TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Nukleartechnik

Further Developments of Multiphysics and Multiscale Methodologies for Coupled Nuclear Reactor Simulations

Armando Miguel Gómez Torres

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor–Ingenieurs
(Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. Wolfgang A. Wall

Prüfer der Dissertation:

1. Univ.-Prof. Rafael Macián-Juan, PhD.
2. Univ.-Prof. Dr. Kostadin Ivanov,

   Karlsruher Institut für Technologie.

Die Dissertation wurde am 30.09.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 22.11.2011 angenommen.

Gedruckt mit Unterstützung

des Deutschen Akademischen Austauschdienstes

A Dios.

A mi esposa Laura.

A mis hijos Verónika y Johannes Armando.

A mis Padres María y Armando.

A mis hermanos Maribel, Jesús Pablo y María Tonantzin.

ACKNOWLEDGEMENT

This thesis would not have been possible unless the support of several people and Institutions.

First of all I would like to thank to the National Institute for Nuclear Research in Mexico (ININ: Instituto Nacional de Investigaciones Nucleares) and especially to M. en C. Raul Ortiz Magaña, Dr. Luis Carlos Longoria and Dr. Javier Palacios who always believed on me and gave me the opportunity to come to Germany to study my PhD.

I would like to show also my gratitude to the German institutions that have made this possible:

- The German Academic Exchange Service (DAAD), for all the financial support for the development of this thesis. Especially thanks to Brigitte Basu who always had a prompt answer and Frederick Stamm.

- The Institute for Neutron Physics and Reactor Technology of the Karlsruhe Institute of Technology (INR-KIT) for giving me the opportunity to participate in international projects and gave me a place to work. I owe my deepest gratitude to Dr. Victor Hugo Sánchez Espinoza. He opened to me the doors of Germany, trusted in me, and had always time for technical discussion, but above all, I sincerely thank him for his invaluable friendship and support.

- The Institute for Nuclear Technology at the Technical University Munich (TUM), in particular I am very grateful with Prof. Rafael Macián Juan that accepted to be my thesis supervisor.

- The Institute of Safety Research of the Helmholtz-Zentrum Dresden-Rossendorf for all the support with the use of DYN3D. Especial thanks to Dr. Sören Kliem, Dr. Ulrich Rohde and Mr. Andre Gommlich. They were not only excellent colleagues but also became very good friends.

It was an honor for me to have Prof. Kostadin Ivanov as a co-referent of my thesis. He had always time for my questions and for fruitful discussions.

I am indebted to many of my colleagues at KIT that supported me in the technical and administrative tasks. I specially thank Uwe Imke and Mrs. Petra Klug and to the group of friends (LM), particularly Wadim Jäger, Christoph Hartmann, Jorge Pérez, Javier Jiménez, Manuel Calleja, Rodrigo Gómez, Roberto López and Aleksandar Ivanov.

Last but not least, I want to say thanks to my lovely wife Laura and to my mischievous children Veronika and Johannes Armando. They are the source of inspiration that encouraged me to look always forward.

ABSTRACT

One of the main subjects related with the use of nuclear energy is Nuclear Safety. Nowadays, safety analyses are commonly carried out based on a Best-Estimate (BE) approach, trying to simulate the physical phenomena taking place in the core, the coolant loops and the balance of plant as accurately as possible. In order to achieve the most realistic description of the neutron flux distribution and its coupling to the thermal-hydraulic phenomena within the core, advanced multidimensional reactor dynamics codes have been developed and validated in the last decades. These state-of-the-art codes are able to predict, for instance, non-symmetrical core power perturbations and they can calculate safety margins more accurately than the former developments (based in point kinetics), by using 3D core models with a spatial resolution at fuel assembly level for a wide range of operational transients and postulated accidents. Such a level of approximation is acceptable to predict most safety-relevant variables, but there are also some important variables for safety, which must be evaluated based on local pin-level conditions, e.g. maximal cladding and fuel centreline temperatures.

This dissertation has followed two goals that extend currently used nodal reactor simulations at the fuel assembly level to heterogeneous reactor simulations at fuel rod level for detailed design and safety evaluations of nuclear reactors.

The first one considers the extension of the pin power reconstruction method used in the reactor dynamics code DYN3D in connection with nodal diffusion solutions. The flexibility of the new development allows local refinement in the spatial mesh for specific regions of interest (where a local perturbation occurs) or even having a whole core with pin-by-pin resolution. A detailed description of the integration of this extended version of DYN3D in the European Nuclear Reactor Simulation Platform (NURESIM) is also presented. This integration is a step forward in the direction of two-level coupling with a subchannel analysis code, which is one of the major objectives of NURESIM platform.

The second one focuses on the development of a novel two-way pin-based coupling of the simplified transport (SP_3) version of DYN3D with the subchannel code SUBCHANFLOW. The new coupled code system (DYNSUB) allows for a more realistic description of the core behaviour under steady state and transients conditions. The details of the internal coupling approach of both codes together with the implementation as well as selected results of the verification and validation work are presented and discussed. The comparison of the results predicted by DYNSUB with the ones of coarser coupled solutions have shown important deviations in the local safety parameters demonstrating the novel capabilities of the developed coupled system DYNSUB. The implications of such deviations for the assessment of the safety features of nuclear reactors are discussed.

Finally, further work related to physical model developments and validation for the improvement of DYNSUB is proposed.

KURZFASSUNG

Einer der wichtigsten Punkte in Bezug auf die Nutzung der Kernenergie ist die damit verbundene Reaktorsicherheit. In der Regel werden zur Durchführung von Sicherheitsanalysen sogenannten „Best-Estimate“ Methoden angewandt. Dabei wird versucht, die physikalischen Effekte und Phänomene im Reaktorkern, in den Kühlkreisläufen und den zugehörigen Systemen so genau wie möglich zu simulieren. Um eine realistische Beschreibung der räumlichen Neutronflussverteilung zu berechnen und deren Einfluss auf thermohydraulische Phänomene im Reaktorkern zu berücksichtigen wurden in den letzten Jahrzehnten fortschrittliche mehrdimensionale Reaktordynamikprogramme entwickelt und validiert. Diese Programme sind u.a. in der Lage asymmetrische Kernleistungsdichten detailliert zu berücksichtigen. Sicherheitsmargen können durch die Verwendung von 3D Kernmodellen mit räumlich aufgelösten Brennelementen für eine große Auswahl von Betriebstransienten und postulierten Störfällen genauer bestimmt werden als mit herkömmlichen Punktkinetikmodellen. Dieser Grad der Vereinfachung ist akzeptabel um die meisten sicherheitsrelevanten Parameter zu berechnen. Für einige wichtige Sicherheitsparameter, z.B. die maximale Hüllrohr- und Brennstofftemperatur, ist es aber notwendig, den Reaktorkern bis auf Brennstabebene räumlich aufzulösen.

In dieser Arbeit wurden zwei Ziele verfolgt, um die existierenden nodalen Methoden so weiterzuentwickeln, dass detaillierte Design- und Sicherheitsbewertungen mit einer räumlichen Auflösung bis zur Brennstabebene durchgeführt werden können.

Das erste Ziel beinhaltet die Erweiterung der „Pin Power Reconstruction Methods“ im Zusammenhang mit nodalen Diffusionsapproximationen, wie sie in dem Reaktordynamik-programm DYN3D zur Anwendung kommen. Durch die neu gewonnene Flexibilität der Weiterentwicklung können Bereiche von besonderem Interesse (Bereiche mit lokalen Störungen) räumlich genauer aufgelöst werden bzw. der gesamte Reaktorkern kann Brennstab für Brennstab dargestellt werden. Zusätzlich wird die Integration der weiterentwickelten DYN3D Version in die NURESIM Plattform (Europäische Reaktorsimulationsplattform) durchgeführt. Diese Integration ist ein Schritt in Richtung von „Zwei-Wege-Kopplungen“ mit Unterkannalprogrammen, welche ein wichtiges Ziel der NURESIM Plattform darstellt.

Das zweite Ziel betrifft die Entwicklung einer neuen, „Zwei-Wege“ Kopplung von der DYN3D Version mit vereinfachter Transportlösung und dem Unterkannalprogramm SUBCHANFLOW. Das neu entwickelte Programmsystem, DYNSUB, erlaubt eine realistischere Beschreibung des Reaktorkernverhaltens während stationärer und transienter Vorgänge. Die Einzelheiten des internen Kopplungsansatzes zwischen beiden Programmen sowie deren Integration wird präsentiert und erläutern Ausgewählte Ergebnisse zur Verifikation und Validierung werden ebenfalls gezeigt und diskutiert.

Deutliche Unterschiede konnten zwischen DYNSUB und gekoppelten Programmen mit geringerer räumlicher Auflösung für lokale Sicherheitsparameter festgestellt werden. Dies demonstriert die Fähigkeiten des neuen gekoppelten Systems DYNSUB. Die Auswirkungen dieser Unterschiede auf die Bewertung von Sicherheitseigenschaften von Kernreaktoren werden diskutiert. Des Weiteren werden Schritte zur Weiterentwicklung und Validierung des Programmsystems DYNSUB vorgeschlagen.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

1.1 Motivation of the Dissertation

1.1.1 Nuclear energy facts

Nuclear generation began more than 50 years ago. Some two-thirds of world population lives in nations where nuclear power plants are an integral part of electricity production and industrial infrastructures. Half the world's people live in countries where new nuclear power reactors are in planning or under construction. Rapid global expansion of nuclear power would require an acceleration of existing strategies and policies for fulfilling the rising electricity production expectations.

From the late 1970s to about 2002 the nuclear power industry suffered a certain decline and stagnation. Many reactor orders from the 1970s were cancelled and few new reactors were ordered and the number of those coming on line from the mid 1980s little more than matched retirements, although capacity increased by nearly one third and output increased up to 60% due to power up-rates and improved load factors. The share of nuclear in world electricity from mid 1980s remained fairly constant at 16-17%. The uranium price dropped accordingly, and also because of an increase in secondary supplies. Oil companies which had entered the uranium field bailed out, and there was a consolidation of uranium producers.

However, by the late 1990s the first of the BWR third-generation reactor, the Kashiwazaki-Kariwa 6 - a 1350 MWe Advanced BWR (ABWR) - was commissioned in Japan. This was a sign of the recovery to come.

In the new century several factors have contributed to the revival of the nuclear energy option: the world-wide increase of the electricity demand, specifically in emerging countries, the awareness of the importance of security of supply, and, finally, the need to limit carbon dioxide emissions due to growing concerns about global warming.

In 2004, the first PWR third-generation unit was ordered in Finland: a 1600 MWe European PWR (EPR). Two similar units are planned for France as the first of a full fleet replacement. In the USA the 2005 Energy Policy Act provided incentives for establishing a new-generation of power reactors with enhanced safety and economic performance.

Furthermore, the Generation IV International Forum (GIF) was initiated in 2000 and formally chartered in mid 2001. In 2002 GIF announced the selection of six reactor technologies which they believe will represent the future shape of nuclear energy. These were selected on the basis of being clean, safe and cost-effective means of meeting increased energy demands on a sustainable basis, while being resistant to diversion of materials for weapons proliferation and secure from terrorist attacks. All six systems could offer a closed fuel cycle to maximize the resource base and minimize high-level wastes to be sent to a repository. Three of the six are fast neutron reactors (FR), one can be built as a fast reactor, one is described as epithermal, and only two operate with thermal neutrons like today's plants. Only one of them is cooled by light water, whereas two are helium-cooled and the others have lead-bismuth, sodium or fluoride salt coolants. The latter three operate at low pressure, with significant safety advantages. The last one has the uranium fuel dissolved in the circulating coolant. Temperatures range from 510°C to 1000°C, compared with less than 330°C for today's light water reactors, and

this means that four of them can be used for thermochemical hydrogen production and other industrial uses that require high temperatures.

