

SPECTRA

Computer Code Manuals


Volume 4 - Code Structure, Development, Hardware and Software Requirements


Version 24-02

M.M. Stempniewicz

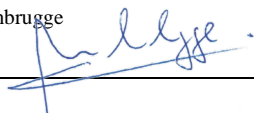
Arnhem, February 2024

K6223/24.277594 MSt-2402

author : M.M. Stempniewicz 

reviewed : E.A.R. de Geus 

25 page(s)

approved : J.J. Meulenbrugge 

Spectra-Vol4.doc

© NRG 2024

Subject to agreement with the client, the information contained in this report may not be disclosed to any third party and NRG is not liable for any damage arising out of the use of such information.

EU DuC = N

Goods labeled with an EU DuC (European Dual-use Codification) not equal to 'N' are subject to European and national export authorization when exported from the EU and may be subject to national export authorization when exported to another EU country as well. Even without an EU DuC, or with EU DuC 'N', authorization may be required due to the final destination and purpose for which the goods are to be used. No rights may be derived from the specified EU DuC or absence of an EU DuC.

Contents

Abstract	5
Explanation of names and abbreviations	6
1 Introduction	7
2 Code Structure	8
2.1 Introduction	8
2.2 Code Structure	8
3 Maintenance and Development of the Code	16
3.1 Bug Fixing Procedure	16
3.2 Code Modification Procedure	17
3.3 Code Development Procedure	17
3.4 Code Compilation and Quality Check Procedure	18
3.5 Code Release Procedure	19
4 Hardware and Software Requirements	21
4.1 Operating Systems	21
4.2 Requirements	21
Literature	22
Appendix A Bug Report Form (QA-SPE-004)	23
Appendix B Code Modification Form (QA-SPE-005)	24
Appendix C Release Notes Form (QA-SPE-006)	25

Abstract

SPECTRA (Sophisticated Plant Evaluation Code for Thermal-hydraulic Response Assessment) is a fully integrated system analysis code, that models thermal-hydraulic behavior of Nuclear Power Plants, including reactor cooling system, emergency and control systems, containment, reactor building, etc. of various reactor types, like BWR, PWR, HTR. It can also be used to assess thermal-hydraulic response of non-nuclear plants, for example cooling systems of chemical reactors.

The full documentation of SPECTRA consists of the following four volumes:

- Volume 1: Program Description
- Volume 2: User's Guide
- Volume 3: Verification and Validation
- Volume 4: Code Structure, Development, Hardware and Software Requirements

This report presents Volume 4 of the SPECTRA Code Manuals - Code Structure, Development, Hardware and Software Requirements.

The SPECTRA Manuals are freely available in internet and are also supplied together with the SPECTRA code. The Volume 4 of the Code Manuals is provided in the file **Spectra-Vol4.pdf**.

Explanation of names and abbreviations

CF	Control Function
CV	Control Volume
DIA	Diagnostics file
ICF	Initial Condition File
IT	Isotope Transformation
JN	CV Junction
MP	Material Properties
OUT	Output file
OX	Metal Oxidation
PLT	Plot file
RK	Reactor Kinetics
RT	Radioactive Particle Transport
SC	1-D Solid Heat Conductor
SPECTRA	S ophisticated P lant E valuation C ode for T hermal-hydraulic R esponse A ssessment
TC	2-D Solid Heat Conductor
TF	Tabular Function
TFD	Tabular Function Data file
TR	Thermal Radiation

1 Introduction

SPECTRA (Sophisticated Plant Evaluation Code for Thermal-hydraulic Response Assessment) is a fully integrated system analysis code, that models the thermal-hydraulic behavior of Nuclear Power Plants, including reactor cooling system, emergency and control systems, containment, reactor building, etc. of various reactor types, like BWR, PWR, HTR. It can also be used to assess thermal-hydraulic response of non-nuclear plants, for example cooling systems of chemical reactors. The SPECTRA Code Manuals consists of the following four volumes:

- Volume 1: Program Description.
- Volume 2: User's Guide.
- Volume 3: Verification and Validation.
- Volume 4: Code Structure, Development, Hardware and Software Requirements

This report presents Volume 4 of the SPECTRA Code Manuals - Code Structure, Development, Hardware and Software Requirements. This volume presents information on the:

- code structure,
- maintenance and development of the code, including:
 - bug fixing procedure,
 - code modification procedure,
 - code development procedure,
 - code compilation and quality check procedure,
 - code release procedure,
- hardware and software requirements, including:
 - operating systems,
 - information on the hardware and software requirements.

2 Code Structure

2.1 Introduction

SPECTRA is written in FORTRAN-77 and was developed for the Windows and Linux operating systems. Depending on the user's needs, one or both versions may be distributed. The basic approach taken in developing the models was to obtain a broad scope of modeling and possibly fastest execution, at the expense (when necessary) of the memory requirement.

2.2 Code Structure

The structure of SPECTRA is shown in Figure 2-1. The structure of the program is described in terms of *Packages*, *Models* and *Equations*, defined below.

- "*Packages*"

The total program consists of 20 blocks, referred to as "*Packages*". Each *Package* is identified in Figure 2-1 by a two character symbol enclosed by "=" signs. The *Packages* are characterized by:

- Each *Package* resides in a separate subdirectory of the SPECTRA source code directory.
- Description of each *Package* is given in a separate chapter in Volume 1.

- "*Models*"

Packages consist of physical or mathematical models, referred to as "*Models*". An example of a physical *Model* is the critical flow model. An example of mathematical *Model* is a matrix solver (i.e. set of procedures to solve a set of linear equations). *Models* available within each *Package* are shown below, in Figure 2-2 through Figure 2-6. The *Models* are characterized by:

- Each *Model* is written as a separate file (FORTRAN files with extension .FOR).
- Description of each *Model* is given in a separate section in Volume 1.

- "*Equations*"

Models consist of one or more procedures, referred to in this report as "*Equations*". For example the critical flow *Model* consists of *Equations* used for subcooled liquid, superheated steam, perfect gas, etc. The matrix solution *Model* consists of *Equations* to solve full matrices, sparse matrices, tri-diagonal matrices, etc. The *Equations* are characterized by:

- Each *Equation* is written as a separate code segment (subroutine or function).
- Description of each *Equation* is given in a separate sub-section in Volume 1.

