentation to improve the quality aspects like correctness, verifiability, maintainability, modifiability, consistency, traceability, completeness and unambiguity. Following a template that defines the content and structure makes the writing of a document easier and more effective. Therefore, in this research we employ SRS, a standard template for requirements specification for scientific software (discussed in detail in Chapter 3).

Even if the requirements are documented accurately, if adequate information about the design is not presented, the verification of the implementation with the numerical algorithms can become very complicated. The verifier of the software is not necessarily from the same domain as the developer. So, it is the responsibility of the end user developer to provide all the necessary information about the design of the software. The verifier needs to know the decisions made during the implementation phase, the numerical algorithms used, the assumptions made and the techniques selected for developing the models to validate and check the correctness of the software. Even if a single piece is missing, it makes the task of certification very difficult. So completeness is a very important quality which has to be achieved while documenting the design and development for the software to be certified. However the documentation of numerical algorithms, solution techniques and the rationale for selecting these techniques are not always given the value they need, making the traceability between the code and the numerical algorithms difficult. This motivates us to adopt a documentation technique that helps the scientists to record the design and implementation of the code according to their thought flow. A literature review on Literate Programming (LP) given in chapter 2 suggests the use of LP for developing the code, as LP helps in achieving completeness, traceability and verifiability between the theory, de-

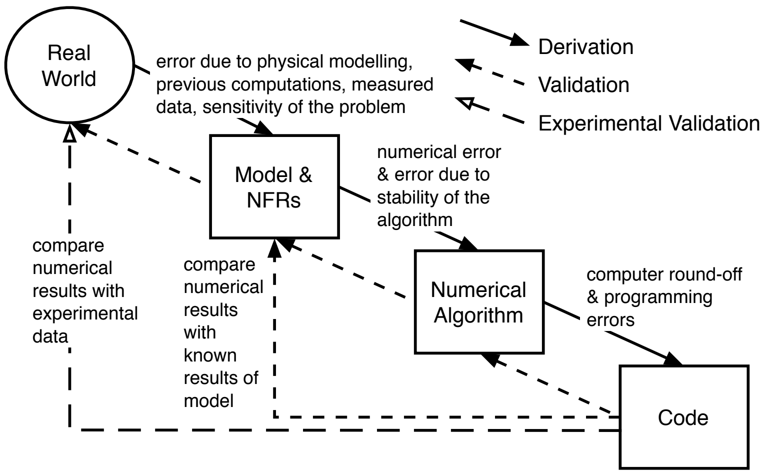

Real World
error due to physical modelling, previous computations, measured data, sensitivity of the problem
Derivation
Validation
Experimental Validation
Model & NFRs
numerical error & error due to stability of the algorithm
Numerical Algorithm
computer round-off & programming errors
Code
compare numerical results with experimental data
compare numerical results with known results of model

Figure 1.1: Procedure for Development of Scientific Software

sign and the implementation. Moreover, LP has been used previously for software quality assurance [1, 16, 17].

By introducing Software Requirements Specification (SRS) and Literate programming into the scientific field, the quality of documentation can be improved. The use of an SRS and LP together will help to increase the confidence in correctness of the software and make verifiability, reliability, reusability and maintenance easier. This way, the software engineering methodologies are being introduced into developing the nuclear safety analysis software, following the refined procedure shown in Figure 1.1. In this figure the term validation is used to denote a check that the downstream stage has produced an artifact that matches the needs of the upstream stage. This terminology is not standard in the industry. In some cases, like at the nuclear power generating company that used the FP software, this process would be termed verification.

Figure 1.1 which is taken from [28] gives the typical model of the scientific software development. There is a similarity between the workflow of scientific software development and the software methodologies, as both the methods recommend the same procedure for developing the software. The first phase of scientific software development, in which the Non functional Requirements (NFRs) are collected and mathematical models (nothing but the functional requirements) are developed, is similar to the requirements phase of the waterfall model. Similarly, the second and third phases of scientific software development in which the numerical algorithms and code are developed are similar to the design phase and implementation phase of the waterfall model respectively. So we apply the software methodologies in developing the scientific software. That is, we use Software Requirement Specification (SRS) to document the NFRs, mathematical models and Literate Programming to design and implement the numerical algorithms of the scientific software.

1.3 Research Problem and Scope

Due to the breadth of the domain of scientific software, we have decided to restrict our research domain to software solving computational physics problems. Since we are given an existing theory manual and code, as mentioned in the research context, we chose documentation of requirements as our starting point. We are not going to deal with the requirements elicitation in this research. Also testing of the software is out of the scope of this thesis, as the theme of this research is to emphasize verification of the software by human experts, rather than building exhaustive test cases for machine processing. Testing is still an important part of the overall certification process, but is not emphasized in the current work. We have narrowed our scope to documentation related problems and accordingly address the following questions as our *research problems*:

1. How to improve the requirements documentation, capturing all the necessary information for developing scientific software.
2. How to achieve the desirable qualities of a good requirements document for scientific software.
3. How to develop the code with traceability to the numerical algorithm choices.
4. How to achieve traceability between the theory, design and the implementation.
5. How to facilitate certification of scientific software via documentation techniques.

   To provide solutions to these questions, our research performed the following activities:

   - We studied existing standards and templates used for documenting requirements.
   - We proposed a systematic approach for requirements documentation by adapting an existing template for scientific computing software.
   - We applied the existing software methodology of Literate Programming (LP) for developing the design and the code of the scientific software simultaneously.
   - We evaluated our proposed requirements template and programming approach by doing a case study on a specific example of safety analysis software: FP.
   - We strengthened our arguments by:
     - Demonstrating how the desired quality attributes of an SRS, as mentioned in Section 2.2, are achieved by using our proposed template.
     - Showing how the code is developed in connection with the SRS and design following the programming approach introduced by this research.
   - We used the results of our case study to support the proposed documentation practice for achieving the goals of certification.

1.4 Organization of Thesis

This thesis is organized in five chapters and four appendices as follows:

- Chapter 1 provides an introduction on the problems faced with the general engineering practice of developing scientific software. Furthermore, it presents the context, motivation, problem and scope of this research.
- Chapter 2 reviews the literature of certification and the two techniques (SRS and LP) proposed by this research for documentation purpose. We discuss the qualities that are to be possessed by a good SRS and define the expectations of the standard

(N286.7), set for the quality assurance of design and development of software for nuclear power plants. We go further showing the connection between the documentation and certification and explaining why we have chosen a documentation driven approach for certification.

- Chapter 3 presents our approach towards the documentation of requirements. We introduce a new template for documenting the requirements and give an overview of our case study. We evaluate the proposed template by discussing the issues faced with the original requirements document of the software under case study and then show how we have resolved those issues by following the proposed template. With the help of excerpts from Appendix A, we show how the desirable qualities of a good SRS were achieved by documenting requirements following the proposed template.

- Chapter 4 presents our approach towards the documentation of the design and implementation. First the information to be documented about the design and programmer's manual according to N286.7 standard is discussed. Then it is shown how the proposed technique can be used for documenting design and implementation in the same document satisfying both the expectations of the N286.7 standard and the desirable qualities of a good document, discusssed in Chapter 2. Furthermore, it is shown how the requirements, design and implementation are documented in connection with each other using excerpts from Appendix B.

- Chapter 5 presents the concluding remarks of this thesis as well as recommendations for future research work.

- Appendix A shows a case study of using the proposed requirements template for documenting FP software.

- Appendix B shows a case study of using the Literate Programming for documenting the design and implementation of FUELTE subroutine of FP software.

- Appendix C gives a checklist of the conditions that are to be satisfied for achieving the correctness and completeness of documentation.

- Appendix D is the make file created to link, compile and run the literate C function and the original FORTRAN code together.

Chapter 2

Background

This chapter provides background information on Software Requirements Specification (SRS) and Literate Programming (LP). These are the two documentation techniques used in the new approach proposed by this thesis for certification of scientific computing software for safety analysis. Section 2.1 discusses the details about Software Requirements Specification (SRS), its role in the software development and the importance of using a requirements template in developing an SRS. The qualities a good SRS should possess are detailed in Section 2.2. Section 2.3 gives the details about what LP is, how a literate program is developed, different LP tools and the advantages of LP. Section 2.4 provides a literature review on certification and shows the connection between certification and documentation, by discussing the N286.7 standards set for developing nuclear software.

2.1 Software Requirements Specification (SRS)

Requirements analysis and documentation is the first phase of developing software in a rational way. Requirements record all the expected characteristics and behaviour of the system. The document that records the requirements is called the *Software Requirements Specification* (SRS). This document describes the functionalities, expected performance, goals, context, design constraints, external interfaces and other quality attributes of the software [10, 11].

The advantages provided by SRS during software development are listed as follows [4, 20, 23, 28, 29]:

- An SRS acts as an official statement of the system requirements for the developers, stakeholders and the end-users.
- Creating the SRS allows for earlier identification of errors and omissions. Fixing errors at the early stages of development is much cheaper than finding and fixing them later.
- The quality of software cannot be properly assessed without a standard against which it should be judged. Verification and Validation (V&V) efforts cannot be fully successful without explicit requirements. Moreover, some requirements documents explicitly address the validation strategies against which the compliance of the software results are to be measured.
- An SRS facilitates the estimation of cost and time involved in the development process.

- An SRS aids in making decisions regarding design and coding of the software by serving as a starting point for the software design phase.
- The SRS aids the software lifecycle by facilitating incremental development. That is, a new version of the software can inherit features of the previous version to upgrade the system by improving the features.

To write an SRS, a common approach that is adopted is the use of a *requirements template*, which provides guidelines for documenting the requirements. It uses a framework that suggests an order for filling in the details. There are many advantages of using a template in writing an SRS [23, 28]. One advantage is that a template increases the adequacy of an SRS by providing a predefined organization that aids in achieving completeness and modifiability. Moreover, a template with a well organized format acts like a checklist for the writer, thus reducing the chances of missing information. Another benefit is that an SRS facilitates the communication between the stakeholders, developers and maintenance staff. A template aids in achieving information hiding through specific guidelines on the appropriate level of abstraction and makes the document more understandable by showing the connections between different sections. Furthermore, templates provide the advantage of facilitating comparison of two SRSs that conform to the same template. Finally, as each section in the template is filled systematically, it helps in information handling. That is, the specifier can organize large quantities of information by systematically locating the contents into respective sections.

There are several existing templates that have been designed for business and real time applications. These templates contain good suggestions on how to avoid complications and how to develop an SRS to achieve qualities of good documentation [7, 11, 15, 23, 28, 30]. There is no universally accepted template for an SRS. This research adapts the SRS template developed for scientific computing software in [28]. In accordance with the research scope (Section 1.3), we focus on requirements documentation in the domain of computational physics.

2.2 Desirable Qualities for an SRS

According to IEEE recommended practice, the qualities that a good SRS should possess are correctness, completeness, consistency, modifiability, degree of importance and/or stability, traceability, unambiguity and verifiability [11]. These qualities along with the additional quality of abstraction, considered as important for this current research, are defined below:

- *Completeness*: An SRS is said to be complete only when it satisfies all of the following conditions:
  - All the requirements of the software are detailed. That is, each and every goal, functionality, attribute, design constraint, value, data, model, symbol, term (with its unit of measurement if applicable), abbreviation, acronym, assumption and performance requirement of the software is defined.
  - The software's response to all classes of inputs, both valid and invalid and for both desired and undesired events is defined.
  - The system's external behaviour and external interfaces are included. Every external requirement laid by a superior specification, such as a system specification, must be considered and dealt with.

  - Every figure, table, model, assumption and definition is properly labelled and referenced throughout the document.
  - The problem statement, context, scope and purpose are clearly stated.

- *Consistency*: An SRS is said to be consistent when no subset of individual requirements specified are in conflict with each other [11]. That is, a specification of an item made at one place in the document should not contradict the specification of the same item at another place. An SRS is said to be consistent when the following conditions are satisfied:
  - There are no two different specifications for the requirement of the same real world object. For instance, if a requirement about a real world object states that the shape of a plate is rectangular, then there should not exist another requirement in the SRS stating the shape of the plate is elliptical.
  - The specification of the functions to be performed by the software must not conflict logically or temporally. For example, if one requirement says that the program must first increment the input A and then multiply with input B, then another requirement should not state that the input A should be multiplied by input B first and then incremented.
  - Each term in the document has a unique symbol and definition.
- *Modifiability*: An SRS should be developed in such a way that it is easily modifiable so that likely changes in future do not destroy the structure of the document. Also it should be easy to reflect the change wherever needed in the document maintaining consistency, traceability and completeness. For an SRS to be modifiable, its format must be structured in a way that fulfills the following:
  - Repetition should be avoided. Every statement should be specified only once in the SRS. Redundancy makes maintenance difficult, because when a statement has to be changed, the change has to be reflected wherever the statement appears in the document. If by mistake the statement or information dependent on the statement is changed at its first occurrence and its second occurrence is not altered, then the SRS would become inconsistent.
  - If a portion of a requirement has to be repeated, cross-referencing should be used to keep track of this occurrence to make the task of alteration and maintenance easier.
  - Every requirement should be specified individually without mixing with other requirements.
  - A well structured template should be used with explicit cross referencing and a table of contents.
- *Degree of Importance and/or Stability*: In an SRS, all the requirements may not be of the same importance. Some might be essential while others might be desirable. Some requirements might change over the course of time. So, the degree of importance or stability of a requirement should be documented by associating an identifier with that particular requirement. The identifier indicates the rank of importance of a requirement as either essential (or) conditional (or) optional. Similarly the stability can be expressed in terms of number of likely changes to a requirement based on experience or knowledge of forthcoming events.

  The quality of degree of importance is not emphasized in this research. The feeling is that for the certification of nuclear safety analysis software, all the requirements are essential and are given equal importance.

- *Traceability*: An SRS should be traceable, as this facilitates maintenance and review. If a change is made to the design or code of the software, then all the requirements relating to those segments have to be modified. Hence it is important to have traceability to every component of the SRS. This can be achieved by giving a unique label to every component and developing the SRS in a hierarchical way so that we can trace back to the first occurrence of each component.
- *Unambiguity*: As the SRS is used in design, implementation, verification, validation and maintenance phases, it is very important that the requirements are adequate, precise, consistent and understandable to the reader. To make the requirements clear, they should be recorded in a way that there is no ambiguity in their specification. An SRS is said to be unambiguous only when every requirement's specification has a unique interpretation. If the term being used has two different meanings and its use in the interpretation is unavoidable, then the term should be included in a glossary with the different meanings of the term, along with their units (if appropriate). Each different meaning should be clearly connected to the appropriate context. The SRS should be unambiguous to all audience, including both the developers and users.
- *Correctness*: There is no direct tool or method for measuring the correctness of an SRS. One way of building confidence in the correctness is by reviewing to ensure that each and every requirement stated in the SRS is the one that the stakeholders and experts desire. By maintaining traceability, consistency and unambiguity, we can reduce the occurrence of errors and make the goal of reviewing for correctness easier. The assessment of correctness is an iterative process, since the users of the developed software will provide feedback and corrections, which can be incorporated into later versions of the code and its associated documentation.
- *Verifiability*: Every requirement in the SRS must be the one fulfilled by the implemented software. Therefore all the requirements in the SRS should be clear, unambiguous and testable so that a person or a machine can verify whether the software product meets the requirement. For an SRS to be verifiable, it is necessary that there be no ambiguity in the specification of the requirement. For example,
  - If a requirement is stated as "The amount of efforts put into maintaining the software should be low", this is an ambiguous statement since we cannot quantify the term 'low'. Instead, if it is specified as "The amount of efforts put into maintaining the software should be $\frac{1}{4}^{\text{th}}$ of the effort put into developing the software", then it is verifiable as it uses measurable quantities to compare and check.
  - If a requirement states that the output of software A should be as accurate as the output of software B, then the verifier will not be in a position to verify this requirement, as it is impossible to define the term 'as accurate as' in an unambiguous way. So, this requirement should be modified to make it more precise as "The relative difference between the output of software A and the output of software B should not be more than 0.05%".
- *Abstract*: Another quality an SRS should possess is being abstract. It should tell us what the software must do and the properties it must possess, but should not speak about how these are to be achieved. For example, a requirement can specify that an Ordinary Differential Equation (ODE) must be solved, but it should not mention that Euler's method should be used to solve the ODE. How to accomplish the requirement is a design decision, which the developer makes during the design phase.

2.3 Literate Programming

Literate Programming (LP) is a programming approach introduced by Donald Knuth, in which the source code is developed along with the logic behind it, using both formal and informal methods. According to Knuth, LP mainly concentrates on explaining to the reader what we want the computer to do, instead of imagining the main task to be instructing a computer on what to do [6]. The developer strives to make the program understandable to the reader, by introducing the concepts in an order that is best for human comprehension [6]. In the case of software quality assurance, the main concern of LP is to deliver a document of publishable quality that can improve the certifier's confidence in the program's validity.

While developing a literate program, the algorithm is divided into smaller modules which contain explanation, definitions and implementation. In each module, the implementation is done as small pieces of code called *chunks* or *sections* [6]. This way, the program is developed as an uninterrupted exposition of logic with scattered snippets of *macros* and source code [32]. Macros are simple title like phrases that hide the chunks of code, abstractions and any lower level macros. In literate programming, both logic and code are written in the same source file. The chunks of code which are interconnected as a "web" can be assembled into a compilable program in a *tangle* process. The extraction of the logic from the LP source is called a *weaving* process. The ouput in this case is a nicely formatted document [6]. The reader uses the logic document for understanding the theory and for verification purposes, while the compiler uses the tangled code to compile and execute the program.

To obtain the two representations from the source file, LP tools like WEB, NOWEB, CWEB, FunnelWeb etc. are to be used. WEB was introduced by Donald Knuth and uses Pascal as its underlying programming language and TeX for typesetting of the documentation [5]. NOWEB is programming language independent and uses HTML for text formatting [2]. The tool that has been used in this research is 'CWEB', which was written by Donald Knuth and Silvio Levy. It uses C / C++ for programming and TeX for typesetting of the documentation [12].

The CWEB tool provides two utilities 'CWEAVE' and 'CTANGLE' which are used to extract the logic and the code respectively from the source file [12]. CWEAVE intertwines the TeX and C portions of each module of the CWEB file and knits them into a structured document. It also takes care of page layout, indentation, fonts, pretty printing of C/C++ code, and generates extensive cross-index information [6, 17]. CTANGLE extracts the code sections from the CWEB file and arranges the sections in the order required by the C file. It also includes line information of the CWEB source file in the generated C/C++ files. Hence, if warnings or errors are detected during compile time or runtime, then the compiler and debugger can give the line number of the CWEB file [17].

The LaTeX CWEB is a bundle that allows us to use LaTeX for marking up the CWEB file thus allowing us to have chapters, sections, subsections, environments, figures, graphics etc in the weaved output file [25]. The cweb-hy class, an extension of CWEB, enables automatic generation of hyperlinks making the navigation through the code in the resulting output file convenient [17].

The advantages in developing a literate program are as follows [17]:

- The process of verification by a human reader is made easier, since the reviewer needs only one document for evaluation.
- As all the necessary information about the theory and the logic behind the program are in the same document along with the code, even a person without any background knowledge on the theory can verify the program.

- As every line of the code can be traced back to either the theory or to the numerical algorithms (design), it helps in increasing the confidence in correctness of the implementation.
- The amount of efforts required for the maintenance of the program can be reduced, as the design and code are kept in sync maintaining consistency.
- A high quality, well structured LP document reduces time for debugging and testing, as errors are detected through proof reading.
- As the code is developed by implementing each requirement as an independent chunk, reusability and modifiability can be achieved.

Due to all the above mentioned reasons, LP eases the process of certification.

2.4 Certification

According to Roache, Certification is an engineering management activity towards Quality Assurance (QA) [21]. Software is said to be certified when an authority or regulatory body gives it an official recognition of meeting certain standards [22]. Its often a misconception that certification means Verification and Validation (V&V). But in the true sense, V&V are just a part of certification, while the process of certification includes many other aspects of the software like documentation of requirements, version control, and the QA system itself [21]. The aim of certification is to ensure that the characteristics of the product being certified are appropriate [14]. Hence the goal of software certification should be to: "...systematically determine, based on the principles of science, engineering and measurement theory, whether a software product satisfies accepted, well-defined and measurable criteria" [9].

As mentioned in the research context, the certification of the FP code is done using N286.7, which is the standard set for QA of design and development of computer programs for nuclear power plants. According to these standards, the software must be developed following the steps given below [3]:

- *Documentation*: The documentation for design and development of the computer programs must be complete and include the following:
  - The problem definition along with an explanation of the nomenclature or any conventions used.
  - A theory manual with the description of the theoretical and mathematical foundations of the computer program.
  - Requirements specification.
  - A design description demonstrating how the specified requirements have been met. It should provide information on identification of the algorithms, computer program structure, data structures, program flow, description of modules, module interfaces and library functions.
  - programmer's manual including information on program flow and structure, how the theory is translated into coding, how to modify and maintain the computer program and conventions on programming practices, such as variable naming and computer program commentary.

As stated in page 13 of [3], if the same information appears in two documents at different levels of detail, then it should be documented *consistently*. For instance, the information about program flow and structure is provided in the so-called Design Description document (See Chapter 4) before the coding is done and the same information is presented in the Programmer's Manual, which is prepared after the coding is complete. In these cases, the information should be consistent and wherever possible, the information should be contained in one document and referred to in the other. This makes the maintenance part easy, as any inevitable changes that are to be made in future shall be made only to one document but gets reflected in the other too.

- *Verification*: The verification will be performed by one or more suitably qualified persons who were not in the development team for checking the *completeness* and *correctness* of the software. For verification purpose, the clause 6 of the standards ([3]) require the following steps to be conducted:
  - The requirements document should be verified to ensure that the specifications are *complete* and address the problem.
  - The theoretical background and mathematical basis for the solution of the problem provided by the theory manual should be verified to ensure that it is appropriate to the intended application of the computer program.
  - The design should be verified to confirm compliance (*correctness*) with requirements.
  - Verification of coding must be done by conducting walkthroughs, independent review of computer program text, mathematical analysis of the computer program functions, and testing.

Although not explicitly stated in N286.7, the standard makes it clear that the qualities required by a nuclear safety analysis software to be certified are as follows:

- Completeness in documentation
- Consistency in documentation
- Modifiability of the software
- Traceability between the requirements specifications, theory, design and implementation
- Correctness of requirements specifications, design models and the code

As the process of certification is required to be a measurement based activity according to [9], the characteristics of the software must be measurable entities (similar to quality of *verifiability* of an SRS).

From the above mentioned points, it is evident that the qualities required by a safety analysis software to get certified are analogous to the desirable qualities for an SRS. So, we chose a documentation driven approach to certify the software. In the new documentation approach, we are going to develop the requirements document and theory manual as one document, SRS. The design document, programmers manual and code are together developed in another document, called the LP Manual, using LP.

Chapter 3

Requirements Documentation by SRS Template

As discussed in Chapter 2, requirements analysis and documentation is the first phase of software development. However, since we were provided with a developed theory manual and code for our case study, as mentioned in Section 1.1, we started our work from the documentation of requirements. Even though the N286.7 standard suggests two documents to record the requirements and theoretical background, we are going to combine both documents into one document. This combination is done because the contents of the theory manual, according to N286.7 standard as described in Section 3.1, can be considered as functional requirements in the adopted SRS template. This chapter introduces the proposed template for recording the requirements and theoretical background.

The organization of this chapter is as follows: First, the essential requirements, mathematical equations and other theoretical information that are required to be documented as per N286.7 standard are discussed in Section 3.1. Second, the new template for the SRS is introduced in Section 3.2. Third a brief introduction is given about the case study in Section 3.3 and finally in Section 3.4, it is shown how the desirable qualities for an SRS (given in Section 2.2) are achieved by the proposed template.

3.1 N286.7 Standard's Expected Documentation of Requirements and Theory

According to clause 11.2.4 of N286.7 standard, the requirements specifications for a computer program should be prepared in such a way that they include the following:

- the name of the computer program
- the functions of the computer program
- hardware, computer program, and user interface requirements
- operating system requirements
- computational speed requirements
- file size and type requirements
- input and output requirements

- data structure and data flow requirements
- programming language
- imposed physical or mathematical models or numerical algorithms
- error detection and handling requirements
- accuracy targets
- requirements on programming practices
- portability requirements

Similarly, the standard requires the theory manual to describe the theoretical and mathematical foundations of the computer program. The theory manual should be developed in such a way that it includes the following:

- the theory and mathematical equations
- assumptions and constraints
- solution techniques and the rationale for selecting these techniques such as accuracy requirements and other limitations;
- any empirical correlations, their range of application, and associated uncertainties
- applicable existing references

As mentioned in the introduction, the current work proposes combining these two documents into one SRS document. This is done to avoid repetition and to make the documentation more in keeping with standard software engineering practice. However, the SRS template being proposed, does not include solution techniques as expected by N286.7 standard. Solution techniques are excluded to achieve the quality of abstraction. The development of solution techniques (algorithms) is handled in the design process and is included in the design manual (see Chapter 4). Also, documenting the non functional requirements such as portability, computational speed are out of the scope of our work. Although nonfunctional requirements and other system isssues would normally be in the SRS, they were not included in the documentation provided by the nuclear power generating company. Rather than inventing nonfunctional requirements without any basis, the decision was made to simply exclude them from the SRS at this time.

3.2 Proposed SRS Template

It is necessary to develop a specific template for requirements documentation of nuclear safety analysis software, as it is unrealistic to expect the documentation of all kinds of scientific software to be the same. The objective is to construct a correct, complete, reusable, maintainable and traceable SRS. For developing the new template for an SRS, we studied the following templates that have been developed earlier for different domains and applications:

- B. Sanga's Proposed Template [23]
- Lei Lai's Proposed Template [28]
- The IEEE Standard 830-1998 [11]

We borrowed the template developed by Lei Lai, Dr. Spencer Smith and Dr. Ridha Khedri [28] for engineering mechanics and adapted it to suit the nuclear physics domain by adding a few new sections. We have chosen Lei Lai's template because it is an example SRS tailored to scientific software systems.

The proposed requirements template, which is given in Figure 3.1, is systematically developed by decomposing the problem into smaller tasks. That is, we try to achieve the goals of the problem by considering the theoretical models and then developing the instance models from them to solve the problem. During this refinement from goals to theory to mathematical models, we apply different assumptions, build general definitions and data definitions, which are also to be documented. The proposed template aids in documenting all the necessary information, as each section has to be considered even if it is inappropriate for the given problem while developing the SRS. This facilitates the achievement of completeness by providing a checklist for the questions that are to be asked and for the information that is to be filled in. However, it is not enough to just fill in the sections of the template for achieving completeness and correctness. One has to make sure that the list of conditions given in the checklist (Appendix C) are satisfied by the SRS. For instance, the template does not document derivation of equations. Yet, for checking the correctness of the data definitions, it is a good idea to include the derivations. Hence both the template and the checklist are to be considered while developing the SRS.

1. Reference Material:
a) Table of Symbols
b) Abbreviations and Acronyms
2. Introduction:
a) Purpose of the Document
b) Scope of the Software Product
c) Organization of the Document
d) Intended Audience
3. General System Description:
a) System Context
b) User Characteristics
c) System Constraints
4. Specific System Description:
a) Problem Description:
i) Background Overview, ii) Terminology Definition, iii) Physical System Description, iv) Goal Statements
b) Solution Specification:
i) Assumptions, ii) Theoretical Models, iii) General Definitions, iv) Data Definitions, v) Instanced Models, vi) Data Constraints, vii) System Behavior
c) Non-functional Requirements:
i) Accuracy of Input Data, ii) Sensitivity of Model, iii) Tolerance of Solution, iv) Solution Validation Strategies, v) Look and Feel Requirements, vi) Usability Requirements, vii) Performance Requirements, viii) Maintainability Requirements, ix) Portability Requirements, x) Security Requirements
5. Other System Issues:
a) Open Issues
b) Off the Shelf Solutions
c) Waiting Room
6. Traceability Matrix
7. List of Possible Changes in the Requirements
8. Values of Auxiliary Constants

Figure 3.1: Contents of SRS

All of the sections of Lei Lai's template are borrowed for the current work. Although as mentioned previously, not all of the details are fleshed out for Section 4.c, Non-functional Requirements and Section 5, Other system issues in the present case study. Furthermore, the following additions were made: A new subsection called General Definitions under Specific System Description and new contents for the Table of Symbols. These changes were helpful for the current case study. The important sections of this template that are used to improve the desired qualities of an SRS are presented below with an introduction to their motivation and content [28].

3.2.1 Table of Symbols

Motivation
To summarize the symbols used in the document and to give a quick reference for the symbols specifically defined in the SRS.

Content
The symbols used in the SRS are explained along with their units. The section is again divided into subsections to record the quantities related to thermal analysis separately from quantities related to nuclear physics. This is to make the context in which the symbol is used clear. Example of table of symbols used in the SRS of FP is included below, as Figure 3.2.

Quantities related to Thermal Analysis:	
C_i	- thermal capacitance terms indexed by i ($\frac{\text{kWs}}{\text{m}^{\text{o}}\text{C}}$)
h_b	- coolant film conductance ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
h_c	- convective heat transfer coefficient between clad and coolant ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
h_{dry}	- convective heat transfer coefficient between fuel surface and coolant at dryout ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
h_g	- effective heat transfer coefficient between clad and fuel surface ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
.	
.	
.	
Quantities related to Nuclear Physics:	
A_k	- value of trip parameter at t_k
K_i	- response fraction
q'_{MWR}	- metal water reaction heat ($\frac{\text{kW}}{\text{m}}$)
q'_{MWRI}	- integrated metal water reaction heat ($\frac{\text{kW}}{\text{m}}$)
.	
.	
.	

Figure 3.2: Excerpt from the SRS showing the Table of Symbols

3.2.2 Goals

Motivation
To collect and document the objectives of a system in the requirements process.

Content
A goal statement should specify the target of the system. The goal must be abstract. That is, it should be a specification indicating what the system is expected to perform, but not the ways of achieving the objective. For instance, a goal from our case study is:

G2: Given the neutron flux versus time as input, predict transient reactor fuel and clad temperatures.

3.2.3 Assumptions

Motivation
To record the assumptions that have to be made or have been made while developing the software.

Content
An assumption is a specification showing the approximation to be made while solving a

problem. We suggest that assumptions are documented with the forward references made to the data using them. An example of documenting assumptions is given below:

A9: The spacial effects are neglected in the reactor kinetics formulations [IM5].

3.2.4 Theoretical Models

Motivation
To develop an understanding of the theory or principles relevant to the problem [28].

Content
The theoretical models are sets of governing equations or axioms that are used to model the problem described in the problem definition section. Theoretical models give an introduction of the theory. In the context of nuclear physics, the theoretical models can be physical laws (include relevant equations), constitutive equations, etc. Given below is an example of a theoretical model from our case study.

Number	T1
Label	**Conservation of energy**
Equation	$-\nabla \mathbf{q} + q''' = \rho C \frac{\partial T}{\partial t}$
Description	The above equation gives the conservation of energy for a time varying heat transfer in a material of specific heat capacity C and density ρ where $\mathbf{q}$ is the thermal flux vector, q''' is the volumetric heat generation, T is the temperature and ∇ is the gradient operator.

The conservation of energy equation is the most important theoretical model of our case study as it forms the foundation for the derivation of the mathematical models of FP. How the symbolic equation of conservation of energy is used in deriving the instance models for FP is shown in Appendix A. The theory must be given as abstractly as possible to make it reusable for other problems. The theory will be later refined to instance models by applying assumptions and definitions. For example, in the theoretical model given above, the coordinate system is not assumed. So it can be used for solving other problems, as it is not specific to one context. In the current case study, shown in Appendix A, a cylindrical coordinate system is assumed, but the general notation usually means that T1 can be used in a different context, say with a cartesian coordinate system.

3.2.5 General Definitions

Motivation
This is the new section included in the template to gather and document all the necessary data that will be repetitively used in deriving different data definitions.

Content
General definitions constitute the laws and equations that will be used indirectly in developing the mathematical models. That is, general definitions are those that will not directly model the problem but will be used in deriving the data definitions, which in turn are used to build the instance models. The general definitions are suggested to be documented using tabular and textual descriptions. The definition includes the following fields:

- Number- This gives the number of the general definition
- Label- This gives the name of the term being defined
- Symbol- This gives the symbol of the term
- Units- This gives the units of the term. The units are mentioned in the $MLTt$ system, where M represents Mass, L represents Length, T represents temperature and t represents time.
- SI equivalent- This gives the units of the term in SI system. That is, in terms of kilograms (kg), meter (m), Kelvin (K) and seconds (s).
- Equation- This gives the definition of the term
- Description- This gives the description of the terms being used in the definition and the information related to the definition.

Example of a general definition is given below:

Number	GD1
Label	**Cylindrical coordinate system**
Units	-
SI equivalent	-
Equation	$\nabla = \hat{r}\frac{\partial}{\partial r} + \hat{\theta}\frac{1}{r}(\frac{\partial}{\partial\theta}) + \hat{z}\frac{\partial}{\partial z}$ where $\hat{r}$, $\hat{\theta}$ and $\hat{z}$ are unit vectors. In matrix notation, this appears as: $\nabla = \begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{1}{r}\frac{\partial}{\partial\theta} \\ \frac{\partial}{\partial z} \end{bmatrix}$ The divergence ∇A is calculated as: $\nabla\mathrm{A} = \frac{\partial(A_r)}{\partial r} + \frac{1}{r}\frac{\partial A_\theta}{\partial\theta} + \frac{\partial A_z}{\partial z}$
Description	The spatial location in a cylindrical coordinate system is expressed in terms of $\hat{r}$, $\hat{\theta}$, $\hat{z}$. The gradient operator is defined as shown above.
Sources	[8, page 12];

3.2.6 Data Definitions

Motivation
To collect and organize all physical data needed to solve the problem [28].

Content
All the symbols that are used in developing the mathematical models of the system are defined using a tabular representation. The symbol should be defined with the meaning of the physical data they represent and needs to be given a unique label to support traceability. If any equation is defined in this section, then it is recommended to include the derivation of that equation under the table of the definition. The data definition includes the same fields as that of the general definitions with an additional source field which gives the main source from which the definition was taken. Example of data definition is given below:

Number	DD3
Label	**Integrated fuel power**
Symbol	$P_{\mathrm{F,SUM}}$
Units	FPS
SI equivalent	-
Equation	$P_{\mathrm{F,SUM}}(t_i)= \int_0^{t_i} q'_{\mathrm{NFRAC}}(t)dt$
Description	The above equation gives the integrated fuel power at t_i, where q'_{NFRAC} is the relative fuel power and $P_{\mathrm{F,SUM}}(t_i)$ is the integrated fuel power at t_i
Sources	[8, page 12];

3.3 An Introduction to our Case Study

In this section, we give a brief introduction of our case study so that the reader can have a better understanding of what this research is based on. The full SRS for this case study is presented in Appendix A. Our case study is on a safety analysis software, referred to as FP, that has been developed by a nuclear power generating company in 1980's. The purpose of FP is to perform thermal analysis of a single fuelpin in the reactor. Each fuelpin includes the following elements as presented in Figure 3.3:

- A fuel pellet made of Uranium dioxide (UO_2).
- The clad material zircaloy covering the pellet.
- Coolant surrounding the clad material.

The software is used for running safety analysis cases. The analysis of one fuelpin by FP is used to give insight into the use of multiple pins. The goals of FP are stated as follows:

G1: Given fuel power versus time as input, predict transient reactor fuel and clad temperatures.

G2: Given the neutron flux versus time as input, predict transient reactor fuel and clad temperatures.

G3: Given the reactivity transient as input, predict transient reactor fuel and clad temperatures.

G4: Given the trip set points, number of trips to initiate shutdown, shutdown reactivity transient as inputs, simulate reactor trip and shutdown.

FP uses point neutron kinetics, decay heat equations, lumped parameter fuel modelling techniques, temperature dependent thermodynamic properties, a metal water reaction model, fuel stored energy, integrated fuel power calculations and trip parameter modelling to do the thermal analysis.

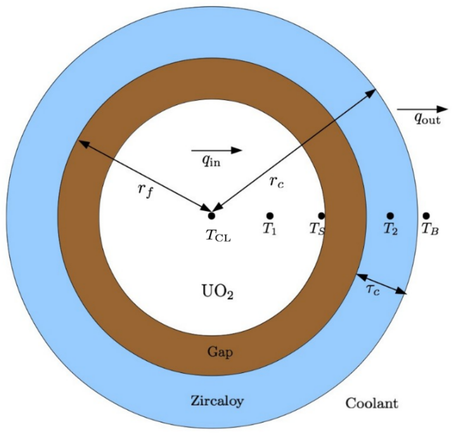


Figure 3.3: Fuel pellet representation

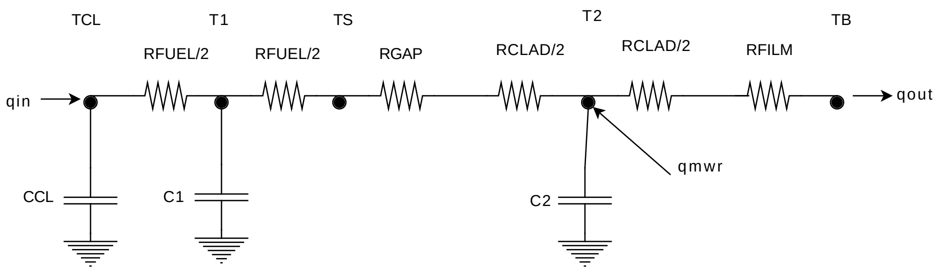


Figure 3.4: Electrical Circuit Analogue

The electrical circuit analogue of fuelpin representation is given by the Figure 3.4. This figure will be discussed further in Section 3.4. A summary of the variables for Figure 3.3 and Figure 3.4 is given below with their interpretation:

T_S	- surface temperature (^{o}C)
T_1	- average fuel temperature (^{o}C)
T_2	- average clad temperature (^{o}C)
T_B	- coolant temperature (^{o}C)
T_{CL}	- centreline temperature (^{o}C)
r_c	- clad radius (m)
r_f	- fuel radius (m)
τ_c	- clad thickness (m)
q_{in}	- input heat ($\frac{\text{kW}}{\text{m}^2{}^o\text{C}}$)
q_{out}	- output heat ($\frac{\text{kW}}{\text{m}^2{}^o\text{C}}$)
q'_{MWR}	- metal water reaction heat ($\frac{\text{kW}}{\text{m}}$)
R_{FUEL}	- thermal resistance of fuel ($\frac{\text{m}^o\text{C}}{\text{kW}}$)
R_{CLAD}	- clad resistance ($\frac{\text{m}^o\text{C}}{\text{kW}}$)
R_{GAP}	- gap resistance ($\frac{\text{m}^o\text{C}}{\text{kW}}$)
R_{FILM}	- coolant film resistance ($\frac{\text{m}^o\text{C}}{\text{kW}}$)
C_1	- thermal capacitance of the fuel ($\frac{\text{kWs}}{\text{m}^o\text{C}}$)
C_2	- thermal capacitance of the clad ($\frac{\text{kWs}}{\text{m}^o\text{C}}$)
C_{CL}	- thermal capacitance at the centerline ($\frac{\text{kWs}}{\text{m}^o\text{C}}$)

The mathematical models for this problem are the ordinary differential equations of different temperatures that are to be predicted, with appropriate initial conditions (see Appendix A for more details). For instance, the instance model for predicting transient fuel temperature is given below:

Number	IM1
Label	Rate of change of average fuel temperature
Equation	$C_1 \frac{dT_1}{dt} = q'_N - \frac{T_1 - T_2}{R_1}$
Description	T_1 is the average fuel temperature T_2 is the clad temperature R_1 is the effective resistance between fuel and clad temperatures C_1 is the thermal capacitance of the fuel q'_N is the linear element power t is the time
Sources	[8, page 6];

To achieve the goal of predicting transient fuel temperatures, we considered some theoretical models and developed the above instance model by making necessary assumptions and using required general definitions, data definitions. The derivation of IM1 is shown in the Appendix A. The functional requirement of FP software is to solve this instance model along with the other instance models.