In Asia and some of the former Soviet Union states different types of nuclear power plants (NPP) are being built and the construction of a large number of NPPs is foreseen to meet the energy demands of their continuously growing economies. For example in China alone a six fold increase in nuclear power capacity by 2020 is planned. There, more than one hundred large NPP units are being considered and the decisions to build them are supported by credible political determination. A large portion of these NPPs are of the latest western design belonging to generation III and III+.

The history of nuclear power which started scientifically in Europe, blossomed commercially in the UK and USA with the latter's technological might, and languished for a few decades, has recovered today a new growth surge in east Asia [WNA20101].

Today nearly 440 nuclear reactors produce electricity around the world. More than 15 countries rely on nuclear power for 25% or more of their electricity. In Europe and Japan, the nuclear share of electricity is over 30%. In the U.S., nuclear power generates 20% of electricity.

1.1.2 Background

One of the main subjects related to the use of nuclear energy is nuclear safety. In order to achieve optimum safety, nuclear power plants have been designed following to the “defence-in-depth” approach, in which the multi-barrier concept plays a central role. The key aspects of this approach are listed below [WNA22010]:

- High quality design & construction,
- Equipment which prevents operational disturbances or human failures and errors developing into problems,
- Comprehensive monitoring and regular testing to detect equipment or operator failures,
- Redundant and diverse systems to control damage to the fuel and prevent significant radioactive releases and
- Provisions to confine the effects of severe fuel damage (or any other problem) to the plant itself.

The main goal of these technical and administrative measures is to assure that the safety goals (core sub-criticality, core cooling, confinement of radioactive material, radiation protection) are fulfilled under any circumstances during normal operation and off-normal states and accidents. These measures can be grouped in Prevention, Monitoring, and Mitigation.

A high safety level is achieved by inherent safety characteristics of the core combined with a well balanced set of passive and/or active safety systems. The inherent safety characteristics of the core are determined by its neutronic design. Thus, the core has to be designed selecting the material composition, the geometry and dimensions so that a sufficient negative fuel tem-

perature (Doppler effect) and void reactivity coefficient is attained to prevent an uncontrollable increase in reactor power. In addition, a redundant core shut-down system is needed to control the fission process in the core and slow it down or step it when necessary. To ensure short and long-term core coolability in all design-basis conditions, an emergency core cooling system (ECCS) is a fundamental part on any nuclear reactor design, which has the mission of removing the generated heat within the reactor and of transferring it to the ultimate heat sink.

1.1.3 Applicability

The safety analysis of nuclear power plants requires a deep understanding of underlying key-physical phenomena that determine the integrity of the physical barriers preventing fission product release, e.g. fuel pellet, fuel rod cladding, pressure piping system, RPV, containment, etc.

At the beginning of the commercial use of nuclear energy for electricity generation about fifty years ago, very conservative safety margins were introduced due to a lack of understanding of the important mechanisms leading to the degradation of safety barriers. Those safety margins set limits on the operation of the nuclear power plants, thus causing increases in the cost of electricity production. The recent availability of powerful computers, together with the continuing accumulation of operational experience and improvements in the physical understanding of plant behaviour at all levels of detail motivate the reconsideration of traditional over-conservative approaches to nuclear safety analysis. Benefits are foreseen in terms of reducing operational costs, improving the common understanding of nuclear safety and of design/operating conditions and, definitely, establishing a basis for advancing the technology [CRISSUEV1] [CRISSUEV2].

Nowadays safety analyses can be carried out based on a best-estimate (BE) approach, meaning that the Thermal-hydraulic (TH) phenomena are simulated as accurately as possible (according to present knowledge). If BE analyses are used for plant licensing purposes, regulators require that the analyses be accompanied by uncertainty evaluation to reveal the uncertainty bands bounding the calculated parameters.

In thermal-hydraulic analyses the power distribution in the fuel rods is usually explicitly specified, for instance by providing time-dependent functions. It can also be obtained from simplified neutron kinetics models simulating, in most cases, only the transient overall core reactivity feedback responses (considering the core as a “point”: point kinetics) or at best the core transient axial responses (one-dimensional kinetics). Consequently, by employing these methodologies the simulation of a detailed spatial 3D core power distribution is not possible. The practice of using 0D or 1D neutronic behaviour modelling requires the application of additional conservatisms when specifying so-called power factors, as they also have to include the uncertainties due to the loss in dimensional resolution of the transient’s true spatial power distribution. Therefore, conservative and unfavourable core power distributions are usually applied to the thermal-hydraulic analysis so as to ensure that the fuel rods will experience more severe conditions than might be expected to prevail in the real scenario.

To achieve a most realistic description of neutron flux behaviour and its relation with the thermal hydraulic physical phenomena within the core, advanced multidimensional reactor

kinetics codes have been developed and validated in the last decades, which are able to predict non-symmetrical core power distributions resulting from most safety relevant perturbations in a more realistic manner than the point kinetic models can provide [Ivanov21999]. Such perturbations can have their origin, for instance, in control rod movements or changes in the thermal-hydraulic conditions of the core coolant-moderator arising from the failure of systems or components, including variations in the concentration of soluble boron in the liquid phase used as a reactivity control mechanism in PWR operation. The induced power variations can include both global core-wide changes but also local changes restricted to only a few fuel assemblies or even to intra-assembly power distribution (so-called pin-by-pin distributions). The original core power level and operational history (burn-up) have also profound influences on the reactor's power response.

In order to simulate the nuclear system response to those perturbations, reactor dynamic codes make use of nuclear cross-section data, providing neutron energy dependent probabilities for specific nuclear reactions to occur. Those data are used as a basis for determining the core state, and include neutron transport theory based corrections to account for the effects of burnable absorbers and flux heterogeneities near fuel assembly limits and core outer regions (reflector sections). Core thermal-hydraulic conditions employed for the simulation of core behaviour have usually been obtained from rather simple thermal-hydraulic models which try to reproduce the actual conditions in the core itself, with adequately specified boundary conditions at core inlet and outlet or, at best, from simple models to simulate the RPV internal flow paths before and after the core.

The continuous advances of computer technology have fostered the world-wide development and use of BE coupled 3D neutron kinetics/thermal-hydraulic system analysis codes. Their main advantage compared with more traditional computer based conservative analysis tools is their capability of describing in a more realistically physical manner the local core neutronic and thermal-hydraulic feedback processes, which determine the core/plant coupled dynamic interactions for a wide range of operational transients or postulated accidents (e.g. reactivity initiated transients, turbine trips, load rejection, main steam line break, anticipated transients without scram) and, hence, to yield more accurate safety margins [Langenbuch], [Ivanov2007], [Bousbia2007], [CRISSUEV1].

In recent years, the state-of-the-art core analysis methodology has been based on two-energy-group neutron diffusion theory, which solves the three-dimensional time-dependent neutron diffusion equation by means of nodal neutronics methods and two-phase flow homogenized thermal-hydraulics models. Typical applications are, for example, steady-state and transient analyses for LWRs, such as PWR Steam Line Breaks [Ivanov1999] or BWR Turbine Trips [Solis2001]. The most detailed spatial resolution of such coupled calculations has been typically the fuel assembly level.

This degree of spatial resolution may not be acceptable, however, if the safety-relevant parameters that determine accident consequences, such as fuel rod enthalpy, departure from nucleate boiling ratio (DNBR), burn-out, maximum fuel rod cladding temperature, fuel rod centre-line temperature, etc., must be evaluated based on local conditions, i.e. in terms of a single rod (pin) response rather than based on an assembly-wise response.

Currently, few codes are able to increase the spatial resolution and resolve selected fuel assemblies with a finer spatial approximation at a pin-subchannel level during a coupled transient calculation. This has been attempted mainly by the use of pin power reconstruction methods such as TRAC-BF1/NEM/COBRA-TF [Solis2002] and RELAP/PANBOX2 [Jackson1995] and [Jackson1999].

Further developments beyond current methodologies are necessary to perform whole core transient calculations at a "pin-subchannel level", taking into account all feedback mechanisms between the neutronics and the thermal hydraulics.

Traditional lower resolution approaches usually yield narrower operation margins because of the conservatism introduced in the coarser spatial detail. Therefore, there is an urgent need to extend the model capability of BE coupled system codes for safety assessment. Better safety margin estimation will contribute to improve operation and support for more efficient power generation (higher power densities following power up-rates, for instance). Additionally, local resolution will enable the calculation of more detailed boundary conditions for fuel behavior codes, so that fuel damage can be better estimated.

1.2 Objectives and General Goals

The main goal of this PhD work is the further development of multiphysics/multiscale methodologies for design and safety evaluations of current and innovative reactor systems by extending the coupling schemes between neutron physics and thermal-hydraulics simulations to finer levels of spatial resolution. Verification and validation of these developments are also an important part of the dissertation.

The ultimate contribution of this thesis is to improve and extend the "nodal" reactor simulations at fuel assembly level to heterogeneous reactor simulations (static and transient) at pin level, making possible the prediction of local safety relevant parameters, e.g. DNB, maximal cladding and centreline temperature, etc., which will reduce the conservatism affecting current methodologies.

1.3 Structure of the Dissertation

The dissertation is divided in 4 chapters. Following this introductory chapter, the state of the art in the coupling of multiphysics and multiscale methodologies together with a brief explanation of the actual available options in the field of coupled developments is described in Chapter 2.

In Chapter 3, as a first approach towards the pin level coupling, extensions to the pin power reconstruction method already implemented in DYN3D are described. The new developed version, capable of calculating pin power distributions in several fuel assemblies or even in the whole core is also tested via a boron dilution transient and a control rod ejection transient. In addition, the use of the new nuclear reactor simulation platform NURESIM, which is based in the open source SALOME platform, with powerful pre-and post processor capabilities and the integration of an extended version of DYN3D in the NURESIM platform is presented.

Chapter 4 contains the original and main development of this dissertation. A real two way pin base coupling of DYN3D-SP3 and SUBCHANFLOW allowing a more realistic description of the core behaviour under steady state and transient conditions is extensively presented and discussed. The full picture of the internal coupling of the codes in steady state and in transient together with the implementation and some representative results are depicted too.

The conclusions, applicability, advantages and limitations of the novel development, together with other potential extensions and improvements of future research will be given in the Conclusions and in the Outlook.

2 State of the Art in Multiphysics/Multiscale Methodologies

2.1 Multiphysics Methodologies

The design and safety assessment of nuclear power plants (NPPs) are complex problems requiring knowledge from different fields such as fluid dynamics, heat transfer, structural mechanics, chemistry, neutronics, etc. Furthermore, these fields are in close interaction with each other, for example: neutronics/thermal-hydraulics, neutronics/pin-mechanics, thermal-hydraulics/pin-mechanics, radiation/structure, fluid/structure, etc. [Jimenez2010]. This multiphysical character of the nuclear technology was treated or described separately from the very beginning in different computer codes due to the limitations of the computer power.