All *Packages*, except for the Main Program, =SPE=, and the numerical solver, =SL=, are grouped into five "Groups of *Packages*" in Figure 2-1. The Groups of *Packages* are introduced here for descriptive convenience. They have no meaning for the internal code structure. The first three Groups of *Packages* contain physical *Models*. The next Group contains mathematical *Models*.

The last Group of *Packages* contain procedures needed for input processing. It should be noted that this Group contains only the main I/O procedures. Most of the *Packages* from the first four Groups have their own specific routines for input processing and editing data.

As indicated by the connecting lines in Figure 2-1, the SPECTRA Main Program, =SPE=, interacts only with I/O procedures and the Numerical Solver. The task of Main Program is to use the I/O Procedures to read input and to initiate the solution by a call to the Numerical Solver.

The Numerical Solver *Package* is responsible for solving all *Equations*, using a stable, implicit method. As indicated in Figure 2-1, the Numerical Solver interacts with each *Package* from the first four Groups of *Packages*. All *Packages* are included in the main iteration loop to obtain the implicit solution. The Solver *Package* is general enough to find solutions of even complex problems. When the solution does not converge using the attempted time step, the time step is automatically reduced.

Most of the code segments (*Equations*) from the first three *Groups of Packages* reflect the real physical laws. Subroutines or functions, in which the *Equations* are coded, contain extensive comments and references to literature where the equations can be found. This makes verification and eventual future modifications relatively easy.

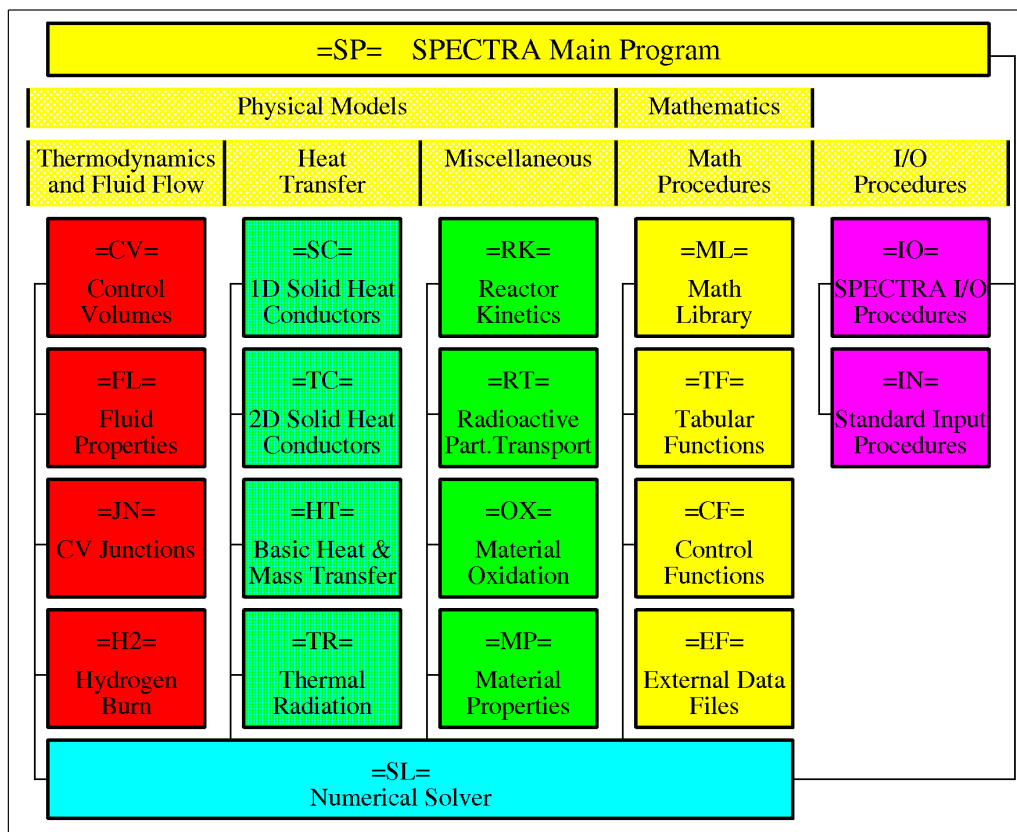


Figure 2-1 SPECTRA code structure.

Figure 2-2 shows *Packages* from the Thermodynamics and Fluid Flow Group. *Models* available in each *Package* are also shown. The *Models* are marked by filled bullets (filled squares). The main assumptions, made to develop the *Models*, are marked by empty bullets (empty squares). This Group consists of the following *Packages*:

- =CV=** Control Volume *Package*. The main *Models* in this *Package* include: mass end energy balance, atmosphere stratification and pool stratification.
- =FL=** Fluid Property *Package*. The gas *Model* considers six built-in gases, treated as real gases, and a number of user-defined gases, treated as ideal gases. Phase change is possible only for steam, for which the saturation (liquid/gas) properties are built-in. In addition to the built-in fluids, user-defined gases (e.g. Ne, Ar, Xe) and liquids (e.g.: liquid metals, molten salts) may be used.
- =JN=** Junction *Package*. The *Models* in this *Package* are: momentum balance, drift flux, vent phenomena, critical flow, wall friction, two-phase pressure drop, pump/compressor and turbine *Models*, and the *Model* to influence gas composition in a junction, for conservative analyses.
- =H2=** Hydrogen Burn *Package*. The *Models* in this *Package* are: temperature-dependent flammability and ignition limits, hydrogen burns including slow deflagrations, fast turbulent deflagrations and detonations.