3.4 Evaluation of the Template

Even though the original FP documentation was better than other software documentations that were developed in 1980's, there were a few issues with the theory manual. This section assesses the proposed template by showing how the desired qualities of a good SRS, as mentioned in Section 2.2, are achieved. To do this, we show the issues faced with the original documentation of requirements in the theory manual and then present our solution. Each issue is organized as follows:

- First, the context under which the issue has occurred is explained by giving details about the goal of the specification, mathematical equation of the specification, components of specification and definitions of the components.
- Second, we discuss how the issue has not met the desired qualities of a good SRS and the problems faced as a result.
- Third, we present the solution provided by the proposed template using an excerpt extracted from the SRS given in Appendix A.
- Finally, comments are made on the excerpt.

3.4.1 Issues in Documentation of R_1 (Effective Thermal Resistance Between T_1 and T_2)

3.4.1.1 Context from the Original Theory Manual

One of the goals of FP, as mentioned in Section 3.3 is to find the transient fuel temperature. The instance model representing this goal is an ODE as given by IM1, with R_1 being the effective resistance between fuel and clad temperatures.

In the original theory manual, the equation for R_1 is documented as,

$$R_1 = \frac{R_{\text{FUEL}}}{2} + R_{\text{GAP}} + \frac{R_{\text{CLAD}}}{2} \tag{3.1}$$

$$R_1 = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r h_g} + \frac{\tau_c}{4\pi r k_c}, \tag{3.2}$$

with k_{AV} being the average thermal conductivity,
r being the fuel radius,
h_g being the gap conductance,
R_{FUEL} being the thermal resistance of fuel,
R_{CLAD} being the clad resistance,
R_{GAP} being the gap resistance,
τ_c being the clad thickness,
k_c being the clad conductivity.

3.4.1.2 Problems

1. *Incompleteness*
   As per the definition of completeness in Section 2.2, every term in the document should be defined. However, the term R_{GAP} that was used in Equation 3.1 was not defined in the manual. So, the original specification of R_1 has failed to achieve the quality of completeness

2. *Deficiencies in checking correctness*
   From Figure 3.4, it is evident that the effective resistance R_1 between T_1 and T_2 is:

   $$R_1 = \frac{R_{\text{FUEL}}}{2} + R_{\text{GAP}} + \frac{R_{\text{CLAD}}}{2}, \tag{3.3}$$

   Hence Equation 3.1 is correct. However, to check the correctness of Equation 3.2, we need the derivations of R_{FUEL}, R_{GAP} and R_{CLAD}. As the derivation for R_{GAP} was not given in the manual, checking the correctness of Equation 3.2 became difficult, as we had to derive the equation of R_{GAP} for the new SRS.

3. *Inconsistencies*
   The inconsistencies in the use of symbols for the terms gap conductivity and fuel radius in the manual are as follows:

   - The fuel radius was symbolized as 'r' in the R_{CLAD} equation while it was represented by 'r_0' in the derivation of average fuel temperature (k_{AV}).
   - The term h_g has been used for representing gap conductance in the R_1 equation while the same h_g denotes effective heat transfer coefficient between clad and fuel surface in the fuel surface temperate (T_S) equation , which is given in the theory manual as,

     $$T_S = T_2 + \frac{q'_N}{2\pi r h_g} \tag{3.4}$$

   The above two issues show that there are inconsistencies in the terms of the R_1 equation.

4. *Verifiability problem*
   As per the definition of verifiability in the Section 2.2, every specification in the document must be the one fulfilled by the software. However, in the original FORTRAN code, R_1 was solved as,
   $$R_1 = \frac{f}{8\pi k_{AV}} + \frac{1}{2\pi r_f h_g} \tag{3.5}$$
   This clearly shows that there is a disconnect between the original theory and the code.

5. *Lack of traceability:*
   The problems regarding traceability are as follows:

   - There was no traceability shown between the figure representing the electrical circuit analogue of the fuel pellet and the derivation of R_1. The figure is required at the time of verifying R_1 as it gives an insight into the derivation of the equation. However, the figure was given at the end of the manual and there was no referencing to it while defining R_1. This made the task of checking for correctness difficult.
   - As mentioned previously, there was no traceability to the term R_{GAP} from R_1 as there was no definition for it in the manual. The lack of traceability leads to a *modifiability* problem as well, since managing changes requires knowledge of the impact of the changes.

3.4.1.3 Quality Improvements by the Proposed Template

To achieve the qualities of a good SRS, the template applies the principle of “separation of concerns” by including different sections so that focus can be on one thing at a time. By dividing the problem into smaller steps and considering each section, it is easier to document all the necessary information completely and correctly. The sections like Theoretical Models, General Definitions, Data Definitions are included before the instance models section for the purpose of systematically solving the problem in a hierarchical way. This way of developing the concrete models from abstract ones helps in achieving completeness, consistency, traceability and verifiability in the documentation. The purpose of including these sections is to document all the background information, physical laws, constitutive equations, rules, principles and physical data required to solve the problem.

To tackle the inconsistency problem, the template includes a section called ‘Table of Symbols’ where all the symbols used in the document are summarized along with their units.

To deal with the problem of traceability, and modifiability, we use cross referencing between the components. The template requires the use of a unique label to each component and the development of models in a hierarchical manner. This aids in making references back and forth, thus achieving traceability.

To solve the problems with completeness and correctness, the template uses the Assumptions, Theoretical models, General definitions and Data definitions sections as shown in the below Excerpt 3.4.1.4. These sections collect all the necessary data needed to build the equations, mathematical models and define them using tabular representation as well as textual description. The mathematical models are built from the theoretical models considering the approximations and using the data definitions. The data definitions are derived from the General definitions. This way of developing the concrete models from the abstract ones, maintaining traceability between them will aid in achieving correctness. As the derivations of the equations are also included in these sections, it makes the checking of correctness easier. Hence the completeness as well as the correctness can be achieved at the same time by documenting every equation, assumption, definition and model in the respective sections. Once the requirement is specified completely, correctly with traceability to its components, then the task of verifiability becomes easier.

3.4.1.4 Excerpt 1 Showing the Documentation of R_1 Using the Proposed Template

The following is an excerpt from the SRS given in Appendix A, highlighting the portions relevant for the definition of R_1. The information not relevant in the current context has been removed. The locations where this has occurred are indicated by vertical ellipses $\left(\vdots\right)$. NOTE: To allow the presentation focus on the main points and to keep the length reasonable, the derivations of the data definitions used in deriving R_1 are not included in the excerpt and can be found in Appendix A.

h_g - effective heat transfer coefficient between clad and fuel surface ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
h_p - initial gap film conductance ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
k_c - clad conductivity ($\frac{\text{kW}}{\text{m}^\text{o}\text{C}}$)
k_{AV} - average thermal conductivity ($\frac{\text{kW}}{\text{m}^\text{o}\text{C}}$)
r_c - clad radius (m)
r_f - fuel radius (m)
R_{FUEL} - thermal resistance of fuel ($\frac{\text{m}^\text{o}\text{C}}{\text{kW}}$)
R_{CLAD} - clad resistance ($\frac{\text{m}^\text{o}\text{C}}{\text{kW}}$)
R_{GAP} - gap resistance ($\frac{\text{m}^\text{o}\text{C}}{\text{kW}}$)
τ_c - clad thickness (m)
$\vdots$

Figure 3.5: Table of Symbols

Assumptions:
A6: Approximation for $\ln \frac{r_o}{r_i}$ as $\frac{\tau_c}{r}$ [DD7].
$\vdots$

General Definitions:

Number	GD3
Label	**Effective thermal resistance**
Symbol	R_{EFF}
Units	ML^3Tt^{-3}
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{EFF}} = \frac{\Delta T}{q}$
Description	In some cases at steady state, the relation between the temperature change (ΔT) and the thermal flux (q) is modelled as ΔT being directly proportional to q. The proportionality constant can be derived using thermodynamic theory and then relabelled as R_{EFF}. This is analogous to the electric circuit equation of $V{=}IR$. As for the case of electric resistors in series, if n resistors $(R_1, R_2.....R_n)$ are connected in series between two temperatures and if constant heat is flowing between those temperatures, then $R_{\text{EFF}} = R_1 + R_2 + ... + R_n$

⋮

Data Definitions:

Number	DD6
Label	**Effective thermal resistance of fuel**
Symbol	R_{FUEL}
Units	ML^3Tt^{-3}
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{FUEL}} = \frac{f}{4\pi k_{\text{AV}}}$
Description	R_{FUEL} is the effective thermal resistance between the temperatures T_{CL} and T_S. $\frac{R_{\text{FUEL}}}{2}$ is the effective thermal resistance between T_{CL} and T_1 and between T_1 and T_S.
Sources	[8, page 3];

Derivation of R_{FUEL}:

⋮

Number	DD7
Label	R_{CLAD}
Units	$M^{-1}L^{-1}Tt^3$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{CLAD}} = \frac{\tau_c}{2\pi r_c k_c}$
Description	The clad resistance is a function of the clad thermal conductivity. It is obtained from the expression for heat transfer by conduction through a hollow cylinder with inner radius r_i and outer radius r_o where k_c is the clad conductivity ($\frac{\text{kW}}{\text{m}^{\text{o}}\text{C}}$) and is given as, $\frac{\Delta T}{q} = \frac{\ln \frac{r_o}{r_i}}{2\pi k_c}$ Taking A6 into consideration, we get $\frac{\Delta T}{q} = \frac{\tau_c}{2\pi r_c k_c}$ Comparison to GD3, shows that effective thermal resistance $R_{\text{CLAD}} = \frac{\tau_c}{2\pi r_c k_c}$
Sources	[8, page 4], [18, page 5] ;

Derivation of $R_{\mathbf{CLAD}}$:

$\vdots$

Number	DD8
Label	R_{GAP}
Units	$M^{-1}L^{-1}Tt^{3}$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{GAP}} = \frac{1}{2\pi r_c h_p}$
Description	R_{GAP} is the gap resistance where r_c is the clad radius (m), and h_p is the initial gap conductance ($\frac{\text{kW}}{\text{m}^{2\text{o}}\text{C}}$), which is an input parameter
Sources	source code

Derivation of $R_{\mathbf{GAP}}$:

$\vdots$

Number	DD10
Label	R_3
Units	$ML^{3}Tt^{-3}$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_3 = \frac{1}{2\pi r_f h_g}$
Description	R_3 is the effective thermal resistance between T_S and T_2 where r_f is the fuel radius (m) h_g is the gap film conductance (kw/$m^{2o}C$), which is given by DD19
Sources	[8, page 5];

Derivation of R_3:

$\vdots$

Number	DD19
Label	h_g
Units	$Mt^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^{2\circ}\text{C}}$
Equation	$h_g = \frac{2k_c h_p}{2k_c + \tau_c h_p}$
Description	h_g is the gap conductance τ_c is the clad thickness h_p is initial gap film conductance k_c is the clad conductivity
Sources	source code

Derivation of h_g:

$\vdots$

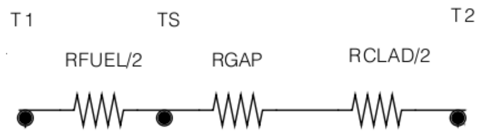


Figure 3.6: Thermal circuit between T_1 and T_2

Number	DD11
Label	R_1
Units	ML^3Tt^{-3}
SI equivalent	$\frac{\text{m}^\text{o}\text{C}}{\text{kW}}$
Equation	$R_1 = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_g}$
Description	R_1 is the thermal resistance between the average fuel temperature (T_1) and sheath temperature (T_2) (see Figure 3.6)
Sources	[8, page 4];

Derivation of R_1:
From the Figure 3.6, the effective resistance R_1 between T_1 and T_2 is:

$$R_1 = \frac{R_{\text{FUEL}}}{2} + R_{\text{GAP}} + \frac{R_{\text{CLAD}}}{2} \tag{3.6}$$

Substituting the values of R_{FUEL}, R_{CLAD} and R_{GAP} from DD6, DD7, DD8 respectively into the above equation, it can be written as,

$$R_1 = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_p} + \frac{\tau_c}{4\pi r_c k_c}, \tag{3.7}$$

3.4.1.5 Comments on the Excerpt

The derivation shows that the documentation in the theory manual was incorrect due to the inconsistent use of h_g. If the h_g in Equation 3.2 had been documented as h_p, then the equation would have been correct.

Since R_3 is the effective resistance of the gap and half of the clad, the Equation 3.6 can be rewritten as,

$$R_1 = \frac{R_{\text{FUEL}}}{2} + R_3 \tag{3.8}$$

Substituting the values of R_{FUEL} and R_3 from DD6 and DD10, respectively, into Equation 3.8 gives:

$$R_1 = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_g} \tag{3.9}$$

This new derivation clearly shows that the effects of R_{GAP} and R_{CLAD} are combined through the calculation of h_g. The newly derived equation matches with the code's implementation of R_1. The incompleteness and inconsistency of the original documentation initially led to the incorrect conclusion that either the theory or the implementation was wrong.

3.4.2 Issues in Documentation of R_2 (Effective Thermal Resistance Between T_B and T_2)

3.4.2.1 Context from the Original Theory Manual

One of the goals of FP, as mentioned in Section 3.3 is to find the transient clad temperature. The instance model representing this goal is an ODE showing the rate of change of clad temperature (T_2) which is given as:

Number	IM2
Label	Rate of change of average clad temperature
Equation	$C_2\frac{dT_2}{dt} = \frac{T_1-T_2}{R_1} + q'_{\text{MWR}} - \frac{T_2-T_B}{R_2}$
Description	T_1 is the average fuel temperature ($^{\text{o}}$C) T_2 is the clad temperature ($^{\text{o}}$C) T_B is the coolant temperature ($^{\text{o}}$C) R_1 is the effective resistance between fuel and clad temperatures ($\frac{\text{m}^o\text{C}}{\text{kW}}$) R_2 is the effective resistance between clad and coolant temperatures ($\frac{\text{m}^o\text{C}}{\text{kW}}$) C_2 is the thermal capacitance of the clad ($\frac{\text{kWs}}{\text{m}^o\text{C}}$) q_{MWR} is the Metal-Water reaction heat ($\frac{\text{kW}}{\text{m}}$)
Sources	[8, page 6];

In the theory manual, the equation for R_2 is documented as,

$$R_2 = [2\pi r_o(h_c + 2k_c/\tau_c)]^{-1} \tag{3.10}$$

where, r_o is the outer clad radius and h_c is the convective heat transfer coefficient between the clad and the coolant

3.4.2.2 Problems

1. *Incompleteness*
   The term h_c used in Equation 3.10 is not defined in the manual. So, the original specification of R_2 failed to satisfy the completeness condition, as every term is required to be defined for completeness to be achieved.

2. *Deficiencies in checking correctness*
   From Figure 3.4, it is evident that the effective resistance R_2 between T_2 and T_B is:

   $$R_2 = \frac{R_{\text{CLAD}}}{2} + R_{\text{FILM}}, \tag{3.11}$$

   To check the correctness of Equation 3.10, we need the derivations of R_{FILM} and R_{CLAD}. As the derivation for R_{FILM} was not given in the manual, checking the correctness of Equation 3.10 became difficult.

3. *Verifiability problem*
   In the original FORTRAN code, R_2 was solved as,

   $$R_2 = \frac{1}{2\pi r_c h_c} \tag{3.12}$$

This shows that there is a disconnect between the original theory and the code. As every specification in the document must be the one fulfilled by the software for verifiability to be achieved, it can be concluded that the verifiability condition has been violated.

4. *Lack of traceability*
   The problems regarding traceability are similar to that of R_1's problems, as follows:
   - There was no traceability shown between the figure representing the electrical circuit analogue of the fuel pellet and the derivation of R_2.
   - As mentioned previously, there was no traceability to the term R_{FILM} from R_2, as there was no definition for it in the manual.
5. *Ambiguity*
   The assumptions necessary to derive R_2 were not documented. This made the task of deriving the definition for R_2 very difficult, as we were trying to get the theory manual's equation of R_2 making our own assumptions. Due to the lack of assumptions, several approximations were made, leading to a state of ambiguity.

3.4.2.3 Quality Improvements by the Proposed Template

To provide a solution to the problems encountered with the original documentation of R_2, we followed the same procedure that was used to deal with problems of R_1. The complete procedure is summarized as follows:

- Completeness and correctness problems were solved using the theoretical models, general definitions and data definitions sections.
- The problem of traceability, and modifiability, were dealt with, by giving a unique label to each component and by providing cross references between the components.
- The verifiability problem was tackled by developing the data in a hierarchical manner, that is, from abstract to concrete. Verifiability has been facilitated by ensuring the data is completely, and correctly documented with traceability to its components.
- The problem of ambiguity was solved by documenting all the necessary assumptions required to derive the terms in the Assumptions section with a forward reference given to the definition or model using it.

The following Excerpt 3.4.2.4 shows how the proposed SRS template solves the problems encountered with the original documentation of R_2. As for the previous excerpt, the derivations of the data definitions used to develop R_2 are not included in the excerpt and can be found in the Appendix A. Also as before, vertical ellipses $\left(\vdots\right)$ are used to indicate material removed from the excerpt. Some of the assumptions, general definitions and data definitions that R_2 uses are not defined in Excerpt 3.4.2.4, as they have already been defined in Excerpt 3.4.1.4. So, to avoid redundancy, we just refer to the previous definitions.

3.4.2.4 Excerpt 2 Showing the Documentation of R_2 Using the Proposed Template

h_c - effective heat transfer coefficient between clad and coolant ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
h_b - coolant film conductance ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
k_c - clad conductivity ($\frac{\text{kW}}{\text{m}^\text{o}\text{C}}$)
r_c - clad radius (m)
R_{FILM} - coolant film resistance ($\frac{\text{m}^\text{o}\text{C}}{\text{kW}}$)
R_{CLAD} - clad resistance ($\frac{\text{m}^\text{o}\text{C}}{\text{kW}}$)
τ_c - clad thickness (m)
$\vdots$

Figure 3.7: Table of Symbols

Assumptions:

A7: Assume isotropic thermal conductivity [T2].
A11: Newton's law of convective cooling applies between the clad surface and the coolant film [DD9].
$\vdots$

Theoretical Models

Number	T2
Label	**Constitutive Equation (Fourier's Law)**
Equation	**q**$=-k(T)\nabla T$
Description	Fourier's law states that the heat flux is propositional to slope or the gradient of temperature, where k is a function of temperature. This law is based on the assumption that the material is isotropic A6.

⋮

General Definitions

Number	GD5
Label	**Newton's law of cooling**
Equation	$q_{\text{newt}} = hA\Delta T(t)$
Description	Newton's law of cooling describes the convection cooling and is stated as "rate of heat loss of a body is proportional to the difference in temperatures between the body and its surroundings." q_{newt} is the thermal flux. h is the heat transfer coefficient (assumed independent of T here) $(\frac{\text{W}}{\text{m}^2\text{K}})$ A is the surface area of the heat being transferred (m^2) $\Delta T(t) = T(t) - T_{\text{env}}$ is the time-dependent thermal gradient between environment and object. Newton's law of cooling can be derived from Fourier's law (T2)

Number	GD6
Label	**Effective heat transfer coefficient**
Equation	$h_{\text{EFF}} = \frac{q}{A\Delta T(t)}$
Description	q is the thermal flux. h_{EFF} is the effective heat transfer coefficient $(\frac{\text{W}}{\text{m}^2\text{K}})$ A is the surface area of the heat being transferred (m^2) $\Delta T(t) = T(t) - T_{\text{env}}$ is the time-dependent thermal gradient between environment and object. The heat transfer coefficient is modelled after Newton's law of cooling. It takes into account all relevant modes of heat transfer.

⋮

Data Definitions

Number	DD7
Label	R_{CLAD}
Units	$M^{-1}L^{-1}Tt^{3}$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{CLAD}} = \frac{\tau_c}{2\pi r_c k_c}$
Description	The clad resistance is a function of the clad thermal conductivity. It is obtained from the expression for heat transfer by conduction through a hollow cylinder with inner radius r_i and outer radius r_o where k_c is the clad conductivity ($\frac{\text{kW}}{m^oC}$) and is given as, $\frac{\Delta T}{q}$=$\frac{\ln\frac{r_o}{r_i}}{2\pi k_c}$ Taking A6 into consideration, we get $\frac{\Delta T}{q}$= $\frac{\tau_c}{2\pi r_c k_c}$ Comparission to GD3, shows that effective thermal resistance $R_{\text{CLAD}} = \frac{\tau_c}{2\pi r_c k_c}$
Sources	[8, page 4], [18, page 5] ;

Number	DD9
Label	R_{FILM}
Units	$M^{-1}L^{-1}Tt^{3}$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{FILM}} = \frac{1}{2\pi r_c h_b}$
Description	R_{FILM} is the coolant film resistance where r_c is the outer clad radius (m), h_b is the coolant film conductance ($\frac{\text{kW}}{m^2C}$) (see Figure 3.8)
Sources	source code

Derivation of R_{FILM}

Taking A10 into consideration, we use Newton's law of cooling to derive R_{FILM}. The area of the clad (A_c) is

$$A_c = 2\pi r_c \tag{3.13}$$

Substituting Equation 3.13 into GD6, and considering h_b as the coolant film conductance, we get,

$$q_{\text{coolant}} = 2\pi r_c h_b \Delta T \tag{3.14}$$

From GD3, the coolant film resistance (R_{FILM}) can be given as,

$$R_{\text{FILM}} = \frac{\Delta T}{q_{\text{coolant}}} \tag{3.15}$$

Substituting Equation 3.14 in Equation 3.15 and simplifying gives,

$$R_{\text{FILM}} = \frac{1}{2\pi r_c h_b} \tag{3.16}$$

Number	DD18
Label	h_c
Units	$Mt^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^{2o}\text{C}}$
Equation	$h_c = \frac{2k_c h_b}{2k_c + \tau_c h_b}$
Description	h_c is the effective heat transfer coefficient between the clad and the coolant τ_c is the clad thickness h_b is initial coolant film conductance k_c is the clad conductivity
Sources	source code

Derivation of h_c:

⋮

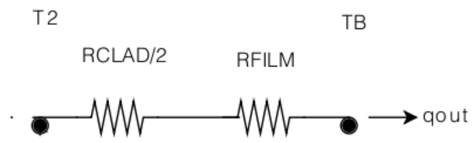


Figure 3.8: Thermal circuit between T_2 and T_B

Number	DD12
Label	R_2
Units	ML^3Tt^{-3}
SI equivalent	$\frac{\text{m}^o\text{C}}{\text{kW}}$
Equation	$R_2 = \frac{1}{2\pi r_c h_c}$
Description	R_2 is the effective thermal resistance between T_B and T_2 r_c is the outer clad radius (m) h_c is the effective heat transfer coefficient between clad and coolant ($\frac{\text{kw}}{\text{m}^{2o}\text{C}}$) which is given by DD18
Sources	[8, page 5];

Derivation of R_2:
Taking A10 into consideration, we use GD6 to derive R_2. Substituting Equation 3.13 into GD6, and considering h_c as the effective heat transfer coefficient between clad and coolant, we get,

$$q = 2\pi r_c h_c \Delta T \tag{3.17}$$

From GD3, R_2 can be given as,

$$R_2 = \frac{\Delta T}{q} \tag{3.18}$$

Substituting Equation 3.17 into Equation 3.18 and simplifying gives,

$$R_2 = \frac{1}{2\pi r_c h_c} \tag{3.19}$$

3.4.2.5 Comments on the Excerpt

The newly derived equation for R_2 matches with the code's implementation of R_2. The incompleteness and inconsistency of the original documentation led to the conclusion that either the theory or the implementation was incorrect.

From Figure 3.8, R_2 is the effective thermal resistance of the coolant film and half of the clad. Adding the R_{FILM} from DD9 with half of the value of R_{CLAD} from DD7, we get

$$R_2 = \frac{1}{2\pi r_c h_b} + \frac{\tau_c}{4\pi r_c k_c} \tag{3.20}$$

Taking the common terms out and rewriting the above equation we get,

$$R_2 = \frac{1}{2\pi r_c}\Big(\frac{1}{h_b} + \frac{\tau_c}{2k_c}\Big) \tag{3.21}$$

The derivation shows that the specification of R_2 in the theory manual was incorrect due to the inconsistent use of h_c and due to the incorrect documenting of the $\frac{R_{\text{CLAD}}}{2}$ part . If the h_c in Equation 3.10 had been documented as $\frac{1}{h_b}$ and $\frac{2k_c}{\tau_c}$ term as $\frac{\tau_c}{2k_c}$, then the equation would have been correct. However, we need further simplification of the equation to make the specification match the code. That is, Equation 3.21 should be simplified as follows:

$$R_2 = \frac{1}{2\pi r_c}\Big(\frac{2k_c + \tau_c h_b}{2k_c h_b}\Big) \tag{3.22}$$

$$= \frac{1}{2\pi r_c\Big(\frac{2k_c h_b}{2k_c + \tau_c h_b}\Big)} \tag{3.23}$$

As $\frac{2k_c h_b}{2k_c + \tau_c h_b}$ is h_c from DD18, substituting h_c in the RHS of the equation gives,

$$R_2 = \frac{1}{2\pi r_c h_c} \tag{3.24}$$

3.4.3 Issues in Documentation of Numerical Algorithm for Solving T_2 Quadratic Equation

3.4.3.1 Context From the Original Theory Manual

The ODE for predicting transient clad temperature T_2 is given by IM2. In steady state, the average clad temperature is determined by setting the time derivative terms to zero in IM1 and IM2 and neglecting the metal water heating term in IM2. The resulting equation is given as,

$$T_2 = T_B + q'_N R_2 \tag{3.25}$$

Substituting value of R_2 from the Equation 3.24 into Equation 3.25 and further simplifying the resulting equation by substituting the value of k_c, the clad conductivity (as given by DD15 of Appendix A) in it, gives a quadratic equation in T_2 as,

$$\frac{4\pi r_o a}{\tau_c} T_2^2 + 2\pi r_o[h_c + \frac{2(b - aT_B)}{\tau_c}]T_2 - T_B 2\pi r_o[h_c + \frac{2b}{\tau_c}] - q'_N = 0, \tag{3.26}$$

where a and b are constants obtained by a least squares fit to tabulated data, which are given as,

$$a = 1.43 \times 10^{-5} \tag{3.27}$$

$$b = 1.17 \times 10^{-2} \tag{3.28}$$

On further simplifying Equation 3.26 we get the quadratic equation in T_2 as,

$$\begin{aligned} 4\pi r_o a T_2^2 + \left(4\pi r_o b - 4\pi r_o a T_B + 2\pi r_o h_c \tau_c\right) T_2 \\ -(4\pi r_o T_B b + 2\pi r_o h_c \tau_c T_B + q'_N \tau_c) = 0 \end{aligned} \tag{3.29}$$

In steady state, the average clad temperature is found by solving the above quadratic equation. The root of this equation gives the initial clad temperature.

3.4.3.2 Problems

1. *Incompleteness*
   The term h_c that was used in Equation 3.26 was not defined in the manual.

2. *Incorrectness*
   From the issues in documentation of R_2 as mentioned in 3.4.2, it is shown that R_2 has been defined incorrectly. As the incorrect R_2 is used in the derivation of the quadratic equation in T_2, the resulting equation too is incorrect.

3. *Verifiability problem*
   As per the definition of verifiability in the Section 2.2, the equation must match with the one being solved in the code. However, in the original FORTRAN code, the quadratic equation in T_2 being solved is,

$$\begin{aligned} 4\pi r_c h_b a T_2^2 + \left(4\pi r_c h_b b - 4\pi r_c h_b a T_B - 2a q'_N\right) T_2 \\ -(4\pi r_c h_b T_B b + 2 q'_N b + q'_N h_b \tau_c) = 0 \end{aligned} \tag{3.30}$$

   This clearly shows the disconnect between the equation documented in the theory manual and the one being solved in the code.

4. *Failing to be abstract*
   Being abstract is one of the qualities that a good requirements document must possess. According to the definition of Abstraction given in the Section 2.2, the specification must tell us what to do but not how to do it. That is, the numerical algorithms used for solving the equations should not be discussed in the requirements documentation, as it is a design decision which should be taken during the design phase. Discussing the quadratic equation in T_2 for getting the value of clad temperature in steady state violates the condition for being abstract. Hence the original theory manual failed to achieve the quality of abstraction.

3.4.3.3 Quality Improvements by the Proposed Template

As the documentation of numerical algorithms is not supposed to be done in the SRS, the problems with quadratic equation of T_2 have been dealt with in the implementation part using Literate programming (see Chapter 4). However, for correctness and completeness, we ensured that all the terms used in the development of the quadratic equation are defined correctly in the SRS using the Data Definitions section. The improvement in documentation of R_2 using the proposed template has been shown in 3.4.2.4.

To ease modification of the value of auxiliary constants, we used the Auxiliary constants section of the template. In this section, the symbols, values, constraints and units of different constants that are required in solving the problem are recorded using tabular representation. Each table in the section corresponds to one particular term and contains all the constants required to define that term. The constants used for defining clad conductivity (k_c), which is used to develop the quadratic equation are documented in the auxiliary constants section as shown below:

TB2 Clad Conductivity

Constant	Value	Constraint	Units
a	1.43×10^{-5}	$T_2 \leq 1000^oC$	-
	2.73×10^{-5}	$T_2 > 1000^oC$	-
b	1.17×10^{-2}	$T_2 \leq 1000^oC$	-
	-1.27×10^{-3}	$T_2 > 1000^oC$	-

The way the algorithm has been developed using the correct R_2 is shown in the Chapter 4. For facilitating traceability and verifiability, we have given unique labels to data definitions, auxiliary constants tables in the SRS and developed the algorithm in the LP Manual making references to the data definitions and auxiliary constants in the SRS, as shown in the Chapter 4.

3.4.4 Issues in Documentation of Dryout Requirements

3.4.4.1 Context

When the reaction heat that is being sent into the coolant (q'_{out}) exceeds a critical value (p_{dry}), the coolant flow gets reduced. As the contact between the coolant and the fuel reduces, the heat transfer from the fuel surface into the coolant deteriorates. At one stage the coolant gets dried out and there will be no more contact between the fuel surface and the coolant. This is called *dryout* and when it occurs, the fuel surface temperature drastically increases. These intense high temperatures damage the fuel cladding. To avoid excessive fuel temperature, the knowledge of the onset of the dryout phenomena is absolutely necessary. When the dryout occurs, the value of heat transfer coefficient between the fuel and coolant changes due to the changing physics of the clad-coolant interaction. That is, the coolant conductance (h_b) is assigned the value of dryout heat transfer coefficient (h_{dry}).

NOTE: The theory manual does not discuss dryout and the context written above is from the knowledge acquired by reverse engineering.

3.4.4.2 Problems

1. *Incompleteness*
   All the information regarding dryout is missing from the theory manual. Neither the condition for checking dryout nor the actions to be taken when dryout occurs were given. Also the output heat to the coolant (q'_{out}), based on which the dryout occurs is not defined.

2. *Lack of traceability*
   As there is no theory documented about dryout and heat out, there is no traceability to those requirements for checking the correctness of their specifications.

3. *Lack of verifiability*
   There was no means to check the correctness of the implementation of dryout and

output heat.

4. *Modifiability*
Due to the lack of completeness and traceability, the task of modifying the implementation in future (if necessary) becomes difficult.

3.4.4.3 Quality Improvements by the Proposed Template

As the theory about dryout and heat out (q'_{out}) was missing, we had to develop the q'_{out} definition using the Theoretical Models, General Definitions and Data Definitions section. For the dryout, as we had no external means to know about the condition and actions to be taken. So, we applied the reverse engineering process to document the specification based on the code implementation. That is, we have taken the condition for dryout from the code and developed a data definition with it, as shown in the following excerpt:

3.4.4.4 Excerpt 3 Showing the Documentation of Dryout and Heat Out (q'_{out}) Using the Proposed Template

The information not relevant in the current context has been removed. The locations where this has occurred are indicated by vertical ellipses $\left(\vdots\right)$.

Table of Symbols:

h_b - coolant film conductance ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
h_{dry} - convective heat transfer coefficient between fuel surface and coolant at dryout ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
p_{dry} - dryout threshold power
q'_{out} - output heat to the coolant ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
$q'_{N_{\text{max}}}$ - linear element power at full power ($\frac{\text{kW}}{\text{m}}$)
T_B - coolant temperature ($^{\text{o}}$C)
$\vdots$

Figure 3.9: Table of Symbols

Data Definitions:

Number	DD27
Label	q'_{out}
Units	$Mt^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^2{}^\text{o}\text{C}}$
Equation	$q'_{\text{out}} = \frac{1}{q'_{N_{\max}}}\left(\frac{T_2-T_B}{R_2}\right)$
Description	q'_{out} is the output heat from the reaction that is sent into the coolant R_2 is the effective resistance between the coolant film and the clad $(\frac{\text{m}^\text{o}\text{C}}{\text{kW}})$ $q'_{N_{max}}$ is the linear element power at full power $(\frac{\text{kW}}{\text{m}^\text{o}\text{C}})$ T_2 is clad temperature ($^\text{o}$C) T_B is coolant temperature ($^\text{o}$C) NOTE: Equation taken from the original FORTRAN code

Derivation of q'_{out}:
From Figure 3.8, the effective resistance R_2 between T_2 and T_B is given by DD12. According to GD3, the R_2 between T_2 and T_B can be given as,

$$R_2 = \frac{\Delta T}{q'_{\text{out}}}, \tag{3.31}$$

where q'_{out} is the heat generated between T_2, T_B and

$$\Delta T = T_2 - T_B \tag{3.32}$$

Substituting Equation 3.32 and DD12 in Equation 3.31 and rearranging gives,

$$q'_{\text{out}} = \frac{(T_2 - T_B)}{R_2} \tag{3.33}$$

Normalizing the above equation by $q'_{N_{\max}}$ (standard presentation) gives,

$$q'_{\text{out}} = \frac{1}{q'_{N_{\max}}}\frac{(T_2 - T_B)}{R_2} \tag{3.34}$$

Number	DD28
Label	dryout
Units	–
SI equivalent	–
Equation	if($q'_{\text{out}} \geq p_{\text{dry}}$) $h_b = h_{\text{dry}}$
Description	We compare the power sent to the coolant (q'_{out}) with the dryout threshold power (p_{dry}) and once it exceeds this maximum permissible power, dryout occurs. When the dryout occurs, the coolant film conductance (h_b) becomes equal to the heat transfer coefficient between the fuel surface and the coolant at dryout (h_{dry}). NOTE: The meaning of h_b is given through the definition of R_{FILM} (DD9)

⋮

3.5 List of Issues Uncovered From the Original Theory Manual

There are some other issues with the original theory manual, but they are not being discussed in detail here. This is because some of the issues are fairly trivial and the problems encountered with the rest of the issues are similar to those mentioned above. A total of 27 issues that have violated the desired qualities of a good SRS including those discussed above, were uncovered by systematic documentation approach and using the checklist given in the Appendix C. The list of issues with the the theory manual, excluding the issues with R_1, R_2, T_2, dryout is provided in the Figure 3.10.

1. The derivation of the basic governing equation for temperature rate of change equations is not given.
2. The parabolic law used in determination of oxidation rate is not stated and its relation in determining oxidation rate is not given.
3. The derivation of R_{CLAD} expression from heat transfer through a hollow cylinder is not given.
4. Information about dryout, saturation is not given.
5. The value of the constant A in the temperature feedback reactivity is missing.
6. Units for temperature feedback reactivity are not given.
7. Inconsistent use of term T for average fuel temperature in the Fuel stored energy equation.
8. The last term of Fuel stored energy equation does not match the code as it is given as -$(E_D/R)T_1$ in manual but code says it is -$(E_D/(RT_1))$.
9. Functional forms of $c_{p,2}$ and $c_{p,3}$ were not stated.
10. Condition of q_{MWR} when $\delta_{\text{ox}} > \tau_c$ is not given in the manual.
11. The unit mk given for the temperature feedback reactivity is ambiguous.
12. Mismatch with the values of constants $K_0, E_D/R$ between the code and the theory.
13. The T used in the Fuel stored energy equation is not average fuel temperature. It is the absolute temperature equivalent of T_1. That is, $T = T_1 + 273$. This is not mentioned in the theory manual.
14. References to the electrical circuit analogue of the fuel pellet representation figure were not given.

Figure 3.10: List of Issues with the Original Theory Manual

Chapter 4

Implementation of the SRS using Literate Programming

Once the requirements are completely documented, the next stage in the software development is design. The design phase includes selection of the solution techniques, data structures, programming language and full development of the numerical algorithms. In this stage of software development, a plan on how to develop the computer program to meet the requirements is made. At this point, in traditional software engineering, the program structure is divided into modules, such that each module deals with a set of functional requirements. Then the modules are designed by preparing interfaces for them. After all the decisions are made, certified software should document the design decisions in such a way that it is clear how the requirements have been met.

After the design is complete, the next step in the waterfall model of a software development process is implementation. In the implementation stage, the design is translated into a computer program. That is, the coding is done for each module individually according to the design, and all the pieces of code are integrated into one main program. While developing the program, the program flow and naming conventions of the variables are often documented in a programmer's manual, so that the maintenance and modification of the program becomes easy. The programmer's manual is also helpful in verification of the implementation with the design and the requirements.