At present, computer power has been substantially increased so that the development of coupled multiphysical solutions is the logical step to follow by integrating the individual codes in the most appropriate way. The main goal of such developments is to supply a more realistic description of key-phenomena for reactor design and safety reducing the degree of conservatism of legacy codes. In addition, the goal is to obtain accurate and validated solutions in a reasonable amount of CPU time. Different multiphysical coupling approaches are under discussion worldwide as it will be briefly discussed in the next sections. Hereby doing a coupling must take into account several aspects, for instance, the coupling approach (integration algorithm or parallel processing), the way of coupling (internal or external), the spatial mapping schemes, and the time synchronisation algorithms, among others [Ivanov2007].

Most of the coupling schemes relay on rigid spatial mappings between the neutronic and thermal hydraulic domains [CRISSUEV1] and [CRISSUEV2]. A novel concept for a very flexible multiphysics and multiscale coupling approach is being developed in the frame of the European Projects NURESIM and NURISP [Cacuci2006], and as a result of the US-Korea collaborative project focused in the development of a numerical nuclear reactor (NNR) [Weber2005] and [Sofu2007]. The goal is to provide validated numerical simulation tools for design safety evaluations that can be easily coupled on-demand by users according to the specific needs e.g. of industry, regulators or R&D institutions in a modern and user friendly simulation platform with powerful pre-and post processor capabilities like for instance SALOME [SALOME].

In principle, an almost complete set of physical phenomena important for design and safety may be considered in such code systems or platforms. Of course, the focus is on most evident and measurable phenomena such as the fission heat source, heat transfer mechanisms for core cooling, together with the thermal behaviour of the materials under extreme stresses that may determine the integrity of the safety barriers under normal and off-normal conditions. Because of that, nuclear thermal and safety analysis are focussed on phenomena such as neutron transport, thermal-hydraulics and thermo-mechanical behaviour of the fuels. A description of each branch will be shortly given.

2.1.1 Neutronics

The design and operation of a nuclear reactor is totally related with the capacity of predicting the distribution in space, energy and time of neutrons in the system. That can be done by solving the neutron transport equation or "Boltzmann equation".

The problem is in general very complex due to the very large number of paths and interactions (nuclear reactions) that a neutron can experience while moving through the system, but it can, in theory, be solved by inserting into the transport equation a complete set of appropriate cross sections, which represent the neutron interaction probabilities, together with the geometrical arrangement of the materials in the system. In practice, however, it is not so simple. First, the energy dependencies of the cross sections for absorption, scattering, fission, etc., are very complicate for some energy intervals and not completely known, and, second, the geometrical arrangement of the materials in the reactor is so complex that the Boltzmann equation cannot be solved in a reasonable time even with super computers. Thus, several strategies have been developed in order to find numerical solutions, for approximated forms of the transport equation [Glasstone1970].

Two big branches are then distinguished for dealing with the transport equation: deterministic methods and stochastic methods (Monte Carlo method) which will be introduced in the next subsections.

2.1.1.1 Monte Carlo method

Monte Carlo methods provide approximate solutions to a variety of mathematical problems by performing statistical sampling experiments. They can be loosely defined as statistical simulation methods, where statistical simulation is defined in quite general terms to be any method that utilizes sequences of random numbers to perform the simulation. Thus Monte Carlo methods are a collection of different methods that basically perform the same process. This process involves performing many simulations using random numbers and probability to get an approximation of the answer to the problem [Pengelly2002].

In neutron transport calculations, the applicability of the Monte Carlo techniques arises from the fact that the macroscopic cross sections may be interpreted as a probability of interaction per unit distance travelled by a neutron [Glasstone1970]. Thus, in the Monte Carlo method, a set of *neutron histories* is generated by following individual neutrons through successive collisions. The locations of actual collisions and the results of such collisions, for instance, direction and energy of the emerging neutron (or neutrons), are determined from the range of possibilities by sets of random numbers. The Monte Carlo approach is superior to the deterministic one, because of the exact geometry representation of the computational domain and of fewer approximations involved in the calculation, e.g. continuous in energy variation of microscopic cross sections, etc. By solving the neutron transport equation with Monte Carlo methods, uncertainties due to the limitation in the number of *neutrons histories* examined arise. Such uncertainties are random and nowadays procedures for reducing them (variances reducing techniques) have been explored and implemented..

Because of its nature, the method is able to provide a reference solution when enough neutrons are tracked. Unfortunately, the more *neutron histories* are considered, the more computational cost is needed. Thus, solving for instance a pin-by-pin problem can lead to unrealistic CPU loads. Furthermore, the lack of understanding on how to represent a time evolving source of neutrons makes the time dependent coupled calculations using Monte Carlo methods not yet feasible.

2.1.1.2 Deterministic methods

Deterministic Methods yield approximate solutions of the Boltzmann equation by means of a discretization of the domain under study. So, discretizations in space, angular direction and energy are normally used as a function of time. Depending on the numerical method for solving the Boltzmann equation, several approximations and methods have been studied, and can be classified in methods solving the integral form of the transport equation and methods solving the integro-differential form [Reus2008] and [Beckert2008]. Additionally to these methods the "diffusion approximation" has been widely used. A short description of the main assumptions in the numerical methods used worldwide is presented hereafter.

- **Methods used for transport's cell lattice calculations.**

In view of the geometrical and material complexity of the nuclear reactor core (large number of absorber rods and fuel elements, often of quite complex design, surrounded by a moderator), it is virtually impossible to obtain analytical solutions of the Boltzmann equation. Approximations such as neutron diffusion theory are no longer valid when the geometric scales involved are small compared with the neutron mean free path (e.g. fuel cell calculations) [Williams1971]. These difficulties can be overcome by the application of integral neutron transport theory of which two main kinds of methods can be distinguished:

  - ***Collision Probability Methods**:* the structure used in nuclear reactor cores is often relatively regular, so that each fuel element and its cladding together with the coolant and/or moderator can be seen as mesh elements called *cells*. Such *cells* are then arranged in a *lattice* in order to define a fuel assembly. In this way, the analysis of a *lattice* can be divided into *cells*, in each of which there are zones (one or more zones can be defined for each material of the *cell*). In these zones, based on reasonable assumptions about the shape of the neutron flux, it is possible to reduce the space and energy dependent integral equations to a set of purely energy dependent ones. Included in these energy-dependent equations, however, are quantities called *collision probabilities* (P_{ij}) which are geometry although not space dependent. They are interpreted as the probability for a neutron emitted isotropically in a zone i to undergo its first collition in zone j. The collision probabilities are in many instances of a universal nature; thus, once calculated, they can be used over and over again for different problems [Williams1971] and [Reus2008].

  - ***The method of characteristics:*** in this method the neutron transport equation is solved via a transformation to a Lagrangian coordinate system; i.e., to the frame-of-reference of the neutron in motion [Taylor2007]. The method of characteristics solves a neutron propagation problem instead of a neutron balance problem (as it is done with most of

the neutron transport methods). It is assumed that neutrons will travel in straight lines until experiencing collisions events, so that the spatial domain can be discretized into a set of straight linear characteristics along which all neutrons are assumed to travel. The method transforms a 2D or 3D transport equation problem to a 1D equation problem that can be solved by direct integration along the neutron propagation path. Procedures to determine the length and orientation of the neutron propagation paths (*ray-tracing* algorithm) are then, required. A fine-mesh spatial discretization together with a discrete ordinates approximation, for the angular domain, are usually used in order to define such *ray-tracing* algorithm. For all the trajectories the outgoing flux is computed, the contribution from this trayectory is added to the average flux on the region and the outgoing flux is used as the incoming flux for the next cell along the trayectory (trayectory sweep process) [Rabiti2006]. The ray-wise integration allows flexibility of the mesh shapes, i.e. the meshes can take any shape and mixture of shapes as in Monte Carlo method [Nam2005].

One disadvantage of such methods is the large amount of memory and computational resources needed. Because of this, it is not possible to apply these methods to real problems dealing with entire cores in 3D static or kinetic reactor simulations.

- **Methods used for transport's core calculations**.

These methods offer an option for modelling larger regions of the reactor (even the whole core). Of course simplifications in geometry and the use of multi-group theory [Glasstone1970] for the treatment of the energy dependence are part of the cost that must be paid. The angular dependence of the neutron flux can be treated numerically in different ways, being series expansion (Method of Spherical Harmonics) and discretization (Method of Discrete Ordinates) the most commons.

- ***Method of Spherical Harmonics:*** commonly known also as $\boldsymbol{P_N}$ method. The procedure is based on the expansion of the angular distribution of the neutron flux in a complete set of orthogonal functions, namely, the Legendre polynomials in one-dimension and or spherical harmonics in multi-dimensional problems. The expansions are truncated after a few terms in order to develop practical methods for solving the resulting form of the neutron transport equation (until order N) [Glasstone1970]. This method produces quickly a large number of unknown functions to be calculated, up to $(N+1)^2$. However assuming rotational symmetry the angular dependence can be simplified and the $\boldsymbol{SP_N}$ method (simplified spherical harmonics method) is obtained in which just $N+3$ unknown functions arise. In Chapter 4 some formalism about this method will be presented.

- ***The Discrete Ordinates Method:*** The essential basis of this method is that the angular distribution of the neutron flux is evaluated in a number of discrete directions instead of using spherical harmonics. By considering enough directions, it is possible, in principle, to obtain a solution of the transport equation to any desired degree of accuracy, subjected only to the limitations of the available computing power. In the solution of practical problems, a discrete energy variable is introduced (multigroup approximation), and a discrete space mesh is used for the spatial coordinates dealing to a full discrete treatment of the independent variables of the time-independent transport equation

(space, direction and energy). Special attention must be paid to the choice of the particular discrete directions, the approximation of the integrals over the direction variable and the approximation of the derivatives of the neutron flux with respect to the components of the angular variable appearing in the Boltzmann equation [Glasstone1970].

- **Methods used for solving the neutron diffusion equation.**

The basis of the neutron diffusion approximation assumes that the neutrons behave according to the gas law [Glasstone1970] and [Kennard1938], i.e., they move from regions of high density towards those of low density. Furthermore, the angle dependence of the scattering process is not considered, leading to an isotropic angular distribution [Glasstone1958]. The finite difference methods and "modern nodal methods" have been used and developed for solving the diffusion equation [Nam2005].

- ***Finite Difference methods:*** are basic numerical techniques in all areas of science and engineering in which the domain is subdivided in small intervals, and the derivative terms in the diffusion equation are approximated by means of finite differences. They have been taken up in reactor physics early on and have been a major workforce up to the 1970's. An analysis of their computational cost shows that acceptable accuracy is obtained by using a mesh spacing of the order of the smallest group diffusion length [Grossman2007]. This leads to heavy computational burdens because of the resulting tremendous number of mesh boxes, and thus unknowns to be solved for. In parallel, several computational techniques have been developed with the hope of reducing the computational storage and execution time. Thus methods like Finite Element Method (FEM) [Grossman2007] and fine mesh finite differences with a domain decomposition methodology through alternate dissections (recently developed at the UPM [Jimenez2010]) help to tackle the problem in some way.