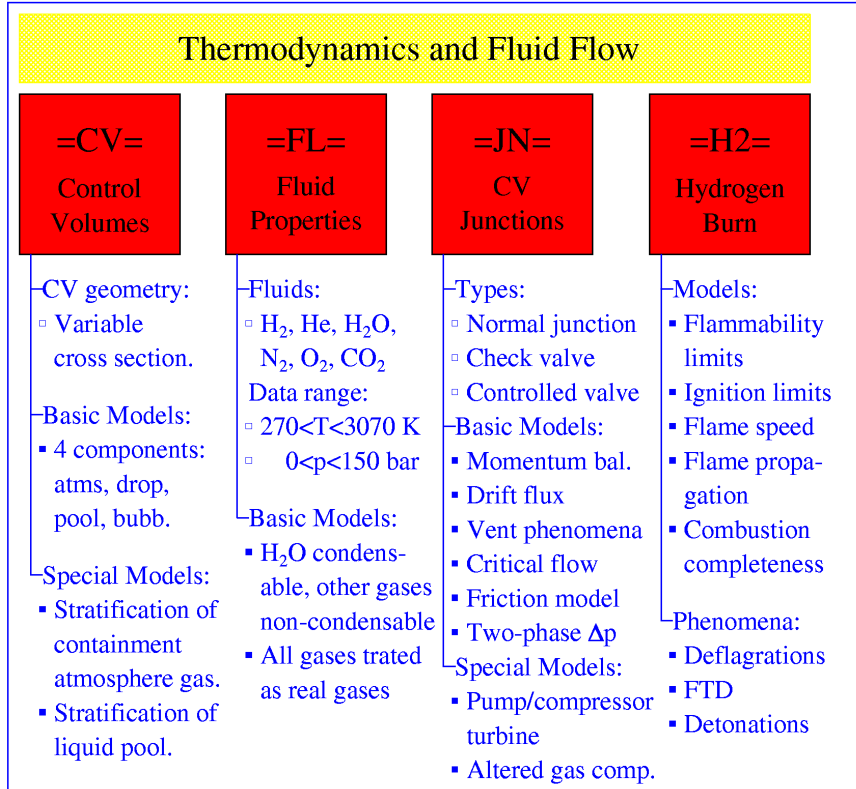


Figure 2-2 SPECTRA code structure - Thermodynamics and Fluid Flow Packages.

Figure 2-3 shows *Packages* from the Heat Transfer Group. This Group consists of the following *Packages*:

- =SC=** 1-D Solid Heat Conductor *Package*. This *Package* contains one *Model*: 1-D transient heat conduction. Multi-dimensional effects may be simulated by connecting several 1-D structures using the so called direct contact conduction model.
- =TC=** 2-D Solid Heat Conductor *Package*. This *Package* contains one *Model*: 2-D transient heat conduction. 3-D effects may be simulated by connecting several 2-D structures using the so called direct contact conduction model.
- =HT=** Basic Heat and Mass Transfer *Package*. This *Package* contains the main *Models* for heat transfer from wall surface and pool surface consisting of wall-to-pool, wall-to-gas and pool-to-gas sub-models. The above *Models* use the following individual heat transfer *Models*: natural convection, forced convection, boiling, including nucleate, transition and film boiling, condensation, heat transfer in two-phase flow. The *Package* includes also the *Model* of non-equilibrium vapor generation rate.
- =TR=** Thermal Radiation *Package*. This *Package* contains net enclosure, grey gas approximation radiative heat exchange *Models*, *Models* for gas radiation properties and surface radiation properties.

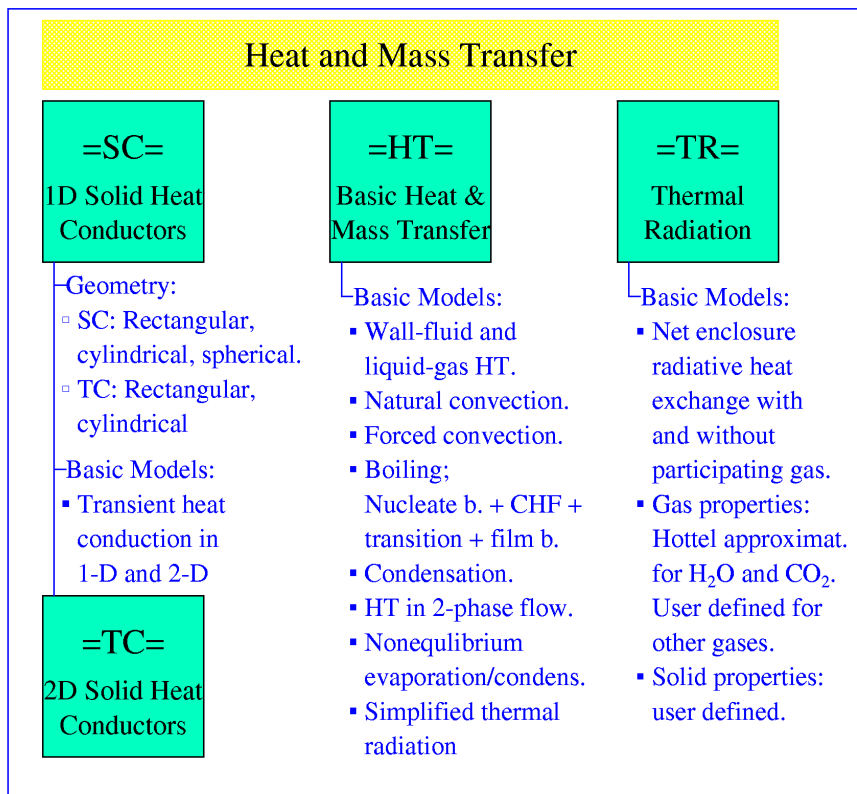


Figure 2-3 SPECTRA code structure - Heat and Mass Transfer Packages.

Figure 2-4 shows *Packages* from the Miscellaneous Group. This Group consists of the following *Packages*:

- =RK=** Reactor Kinetics *Package*, includes the point kinetics (space-independent) and nodal point kinetics, and the circulating fuel *Models*, with reactivity feedbacks from fuel and moderator temperature changes, void fraction changes, and changes of isotope concentrations (for example poisons such as Xe-135). The latter effect is calculated by the Isotope Transformation *Model*.
- =RT=** Radioactive Particle Transport *Package*, includes aerosol transport, deposition and resuspension, radioactive isotope chains.
- =OX=** Metal Oxidation *Package*, includes *Models* for Zircaloy and steel oxidation, as well as a general *Model* for oxidation of any metal. The general *Model* uses parabolic reaction rate. The coefficients for the parabolic reaction rate are defined by the user.
- =MP=** Material Property *Package*. Tables used to define properties of materials used for 1-D and 2-D solid heat conductors.