This chapter is about the design and implementation of the requirements mentioned in the SRS using Literate Programming (LP). The chapter is organized as follows: First, Section 4.1 discusses the information to be documented in the design manual and programmer's manual according to N286.7 standard. Second, Section 4.2 explains how Literate Programming has been used for designing and implementing the overall structure of the function being developed. Third, Section 4.3 gives the advantages of using LP for design and implementation and shows how the desirable qualities of a good document are achieved. Finally, Section 4.4 shows the design and implementation of instance models from the SRS using LP.

4.1 N286.7 Standard's Expected Documentation of Design and Implementation

According to N286.7 standard, the design description manual should be documented in such a way that it includes the following [3]:

- identification of the algorithms
- computer program structure, including data structures and program flow
- description of modules and module interfaces
- library functions

The N286.7 standard require the programmer's manual to be developed in such a way that it includes the following [3]:

- program flow and structure
- how the theory is translated into coding
- how to modify and maintain the computer program
- conventions on programming practices, such as variable naming and computer program

Although the standard requires two different documents for documenting design and implementation, we are going to combine both documents into one document: Literate Programmer's Manual (LPM). This is done because we are going to do the design and implementation simultaneously using LP. So the document generated by LP can act as both design manual and as a programmer's guide.

The division of the program into modules and specification of their interfaces is out of the scope of our work. We instead focus on one subroutine from the original FORTRAN program from developed by the nuclear power generating company. Their code has already been subdivided into subroutines. Our goal is to have the numerical outputs of the new literate program match the output of the original FP code. Keeping the output in sync is more manageable if we restrict our scope to one subroutine at this time. Hence, the division of modules and description of module interfaces is not included in the LPM. Our goal is to redevelop the existing code and document its implementation achieving the qualities of verifiability, traceability and correctness.

4.2 Design and Implementation Using LP

Although in the standard engineering practice, design and implementation are done in two different phases, we can document both the phases together using LP. This can be achieved because the source code is developed along with the logic behind it in LP. That is, first the design of each instance model is done by developing the numerical algorithms required to solve it. Second, the information behind the implementation of the instance model is given. Third, the data definitions required to develop the instance model are implemented as a small pieces of code called chunks. Finally, all the chunks are integrated into one main chunk. This way, all the instance models are designed and implemented as an interconnected web.

In our case study, we have restricted our scope to designing and implementing only those instance models that model the prediction of transient temperatures (both fuel and clad). That is, we are going to develop only the FUELTE subroutine of the original FORTRAN program using LP. As mentioned in Section 2.3, we used CWEB as our LP tool and C as the programming language. The literate C code is implemented in a function named *fuel_temp_*. We had to make a few changes to the original FORTRAN function, to make it run and generate the same results as that of the original code. The changes are summarized below:

- All the arguments (whether input or output values) to the C function (*fuel_temp_*) had to be passed by pointer in C, as FORTRAN by default passes all the arguments by reference.
- The common block variables of the FUELTE subroutine have been passed as input arguments to *fuel_temp_*, since the C implementation does not have access to the common block.
- Hardcoding of values was avoided and constants were used to denote them.
- The developed C function, *fuel_temp_* was called in the place of FUELTE subroutine from the original FORTRAN program to make the code run as one program.
- In the original FORTRAN code, the local variables were not declared under the save statement. So, once the subroutine ended, the local variables of that subroutine did not retain their previous value and hence became undefined. As a result, NaN's and infinities were generated in the output files. To deal with this problem, we have included the *-fno-automatic* option to the gfortran command while compiling the FORTRAN program, as shown in the make file (Appendix D). The -fno-automatic option specifies that all local variables and arrays are to be treated as if they were named in SAVE statements.

Calling of the C function from the FORTRAN code was verified by checking that the code generates the same results as that of the original nuclear power generating company's code. The complete implementation of the FUELTE subroutine using LP is shown in Appendix B.

A list of the changes made to the original FORTRAN code is given in Figure 4.1.

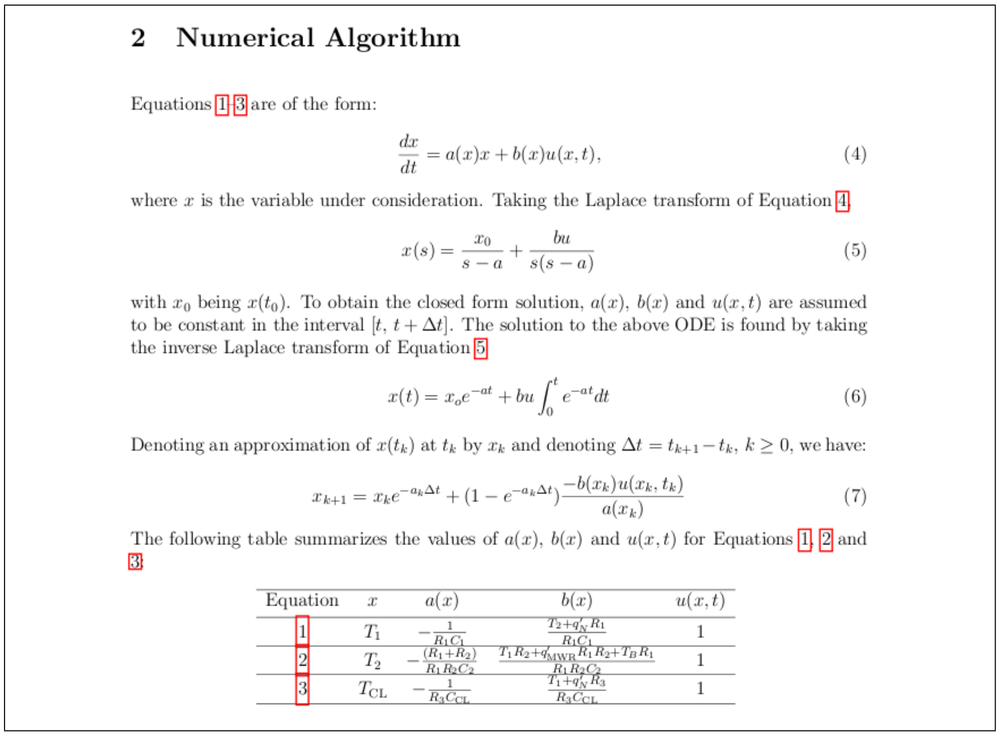
1. The open commands of input and output files are uncommented to make the code run.
2. Changed QN0 of the code to $q'_{\mathrm{N,max}}$ to make more sense.
3. Changed $q_{\mathrm{N,est}}$ equation by adding flux depression factor (f) to match the theory.
4. Added a new variable, pi instead of hardcoding the value 3.1416.
5. Added a new variable, t_std for stored energy calculation instead of hardcoding the value 298 in it.
6. Declared the stored energy constants in the dynamic section too instead of declaring them as global constants.
7. Added variables for $R_{\mathrm{FILM}}, R_{\mathrm{GAP}}, R_1, R_2, R_3$ instead of hard coding their values and to make the implementation match the data definitions from the SRS.

Figure 4.1: List of changes to the original FORTRAN code

For designing and implementing *fuel_temp_*, we will first discuss in Section 4.2.1, the numerical algorithm that has been developed to provide the transient solution to the ODEs represented by the instance models. Second, the algorithm and the overall function for developing the subroutine have been designed in Section 4.2.2. Third, the naming conventions of the input, output and local variables is given in Section 4.2.3. Finally, the design and implementation of the instance models representing the ODEs of different temperatures is done in Section 4.4. Excerpts from the LP document have been used to demonstrate the design and implementation of requirements from the SRS.

4.2.1 Numerical Algorithm for Solving ODEs

All the ODEs of the FP, which are given by instance models IM1, IM2 and IM3 of the SRS are solved using the same generic framework. This framework was originally given in the theory manual provided by the nuclear power generating company. However, as the numerical algorithms should not be discussed in the requirements document, the material about this framework has been removed from the SRS with the intent of keeping it abstract. Instead, the framework is described in the LP document as a numerical design decision. The framework shown in the Figure 4.2, is an excerpt from the LPM which gives the time stepping algorithm for solving ODEs of FP. The algorithm uses the subscript *k* to indicate the current time step.



2 Numerical Algorithm

Equations 1-3 are of the form:

$$\frac{dx}{dt} = a(x)x + b(x)u(x,t), \tag{4}$$

where x is the variable under consideration. Taking the Laplace transform of Equation 4

$$x(s) = \frac{x_0}{s-a} + \frac{bu}{s(s-a)} \tag{5}$$

with x_0 being $x(t_0)$. To obtain the closed form solution, $a(x)$, $b(x)$ and $u(x,t)$ are assumed to be constant in the interval $[t, t+\Delta t]$. The solution to the above ODE is found by taking the inverse Laplace transform of Equation 5

$$x(t) = x_o e^{-at} + bu \int_0^t e^{-at} dt \tag{6}$$

Denoting an approximation of $x(t_k)$ at t_k by x_k and denoting $\Delta t = t_{k+1} - t_k$, $k \geq 0$, we have:

$$x_{k+1} = x_k e^{-a_k \Delta t} + (1 - e^{-a_k \Delta t}) \frac{-b(x_k)u(x_k, t_k)}{a(x_k)} \tag{7}$$

The following table summarizes the values of $a(x)$, $b(x)$ and $u(x,t)$ for Equations 1, 2 and 3:

Equation	x	$a(x)$	$b(x)$	$u(x,t)$
1	T_1	$-\frac{1}{R_1 C_1}$	$\frac{T_2 + q'_N R_1}{R_1 C_1}$	1
2	T_2	$-\frac{(R_1+R_2)}{R_1 R_2 C_2}$	$\frac{T_1 R_2 + q'_{\text{MWR}} R_1 R_2 + T_B R_1}{R_1 R_2 C_2}$	1
3	T_{CL}	$-\frac{1}{R_3 C_{\text{CL}}}$	$\frac{T_1 + q'_N R_3}{R_3 C_{\text{CL}}}$	1

Figure 4.2: Excerpt from LPM showing the numerical algorithm for solving ODEs

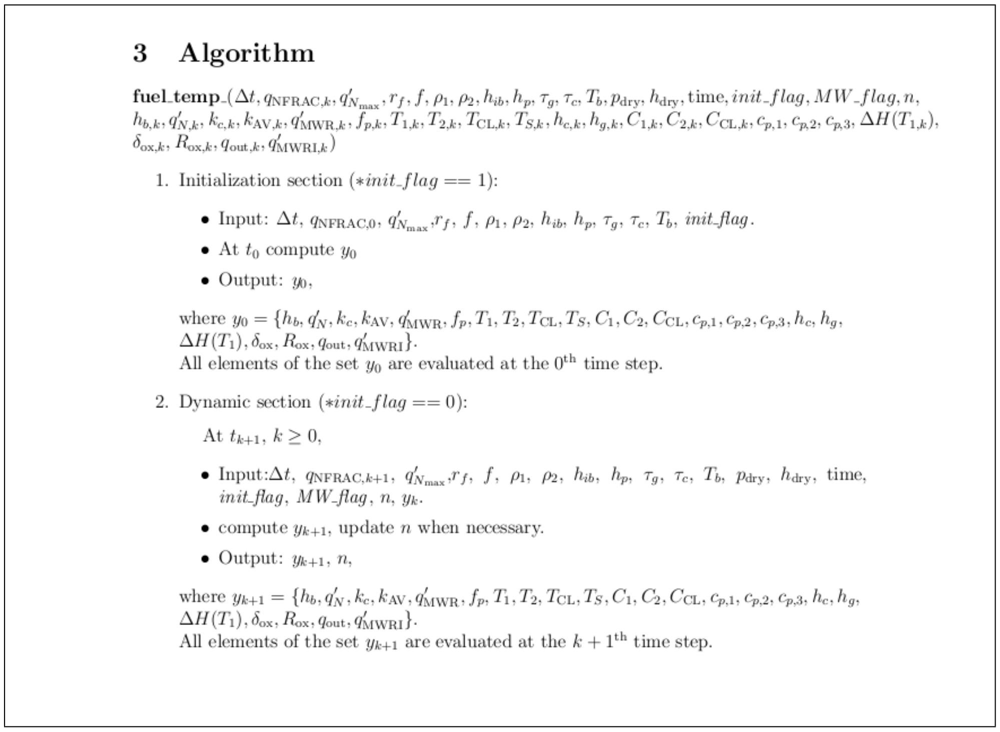
4.2.2 Overall Algorithm and Function of *fuel_temp_*

This section first discusses the overall algorithm of *fuel_temp_*, which is designed to get a clear idea of how to implement the instance models of the SRS. Next, it gives a high level view of the overall function structure.

The algorithm of the *fuel_temp_* function is designed to take both the input parameters as well as the common block parameters of the FUELTE subroutine from the original FORTRAN code, as inputs for the reason mentioned in Section 4.2. For making the algorithm understandable, we use the scientific notation of the variables in LaTeX. The overall algorithm of the *fuel_temp_* function is divided into two sections depending on the value of *init_flag*, which is an input, as follows:

- Initialization section: In this section, the initial values of the variables are found. That is, the values of variables in steady state are calculated.
- Dynamic section: In this section, the transient values of the variables are found.

Under each section, the inputs that are given to that section, the processing of inputs and the outputs from that section are detailed. That is, the algorithm gives a clear statement of the inputs and outputs. Figure 4.3 is an excerpt from the LPM, giving the overall algorithm of of *fuel_temp_*. The subscript *k* in the algorithm plays the same role as it did in Figure 4.2; it denotes the current time step.



3 Algorithm

fuel_temp_$(\Delta t, q_{\text{NFRAC},k}, q'_{N_{\max}}, r_f, f, \rho_1, \rho_2, h_{ib}, h_p, \tau_g, \tau_c, T_b, p_{\text{dry}}, h_{\text{dry}}, \text{time}, init\_flag, MW\_flag, n, h_{b,k}, q'_{N,k}, k_{c,k}, k_{\text{AV},k}, q'_{\text{MWR},k}, f_{p,k}, T_{1,k}, T_{2,k}, T_{\text{CL},k}, T_{S,k}, h_{c,k}, h_{g,k}, C_{1,k}, C_{2,k}, C_{\text{CL},k}, c_{p,1}, c_{p,2}, c_{p,3}, \Delta H(T_{1,k}), \delta_{\text{ox},k}, R_{\text{ox},k}, q_{\text{out},k}, q'_{\text{MWRI},k})$

1. Initialization section ($*init\_flag == 1$):
   - Input: Δt, $q_{\text{NFRAC},0}$, $q'_{N_{\max}}$,r_f, f, ρ_1, ρ_2, h_{ib}, h_p, τ_g, τ_c, T_b, *init_flag*.
   - At t_0 compute y_0
   - Output: y_0,

   where $y_0 = \{h_b, q'_N, k_c, k_{\text{AV}}, q'_{\text{MWR}}, f_p, T_1, T_2, T_{\text{CL}}, T_S, C_1, C_2, C_{\text{CL}}, c_{p,1}, c_{p,2}, c_{p,3}, h_c, h_g, \Delta H(T_1), \delta_{\text{ox}}, R_{\text{ox}}, q_{\text{out}}, q'_{\text{MWRI}}\}$.
   All elements of the set y_0 are evaluated at the 0^{th} time step.

2. Dynamic section ($*init\_flag == 0$):

   At t_{k+1}, $k \geq 0$,
   - Input:Δt, $q_{\text{NFRAC},k+1}$, $q'_{N_{\max}}$,r_f, f, ρ_1, ρ_2, h_{ib}, h_p, τ_g, τ_c, T_b, p_{dry}, h_{dry}, time, *init_flag*, *MW_flag*, n, y_k.
   - compute y_{k+1}, update n when necessary.
   - Output: y_{k+1}, n,

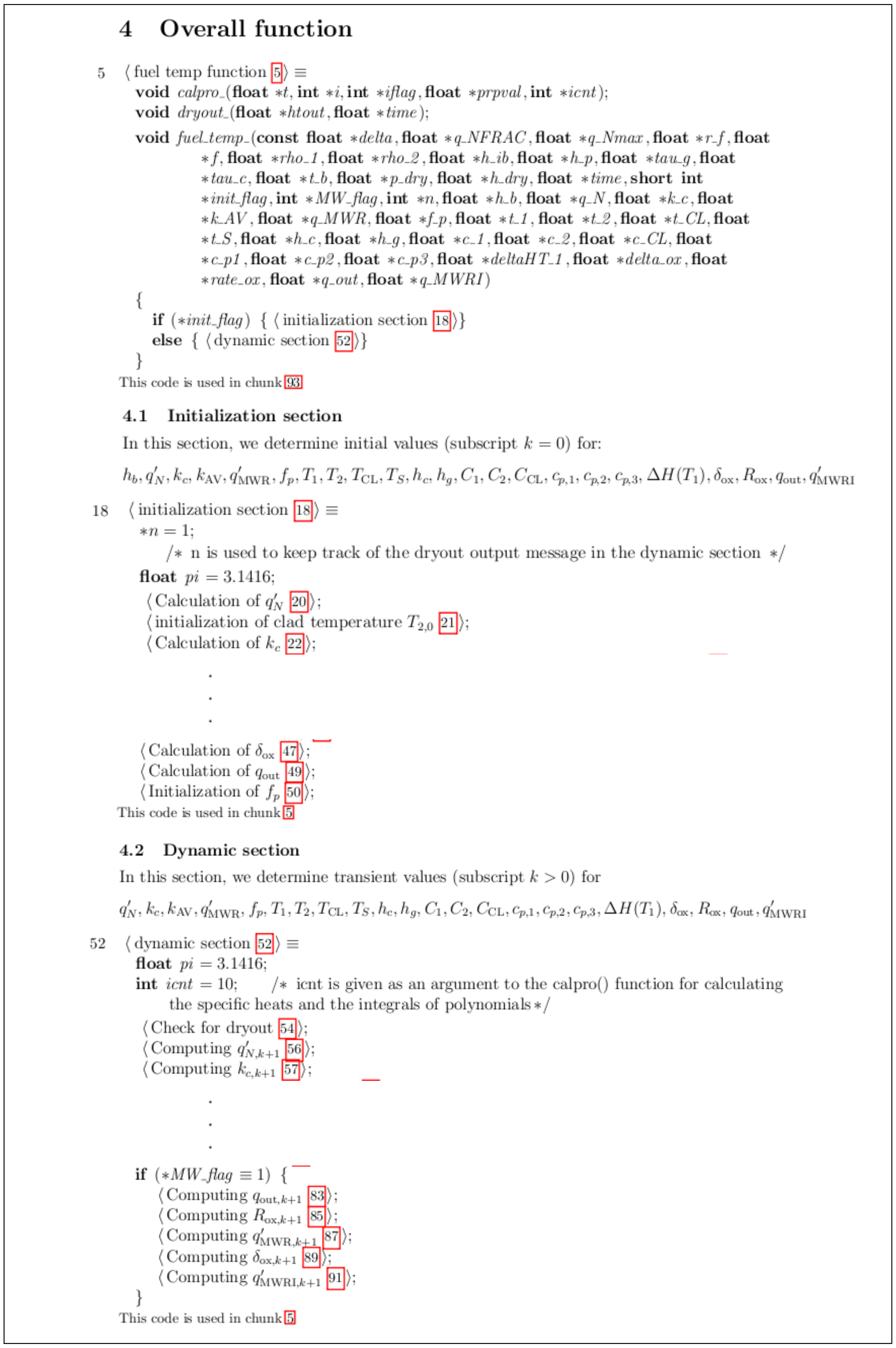
   where $y_{k+1} = \{h_b, q'_N, k_c, k_{\text{AV}}, q'_{\text{MWR}}, f_p, T_1, T_2, T_{\text{CL}}, T_S, C_1, C_2, C_{\text{CL}}, c_{p,1}, c_{p,2}, c_{p,3}, h_c, h_g, \Delta H(T_1), \delta_{\text{ox}}, R_{\text{ox}}, q_{\text{out}}, q'_{\text{MWRI}}\}$.
   All elements of the set y_{k+1} are evaluated at the $k + 1^{\text{th}}$ time step.

Figure 4.3: Excerpt from LPM showing the overall algorithm of *fuel_temp_*

After the algorithm has been developed, the next step is development of the function, in which the mathematical notation of the algorithm is converted into computer code. The development of the function is done as a stepwise refinement. That is, first, the overall function is developed in an abstract way, by calling the chunks which implement the initialization and dynamic sections. Later, the two sections are implemented in detail, as designed in the algorithm, in the chunks that are numbered beside the names of the sections, respectively.

In the coding phase, the scientific notation of the variables is changed to programming notation following the naming conventions which are given in the Section 4.2.3. The input parameters are passed by reference, as FORTRAN by default, uses pass by reference. Since the C function, *fuel_temp_*, has to be called from the FORTRAN code, it is necessary to declare the variables as pointers to make the C function compatible with the FORTRAN code and to make the variables retain their previous values even after exiting the function.

The function *fuel_temp_* calls two FORTRAN code subroutines: *calpro_* and *dryout_*. *calpro_* calculates material properties while *dryout_* outputs a message when dryout occurs. Hence, the overall function includes the declaration of these two subroutines. An excerpt from the LPM, showing the refinement from the abstract view of overall function to the concrete development of initialization and dynamic sections is given in the Figure 4.4.



4 Overall function

5 ⟨ fuel temp function 5 ⟩ ≡

```
void calpro_(float *t, int *i, int *iflag, float *prpval, int *icnt);
void dryout_(float *htout, float *time);
void fuel_temp_(const float *delta, float *q_NFRAC, float *q_Nmax, float *r_f, float
        *f, float *rho_1, float *rho_2, float *h_ib, float *h_p, float *tau_g, float
        *tau_c, float *t_b, float *p_dry, float *h_dry, float *time, short int
        *init_flag, int *MW_flag, int *n, float *h_b, float *q_N, float *k_c, float
        *k_AV, float *q_MWR, float *f_p, float *t_1, float *t_2, float *t_CL, float
        *t_S, float *h_c, float *h_g, float *c_1, float *c_2, float *c_CL, float
        *c_p1, float *c_p2, float *c_p3, float *deltaHT_1, float *delta_ox, float
        *rate_ox, float *q_out, float *q_MWRI)
{
   if (*init_flag) { ⟨ initialization section 18 ⟩}
   else { ⟨ dynamic section 52 ⟩}
}
```

This code is used in chunk 93

4.1 Initialization section

In this section, we determine initial values (subscript $k = 0$) for:

$h_b, q'_N, k_c, k_{AV}, q'_{MWR}, f_p, T_1, T_2, T_{CL}, T_S, h_c, h_g, C_1, C_2, C_{CL}, c_{p,1}, c_{p,2}, c_{p,3}, \Delta H(T_1), \delta_{ox}, R_{ox}, q_{out}, q'_{MWRI}$

18 ⟨ initialization section 18 ⟩ ≡

```
 *n = 1;
     /* n is used to keep track of the dryout output message in the dynamic section */
 float pi = 3.1416;
  ⟨ Calculation of q'_N 20 ⟩;
  ⟨ initialization of clad temperature T_2,0 21 ⟩;
  ⟨ Calculation of k_c 22 ⟩;

            .
            .
            .

 ⟨ Calculation of δ_ox 47 ⟩;
 ⟨ Calculation of q_out 49 ⟩;
 ⟨ Initialization of f_p 50 ⟩;
```

This code is used in chunk 5

4.2 Dynamic section

In this section, we determine transient values (subscript $k > 0$) for

$q'_N, k_c, k_{AV}, q'_{MWR}, f_p, T_1, T_2, T_{CL}, T_S, h_c, h_g, C_1, C_2, C_{CL}, c_{p,1}, c_{p,2}, c_{p,3}, \Delta H(T_1), \delta_{ox}, R_{ox}, q_{out}, q'_{MWRI}$

52 ⟨ dynamic section 52 ⟩ ≡

```
float pi = 3.1416;
int icnt = 10;     /* icnt is given as an argument to the calpro() function for calculating
     the specific heats and the integrals of polynomials */
 ⟨ Check for dryout 54 ⟩;
 ⟨ Computing q'_N,k+1 56 ⟩;
 ⟨ Computing k_c,k+1 57 ⟩;

            .
            .
            .

if (*MW_flag ≡ 1) {
   ⟨ Computing q_out,k+1 83 ⟩;
   ⟨ Computing R_ox,k+1 85 ⟩;
   ⟨ Computing q'_MWR,k+1 87 ⟩;
   ⟨ Computing δ_ox,k+1 89 ⟩;
   ⟨ Computing q'_MWRI,k+1 91 ⟩;
}
```

This code is used in chunk 5

Figure 4.4: Excerpt from LPM showing the overall function and the implementation of initialization and dynamic sections of *fuel_temp_*

4.2.3 Naming Convention of Variables for *fuel_temp_*

This section summarizes all the relevant parameters of the function. Excerpts from the LPM are used to show the naming conventions of the input, output and local variables. That is, the excerpts show how the scientific notation of the parameter is changed into the programming notation. In the excerpts from the LPM, the items under the parameter column give the variables used in the computer code while the items under store column give the mathematical notation of their respective variables (See Figure 4.5). Some of the material has been removed from the excerpts to keep their size small enough to fit in one page.The locations where this has occurred are indicated by vertical ellipses $(\vdots)$. The full list of naming conventions is given in the Appendix B.

Some of the variables that are passed as arguments to the *fuel_temp_* function are input variables while the others are in-out variables. That is, the former case variables are pure inputs while in the later case the variables act as both inputs, as well as outputs. So, the in-out variables store one value when taken as input and a different value when given as output at the same time step. This makes the verification of the implementation with the design difficult. The reader might find it difficult to understand, which value a variable possesses at a particular time step. So, to deal with this problem, the naming convention of the parameters has been divided into three lists as follows:

- Input parameters
- Output parameters
- Local variables

In the input parameters list, first the naming conventions of pure inputs are stated and then the naming conventions of the in-out variables, when acting as inputs are given. The value stored by the input variable during the dynamic section, at time step k is also given. The value for initialization section is not mentioned, because the input variables can have any value as they are not used in the calculations during initialization. Figure 4.5 is an excerpt from LPM showing the naming convention of input parameters.

The output parameters list is again subdivided as:

- output parameters from the initialization section
- output parameters from the dynamic section

This is done because the output variable x stores the initial value x_k at time step $k = 0$ while it stores the value x_{k+1} at time step $k > 0$. That is, when coming from the initialization section, the output variable stores value at the current time step, while it stores value of the next time step when coming out of the dynamic section. So, to make the concept of explicit methods clear, the output list has been subdivided. Figure 4.6 is an excerpt from LPM showing the naming convention of output parameters.

The local variables, which are used for the calculation of effective thermal resistances, are given as a separate list. The list is divided on the basis of the sections in the same way as done for output parameters and for the same reasons as specified for output parameters. Figure 4.7 is an excerpt from LPM showing the naming convention of local variables.

Input Parameters

parameter	stores
$*delta$	Δt
$*q\_NFRAC$	q_{NFRAC}
$*r\_f$	r_f
$*f$	f
$*rho\_1$	ρ_1
.	
.	
.	
$*q\_N$	$q'_{N,k}$, $k \geq 0$, if $\neg *init\_flag$
$*k\_c$	$k_{c,k}$, $k \geq 0$, if $\neg *init\_flag$
$*q\_MWR$	$q'_{\text{MWR},k}$, $k \geq 0$, if $\neg *init\_flag$
$*f\_p$	$P_{\text{F,SUM},k}$ $k \geq 0$, if $\neg *init\_flag$
$*t\_1$	$T_{1,k}$, $k \geq 0$, if $\neg *init\_flag$
$*t\_2$	$T_{2,k}$, $k \geq 0$, if $\neg *init\_flag$
.	
.	
.	

Figure 4.5: Excerpt from LPM showing the naming convention of input parameters

Output Parameters from the Initialization section

if $*init\_flag == 1$,

parameter	stores
$*n$	1
$*h\_b$	h_{ib}
$*q\_N$	$q'_{N,0}$
$*k\_c$	$k_{c,0}$
$*q\_MWR$	$q'_{\text{MWR},0}$
$*f\_p$	$P_{\text{F,SUM},0}$
$*t\_1$	$T_{1,0}$
.	
.	
.	

Output Parameters from the Dynamic section

If $\neg *init\_flag$,

parameter	stores
$*n$	1 or 2
$*h\_b$	h_{ib} or h_{dry}
$*q\_N$	$q'_{N,k+1}$
$*k\_c$	$k_{c,k+1}$
$*q\_MWR$	$q'_{\text{MWR},k+1}$
$*f\_p$	$P_{\text{F,SUM},k+1}$
$*t\_1$	$T_{1,k+1}$
.	
.	
.	

Figure 4.6: Excerpt from LPM showing the naming convention of output parameters

Local Variables for the Effective thermal resistance in the Initialization section

parameter	stores
r_1	$R_{1,0}$
r_2	$R_{2,0}$
r_3	$R_{3,0}$
r_fuel	$R_{\text{FUEL},0}$

Local Variables for the Effective thermal resistance in the Dynamic section

parameter	stores
r_1	$R_{1,k+1}$
r_2	$R_{2,k+1}$
r_3	$R_{3,k+1}$
r_CL	$R_{\text{CL},k+1}$

Figure 4.7: Excerpt from LPM showing the naming convention of local variables

4.3 Evaluation of LP

Even though LP is a programming technique, its main idea is to make the design and logic behind the code understandable to the human reader. Documenting the design and program flow using LP achieves most of the qualities of good documentation (mentioned in Section 2.2) and at the same time meets the expectations of the N286.7 standard for the design and programmer's manuals. In this section, we discuss the advantages of LP, by showing the steps we have taken to achieve the desired qualities of a good document as follows:

1. *Completeness:* While developing the LP source file, the main algorithm of the program is divided into smaller parts which contain explanation, definitions and implementation. As all the theory and numerical algorithms necessary for the implementation are presented before the coding is done, the quality of completeness can be achieved.

2. *Correctness:* As LP is developed in connection with the SRS, checking the correctness becomes easier. That is, before implementing each term, the definition of the term is taken from the SRS and given again in the LP document. This way of implementing each term as a chunk in connection with SRS aids in checking whether the program is implementing the right requirement. Confidence in correctness is built by verifying that every line of code either traces back to a description of the numerical algorithm (in the LPM) (or) to a data definition (or) to an instance model (or) to an assumption (or) to a value from the auxiliary constants table in the SRS.

3. *Consistency:* The following steps were taken to ensure consistency:

   - The naming conventions of the variables have been given during the design of the algorithm. As the code was developed following the naming conventions, the probability of inconsistencies has been reduced.
   - Each term was developed as an individual chunk only once and has been reused wherever necessary. Hence the definition and implementation of the same term is consistent throughout the document.

4. *Traceability:* For achieving the quality of traceability, the following two cases must be satisfied:

   - Traceability between the components of LP.
   - Traceability between the SRS and LP.

   For traceability between the components of LP, we have taken the following measures:

   - Each equation, definition, table has been labelled and referenced, wherever necessary in the document.
   - In LP, the program is developed as a web of interconnected chunks. LP automatically assigns a number to each chunk. When a chunk is used somewhere in the development of an algorithm or in the implementation of an instance model, LP automatically generates a hyperlink to the place where the chunk is being used. Also, at the place where the chunk is being called, LP mentions the number of the chunk that gives the code of the definition. So, the reader will know where the chunk is being used and also where it has come from during the implementation. This makes the traceability between the chunks very easy.

For traceability between LP and SRS, cross referencing was done between the two documents to achieve traceability. That is, while implementing each term, it was referred back to the data definitions section of SRS from where we got the definition of the term. This way of development of LP in connection with SRS, made the traceability between them possible.

As the numerical algorithm is designed to solve the instance models of the SRS, and as the code of the function is developed from the numerical algorithm developed in LPM and the data definitions from the SRS, the traceability between the theory, design and implementation can be achieved.

5. *Verifiability:* The quality of verifiability has been achieved due to the following reasons:

    - As all the necessary information behind the implementation like development of numerical algorithms, solution techniques, assumptions and the program flow is given, the verification of the implementation against the design has been made easier.
    - As traceability between the SRS and LP has been maintained, compliance of the design and implementation with requirements can be checked.
    - As documentation of the design and code is made together in the same document, it is sufficient for the verifier to have the SRS and LP to confirm the correctness of the software.

6. *Unambiguity:* As each term or model is developed only once as a chunk and reused where ever necessary, there is no chance of having two different implementations for the same definition. Also as everything behind the implementation is provided in detail, the chances of having two different interpretations can be reduced. Hence the quality of unambiguity can be achieved.

7. *Modifiability:* As each term is being implemented only once, the task of modification becomes easier. If in future, the implementation has to be changed, only that chunk consisting of the code has to be modified. As repetition is avoided, consistency will not be affected by the modification. Also as traceability and consistency are maintained, the modifiability becomes easier.

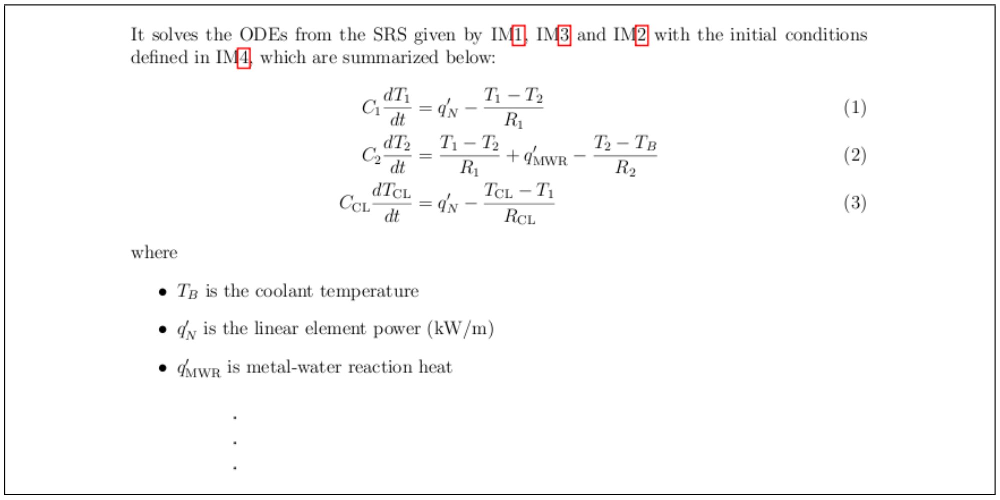
4.4 Design and Implementation of Instance Models

This section shows how the design and implementation have been made together using LP. That is, how numerical algorithms have been developed in connection with SRS, how the logic has been given, how the code was developed as chunks and how the chunks are interconnected. From the excerpts used in the explanation of design and implementation of instance models, the achievement of desired qualities of a good documentation become evident.

Before the design of the instance models from the SRS is done, the mathematical models are summarized by giving their equations, so that they can be referred back while developing the numerical algorithms to solve them. Figure 4.8 shows how the IMs from the SRS were summarized.

4.4.1 Steady State Computation

After giving an overview of the instance models, first the design of the numerical algorithms for solving each instance model in the steady state is done. While developing the numerical


It solves the ODEs from the SRS given by IM1, IM3 and IM2 with the initial conditions defined in IM4, which are summarized below:

$$C_1 \frac{dT_1}{dt} = q'_N - \frac{T_1 - T_2}{R_1} \tag{1}$$

$$C_2 \frac{dT_2}{dt} = \frac{T_1 - T_2}{R_1} + q'_{\text{MWR}} - \frac{T_2 - T_B}{R_2} \tag{2}$$

$$C_{\text{CL}} \frac{dT_{\text{CL}}}{dt} = q'_N - \frac{T_{\text{CL}} - T_1}{R_{\text{CL}}} \tag{3}$$

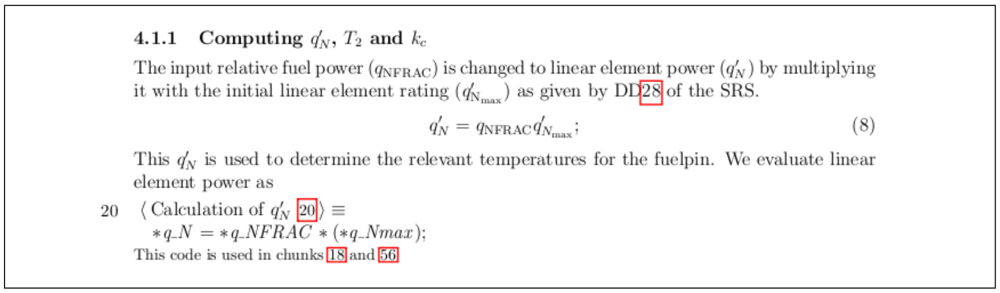
where

- T_B is the coolant temperature
- q'_N is the linear element power (kW/m)
- q'_{MWR} is metal-water reaction heat

.
.
.

Figure 4.8: Excerpt from LPM showing the summary of instance models from the SRS

algorithms, the logic and concepts behind the development are given in an order that is best for human understanding. When a concept is introduced, the data definition equation used in it is given with the chunk of implementation, to make verification easier. When the equations of the data definitions and the code implementing them are presented together, the lines of code can be compared to the definition and the correctness of the implementation can be easily checked. An example, showing how the verification of the code with the design is achieved, is given by Figure 4.9.