- ***Modern Nodal Methods:*** they have taken a firm place in the current production codes for reactor design as a main computational engine. They use very coarse meshes resulting in dramatic reduction in computing time compared to the finite difference methods. They also attain very high accuracy by careful treatment in discretizing the diffusion equations to enforce neutron balance. Modern nodal methods are based on a large coarse mesh (of an assembly size) of so-called "nodes" whose properties are constant. The first step in nodal methods is homogenization (plus energy group condensation). The homogenization procedure provides equivalent constant properties in a node which is physically heterogeneous. This is a process that converts a physical system that is difficult to analyze to a mathematical system that is easier to analyze but gives equivalent solutions for important parameters like reaction rates of the node and multiplication factor of the core. The Nodal Expansion Method (NEM) and Analytical Nodal Method (ANM) are the most common methods currently used. A transverse integration is done in one or two of the space directions, leading to a simplified scheme in two or one dimensions respectively, with a polynomial solution (NEM) or even analytical (ANM). Details about the nodal method will be given in the next chapter. Some other methods do not use transverse integration like the Analytic Function Expansion Nodal Method (AFEN), instead the analytical solutions of the diffusion equation in three dimensions can be written in terms of a series in which each term satisfies the diffusion equation exactly, the expression is then truncated to eight analytical basis functions and solved in each node fixing the number of boundary condition to the

number of unknowns [Nam2005]. For minimizing memory requirements some acceleration methods can also be added to the nodal solution like the Coarse Mesh Finite Difference Acceleration (CMFD) or Coarse Group Rebalance Acceleration (CGR).

All the methods previously described are focused in getting an accurate enough description of the neutron flux in a reactor. Different methods may be used according to the compromise between computational burden and accuracy desired. Details in some of the methods used in this dissertation will be given in the following chapters.

2.1.2 Thermal-hydraulics

The evaluation of the safety of NPPs is closely related to the ability to determine the temporal and spatial distributions of the fluid thermal-hydraulic conditions along with associated effects from heat sources and heat sinks throughout the reactor coolant system, and especially in the core region. On-line measurements at different locations of the NPP primary and secondary systems can provide valuable information in this context, but important thermal-hydraulic details within, for instance, the fuel assemblies will not be revealed through such means. The established method to evaluate those complex conditions is by deployment of advanced numerical thermal-hydraulic simulation tools based on well validated physical and numerical solution models to predict the behaviour of nuclear power plants.

The physical phenomena to be simulated are very numerous and, some times, problem-dependent. Heat conduction, fluid dynamics, heat transfer for single and two phase flow, boron transport in liquid, non-condensable gases in the gas phase, special models (reflood, stratification, condensation, critical flow, counter current flow, etc.) conform a set of physical effects that must be taken into account for most analyses.

The models and types of computer codes are related to the problem to be solved and to the degree of detail desired. Based on that, three kinds of thermal-hydraulic solvers are usually distinguished and it will be briefly described in the next subsections.

2.1.2.1 System codes

System codes are very useful computer tools applied for NPP safety analyses and evaluations of nuclear plant responses to different kinds of transient. Their validation database is very large and covers a wide range of physical condition. Moreover the accumulated experiences from an ever-growing amount of applications provide comprehensive guidance for the code applications. In this kind of codes the reactor hydraulic system is modelled with one-dimensional fluid computational elements, interconnected by means of flow junctions to build the entire system. Special models to simulate the behaviour of coolant pumps, pressurizers, steam generators, etc…, are also included. Conservation of mass, momentum and energy are solved in such a 1D network by means of numerical methods that provide the time evolution of the thermal-hydraulic variables of the system. Some of these codes make use of a more sophisticated modelling for the reactor vessel and core. These models comprise a three-dimensional coarse mesh in cylindrical geometry (r,θ,z), with azimuthal sectors (θ) that can track the flow directions from each of the hot and cold lines, radial sectors (r) to define the flow in the downcomer, flow by-pass channels different core regions, and several axial (z) sectors that provide a refined simulation of the down and up-ward flow in the vessel.

Such a meshing is an approximation to the real geometry of the reactor vessel, but it can give a better description of the actual flow in the vessel than 1D simpler models can. In order to adjust the mass flow rates in the different flow paths modelled in the vessel, pressure loss coefficients must be adjusted to obtain the pressure drops as close to the actual vessel as possible.

All modern system codes are also able to simulate one- and two-phase flows both in steady state and transient conditions [Jimenez2010].

Despite considerable validation efforts carried out over the years, there are still areas of code applications that reveal limited experience, e.g. transients for which pressure wave propagation effects are important and for which strong 3-D flow and re-circulating flow formations occur in certain parts of a nuclear power plant. These more demanding kinds of flow conditions can have a great influence on the course of certain transients [CRISSUEV1].

2.1.2.2 Subchannel codes

Improving on the coarse meshes used by system codes, better modelling of the thermal-hydraulic conditions inside fuel assemblies in the vicinity of fuel rods is possible by means of sub-channel codes. The prediction of two-phase flow and boiling behaviour at sub-channel level is an issue of great interest in nuclear reactor safety studies. The distribution of pressure drop and void fraction inside a reactor fuel assembly depends on local thermal-hydraulic conditions, such as mass fluxes in fuel channels, pressure gradients between flow channels, arrangement of the rods, etc. With a validated sub-channel code, it is possible to simulate the thermal-hydraulic behaviour of a single fuel assembly, or even of the entire core [GomezR2010].

Sub-channel analysis codes offer a large set of possibilities for representing the physical behaviour of the flow ranging from a one phase “pseudo-fluid” model (equations dealing with the mixture liquid–vapour) until a multi-phase representation of the flow base on several fluid fields, e.g. liquid steam, liquid droplets entrained by the steam flow, bubbles entrained by the liquid, etc.. As expected, the more complex flow model used, the larger is the number of conservation equations and constitutive relations needed to close the system of equations, thus adding more complexity to the problem.

One of the coarsest approximations used by sub-channel analysis codes consists in consider that the fluid moves predominantly in the axial direction. The approximation assumes the reduction of the three-dimensional momentum equations to two equations, one for the flow in the predominant direction (axial) and the other one transversal to the axial one.

The reduction of dimensionality makes it necessary the introduction of cross flow, in order to take into account the interchange of mass in the transversal direction across the interfaces that separate the sub-channels in the fuel assemblies. The cross flow term in two-phase flow conditions (especially for BWR or during PWR’s transients) is modelled with three different components [Sadatomi1994]:

- diversion cross flow: caused in non-equilibrium flow conditions by transversal gradients of pressure (e.g., geometrical changes),

- void drift flow: caused by the displacement of vapour bubbles to regions with high fluid velocity in non-equilibrium flow conditions, and

- turbulent mixing flow: a result of inter-subchannel mixing caused by the turbulence of the fluid occurring in both equilibrium and non-equilibrium flows.

When the cross flow terms are neglected, the formulation is equivalent to considering one-dimensional closed channels, similar to simplified one-dimensional thermal-hydraulic models, but with a finer computational mesh resolution.

2.1.2.3 Computational Fluid Dynamic (CFD) codes

The task of obtaining solutions to the governing equations of fluid mechanics represents one of the most challenging problems in science and engineering. In most instances, the mathematical formulations of the fundamental laws of fluid mechanics are expressed as partial differential equations. The governing equations of fluid mechanics form a set of coupled, non linear partial differential equations which must be solved within an irregular domain subject to various initial and boundary conditions. Analytical solutions although are limited. Experimental fluid mechanics can provide some information regarding a particular flow parameter or coefficients volume-averaged (turbulent interchanges of mass, energy or momentum) to be used, for instance, in the subchannel or system codes. However, the limitation on the hardware (measurement techniques) and the difficulty in adequately simulating the phenomenon, as well as the extraction of measured data makes it an impractical means of obtaining such coefficients or empirical laws for many problems.

A technique that has gained popularity in recent years is Computational Fluid Dynamics (CFD). The fundamental bases of almost all CFD problems are the Navier–Stokes equations, which define any single-phase fluid flow [Kuzmin2010]. Improvements in computer hardware, resulting in increased memory and efficiency have made possible to solve the equations in fluid mechanics using a variety of numerical techniques. These advancements have stimulated the introduction of newer numerical techniques. Unlike experimental fluid mechanics, the geometry and flow conditions can be easily varied to obtain various design goals. The volume averaged coefficients are in this context not any more needed. The fundamental concept of numerical schemes is based on the approximation of partial derivatives by algebraic expressions that can be solved numerically using computers [Hoffmann2000].

Although the improvements in the computer power and in the efficiency of the numerical schemes used in the CFD codes, the integration of these codes in coupled systems results prohibitive due to the huge amount of computational resources needed. However, CFD tools can be used for validating the volume-averaged coefficients used in the system or subchannel codes, or in the calculation of more detailed boundary conditions in key parts of the system. Finally, the CFD models for two phase flow are still under development and not yet mature for industrial applications.

2.1.2.4 Direct Numerical Simulations (DNS) codes

Direct Numerical Simulations (DNS) is a technique in CFD in which the Navier-Stokes equations are numerically solved without any turbulence model. This means that the whole range of spatial and temporal scales of the turbulence must be resolved in the computational mesh, from the largest integral scale down to the smallest scale (Kolmogorov scale).

In the general perspective of two-phase flow modelling, the purpose of DNS is to go beyond the current limitations of two-phase flow simulations by studying basic physical mechanisms, thus obtaining detailed information not accessible through experiments and assessing, improving or determining averaged closure relations. However, due to the large computational cost required in such calculations, it is necessary to define the inevitably simple cases where DNS can be effectively used, as thoroughly discussed in [Jamet2010].

2.1.3 Fuel rod mechanics

In nuclear reactors, the fuel is exposed to very strong thermal, mechanical and radiation loads during its operational life, which cause significant changes to its microstructure altering its mechanical and thermal properties.

The material composition of the fuel also changes during burn-up resulting in several consequences for the fuel rod. From an economic point of view, a better fuel consumption allowing longer reactor operation periods without refuelling is desirable; however, there is a limit to how much a fuel rod can remain in the core. The fission reactions generate fission products and actinides in the fuel which affect the fuel thermal properties, and the released fission gases accumulate within the fuel crystalline structure in the grain boundaries. They eventually migrate to the gap causing the internal rod pressure to increase. In addition, cracking of the fuel pellets caused by thermal stresses and the accumulation of fission products, results in pellet-clad contact, which can increase heat transfer, reducing the fuel rod temperatures, but also increase the mechanical stresses in the clad, which could lead to clad failure. Under normal operating conditions and in case of accidents, fuel failures must remain extremely low.

There are two types of codes for the analysis of nuclear fuel behaviour, namely the simplified and the mechanistic codes. The first type solves the heat transfer equations and is able to calculate the deformations based in the properties of the materials given by input. The mechanistic codes are able to calculate directly those properties as a function of the operational conditions and the history of the fuel (position in the core, different burn-up conditions, hard or soft neutronics spectra, etc.). Usually the simplified codes are integrated as parts of system analysis codes that deal with core behaviour (neutronics or thermal-hydraulic) to predict fuel and clad temperatures. Temperature gradients are dominant in the radial direction and, for this reason; the heat transfer equation is solved in one single direction. Mechanistic codes can also be integrated but only for some specific applications. They offer heat transfer models from 1 to 3 dimensions, and include the capabilities of time dependent analyses for fast transients or for fuel burn-up calculations.

2.2 Multiscale Methodologies

A multiscale approach tries to describe all phenomena occurring in a physical system at different spatial scales and how they are connected, For instance, in a nuclear reactor, the reactor vessel's radius can be several meters long, an assembly is usually 20 cm, the fuel rod diameter is approximately 1 cm and the size of the bubbles in the coolant is of the order of millimetres. The fluid passing through all these elements is governed by the same physical laws, however, the hypotheses used in every scale are chosen in agreement with the effects that must be reproduced at a given scale.