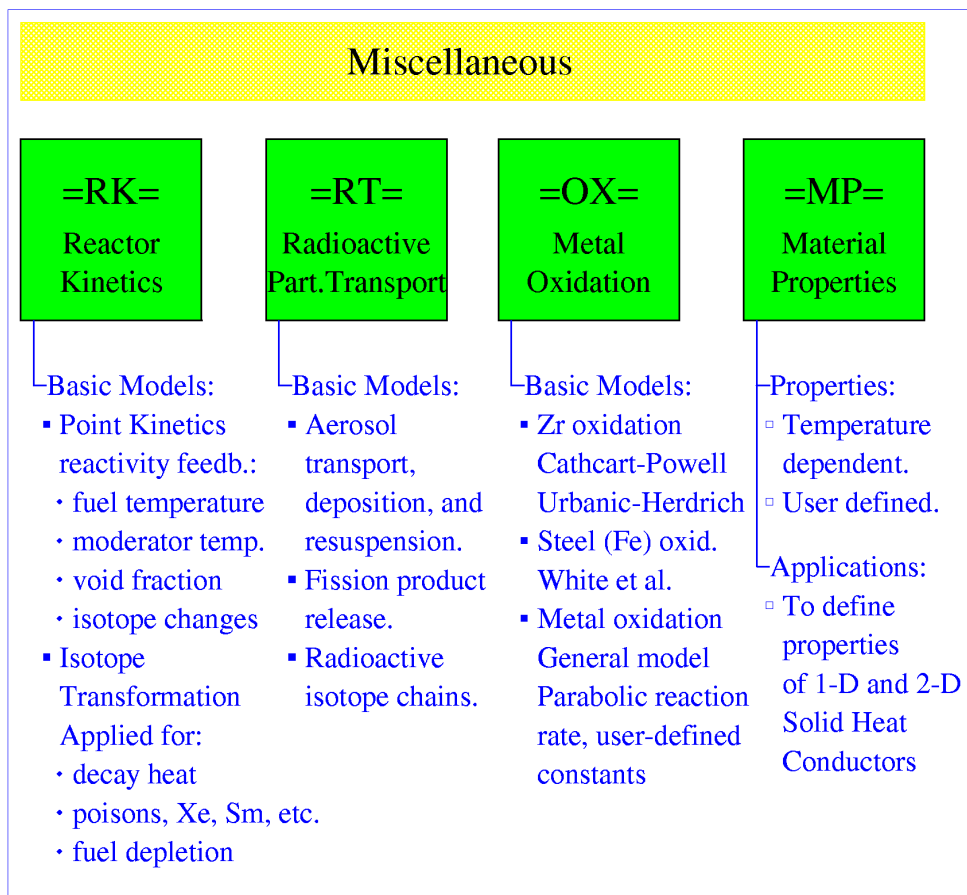


Figure 2-4 SPECTRA code structure - Miscellaneous Packages.

Figure 2-5 shows *Packages* from the Math Procedures Group. This Group consists of the following *Packages*:

- =ML=** Math Library *Package*. This package contains standard models taken from the Numerical Recipes in FORTRAN [2].
- =TF=** Tabular Function *Package*. Tabular functions allow the user to define time dependent parameters. All tabular functions are assumed to be functions of time. This allows eliminating Tabular Functions from the iteration procedure to obtain the implicit solution. Tabular Functions may be defined interactively during program execution. With the interactive TF the user or an external code can communicate with SPECTRA during the execution; thus allowing the use of SPECTRA as a background code in a plant simulator.
- =CF=** Control Function *Package*. These are general functions, dependent on arbitrary arguments. Control Functions are by default included in the main iteration loop to obtain the implicit solution.
- =EF=** External Data File *Package*. This package contains functions for writing and reading information to and from external files.

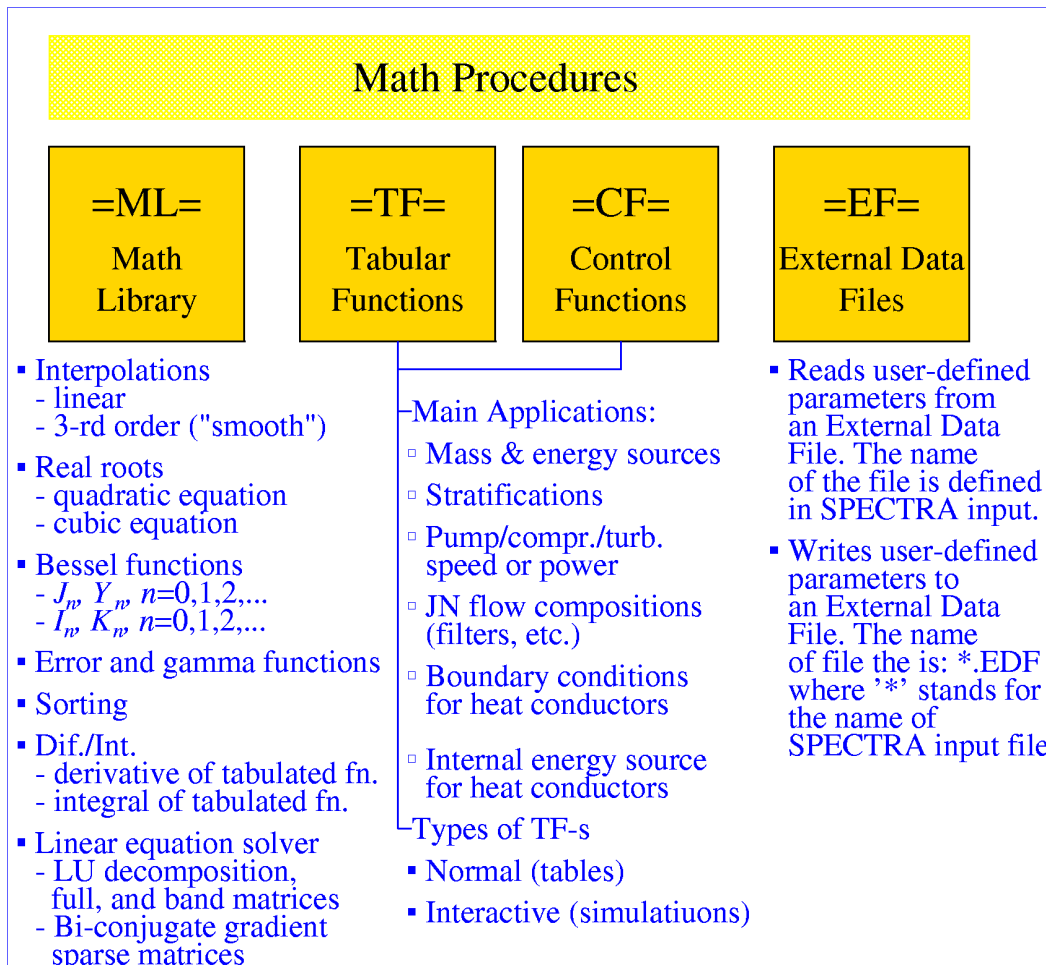


Figure 2-5 SPECTRA code structure - Math Packages.