4.1.1 Computing q'_N, T_2 and k_c

The input relative fuel power (q_{NFRAC}) is changed to linear element power (q'_N) by multiplying it with the initial linear element rating ($q'_{N_{\max}}$) as given by DD28 of the SRS.

$$q'_N = q_{\text{NFRAC}} q'_{N_{\max}}; \tag{8}$$

This q'_N is used to determine the relevant temperatures for the fuelpin. We evaluate linear element power as

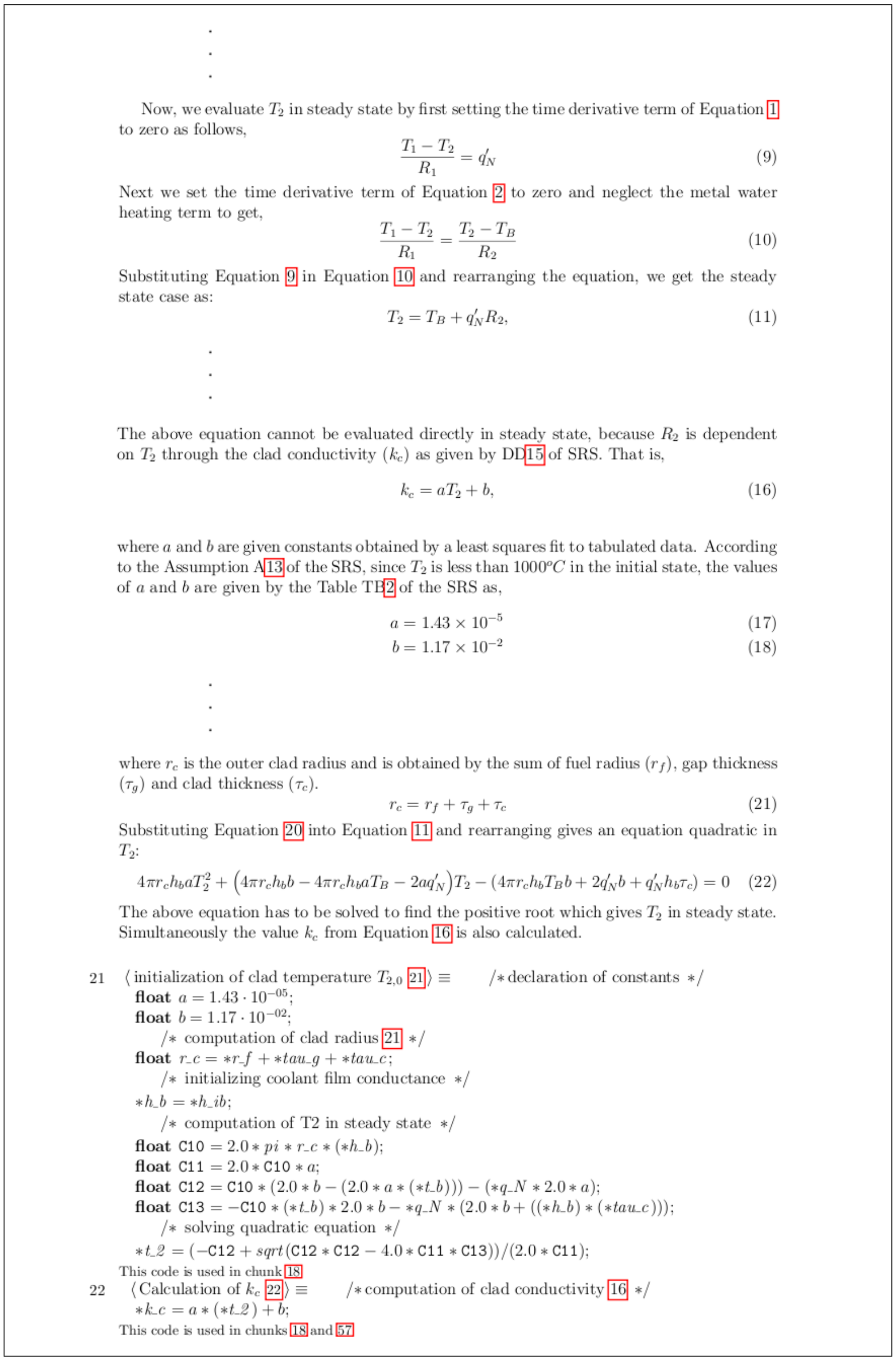
20 ⟨ Calculation of q'_N 20 ⟩ ≡
```
*q_N = *q_NFRAC * (*q_Nmax);
```
This code is used in chunks 18 and 56

Figure 4.9: Excerpt from LPM showing the implementation of q'_{NFRAC} in the steady state

While introducing the concepts, references are made to the relevant data definitions, assumptions and auxiliary constants defined in the SRS. Thus the traceability between the concepts being introduced during design and the SRS can be achieved. For instance, while finding the initial value of T_2, we need to calculate R_2, which introduces the concept of clad conductivity (k_c), as the determination of R_2 depends on k_c. So, when concept of k_c is introduced, references are made to the SRS where the definition of k_c and the values of its constants are given. Figure 4.10 is an excerpt from the LPM, where the traceability between the SRS and the definition of k_c is shown.

By introducing the concepts in connection with SRS, and making the necessary assumptions, the numerical algorithms necessary to find the initial values of the instance models are developed. Once the algorithm is developed, it is immediately implemented as a chunk. Figure 4.10 is an excerpt from the LPM, showing the development of the numerical algorithm for solving T_2 in steady state. The full design is not given in the excerpt to keep its size small enough to fit on a page. However, the detailed development of the algorithm can be found in Appendix B. The excerpt shows how the algorithm is developed in connection with the SRS and how the code is developed in accordance with the design, making verification easier. Also, the excerpt shows how the numbering of the chunks is done and how LP mentions the place where the chunk can be used, achieving traceability.

Once a chunk is developed, it is called from the main function. In this way, each chunk is implemented and the initialization section is built by calling all the chunks that are developed, to find the initial values as shown in the Figure 4.4.



.
.
.

Now, we evaluate T_2 in steady state by first setting the time derivative term of Equation 1 to zero as follows,

$$\frac{T_1 - T_2}{R_1} = q'_N \tag{9}$$

Next we set the time derivative term of Equation 2 to zero and neglect the metal water heating term to get,

$$\frac{T_1 - T_2}{R_1} = \frac{T_2 - T_B}{R_2} \tag{10}$$

Substituting Equation 9 in Equation 10 and rearranging the equation, we get the steady state case as:

$$T_2 = T_B + q'_N R_2, \tag{11}$$

.
.
.

The above equation cannot be evaluated directly in steady state, because R_2 is dependent on T_2 through the clad conductivity (k_c) as given by DD15 of SRS. That is,

$$k_c = aT_2 + b, \tag{16}$$

where a and b are given constants obtained by a least squares fit to tabulated data. According to the Assumption A13 of the SRS, since T_2 is less than 1000^oC in the initial state, the values of a and b are given by the Table TB2 of the SRS as,

$$a = 1.43 \times 10^{-5} \tag{17}$$
$$b = 1.17 \times 10^{-2} \tag{18}$$

.
.
.

where r_c is the outer clad radius and is obtained by the sum of fuel radius (r_f), gap thickness (τ_g) and clad thickness (τ_c).

$$r_c = r_f + \tau_g + \tau_c \tag{21}$$

Substituting Equation 20 into Equation 11 and rearranging gives an equation quadratic in T_2:

$$4\pi r_c h_b a T_2^2 + \left(4\pi r_c h_b b - 4\pi r_c h_b a T_B - 2aq'_N\right)T_2 - (4\pi r_c h_b T_B b + 2q'_N b + q'_N h_b \tau_c) = 0 \tag{22}$$

The above equation has to be solved to find the positive root which gives T_2 in steady state. Simultaneously the value k_c from Equation 16 is also calculated.

```
21  ⟨initialization of clad temperature T_{2,0} 21⟩ ≡      /* declaration of constants */
      float a = 1.43 · 10^{-05};
      float b = 1.17 · 10^{-02};
          /* computation of clad radius 21 */
      float r_c = *r_f + *tau_g + *tau_c;
          /* initializing coolant film conductance */
      *h_b = *h_ib;
          /* computation of T2 in steady state */
      float C10 = 2.0 * pi * r_c * (*h_b);
      float C11 = 2.0 * C10 * a;
      float C12 = C10 * (2.0 * b − (2.0 * a * (*t_b))) − (*q_N * 2.0 * a);
      float C13 = −C10 * (*t_b) * 2.0 * b − *q_N * (2.0 * b + ((*h_b) * (*tau_c)));
          /* solving quadratic equation */
      *t_2 = (−C12 + sqrt(C12 * C12 − 4.0 * C11 * C13))/(2.0 * C11);
    This code is used in chunk 18
22  ⟨Calculation of k_c 22⟩ ≡      /* computation of clad conductivity 16 */
      *k_c = a * (*t_2) + b;
    This code is used in chunks 18 and 57
```

Figure 4.10: Excerpt from LPM showing the design and implementation of T_2 in the steady state

4.4.2 Transient State Computation

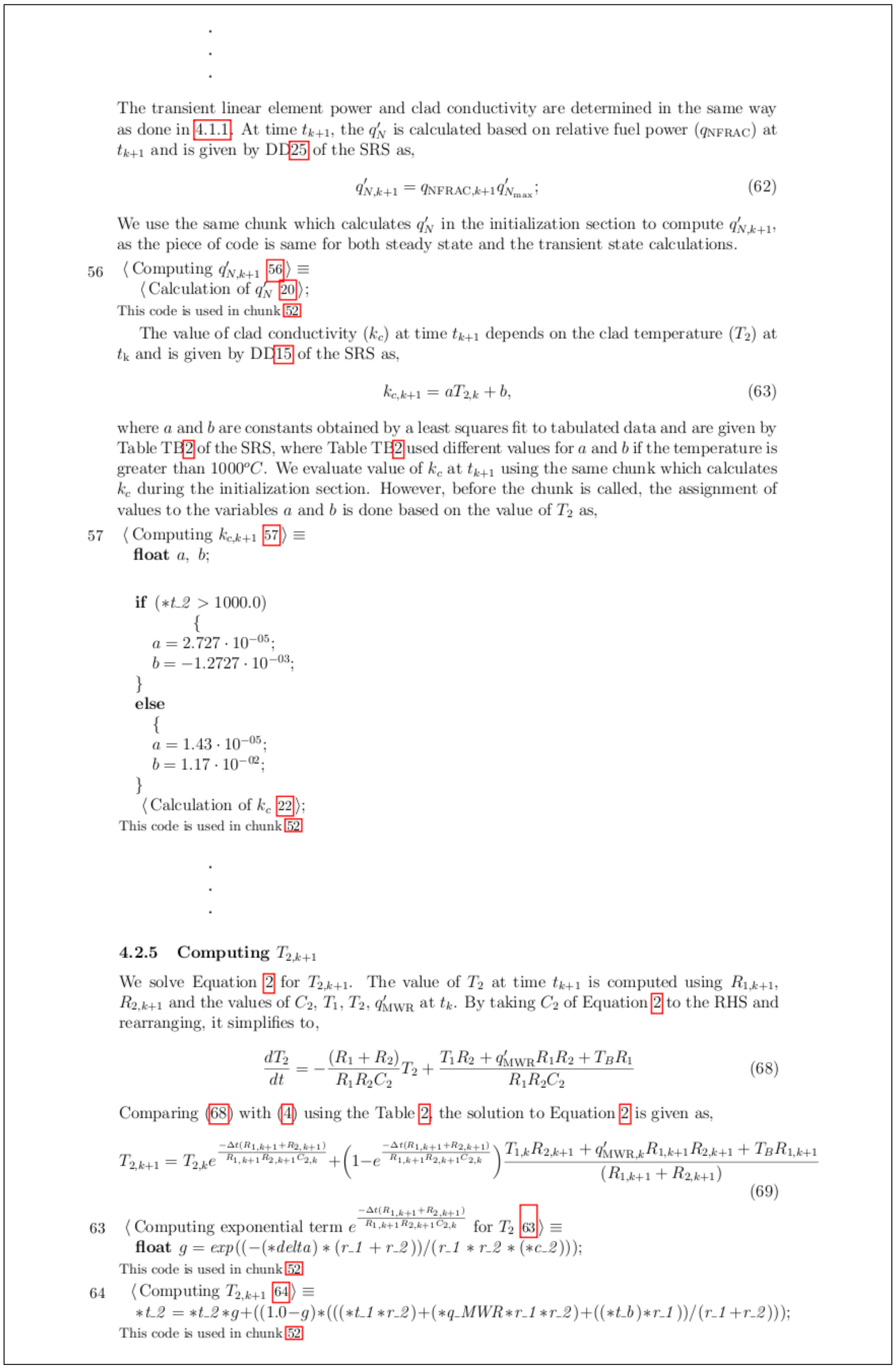
After the initial values are computed in the initialization section, the next step is to design and implement the dynamic section. That is, we determine transient values (subscript $k > 0$) for the instance models and the data definitions. In this section we do not develop any numerical algorithms, as we have already discussed the algorithm that has to be used for finding the transient values in Section 4.2.1. Figure 4.2 gives the numerical algorithm designed for solving the instance models of the SRS. So, in the dynamic calculations section, the flow of logic is given and the ODEs are solved using the developed numerical algorithm.

When the data definitions are to be implemented, their definitions are summarized with the help of subscript k. This is done with the intention of clarifying which time step value is being used by the variable in the definition. While the definition is given, references are made to the SRS in the same way as done in the initialization section, for achieving traceability. Then, the data is implemented under the definition to make the verifiability easier. If the definition of the data is the same for both steady state as well as transient state calculations, then the chunk that has been developed in the initialization section is reused in the dynamic section to maintain consistency and achieve unambiguity. As mentioned in the Section 4.3, each data definition is implemented only once to avoid any confusions and to make the tasks of maintainability and modifiability easier.

Figure 4.11 is an excerpt from LPM showing the design and implementation of T_2 in transient state. From the figure it can be seen, how the subscript k was used in the data definitions to indicate the time step value of the variables and how the chunks from the initialization section were reused. For the calculation of q'_N in the dynamic state, the chunk which calculated q'_N in the initialization section was reused as both the states uses the same piece of code (same data definition).

For implementing the instance models, first each instance model is rearranged to match the differential equation form given in the numerical algorithm of Section 4.2.1 (See Figure 4.2). Then the transient solution is given by comparing the modified instance model equation and the ODE being solved by the numerical algorithm. The implementation of the algorithm is done as chunks immediately. In this way, each data definition and instance model is implemented and the dynamic section is built by calling all the chunks that are developed, to find the transient values as shown in the Figure 4.4.

Figure 4.11 shows the development of the transient solution for T_2 by rearranging and comparing it with the ODE of the numerical algorithm discussed in Section 4.2.1. Also it can be seen, how the implementation of chunks was done achieving verifiability, correctness and completeness.



.
.
.

The transient linear element power and clad conductivity are determined in the same way as done in 4.1.1. At time t_{k+1}, the q'_N is calculated based on relative fuel power (q_{NFRAC}) at t_{k+1} and is given by DD25 of the SRS as,

$$q'_{N,k+1} = q_{\text{NFRAC},k+1} q'_{N_{\max}}; \tag{62}$$

We use the same chunk which calculates q'_N in the initialization section to compute $q'_{N,k+1}$, as the piece of code is same for both steady state and the transient state calculations.

56 $\langle$Computing $q'_{N,k+1}$ 56$\rangle \equiv$
  $\langle$Calculation of q'_N 20$\rangle$;
This code is used in chunk 52

The value of clad conductivity (k_c) at time t_{k+1} depends on the clad temperature (T_2) at t_k and is given by DD15 of the SRS as,

$$k_{c,k+1} = aT_{2,k} + b, \tag{63}$$

where a and b are constants obtained by a least squares fit to tabulated data and are given by Table TB2 of the SRS, where Table TB2 used different values for a and b if the temperature is greater than 1000^oC. We evaluate value of k_c at t_{k+1} using the same chunk which calculates k_c during the initialization section. However, before the chunk is called, the assignment of values to the variables a and b is done based on the value of T_2 as,

57 $\langle$Computing $k_{c,k+1}$ 57$\rangle \equiv$

```
float a, b;

if (*t_2 > 1000.0)
        {
    a = 2.727 · 10^-05;
    b = -1.2727 · 10^-03;
}
else
    {
    a = 1.43 · 10^-05;
    b = 1.17 · 10^-02;
}
 ⟨Calculation of k_c 22⟩;
```

This code is used in chunk 52

.
.
.

4.2.5 Computing $T_{2,k+1}$

We solve Equation 2 for $T_{2,k+1}$. The value of T_2 at time t_{k+1} is computed using $R_{1,k+1}$, $R_{2,k+1}$ and the values of C_2, T_1, T_2, q'_{MWR} at t_k. By taking C_2 of Equation 2 to the RHS and rearranging, it simplifies to,

$$\frac{dT_2}{dt} = -\frac{(R_1 + R_2)}{R_1 R_2 C_2} T_2 + \frac{T_1 R_2 + q'_{\text{MWR}} R_1 R_2 + T_B R_1}{R_1 R_2 C_2} \tag{68}$$

Comparing (68) with (4) using the Table 2, the solution to Equation 2 is given as,

$$T_{2,k+1} = T_{2,k} e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}} + \left(1 - e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}}\right) \frac{T_{1,k}R_{2,k+1} + q'_{\text{MWR},k} R_{1,k+1} R_{2,k+1} + T_B R_{1,k+1}}{(R_{1,k+1} + R_{2,k+1})} \tag{69}$$

63 $\langle$Computing exponential term $e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}}$ for T_2 63$\rangle \equiv$

```
float g = exp((-(*delta) * (r_1 + r_2))/(r_1 * r_2 * (*c_2)));
```

This code is used in chunk 52

64 $\langle$Computing $T_{2,k+1}$ 64$\rangle \equiv$

```
*t_2 = *t_2*g+((1.0-g)*(((*t_1*r_2)+(*q_MWR*r_1*r_2)+((*t_b)*r_1))/(r_1+r_2)));
```

This code is used in chunk 52

Figure 4.11: Excerpt from LPM showing the design and implementation of T_2 in transient state

Chapter 5

Conclusions

With the general engineering practices being followed for the development of scientific software, the documentation of requirements and design are not being given the importance they deserve. The introduction of software engineering methodologies into the scientific field will reduce this problem. The goal of this research is to make the scientists understand the gravity of the problems that arise for software that is poorly documented. By the introduction of the new template for the SRS, we hope that the people from the scientific field will learn how to document all the requirements and avoid problems of inconsistency and incompleteness. Also, by introducing Literate Programming for documenting the design and implementation, we wish that the developers will find it easy to maintain the traceability between the theory, design and code, thus making the verification part easier during the certification. The summary of this thesis is provided in Section 5.1 and future work is given in the Section 5.2.

5.1 Summary

In this section, we are going to summarize how the research problems stated in Chapter 1 are solved through this research by considering each problem in turn.

1. *How to improve the requirements documentation capturing all the necessary information for developing scientific software?*

   This thesis proposes a template for the SRS, which helps in systematically developing the requirements document. The template helps in achieving completeness, as sections of it act as a checklist to the developer and force him to fill in the necessary information. As the template is developed following the principle of separation of concerns, each and every section can be dealt with individually and the document can be developed in detail by refining from goals to mathematical instanced models. This way, the proposed template provides guidelines for documenting the requirements by suggesting an order for filling in the details, thus reducing chances of missing information.

2. *How to achieve the desirable qualities of a good requirements document for scientific software?*

   The proposed template helps to divide the problem into smaller parts and develop each part completely, correctly and consistently with other parts by maintaining traceability and unambiguity. How the desired qualities are achieved by the template are summarized below:

- Documenting all the background information, physical laws, assumptions, constitutive equations, rules, principles, physical data, constraints and constant values in different sections like Theoretical models, General Definitions, Data Definitions, Assumptions, Data Constraints, and Auxiliary constants, aids in achieving completeness.
- The sections for Assumptions, Theoretical models, General Definitions and Data Definitions, aid in developing the functional requirements systematically in a hierarchical manner. This way of development from abstract to concrete helps in achieving correctness, as the application of assumptions to the right governing equations results in correct definitions, and the use of these correct definitions with correct assumptions will result in deriving the correct instanced models.
- The qualities of traceability and modifiability have been achieved by giving a unique label to each component and using cross referencing between the components of the SRS. The hierarchical development of the models also helps in achieving traceability and modifiability, because several concrete models can be developed from the same abstract model by applying different assumptions and data definitions. So, when a concrete model has to be changed, only the assumptions and data definitions part will be affected, while the rest of the development remains the same making the modifications easier.
- By encouraging the hierarchical development of models in the SRS, the software qualities like maintainability and reusability can be achieved, thus preventing the cost of recertification from becoming too large.
- The Table of Symbols section is used to achieve consistency, as it summarizes all the symbols used in the document along with their units. By making clear which symbol represents which term right at the beginning and by following the naming convention, the problem of inconsistency can be tackled.
- By including the derivations of the equations under their definitions, completeness as well as correctness can be improved.
- The SRS can achieve the quality of being abstract by not discussing the numerical algorithms and other design decisions within it.
- By defining each term only once in the data definition section and by developing the functional requirements individually, without mixing with others in the instance models section, the quality of unambiguity can be achieved. Also in the future, if a requirement has to be changed, then the change can be made in only one place, thus helping the maintainers in putting less effort and time towards reflecting that change everywhere in the software.
- As the requirements are specified completely, correctly with traceability to their components, the task of verification can be facilitated.

3. *How to develop the code with traceability to the numerical algorithm choices?*

   LP develops the code and design in the same document while maintaining traceability between them. Also, as the lines of code in the chunks are developed making references to the numerical algorithms and other definitions used in the design, traceability between the code and design can be achieved.

4. *How to achieve traceability between the theory, numerical algorithms and the implementation?*

   As the LP Manual develops the design and the code together, it is easier to maintain traceability between the numerical algorithms and the implementation. While using a term (or) a constant (or) a model in the design, it is referred back to the data definitions section, auxiliary constants section, instance models section of the SRS respectively from the LP Manual. This way of developing the LP Manual in connection with the SRS by making cross referencing between the two documents helps in achieving traceability between the theory, numerical algorithms and the implementation.

5. *How to facilitate certification of scientific software via documentation techniques?*

   As the expected qualities of software (according to N286.7 standard) to be certified are similar to the desirable qualities for a good SRS, the certification of the software can become easier by developing the requirements, design and implementation documents achieving the desirable qualities. As the SRS and the LP Manual are developed in such a way that they satisfy the conditions of a good documentation, the tasks of certification like verification, maintenance, and checking for modifiability became easier, thus facilitating the certification process.

5.2 Future work

This section suggests the following work to encourage further research in the documentation of scientific software and to enhance the effectiveness of our research:

1. Documentation of Non-functional Requirements:
   The Non-functional Requirements mentioned in Section 4.c and Other System Issues mentioned in Section 5 of Figure 3.1 are not documented in the Appendix A, as the original theory manual did not contain information on them. So, these requirements should be elicited and documented in the SRS to enhance it and make it complete.

2. Development of tools:

   - In the LP Manual, while implementing the data definitions from the SRS, the definitions of the data were again given in the LP Manual to make verification easier. Currently, redefining the definition was done manually by copying and pasting between the documents, and only backward references were given from the LP Manual to the SRS. Tools can be developed for transclusion between the SRS and the LP Manual. That is, for automatic generation of information in the LP Manual that is to be reused from the SRS. The tool should be able to facilitate bidirectional traceability by generating hyperlinks in the SRS and the LP Manual at the places where the information is being shared.
   - Tools can be developed to have multiple views of documentation, with each showing the level of detail required. For instance, the reader might not want to look at the derivations while verifying the implementation against the requirements. In this case the tool should be able to temporarily hide these details.
   - Tools can be developed to automatically generate graphical views of interconnections between different sections (Goals, Theoretical models, Data Definitions etc) of the SRS. Currently, we have manually produced the traceability matrix for analyzing the interconnections between different sections of the

SRS. This work can be done by a developing a tool which scans the document to identify the labels and references and automatically produces the matrix and its graphical visualization.

3. Redesign of code:
The original FORTRAN code is not properly modularized. That is, the principle of information hiding was not applied during its development. For example, the numerical algorithm used for solving the ODEs is not developed as a seperate subroutine. This leads to modifiability problems, as the entire subroutine performing the calculations has to be modified, if the numerical algorithm needs to be changed. Instead, if the independent changeable system details are developed as separate modules, the maintenance and modifiability become easier, as we have to change only the module we are concerned with. Hence, it is a good idea to redesign the entire FORTRAN code in a way that it applies the information hiding principle, keeping the implementation details of each module as a secret from the rest of the program.

4. Implementation of all the subroutines using LP:
As we have constricted our scope to developing only the FUELTE subroutine of the original FORTRAN code, the future work can be designing and implementing the rest of the subroutines of FP using LP.

5. Explore the use of the SRS and the LP Manual for vendor supplied software:
In many cases, nuclear power generating companies uses software developed by third parties, rather than in-house developed software. The methodologies developed in this thesis should be examined to determine how they can be adapted to this context.

6. Testing:
This thesis does not deal with testing, as the emphasis was on verification of software by human experts. Yet, as testing is an important part of the certification process, test cases should be built for sensitivity, uncertainity and correlation analysis of the software.

Bibliography

[1] A. P. Moore and C. N. Payne, Jr. Increasing assurance with literate programming techniques. pages 187–198, 1996.

[2] Andrew L. Johnson and Brad C. Johnson. Literate Programming Using `Noweb`. *j-LINUX-JOURNAL*, 42:64–69, October 1997.

[3] CSA. Quality assurance of analytical, scientific, and design computer programs for nuclear power plants. Technical Report N286.7-99, Canadian Standards Association, March 1999.

[4] Alan M. Davis. *Software Requirements: Objects, Functions, and States*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1993.

[5] Donald E. Knuth. The web system of structured documentation. Stanford Computer Science Report STAN-CS-83-980, Stanford University, Stanford, CA, September 1983.

[6] Donald E. Knuth. *Literate Programming*, volume 1 of *CSLI Lecture Notes Number 27*. pub-SUCSLI, pub-SUCSLI:adr, 1992.

[7] ESA. ESA software engineering standards, PSS-05-0 issue 2. Technical report, European Space Agency, February 1991.

[8] FPManual. *FP Manual*. Nuclear Studies and Safety Department, March 1982.

[9] John Hatcliff, Mats Heimdahl, Mark Lawford, Tom Maibaum, Alan Wassyng, and Fred Wurden. A software certification consortium and its top 9 hurdles. *Electronic Notes in Theoretical Computer Science*, 238(4):12, 2009.

[10] Kathryn L. Heninger. Specifying software requirement for complex system: New techniques and their application. *IEEE Transactions on Software Engineering*, 6(1):2–13, January 1980.

[11] IEEE. *Recommended practice for Software Requirements Specifications*. IEEE, June 1998.

[12] Donald E. Knuth and Silvio Levy. *The CWEB System of Structured Documentation, Version 3.0*. pub-AW, pub-AW:adr, 1993.

[13] Lei Lai. Requirements documentation for engineering mechanics software: Guidelines, template and a case requirements documentation for engineering mechanics software: Guidelines, template and a case studycase study. Master's thesis, McMaster University, Hamilton, Ontario, Canada, September 2004.

[14] T. Maibaum and A. Wassyng. A product-focused approach to software certification, 2008.

[15] NASA. Software requirements DID, SMAP-DID-P200-SW, release 4.3. Technical report, National Aeronautics and Space Agency, 1989.

[16] Nedialko S. Nedialkov. VNODE-LP — A Validated Solver for Initial Value Problems in Ordinary Differential Equations. Technical Report CAS-06-06-NN, Department of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, Ontario, L8S 4K1, 2006.

[17] Nedialko S. Nedialkov. Implementing a Rigorous ODE Solver through Literate Programming. Technical Report CAS-10-02-NN, Department of Computing and Software, McMaster University, 2010.

[18] M. Ortuno, A. Marquez, S.Gallego, C.Neipp, and A. Belndez. An experiment in heat conduction using hollow cylinders. *European Journal of Physics*, 32:3–6, 2011.

[19] D. L. Parnas, R. Janicki, and J. Zucker. Tabular representation in relational documents. Technical Report CRL 313, McMaster University, February 1996.

[20] David L. Parnas and P.C. Clements. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, 12(2):251–257, February 1986.

[21] Patrick J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico, 1998.

[22] Gonzalo Sanchez, Spencer Smith, and Ned Nedialkov. Certification of Scientific Computing Software. Technical Report 2, McSCert, July 2011.

[23] B. Sanga. Assessing and improving the quality of software requirements specification documents (SRSDs). Thesis for M.Sc. program, Computing and Software Department, McMaster University, Hamilton, ON, August 2003.

[24] G. S Sawhney. *Heat and Mass Transfer*. LK International Publishing House Pvt. Ltd, S-25, Green Park Extension, Uphaar Cinema Market, New Delhi- 110 016(India), second edition, 2010.

[25] Joachim Schrod. CTAN, the Comprehensive TEX Archieve Network, 2013.

[26] Judith Segal and Chris Morris. Developing scientific software. *IEEE Software*, 25(4), July/August 2008.

[27] W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ågerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, Paris, France, August 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

[28] W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13, 2007.

[29] I. Sommerville and P. Sawyer. *Requirement Engineering: A Good Practice Guide*. John Wiley & Sons Ltd., 1997.

[30] R. H. Thayer and M. Dorfman, editors. *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society, Washington, DC, USA, 2nd edition, 2000.

[31] Wikipedia. Computational science, June 2013.

[32] Wikipedia. Literate programming, May 2013.

Appendix A

Software Requirement Specification for FP

Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are employed as described below. For each unit, the symbol is given followed by a description of the unit with the SI name in parentheses.

m	- for length (metre)
kg	- for mass (kilogram)
s	- for time (second)
K	- for temperature (kelvin)
^oC	- for temperature (centigrade)
J	- for energy (joule, J=$\frac{\text{kgm}^2}{\text{s}^2}$)
cal	- for energy (calorie, cal $\approx 4.2\ \frac{\text{kgm}^2}{\text{s}^2}$)
mol	- for amount of substance (mole)
W	- for power (watt, W=$\frac{\text{kgm}^2}{\text{s}^3}$)

A.1 Reference Material

This section records the information of the SRS in a form that allows easy reference throughout the document.

A.1.1 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made with the goal of being consistent with the nuclear physics literature and that used in the FP manual. The SI units are listed in brackets following the definition of the symbol.

A.1.1.1 Quantities related to Thermal Analysis

C_i	- thermal capacitance terms indexed by i ($\frac{\mathrm{kWs}}{\mathrm{m^oC}}$)
h_b	- coolant film conductance ($\frac{\mathrm{kW}}{\mathrm{m^2C}}$)
h_c	- convective heat transfer coefficient between clad and coolant ($\frac{\mathrm{kW}}{\mathrm{m^2C}}$)
h_{dry}	- convective heat transfer coefficient between fuel surface and coolant at dryout ($\frac{\mathrm{kW}}{\mathrm{m^2C}}$)
h_g	- effective heat transfer coefficient between clad and fuel surface ($\frac{\mathrm{kW}}{\mathrm{m^2C}}$)
h_p	- initial gap film conductance ($\frac{\mathrm{kW}}{\mathrm{m^2C}}$)
k_c	- clad conductivity ($\frac{\mathrm{kW}}{\mathrm{m^oC}}$)
k_{AV}	- average thermal conductivity ($\frac{\mathrm{kW}}{\mathrm{m^oC}}$)
N	- Neutron flux
p_{dry}	- dryout threshold power
q	- heat flux ($\frac{\mathrm{kW}}{\mathrm{m^2}}$)
q'''	- volumetric heat generation ($\frac{\mathrm{kW}}{\mathrm{m^3}}$)
r_c	- clad radius (m)
r_f	- fuel radius (m)
R	- resistance ($\frac{\mathrm{m^oC}}{\mathrm{kW}}$)
R_{FUEL}	- thermal resistance of fuel ($\frac{\mathrm{m^oC}}{\mathrm{kW}}$)
R_{CLAD}	- clad resistance ($\frac{\mathrm{m^oC}}{\mathrm{kW}}$)
R_{GAP}	- gap resistance ($\frac{\mathrm{m^oC}}{\mathrm{kW}}$)
R_{FILM}	- coolant film resistance ($\frac{\mathrm{m^oC}}{\mathrm{kW}}$)
T_{CL}	- centreline temperature (^{o}C)
T_S	- surface temperature (^{o}C)
T_1	- average fuel temperature (^{o}C)
T_2	- average clad temperature (^{o}C)
T_B	- coolant temperature (^{o}C)
t	- time (s)
ρ_1	- fuel density ($\frac{\mathrm{kJ}}{\mathrm{kg^oC}}$)
ρ_2	- clad density ($\frac{\mathrm{kJ}}{\mathrm{kg^oC}}$)
τ_c	- clad thickness (m)
$c_{p,1}$	- specific heat corresponding to fuel average temperature ($\frac{\mathrm{kJ}}{\mathrm{kg^oC}}$)
$c_{p,2}$	- specific heat corresponding to clad average temperature ($\frac{\mathrm{kJ}}{\mathrm{kg^oC}}$)

$c_{p,3}$	- specific heat corresponding to fuel centerline temperature ($\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$)

A.1.1.2 Quantities related to Nuclear Physics

A_k	- value of trip parameter at t_k
K_i	- response fraction
q'_{MWR}	- metal water reaction heat ($\frac{\text{kW}}{\text{m}}$)
q'_{MWRI}	- integrated metal water reaction heat
q'_N	- linear element power ($\frac{\text{kW}}{\text{m}}$)
q'_{NFRAC}	- relative fuel power
$q'_{N_{\text{max}}}$	- linear element power at full power ($\frac{\text{kW}}{\text{m}}$)
q_{in}	- input heat ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
q_{out}	- output heat ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
q'_{out}	- output heat to the coolant ($\frac{\text{kW}}{\text{m}^2\text{C}}$)
W_i	- relative decay heat amplitude for i^{th} group
α_i	- i^{th} delay fraction for incore flux detector signal
β_i	- delayed neutron fraction
γ_i	- decay fraction
λ_i	- delay constant (s^{-1})
τ_A	- amplifier time constant (s)
ψ_i	- i^{th} decay constant for incore flux detector signal (s^{-1})
$\int$	- integration
$\bigtriangledown$	- gradient operator
UO_2	- uranium dioxide
ρ	- reactivity
f	- average flux depression factor
1^*	- prompt generation time (s)
$\Delta H(T_{\text{abs}})$	- fuel stored energy ($\frac{J}{kg}$)

Prefixes

Δ	- finite change in following quantity
d	- infinitesimal change in the following quantity

A.1.2 Abbreviations and Acronyms

SRS	- Software Requirements Specification
TFR	- Temperature Feedback Reactivity
MWR	- Metal Water Reaction
NOP	- Neutron Over Power
IM	- Instance Model
TM	- Theoretical Model
A	- Assumption
PS	- Physical System Description
G	- Goal

A.2 Introduction

This introduction provides an overview of the Software Requirement Specification (SRS) for fuelpin analysis within a nuclear reactor. The requirements are based on an existing theory manual and an existing code developed by a nuclear power generating company, henceforth called FP. This section explains the purpose of this document, the scope of the system, the organization of the document and the characteristics of the intended readers.

A.2.1 Purpose of Document

The main purpose of this document is to assist in the certification process for the FP code. This document provides the requirements that the FP code should implement. In particular, the goals and theoretical models used in the FP code are detailed and refined with an emphasis on explicitly identifying assumptions and unambiguous definitions. The relevant theory for FP that is presented in this SRS includes:

- the lumped parameter fuel modelling technique
- temperature dependent thermodynamic properties
- point neutron kinetics
- decay heat equations
- trip parameter modelling
- metal water reaction model
- fuel stored energy and integrated fuel power calculations.

A.2.2 Scope

The scope of the product is limited to thermal analysis and reactor physics relevant to modelling a single fuelpin. It does not include mechanical analysis or fluid dynamics. The fuelpin is modelled in isolation with no interaction between adjacent fuelpins. Given the appropriate inputs, the code for FP is intended to do the following:

- Predict temperature histories for the reactor fuel and clad.
- Calculate the integrated fuel power and the change in UO_2 enthalpy from room temperature to the average fuel temperature.
- Model the dynamic response of signal amplifiers and trip logic.
- Model dynamically compensated self-powered in-core flux detector signals that form part of the neutron overpower trip systems.

A.2.3 Organization of Document

The organization of this document follows the template for an SRS for Scientific Computing Software proposed by [13] and [27].

A.2.4 Intended Audience

This document will be helpful to Nuclear Safety Analysts to build confidence in the theoretical model and associated code implementing it. This document will also be used as part of the certification process for FP.

A.3 General System Description

General System Description provides general information about the system, identifies the interfaces between the system and its environment, describes the user characteristics and the system constraints.

A.3.1 System Context

1. FP in a larger context of reactor analysis is used for the following:
   - running safety analysis cases.
   - model one pin to give insight into the use of multiple pins.
   - iterative part of design and safety analysis (separation of concerns so that focus can be on one thing at a time).
2. The system takes either fuel power versus time or neutron flux versus time or the reactivity transient as input and predicts the output transient reactor fuel temperature and average clad temperature as follows:
   - If neutron flux versus time is given as input, the transient fuel power is generated from fission power and decay heat components.
   - If the reactivity transient is given as input, the point neutron kinetics model is used to generate the neutron flux transient.
3. The system takes either the neutron flux or the flux detector signal and the shutdown reactivity characteristic, log rate, linear rate and NOP as trip set points to simulate the reactor trip and shutdown.

A.3.2 User Characteristics

The end user of FP should have at least an undergraduate degree in Engineering or Science. Moreover, to understand the theory behind this project, the user should have the equivalent knowledge to an introductory course on each of thermodynamics and reactor physics.

A.3.3 System Constraints

None present.

A.4 Specific System Description

The specific system description includes all of the SRS software requirements in sufficient detail to enable design and testing of a system that will satisfy the requirements [27].

A.4.1 Problem Description

FP is a computer program developed to simulate the reactor trip and shutdown by predicting transient reactor fuel and clad temperatures based on a lumped parameter modelling approach, by incorporating a point neutron kinetics model and flexible transient control logic.

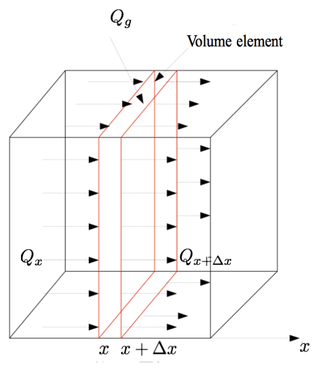


Figure A.1: One dimensional heat conduction through a volume element

A.4.1.1 Background

Many concepts are important for understanding the theory behind FP. As a brief summary, the topics include the following: thermodynamics, point neutron kinetics, flexible transient control logic, trip logic, oxidization, enthalpy, lumped parameter modelling, heat transfer, integration and differentiation, flux depression, linear element power, thermal resistance, capacitance and conductivity, fuelpin configuration, specific heat, neutron flux, delayed neutron and prompt generations and linear interpolation.

Understanding the thermodynamic model of a fuelpin requires a basic understanding of heat transfer. To illustrate the rational behind the thermal model of the fuelpin, the heat transfer in one dimension is shown below. Following this, the analogy between heat and electrical conduction is described as it is helpful in understanding the instance models.