The degree of detail or refinement, as expected, is strongly related to the computing power available. The wider scales have been explored and used for several years; methods have been verified and validated. More recent developments move always in the direction of understanding the most basic phenomena taking place at smaller scales.

2.2.1 Multiscale approach in the neutronic branch

The neutronic core behaviour can be described in a coarser manner (macro scale) by means of point kinetics approximations, in which the neutron flux shape does not change appreciably during changes in its magnitude, which essentially reduces the core modelling to a "point". The point kinetics approximation condenses (integrates) the space and energy dependence of the neutron flux in the Boltzmann equations to make it independent of space and energy (for a one-energy group reactor). In this manner, it is possible to follow the response of the reactor flux (power) in terms of global changes and has been widely used in safety analysis, in which the neutron flux conserves spatial symmetry in the core. However, local perturbations in the core cannot be followed by means of this approach. In a more detailed representation, but still at the macro scale level, the time-dependent neutron diffusion equation can be solved in 3D for spatially homogenized fuel assemblies giving more detailed information regarding the time and space evolution of the neutron flux.

The development of more heterogeneous fuel designs and the inclusion in them of, for instance, MOX fuels, burnable poisons, water holes, etc. have made necessary the transition from FA-based analysis to fuel rod-based neutronic simulations. Such step in the direction of a more detailed scale (meso-scale) has resulted in the development and use of more geometrically refined approaches such as pin-power reconstruction methods or transport solutions at pin-level.

2.2.2 Multiscale approach in the thermal-hydraulic and thermo-mechanical branch

The thermal-hydraulic behaviour of a nuclear reactor can be described also at different scales. The wider scale (macro-scale) is usually carried out by using system analysis codes already described before. They aim at reproducing the behaviour of the different components of a nuclear power plant (steam generators, primary and secondary loops, pumps, condenser, reactor vessel, etc.) without paying much attention to physical processes at scales smaller than the component main dimensions (except in the core). Specific models and simplifications for this level of detail considered are used and the components are usually interconnected at similar scales and to coarse inlet and outlet boundary conditions.

Descending in thermal-hydraulic spatial resolution scales, the heat removal of the single pins inside a fuel assembly considering the local phenomena at sub-channel scale is analysed and the effects of cross flow mixing between adjacent subchannels is also considered. Sub-channel codes and some coarse mesh CFD models are representative of this meso-scale level.

Finally, if the physical phenomena taking place at a very small spatial scale (Micro), e.g. boundary layer effects, turbulences, fluid temperature, void and velocity gradients inside a subchannel, etc., are of interest, fine mesh simulations with CFD codes can be used to simulate physical processes at very small scales. When physical processes may be determined by phenomena occurring at smaller scales, as for instance Departure from Nucleate Boiling (DNB) direct numerical simulations (DNS) can be used, for instance, to study the dynamics of the growth of a few bubbles with their eventual migration and bubble column formation near wall region [Jamet2010]. But as previously discussed, they are very computer resource intensive, which limits their applicability.

Table 2-I shows a classification of codes in the multiphysics and multiscale branches and provides some examples of codes dealing with the specific tasks.

Table 2-I Classification of different codes.

Type of codes	Physics	Scale	Examples
System	Thermal-hydraulics	Macro	ATHLET [Lerchl1998], RELAP5 [RELAP2000], TRACE [TRACE2008], CATHARE [Barre1990]
Cell Lattice	Neutronics Deterministic	Meso	HELIOS [Pralong2005], CASMO [Rhodes2006], APOLLO2 [SanchezR2003], SCALE [Bowman2007], DRAGON [Marleau2008]
Monte Carlo	Neutronics Stochastic	Meso	MCNP [MCNP2006] , TRIPOLI [Brun2009], SERPENT [Leppanen2009], KENO [Hollenbach2005]
Reactor Dynamic	Neutronics	Meso	DYN3D* [Grundmann2005], NEM [Beam1999], NESTLE [Turinsky1994], PARCS* [Downar2006], QUABOX/CUBBOX [Langenbuch1984], CRONOS [Lautard1999], TORT-TD [Seubert2011], DeCART [Han2004], COBAYA [Jimenez2010]
Subchannel	Thermal-hydraulics	Meso	COBRA[Wheeler1976], MATRA [MATRA1998], FLICA4[Toumi2000],SUBCHANFLOW [GomezR2010]
CFD/DNS	Thermal-hydraulics	Macro Meso Micro	ANSYS-CFX [ANSYS2009], TRIO-U [Barthel2009],TURBIT [Grötzbach1977], NEPTUNE-CFD [Guelfi2005], OpenFOAM [OPENFOAM2010]
Fuel Mechanics	Fuel behaviour	Meso	DRACCAR [Papin2006], FRAPCON [Berna1997] TRANSURANUS [Lassmann1992], SCANAIR [Federici], FRAPTRAN [Cunningham]

* A simplified thermal-hydraulics and fuel rod mechanics model are included in such codes for taking into account the effect of thermal-hydraulic feedback, as it will be discussed in the next sections.

2.3 Coupled Codes Methodologies

The coupling between multiphysics and multiscale methodologies is a very complex subject with many possible combinations. It is out of the scope of this work to make an exhaustive analysis of each option. An extensive description of it is given in [CRISSUEV2], and some details and key points are found in [Ivanov2007], [Royer2008] and [Bousbia2007]. In this dissertation, a simplified but complete description of the requirements that must be considered while developing a coupled neutronics/thermal-hydraulics calculational approach is presented. Furthermore, the current status of worldwide developments is shortly discussed.

In the past, the thermal-hydraulic analyses of plant transients and of reactor core behaviour were performed separately, even though they may address the same reactor conditions.

The thermal-hydraulic analysis made use of simplified neutronic models and focused on the entire reactor system, including when needed the balance of plant. The result of such simulations provided the necessary boundary conditions for the core, such as mass flow and temperature distribution of the coolant at the core inlet together with the time-functions for pressure, which was then analyzed with detailed 3D neutronics models. However, in reality these boundary conditions are functions of the power generation in the reactor core. The application of these models is, therefore, limited by the consideration of proper core-thermal-hydraulic interface conditions and it may lead to very unrealistic accident conditions if all uncertainties are taken into account by demanding conservative boundary conditions [Langenbuch].

The coupled code calculation approach constitutes the normal evolution of these methods. This is especially true in cases where strong feedback between the core neutronics behaviour and the plant thermal-hydraulics is present, as well as in situations in which neutron flux distortions and excursions are important and its spatial distribution changes during the transient.

In the case of system codes coupled with 3D neutron kinetics models, six basic components of the coupling methodologies have been identified in order to be able to couple two codes [CRISSUEV2] and [Ivanov2007]. Thus, the way of coupling (internal or external); the coupling approaches (serial integration or parallel processing coupling), spatial mesh overlays (fixed or flexible), coupled time steps algorithms (synchronization of the time steps), coupling numerics (explicit, semi-implicit and implicit) and coupled convergence schemes must be considered and implemented.

New trends in the multiscale and multiphysics developments are characterized by a transition from fuel assembly based (FA-based) to more spatially refined solutions (pin by pin) e.g., by coupling 3D kinetic models with subchannel codes. In these developments, the six basic aspects mentioned before must be taken into account but with some peculiarities. For instance in a coupling scheme System code – 3D Neutronics, an integration algorithm usually considers the treatment of the neutronics code as a subroutine of the system code [Ivanov2007], [Bousbia2007] and [Langenbuch], whereas in a 3D Neutronics – Subchannel code coupling, the subchannel code is implemented as a subroutine in the 3D Neutronics solver [Cacuci2000] [Grundmann2005] and [Jimenez2010].

In the next subsections the main trends in couplings nuclear reactor behaviour simulations are briefly described.

2.3.1 Reactor dynamics code coupled with System code

As already discussed, the development of coupled codes in the past years has been fostered by safety analysis requirements. Although the most important part of a nuclear reactor is the core, several accidents are originated in the primary or secondary loops or even in some other components like the turbine. As an example, the Main Steam Line Break (MSLB) accident originates by the double-ended guillotine break of the steam line. Positive reactivity is inserted by the cooling of the primary water following the depressurisation of the steam generator (SG). The "plug" of cold water typically reaches the core a few seconds after the break occurs in the steam line. A regional core power increase may occur due to partial mixing in the reactor pressure vessel's downcomer of cold water from the affected SG with hot water from the intact SG. Thus, the non-uniformity of the core inlet temperature distribution that will cause a non-uniform reactivity increase in the core with associated non-uniform power responses justifies the coupled analysis [CRISSUEV1].

The first approximation in the analysis of such accidents was the use of point kinetics models or 1-D neutron kinetics models in the thermal-hydraulics system codes. In most codes the point kinetics can be related to several parallel coolant channels and corresponding fuel rod models describing parts of the reactor core. The increased use of 3-D neutron kinetics models and the made cost-effective by the increasing computer power available has replaced the point kinetics and 1-D neutron kinetic approaches for core transient analysis in which asymmetric core reactivity changes take place.

New coupled systems like ATHLET/DYN3D [Grundmann1998], TRAC-PF/NEM [IvanovK1999], RELAP5/PARCS [Barber1998], TRACE-PARCS [Xu2009], CATHARE-CRONOS2 [Mignot2004] among others constitute a new generation in reactor safety analyses.

The coupling can be achieved in three different ways: internal, external and combined.

2.3.1.1 Internal coupling

With internal coupling (Figure 2–1), the 3-D nodal neutron kinetics model is integrated into the core thermal-hydraulic model of the system code. Each neutronic node is coupled directly to a core thermal-hydraulic computational volume in the system code. The mesh sizes could be different, but in such a case an interpolation procedure for the parameter transfer would be necessary. Even though this method requires the exchange of a significant amount of information between the two codes (power and thermal-hydraulic feedback "TH–FB"), it also allows for detailed and direct system calculations. One major disadvantage of this method is that it involves significant modifications in both codes. The modifications, however, can be done in such a way that if new versions of the codes are released, or if it is desired the coupling with some other code, no changes or minimal changes of the new coupling routines are necessary to generate the coupled code.

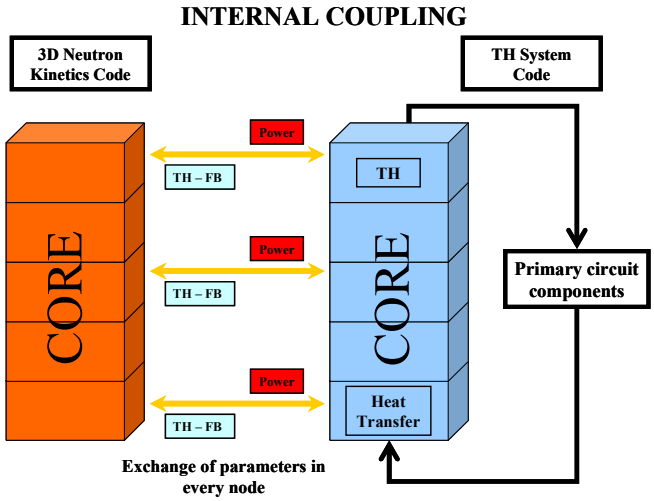


Figure 2–1 Internal coupling between a reactor dynamics code and a system code.