Figure 2-6 shows *Packages* from the I/O Procedures Group. This Group consists of the following *Packages*:

=IO= SPECTRA I/O Procedures *Package*. The *Package* contains the main procedures to read input files and write output files. This *Package* is only a "coordinator" of input and output processing. The following *Packages*: =CV=, =JN=, =SC=, =TC=, =TR=, =TF=, =CF=, =H2=, =OX=, =MP=, =SL=, have their own procedures, responsible for reading, checking, and writing data belonging to the appropriate *Package*.

=IN= Standard Input Procedures *Package*. The *Package* contains standard procedures to perform preliminary processing of input (remove comments, perform file attachments, etc.).

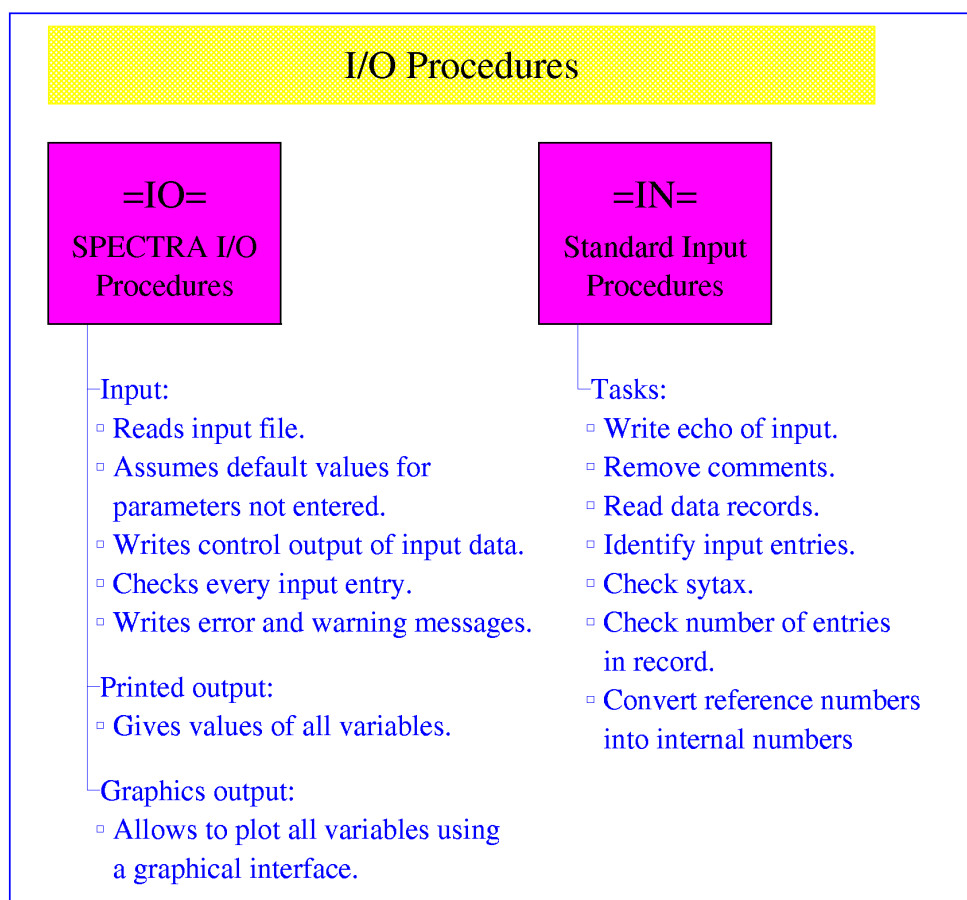


Figure 2-6 SPECTRA code structure - Input / Output Packages.

The last two *Packages*, not belonging to any of the Groups discussed above, are the Numerical Solver *Package* (Figure 2-7) and the SPECTRA Main Program.

=SL= The Numerical Solver *Package*. This *Package* is responsible for solving the problem using the implicit scheme (the Control Functions may be excluded, individually or globally, from the implicit solution by the user).

=SPE= SPECTRA Main Program. The tasks of the Main Program are:

- To initiate reading and processing input. This is done by a call to the main subroutine from the =IO= *Package*.
- To initiate problem solution if the input is read properly. This is done by a call to the main subroutine from the =SL= *Package*. If errors are detected by =IO= *Package* then Main Program terminates the execution and writes an appropriate message.