Heat transfer in one dimensional Cartesian coordinate system

The general heat transfer equation in 1D Cartesian coordinates can be obtained from an energy balance on a volume element in Cartesian coordinates [24, page 34–36]. Figure A.1 shows a thin element of thickness Δx on a large plate.
Energy balance on this element during small time interval Δt is given as:

$$\text{(Rate of heat transfer at } x) - \text{(Rate of heat transfer at } x+\Delta x) + \text{(Rate of heat generation inside the element)=(Rate of change of energy content of the element)} \tag{A.1}$$

Using Q for the rate of heat transfer and E for the heat energy content, this equation can be rewritten as:

$$Q_x - Q_{x+\Delta x} + Q_g = \frac{\Delta E}{\Delta t}, \tag{A.2}$$

where Q_x is the rate of heat transfer in the x direction, Q_g is rate of heat generation inside the element and ΔE is the energy content of the element. If the volumetric heat generation in the element is q''' and the area of the plate is A, then the heat generated

$$Q_g = q''' A \Delta x \tag{A.3}$$

The change in energy content of the element in time Δt is $\Delta E = E_{t+\Delta t} - E_t$
The temperature (T) can be introduced through the relation that $E = \rho CVT$, where ρ is the

density, C is the specific heat capacity and V is the volume of the element. In this case, $V = A\Delta x$.

$$\Delta E = \rho C_p A\Delta x(T_{t+\Delta t} - T_t) \tag{A.4}$$

Substituting A.3, A.4 in A.2,

$$Q_x - Q_{x+\Delta x} + q'''A\Delta x = \rho CA\Delta x\frac{T_{t+\Delta t} - T_t}{\Delta t} \tag{A.5}$$

Dividing by $A\Delta x$ gives,

$$-(\frac{Q_{x+\Delta x} - Q_x}{A\Delta x}) + q''' = \rho C\frac{T_{t+\Delta t} - T_t}{\Delta t} \tag{A.6}$$

Taking the limit as $\Delta x \to 0$ and $\Delta t \to 0$ yields

$$-\frac{1}{A}\frac{\partial Q}{\partial x} + q''' = \rho C\frac{\partial T}{\partial t} \tag{A.7}$$

According to Fourier's law,

$$Q = -kA\frac{\partial T}{\partial x} \tag{A.8}$$

substituting A.8 into A.7,

$$-\frac{1}{A}\frac{\partial}{\partial x}(-kA\frac{\partial T}{\partial x}) + q''' = \rho C\frac{\partial T}{\partial t} \tag{A.9}$$

$$\frac{\partial}{\partial x}(k\frac{\partial T}{\partial x}) + q''' = \rho C\frac{\partial T}{\partial t} \tag{A.10}$$

If k is temperature independent, then the above equation simplifies to:

$$k\frac{\partial^2 T}{\partial x^2} + q''' = \rho C\frac{\partial T}{\partial t} \tag{A.11}$$

Analogy between heat conduction and electrical conduction

Figure A.2 shows the flow of current in a circuit. From Ohm's law, we know that voltage (V) is directly proportional to resistance (R) when current (I) is kept constant;
that is,

$$V = IR \tag{A.12}$$

When there are n resistors connected in series with resistances $R_1, R_2, R_3, ...R_n$, the current (I) is same through each resistor. The, voltage drop across all of the resistors is directly proportional to the effective resistance (R_e).

$$V = IR_e, \tag{A.13}$$

where

$$R_e = R_1 + R_2 + R_3..... + R_n \tag{A.14}$$

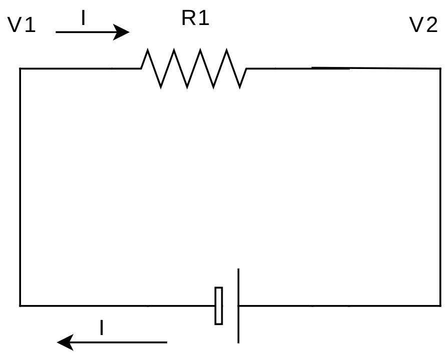


Figure A.2: Electric circuit

Figure A.3 shows the heat flow in a slab. The thermal analogue of Ohm's law is written as,

$$\Delta T = QR_e \tag{A.15}$$

That is, the temperature drop between the surfaces of a slab is directly proportional to the thermal resistance between the surfaces, where Q is the rate of heat conduction. From Fourier's law,

$$Q = -kA\frac{dT}{dx} \tag{A.16}$$

$$dT = -\frac{Q}{kA}dx \tag{A.17}$$

Integrating LHS of A.17 from T_1 to T_2 and RHS of A.17 from x_1 to x_2,

$$\int_{T_1}^{T_2} dT = -\frac{Q}{kA}\int_{x_1}^{x_2} dx \tag{A.18}$$

$$\int_{T_2}^{T_1} dT = \frac{Q}{kA}\int_{x_1}^{x_2} dx \tag{A.19}$$

$$\int_{T_2}^{T_1} dT = \frac{Q}{kA}(x_2 - x_1) \tag{A.20}$$

Since $x_2 - x_1$ is the length of the element,

$$T_1 - T_2 = Q\frac{L}{kA} \tag{A.21}$$

$$\Delta T = Q\frac{L}{kA} \tag{A.22}$$

Comparing A.15 with A.22, the resistance between the surfaces is defined as,

$$R_e = \frac{L}{kA}, \tag{A.23}$$

where A is the effective area of the resistance and k is the thermal conductivity.

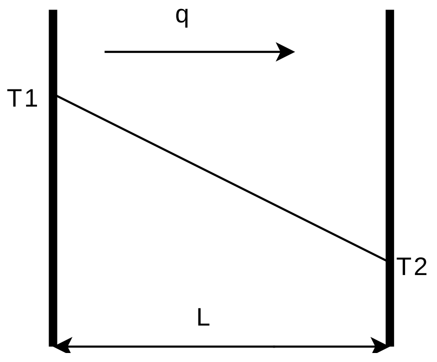


Figure A.3: Diagram representing heat flow in a slab

Analogy between thermal capacitance and electrical capacitance
For electrical circuits, we have:

$$C\frac{dV}{dt} = I_{\text{in}} - I_{\text{out}}, \tag{A.24}$$

where C is the capacitor,
V is the voltage of the circuit,
I_{in} and I_{out} are the currents coming in and going out of the circuit respectively.
Equation A.24 is analogous to the heat transfer equation:

$$C\frac{dT}{dt} = q_{in} - q_{out}, \tag{A.25}$$

where C is the capacitor,
T is the temperature,
q_{in} and q_{out} are the heats coming in and going out of the circuit respectively.

A.4.1.2 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning. This is provided with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- Decay heat: The heat released as a result of radioactive decay.
- Delayed neutron: Neutron emitted by one of the fission products anytime from a few milliseconds to a few minutes later.
- Delayed neutron precursors: Neutron-emitting fission fragments that undergo a stage of radioactive decay and yield an additional neutron called a delayed neutron.
- Fuel pellet: a piece of nuclear fuel usually in the shape of a sphere or cylinder.
- Flux depression: The lowering of the particle's flux density in the neighbourhood of an object due to absorption of particles in the object.
- Heat Flux: The rate of heat energy transfer per unit area.
- Linear Element power: The power generated per unit length of the fuelpin.

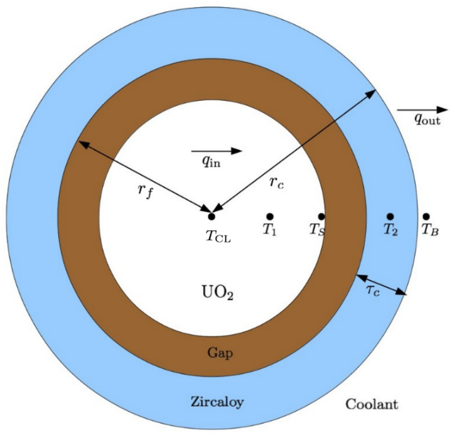


Figure A.4: Fuel pellet representation

- Prompt neutron: A neutron immediately emitted by a nuclear fission event.
- Reactor trip: Emergency shutdown in the nuclear reactors.
- Specific heat: heat capacity per unit mass
- Thermal Capacitance: The amount of heat required to change a substance's temperature by a given amount.
- Thermal Conduction: the transfer of heat energy through a substance.
- Thermal Diffusivity: The thermal conductivity divided by density and specific heat capacity at constant pressure.
- Thermal Resistance: Measure of a temperature difference by which an object or material resists a heat flow through it.
- Transient: Changing with time.

A.4.1.3 Physical System Description

The physical system of the FP, as shown in Figure A.4 includes the following elements:

PS1: Fuel pellet made of Uranium dioxide (UO_2).

PS2: The clad material zircaloy covering the pellet.

PS3: Coolant surrounding the clad material.

NOTE: The temperatures T_{CL}, T_1, T_S, T_2, T_B in the Figure A.4 will be discussed later in this document.

A.4.1.4 Goal Statements

The goals of FP are as follows:

G1: Given fuel power versus time as input, predict transient reactor fuel and clad temperatures.

G2: Given the neutron flux versus time as input, predict transient reactor fuel and clad temperatures.

G3: Given the reactivity transient as input, predict transient reactor fuel and clad temperatures.

G4: Given the trip setpoints, number of trips to initiate shutdown, shutdown reactivity transient as inputs, simulate reactor trip and shutdown.

A.4.2 Solution Characteristics Specification

This section specifies the solution characteristics of the intended software product. The purpose is to reduce the physical problem described in Section A.4.1 to one expressed in mathematical terms. The section starts with description of the assumptions that are made, and then describes the theoretical models that are relevant for the FP problem. Data definitions, instanced models, data constraints, and the expected system behaviour are also given.

A.4.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the data definition or the instance model in which the respective assumption is used.

A1: Axial conduction in the pellet is ignored [GD2].

A2: The pellet has radial symmetry [GD2].

A3: Averaged thermal conductivity value is considered [DD6].

A4: The Urbanic, Heidrick model is used in modelling metal water reaction [DD5].

A5: Approximation of $\ln \frac{r_o}{r_i}$ as $\frac{\tau_c}{r}$ and r_o as r_i [DD7].

A6: Assume isotropic thermal conductivity [T2].

A7: Cylindrical coordinate system is used [GD2].

A8: The spacial effects are neglected in the reactor kinetics formulations [IM5].

A9: Newton's law of convective cooling applies in the gap between the pellet surface and the clad [DD8].

A10: Newton's law of convective cooling applies between the clad surface and the coolant film[DD9, DD12].

A11: Unit length of fuel rod is being modelled [GD2].

A12: Initially, the average clad temperature (T_2) is less than 1000^oC.

A13: The clad thickness (τ_c) is constant even if the zircaloy is consumed in the metal water reaction.

A.4.2.2 Theoretical Models

This section focuses on the general equations, laws used to model a fuelpin.

Number	T1
Label	**Conservation of energy**
Equation	$-\nabla \mathbf{q} + q''' = \rho C \frac{\partial T}{\partial t}$
Description	The above equation gives the conservation of energy for a time varying heat transfer in a material of specific heat capacity C and density ρ where $\mathbf{q}$ is the thermal flux vector, q''' is the volumetric heat generation, T is the temperature and ∇ is the gradient operator.

Number	T2
Label	**Constitutive Equation (Fourier's Law)**
Equation	$\mathbf{q}{=}-k(T)\nabla T$
Description	Fourier's law states that the heat flux is propositional to slope or the gradient of temperature, where k is a function of temperature. This law is based on the assumption that the material is isotropic (A6).

Number	T3
Label	**Space-Time kinetics**
Equation	Beyond the scope of this document.
Description	Space-Time kinetics give the relative distribution of the neutrons over space and time.

Number	T4
Label	**Decay Heat Equations**
Equation	Beyond the scope of this document.
Description	Decay heat equations are used in finding the fuel power when neutron flux is given. It is a summation of the fuel power generated by prompt fissions and the fuel power generated by delayed decay heat components due to fission product decay.

A.4.2.3 General Definitions

This section collects the laws and equations that will be used in deriving the data definitions, which in turn are used to build the instance models

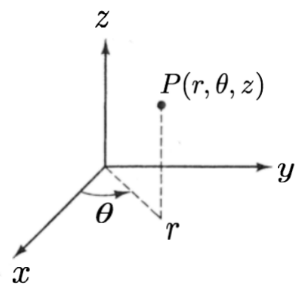


Figure A.5: Cylindrical coordinate system

Number	GD1
Label	**Cylindrical coordinate system**
Units	-
SI equivalent	-
Equation	$\nabla = \hat{r}\frac{\partial}{\partial r} + \hat{\theta}\frac{1}{r}(\frac{\partial}{\partial \theta}) + \hat{z}\frac{\partial}{\partial z}$ where $\hat{r}$, $\hat{\theta}$ and $\hat{z}$ are unit vectors. In matrix notation, this appears as: $\nabla = \begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{1}{r}\frac{\partial}{\partial \theta} \\ \frac{\partial}{\partial z} \end{bmatrix}$ The divergence ∇A is calculated as: $\nabla \text{A} = \frac{\partial(A_r)}{\partial r} + \frac{1}{r}\frac{\partial A_\theta}{\partial \theta} + \frac{\partial A_z}{\partial z}$
Description	The spatial location in a cylindrical coordinate system is expressed in terms of $\hat{r}$, $\hat{\theta}$, $\hat{z}$ as shown in the Figure A.5. The gradient operator is defined as shown above.
Sources	[8, page 12];

Number	GD2
Label	**Average temperature of a hollow cylinder**
Units	$M^0L^0t^0T$
SI equivalent	^oC
Symbol	T_{AVG}
Equation	$T_{\text{AVG}} = \frac{1}{A}\int_A T(r)dA$, with $T(r)$ satisfying $\frac{1}{r}\frac{d}{dr}(kr\frac{dT(r)}{dr}) + q''' = 0$
Description	T_{AVG} is the average temperature of the cylinder, A is the area of the cylinder and $T(r)$ is the temperature at radius r.

Detailed derivation of average temperature:
Applying the Fourier's law from TM2 to Conservation of energy equation in TM1, gives

$$\nabla k \nabla T + q''' = \rho C \frac{\partial T}{\partial t} \tag{A.26}$$

In steady state, the transient features die and (29) changes to

$$\nabla k \nabla T + q''' = 0 \tag{A.27}$$

Applying A7 and writing the above equation in cylindrical coordinate system (GD1),

$$k[\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial T}{\partial r})+\frac{1}{r^2}\frac{\partial}{\partial \theta}(\frac{\partial T}{\partial \theta})+\frac{\partial}{\partial z}(\frac{\partial T}{\partial z})]+q'''=0 \tag{A.28}$$

Ignoring axial conduction (A1), makes $\frac{1}{r^2}\frac{\partial}{\partial \theta}(k\frac{\partial T}{\partial \theta})=0$ and having radial symmetry (A2) makes $\frac{\partial}{\partial z}(k\frac{\partial T}{\partial z})=0$
So, applying A1 and A2 to (31), it simplifies to:

$$\frac{1}{r}\frac{d}{dr}(kr\frac{dT}{dr})+q'''=0 \tag{A.29}$$

Average temperature of the hollow cylinder is found by taking the volume averaged value of the temperature at radius r. Let r_1 be the inner radius and r_2 be the outer radius of a hollow cylinder. Then the average temperature of the cylinder is given by

$$T_{\text{AVG}}=\frac{\int_V T(r,z,\theta)dV}{V} \tag{A.30}$$

Taking A1 and A2 and A11 into account,

$$T_{\text{AVG}}=\frac{1}{A}\int_A T(r)dA \tag{A.31}$$

Number	GD3
Label	**Effective thermal resistance**
Symbol	R_{EFF}
Units	$M^{-1}L^{-2}Tt^3$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{EFF}}=\frac{\Delta T}{q}$
Description	In some cases at steady state, the relation between the temperature change (ΔT) and the thermal flux (q) is modelled as ΔT being directly proportional to q. The proportionality constant can be derived using thermodynamic theory and then relabelled as R_{EFF}. This is analoguos to the electric circuit equation of V=IR. As for the case of electric resistors in series, if n resistors $(R_1,R_2.....R_n)$ are connected in series between two temperatures and if constant heat is flowing between those temperatures, then $R_{\text{EFF}}=R_1+R_2+....+R_n$

Number	GD4
Label	**Rate of change of temperature**
Equation	$C\frac{dT_{\text{AVG}}}{dt}=q_{\text{in}}-q_{\text{out}}+q_g$
Description	The basic equation governing the rate of change of temperature with time. where C is the thermal capacitance ($\frac{\text{kW}s}{m^oC}$) $q_{\text{in}},q_{\text{out}}$ are the linear in and out heat transfer rates respectively ($\frac{\text{kW}}{m}$) and q_g is the rate of internal heat generated.

Detailed derivation of Rate of change of temperature :
Integrating T1 over the volume (V),

$$-\int_V \nabla\mathbf{q}dV+\int_V q'''dV=\int_V \rho C\frac{\partial T}{\partial t}dV \tag{A.32}$$

Applying Gauss's Divergence theorem to the first term over surface S,

$$-\int_S \mathbf{q}\cdot\hat{\mathbf{n}}dS+\int_V q'''dV=\int_V \rho C\frac{\partial T}{\partial t}dV \tag{A.33}$$

Taking A11 into consideration, the volume average gets changed to area average.

$$-\int_S \mathbf{q}\cdot\hat{\mathbf{n}}dS+\int_A q'''dA=\int_A \rho C\frac{\partial T}{\partial t}dA \tag{A.34}$$

Consider a hollow cylinder as in Figure A.6. The integral over the surface can be summarized as $q_{in}-q_{out}$. The integral of q''' over the area becomes the generated thermal energy q'_g. Hence (A.34) can be written as,

$$q_{\text{in}}-q_{\text{out}}+q_g=\int_A \rho C\frac{\partial T}{\partial t}dA \tag{A.35}$$

Taking the time derivative of GD2.

$$\frac{dT_{\text{AVG}}}{dt}=\frac{1}{A}\int_A \frac{dT}{dt}dA \tag{A.36}$$

Rearranging the above equation,

$$A\frac{dT_{\text{AVG}}}{dt}=\int_A \frac{dT}{dt}dA \tag{A.37}$$

Assuming there are representative values of specific heat (c_p) and density (ρ) and multiplying the above equation with ρC_{rep}:

$$\rho C_{rep}A\frac{dT_{\text{AVG}}}{dt}=\int_A \rho C_{rep}\frac{\partial T}{\partial t}dA \tag{A.38}$$

Replacing the RHS of (A.35) with the LHS of (A.38),

$$\rho C_{rep}A\frac{dT_{\text{AVG}}}{dt}=q_{\text{in}}-q_{\text{out}}+q_g \tag{A.39}$$

Number	GD5
Label	**Newton's law of cooling**
Units	$MLt^{-3}T^0$
SI equivalent	$\frac{\text{kW}}{\text{m}}$
Equation	$q_{\text{newt}}=hA\Delta T(t)$
Description	Newton's law of cooling describes the convection cooling and is stated as "rate of heat loss of a body is proportional to the difference in temperatures between the body and its surroundings." q_{newt} is the thermal flux. h is the heat transfer coefficient (assumed independent of T here) $(\frac{\text{W}}{\text{m}^2\text{K}})$ A is the surface area of the heat being transferred (m^2) $\Delta T(t)=T(t)-T_{\text{env}}$ is the time-dependent thermal gradient between environment and object. Newton's law of cooling can be derived from Fourier's law (T2)

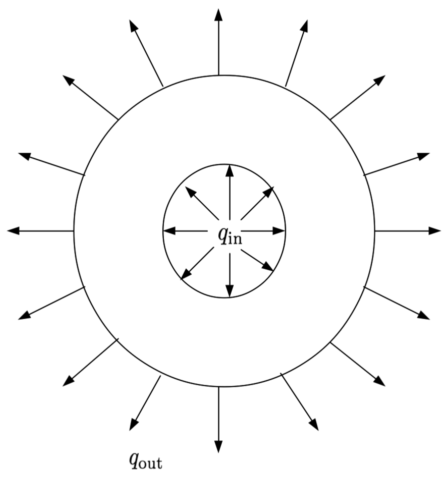


Figure A.6: Heat transfer in a hollow cylinder

Number	GD6
Label	**Effective heat transfer coefficient**
Units	$M^{-1}L^{2}t^{-3}T^{-1}$
SI equivalent	$\frac{\text{W}}{\text{m}^2\text{K}}$
Equation	$h_{\text{EFF}} = \frac{q}{A\Delta T(t)}$
Description	q is the thermal flux. h_{EFF} is the effective heat transfer coefficient ($\frac{\text{W}}{\text{m}^2\text{K}}$) A is the surface area of the heat being transferred (m^2) $\Delta T(t) = T(t) - T_{\text{env}}$ is the time-dependent thermal gradient between environment and object. The heat transfer coefficient is modelled after Newton's law of cooling. It takes into account all relevant modes of heat transfer.

A.4.2.4 Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also declared.

Number	DD1
Label	**Relation between linear element power and volumetric heat generation**
Symbol	q'_N
Units	$MLt^{-3}T^0$
SI equivalent	$\frac{\text{kW}}{\text{m}}$
Equation	$q'_N = \pi r_f^2 q'''$
Description	q''' is the volumetric heat generation and r_f is the fuel radius. The linear element power (q'_N) is found by multiplying the volumetric heat generation by the cross-sectional area of the fuel element.
Sources	[8, page 2–3];

Number	DD2
Label	**Fuel stored energy**
Symbol	$\Delta H(T_{\text{abs}})$
Units	$M^0L^2t^{-2}T^0$
SI equivalent	$\frac{\text{J}}{\text{kg}}$
Equation	$\Delta H(T_{\text{abs}}) = K_0(K_1\theta((e^{\frac{\theta}{T_{\text{abs}}}} - 1)^{-1} - (e^{\frac{\theta}{298}} - 1)^{-1}) + K_2(T_{\text{abs}}^2 - 298^2) + K_3 e^{\frac{-E_D}{R_D T_{\text{abs}}}})$
Description	The stored energy ($\Delta H(T_{\text{abs}})$) calculated is the change in fuel enthalpy from room temperature (298^oK) to the absolute value of the average fuel temperature T_1 (T_{abs}). The values of the constants are given by the Table TB5
Sources	[8, page 12];

Number	DD3
Label	**Integrated fuel power**
Symbol	$P_{\text{F,SUM}}$
Units	FPS
SI equivalent	-
Equation	$P_{\text{F,SUM}}(t_i)= \int_0^{t_i} q'_{\text{NFRAC}}(t)dt$
Description	The above equation gives the integrated fuel power at t_i, where q'_{NFRAC} is the relative fuel power and $P_{\text{F,SUM}}(t_i)$ is the integrated fuel power at t_i
Sources	[8, page 12];

Number	DD4
Label	**Temperature feedback reactivity**
Symbol	$\rho_{\text{TFB,i}}$
Units	-
SI equivalent	mk
Equation	$\rho_{\text{TFB,i}} = A(T_{1,i} - T_{1,0})$
Description	A is a constant ($\frac{\text{mk}}{^{o}\text{C}}$) and $T_{1,0}$ as the initial average temperature (^{o}C)
Sources	[8, page 11];

Number	DD5
Label	**Metal water reaction**
Symbol	q'_{MWR}
Units	$MLt^{-3}T^{0}$
SI equivalent	$\frac{\text{kW}}{\text{m}}$
Equation	$R_{\text{ox}} = [\frac{A}{1.56\delta_{\text{ox}}}]e^{\frac{-B}{R(T_2+273)}}$ if ($\delta_{\text{ox}} \geq \tau_c$) $q'_{\text{MWR}} = 0$ else $q'_{\text{MWR}} = R_{\text{ox}} 2\pi r_c \rho_2 q_r$
Description	δ_{ox} is the reacted zircaloy thickness (m) R_{ox} is the rate of oxidization q'_{MWR} is the metal water reaction heat ($\frac{\text{kW}}{\text{m}}$) ρ_2 is the clad density ($\frac{\text{kg}}{\text{m}^3}$) r_c is the clad radius τ_c is the clad thickness T_2 is the average clad temperature q_r is the heat of reaction ($\frac{\text{kJ}}{\text{kg}}$) and its value is given in (TB1) A, B/R are constants with their values given in (TB1) The Urbanic, Heidrick model is used to predict the oxidation rate from the parabolic rate law (A4)
Sources	[8, page 11];

Number	DD6
Label	**Effective thermal resistance of fuel**
Symbol	R_{FUEL}
Units	$M^{-1}L^{-2}t^{3}T$
SI equivalent	$\frac{\text{m}^{o}\text{C}}{\text{kW}}$
Equation	$R_{\text{FUEL}} = \frac{f}{4\pi k_{\text{AV}}}$
Description	R_{FUEL} is the effective thermal resistance between the temperatures T_{CL} and T_S. $\frac{R_{\text{FUEL}}}{2}$ is the effective thermal resistance between T_{CL} and T_1 and between T_1 and T_S f is the flux depression factor k_{AV} is the average fuel conductivity
Sources	[8, page 3];

Detailed derivation of R_{FUEL}:

From Equation A.29 of (GD2),

$$\frac{1}{r}\frac{d}{dr}(kr\frac{dT}{dr}) + q''' = 0 \tag{A.40}$$

Integrating and rearranging the above equation gives:

$$kr\frac{dT}{dr} = -\int_0^r rq'''dr \tag{A.41}$$

$$\Rightarrow kr\frac{dT}{dr} = \frac{-q'''r^2}{2} \tag{A.42}$$

$$\Rightarrow k\frac{dT}{dr} = \frac{-q'''r}{2} \tag{A.43}$$

$$\Rightarrow kdT = \frac{-q'''r}{2}dr \tag{A.44}$$

For a fuel pellet with outer radius r_f, integrating the RHS of (A.44) from 0 to r_f and LHS of (A.44) from T_{CL} to T_S,

$$\int_{T_{\text{CL}}}^{T_S} dT = \frac{-q'''}{2}\int_0^{r_f} \frac{r}{k}dr \tag{A.45}$$

$$\Rightarrow T_S - T_{\text{CL}} = \Delta T = -q'''\frac{r_f^2}{4k} \tag{A.46}$$

Multiplying both sides of (A.46) by minus sign,

$$T_{\text{CL}} - T_S = \Delta T = q'''\frac{r_f^2}{4k} \tag{A.47}$$

Applying A3 to (A.47),

$$T_{\text{CL}} - T_S = \Delta T = q'''\frac{r_f^2}{4k_{\text{AV}}} \tag{A.48}$$

Replacing the volumetric heat generation by the linear element power using the relation from DD1, removes the dependence of ΔT on the pellet radius. Rewriting (A.48),

$$T_{\text{CL}} - T_S = \frac{q_N'}{4\pi k_{\text{AV}}} \tag{A.49}$$

Taking flux depression factor (f) of the fuel pellet into consideration, (A.49) can be written as,

$$T_{\text{CL}} - T_S = \Delta T = (\frac{f}{4\pi k_{\text{AV}}})q_N' \tag{A.50}$$

Comparing the above equation to (GD3) shows that the effective thermal resistance, R_{FUEL} in this case, is:

$$R_{\text{FUEL}} = \frac{f}{4\pi k_{\text{AV}}} \tag{A.51}$$

Number	DD7
Label	R_{CLAD}
Units	$M^{-1}L^{-2}t^{3}T$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{CLAD}} = \frac{\tau_c}{2\pi r_c k_c}$
Description	The clad resistance is a function of the clad thermal conductivity. It is obtained from the expression for heat transfer by conduction through a hollow cylinder with inner radius r_i and outer radius r_o where k_c is the clad conductivity ($\frac{\text{kW}}{\text{m}^{\text{o}}\text{C}}$) and is given as, $\frac{\Delta T}{q}=\frac{\ln\frac{r_o}{r_i}}{2\pi k_c}$ Taking A5 into consideration, we get $\frac{\Delta T}{q}= \frac{\tau_c}{2\pi r_c k_c}$ Comparission to GD3, shows that effective thermal resistance $R_{\text{CLAD}} = \frac{\tau_c}{2\pi r_c k_c}$
Sources	[8, page 4], [18, page 5] ;

Number	DD8
Label	R_{GAP}
Units	$M^{-1}L^{-2}t^{3}T$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{GAP}} = \frac{1}{2\pi r_c h_p}$
Description	R_{GAP} is the gap resistance where r_c is the clad radius (m), h_p is the initial gap conductance ($\frac{\text{kW}}{\text{m}^{2\text{o}}\text{C}}$) which is an input parameter
Sources	source code

Derivation of R_{GAP}

Taking A9 into consideration, we use Newton's law of cooling to derive R_{GAP}. The area of the clad (A_c) is

$$A_c = 2\pi r_c \tag{A.52}$$

Substituting Equation A.52 into GD5, and considering h_p as the initial gap conductance (heat transfer coefficient), we get,

$$q_{\text{gap}} = 2\pi r_c h_p \Delta T \tag{A.53}$$

From GD3, the gap resistance (R_{GAP}) can be given as,

$$R_{\text{GAP}} = \frac{\Delta T}{q_{\text{gap}}} \tag{A.54}$$

Substituting Equation A.53 into Equation A.54 and simplifying gives,

$$R_{\text{GAP}} = \frac{1}{2\pi r_c h_p} \tag{A.55}$$

Number	DD9
Label	R_{FILM}
Units	$M^{-1}L^{-2}t^{3}T$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_{\text{FILM}} = \frac{1}{2\pi r_c h_b}$
Description	R_{FILM} is the coolant film resistance where r_c is the outer clad radius (m), h_b is the coolant film conductance ($\frac{\text{kW}}{m^2C}$) (Figure A.8) NOTE: Equation taken from the code
Sources	source code

Derivation of R_{FILM}
Assuming that Newton's law of convective cooling applies between the clad and the coolant (A10), we use Newton's law of cooling to derive R_{FILM}. The area of the clad (A_c) is

$$A_c = 2\pi r_c \tag{A.56}$$

Substituting Equation A.56 into GD5, and considering h_b as the coolant film conductance, we get,

$$q_{\text{coolant}} = 2\pi r_c h_b \Delta T \tag{A.57}$$

From GD3, the coolant film resistance (R_{FILM}) can be given as,

$$R_{\text{FILM}} = \frac{\Delta T}{q_{\text{coolant}}} \tag{A.58}$$

Substituting Equation A.57 in Equation A.58 and simplifying gives,

$$R_{\text{FILM}} = \frac{1}{2\pi r_c h_b} \tag{A.59}$$

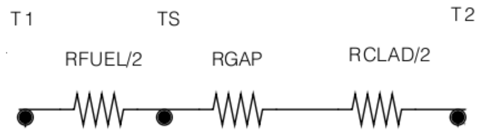


Figure A.7: Thermal circuit between T_1 and T_2

Number	DD10
Label	R_3
Units	$M^{-1}L^{-2}t^3T$
SI equivalent	$\frac{\text{m}^o\text{C}}{\text{kW}}$
Equation	$R_3 = \frac{1}{2\pi r_f h_g}$
Description	R_3 is the effective thermal resistance between T_S and T_2 (Figure A.8) where r_f is the fuel radius (m) h_g is the gap film conductance (kw/$m^{2o}C$) which is given by DD19
Sources	[8, page 5];

Detailed derivation of R_3:
From the Figure A.7, the effective resistance R_3 between T_S and T_2 is:

$$R_3 = R_{\text{GAP}} + \frac{R_{\text{CLAD}}}{2} \tag{A.60}$$

From GD3, the heat flux between the fuel surface and clad can be given as,

$$q = \frac{\Delta T}{R_3}, \tag{A.61}$$

Taking h_g as the effective heat transfer coefficient between fuel surface and clad, we get the heat flux between the fuel surface and clad from GD6 as

$$q = h_g A_f \Delta T, \tag{A.62}$$

where A_f is the area of the clad given as

$$A_f = 2\pi r_f \tag{A.63}$$

Comparing Equation A.62 and Equation A.61,

$$\frac{\Delta T}{R_3} = h_g A_f \Delta T \tag{A.64}$$

Replacing A_f with its value from Equation A.63 and further simplifying, we get,

$$R_3 = \frac{1}{2\pi r_f h_g} \tag{A.65}$$

Number	DD11
Label	R_1
Units	$M^{-1}L^{-2}t^3T$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_1 = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_g}$
Description	R_1 is the thermal resistance between the average fuel temperature (T_1) and sheath temperature (T_2) (Figure A.7) k_{AV} is the average fuel conductivity r_f is the fuel radius (m) h_g is the gap film conductance (kw/$m^{2o}C$) which is given by DD19
Sources	[8, page 4];

Derivation of R_1:
From the Figure A.7, the effective resistance R_1 between T_1 and T_2 is:

$$R_1 = \frac{R_{\text{FUEL}}}{2} + R_{\text{GAP}} + \frac{R_{\text{CLAD}}}{2} \tag{A.66}$$

From Equation A.60, since R_3 is the effective resistance of gap and half of the clad, the above equation can be rewritten as,

$$R_1 = \frac{R_{\text{FUEL}}}{2} + R_3 \tag{A.67}$$

Substituting the values of R_{FUEL}, R_3 from DD6 and DD10 respectively into Equation A.67 gives:

$$R_1 = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_g} \tag{A.68}$$

Number	DD12
Label	R_2
Units	$M^{-1}L^{-2}t^3T$
SI equivalent	$\frac{\text{m}^{\text{o}}\text{C}}{\text{kW}}$
Equation	$R_2 = \frac{1}{2\pi r_c h_c}$
Description	R_2 is the effective thermal resistance between T_B and T_2 r_c is the outer clad radius (m) h_c is the effective heat transfer coefficient between clad and coolant ($\frac{\text{kw}}{\text{m}^{2o}\text{C}}$) which is given by DD18
Sources	[8, page 5];

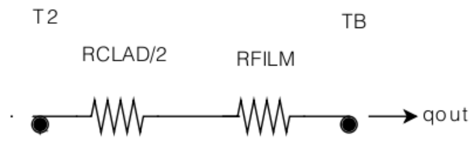


Figure A.8: Thermal circuit between T_2 and T_B

Derivation of R_2
Assuming that Newton's law of convective cooling applies between the clad and the coolant (A10), we use GD6 to derive R_2. Substituting Equation A.56 into GD6, and considering h_c as the effective heat transfer coefficient between clad and coolant, we get,

$$q = 2\pi r_c h_c \Delta T \tag{A.69}$$

From GD3, R_2 can be given as,

$$R_2 = \frac{\Delta T}{q} \tag{A.70}$$

Substituting Equation A.69 into Equation A.70 and simplifying gives,

$$R_2 = \frac{1}{2\pi r_c h_c} \tag{A.71}$$

Number	DD13
Label	R_{CL}
Units	$M^{-1}L^{-2}t^{3}T$
SI equivalent	$\frac{\mathrm{m^oC}}{\mathrm{kW}}$
Equation	$R_{\mathrm{CL}} = \frac{f}{8\pi k_{\mathrm{AV}}}$
Description	R_{CL} is the effective thermal resistance at the centerline, where f is the flux depression factor k_{AV} is the average fuel conductivity
Sources	[8, page 5];

Detailed derivation of R_{CL}:
At some radius r, the temperature T_r is given by,

$$T_r = T_{\mathrm{CL}} - q''' \frac{r^2}{4k_{\mathrm{AV}}} \tag{A.72}$$

$$= T_S + q''' \frac{r_f^2 - r^2}{4k_{\mathrm{AV}}} \tag{A.73}$$

From GD2, the average fuel temperature T_1 is defined as the area averaged value of T_r and is given by,

$$T_1 = T_S + \frac{q'''}{4k_{\mathrm{AV}}\pi r_f^2} \int_{r=0}^{r=r_f} (r_f^2 - r^2) 2\pi r dr \tag{A.74}$$

Performing the integration of (A.74) and rearranging, we get,

$$T_1 - T_S = q''' \frac{r_f^2}{8\pi k_{\mathrm{AV}}} \tag{A.75}$$

Comparing the above equation to (GD3) shows that the effective thermal resistance between T_1 and T_S (R_{CL})in this case is:

$$R_{\text{CL}} = \frac{f}{8\pi k_{\text{AV}}} = \frac{R_{\text{FUEL}}}{2} \tag{A.76}$$

Number	DD14
Label	**Thermal capacitance terms**
Units	$ML^2t^{-2}T^{-1}$
SI equivalent	$\frac{\text{kWs}}{\text{m}^o\text{C}}$
Equation	$C_1 = \pi r_f^2 c_{p,1}\rho_1$ $C_2 = 2\pi r_c \tau_c c_{p,2}\rho_2$ $C_{\text{CL}} = \pi r_f^2 c_{p,3}\rho_1$
Description	$c_{p,1}, c_{p,2}$ and $c_{p,3}$ are the specific heats corresponding to the fuel average, clad and fuel centreline temperatures respectively ($\frac{\text{kJ}}{\text{kg}^o\text{C}}$). ρ_1 and ρ_2 are the fuel and clad densities respectively ($\frac{\text{kJ}}{\text{kg}^o\text{C}}$). r_f and r_c are the fuel and clad radius (m) τ_c is the clad thickness
Sources	[8, page 5];

Number	DD15
Label	k_c
Units	$ML^1t^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^o\text{C}}$
Equation	$k_c = aT_2 + b$
Description	k_c is the clad conductivity where a and b are constants obtained by a least squares fit to tabulated data (TB2).
Sources	[8, page 6];

Number	DD16
Label	K_{AV} **as a polynomial**
Units	$ML^1t^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^o\text{C}}$
Equation	$k = x_1 T + x_0$
Description	k_{AV} is a temperature dependant non-linear variable and is represented as a first order polynomial function of temperature. The values of x_0 and x_1 are given in the (TB3).
Sources	-

Number	DD17
Label	$c_{p,1}$ **as a polynomial**
Units	$M^0L^2t^{-2}T^{-1}$
SI equivalent	$\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$
Equation	$c_p = y_2T^2 + y_1T + y_0$
Description	$c_{p,1}$ is a temperature dependant non-linear variable and is represented as a second order polynomial function of temperature. The values of y_0, y_1 and y_2 are given in the (TB4).
Sources	

Number	DD18
Label	h_c
Units	$ML^0t^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^{2\text{o}}\text{C}}$
Equation	$h_c = \frac{2k_ch_b}{2k_c+\tau_ch_b}$
Description	h_c is the effective heat transfer coefficient between the clad and the coolant τ_c is the clad thickness h_b is initial coolant film conductance k_c is the clad conductivity NOTE: Equation taken from the code
Sources	source code

Derivation of h_c:
From Figure A.8, R_2 is the effective thermal resistance of the coolant film and half of the clad. Adding the R_{FILM} from DD9 with half of the value of R_{CLAD} from DD7, we get

$$R_2 = \frac{1}{2\pi r_c h_b} + \frac{\tau_c}{4\pi r_c k_c} \tag{A.77}$$

Taking the common terms out and rewriting the above equation,

$$R_2 = \frac{1}{2\pi r_c}\left(\frac{1}{h_b} + \frac{\tau_c}{2k_c}\right) \tag{A.78}$$

$$= \frac{1}{2\pi r_c}\left(\frac{2k_c + \tau_c h_b}{2k_c h_b}\right) \tag{A.79}$$

$$= \frac{1}{2\pi r_c\left(\frac{2k_ch_b}{2k_c+\tau_ch_b}\right)} \tag{A.80}$$

But from DD12, R_2 is given as,

$$R_2 = \frac{1}{2\pi r_c h_c} \tag{A.81}$$

Comparing Equation A.80 and Equation A.81 and rearranging gives h_c as,

$$h_c = \frac{2k_ch_b}{2k_c + \tau_c h_b} \tag{A.82}$$

Number	DD19
Label	h_g
Units	$ML^0t^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^2{}^\circ\text{C}}$
Equation	$h_g = \frac{2k_ch_p}{2k_c+\tau_ch_p}$
Description	h_g is the gap conductance τ_c is the clad thickness h_p is initial gap film conductance k_c is the clad conductivity NOTE: Equation taken from the code
Sources	source code