2.3.1.2 External coupling

In external coupling (Figure 2–2) the entire core is eliminated from the plant model in the system code. The core is completely modelled by the neutronics code. The thermal-hydraulics of the system is split in two parts: one part describing the core by means of simplified or sub-channel model already integrated in the neutronics code and the other part models the plant coolant system. As a consequence the coupled is performed via boundary conditions at the top and bottom of the core, i.e. pressure, core flow "G", enthalpies and boron concentration have to be transferred. This method facilitates the coupling procedure because it requires little or no modification of the thermal-hydraulics or neutron kinetics codes to be performed. However, there are certain problems with the external coupling method associated with the different thermal-hydraulic models for core and system modelling, which can lead to numerical instabilities and slow convergence. The coupling of the two thermal-hydraulic parts is usually done explicitly and very small time steps are necessary for stable calculations.

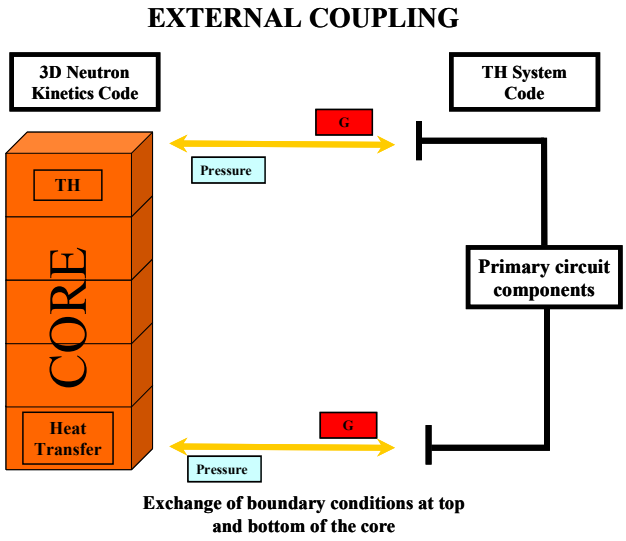


Figure 2–2 External coupling between a reactor dynamics code and a system code.

2.3.1.3 Combined coupling

This coupling scheme is a combination of the external and internal cases. The thermal-hydraulics of the core is calculated in parallel in the neutronics code as well as in the system code. The thermal-hydraulic core boundary conditions such as the pressure at the core outlet and the mass flow rates, the coolant temperatures and the boron concentration at the core inlet of the neutronics code are obtained from the system code. The power distribution calculated by the neutronics code is transferred to the system code. The influence of the core thermal-hydraulics on the nuclear power is calculated within the neutronics model. Two mapping schemes must be considered, the one of the thermal-hydraulic model integrated in the neutronic code and another in charge of the coupling between the channels of the neutronics and the system codes. Whereas the authors of [Grundmann2005] and [Kliem2010] refers to this coupling as "parallel coupling", here it is defined as "combined coupling" in order to avoid misunderstandings with the parallel processing coupling described at [Ivanov2007] in which the data exchange is done using a Parallel Virtual Machine (PVM) or Message Passing Interface (MPI) environments. The Figure 2–3 shows a diagram of this case.

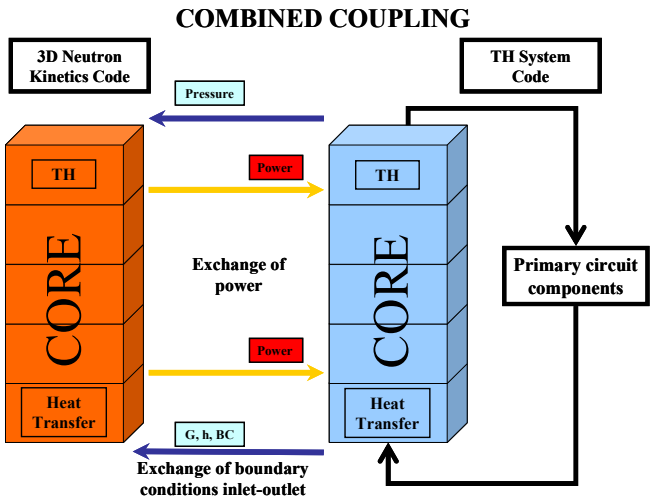


Figure 2–3 Combined coupling between a reactor dynamics code and a system code.

2.3.1.4 Multiscale refinements

Refinements in the neutronic solver have also been explored. Thus, the time dependent 3D discrete ordinates code TORT-TD has been internally coupled with the system code ATHLET [Seubert2007] and [Seubert2008] allowing 3D pin by pin analysis of transients in few energy groups. With this methodology some new phenomena can be analysed. TORT-TD is able to predict the pin power behaviour and ATHLET simulates all thermal fluid-dynamics and heat transport processes in the nuclear power plant. Although some small problems (2 x 2 mini-core and one assembly) are solved by coupling one thermal-hydraulic channel with one neutronic node (pin based), in the most general cases ATHLET uses a single thermal-hydraulic channel to simulate each fuel assembly (for instance while modelling a real core geometry) due to memory requirements.

2.3.2 Reactor dynamics code coupled with simplified thermal-hydraulics code

The external and combined coupling described in the past subsection requires that the reactor dynamics code contains or is internally coupled with a two phase flow thermal-hydraulics model capable of calculating assembly averaged feedback parameters. The reactor dynamics code uses these feedback parameters to actualize the cross sections for a later calculation of the reactor power. The "core" thermal-hydraulics model can be a 1-D two phase flow model assigning parallel channels to each assembly (or group of assemblies) like in DYN3D with its thermal-hydraulics model FLOCAL [Grundmann2005], or the thermal-hydraulic calculation could be done via the internal coupling with a thermal-hydraulics code considering cross flow, e.g. COBRA-TF/QUABOX-CUBBOX [Perin2010], ANDES/COBRA [Lozano2008] or CTF/NEM [Gouja2010]. Although the coupling presented in [Perin2010] is in every case performed at an assembly based scale, a brief discussion is also given in the direction of multi-scale methodologies. The access of local safety values and all the physical phenomena inside a fuel assembly can be better modelled using a sub-channel code (pin based) for the core representation. The current coupling of COBRA-TF with QUABOX-CUBBOX for instance, represents a preliminary but essential step in the code coupling strategy addressing the most detailed coupling methodologies.

2.3.3 Reactor dynamics code coupled with subchannel codes

As a second step in the direction of meso-scale, a new generation of high fidelity codes is under development. Several strategies have been searched in order to extend the methodologies for a more detailed and physical sound prediction of the pin/subchannel coupling.

2.3.3.1 Nodal diffusion solution with pin power reconstruction method

The pin power reconstruction method offers a very fast calculation of pin power distribution via a combination of the analytical solution of the two groups' diffusion equation plus the use of form functions taking into account internal heterogeneities (form functions must be produced in advance with cell lattice codes). Although the method can give a good approximation of the pin power distribution, the coupling used in such calculation is done at a fuel assembly scale. The analytical functions obtained with the method use as boundary conditions the values of currents and fluxes coming from the numerical solution of the diffusion equation at the nodal level. A real coupling at a pin level has not yet been attempted.

Developments were, however, done in some coupled systems like TRAC-PF1/NEM in order to include a pin power reconstruction scheme coupled to a subchannel model (based in COBRA-TF) aiming to improve transient fuel rod response [Ziabletsev2004] and [Solis2004]. Temporal adaptive algorithms were also implemented in such developments [Solis2002]. The refined analysis is performed in one hot channel that in some cases must be specified in advance like in DYN3D and in other cases can be dynamically identified like in TRAC-PF1/NEM/COBRA-TF. In any case the calculation is performed in one channel and some effects like the cross flow from neighbour subchannels cannot be considered.

2.3.3.2 Simplified transport solution (SP_3)

Many two group nodal diffusion solvers like PARCS, DYN3D, CRONOS2 have been extended to a multi-group diffusion and a time dependent SP3-transport solution or to the discrete ordinates method used for instance in TORT-TD. Such transport methods allow the predictions of the local power at pin level if appropriate pin based cross sections (XS) are provided.

Initially the one and half way coupling approach has been chosen. The neutronic solver gives a detailed pin power distribution to the thermal-hydraulic model. The thermal-hydraulics solver (usually a one dimensional model like FLOCAL in DYN3D or the use of COBRA-TF as channel code coupled with TORT-TD [Christienne2010]) predicts nodal averaged values (fuel assembly level) for the calculation of feedback parameters. The communication between the models in the direction neutron kinetics to thermal-hydraulics requires an average process and although the neutronic solution is more detailed, the thermal-hydraulic one does not take advantage of the pin power distribution coming from the neutronic solver. Furthermore, the actualization of cross sections is done with nodal averaged thermal-hydraulic parameters leading to an additional loss of information Figure 2–4.

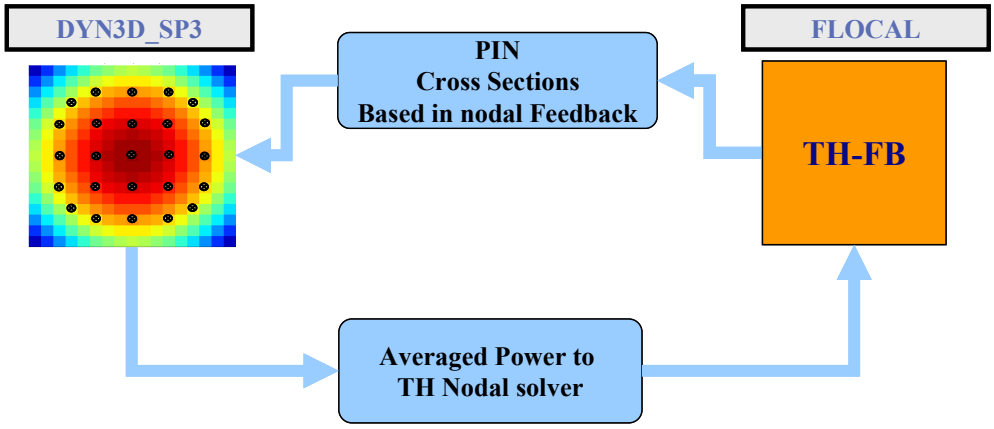


Figure 2–4 One and a half way coupling.

The development of advanced coupling methodologies for a more realistic description of the core behaviour under transients using a two way coupling is the main topic of this dissertation work and will be extensively discussed in Chapter 4.

2.3.4 Reactor dynamics code or system code coupled with CFD

Although the subchannel approach gives a better prediction of the fluid behaviour, it is also true that these codes use also volume averaged coefficients for the analysis of turbulent phenomena. CFD codes have more detailed turbulence models and do not need to include such averaged coefficients making them very attractive. The problem with the CFD codes is the CPU requirements that are currently very big and for solving large problems it is simply not practical. However, CFD codes have been used for validation of the models and coefficients used for instance, in the subchannel codes. Furthermore nowadays CFD codes have been in-

tegrated or coupled for modelling not the whole but certain crucial parts of the system that require a higher level of detail for a more precise evaluation of the effects of mixing and turbulence, as well as for improving the boundary conditions used by the other coupled techniques already described (prediction of distribution of temperatures at the inlet of the reactor vessel). Thus, new developments like the system code ATHLET with ANSYS-CFX [Papukchiev2009], CATHARE with TRIO-U [Vyskocil2011] or DYN3D with ANSYS-CFX [Kliem22010] have emerged.