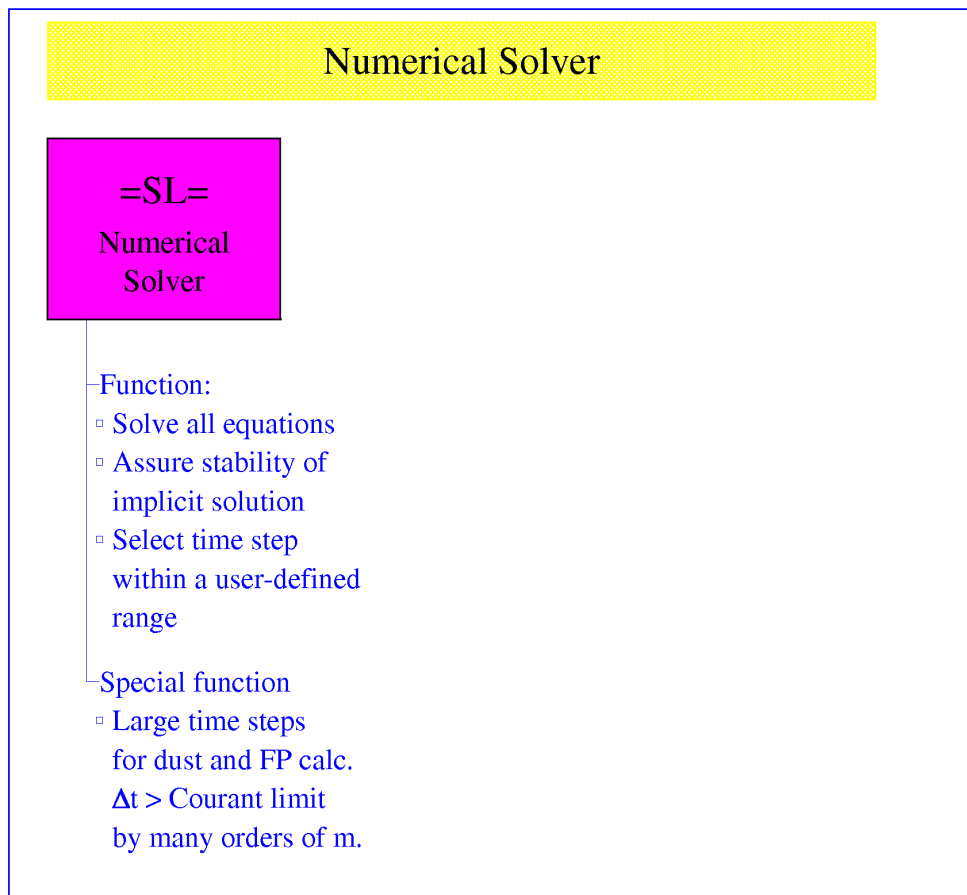


Figure 2-7 SPECTRA code structure - Numerical Solver Package

3 Maintenance and Development of the Code

The SPECTRA code has been developed under NRG's corporate Quality Management System and satisfies the requirements of ANSI/ANS standards regarding code development, programming and V&V. This software administration instruction was prepared as a supporting document for NRG's corporate QA procedure CS-MA-OD-021456 "Controlled development and maintenance of software for numerical analysis" [5] and administrative instruction [6]. This chapter describes the activities related to code maintenance and development, in particular the bug fixing procedure (section 3.1), the code modification procedure (section 3.2), and the code development procedure (section 3.3). Next, a description of the code compilation and quality checks is described (section 3.4). Finally the code release procedure is described (section 3.5).

3.1 Bug Fixing Procedure

The bug fixing procedure is based on the SPECTRA Software Administration Instruction [6]. The SPECTRA Software Administrator is NRG's single point of contact for reporting errors, malfunctions, users comment and other SPECTRA software related issues. The bug reports are stored in the code bug folder Z-BUG. The administrative channels for reporting errors are provided in the release documentation of the code. The incoming correspondence will be reviewed and, when needed, the following actions will be taken:

- A solution for the error/malfunction or the mentioned issue is defined as soon as possible.
- Within 5 working days the parties concerned will be informed about the proposed solution for the reported issue.
- The final solution for the reported issue will be communicated with the party concerned as soon as possible.
- In case a reported issue and its solution are of importance for all SPECTRA users, they will be informed about it. The users may request an immediate (unscheduled) release of the new code version. If such a request is not made, the solution will be available for the users after the next scheduled release of the new code version.
- The release procedure depends on the priority, that is being assigned by the user:
 - **Very high:** solution needed a.s.a.p., i.e. within days. The bug-fix code version (old code release with the bug fixed) will be delivered to the user reporting the bug as soon as the bug is fixed. There will be no new code release - solution will be available to all other users when the regular new code release takes place.
 - **High:** solution is needed within weeks. The bug is fixed and the new code release is forced a.s.a.p., typically within 1 month.
 - **Medium:** solution is needed soon but not critical. The bug is fixed and the originator is informed. The new code release is forced at earliest convenience, typically within a few months.
 - **Low:** bug is not critical, waiting time of 1 - 2 years is acceptable. The bug is fixed and the originator is informed but there is no forced new code release. The version with the bug fix will become available to all users when the new code release is ready (typically the intervals between the new code releases is between 0.5 year and 2.0 years).

The bug report form is shown in Appendix A. The first part (problem report) is filled out by the code user. The second part (problem solution) is filled by the code developer. In case of small modifications (single file, <10 source code lines), the solution part of the bug report is the

documentation of the change. In cases when larger modification is needed, the code modification is recorded and documented separately within in the code modification procedure (section 3.2).

3.2 Code Modification Procedure

Every code modification, whether it originates from a user-supplied bug or request, or from the regularly scheduled code updates, is documented using the following procedure.

- The modification is registered in the code modification folder Z-MOD, as a sub-folder with an appropriate date.
- The files are stored in the subfolders OLD and NEW, which allows easy comparison and identification of all changes that were introduced
- The modification is described in the code modification form, including:
 - the old situation (e.g. problem detected by the developer, bug report, etc.)
 - the new situation - what was changed, how the bug was fixed, etc.
 - list of all source code files that were modified.
 - the motivation for the modification, including an example case (if possible), a case which caused the error (if applicable).

The code modification form is shown in Appendix B.

The activities described above take place after the modification was tested to verify the correctness of the introduced changes. If the modification involves changes to the text in the Code Manuals, then those changes are made in the appropriate Volumes of the Code Manuals, simultaneously with filling the code modification form.

The above described procedure provides an easy way to document all changes that are made to the code and keep control of the code versions. It has been applied since the code development started. Recently a second method was added using the software tool GIT. Currently both methods are applied independently for version control.

3.3 Code Development Procedure

The term code development is used here to mean significant code changes, caused by adding a new model to the code. The code development is performed following the recommendations set by the ANS standard on computer programming. ANSI/ANS-10.2-1988 [1], as discussed in [7]. As described in detail below, the development procedure is simultaneous with the corresponding updates of the code documentation. The documentation of the computer code SPECTRA is written following the recommendations set by the ANS standard on documentation of computer software, ANSI/ANS-10.3-1995 [3], as discussed in [8].

The code development procedure is based on the SPECTRA Software Administration Instruction [6]. The nodal point kinetics, which was added recently to the code, is used here as an example. The following procedure is applied.

- The theoretical basis and all governing equations are described in the Volume 1 of the Code Manuals (the “theory manual”). If applicable, the conversion from the source equation to the numerical approximation is given, so that the description provides an exact recipe for the code development. In the considered example, this was written as section 9.5 - “Nodal

Point Kinetics”. The source equations are converted to the matrix form used in the coding and the matrix coefficients are derived (see section 9.5.3 in Volume 1).