Derivation of h_g:
From the Figure A.7, the effective thermal resistance between T_2 and T_S is, the effective resistance of the gap film and half of the clad. Adding the R_{GAP} from DD8 with half of the value of R_{CLAD} from DD7, we get

$$R_3 = \frac{1}{2\pi r_c h_p} + \frac{\tau_c}{4\pi r_c k_c} \tag{A.83}$$

where R_3 is the effective resistance
Taking the common terms out and rewriting the above equation,

$$R_3 = \frac{1}{2\pi r_c}\Big(\frac{1}{h_p} + \frac{\tau_c}{2k_c}\Big) \tag{A.84}$$

$$= \frac{1}{2\pi r_c}\Big(\frac{2k_c+\tau_c h_p}{2k_c h_p}\Big) \tag{A.85}$$

$$= \frac{1}{2\pi r_c\Big(\frac{2k_c h_p}{2k_c+\tau_c h_p}\Big)} \tag{A.86}$$

But from DD10, R_3 is given as,

$$R_3 = \frac{1}{2\pi r_f h_g} \tag{A.87}$$

Comparing Equation A.87 and Equation A.86 and rearranging gives h_g as,

$$h_g = \frac{2k_c h_p}{2k_c + \tau_c h_p} \tag{A.88}$$

Number	DD20
Label	**Incore flux detector signal**
Units	-
SI equivalent	-
Equation	$D = (1-\hat{\alpha})N + \Sigma_{i=1}^{5} d_i$ $d_i = \hat{\psi}_i(\hat{\alpha}_i N - d_i)$
Description	D is the relative detector signal amplitude $\hat{\alpha}_i$ is the i^{th} delay fraction $\hat{\alpha}$ is the total delayed fraction $\hat{\psi}_i$ is the i^{th} decay constant (s^{-1}) d_i is the relative amplitude of delay group i N is the neutron flux
Sources	[8, page 10];

Number	DD21
Label	**Compensated detector signal**
Units	-
SI equivalent	-
Equation	$D'(s) = D(K_1 + \frac{K_2}{1+T_2 s} + \frac{K3}{1+T_3 s})$
Description	The compensated signal D' is given by using the Laplace transformation of $D(s)$ D is the relative detector signal amplitude K_1 is the prompt response fraction K_2, K_3 are the delayed response fractions T_2, T_3 are the delay times (s)
Sources	[8, page 10];

Number	D22
Label	**Signal amplifier response**
Units	-
SI equivalent	-
Equation	$A^*_K = A^*_{K-1} e^{\frac{-\Delta t}{\tau_A}} + (1 - e^{\frac{-\Delta t}{\tau_A}}) A_K$
Description	A_K is the value of the trip parameter at t_k A^*_K is the filtered value of the trip parameter at t_k τ_A is the amplifier time constant(s) The delay introduced by the signal amplifier is simulated for each trip parameter by the above first order filter. The log rate signal is filtered by two cascaded first order filters.
Sources	[8, page 11];

Number	DD23
Label	**Fuel surface temperature**
Units	$M^0 L^0 t^0 T$
SI equivalent	^{o}C
Equation	$T_S = T_2 + \frac{T_1 - T_2}{R_1} R_3$
Description	T_S is the surface fuel temperature T_1 is the average fuel temperature T_2 is the average clad temperature R_1 is the effective resistance between average fuel and average clad temperatures ($\frac{\text{m}^o\text{C}}{\text{kW}}$) R_3 is the effective resistance between the clad and the fuel surface temperatures ($\frac{\text{m}^o\text{C}}{\text{kW}}$)
Sources	[8, page 6];

Derivation of T_S:
From GD3, the heat flux generated between T_S and T_2 can be given as,

$$q_{\text{surface}} = \frac{T_S - T_2}{R_{\text{GAP}} + \frac{R_{\text{CLAD}}}{2}}, \tag{A.89}$$

where $R_{\text{GAP}} + \frac{R_{\text{CLAD}}}{2}$ is the effective resistance between T_S and T_2 and $T_S - T_2 = \Delta T$.
Replacing effective resistance between T_S and T_2 by R_3 (DD10), Equation A.89 simplifies

to,

$$q_{\text{surface}} = \frac{T_S - T_2}{R_3}, \tag{A.90}$$

Similarly the heat flux generated between T_1 and T_2 can be given as,

$$q_{\text{avgfuel}} = \frac{T_1 - T_2}{R_1}, \tag{A.91}$$

where R_1 is the effective resistance between T_1 and T_2 and $T_1 - T_2 = \Delta T$.
By the continuityof thermal flux, Equation A.90 becomes equal to Equation A.91. That is,

$$\frac{T_S - T_2}{R_3} = \frac{T_1 - T_2}{R_1} \tag{A.92}$$

Rearranging the above equation gives,

$$T_S = T_2 + \frac{T_1 - T_2}{R_1} R_3 \tag{A.93}$$

Number	DD24
Label	**Linear and Log rates**
Symbol	$R_{\text{LIN}}, R_{\text{LOG}}$
Units	$M^0L^0t^0T^{-1}$
SI equivalent:	s^{-1}
Equation	$R_{\text{LIN}} = \frac{N_k - N_{k-1}}{\Delta t}$ $R_{\text{LOG}} = \frac{\ln N_k - \ln N_{k-1}}{\Delta t}$
Description	R_{LIN} is the linear rate (s^{-1}) R_{LOG} is the log rate (s^{-1}) N_k is the relative neutron flux at t_k Δt is the solution interval $t_k - t_{k-1}$
Sources	[8, page 10];

Number	DD25
Label	**Relation between linear element power and relative fuel power**
Units	$MLt^{-3}T^0$
SI equivalent	$\frac{\text{kW}}{\text{m}}$
Equation	$q'_N = q'_{\text{NFRAC}} q'_{N_{\max}}$
Description	q'_{NFRAC} is the relative fuel power q'_N is linear element power ($\frac{\text{kW}}{\text{m}}$) $q'_{N_{\max}}$ is the full power linear element power ($\frac{\text{kW}}{\text{m}}$)
Sources	[8, page 9];

Number	DD26
Label	q'_{MWRI}
Units	-
SI equivalent	-
Equation	$q'_{\text{MWRI}}(t_i) = \frac{1}{q'_{N_{\max}}} \int_0^{t_i} q'_{\text{MWR}}(t)dt$
Description	$q'_{\text{MWRI},N}$ gives the integrated metal water reaction heat at t_N. It is a summation of q'_{MWR} normalized by $q'_{N_{\max}}$ at each time step $q'_{\text{MWR},i}$ is the metal water reaction heat at t_i

Number	DD27
Label	q'_{out}
Units	$Mt^{-3}T^{-1}$
SI equivalent	$\frac{\text{kW}}{\text{m}^2{}^\text{o}\text{C}}$
Equation	$q'_{\text{out}} = \frac{1}{q'_{N_{\max}}}\left(\frac{T_2-T_B}{R_2}\right)$
Description	q'_{out} is the output heat from the reaction that is sent into the coolant R_2 is the effective resistance between the coolant film and the clad ($\frac{\text{m}^\text{o}\text{C}}{\text{kW}}$) $q'_{N_{max}}$ is the linear element power at full power ($\frac{\text{kW}}{\text{m}^\text{o}\text{C}}$) T_2 is average clad temperature T_B is coolant temperature NOTE: Equation taken from the code

Derivation of q'_{out}:
From the Figure A.8, the effective resistance R_2 between T_2 and T_B is given by DD12. According to GD3, the R_2 between T_2 and T_B can be given as,

$$R_2 = \frac{\Delta T}{q'_{\text{out}}}, \tag{A.94}$$

where q'_{out} is the heat generated between T_2, T_B and

$$\Delta T = T_2 - T_B \tag{A.95}$$

Substituting Equation A.95 and DD12 in A.94 and rearranging gives,

$$q'_{\text{out}} = \frac{(T_2 - T_B)}{R_2} \tag{A.96}$$

Normalizing the above equation by $q'_{N_{\max}}$ (standard presentation) gives,

$$q'_{\text{out}} = \frac{1}{q'_{N_{\max}}}\frac{(T_2 - T_B)}{R_2} \tag{A.97}$$

Number	DD28
Label	dryout
Units	$-$
SI equivalent	$-$
Equation	if($q'_{\text{out}} \geq p_{\text{dry}}$) $h_b = h_{\text{dry}}$
Description	We compare the power sent to the coolant (q'_{out}) with the dryout threshold power (p_{dry}) and once it exceeds this maximum permissible power, dryout occurs. When the dryout occurs, the coolant film conductance (h_b) becomes equal to the heat transfer coefficient between the fuel surface and the coolant at dryout (h_{dry}). NOTE: The meaning of h_b is given through the definition of R_{FILM} (DD9)

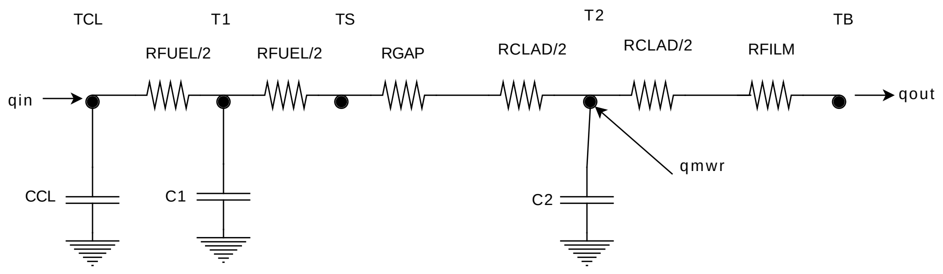


Figure A.9: Electrical Circuit Analogue

A.4.2.5 Instance Models

This section reduces the problem defined in the problem description into one which is expressed in mathematical terms. It uses concrete symbols defined in section “Data Definitions” to replace the abstract symbols in the models identified in the section “Theoretical Models” and “Genenral Definitios”.

Number	IM1
Label	Rate of change of average fuel temperature
Equation	$C_1 \frac{dT_1}{dt} = q'_N - \frac{T_1 - T_2}{R_1}$
Description	T_1 is the average fuel temperature T_2 is the average clad temperature R_1 is the effective resistance between average fuel and average clad temperatures C_1 is the thermal capacitance of the fuel
Sources	[8, page 6];

Derivation of Rate of change of average fuel temperature:
To find the rate of change of average fuel temperature, we use GD4. Consider $c_{p,1}$ as the specific heat evaluated at T_1, ρ_1 as the density of the fuel and A_f as area of fuelpellet which is given as,

$$A_f = \pi r_f^2 \tag{A.98}$$

Now substituting $c_{p,1}$, ρ_1 and Equation A.98 in Equation A.39 gives,

$$\rho_1 c_{p,1} \pi r_f^2 \frac{dT_1}{dt} = q_{\text{in}} - q_{\text{out}} + q_g \tag{A.99}$$

From DD14 the capacitance term C_1 is given as,

$$C_1 = \pi r_f^2 c_{p,1} \rho_1 \tag{A.100}$$

Rewriting Equation A.99 and substituting Equation A.100 in Equation A.99 gives

$$C_1 \frac{dT_1}{dt} = q_{\text{in}} - q_{\text{out}} + q_g \tag{A.101}$$

The amount of q_{in} is zero at T_1. That is,

$$q_{\text{in}} = 0 \tag{A.102}$$

The value for q_{out} is given by the flux between T_1 and T_2. From the Figure A.9,

$$q_{\text{out}} = \frac{T_1 - T_2}{R_1}, \tag{A.103}$$

where T_2 is the average clad temperature and R_1 is the effective thermal resistance between T_1 and T_2 as given in DD11.
The integral of heat generation is,

$$q_g = q'_N \tag{A.104}$$

Substituting Equation A.102, Equation A.103 and Equation A.104 in Equation A.101 and rearranging gives,

$$C_1 \frac{dT_1}{dt} = q'_N - \frac{T_1 - T_2}{R_1} \tag{A.105}$$

Number	IM2
Label	Rate of change of average clad temperature
Equation	$C_2 \frac{dT_2}{dt} = \frac{T_1 - T_2}{R_1} + q'_{\text{MWR}} - \frac{T_2 - T_B}{R_2}$
Description	T_1 is the average fuel temperature T_2 is the average clad temperature T_B is the coolant temperature R_1 is the effective resistance between average fuel and average clad temperatures R_2 is the effective resistance between clad and coolant temperatures C_2 is the thermal capacitance of the clad q_{MWR} is the Metal-Water reaction heat
Sources	[8, page 6];

Derivation of Rate of change of average clad temperature:
Similar to the rate of change of average fuel temperature, to find the rate of change of average clad temperature, we use GD4. Consider $c_{p,2}$ as the specific heat evaluated at T_2, ρ_2 as the density of the clad and A_c as area of clad which is given as,

$$A_c = 2\pi r_c \tau_c \tag{A.106}$$

Now substituting $c_{p,2}$, ρ_2 and Equation A.106 in Equation A.39 gives,

$$\rho_2 c_{p,2} 2\pi r_c \tau_c \frac{dT_2}{dt} = q_{\text{in}} - q_{\text{out}} + q_g \tag{A.107}$$

From DD14 the capacitance term C_2 is given as,

$$C_2 = 2\pi r_c \tau_c c_{p,2} \rho_2 \tag{A.108}$$

Rewriting Equation A.107 and substituting Equation A.108 in Equation A.107 gives

$$C_2 \frac{dT_2}{dt} = q_{\text{in}} - q_{\text{out}} + q_g \tag{A.109}$$

q_{in} at T_2 is the amount of q_{out} at T_1 (A.103). That is,

$$q_{\text{in}} = \frac{T_1 - T_2}{R_1} \tag{A.110}$$

The value for q_{out} is given by the flux between T_2 and T_B. From the Figure A.9,

$$q_{\text{out}} = \frac{T_2 - T_B}{R_2}, \tag{A.111}$$

where R_2 is the effective thermal resistance between T_2 and T_B as given in DD12.
The integral of heat generation is,

$$q_g = q'_{\mathrm{MWR}} \tag{A.112}$$

Substituting Equation A.110, Equation A.111 and Equation A.112 in Equation A.109 and rearranging gives,

$$C_2 \frac{dT_2}{dt} = \frac{T_1 - T_2}{R_1} + q'_{\mathrm{MWR}} - \frac{T_2 - T_B}{R_2} \tag{A.113}$$

Number	IM3
Label	Rate of change of centerline temperature
Equation	$C_{\mathrm{CL}} \frac{dT_{\mathrm{CL}}}{dt} = q'_N - \frac{T_{\mathrm{CL}} - T_1}{R_{\mathrm{CL}}}$
Description	T_1 is the average fuel temperature T_{CL} is the centerline temperature R_{CL} is the effective resistance t the centerline C_{CL} is the thermal capacitance at the centerline q'_N is the linear element power
Sources	[8, page 6];

Derivation of Rate of centerline temperature:
Similar to the rate of change of average fuel temperature, to find the rate of change of centerline temperature, we use GD4. Consider $c_{p,3}$ as the specific heat evaluated at T_{CL}, ρ_1 as the density of the fuel and A_f as area of fuel pellet which is given by Equation A.98. Now substituting $c_{p,3}$, ρ_1 and Equation A.98 in Equation A.39 gives,

$$\rho_1 c_{p,3} \pi r_f^2 \frac{dT_{\mathrm{CL}}}{dt} = q_{\mathrm{in}} - q_{\mathrm{out}} + q_g \tag{A.114}$$

From DD14 the capacitance term C_{CL} is given as,

$$C_{\mathrm{CL}} = \pi r_f^2 c_{p,3} \rho_1 \tag{A.115}$$

Rewriting Equation A.114 and substituting Equation A.115 in Equation A.114 gives

$$C_{\mathrm{CL}} \frac{dT_{\mathrm{CL}}}{dt} = q_{\mathrm{in}} - q_{\mathrm{out}} + q_g \tag{A.116}$$

From the Figure A.9, q_{in} at T_{CL} is the linear element power q'_N. That is,

$$q_{\mathrm{in}} = q'_N \tag{A.117}$$

The value for q_{out} is given by the flux between T_{CL} and T_1. From the Figure A.9,

$$q_{\mathrm{out}} = \frac{T_{CL} - T_1}{R_{\mathrm{CL}}}, \tag{A.118}$$

where R_{CL} is the effective thermal resistance between T_{CL} and T_1 as given in DD13.
The integral of heat generation is,

$$q_g = 0 \tag{A.119}$$

Substituting Equation A.117, Equation A.118 and Equation A.119 in Equation A.116 and rearranging gives,

$$C_{\mathrm{CL}} \frac{dT_{\mathrm{CL}}}{dt} = q'_N - \frac{T_{\mathrm{CL}} - T_1}{R_{\mathrm{CL}}} \tag{A.120}$$

Number	IM4
Label	Initial thermal conditions
Equation	$\frac{dT_1}{dt} = 0$ (IM1) $\frac{dT_2}{dt} = 0$ (IM2) $k_c = aT_2 + b$ (DD15) $q'_{\text{MWR}} = 0$ $q'_{\text{N,est}} = 4\pi \int_{T_S}^{T_{\text{CL,est}}} kdT = q'_N$
Description	The initial values that are needed to start the simulation are calculated using the above conditions. T_1 is the average fuel temperature T_2 is the average clad temperature T_S is the surface temperature $T_{\text{CL,est}}$ is the estimate of centerline temperature k_c is the clad conductivity which is given by DD15 a, b are constants with their values given by Table TB2 q'_{MWR} is the Metal-Water reaction heat $q'_{\text{N,est}}$ is the estimate of linear element power k is the fuel conductivity
Sources	[8, page 6];

Number	IM5
Label	Point neutron kinetics
Equation	$\dot{N} = [\frac{\rho-\beta}{l^*}]N + \Sigma_{i=1}^{6}\lambda_i c_i$ $\dot{c_i} = -\lambda_i c_i + (\frac{\beta_i}{l^*})N$
Description	N is the neutron flux ρ is the reactivity β_i is the fraction associated with the i^{th} group of delayed neutron precursors β is total delayed neutron fraction l^* is the prompt generation time(s) λ_i is the decay constant associated with the i^{th} group of delayed neutron precursors (s^{-1}) c_i is the delayed neutron precursor for the i^{th} group To make the relative distribution of the neutrons uniform, the space effects on the kinetics equations are to be eliminated. Hence taking A8 into consideration, i.e considering only time effects, the space-time kinetics equations of T3 is reduced to the point kinetics equations. If the reactivity transient is given as input, the transient neutron flux is obtained by solving the point kinetics equations.
Sources	[8, page 6];

Number	IM6
Label	Decay Heat Equations
Equation	$q'_{\text{NFRAC}} = (1-\gamma)N + \Sigma_{i=1}^{3}\lambda_{h,i}W_i$ $\dot{W}_i = -\lambda_{h,i}W_i + \gamma_i N$
Description	N is the neutron flux q'_{NFRAC} is the relative fuel power γ_i is the power fraction associated with the i^{th} decay heat group $\lambda_{h,i}$ is the decay constant associated with the i^{th} decay heat group (s^{-1}) γ is the total delayed power fraction W_i is the relative decay heat amplitude for the i^{th} group The decay equations are used in generating fuel power. The total fuel power is a summation of a prompt neutronic power component (the prompt fission power) and three delayed decay heat components due to fission product decay. If the neutron flux transient is given as input or from the point kinetics routine, the transient fuel power is obtained by solving the decay heat equations.
Sources	[8, page 6];

A.4.2.6 Data Constraints

This section is to clarify the environmental and system limitations imposed on admissible data. It gives the system constraints on the data to validate the models identified in the section “instance Models”. These constraints are listed in the table below. The column physical constraints gives the physical limitations on the range of the values that can be taken by the variable and are given by the domain expert while the column system constraints gives the system limitations on the range of values that can be taken by the variable and are given by the developer.

Variable	Type	Unit	Physical Constraints	System Constraints	Typical value	Property
$\lambda_{h,1}$	Real	s^{-1}	$-\infty < \lambda_{h,1} < \infty$		0.10368	IN
$\lambda_{h,2}$	Real	s^{-1}	$-\infty < \lambda_{h,2} < \infty$		0.000142	IN
$\lambda_{h,3}$	Real	s^{-1}	$-\infty < \lambda_{h,3} < \infty$		0.00476	IN
γ_1	Real	-	$-\infty < \gamma_1 < \infty$		0.0226	IN
γ_2	Real	-	$-\infty < \gamma_2 < \infty$		0.02311	IN
γ_3	Real	-	$-\infty < \gamma_3 < \infty$		0.02078	IN
N	Real		to be discussed			IN-OUT
ρ_1	Real	$\frac{kg}{m^3}$	$-\infty < \rho_1 < \infty$		10600	IN
ρ_2	Real	$\frac{kg}{m^3}$	$-\infty < \rho_2 < \infty$		6570	IN
ρ(reactivity)	Real		to be discussed		0	IN-OUT
τ_c	Real	m	$-\infty < \tau_c < \infty$		0.00042	IN
τ_g	Real	m	$-\infty < \tau_g < \infty$		0.00004	IN
τ_A	Real	-	$-\infty < \tau_A < \infty$		0.015	IN
ΔT	Real	s	$0 < \Delta T \leq 0.0001$		0.01	IN
f	Real	-	$-\infty < f < \infty$		1.0	IN
l^*	Real	s	$-\infty < l^* < \infty$		0.893×10^{-3}	IN
β_1	Real	-	$-\infty < \beta_1 < \infty$		1.769×10^{-4}	IN
β_2	Real	-	$-\infty < \beta_2 < \infty$		11.498×10^{-4}	IN
β_3	Real	-	$-\infty < \beta_3 < \infty$		10.191×10^{-4}	IN
β_4	Real	-	$-\infty < \beta_4 < \infty$		21.057×10^{-4}	IN
β_5	Real	-	$-\infty < \beta_5 < \infty$		7.726×10^{-4}	IN
β_6	Real	-	$-\infty < \beta_6 < \infty$		1.962×10^{-4}	IN
β_7	Real	-	$-\infty < \beta_7 < \infty$		1.61×10^{-7}	IN
β_8	Real	-	$-\infty < \beta_8 < \infty$		3.23×10^{-7}	IN
β_9	Real	-	$-\infty < \beta_9 < \infty$		1.03×10^{-6}	IN
β_{10}	Real	-	$-\infty < \beta_{10} < \infty$		7.48×10^{-6}	IN
β_{11}	Real	-	$-\infty < \beta_{11} < \infty$		6.61×10^{-6}	IN
β_{12}	Real	-	$-\infty < \beta_{12} < \infty$		1.080×10^{-5}	IN
β_{13}	Real	-	$-\infty < \beta_{13} < \infty$		2.240×10^{-5}	IN
β_{14}	Real	-	$-\infty < \beta_{14} < \infty$		6.52×10^{-5}	IN
β_{15}	Real	-	$-\infty < \beta_{15} < \infty$		2.080×10^{-4}	IN
λ_1	Real	s^{-1}	$-\infty < \lambda_1 < \infty$		12.778×10^{-3}	IN
λ_2	Real	s^{-1}	$-\infty < \lambda_2 < \infty$		31.535×10^{-3}	IN
λ_3	Real	s^{-1}	$-\infty < \lambda_3 < \infty$		122.197×10^{-3}	IN
λ_4	Real	s^{-1}	$-\infty < \lambda_4 < \infty$		32.282×10^{-2}	IN
λ_5	Real	s^{-1}	$-\infty < \lambda_5 < \infty$		1389.289×10^{-3}	IN
λ_6	Real	s^{-1}	$-\infty < \lambda_6 < \infty$		3778.336×10^{-3}	IN
λ_7	Real	s^{-1}	$-\infty < \lambda_7 < \infty$		6.26×10^{-7}	IN
λ_8	Real	s^{-1}	$-\infty < \lambda_8 < \infty$		3.63×10^{-6}	IN
λ_9	Real	s^{-1}	$-\infty < \lambda_9 < \infty$		4.37×10^{-5}	IN
λ_{10}	Real	s^{-1}	$-\infty < \lambda_{10} < \infty$		0.117×10^{-3}	IN
λ_{11}	Real	s^{-1}	$-\infty < \lambda_{11} < \infty$		0.428×10^{-3}	IN
λ_{12}	Real	s^{-1}	$-\infty < \lambda_{12} < \infty$		0.150×10^{-2}	IN
λ_{13}	Real	s^{-1}	$-\infty < \lambda_{13} < \infty$		0.481×10^{-2}	IN

Table A.1: Table showing Data Constraints

Variable	Type	Unit	Physical Constraints	System Constraints	Typical value	Property
λ_{14}	Real	s^{-1}	$-\infty < \lambda_{14} < \infty$		$0.169*10^{-1}$	IN
λ_{15}	Real	s^{-1}	$-\infty < \lambda_{15} < \infty$		0.277	IN
$q'_{N,\max}$	Real	$\frac{kW}{m}$	$35 \le q'_{N,\max} \le 65$		62.9540	IN
h_b	Real	$\frac{kW}{m^2 {}^oC}$	$-\infty < h_b < \infty$		50	IN
h_p	Real	$\frac{kW}{m^2 {}^oC}$	$-\infty < h_p < \infty$		10	IN
T_B	Real	${}^oC$	$-\infty < T_B < \infty$		305.0	IN
r_f	Real	m	$-\infty < r_f < \infty$		0.00612	IN
p_{dry}	Real	fp	$-\infty < p_{\text{dry}} < \infty$		1.176	IN
h_{dry}	Real	$\frac{kW}{m^2 {}^oC}$	$-\infty < h_{\text{dry}} < \infty$		2.0	IN

Table A.2: Table showing Data Constraints

A.4.2.7 System Behavior

This section gives a detailed description of the system's functionalities based on the information in the sections"Data Constraints" and "Instance Models". It formally specifies the flow of processing the data. That is, from getting the input, applying the models and producing the output. The responses to undesired situations such as the errors that are to be generated if the data constraints are not satisfied are also stated. The contents of this section are used in design and testing.

Step 1: Read data from the Input file

```
1. Read the driving transient data
     If the driving transient is reactivity, then
       Read the reactivity transient data ($\rho(t)$)
     Else if the driving transient is neutron flux, then
       Read the Neutron flux transient data ($N(t)$)
     Else
       Read Fuel power transient data ($q'_{\text{NFRAC}}(t)$)

2. Read the inputs
     Read the inputs $\Delta t, \tau_c, \tau_g, h_g, h_b, \rho 1, \rho 2, r_i$
     If the driving transient is $\rho(t)$, then
       Read $l^*$, $\beta_i$, $\gamma_i$
     If the driving transient is $\rho(t)$ or $N(t)$, then
       Read the inputs $\gamma_{h,i}$, $\lambda_i$
```

Step 2: Process/Output:

```
  For each time step,

1. If the driving transient is $\rho(t)$, then
     Determine reactivity based on current time
   Else if the driving transient is $N(t)$, then
     Determine neutron flux based on current time
   Else if the driving transient is $q'_{\text{NFRAC}}(t)$, then
     Determine fuel power based on current time

2. If the driving transient is $\rho(t)$, then
```

(a) Generate the Neutron flux solving Point kinetics equations from IM5

(b) Output Neutron flux (N)

Else if the driving transient is $\rho(t)$ or $N(t)$, then

(a) Generate the relative fuel power by solving Decay heat equations from IM6

(b) Output the relative fuelpower (q'_{NFRAC})

Else if the driving transient is $\rho(t)$ or $N(t)$ or $q'_{\text{NFRAC}}(t)$, then

(a) Change the relative fuel power to linear element power using DD25

(b) Use the linear element power to determine average fuel temperature, average clad temperature and centerline temperature by solving IM1, IM3 ,IM2 respectively

(c) Output T_1, T_2, T_{CL}

(d) Use T_1, T_2 to determine Surface temperature (T_S) by solving DD23

(e) Output T_S

(f) Using the generated T_1, find the fuel stored energy ($\Delta H(T_1)$) using DD2 and the power to the coolant (q'_{out}) using DD27

(g) Find the fuel power (f_p) using DD3
If Metal water reaction's calculations are desired, then

(h) Read p_{dry}, h_{dry}, δ_{ox}

(i) Calculate the Metal water reaction heat using DD5 and DD26

A.4.3 Nonfunctional Requirements

This section specifies system requirements that consider the quality and behaviour of the system as a whole. It provides different specifications for the system, so that it is found acceptable and pleasant to use.These include: Accuracy/performance requirements, maintainability requirements.

A.4.3.1 Accuracy

The relative error between FP code and HOTSPOT code for the test cases specified in the FP Engineer's manual should not be more than 0.05

A.4.3.2 Maintainability

The effort put in maintaining the product should be less than 1/4th of the amount of efforts put in building, developing the software.

A.4.3.3 Solution Validation Strategies

This section establishes the strategies for validating the software product, and the specific tests to be performed to assert it complies with the requirements specification defined in the previous section. To validate the solution produced by the software,

1. Results of the FP code are compared with those of the HOTSPOT code for a test transient.

- An idealized transient involving severe overpower, combined with step changes in coolant temperature and an order of magnitude reduction in convective heat transfer to coolant, should be run with both codes.
- Runs are to performed both with and without average flux depression factor.

A.5 Other System Issues

This section provides additional information on the side" about FP

A.5.1 Open Issues

None present

A.5.2 Off-the-Shelf Solutions

None present

A.5.3 Waiting Room

None Present

A.6 Traceability Matrix

The purpose of this matrix is to provide an easy reference on what has to be additionally modified if a certain component is changed. Every time a component is changed, then the items in the column of that component which cross into an "X" should be modified as well.

NOTE: Traceability Matrix of a subset of the components is developed to keep the matrix fit in one page. The references between the other items would be documented in a similar manner. Building a tool to automatically generate the graphical representation of the matrix by scanning the labels and references can be a future work.

	T1	T2	T3	A1	A2	A3	GD1	GD2	GD3	GD4	GD5	DD6	DD7	DD8	DD9
GD1															
GD2				X	X		X								
GD3															
GD4	X							X							
GD5		X													
GD6		X													
DD1															
DD2															
DD3															
DD4															
DD5															
DD6								X	X						
DD7									X						
DD8									X		X				
DD9									X		X				
DD10									X						
DD11												X			
DD12									X						
DD13								X	X						
DD14															
DD15															
DD16															
DD17															
Dd18													X		X
DD19													X	X	
DD20															
DD21															
DD22															
DD23						X			X						
DD24															
DD25															
DD26															
DD27									X						
DD28															X
IM1										X					
IM2										X					
IM3										X					
IM5			X												

Table A.3: Traceability Matrix Showing the Connections Between Items of Different Sections

A.7 Auxillary Constants

TB1 Metal Water Reaction

Constant	Value	Constraint	Units
A	6.48×10^{-8}	$T_2 \leq 1580^oC$	-
	1.0×10^{-6}	$T_2 > 1580^oC$	-
B/R	13586.0	$T_2 \leq 1580^oC$	-
	16014.0	$T_2 > 1580^oC$	-
q_r	6500.0	-	$\frac{\text{kJ}}{\text{kg}}$

TB2 Clad Conductivity

Constant	Value	Constraint	Units
a	1.43×10^{-5}	$T_2 \leq 1000^oC$	-
	2.73×10^{-5}	$T_2 > 1000^oC$	-
b	1.17×10^{-2}	$T_2 \leq 1000^oC$	-
	-1.27×10^{-3}	$T_2 > 1000^oC$	-

TB3 Coefficients for polynomial $K_{\mathbf{AV}}$

Coefficient	Value	Constraint	Units
x_0			
x_1			

TB4 Coefficients for polynomial $c_{p,1}$

Coefficient	Value	Constraint	Units
y_0			
y_1			
y_2			

TB5 Fuel Stored Energy

Constant	Value	Constraint	Units
K_0	15.496	-	-
K_1	19.145	-	$\frac{\text{cal}}{\text{moleK}}$
K_2	7.84733×10^{-4}	-	$\frac{\text{cal}}{\text{moleK}^2}$
K_3	5.64373×10^{6}	-	$\frac{\text{cal}}{\text{mole}}$
θ	535.285	-	K
E_D/R	1.8971×10^{4}	-	K

Appendix B

LP Manual

B.1 Overview

Given relative fuel power (q'_{NFRAC}) as input, the function *fuel_temp_* calculates the following:

1. Average fuel temperature (T_1),
2. Average Clad temperature (T_2),
3. Centerline temperature (T_{CL}),
4. Surface temperature (T_S), and
5. Stored fuel energy $(\Delta H(T_{\text{abs}}))$.
6. Integrated fuel power (f_p).
7. Power to the coolant (q'_{out}).
8. Metal water reaction heat (q'_{MWR}).

This function uses

- the material properties of the clad and fuelpellet
- lumped parameter methods and
- initial conditions

It solves the ODEs from the SRS given by IM1, IM3 and IM2 with the initial conditions defined in IM4, which are summarized below:

$$C_1 \frac{dT_1}{dt} = q'_N - \frac{T_1 - T_2}{R_1} \tag{B.1}$$

$$C_2 \frac{dT_2}{dt} = \frac{T_1 - T_2}{R_1} + q'_{\text{MWR}} - \frac{T_2 - T_B}{R_2} \tag{B.2}$$

$$C_{\text{CL}} \frac{dT_{\text{CL}}}{dt} = q'_N - \frac{T_{\text{CL}} - T_1}{R_{\text{CL}}} \tag{B.3}$$

where

- T_B is the coolant temperature
- q'_N is the linear element power (kW/m)
- q'_{MWR} is metal-water reaction heat (kW/m)
- C_1 is the thermal capacitance of the fuel (kWs/(m°C))
- C_2 is the thermal capacitance of the clad (kWs/(m°C))
- C_{CL} is the thermal capacitance at the centerline (kWs/(m°C))
- R_1 is the effective thermal resistance between T_1 and T_2 (m°C/kW)
- R_2 is the effective thermal resistance between coolant and half of the clad (m°C/kW)
- R_{CL} is the thermal resistance between T_{CL} and T_1 (m°C/kW)
- t is time

B.2 Numerical Algorithm

For solving the instance models of the SRS, an explicit method has been used because smaller time steps were used while predicting transient temperatures and because explicit methods are easy to understand during verification by a human reader.
Equations B.1–B.3 are of the form:

$$\frac{dx}{dt} = a(x)x + b(x)u(x,t), \tag{B.4}$$

where x is the variable under consideration. Taking the Laplace transform of Equation B.4,

$$x(s) = \frac{x_0}{s-a} + \frac{bu}{s(s-a)} \tag{B.5}$$

with x_0 being $x(t_0)$. To obtain the closed form solution, $a(x)$, $b(x)$ and $u(x,t)$ are assumed to be constant in the interval $[t, t+\Delta t]$. The solution to the above ODE is found by taking the inverse Laplace transform of Equation B.5

$$x(t) = x_o e^{-at} + bu \int_0^t e^{-at} dt \tag{B.6}$$

Denoting an approximation of $x(t_k)$ at t_k by x_k and denoting $\Delta t = t_{k+1} - t_k$, $k \geq 0$, we have:

$$x_{k+1} = x_k e^{-a_k \Delta t} + (1 - e^{-a_k \Delta t}) \frac{-b(x_k)u(x_k,t_k)}{a(x_k)} \tag{B.7}$$

The following table summarizes the values of $a(x)$, $b(x)$ and $u(x,t)$ for Equations B.1, B.2 and B.3:

Equation	x	$a(x)$	$b(x)$	$u(x,t)$
B.1	T_1	$-\frac{1}{R_1C_1}$	$\frac{T_2+q'_NR_1}{R_1C_1}$	1
B.2	T_2	$-\frac{(R_1+R_2)}{R_1R_2C_2}$	$\frac{T_1R_2+q'_{\text{MWR}}R_1R_2+T_BR_1}{R_1R_2C_2}$	1
B.3	T_{CL}	$-\frac{1}{R_3C_{\text{CL}}}$	$\frac{T_1+q'_NR_3}{R_3C_{\text{CL}}}$	1

Table B.1: Table of functions for ODEs representing different instance models

B.3 Algorithm

fuel_temp_$(\Delta t, q_{\text{NFRAC},k}, q'_{N_{\max}}, r_f, f, \rho_1, \rho_2, h_{ib}, h_p, \tau_g, \tau_c, T_b, p_{\text{dry}}, h_{\text{dry}}, \text{time}, init\_flag, MW\_flag, n,$ $h_{b,k}, q'_{N,k}, k_{c,k}, k_{\text{AV},k}, q'_{\text{MWR},k}, f_{p,k}, T_{1,k}, T_{2,k}, T_{\text{CL},k}, T_{S,k}, h_{c,k}, h_{g,k}, C_{1,k}, C_{2,k}, C_{\text{CL},k}, c_{p,1,k}, c_{p,2,k}, c_{p,3,k}, \Delta H(T_{\text{abs},k}),$ $\delta_{\text{ox},k}, R_{\text{ox},k}, q'_{\text{out},k}, q'_{\text{MWRI},k})$

1. Initialization section ($*init\_flag == 1$):
   - Input: Δt, $q_{\text{NFRAC},0}$, $q'_{N_{\max}}$,r_f, f, ρ_1, ρ_2, h_{ib}, h_p, τ_g, τ_c, T_b, *init_flag*.
   - At t_0 compute y_0
   - Output: y_0,

   where $y_0 = \{h_b, q'_N, k_c, k_{\text{AV}}, q'_{\text{MWR}}, f_p, T_1, T_2, T_{\text{CL}}, T_S, C_1, C_2, C_{\text{CL}}, c_{p,1}, c_{p,2}, c_{p,3}, h_c, h_g,$ $\Delta H(T_{\text{abs}}), \delta_{\text{ox}}, R_{\text{ox}}, q'_{\text{out}}, q'_{\text{MWRI}}\}$.
   All elements of the set y_0 are evaluated at the 0th time step.

2. Dynamic section ($*init\_flag == 0$):

   At t_{k+1}, $k \geq 0$,

   - Input:Δt, $q_{\text{NFRAC},k+1}$, $q'_{N_{\max}}$,r_f, f, ρ_1, ρ_2, h_{ib}, h_p, τ_g, τ_c, T_b, p_{dry}, h_{dry}, time, *init_flag*, *MW_flag*, n, y_k.
   - compute y_{k+1}, update n when necessary.
   - Output: y_{k+1}, n,

   where $y_{k+1} = \{h_b, q'_N, k_c, k_{\text{AV}}, q'_{\text{MWR}}, f_p, T_1, T_2, T_{\text{CL}}, T_S, C_1, C_2, C_{\text{CL}}, c_{p,1}, c_{p,2}, c_{p,3}, h_c, h_g,$ $\Delta H(T_{\text{abs}}), \delta_{\text{ox}}, R_{\text{ox}}, q'_{\text{out}}, q'_{\text{MWRI}}\}$.
   All elements of the set y_{k+1} are evaluated at the $k+1$th time step.