2.3.5 Monte Carlo codes coupled with thermal-hydraulics codes

The development of coupling schemes between Monte Carlo and subchannel codes is driven by the need to improve the design tools reducing embedded conservatisms. Different coupled solutions have been developed [SanchezV2009]. Also CFD codes have been coupled with Monte Carlo codes for detailed simulations of small problems [Nuttin2004] and [Seker2007]. The coupling of such codes results in a very intensive CPU calculation, due to the nature of the Monte Carlo solutions. Hence developments aimed at exploring new solutions and to make use of high performance computing (HPC). Coupled systems like MCNP with COBRA-TF [SanchezV2009] are able to predict pin power for fuel assemblies at steady state. The far goal is to enhance the methodologies so that whole core pin-by-pin simulations become feasible in reasonable time and acceptable accuracy.

3 Extension of the Pin Power Reconstruction Capability of DYN3D

3.1 Introduction

As discussed in the previous chapter, the pin power reconstruction method offers a very fast calculation of pin power distribution in a fuel assembly via a combination of the analytical solutions of the two group's diffusion equation and the form functions taking into account internal heterogeneities in a fuel assembly. The pin power reconstruction method was originally developed as an extension of the nodal method that allows to determine local flux and power distributions in a consistent way and which removes the inherent limitation of loss of detailed information inside the nodes in the conventional nodal coarse-mesh solvers [Koebke1977]. Once the nodal solver based in the Nodal Expansion Method introduced in the past chapter is employed in the nodal core calculation, the two dimensional diffusion equation must be solved in the selected assembly (for each axial layer) in order to reconstruct the pin power distribution. The accuracy of the reconstructed pin power distribution comparing with the direct pin by pin transport solution relies on the reconstruction method used [Han2009]. Several pin power reconstruction methods have been developed in the past for few energy group problems. The methods differ mostly in how to represent and generate the intranodal flux. At the beginning (end of 70's) two-dimensional polynomials were used [Koebke1977], later on (second half of 80's) exponential functions were introduced, more recently (90's) analytical functions in two energy groups were implemented [Boer1992] and[Joon1989], and further developments and extensions can be found in recent years in the literature, e.g. the use of polynomials (for the fast and epithermal group) and hyperbolic functions (for the thermal group) in the few group BWR core simulator NEREUS [Iwamoto1999], the sub-mesh solution of the SIMULATE-4 code [Bahadir2006], or the group decoupled multi-group pin power reconstruction utilizing nodal solution 1D flux profiles [Lulin2010]. Moreover, a three dimensional pin power reconstruction method is described and presented in [Tohjoh2006].

In this Chapter, a description of the reactor dynamics code for thermal reactors DYN3D, in which all the developments of this dissertation lays on, is presented. For the sake of description of the pin power reconstruction method used in DYN3D, a first section with a short introduction to the nodal expansion method used to solve the two group diffusion equation is presented followed by the description of the pin power reconstruction method implemented in DYN3D. A following section reports the original contribution of this thesis to the extension of the method in the direction of having pin power reconstruction not only in one assembly but in several or even in the whole core. The concept of non-conform geometry will be described and introduced in order to illustrate the uses of the new developments. Additionally, the integration of this new version of the code in the SALOME platform as a part of the tasks of the Karlsruhe Institute of Technology (KIT) in the NURISP project is discussed at the end of this Chapter. Potential applications of such extension and integration are briefly discussed at the end of the section.

3.2 Description of DYN3D

DYN3D is a **DYN**amical **3-D**imensional code for thermal reactor cores. Written in FORTRAN90 and developed at Forschungszentrum Dresden-Rossendorf [Grundmann2005]. The 3-dimensional neutron kinetics model of the code is solved by means of nodal expansion methods applied to the two energy group neutron diffusion equation in hexagonal and rectangular geometry (recently a new version dealing with a multigroup formulation has been developed but without the pin power reconstruction formalism).

DYN3D has also a thermal-hydraulic model (FLOCAL) for single and two phase flow problems and includes a fuel rod model. The fuel assemblies are simulated by parallel coolant channels coupled hydraulically through the condition of equal pressure drop over all core channels.

To calculate the pin with the maximum power in a selected assembly, a two-dimensional flux reconstruction of the nodal flux can be made on the basis of the node homogenized cross sections. The method of successive smoothing is applied for the reconstruction of the neutron flux in selected assemblies as it will be described in the next sections. The neutron flux is reconstructed by an analytical solution of the two dimensional diffusion equations in each axial layer of the homogenized assembly and can then be multiplied by the form functions contained in the macroscopic cross-section library.

Additionally Assembly Discontinuity Factors (ADF) can be used for the correction of homogenization errors. Depletion calculations can be also performed to determine the starting point of a transient. Furthermore the steady state concentrations of neutron poisons, e.g. ^{10}B, can be calculated and the transient behaviour of ^{135}Xe and ^{147}Sm can be analyzed. The decay heat can be also taken into account based on the power history and the radioactive decay of fission products during a transient.

3.3 Theoretical Bases

The neutron kinetics model in DYN3D is based on the solution of the three dimensional two group neutron diffusion equation by nodal expansion methods. The macroscopic cross sections are considered constant in each spatial node. The method of transverse leakage approximation is applied to the 3-D diffusion equation leading to three one-dimensional equations coupled via the transversal leakage terms. In each energy group, the flux is then expanded in each direction by using two dimensional second order polynomials and exponential functions being the solutions of the homogeneous equation. An implicit finite differences scheme with exponential transformation is used for the time integration over the neutronic time step.

Next, a brief description of the solution method implemented in DYN3D will be given, preceded by a description of the form of the diffusion equation solved by the code.

3.3.1 Diffusion equation

The time dependent neutron diffusion equation in its more general differential form can be presented as follows:

$$\frac{1}{V(E)}\frac{\partial}{\partial t}\phi(\vec{r},E,t) = -\left[-\nabla\cdot D(\vec{r},E,t)\nabla\phi(\vec{r},E,t)+\Sigma_t(\vec{r},E,t)\phi(\vec{r},E,t)\right]$$
$$+\int_0^\infty \Sigma_s(\vec{r},E'\to E,t)\phi(\vec{r},E,t)dE' + \chi(E)\int_0^\infty \left(1-\beta(E')\right)\nu(\vec{r},E',t)\Sigma_f(\vec{r},E',t)\phi(\vec{r},E',t)dE' \tag{3.1}$$
$$+\sum_{i=1}^{I_p}\chi_i(E)\lambda_i C_i(\vec{r},t)$$

with the equations for the concentrations of precursors of delayed neutrons in the form:

$$\frac{\partial}{\partial t}C_i(\vec{r},t) = \int_0^\infty \beta_i(E')\nu(\vec{r},E',t)\Sigma_f(\vec{r},E',t)\phi(\vec{r},E',t)dE' - \lambda_i C_i(\vec{r},t) \tag{3.2}$$

$$i = 1,\ldots,I_P;\quad \forall(\vec{r},E,t)\in\Omega\times[0,\infty)\times[0,T]$$

and the standard notation:

$\vec{r}$	position vector,
E	energy,
t	time,
$V(E)$	velocity of the neutrons as a function of the energy,
$\phi(\vec{r},E,t)$	neutron flux,
$D(\vec{r},E,t)$	diffusion coefficient,
$\Sigma_t(\vec{r},E,t)$	total macroscopic cross section,
$\Sigma_s(\vec{r},E'\to E,t)$	scattering cross section from E' to E in $\vec{r}$ and t,
$\nu(\vec{r},E,t)$	mean number of neutrons obtained from fission,
$\Sigma_f(\vec{r},E,t)$	fission macroscopic cross section,
$\chi(E)$	normalized fission spectrum for prompt neutrons,
$C_i(\vec{r},t)$	concentration of precursor of delayed neutrons of group i,
$\chi_i(E)$	normalized fission spectrum for delayed neutrons,
λ_i	effective decay constant of the group i of precursor of delayed neutrons,
β_i	effective fraction of delayed neutrons from group i,
$\beta(E)$	total fraction of delayed neutrons at energy E
I_p	total number of precursor's groups,
Ω	volume of the reactor and
T	total time of interest.

The total macroscopic transport cross section $\Sigma_t(\vec{r},E,t)$ includes absorption $\Sigma_a(\vec{r},E,t)$ and scattering $\Sigma_s(\vec{r},E,t)$. The absorption is itself the sum of two terms, the fission cross section $\Sigma_f(\vec{r},E,t)$ and the capture cross section $\Sigma_c(\vec{r},E,t)$.

The diffusion equation results from making a balance in a volume element in an energy interval $d\vec{r}dE$ for neutrons with energies in the interval $[E-dE,E+dE]$ located in a volume $[\vec{r}-d\vec{r},\vec{r}+d\vec{r}]$ and time t. The left-hand side term in (3.1) represents the change in time of the neutron density, which is given by the production mechanisms minus the loss mechanisms per unit time in $d\vec{r}dE$. The first two terms in the right-hand side of (3.1) (in brackets) represent the losses (net neutron flux leaving the volume Ω, and the absorptions and scatterings ocurring in the volume $[\vec{r}-d\vec{r},\vec{r}+d\vec{r}]$ for neutrons of energy $[E-dE,E+dE]$ respectively). The last three terms compute the productions of neutrons in differential volume with energies about E: the first represents all the neutrons with energy E' that after being scattered get the energy E. The second one corresponds to the prompt neutrons that are born directly from the fission with energy E. Finally the third one represents the neutrons with energy E produced by the decay of the delayed neutrons precursors.

The set of equations (3.2) represent the time variation of the concentrations of delayed neutron precursors. The first term in the right-hand side corresponds to the production of nuclei of the precursor group i appearing after fission. The second term represents the losses due to the decay of the precursors in group i.

One of the first approximations used in neutron diffusion theory is the discretization of the continuous neutron energy interval using the multi-group theory [Glasstone1970]. The energy interval is divided into several groups in such a way that the cross sections in each group can be replaced by energy-averaged constant values. Thus, defining the neutron flux of the group g as:

$$\phi_g(\vec{r},t) \equiv \int_{E_g}^{E_{g-1}} \phi(r,E,t)dE \tag{3.3}$$

and the macroscopic cross sections in every group as:

$$\Sigma_g(\vec{r},t) \equiv \frac{1}{\phi_g}\int_{E_g}^{E_{g-1}} \Sigma(E)\phi(\vec{r},E,t)\,dE \tag{3.4}$$

A new formulation of the diffusion equation is then possible:

$$\begin{aligned}\frac{1}{V_g}\frac{\partial}{\partial t}\phi_g(\vec{r},t) &= \nabla\cdot D_g(\vec{r},t)\nabla\phi_g(\vec{r},t) - \Sigma_{R,g}(\vec{r},t)\phi_g(\vec{r},t) + \sum_{\substack{g'=1\\ g'\neq g}}^{G}\Sigma_{s,g'\to g}(\vec{r},t)\phi_{g'}(\vec{r},t) + \\ &\frac{\chi_g}{k_{eff}}\sum_{g'=1}^{G}(1-\beta_g)\nu_{g'}(\vec{r},t)\Sigma_{f,g'}(\vec{r},t)\phi_{g'}(\vec{r},t) + \sum_{i=1}^{I_p}\chi_{g,i}\lambda_i C_i(\vec{r},t) \quad g=1,\ldots,G\end{aligned} \tag{3.5}$$

and

$$\frac{\partial}{\partial t}C_i(\vec{r},t)=\frac{1}{k_{eff}}\sum_{g=1}^{G}\beta_{g,i}\nu_g(\vec{r},t)\Sigma_{f,g}(\vec{r},t)\phi_g(\vec{r},t)-\lambda_i C_i(\vec{r},t)\quad i=1,\ldots,I_p \tag{3.6}$$

where the removal macroscopic cross section $\Sigma_{R,g}(\vec{r},t)$ is defined as:

$$\Sigma_{R,g}(\vec{r},t)=\Sigma_{t,g}(\vec{r},t)-\Sigma_{s,g\to g}(\vec{r},t) \tag{3.7}$$

and the effective multiplication factor k_{eff} has being introduced.