- Based on the theory manual, the equations are coded in SPECTRA source code in the model files. The coding related to the required additional input parameters is added in the input processing files.
- The required additional input parameters are described in the Volume 2 of the Code Manuals (the “user’s guide”). In the present example these are input records 750XXX, 750990, 750999, 751XXX, 755XXX, and 760XXX.
- The input procedures and input diagnostics are tested by:
 - checking if the output of the input values during the input processing stage is correct,
 - checking if error diagnostics is displayed correctly (by inserting deliberate input errors in the tested input file) and if all the warnings are displayed correctly (again by deliberately inserting inputs that lead to warning).
- When the input procedures are checked and it is certain that the input is read correctly, the verification and validation (V&V) of the new code commences. V&V of SPECTRA is performed following the recommendations set by the ANS guidelines for the V&V of scientific and engineering computer programs for the nuclear industry [4], as discussed in [9]. Relevant test cases are selected and the calculated results are compared to the available data. With increasing complexity of the new model, the number of test cases increases. In the considered example, four test cases were selected, including three verification tests, where the code results are compared to the exact solution, and one validation case, where the code results are compared to the results obtained with a 3D neutronics code.
- The results of the test cases are described in the Volume 3 of the Code Manuals (the “V&V manual”). In the considered example, this was written as section 3.9.4 - “Nodal Point Kinetics”, including:
 - the three verification cases, comparing the nodal kinetics to the theoretical solutions (see sections 3.9.4.1 through 3.9.4.3 in Volume 3), and
 - the validation case, comparing the code results with the results of the 3D neutronic code SERPENT.

In the present example the new model is relatively simple and only a few tests were considered sufficient. In the case of an earlier modification, extension of the point kinetics for circulating fuel, the model inserted was far more complicated and 147 tests were selected (see section 3.9.3.15 in Volume 3).

When all the testing is completed, the new code version passes the code modification procedure, described in section 3.2. The documentation is reviewed by the one of the members of SPECTRA Support Team [6] and, if needed, by a specialist in the field (for example, the nodal kinetics model was reviewed by a senior neutronic expert from NRG, Steven van den Marck).

3.4 Code Compilation and Quality Check Procedure

Code compilation is done using two different FORTRAN compilers. Currently the compilers used are:

- Intel compiler (Intel(R) MPI Library 2019 Update 5 for Windows, Target Build Environment for Intel(R) 64 applications, Copyright 2007-2019 Intel Corporation.)
- Lahey-Fujitsu compiler (LF Fortran 7.8)

The Intel compiler gives faster running executable code, therefore it is used as a primary tool for compiling the code. The LF Fortran compiler gives sometimes better diagnostics (more warning

messages), therefore it is used as an alternative tool for a code quality check. For the Linux version only the Intel compiler is used.

The code quality checks are conducted as follows

- Perform compilation using two different compilers. All errors and all warning messages must be eliminated.
- Perform code quality check using dedicated software. The Forcheck program [10] is being used. The program performs an analysis of Fortran source code or separate Fortran program units. It detects more anomalies than most compilers do. All error messages must be eliminated. All warnings must be carefully reviewed and if possible eliminated.

3.5 Code Release Procedure

The code release procedure is based on the SPECTRA Software Administration Instruction [6]. The following procedure is applied.

- The installation test is run. This is the Establishment of Flow Problem (EFP). The test is described in Volume 3. The results are compared to the earlier code version results.
- All the qualification tests are run. Results are compared to the earlier code version results. Currently there are two sets of qualification tests:
 - General qualification tests (folder \Z-INPUTS\Qualification-tests\)
 - HTR-specific qualification tests (folder \Z-INPUTS\Qualification-tests-HTR\)

An automatic verification procedure, described in Volume 2, is used. The automatic verification indicates for which tests the results are identical and for which the results are different. All differences in results are then evaluated manually. The difference may come from:

- Round-off errors. Those are typically seen in the least significant digits in the output file and do not show up if the results are plotted as graphs.
- Code modification. For example if a more accurate procedure is implemented in the code the new results will be different (more accurate) than the old results.
- There may be new input parameters and/or new output parameters, appearing in the output file.

The above differences are accepted. Other differences must be investigated and solved.

- When all eventual differences are resolved, the code is accepted and the following activities are performed:
 - A new number is assigned and the code is recompiled. Since then the new code version appears in every output file.
 - All the qualification tests are re-run with the new code version (and the new version identifier). The new output file become the new reference cases for a future version.
 - The installation test is run on Windows and Linux. A check is made if the that the results are different. Usually there are no differences, which means that the only difference showing up in the output file is the code version identifier, e.g.:

```
=====
0318:
0319: =SPE= SPECTRA Version 22-04, Apr. 2022, Windows
0320:
0321: Sophisticated
0322: Plant
0323: Evaluation
0324: Code for
0325: Thermal-hydraulic
0326: Response
0327: Assessment
0328:
0329: Validity: 12/2025, Applicability: AppC = 0000
0330:
0331:
0332:
0333: EFP - Establishment of Flow Problem
0334:
=====
0318:
0319: =SPE= SPECTRA Version 22-04, Apr. 2022, Linux
0320:
0321: Sophisticated
0322: Plant
0323: Evaluation
0324: Code for
0325: Thermal-hydraulic
0326: Response
0327: Assessment
0328:
0329: Validity: 12/2025, Applicability: AppC = 0000
0330:
0331:
0332:
0333: EFP - Establishment of Flow Problem
0334:
```

Sometimes a small differences may be encountered occasionally on less significant decimals, due to roundoff errors on different systems.

- The release notes are prepared, which contain the following information:
 - Unique release version number to identify the release.
 - List of folders and main files that are included in the delivery.
 - Platform hardware and software requirements.
 - List with a general overview of new features and fixes that are included.
 - Contact details of the SPECTRA Software Administrator (name, e-mail address, phone, postal address).

The code release form is shown in Appendix C.

4 Hardware and Software Requirements

4.1 Operating Systems

SPECTRA is written in FORTRAN-77 and was developed for the Windows and Linux operating systems. Two code versions are available and may be distributed to the users:

- Windows version,
- Linux version.