B.4 Overall function

```
2  ⟨fuel temp function 2⟩ ≡
   void calpro_(float *t, int *i, int *iflag, float *prpval, int *icnt);
   void dryout_(float *htout, float *time);
   void fuel_temp_(const float *delta, float *q_NFRAC, float *q_Nmax, float *r_f, float
           *f, float *rho_1, float *rho_2, float *h_ib, float *h_p, float *tau_g, float
           *tau_c, float *t_b, float *p_dry, float *h_dry, float *time, short int
           *init_flag, int *MW_flag, int *n, float *h_b, float *q_N, float *k_c, float
           *k_AV, float *q_MWR, float *f_p, float *t_1, float *t_2, float *t_CL, float
           *t_S, float *h_c, float *h_g, float *c_1, float *c_2, float *c_CL, float
           *c_p1, float *c_p2, float *c_p3, float *deltaHT_abs, float *delta_ox, float
           *rate_ox, float *q_out, float *q_MWRI)
   {
     if (*init_flag) { ⟨initialization section 15⟩}
     else { ⟨dynamic section 53⟩}
   }
```

This code is used in chunk 94

The following sections show the connections between the theory and the numerical algorithm to the implementation.

B.5 Naming Conventions

Input Parameters

3 **void** *fuel_temp_*(**const float** ∗*delta*, **float** ∗*q_NFRAC*, **float** ∗*q_Nmax*, **float** ∗*r_f*, **float** ∗*f*, **float** ∗*rho_1*, **float** ∗*rho_2*, **float** ∗*h_ib*, **float** ∗*h_p*, **float** ∗*tau_g*, **float** ∗*tau_c*, **float** ∗*t_b*, **float** ∗*p_dry*, **float** ∗*h_dry*, **float** ∗*time*, **short int** ∗*init_flag*, **int** ∗*MW_flag*, **int** ∗*n*, **float** ∗*h_b*, **float** ∗*q_N*, **float** ∗*k_c*, **float** ∗*k_AV*, **float** ∗*q_MWR*, **float** ∗*f_p*, **float** ∗*t_1*, **float** ∗*t_2*, **float** ∗*t_CL*, **float** ∗*t_S*, **float** ∗*h_c*, **float** ∗*h_g*, **float** ∗*c_1*, **float** ∗*c_2*, **float** ∗*c_CL*, **float** ∗*c_p1*, **float** ∗*c_p2*, **float** ∗*c_p3*, **float** ∗*deltaHT_abs*, **float** ∗*delta_ox*, **float** ∗*rate_ox*, **float** ∗*q_out*, **float** ∗*q_MWR1*)

On input,

parameter	stores
∗*delta*	Δt
∗*q_NFRAC*	q'_{NFRAC}
∗*q_Nmax*	$q'_{N_{\max}}$
∗*r_f*	r_f
∗*f*	f
∗*rho_1*	ρ_1
∗*rho_2*	ρ_2
∗*h_ib*	h_{ib}
∗*h_p*	h_p
∗*tau_g*	τ_g
∗*tau_c*	τ_c
∗*t_b*	T_B
∗*p_dry*	p_{dry}
∗*h_dry*	h_{dry}
∗*time*	time
∗*init_flag*	0 or 1
∗*MW_flag*	0 or 1
∗*n*	1 or 2
∗*h_b*	h_{ib} or h_{dry}
∗*q_N*	$q'_{N,k}$, $k \geq 0$, if $\neg$∗*init_flag*
∗*k_c*	$k_{c,k}$, $k \geq 0$, if $\neg$∗*init_flag*
∗*k_AV*	$k_{\mathrm{AV,k}}$, $k \geq 0$, if $\neg$∗*init_flag*
∗*q_MWR*	$q'_{\mathrm{MWR},k}$, $k \geq 0$, if $\neg$∗*init_flag*
∗*f_p*	$P_{\mathrm{F,SUM},k}$ $k \geq 0$, if $\neg$∗*init_flag*
∗*t_1*	$T_{1,k}$, $k \geq 0$, if $\neg$∗*init_flag*
∗*t_2*	$T_{2,k}$, $k \geq 0$, if $\neg$∗*init_flag*

parameter	stores
**t_CL*	$T_{\mathrm{CL},k}$, $k \geq 0$, if $\neg *init\_flag$
**t_S*	$T_{S,k}$, $k \geq 0$, if $\neg *init\_flag$
**h_c*	$h_{c,k}$, $k \geq 0$, if $\neg *init\_flag$
**h_g*	$h_{g,k}$, $k \geq 0$, if $\neg *init\_flag$
**c_1*	$C_{1,k}$, $k \geq 0$, if $\neg *init\_flag$
**c_2*	$C_{2,k}$, $k \geq 0$, if $\neg *init\_flag$
**c_CL*	$C_{\mathrm{CL},k}$, $k \geq 0$, if $\neg *init\_flag$
**c_p1*	$c_{p,1,k}$, $k \geq 0$, if $\neg *init\_flag$
**c_p2*	$c_{p,2,k}$, $k \geq 0$, if $\neg *init\_flag$
**c_p3*	$c_{p,3,k}$, $k \geq 0$, if $\neg *init\_flag$
**deltaHT_abs*	$\Delta H(T_{\mathrm{abs},k})$, $k \geq 0$, if $\neg *init\_flag$
**delta_ox*	$\delta_{\mathrm{ox},k}$, $k \geq 0$, if $\neg *init\_flag$
**rate_ox*	$R_{\mathrm{ox},k}$ $k \geq 0$, if $\neg *init\_flag$
**q_out*	$q'_{\mathrm{out},k}$ $k \geq 0$, if $\neg *init\_flag$
**q_MWRI*	$q'_{\mathrm{MWRI},k}$, $k \geq 0$, if $\neg *init\_flag$

For $*init\_flag = 1$, that is, when time step $k = 0$, all the input variables with subscript k can have any value, as they are not used in any calculations during the initialization.

Output Parameters from the Initialization section

5 **void** *fuel_temp_*(**const float** $*$*delta*, **float** $*$*q_NFRAC*, **float** $*$*q_Nmax*, **float** $*$*r_f*, **float** $*$*f*, **float** $*$*rho_1*, **float** $*$*rho_2*, **float** $*$*h_ib*, **float** $*$*h_p*, **float** $*$*tau_g*, **float** $*$*tau_c*, **float** $*$*t_b*, **float** $*$*p_dry*, **float** $*$*h_dry*, **float** $*$*time*, **short int** $*$*init_flag*, **int** $*$*MW_flag*, **int** $*$*n*, **float** $*$*h_b*, **float** $*$*q_N*, **float** $*$*k_c*, **float** $*$*k_AV*, **float** $*$*q_MWR*, **float** $*$*f_p*, **float** $*$*t_1*, **float** $*$*t_2*, **float** $*$*t_CL*, **float** $*$*t_S*, **float** $*$*h_c*, **float** $*$*h_g*, **float** $*$*c_1*, **float** $*$*c_2*, **float** $*$*c_CL*, **float** $*$*c_p1*, **float** $*$*c_p2*, **float** $*$*c_p3*, **float** $*$*deltaHT_abs*, **float** $*$*delta_ox*, **float** $*$*rate_ox*, **float** $*$*q_out*, **float** $*$*q_MWRI*)

On output,
if $*$*init_flag* $\equiv 1$,

parameter	stores
$*$*n*	1
$*$*h_b*	h_{ib}
$*$*q_N*	$q'_{N,0}$
$*$*k_c*	$k_{c,0}$
$*$*k_AV*	$k_{\mathrm{AV},0}$
$*$*q_MWR*	$q'_{\mathrm{MWR},0}$
$*$*f_p*	$P_{\mathrm{F,SUM},0}$
$*$*t_1*	$T_{1,0}$
$*$*t_2*	$T_{2,0}$
$*$*t_CL*	$T_{\mathrm{CL},0}$
$*$*t_S*	$T_{S,0}$
$*$*h_c*	$h_{c,0}$
$*$*h_g*	$h_{g,0}$
$*$*c_1*	$C_{1,0}$
$*$*c_2*	$C_{2,0}$
$*$*c_CL*	$C_{\mathrm{CL},0}$
$*$*c_p1*	$c_{p,1,0}$
$*$*c_p2*	$c_{p,2,0}$
$*$*c_p3*	$c_{p,3,0}$
$*$*deltaHT_abs*	$\Delta H(T_{\mathrm{abs},0})$
$*$*delta_ox*	$\delta_{\mathrm{ox},0}$
$*$*rate_ox*	$R_{\mathrm{ox},0}$
$*$*q_out*	$q'_{\mathrm{out},0}$
$*$*q_MWRI*	$q'_{\mathrm{MWRI},0}$

Output Parameters from the Dynamic section

8

On output,

If $\neg *$*init_flag*,

parameter	stores
$*$*n*	1 or 2
$*$*h_b*	h_{ib} or h_{dry}
$*$*q_N*	$q'_{N,k+1}$
$*$*k_c*	$k_{c,k+1}$
$*$*k_AV*	$k_{\text{AV},k+1}$
$*$*q_MWR*	$q'_{\text{MWR},k+1}$
$*$*f_p*	$P_{\text{F,SUM},k+1}$
$*$*t_1*	$T_{1,k+1}$
$*$*t_2*	$T_{2,k+1}$
$*$*t_CL*	$T_{\text{CL},k+1}$
$*$*t_S*	$T_{S,k+1}$
$*$*h_c*	$h_{c,k+1}$
$*$*h_g*	$h_{g,k+1}$
$*$*c_1*	$C_{1,k+1}$
$*$*c_2*	$C_{2,k+1}$
$*$*c_CL*	$C_{\text{CL},k+1}$
$*$*c_p1*	$c_{p,1,k+1}$
$*$*c_p2*	$c_{p,2,k+1}$
$*$*c_p3*	$c_{p,3,k+1}$
$*$*deltaHT_abs*	$\Delta H(T_{\text{abs},k+1})$
$*$*delta_ox*	$\delta_{\text{ox},k+1}$
$*$*rate_ox*	$R_{\text{ox},k+1}$
$*$*q_out*	$q'_{\text{out},k+1}$
$*$*q_MWRI*	$q'_{\text{MWRI},k+1}$

NOTE: The *fuel_temp_* function calls two fuelpin15.f functions- '*calpro_*' which calculates material properties and '*dryout_*' which outputs a message when dryout occurs. The interfaces for these functions are not specified in this document, but the relevant terms that they define are explained as they arise in the documentation.

Local Variables for the Effective thermal resistance in the Initialization section

11

parameter	stores
r_1	$R_{1,0}$
r_2	$R_{2,0}$
r_3	$R_{3,0}$
r_fuel	$R_{\text{FUEL},0}$

Local Variables for the Effective thermal resistance in the Dynamic section

13

parameter	stores
r_1	$R_{1,k+1}$
r_2	$R_{2,k+1}$
r_3	$R_{3,k+1}$
r_CL	$R_{\text{CL},k+1}$

B.6 Initialization section

In this section, we determine initial values (subscript $k = 0$) for:

$h_b, q'_N, k_c, k_{\text{AV}}, q'_{\text{MWR}}, f_p, T_1, T_2, T_{\text{CL}}, T_S, h_c, h_g, C_1, C_2, C_{\text{CL}}, c_{p,1}, c_{p,2}, c_{p,3}, \Delta H(T_{\text{abs}}), \delta_{\text{ox}}, R_{\text{ox}}, q'_{\text{out}}, q'_{\text{MWRI}}$

```
15  ⟨initialization section 15⟩ ≡
      ∗n = 1;
         /∗ n is used to keep track of the dryout output message in the dynamic section ∗/
      float pi = 3.1416;
      ⟨Calculation of $q'_N$ 17⟩;
      ⟨initialization of average clad temperature $T_{2,0}$ 18⟩;
      ⟨Calculation of $k_c$ 19⟩;
      ⟨Calculation of heat transfer coefficient ($h_c$) and the gap conductance ($h_g$) 21⟩;
      ⟨initialization of surface temperature ($T_{S,0}$) 22⟩;
      ⟨convergence routine to determine $k_{\text{AV},0}$ and $T_{\text{CL},0}$ 28⟩;
      ⟨Calculation of $R_1$ 30⟩;
      ⟨initialization of average fuel temperature $T_{1,0}$ 31⟩;
      ⟨declaration of constants for stored energy 32⟩;
      ⟨$\Delta H(T_{\text{abs}})$ 33⟩;
      ⟨Calpro function for $C_1$ and $c_{p,1}$ 35⟩;
      ⟨Calculation of $C_1$ and $c_{p,1}$ 36⟩;
      icnt = 10;
         /∗ icnt is given as an argument to the calpro() function for calculating the specific
            heats and the integrals of polynomials∗/
      ⟨Calpro function for $C_2$ and $c_{p,2}$ 38⟩;
      ⟨Calculation of $C_2$ and $c_{p,2}$ 39⟩;
      ⟨Calpro function for $C_{\text{CL}}$ and $c_{p,3}$ 41⟩;
      ⟨Calculation of $C_{\text{CL}}$ and $c_{p,3}$ 42⟩;
      ⟨initialization of constants for $R_{\text{ox}}$ 44⟩;
      ⟨Calculation of $R_{\text{ox}}$ 45⟩;
      ⟨Calculation of $q'_{\text{MWR}}$ 46⟩;
      ⟨initialization of $q'_{\text{MWRI}}$ 47⟩;
      ⟨Calculation of $\delta_{\text{ox}}$ 48⟩;
      ⟨Calculation of $q'_{\text{out}}$ 50⟩;
      ⟨initialization of $f_{p,0}$ 51⟩;
```

This code is used in chunk 2

B.6.1 Computing q'_N, T_2 and k_c

The input relative fuel power (q'_{NFRAC}) is changed to linear element power (q'_N) by multiplying it with the initial linear element rating ($q'_{\text{N}_{\max}}$) as given by DD25 of the SRS.

$$q'_N = q'_{\text{NFRAC}} q'_{N_{\max}}; \tag{B.8}$$

This q'_N is used to determine the relevant temperatures for the fuelpin. We evaluate linear element power as

```
17  ⟨Calculation of $q'_N$ 17⟩ ≡
      ∗q_N = ∗q_NFRAC ∗ (∗q_Nmax);
```

This code is used in chunks 15 and 57

Now, we evaluate T_2 in steady state by first setting the time derivative term of Equation B.1 to zero as follows,

$$\frac{T_1 - T_2}{R_1} = q'_N \tag{B.9}$$

Next we set the time derivative term of Equation B.2 to zero and neglect the metal water heating term to get,

$$\frac{T_1 - T_2}{R_1} = \frac{T_2 - T_B}{R_2} \tag{B.10}$$

Substituting Equation B.9 in Equation B.10 and rearranging the equation, we get the steady state case as:

$$T_2 = T_B + q'_N R_2, \tag{B.11}$$

where R_2 is given by DD12 of the SRS as,

$$R_2 = \frac{1}{2\pi r_c h_c} \tag{B.12}$$

From DD18 of the SRS, we have the equation for h_c as,

$$h_c = \frac{2k_c h_b}{2k_c + \tau_c h_b} \tag{B.13}$$

Substituting Equation B.13 into Equation B.12, we get,

$$R_2 = \frac{1}{2\pi r_c \left(\frac{2k_c h_b}{2k_c + \tau_c h_b}\right)} \tag{B.14}$$

$$= \frac{1}{2\pi r_c}\left(\frac{2k_c + \tau_c h_b}{2k_c h_b}\right) \tag{B.15}$$

The above equation cannot be evaluated directly in steady state, because R_2 is dependent on T_2 through the clad conductivity (k_c) as given by DD15 of SRS. That is,

$$k_c = aT_2 + b, \tag{B.16}$$

where a and b are given constants obtained by a least squares fit to tabulated data. According to the Assumption A12 of the SRS, since T_2 is less than $1000^o C$ in the initial state, the values of a and b are given by the Table TB2 of the SRS as,

$$a = 1.43 \times 10^{-5} \tag{B.17}$$

$$b = 1.17 \times 10^{-2} \tag{B.18}$$

So, taking the expression for k_c from Equation B.16, substituting it into Equation B.15 gives

$$R_2 = \frac{2(aT_2 + b) + \tau_c h_b}{4\pi r_c h_b (aT_2 + b)}, \tag{B.19}$$

On further simplification, Equation B.19 becomes,

$$R_2 = \frac{2aT_2 + 2b + \tau_c h_b}{4\pi r_c h_b a T_2 + 4\pi r_c h_b b}, \tag{B.20}$$

where r_c is the outer clad radius and is obtained by the sum of fuel radius (r_f), gap thickness (τ_g) and clad thickness (τ_c).

$$r_c = r_f + \tau_g + \tau_c \tag{B.21}$$

Substituting Equation B.20 into Equation B.11 and rearranging gives an equation quadratic in T_2:

$$4\pi r_c h_b a T_2^2 + \left(4\pi r_c h_b b - 4\pi r_c h_b a T_B - 2aq'_N\right)T_2 - \left(4\pi r_c h_b T_B b + 2q'_N b + q'_N h_b \tau_c\right) = 0 \tag{B.22}$$

The above equation has to be solved to find the positive root which gives T_2 in steady state. Simultaneously the value k_c from Equation B.16 is also calculated.

18 ⟨initialization of average clad temperature $T_{2,0}$ 18⟩ ≡ /∗declaration of constants ∗/

```
float $a = 1.43 \cdot 10^{-05}$;
float $b = 1.17 \cdot 10^{-02}$;
    /∗ computation of clad radius B.21 ∗/
float $r\_c = *r\_f + *tau\_g + *tau\_c$;
    /∗ initializing coolant film conductance ∗/
$*h\_b = *h\_ib$;
    /∗ computation of T2 in steady state ∗/
float C10 $= 2.0 * pi * r\_c * (*h\_b)$;
float C11 $= 2.0 *$ C10 $* a$;
float C12 $=$ C10 $* (2.0 * b - (2.0 * a * (*t\_b))) - (*q\_N * 2.0 * a)$;
float C13 $= -$C10 $* (*t\_b) * 2.0 * b - *q\_N * (2.0 * b + ((*h\_b) * (*tau\_c)))$;
    /∗ solving quadratic equation ∗/
$*t\_2 = (-$C12 $+ sqrt($C12 $*$ C12 $- 4.0 *$ C11 $*$ C13$))/(2.0 *$ C11$)$;
    /∗computation of initial clad conductivity B.16 ∗/
```

This code is used in chunk 15

19 ⟨Calculation of k_c 19⟩ ≡

```
$*k\_c = a * (*t\_2) + b$;
```

This code is used in chunk 15

B.6.2 Computing h_c, h_g and T_S

Using this clad conductivity (k_c), we compute the heat transfer coefficient (h_c) and the gap conductance (h_g) as DD18 and DD19 of the SRS, respectively. That is,

$$h_c = \frac{2k_c h_b}{2k_c + \tau_c h_b}, \tag{B.23}$$

$$h_g = \frac{2k_c h_p}{2k_c + \tau_c h_p} \tag{B.24}$$

21 ⟨Calculation of heat transfer coefficient (h_c) and the gap conductance (h_g) 21⟩ ≡

```
    /∗ calculation of heat transfer coefficient ∗/
$*h\_c = (2 * (*k\_c) * (*h\_b))/((2 * (*k\_c)) + (*tau\_c * (*h\_b)))$;
    /∗ calculation of gap conductance ∗/
$*h\_g = (2 * (*k\_c) * (*h\_p))/((2 * (*k\_c)) + (*tau\_c * (*h\_p)))$;
```

This code is used in chunks 15 and 60

At each time step, the surface temperature (T_S) is calculated based on the clad and average fuel temperatures as given by DD23 of the SRS as:

$$T_S = T_2 + \frac{T_1 - T_2}{R_1} R_3, \tag{B.25}$$

where R_3 is calculated as given by DD10 of the SRS.

$$R_3 = \frac{1}{2\pi r_f h_g} \tag{B.26}$$

The surface temperature in steady state ($T_{S,0}$) is evaluated using $T_{2,0}$ and by setting Equation B.1 to zero as shown in Equation B.9. Substituting Equation B.9 in Equation B.25 gives the steady state case of T_S as:

$$T_{S,0} = T_{2,0} + q'_{N,0} R_{3,0}, \tag{B.27}$$

where $R_{3,0}$ is calculated based on $h_{g,0}$.

22 ⟨initialization of surface temperature ($T_{S,0}$) 22⟩ ≡ /* calculation of R_3 */
```
float r_3 = 1/(2 * pi * (*r_f) * (*h_g));
    /* calculation of T_{S,0} */
*t_S = *t_2 + (*q_N * r_3);
```
This code is used in chunk 15

B.6.3 Computing T_{CL} and k_{AV}

Given this T_S and q'_N, the centerline temperature (T_{CL}) is calculated as given by Equation A.50 of the SRS. That is, in steady state,

$$T_{\text{CL}} = T_S + R_{\text{FUEL}} q'_N, \tag{B.28}$$

where R_{FUEL} is given by DD6 of the SRS as,

$$R_{\text{FUEL}} = \frac{f}{4\pi k_{\text{AV}}}, \tag{B.29}$$

where f is the flux depression factor (constant obtained from the input file) and k_{AV} is the average fuel conductivity.

24 ⟨computation of T_{CL} 24⟩ ≡
```
float r_fuel;
r_fuel = *f/(4.0 * pi * (*k_AV));
*t_CL = *t_S + (r_fuel * (*q_N));
```
This code is used in chunk 28

The above computation requires the average thermal conductivity (k_{AV}), but this value is not initially known.. Since k_{AV} is a temperature-dependent, non-linear variable, an iterative procedure converging on mutually consistent values for k_{AV} and T_{CL} is needed. An initial estimate of k_{AV} ($k_{\text{AV,est}}$) fixes the $T_{\text{CL,est}}$. To update k_{AV}, we need an estimate of linear element power ($q'_{\text{N,est}}$) which is computed from the current $T_{\text{CL,est}}$. Rewriting Equation A.45 of the SRS in terms of q'_N by using DD1 of the SRS and taking flux depression factor into consideration gives,

$$\int_{T_{\text{CL}}}^{T_S} dT = \frac{-f q'_N}{2\pi r_f^2} \int_0^{r_f} \frac{r}{k} dr \tag{B.30}$$

Integrating the RHS we have,

$$\int_{T_{\mathrm{CL}}}^{T_S} dT = \frac{-fq'_N}{4\pi k} \tag{B.31}$$

Rearranging Equation B.31 and integrating the LHS of the equation from T_S to $T_{\mathrm{CL,est}}$ generates the estimate of linear element power ($q'_{\mathrm{N,est}}$) as,

$$q'_{\mathrm{N,est}} = 4\pi \int_{T_S}^{T_{\mathrm{CL,est}}} \frac{kdT}{f} \tag{B.32}$$

where k is a first order polynomial function of temperature and is given by DDL-pkav of the SRS as,

$$k = x_1 T + x_0 \tag{B.33}$$

Let

$$K(T) = \int kdT, \tag{B.34}$$

Hence,

$$q'_{\mathrm{N,est}} = \frac{4\pi}{f}[K(T)]_{T_S}^{T_{\mathrm{CL,est}}}, \tag{B.35}$$

$$= \frac{4\pi}{f}\left(K(T_{\mathrm{CL,est}}) - K(T_S)\right) \tag{B.36}$$

25 $\langle$estimation of q'_N 25$\rangle \equiv$
float *q_NEST*;
$q\_NEST = ((4.0 * pi) * (t\_e - t\_a))/(*f);$ /* t_e and t_a are $K(T_{\mathrm{CL,est}})$ and $K(T_S)$ respectively which are evaluated by calpro function */
This code is used in chunk 28

Substituting Equation B.29 in Equation B.28 and rearranging gives,

$$k_{\mathrm{AV}} = \frac{fq'_{\mathrm{N}}}{4\pi(T_{\mathrm{CL}} - T_S)} \tag{B.37}$$

The estimate of the element power from Equation B.36 is compared to the actual value and used to update k_{AV}. The relationship between k_{AV} and and q'_N is given by the first order Taylor series expansion of k_{AV} with respect to q'_N as,

$$k_{\mathrm{AV,i+1}} = k_{\mathrm{AV,i}} + \frac{dk_{\mathrm{AV}}}{dq'_{\mathrm{N}}}\Delta q'_{\mathrm{N}} \tag{B.38}$$

Differentiating Equation B.37 with respect to q'_{N} gives,

$$\frac{dk_{\mathrm{AV}}}{dq'_{\mathrm{N}}} = \frac{f}{4\pi(T_{\mathrm{CL}} - T_S)} \tag{B.39}$$

The change in q'_N ($\Delta q'_N$) is the difference between the estimated and actual values.

$$\Delta q'_{\mathrm{N}} = q'_{\mathrm{N,est,i}} - q'_N \tag{B.40}$$

Substituting Equation B.39 and Equation B.40 in Equation B.38,

$$k_{\mathrm{AV,i+1}} = k_{\mathrm{AV,i}} + \frac{(fq'_{\mathrm{N,est,i}} - fq'_N)}{4\pi(T_{\mathrm{CL}} - T_S)} \tag{B.41}$$

26 $\langle$update k_{AV} 26$\rangle \equiv$
$*k\_AV = *k\_AV + (((*f * q\_NEST - (*f * (*q\_N))))/(4.0 * pi * (*t\_CL - *t\_S)));$
This code is used in chunk 28

We compute the relative error (normalized difference between the actual and estimated values) of q'_N as a condition for convergence.

27 $\langle$relative error computation 27$\rangle \equiv$

```
$re = (*q\_N - q\_NEST)/(*q\_N);$
```

This code is used in chunk 28

Now we can put the above together to evaluate $k_{\mathrm{AV},0}$ and $T_{\mathrm{CL},0}$ using the described convergence routine.

28 $\langle$convergence routine to determine $k_{\mathrm{AV},0}$ and $T_{\mathrm{CL},0}$ 28$\rangle \equiv$

```
float t_a, t_e;
int i, iflag;
int icnt;      /* icnt is given as an argument to the calpro() function for calculating the
      specific heats and the integrals of polynomials */
i = 1;
iflag = 1;
icnt = 0;
float ts = *t_S;
calpro_(&ts, &i, &iflag, &t_a, &icnt);      /* function calpro evaluates $t_a$ which is the
      integral of polynomial for $k_{\mathrm{AV}}$ at $T_S$ $(K(T_S))$ */
int idnt = 4;
    /* initial estimate of $k_{AV}$ */
*k_AV = 0.00255;
    /* initial estimate of relative error for convergence */
float re;
do {
    ⟨computation of $T_{\mathrm{CL}}$ 24⟩;
        /* function calpro evaluates the integral of polynomial for $k$ $(t_e)$ at $T_{\mathrm{CL}}$ */
  float tcl = *t_CL;
  calpro_(&tcl, &i, &iflag, &t_e, &idnt);      /* function calpro evaluates $t_e$ which is
        the integral of polynomial for $k_{\mathrm{AV}}$ at $T_{\mathrm{CL}}$ $(K(T_{\mathrm{CL}}))$ */
  ⟨estimation of $q'_N$ 25⟩;
  ⟨relative error computation 27⟩;
  if (fabsf(re) $\leq$ 0.0004) break;
  ⟨update $k_{\mathrm{AV}}$ 26⟩;
} while (1);
```

This code is used in chunk 15

B.6.4 Computing T_1

With k_{AV} determined from the above routine, we can determine the average fuel temperature (T_1) by setting the time derivative term of Equation B.1 to zero. That is, in steady state,

$$T_1 = T_2 + q'_N R_1, \tag{B.42}$$

where R_1 is the effective thermal resistance between T_1 and T_2. The value of R_1 is given by DD11 of the SRS as:

$$R_1 = \frac{f}{8\pi k_{\mathrm{AV}}} + \frac{1}{2\pi r_f h_g} \tag{B.43}$$

30 $\langle$Calculation of R_1 30$\rangle \equiv$

```
float r_1 = (*f/(8 * pi * (*k_AV))) + (1/(2 * pi * (*r_f) * (*h_g)));
```

This code is used in chunks 15 and 62

31 $\langle$initialization of average fuel temperature $T_{1,0}$ 31$\rangle \equiv$
$*t\_1 = *t\_2 + (*q\_N * r\_1);$
This code is used in chunk 15

B.6.5 Computing $\Delta H(T_{\mathbf{abs}})$

Now we compute the stored fuel energy, which depends on the average fuel temperature. It is the change in fuel enthalpy from standard room temperature ($T_{std} = 298$K) to the absolute value of the average fuel temperature T_1 and is given by DD2 of the SRS as:

$$\Delta H(T_{\text{abs}}) = K_0\left(K_1\theta\left(\left(e^{\theta/T_{\text{abs}}}-1\right)^{-1} - \left(e^{\theta/T_{std}}-1\right)^{-1}\right) + K_2(T_{\text{abs}}^2 - T_{std}^2) + K_3 e^{-E_D/(R_D T_{\text{abs}})}\right), \tag{B.44}$$

where $K_0, K_1, K_2, K_3, \theta, E_D, R_D$ are constants whose values are given by the TB5 of SRS as:

Constant	Value	Units
K_0	15.496	-
K_1	19.145	cal/moleK
K_2	7.84733×10^{-4}	cal/(moleK^2)
K_3	5.64373×10^{6}	cal/mole
θ	535.285	K
E_D	37.6946×10^{3}	
R_D	1.987	

32 $\langle$declaration of constants for stored energy 32$\rangle \equiv$ /* declaration of constants */
float `K0` $= 15.49 \cdot 10^{-03}$;
float `K1` $= 19.145$;
float `K2` $= 7.84733 \cdot 10^{-04}$;
float `K3` $= 5.64373 \cdot 10^{06}$;
float `THETA` $= 535.285$;
float `E_D` $= 37.6946 \cdot 10^{03}$;
float `R_D` $= 1.987$;
float $t\_std = 298$;
This code is used in chunks 15 and 75

Evaluation of the stored energy

33 $\langle \Delta H(T_{\text{abs}})$ 33$\rangle \equiv$
float t_abs;
$t\_abs = *t\_1 + 273.0;$
$*deltaHT\_abs = \texttt{K0} * (\texttt{K1} * \texttt{THETA} * ((1/(exp(\texttt{THETA}/t\_abs) - 1)) - (1/(exp(\texttt{THETA}/t\_std) - 1))) + \texttt{K2} * (t\_abs * t\_abs - t\_std * t\_std) + \texttt{K3} * exp(-\texttt{E\_D}/(\texttt{R\_D} * t\_abs)));$
This code is used in chunks 15 and 75

B.6.6 Computing C_1, $c_{p,1}$

We initialize the thermal capacitances C_1, C_2, C_{CL} which will be used later in determining the transient temperatures in the dynamic section.
C_1 is the thermal capacitance of the fuel ($\frac{\text{kWs}}{\text{m}^o\text{C}}$) and is given by DD14 of the SRS as,

$$C_1 = \pi r_f^2 \rho_1 c_{p,1}, \tag{B.45}$$

where
ρ_1 is the fuel density ($\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$)
r_f is the fuel radius (m)
$c_{p,1}$ is the specific heat corresponding to the fuel average temperature ($\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$)
$c_{p,1}$ is represented as a second order polynomial function of temperature and is given by DD17 of the SRS as,

$$c_{p,1} = y_2T^2 + y_1T + y_0 \tag{B.46}$$

The average value of $c_{p,1}$ is explicitly obtained by finding the average $c_{p,1}$ by integrating Equation B.46 from T_S to $T_{\text{CL,est}}$ and dividing by $T_{\text{CL}} - T_S$. That is,

$$c_{p,1_{\text{AV}}} = \frac{1}{T_{\text{CL}} - T_S}\int_{T_S}^{T_{\text{CL}}} c_{p,1}dT \tag{B.47}$$

Let

$$C_p(T) = \int c_{p,1}dT, \tag{B.48}$$

Hence,

$$c_{p,1_{\text{AV}}} = \frac{1}{T_{\text{CL}} - T_S}[C_p(T)]_{T_S}^{T_{\text{CL}}}, \tag{B.49}$$

$$= \frac{\left(C_p(T_{\text{CL}}) - C_p(T_S)\right)}{T_{\text{CL}} - T_S} \tag{B.50}$$

35 ⟨Calpro function for C_1 and $c_{p,1}$ 35⟩ ≡
```
float t_c, t_d;
    /* function calpro evaluates $C_p(T_S)$ */
int j = 2;
int jflag = 3;
ts = *t_S;
float tcl = *t_CL;
calpro_(&ts, &j, &jflag, &t_c, &idnt);
    /*function calpro evaluates $C_p(T_{CL})$ */
calpro_(&tcl, &j, &jflag, &t_d, &idnt);
```
This code is used in chunk 15

36 ⟨Calculation of C_1 and $c_{p,1}$ 36⟩ ≡   /* calculation of specific heat of the fuel */
```
*c_p1 = (t_d - t_c)/(*t_CL - *t_S);
    /* calculation of C1 */
*c_1 = pi * (*r_f) * (*r_f) * (*rho_1) * (*c_p1);
```
This code is used in chunks 15 and 78

B.6.7 Computing C_2, $c_{p,2}$

C_2 is the thermal capacitance of the clad ($\frac{\text{kWsec}}{\text{m}^{\text{o}}\text{C}}$) and is given by DD14 of SRS as,

$$C_2 = 2\pi r_c \tau_c \rho_2 c_{p,2}, \tag{B.51}$$

where r_c is the outer clad radius (m)
τ_c is the clad thickness (m)
$c_{p,2}$ is the specific heat corresponding to the clad temperature ($\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$)
ρ_2 is the clad density ($\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$)
We evaluate capacitance C_2 for T_2 as:

38 $\langle$Calpro function for C_2 and $c_{p,2}$ 38$\rangle \equiv$
```
int k = 3;
int kflag = 2;
    /* function calpro evaluates the specific heat of the clad ($c_{p,2}$) at $T_2$ */
float t2 = *t_2;
float cp2;
calpro_(&t2, &k, &kflag, &cp2, &idnt);
```
This code is used in chunk 15

39 $\langle$Calculation of C_2 and $c_{p,2}$ 39$\rangle \equiv$
```
*c_p2 = cp2;
    /* calculation of C2 */
*c_2 = 2 * pi * r_c * (*tau_c) * (*rho_2) * (*c_p2);
```
This code is used in chunks 15 and 79

B.6.8 Computing C_{CL}, $c_{p,3}$

C_{CL} is the thermal capacitance at the centerline ($\frac{\text{kWs}}{\text{m}^{\text{o}}\text{C}}$) and is given by DD14 of SRS as,

$$C_{\text{CL}} = \pi r_f^2 c_{p,3} \rho_1, \tag{B.52}$$

where r_f is the fuel radius(m)
$c_{p,3}$ is the specific heat corresponding to the fuel centreline temperature ($\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$).
ρ_1 is the fuel density ($\frac{\text{kJ}}{\text{kg}^{\text{o}}\text{C}}$).

We evaluate capacitance C_{CL} for T_{CL} as:

41 $\langle$Calpro function for C_{CL} and $c_{p,3}$ 41$\rangle \equiv$
```
    /* function calpro evaluates the specific heat at the centerline ($c_{p,3}$) at $T_{CL}$ */
int l = 2;
int lflag = 2;
tcl = *t_CL;
float cp3;
calpro_(&tcl, &l, &lflag, &cp3, &idnt);
```
This code is used in chunk 15

42 $\langle$Calculation of C_{CL} and $c_{p,3}$ 42$\rangle \equiv$
```
*c_p3 = cp3;
    /* calculation of $C_{CL}$ */
*c_CL = pi * (*r_f) * (*r_f) * (*rho_1) * (*c_p3);
```
This code is used in chunks 15 and 80

B.6.9 Computing δ_{ox}, R_{ox} and q'_{MWR}

The zircaloy clad material oxidizes exothermically when exposed to high temperature steam, resulting in additional heat input (q'_{MWR}) to the clad. The rate of oxidization (R_{ox}) depends on the average clad temperature (T_2) and the thickness of the reacted zircaloy (δ_{ox}) and is given by DD5 of the SRS as,

$$R_{\text{ox}} = \frac{A}{1.56\delta_{ox}} e^{\frac{-B}{R(T_2+273)}}, \tag{B.53}$$

where the values of constants A, B/R are given by Table TB1 of the SRS. According to the Assumption A12 of the SRS, since T_2 is less than 1000^oC in the initial state, the values of A and B/R are given as,

$$A = 6.48 \times 10^{-8} \tag{B.54}$$

$$B/R = 13586.0 \tag{B.55}$$

The thickness of the reacted zircaloy (δ_{ox}) is initialized to 1.0×10^{-6}.

44 $\langle$initialization of constants for R_{ox} 44$\rangle \equiv$ $/*$ initialization of $\delta_{\text{ox},0}$ $*/$
$*delta\_ox = 1.0 \cdot 10^{-06}$;
$/*$ initialization of constants A and B/R $*/$
float $A = 6.48 \cdot 10^{-08}$;
float $BbyR = 13586.0$;
This code is used in chunk 15

45 $\langle$Calculation of R_{ox} 45$\rangle \equiv$
$*rate\_ox = (A/(1.56 * (*delta\_ox))) * exp(-(BbyR)/(*t\_2 + 273.0))$;
This code is used in chunks 15 and 86

Now using this R_{ox}, the metal water reaction heat (q'_{MWR}) can be calculated as given by DD5 of the SRS.

$$q'_{\text{MWR}} = R_{\text{ox}} 2\pi r_c \rho_2 q_r, \tag{B.56}$$

where q_r is the heat of reaction and its value (6500.0) is given by Table TB1 of the SRS. The integrated metal water heat (q'_{MWRI}) is initialized to zero.

46 $\langle$Calculation of q'_{MWR} 46$\rangle \equiv$
float $q\_r = 6500.0$;
$*q\_MWR = *rate\_ox * 2 * pi * r\_c * (*rho\_2) * q\_r$;
This code is used in chunks 15 and 88

47 $\langle$initialization of q'_{MWRI} 47$\rangle \equiv$
$*q\_MWRI = 0.0$;
This code is used in chunk 15

As the reaction takes place, the clad material is oxidized and the thickness of the reacted zircaloy clad material is found by using Euler's method for solving an ODE.

$$\delta_{\text{ox},i+1} = \delta_{\text{ox},i} + \frac{d\delta_{\text{ox}}}{dt}\Delta t \tag{B.57}$$

Since the derivative of oxidized material with respect to time is rate of oxidization, that is,

$$\frac{d\delta_{\text{ox}}}{dt} = R_{\text{ox}} \tag{B.58}$$

Substituting (B.58) in (B.57),

$$\delta_{\text{ox},i+1} = \delta_{\text{ox},i} + R_{\text{ox}}\Delta t \tag{B.59}$$

48 $\langle$Calculation of δ_{ox} 48$\rangle \equiv$
$*delta\_ox = *delta\_ox + *rate\_ox * (*delta)$;
This code is used in chunks 15 and 90

B.6.10 Computing q'_{out} and initializing f_p

The output heat from the reaction is sent into the coolant. This heat to the coolant which is given by DD27 of the SRS is normalized by $q'_{N_{\max}}$ for easier understanding and comparission purposes. In other words the normalization is done since this is a standard form for presenting this information. Hence, the heat out is given as,

$$q'_{\text{out}} = \frac{1}{q'_{N_{\max}}}\left(\frac{T_2 - T_B}{R_2}\right), \tag{B.60}$$

where R_2 is the effective resistance between coolant film and the clad and is given by DD12 of the SRS as,

$$R_2 = \frac{1}{2\pi r_c h_c} \tag{B.61}$$

50 ⟨Calculation of q'_{out} 50⟩ ≡
```
  float r_2 = 1/(2 * pi * r_c * (*h_c));
  *q_out = (*t_2 - *t_b)/(r_2 * (*q_Nmax));
```
This code is used in chunks 15 and 84

The Integrated fuel power (f_p) as given by DD3 of the SRS, is a summation of the fuel powers at each time step. At t_0, no reaction takes place and hence no fuel power is generated. So, initially the integrated fuel power is set to zero.

51 ⟨initialization of $f_{p,0}$ 51⟩ ≡
```
  *f_p = 0.0;
```
This code is used in chunk 15

B.7 Dynamic section

In this section, we determine transient values (subscript $k > 0$) for

$q'_N, k_c, k_{\text{AV}}, q'_{\text{MWR}}, f_p, T_1, T_2, T_{\text{CL}}, T_S, h_c, h_g, C_1, C_2, C_{\text{CL}}, c_{p,1}, c_{p,2}, c_{p,3}, \Delta H(T_{\text{abs}}), \delta_{\text{ox}}, R_{\text{ox}}, q'_{\text{out}}, q'_{\text{MWRI}}$

53 ⟨dynamic section 53⟩ ≡
```
  float pi = 3.1416;
  int icnt = 10;     /* icnt is given as an argument to the calpro() function for calculating
        the specific heats and the integrals of polynomials */
   ⟨Check for dryout 55⟩;
   ⟨Computing $q'_{N,k+1}$ 57⟩;
   ⟨Computing $k_{c,k+1}$ 58⟩;
   ⟨Computing $h_{c,k+1}$ and $h_{g,k+1}$ 60⟩;
   ⟨Computing $R_{1,k+1}$ and $R_{2,k+1}$ 62⟩;
   ⟨Computing exponential term $e^{\frac{-\Delta t}{R_{1,k+1}C_{1,k}}}$ for $T_1$ 67⟩;
   ⟨Computing exponential term $e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}}$ for $T_2$ 64⟩;
   ⟨Computing exponential term $e^{\frac{-\Delta t}{R_{\text{CL},k+1}C_{\text{CL},k}}}$ for $T_{\text{CL}}$ 70⟩;
   ⟨Computing $T_{2,k+1}$ 65⟩;
   ⟨Computing $T_{1,k+1}$ 68⟩;
   ⟨Computing $T_{\text{CL},k+1}$ 71⟩;
   ⟨Computing $T_{S,k+1}$ 73⟩;
```

```
    ⟨Computing $\Delta H(T_{\text{abs},k+1})$ 75⟩;
    ⟨Computing $P_{\text{F,SUM},k+1}$ 76⟩;
    ⟨Computing $C_{1,k+1} = \pi r_f^2 \rho_1 c_{p,1,k+1}$ 78⟩;
    ⟨Computing $C_{2,k+1} = 2\pi r_c \tau_c \rho_2 c_{p,2,k+1}$ 79⟩;
    ⟨Computing $C_{\text{CL},k+1} = \pi r_f^2 \rho_1 c_{p,3,k+1}$ 80⟩;
    ⟨Computing $k_{\text{AV},k+1}$ 82⟩;
  if $(*MW\_flag \equiv 1)$ {
      ⟨Computing $q'_{\text{out},k+1}$ 84⟩;
      ⟨Computing $R_{\text{ox},k+1}$ 86⟩;
      ⟨Computing $q'_{\text{MWR},k+1}$ 88⟩;
      ⟨Computing $\delta_{\text{ox},k+1}$ 90⟩;
      ⟨Computing $q'_{\text{MWRI},k+1}$ 92⟩;
  }
```

This code is used in chunk 2

B.7.1 Checking for Dryout

We check for dryout using the condition given in DD28 of the SRS. If the dryout occurs, we output a message notifying the time and heat out at which it occured and assign the heat transfer coefficient between the fuel surface and the coolant at dryout (h_{dry}) to the coolant film conductance (h_b).

```
55  ⟨Check for dryout 55⟩ ≡        /* check for dryout */
  if $(*q\_out \geq *p\_dry \wedge *n \equiv 1)$
          {
    float $qout$, $tym$;
    $qout = *q\_out$;
    $tym = *time$;
        /* calling fuelpin15.f subroutine '$dryout\_$' to write out fuel sheath dryout time
          and $q'_{\text{out}}$ */
    $dryout\_(\&qout, \&tym)$;
  }
  if $(*q\_out \geq *p\_dry)$
          {
    $*n = 2$;
        /* assigning dryout heat tranfer coefficient to the coolant conductance */
    $*h\_b = *h\_dry$;
  }
```

This code is used in chunk 53

B.7.2 Computing $q'_{N,k+1}$ and $k_{c,k+1}$

The transient linear element power and clad conductivity are determined in the same way as done in B.6.1. At time t_{k+1}, the q'_N is calculated based on relative fuel power (q'_{NFRAC}) at t_{k+1} and is given by DD25 of the SRS as,

$$q'_{N,k+1} = q'_{\text{NFRAC},k+1} q'_{N_{\max}}; \tag{B.62}$$

We use the same chunk which calculates q'_N in the initialization section to compute $q'_{N,k+1}$, as the piece of code is same for both steady state and the transient state calculations.

```
57  ⟨Computing $q'_{N,k+1}$ 57⟩ ≡
    ⟨Calculation of $q'_N$ 17⟩;
```

This code is used in chunk 53

The value of clad conductivity (k_c) at time t_{k+1} depends on the average clad temperature (T_2) at t_k and is given by DD15 of the SRS as,

$$k_{c,k+1} = aT_{2,k} + b, \tag{B.63}$$

where a and b are constants obtained by a least squares fit to tabulated data and are given by Table TB2 of the SRS, where Table TB2 used different values for a and b if the temperature is greater than 1000^oC. We evaluate value of k_c at t_{k+1} as,

58 ⟨Computing $k_{c,k+1}$ 58⟩ ≡
```
float a, b;
if (*t_2 > 1000.0)
        {
    a = 2.727 · 10^{-05};
    b = -1.2727 · 10^{-03};
}
else
    {
    a = 1.43 · 10^{-05};
    b = 1.17 · 10^{-02};
}
*k_c = a * (*t_2) + b;
```
This code is used in chunk 53

B.7.3 Computing $h_{c,k+1}$ and $h_{g,k+1}$

Now using the $k_{c,k+1}$, we compute the heat transfer coefficient (h_c) and the gap conductance (h_g) at t_{k+1} in the same way we did in B.6.2 as,

$$h_{c,k+1} = \frac{2k_{c,k+1}h_b}{2k_{c,k+1} + \tau_c h_b}. \tag{B.64}$$

$$h_{g,k+1} = \frac{2k_{c,k+1}h_p}{2k_{c,k+1} + \tau_c h_p}. \tag{B.65}$$

Hence, reusing the chunk that has calculated h_c and h_g in the initialization section, $h_{c,k+1}$ and $h_{g,k+1}$ are computed as,

60 ⟨Computing $h_{c,k+1}$ and $h_{g,k+1}$ 60⟩ ≡
  ⟨Calculation of heat transfer coefficient (h_c) and the gap conductance (h_g) 21⟩;
This code is used in chunk 53

B.7.4 Computing $R_{1,k+1}$ and $R_{2,k+1}$

R_1 at t_{k+1} is computed in the same way we did in B.6.4 by taking the value of h_g at t_{k+1} and k_AV at t_k as,

$$R_{1,k+1} = \frac{f}{8\pi k_{\text{AV},k}} + \frac{1}{2\pi r_f h_{g,k+1}} \tag{B.66}$$

So, for computing $R_{1,k+1}$, we reuse the same piece of code that computes R_1 at steady state.
R_2 at t_{k+1} is computed taking the value of $h_{c,k+1}$ and is given by DD12 of SRS as,

$$R_{2,k+1} = \frac{1}{2\pi r_c h_{c,k+1}}. \tag{B.67}$$

62 $\langle$Computing $R_{1,k+1}$ and $R_{2,k+1}$ 62$\rangle \equiv$
$\langle$Calculation of R_1 30$\rangle$; /* computation of clad radius */
float *r_c* = **r_f* + (**tau_g*) + (**tau_c*);
float *r_2* = 1.0/(2.0 * *pi* * *r_c* * (**h_c*));
This code is used in chunk 53

B.7.5 Computing $T_{2,k+1}$

We solve Equation B.2 for $T_{2,k+1}$. The value of T_2 at time t_{k+1} is computed using $R_{1,k+1}$, $R_{2,k+1}$ and the values of C_2, T_1, T_2, q'_{MWR} at t_k. By taking C_2 of Equation B.2 to the RHS and rearranging, it simplifies to,

$$\frac{dT_2}{dt} = -\frac{(R_1+R_2)}{R_1R_2C_2}T_2 + \frac{T_1R_2 + q'_{\text{MWR}}R_1R_2 + T_BR_1}{R_1R_2C_2} \tag{B.68}$$

Comparing (B.68) with (B.4) using the Table B.2, the solution to Equation B.2 is given as,

$$T_{2,k+1} = T_{2,k}e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}} + \left(1-e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}}\right)\frac{T_{1,k}R_{2,k+1} + q'_{\text{MWR},k}R_{1,k+1}R_{2,k+1} + T_BR_{1,k+1}}{(R_{1,k+1}+R_{2,k+1})} \tag{B.69}$$

64 $\langle$Computing exponential term $e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}}$ for T_2 64$\rangle \equiv$
float *g* = *exp*((−(**delta*) * (*r_1* + *r_2*))/(*r_1* * *r_2* * (**c_2*)));
This code is used in chunk 53

65 $\langle$Computing $T_{2,k+1}$ 65$\rangle \equiv$
t_2* = **t_2* * *g* + ((1.0 − *g*) * (((t_1* * *r_2*) + (**q_MWR* * *r_1* * *r_2*) + ((**t_b*) * *r_1*))/(*r_1* + *r_2*)));
This code is used in chunk 53

B.7.6 Computing $T_{1,k+1}$

We solve Equation B.1 for $T_{1,k+1}$. The value of T_1 at time t_{k+1} is computed using $R_{1,k+1}$, $T_{2,k+1}$, $q'_{N,k+1}$ and the values of C_1, T_1 at t_k. By taking C_1 of Equation B.1 to the RHS and rearranging, it simplifies to,

$$\frac{dT_1}{dt} = -\frac{1}{R_1C_1}T_1 + \frac{T_2 + q'_NR_1}{R_1C_1} \tag{B.70}$$

Comparing Equation B.70 with Equation B.4, using Table B.2, the solution to Equation B.1 is given as,

$$T_{1,k+1} = T_{1,k}e^{\frac{-\Delta t}{R_{1,k+1}C_{1,k}}} + \left(1-e^{\frac{-\Delta t}{R_{1,k+1}C_{1,k}}}\right)(R_{1,k+1}q'_{N,k+1} + T_{2,k+1}) \tag{B.71}$$

67 $\langle$Computing exponential term $e^{\frac{-\Delta t}{R_{1,k+1}C_{1,k}}}$ for T_1 67$\rangle \equiv$
float *j* = *exp*(−(**delta*)/(*r_1* * (**c_1*)));
This code is used in chunk 53

68 $\langle$Computing $T_{1,k+1}$ 68$\rangle \equiv$
t_1* = *j* * (t_1*) + ((1.0 − *j*) * (*r_1* * (**q_N*) + (**t_2*)));
This code is used in chunk 53

B.7.7 Computing $T_{\mathrm{CL},k+1}$

Now we solve Equation B.3 for $T_{\mathrm{CL},k+1}$. The value of T_{CL} at time t_{k+1} is computed using $T_{1,k+1}$, $q'_{N,k+1}$ and the values of C_{CL}, T_{CL} and k_{AV} at t_k. By taking C_{CL} of Equation B.3 to the RHS and rearranging, it simplifies to,

$$\frac{dT_{\mathrm{CL}}}{dt} = -\frac{1}{R_{\mathrm{CL}}C_{\mathrm{CL}}}T_{\mathrm{CL}} + \frac{T_1 + q'_N R_{\mathrm{CL}}}{R_{\mathrm{CL}}C_{\mathrm{CL}}}, \tag{B.72}$$

where R_{CL} is the one half of the fuel resistance ($\mathrm{R_{FUEL}}$) and is given by DD13 of the SRS as,

$$R_{\mathrm{CL},k+1} = \frac{f}{8\pi k_{\mathrm{AV},k}} \tag{B.73}$$

Comparing Equation B.72 with Equation B.4 and using the Table B.2, the solution to Equation B.3 is given as,

$$T_{\mathrm{CL},k+1} = T_{\mathrm{CL},k}e^{\frac{-\Delta t}{R_{\mathrm{CL},k+1}C_{\mathrm{CL},k}}} + \left(1 - e^{\frac{-\Delta t}{R_{\mathrm{CL},k+1}C_{\mathrm{CL},k}}}\right)(R_{\mathrm{CL},k+1}q'_{N,k+1} + T_{1,k+1}) \tag{B.74}$$

70 ⟨Computing exponential term $e^{\frac{-\Delta t}{R_{\mathrm{CL},k+1}C_{\mathrm{CL},k}}}$ for T_{CL} 70⟩ ≡
```
float r_CL = *f/(8.0*pi*(*k_AV));      /* calculation of R_CL,k+1 */
float m = exp(-(*delta)/(r_CL*(*c_CL)));      /* calculation of exponential term */
```
This code is used in chunk 53

71 ⟨Computing $T_{\mathrm{CL},k+1}$ 71⟩ ≡
```
*t_CL = m*(*t_CL) + ((1.0 - m)*(r_CL*(*q_N) + (*t_1)));
```
This code is used in chunk 53

B.7.8 Computing $T_{S,k+1}$

The value of T_S at time t_{k+1} is calculated based on $T_{1,k+1}$, $T_{2,k+1}$ and is given by DD23 of the SRS as,

$$T_{S,k+1} = T_{2,k+1} + \frac{T_{1,k+1} - T_{2,k+1}}{R_{1,k+1}}R_{3,k+1}, \tag{B.75}$$

where $R_{3,k+1}$ is calculated as given by DD10 of the SRS as:

$$R_{3,k+1} = \frac{1}{2\pi r_f h_{g,k+1}} \tag{B.76}$$

73 ⟨Computing $T_{S,k+1}$ 73⟩ ≡
```
float r_3 = 1/(2*pi*(*r_f)*(*h_g));      /* calculation of gap resistance */
*t_S = *t_2 + ((*t_1 - *t_2)/(r_1)*(r_3));
```
This code is used in chunk 53

B.7.9 Computing $\Delta H(T_{\mathbf{abs}})$ and $P_{F,SUM}$

Now we compute the transient stored fuel energy in the same way we did in the initialization section (B.6.5). The stored fuel energy at time t_{k+1} depends on the value of absolute value of T_1 at t_{k+1} and is given by DD2 of the SRS as:

$$\Delta H(T_{\mathrm{abs},k+1}) = K_0\left(K_1\theta\left(\left(e^{\theta/T_{\mathrm{abs},k+1}}-1\right)^{-1}-\left(e^{\theta/T_{std}}-1\right)^{-1}\right)+K_2(T^2_{\mathrm{abs},k+1}-T^2_{std})+K_3e^{-E_D/(R_DT_{\mathrm{abs},k+1})}\right), \quad (B.77)$$

where the values of the constants are given in the initialization section. Reusing the chunks that initialize the constants and compute $\Delta H(T_{\mathrm{abs}})$ in the initialization section, we can compute $\Delta H(T_{\mathrm{abs},k+1})$ as,

75 ⟨Computing $\Delta H(T_{\mathrm{abs},k+1})$ 75⟩ ≡
```
⟨declaration of constants for stored energy 32⟩;
⟨ΔH(T_abs) 33⟩;
```
This code is used in chunk 53

The integrated fuel power ($P_{\mathrm{F,SUM}}$) at each time step t_{k+1}, is based on the relative fuel power ($q'_{\mathrm{NFRAC},k+1}$) and is given by the numerical approximation of the integral version shown in DD3 of the SRS as:

$$P_{\mathrm{F,SUM},k+1} = \sum_{0}^{i=k+1} q'_{\mathrm{NFRAC},i}\Delta t_i, \quad (B.78)$$

where $q_{\mathrm{NFRAC},i}$ is the relative fuel power at t_i.

76 ⟨Computing $P_{\mathrm{F,SUM},k+1}$ 76⟩ ≡
```
*f_p = *f_p + (*q_NFRAC * (*delta));
```
This code is used in chunk 53

B.7.10 Computing $C_1, C_2, C_3, c_{p,1}, c_{p,2}, c_{p,3}$

We evaluate the thermal capacitances and the specific heats in the same way we did in the initialization sections B.6.6, B.6.7 and B.6.8. So we reuse the chunks that have implemented the capacitances C_1, C_2, C_3 and their respective specific heats in the initialization section to compute the thermal capacitances and the specific heats at t_{k+1}.
At time t_{k+1}, the average fuel specific heat ($c_{p,1}$) is computed based on $T_{\mathrm{CL},k+1}$ and $T_{S,k+1}$. Taking this $c_{p,1,k+1}$, we evaluate $C_{1,k+1}$ as given in Equation B.45.

78 ⟨Computing $C_{1,k+1} = \pi r_f^2\rho_1 c_{p,1,k+1}$ 78⟩ ≡
```
float t_c, t_d;
int i, iflag;
i = 2;
iflag = 3;
float ts = *t_S;
float tcl = *t_CL;
    /* function calpro evaluates C_p(T_S,k+1) */
calpro_(&ts, &i, &iflag, &t_c, &icnt);
    /*function calpro evaluates C_p(T_CL,k+1) */
calpro_(&tcl, &i, &iflag, &t_d, &icnt);
⟨Calculation of C_1 and c_p,1 36⟩;
```
This code is used in chunk 53

At time t_{k+1}, the specific heat of the clad ($c_{p,2}$) is computed based on $T_{2,k+1}$. Taking this $c_{p,2,k+1}$, we evaluate $C_{2,k+1}$ as given in Equation B.51.

79 $\langle$Computing $C_{2,k+1} = 2\pi r_c \tau_c \rho_2 c_{p,2,k+1}$ 79$\rangle \equiv$

```
        /* calculation of specific heat of the clad ( $c_{p,2,k+1}$) at $T_2$ by calpro */
    i = 3;
    iflag = 2;
    float t2 = *t_2;
    float cp2;
    calpro_(&t2, &i, &iflag, &cp2, &icnt);
    ⟨Calculation of $C_2$ and $c_{p,2}$ 39⟩;
```

This code is used in chunk 53

At time t_{k+1}, the specific heat at the centerline ($c_{p,3}$) is computed based on $T_{\text{CL},k+1}$. Taking this $c_{p,3,k+1}$, we evaluate $C_{\text{CL},k+1}$ as given in Equation B.52.

80 $\langle$Computing $C_{\text{CL},k+1} = \pi r_f^2 \rho_1 c_{p,3,k+1}$ 80$\rangle \equiv$

```
        /*calculation of specific heat $c_{p,3,k+1}$ at $T_{\text{CL}}$ by calpro */
    i = 2;
    iflag = 2;
    tcl = *t_CL;
    float cp3;
    calpro_(&tcl, &i, &iflag, &cp3, &icnt);
    ⟨Calculation of $C_{\text{CL}}$ and $c_{p,3}$ 42⟩;
```

This code is used in chunk 53

B.7.11 Computing k_{AV}

Since k_{AV} is represented as first order polynomial function of temperature, at time t_{k+1}, the average fuel conductivity is explicitly obtained by integrating that expression from T_S to T_{CL}. That is,

$$k_{\text{AV}} = \int_{T_S}^{T_{\text{CL}}} \frac{kdT}{(T_{\text{CL}} - T_S)} \tag{B.79}$$

$$= \frac{1}{(T_{\text{CL}} - T_S)} [K(T)]_{T_S}^{T_{\text{CL}}}, \tag{B.80}$$

where

$$K(T) = \int kdT, \tag{B.81}$$

Hence,

$$k_{\text{AV}} = \frac{K(T_{\text{CL}}) - K(T_S)}{(T_{\text{CL}} - T_S)} \tag{B.82}$$

82 $\langle$Computing $k_{\text{AV},k+1}$ 82$\rangle \equiv$

```
    float t_a;      /*evaluation of $K(T)$ at $T_S$ by calpro*/
    i = 1;
    iflag = 1;
    ts = *t_S;
    calpro_(&ts, &i, &iflag, &t_a, &icnt);
    float t_e;      /*evaluation of $K(T)$ at $T_{\text{CL}}$ by calpro*/
```

```
$tcl = *t\_CL;$
$calpro\_(\&tcl, \&i, \&iflag, \&t\_e, \&icnt);$
    /* calculation of average fuel conductivity */
$*k\_AV = (t\_e - t\_a)/(*t\_CL - *t\_S);$
```
This code is used in chunk 53

B.7.12 Computing q'_{out}

We calculate the heat out (q'_{out}) in the same way as done in the initialization section (B.6.10). The heat out at time t_{k+1} depends on the value of T_2 and h_c at t_{k+1} and is given by DD27 of the SRS as:

$$q'_{\text{out},k+1} = \frac{1}{R_{2,k+1}}\Big(\frac{T_{2,k+1} - T_B}{q'_{N_{\max}}}\Big), \tag{B.83}$$

where $R_{2,k+1}$ is the effective resistance between the clad and the coolant film and is given by DD12 of the SRS as,

$$R_{2,k+1} = \frac{1}{2\pi r_c h_{c,k+1}} \tag{B.84}$$

Reusing the chunk calculating q'_{out} from the initializing section, we can calculate $q'_{\text{out},k+1}$

84 ⟨Computing $q'_{\text{out},k+1}$ 84⟩ ≡
```
    ⟨Calculation of $q'_{\text{out}}$ 50⟩;
```
This code is used in chunk 53

B.7.13 Computing rate of oxidation

We calculate the rate of oxidation (R_{ox}) in the same way as we did in the initialization section (B.6.9). The R_{ox} at time t_{k+1} depends on the value of T_2 at t_{k+1} and δ_{ox} at t_k and is given by DD5 of the SRS,

$$R_{\text{ox},k+1} = \frac{A}{1.56\delta_{\text{ox},k}} e^{\frac{-B}{R(T_{2,k+1}+273)}}, \tag{B.85}$$

where the values of constants A, B/R are given by Table TB1 of the SRS. Table TB1 uses different values for A and B/R if the temperature is greater than 1580^oC. We evaluate value of R_{ox} at t_{k+1} using the same chunk which calculates R_{ox} during the initialization section. However, before the chunk is called, the assignment of values to the variables A and $BbyR$ is done based on the value of T_2 as given by Table TB1 of the SRS.

86 ⟨Computing $R_{\text{ox},k+1}$ 86⟩ ≡
```
**float** $A$, $BbyR$;
**if** $(*t\_2 \le 1580.0)$ {
    $A = 6.48 \cdot 10^{-08};$
    $BbyR = 13586.0;$
}
**else** {
    $A = 1.0 \cdot 10^{-06};$
    $BbyR = 16014.0;$
}
⟨Calculation of $R_{\text{ox}}$ 45⟩;
**if** $(*t\_2 \ge 1850.0)$ {
    $*rate\_ox = (A/(1.56 * (*delta\_ox))) * exp(-(BbyR)/(1850.0 + 273.0));$
}
```
This code is used in chunk 53

B.7.14 Computing metal water reaction heat (q_{MWR})

We calculate the metal water reaction heat in the same way as we did in the initialization section (B.6.9). The q_{MWR} at time t_{k+1} depends on the value of R_{ox} at t_{k+1} and is given by DD5 of SRS as:

$$q'_{\text{MWR},k+1} = R_{\text{ox},k+1} 2\pi r_c \rho_2 q_r, \tag{B.86}$$

So, for evaluating $q'_{\text{MWR},k+1}$, we reuse the chunk that calculates q'_{MWR} in the steady state. Taking assumption A13 of the SRS into consideration, when all the clad material gets oxidized, that is, when the thickness of the reacted zircaloy (δ_{ox}) becomes equal to or greater than the clad thickness (τ_c), then there will not be any more metal water reaction taking place and hence no more q'_{MWR} will be generated. That is,

$$\delta_{\text{ox}} \geq \tau_c \Rightarrow q'_{\text{MWR}} = 0 \tag{B.87}$$

```
88  ⟨Computing $q'_{\text{MWR},k+1}$ 88⟩ ≡
      ⟨Calculation of $q'_{\text{MWR}}$ 46⟩;
      if (*delta_ox ≥ *tau_c) {
         *q_MWR = 0.0;
      }
```
This code is used in chunk 53

B.7.15 Computing oxidation layer thickness

We calculate the oxidation layer thickness in the same way as we did in the initialization section (B.6.9). The δ_{ox} at time t_{k+1} depends on the value of R_{ox} at t_{k+1} and is given as:

$$\delta_{\text{ox},k+1} = \delta_{\text{ox},k} + R_{\text{ox},k+1}\Delta t \tag{B.88}$$

So, for evaluating $\delta_{\text{ox},k+1}$, we reuse the chunk that calculates δ_{ox} in the steady state. But once the δ_{ox} becomes equal to or greater than the clad thickness (τ_c), as there will not be anymore metal water reaction taking place, the rate of oxidation of the clad becomes zero. That is,

$$\delta_{\text{ox}} \geq \tau_c \Rightarrow R_{\text{ox}} = 0 \tag{B.89}$$

```
90  ⟨Computing $\delta_{\text{ox},k+1}$ 90⟩ ≡
      if (*delta_ox ≥ *tau_c) {
         *rate_ox = 0.0;
      }
      ⟨Calculation of $\delta_{\text{ox}}$ 48⟩;
```
This code is used in chunk 53

B.7.16 Computing Integrated metal water reaction heat (q'_{MWRI})

The integrated metal water reaction heat is a summation of q'_{MWR} normalized by $q'_{N_{\max}}$ at each time step. At time t_{k+1}, the q'_{MWRI} is based on $q'_{\text{MWR},k+1}$ and is given by the numerical approximation of the integral form given in DD26 of the SRS as,

$$q'_{\text{MWRI},k+1} = \frac{1}{q'_{N_{\max}}} \sum_{i=0}^{k+1} q'_{\text{MWR},i}\Delta t_i \tag{B.90}$$

```
92  ⟨Computing $q'_{\text{MWRI},k+1}$ 92⟩ ≡
      *q_MWRI = *q_MWRI + ((*q_MWR/(*q_Nmax)) * (*delta));
```
This code is used in chunk 53

We store the program into the C file

94 ⟨fuel_temp.c 94⟩ ≡

```
#include <math.h>
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
   ⟨fuel temp function 2⟩;
```

Appendix C

Checklist

1. All the goals of the software are defined.
2. The problem statement, context, scope and purpose are clearly stated.
3. All the required background information is given.
4. All the abbreviations and acronyms are defined.
5. All the abbreviations and acronyms given are used.
6. All the constraints are mentioned.
7. Every term is defined with its unit of measurement.
8. All the laws and equations needed to build the mathematical models are given.
9. All the mathematical models are developed.
10. Derivations of all the models and necessary data definitions are given.
11. Every assumption is recorded, labelled and referenced at least once.
12. Every figure and table is labelled and is referenced.
13. Every symbol is defined with its unit of measurement.
14. Every defined symbol is used.
15. Values of all the constants are given.
16. Cross referencings are made throughout the document.
17. Each term in the document has a unique symbol and definition.
18. Each requirement is specified only once.
19. Every line of the code must trace back to either the definitions or assumptions or models or constants of the SRS or to the numerical algorithm or assumptions used in the LP Manual.
20. Each data definition and instance model is implemented only once and the chunks are reused where necessary.
21. Every data definition or model mentioned in the LP Manual should be referenced back to the SRS.

Appendix D

makefile

```
FC= gfortran
CC= gcc
CFLAGS = -g -c -Wall -ansi
FFLAGS = -g -c -Wall -fno-automatic

fp:
ctangle fuel_temp
gcc $(CFLAGS) fuel_temp.c
gfortran $(FFLAGS) fp.f
gfortran -o test fp.o fuel_temp.o

pdf:
cweave fuel_temp.w
pdflatex fuel_temp

clean:
@-$(RM) *.aux *.idx *.out *.scn *.toc *.c *.log fuel_temp *.o
```

Index

A: 44 86.
a: 18 58.
b: 18 58.
BbyR: 44 45 86.
c_CL: 2 3 5 8 11 42 70.
c_p1: 2 3 5 8 11 36.
c_p2: 2 3 5 8 11 39.
c_p3: 2 3 5 8 11 42.
c_1: 2 3 5 8 11 36 67.
c_2: 2 3 5 8 11 39 64.
*calpro_*: 2 11 28 35 38 41 78 79 80 82.
cp2: 38 39 79.
cp3: 41 42 80.
C10: 18.
C11: 18.
C12: 18.
C13: 18.
delta: 2 3 5 48 64 67 70 76 92.
delta_ox: 2 3 5 8 11 44 45 48 86 88 90.
deltaHT_abs: 2 3 5 8 11 33.
*dryout_*: 2 11 55.
E_D: 32 33.
exp: 33 45 64 67 70 86.
f: 2 3 5.
f_p: 2 3 5 8 11 51 76.
fabsf: 28.
*fuel_temp_*: 1 2 3 5 11.
g: 64.
h_b: 2 3 5 8 11 18 21 55.
h_c: 2 3 5 8 11 21 50 62.
h_dry: 2 3 5 55.
h_g: 2 3 5 8 11 21 22 30 73.
h_ib: 2 3 5 18.
h_p: 2 3 5 21.
htout: 2.
i: 2 28 78.
icnt: 2 15 28 53 78 79 80 82.
idnt: 28 35 38 41.
iflag: 2 28 78 79 80 82.
init_flag: 1 2 3 5 8 11.
j: 35 67.
jflag: 35.
k: 38.
k_AV: 2 3 5 8 11 24 26 28 30 70 82.
k_c: 2 3 5 8 11 19 21 58.
kflag: 38.
K0: 32 33.
K1: 32 33.
K2: 32 33.
K3: 32 33.
l: 41.
lflag: 41.
m: 70.
MW_flag: 2 3 5 53.
n: 2 3 5.
p_dry: 2 3 5 55.
pi: 15 18 22 24 25 26 30 36 39 42 46 50 53 62 70 73.
prpval: 2.
q_MWR: 2 3 5 8 11 46 65 88 92.
q_MWRI: 2 3 5 8 11 47 92.
q_N: 2 3 5 8 11 17 18 22 24 26 27 31 68 71.
q_NEST: 25 26 27.
q_NFRAC: 2 3 5 17 76.
q_Nmax: 2 3 5 17 50 92.
q_out: 2 3 5 8 11 50 55.
q_r: 46.
qout: 55.
r_c: 18 39 46 50 62.
r_CL: 14 70 71.
R_D: 32 33.
r_f: 2 3 5 18 22 30 36 42 62 73.
r_fuel: 12 24.
r_1: 12 14 30 31 64 65 67 68 73.
r_2: 12 14 50 62 64 65.
r_3: 12 14 22 73.
rate_ox: 2 3 5 8 11 45 46 48 86 90.
re: 27 28.

rho_1: 2 3 5 36 42.
rho_2: 2 3 5 39 46.
sqrt: 18.
t: 2.
t_a: 25 28 82.
t_abs: 33.
t_b: 2 3 5 18 50 65.
t_c: 35 36 78.
t_CL: 2 3 5 8 11 24 26 28 35 36 41 71 78 80 82.
t_d: 35 36 78.
t_e: 25 28 82.
t_S: 2 3 5 8 11 22 24 26 28 35 36 73 78 82.
t_std: 32 33.
t_1: 2 3 5 8 11 31 33 65 68 71 73.
t_2: 2 3 5 8 11 18 19 22 31 38 45 50 58 65 68 73 79 86.
tau_c: 2 3 5 18 21 39 62 88 90.
tau_g: 2 3 5 18 62.
tcl: 28 35 41 78 80 82.
`THETA`: 32 33.
time: 2 3 5 55.
ts: 28 35 78 82.
tym: 55.
t2: 38 79.

List of Refinements

$\langle \Delta H(T_{\text{abs}})$ 33$\rangle$ Used in chunks 15 and 75.

$\langle$Calculation of C_1 and $c_{p,1}$ 36$\rangle$ Used in chunks 15 and 78.

$\langle$Calculation of C_2 and $c_{p,2}$ 39$\rangle$ Used in chunks 15 and 79.

$\langle$Calculation of C_{CL} and $c_{p,3}$ 42$\rangle$ Used in chunks 15 and 80.

$\langle$Calculation of R_1 30$\rangle$ Used in chunks 15 and 62.

$\langle$Calculation of R_{ox} 45$\rangle$ Used in chunks 15 and 86.

$\langle$Calculation of δ_{ox} 48$\rangle$ Used in chunks 15 and 90.

$\langle$Calculation of k_c 19$\rangle$ Used in chunk 15.

$\langle$Calculation of q'_N 17$\rangle$ Used in chunks 15 and 57.

$\langle$Calculation of q'_{MWR} 46$\rangle$ Used in chunks 15 and 88.

$\langle$Calculation of q'_{out} 50$\rangle$ Used in chunks 15 and 84.

$\langle$Calculation of heat transfer coefficient (h_c) and the gap conductance (h_g) 21$\rangle$ Used in chunks 15 and 60.

$\langle$Calpro function for C_1 and $c_{p,1}$ 35$\rangle$ Used in chunk 15.

$\langle$Calpro function for C_2 and $c_{p,2}$ 38$\rangle$ Used in chunk 15.

$\langle$Calpro function for C_{CL} and $c_{p,3}$ 41$\rangle$ Used in chunk 15.

$\langle$Check for dryout 55$\rangle$ Used in chunk 53.

$\langle$Computing $C_{1,k+1} = \pi r_f^2 \rho_1 c_{p,1,k+1}$ 78$\rangle$ Used in chunk 53.

$\langle$Computing $C_{2,k+1} = 2\pi r_c \tau_c \rho_2 c_{p,2,k+1}$ 79$\rangle$ Used in chunk 53.

$\langle$Computing $C_{\text{CL},k+1} = \pi r_f^2 \rho_1 c_{p,3,k+1}$ 80$\rangle$ Used in chunk 53.

$\langle$Computing $P_{\text{F,SUM},k+1}$ 76$\rangle$ Used in chunk 53.

$\langle$Computing $R_{1,k+1}$ and $R_{2,k+1}$ 62$\rangle$ Used in chunk 53.

$\langle$Computing $R_{\text{ox},k+1}$ 86$\rangle$ Used in chunk 53.

$\langle$Computing $T_{1,k+1}$ 68$\rangle$ Used in chunk 53.

$\langle$Computing $T_{2,k+1}$ 65$\rangle$ Used in chunk 53.

$\langle$Computing $T_{S,k+1}$ 73$\rangle$ Used in chunk 53.

$\langle$Computing $T_{\text{CL},k+1}$ 71$\rangle$ Used in chunk 53.

$\langle$Computing $\Delta H(T_{\text{abs},k+1})$ 75$\rangle$ Used in chunk 53.

$\langle$Computing $\delta_{\text{ox},k+1}$ 90$\rangle$ Used in chunk 53.

$\langle$Computing $h_{c,k+1}$ and $h_{g,k+1}$ 60$\rangle$ Used in chunk 53.

$\langle$Computing $k_{\text{AV},k+1}$ 82$\rangle$ Used in chunk 53.

$\langle$Computing $k_{c,k+1}$ 58$\rangle$ Used in chunk 53.

$\langle$Computing $q'_{N,k+1}$ 57$\rangle$ Used in chunk 53.

$\langle$Computing $q'_{\text{MWRI},k+1}$ 92$\rangle$ Used in chunk 53.

$\langle$Computing $q'_{\text{MWR},k+1}$ 88$\rangle$ Used in chunk 53.

$\langle$Computing $q'_{\text{out},k+1}$ 84$\rangle$ Used in chunk 53.

$\langle$Computing exponential term $e^{\frac{-\Delta t(R_{1,k+1}+R_{2,k+1})}{R_{1,k+1}R_{2,k+1}C_{2,k}}}$ for T_2 64$\rangle$ Used in chunk 53.

$\langle$Computing exponential term $e^{\frac{-\Delta t}{R_{1,k+1}C_{1,k}}}$ for T_1 67$\rangle$ Used in chunk 53.

$\langle$ Computing exponential term $e^{\frac{-\Delta t}{R_{\mathrm{CL},k+1} C_{\mathrm{CL},k}}}$ for T_{CL} 70 $\rangle$ Used in chunk 53.
$\langle$ computation of T_{CL} 24 $\rangle$ Used in chunk 28.
$\langle$ convergence routine to determine $k_{\mathrm{AV},0}$ and $T_{\mathrm{CL},0}$ 28 $\rangle$ Used in chunk 15.
$\langle$ declaration of constants for stored energy 32 $\rangle$ Used in chunks 15 and 75.
$\langle$ dynamic section 53 $\rangle$ Used in chunk 2.
$\langle$ estimation of q'_N 25 $\rangle$ Used in chunk 28.
$\langle$ fuel temp function 2 $\rangle$ Used in chunk 94.
$\langle$ `fuel_temp.c` 94 $\rangle$
$\langle$ initialization of $f_{p,0}$ 51 $\rangle$ Used in chunk 15.
$\langle$ initialization of q'_{MWRI} 47 $\rangle$ Used in chunk 15.
$\langle$ initialization of average clad temperature $T_{2,0}$ 18 $\rangle$ Used in chunk 15.
$\langle$ initialization of average fuel temperature $T_{1,0}$ 31 $\rangle$ Used in chunk 15.
$\langle$ initialization of constants for R_{ox} 44 $\rangle$ Used in chunk 15.
$\langle$ initialization of surface temperature ($T_{S,0}$) 22 $\rangle$ Used in chunk 15.
$\langle$ initialization section 15 $\rangle$ Used in chunk 2.
$\langle$ relative error computation 27 $\rangle$ Used in chunk 28.
$\langle$ update k_{AV} 26 $\rangle$ Used in chunk 28.