The differences in the source code (which are very limited) are clearly marked:

- In the Windows version, the Linux specific lines are preceded by the text C-LINX in the source code and thus appear as comment lines.
- In the Linux version, the Windows specific lines are preceded by the text C-WIND in the source code and thus appear as comment lines.

The post-processing tool, Visor, is available only for the Windows platform.

4.2 Requirements

SPECTRA can be installed on any modern computer. The memory size is not a critical issue for the present-day computers. A preferred option is to use a computer with a fast processor, to reduce the running times.

The user obtains the executable code, ready to be run on a Windows or Linux platform. There is no need for the user to have a FORTRAN compiler. There is no need for any additional software.

Literature

- [1] "Recommended Programming Practices to Facilitate the Portability of Scientific and Engineering Computer Programs", ANSI/ANS-10.2-1988, Revision of ANSI/ANS-10.2-1982, Prepared by the American Nuclear Society Standards Committee Working Group 10.2, Published by the American Nuclear Society, 555 North Kensington Avenue, La Grange Park, Illinois 60525 USA, April 1988.
- [2] W.H. Press, S.A. Teukolsky, W.T. Wetterling, B.P. Flannery, "Numerical Recipes in FORTRAN. The Art of Scientific Programming", Second edition, ISBN 0-521-43064-X, Cambridge University Press, 1992.
- [3] "Documentation of Computer Software, an American National Standard", ANSI/ANS-10.3-1995, Prepared by the American Nuclear Society Standards Committee Working Group 10.3, Published by the American Nuclear Society, 555 North Kensington Avenue, La Grange Park, Illinois 60525 USA, April 1995.
- [4] "Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry", ANSI/ANS-10.4-1987, Prepared by the American Nuclear Society Standards Committee Working Group 10.4, Published by the American Nuclear Society, 555 North Kensington Avenue, La Grange Park, Illinois 60525 USA, May 1987.
- [5] D.E.C. Scherpenzeel, "Controlled development and maintenance of software for numerical analysis", NRG Management system document CS-MA-OD-021456.
- [6] D.E.C. Scherpenzeel, "SPECTRA Software Administration Instruction", NRG report K6223/22.249713, November 2022.
- [7] D.E.C. Scherpenzeel, M.M. Stempniewicz, "Programming Rules Applied in Development of SPECTRA - an Overview Report", NRG-K6223/22.247911, December 2022.
- [8] D.E.C. Scherpenzeel, M.M. Stempniewicz, "Documentation of SPECTRA - an Overview Report", NRG-K6223/22.247910, December 2022.
- [9] D.E.C. Scherpenzeel, M.M. Stempniewicz, "Verification & Validation of SPECTRA - an Overview Report", NRG-K6223/22.247912, December 2022.
- [10] https://www.cs-software.com/software/forcheck/forcheck_info.html

Appendix A Bug Report Form (QA-SPE-004)

Part 1 of the bug report form - problem report - to be filled by the user:

Bug Report	QA-SPE-004
Date:	yyyy-mm-dd

Bug Report

TITLE: *Title text*

PART 1: PROBLEM REPORT

PROBLEM DISCOVERED

COMPANY	USER NAME
<i>Company name</i>	<i>Name</i>

PROBLEM DESCRIPTION

A short description of the problem

PRIORITY

- VERY HIGH
- HIGH
- MEDIUM
- LOW

- **Very high:** solution needed a.s.a.p., within days. The bug-fix code version (old code release with the bug fixed) will be delivered to the user reporting the bug as soon as the bug is fixed. There will be no new code release - solution will be available to all other users when the regular new code release takes place.
- **High:** solution is needed within weeks. The bug is fixed and the new code release is forced a.s.a.p., typically within 1 month.
- **Medium:** solution is needed soon but not critical. The bug is fixed and the originator is informed. The new code release is forced at earliest convenience, typically within a few months.
- **Low:** bug is not critical, waiting time of 1 - 2 years is acceptable. The bug is fixed and the originator is informed but there is no forced new code release. The version with the bug fix will become available to all users when the new code release is ready (typically the intervals between the new code releases is between 0.5 year and 2.0 years).

Part 2 of the bug report form - Problem solution - to be filled by the SPECTRA developer

Bug Report	QA-SPE-004
Originator: <i>Name</i>	<i>Signature</i>
Reviewer: <i>Name</i>	<i>Signature</i>

PART 2: PROBLEM SOLUTION

PROBLEM SOLUTION

A short description of the problem solution.

Modified files:

List of all source files that were modified, including:

- Package identifier, e.g.: =RT=
- Source file name, e.g.: RADTRA.POR

All modified files are stored together with this document, in the sub-folders: "OLD" and "NEW".

BUG REMOVED IN THE FOLLOWING CODE VERSION

Code version where the bug is removed.

Appendix B Code Modification Form (QA-SPE-005)

Modification Report	QA-SPE-005
Originator: <i>Name</i>	<i>Signature</i>
Reviewer: <i>Name</i>	<i>Signature</i>

Modification of: *DATE*.

TITLE: *Title text*

OLD

A short description of the old situation, problem (if applicable).

NEW

A short description of the new situation, problem solution (if applicable)

Modified files:

List of all source files that were modified, including:

- Package identifier, e.g.: =RT=
- Source file name, e.g.: RADTRA.FOR

All modified files are stored together with this document, in the sub-folders: "OLD" and "NEW".

Motivation:

Motivation for the modification, including example case (if possible), a case which caused error (if applicable).

Appendix C Release Notes Form (QA-SPE-006)

SPECTRA Release Notes	QA-SPE-006
Originator: <i>Name</i>	<i>Signature</i>
Reviewer: <i>Name</i>	<i>Signature</i>

Date

SPECTRA Release Notes

Code Version

- Code version released: *Version full name and number*
- Location at sftp site (external users):
 - *Confidential (executable codes):*
 - *Location*
 - *Open (code manuals, additional open documents, input files):*
 - *Location*
- Location at NRG network drive (restricted to qualified users):
 - *Confidential (executable codes)*
 - *Location*
 - *Open (code manuals, additional open documents, input files):*
 - *Location*

Updates compared to earlier version

List of updates

User Requests for Future Versions

List of known requests

SPECTRA Software Administrator

*Contact details of the SPECTRA Software Administrator
(name, e-mail address, phone, postal address)*