By using the relationship between neutron current and neutron flux given by the Fick's law

$$J_g(\vec{r},t)=-D_g(\vec{r},t)\nabla\phi_g(\vec{r},t) \tag{3.8}$$

and considering only two energy groups, the formalism of the time dependent neutron diffusion equation in two-groups is:

$$\frac{1}{V_1}\frac{\partial}{\partial t}\phi_1(\vec{r},t)=-\nabla J_1(\vec{r},t)-\Sigma_{R,1}(\vec{r},t)\phi_1(\vec{r},t)+\frac{1}{k_{eff}}\sum_{g'=1}^{G}\left(1-\beta_g\right)\nu\Sigma_{f,g'}(\vec{r},t)\phi_{g'}(\vec{r},t)+\sum_{i=1}^{I_p}\lambda_i C_i(\vec{r},t)$$

$$\frac{1}{V_2}\frac{\partial}{\partial t}\phi_2(\vec{r},t)=-\nabla J_2(\vec{r},t)-\Sigma_{a,2}(\vec{r},t)\phi_2(\vec{r},t)+\Sigma_{s,1\to 2}(\vec{r},t)\phi_1(\vec{r},t) \tag{3.9}$$

and

$$\frac{d}{dt}C_i(\vec{r},t)=\frac{1}{k_{eff}}\sum_{g=1}^{2}\beta_{g,i}\nu\Sigma_{f,g}(\vec{r},t)\phi_g(\vec{r},t)-\lambda_i C_i(\vec{r},t);\quad i=1,\ldots,I_p \tag{3.10}$$

with

$$\beta_g=\sum_{i=1}^{I_p}\beta_{g,i};\ g=1,2 \tag{3.11}$$

Some assumptions have been made. First of all, the up-scattering term $\Sigma_{s,2\to 1}(\vec{r},t)$ has been neglected, as it is usually done in two-group theory [Glasstone1958]. Thus, the removal cross section of group 2 results to be equal to the absorption $\Sigma_{a,2}(\vec{r},t)$, taken into account that $\Sigma_{t,2}(\vec{r},t)=\Sigma_{s,2\to 2}(\vec{r},t)+\Sigma_{s,2\to 1}(\vec{r},t)+\Sigma_{a,2}(\vec{r},t)$. Finally it was considered that all the neutrons coming from the fission are produced with fast energies, i.e., $\chi_1=1$ and $\chi_2=0$.

3.3.2 The nodal expansion method used in DYN3D

For the sake of simplicity, the method will be introduced for the static case. A brief discussion about the time dependent solution will be given at the end of this subsection.

Assuming that the time derivatives in equations (3.9) and (3.10) vanish, the system to be solved is given by:

$$\nabla J_1(\vec{r})+\Sigma_{R,1}(\vec{r})\phi_1(\vec{r})=\frac{1}{k_{eff}}\sum_{g'=1}^{G}\left(1-\beta_g\right)\nu\Sigma_{f,g'}(\vec{r})\phi_{g'}(\vec{r})+\sum_{i=1}^{I_p}\lambda_i C_i(\vec{r})$$

$$\nabla J_2(\vec{r})+\Sigma_{a,2}(\vec{r})\phi_2(\vec{r})=\Sigma_{s,1\to 2}(\vec{r})\phi_1(\vec{r}) \tag{3.12}$$

and

$$\lambda_i C_i(\vec{r},t)=\frac{1}{k_{eff}}\sum_{g=1}^{2}\beta_{g,i}\nu\Sigma_{f,g}(\vec{r},t)\phi_g(\vec{r},t);\quad i=1,\ldots,I_p \tag{3.13}$$

Substituting (3.13) in (3.12), the final system of differential equations in steady state arises:

$$\nabla J_1(\vec{r})+\Sigma_{R,1}(\vec{r})\phi_1(\vec{r})=\frac{1}{k_{eff}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r})\phi_{g'}(\vec{r})$$

$$\nabla J_2(\vec{r})+\Sigma_{a,2}(\vec{r})\phi_2(\vec{r})=\Sigma_{s,1\to 2}(\vec{r})\phi_1(\vec{r}) \tag{3.14}$$

In most of the nodal methods applied for Cartesian geometry, three one-dimensional equations are derived by transverse integration over the other two perpendicular directions, i.e., one equation in the variable x is the result of integrating (3.14) over y and z direction. The one-dimensional equations will be coupled via the transversal leakage. Considering that the nodes in which the system must be solved are cubes with volume:

$$V^n=a_x^n\cdot a_y^n\cdot a_z^n \tag{3.15}$$

a transversal integration of equation (3.14) for example for the coordinate x will lead to:

$$-\frac{D_1}{a_y a_z}\frac{d^2}{dx^2}\int_{-\frac{a_y}{2}}^{\frac{a_y}{2}}\int_{-\frac{a_z}{2}}^{\frac{a_z}{2}}\phi_1(\vec{r})\,dydz+\frac{\Sigma_{R,1}}{a_y a_z}\int_{-\frac{a_y}{2}}^{\frac{a_y}{2}}\int_{-\frac{a_z}{2}}^{\frac{a_z}{2}}\phi_1(\vec{r})\,dydz=$$

$$\frac{1}{k_{eff}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}\frac{1}{a_y a_z}\int_{-\frac{a_y}{2}}^{\frac{a_y}{2}}\int_{-\frac{a_z}{2}}^{\frac{a_z}{2}}\phi_{g'}(\vec{r})\,dydz-L_1(x) \tag{3.16}$$

$$-\frac{D_2}{a_y a_z}\frac{d^2}{dx^2}\int_{-\frac{a_y}{2}}^{\frac{a_y}{2}}\int_{-\frac{a_z}{2}}^{\frac{a_z}{2}}\phi_2(\vec{r})\,dydz+\frac{\Sigma_{a,2}}{a_y a_z}\int_{-\frac{a_y}{2}}^{\frac{a_y}{2}}\int_{-\frac{a_z}{2}}^{\frac{a_z}{2}}\phi_2(\vec{r})\,dydz=\frac{\Sigma_{s,1\to 2}}{a_y a_z}\int_{-\frac{a_y}{2}}^{\frac{a_y}{2}}\int_{-\frac{a_z}{2}}^{\frac{a_z}{2}}\phi_1(\vec{r})\,dydz-L_2(x)$$

where the transverse integrated leakage term $L_g(x)$ is given by

$$L_g(x) = -\frac{D_g}{a_y a_z} \int_{-\frac{a_y}{2}}^{\frac{a_y}{2}} \int_{-\frac{a_z}{2}}^{\frac{a_z}{2}} \left(\frac{d^2}{dy^2} + \frac{d^2}{dz^2} \right) \phi_g(\vec{r})\, dydz \tag{3.17}$$

Defining:

$$\hat{\phi}_g = \frac{1}{a_y a_z} \int_{-\frac{a_y}{2}}^{\frac{a_y}{2}} \int_{-\frac{a_z}{2}}^{\frac{a_z}{2}} \phi_g(\vec{r})\, dydz \tag{3.18}$$

$$\hat{F}_g(x) = \begin{cases} \frac{1}{k_{eff}} \sum_{g'=1}^{G} \nu\Sigma_{f,g'} \frac{1}{a_y a_z} \int_{-\frac{a_y}{2}}^{\frac{a_y}{2}} \int_{-\frac{a_z}{2}}^{\frac{a_z}{2}} \phi_{g'}(\vec{r})\, dydz & \text{for g} = 1 \\ \frac{\Sigma_{s,1\to 2}}{a_y a_z} \int_{-\frac{a_y}{2}}^{\frac{a_y}{2}} \int_{-\frac{a_z}{2}}^{\frac{a_z}{2}} \phi_1(\vec{r})\, dydz & \text{for g} = 2 \end{cases} \tag{3.19}$$

and

$$\Sigma_g = \begin{cases} \Sigma_{R,1} & \text{for g} = 1 \\ \Sigma_{a,2} & \text{for g} = 2 \end{cases} \tag{3.20}$$

The system (3.16) can be written like

$$-D_g \frac{d^2}{dx^2} \hat{\phi}_g(x) + \Sigma_g \hat{\phi}_g(x) = \hat{F}_g(x) - \hat{L}_g(x) \tag{3.21}$$

The one-dimensional transverse-integrated flux is now expanded in second order polynomials and exponential functions, which are the solutions of the homogeneous differential equation (3.21) in the form:

$$\hat{\phi}_g(x) = \sum_{i=0}^{2} c_{g,i} h_i\left(\frac{x}{a_x}\right) + a_{g,1} e^{B_g x} + a_{g,2} e^{-B_g x} \tag{3.22}$$

with the Legendre polynomials

$$\begin{aligned} h_0(u) &= 1 \\ h_1(u) &= 2\sqrt{3}u \\ h_2(u) &= 6\sqrt{5}\left(u^2 - \frac{1}{12}\right) \end{aligned} \tag{3.23}$$

satisfying the orthogonal condition

$$\int_{-\frac{a_u}{2}}^{\frac{a_u}{2}} h_i\left(\frac{a_u}{2}\right) h_j\left(\frac{a_u}{2}\right) du = a_u \delta_{ij} \tag{3.24}$$

and

$$B_g = \sqrt{\frac{\Sigma_g}{D_g}} \tag{3.25}$$

Assuming that the source term $\hat{F}_g(x)$ and the transversal leakage $L_g(x)$ have a more smooth behaviour than the neutron flux, they are expanded only in quadratic polynomials:

$$\hat{F}_g(x) = \sum_{i=0}^{2} f_{g,i} h_i\left(\frac{x}{a_x}\right) \tag{3.26}$$

$$L_g(x) = \sum_{i=0}^{2} l_{g,i} h_i\left(\frac{x}{a_x}\right) \tag{3.27}$$

Introducing the expansions (3.22), (3.26) and (3.27) in (3.21) and performing the derivatives, expressions for the coefficients can be found:

$$c_{g,0} = \frac{f_{g,0} + l_{g,0} + 12\sqrt{5}\frac{D_g}{a_x^2} c_{g,2}}{\Sigma_g}$$

$$c_{g,i} = \frac{f_{g,i} + l_{g,i}}{\Sigma_g} \quad i = 1,2 \tag{3.28}$$

For an inner iteration, which solves the differential equations, the coefficients of the source (3.26) and leakage term (3.27) as well as the partial currents of the last external iteration are given.

With initial values of the node-averaged flux, the coefficients c_g are found for a later calculation of the outgoing partial currents and actualization of node-averaged fluxes. By means of these outgoing partial currents and node-averaged fluxes, the incoming partial currents in the interfaces and boundaries are obtained for the next inner iteration. At the end of the inner iteration process the coefficients c_g are actualized. These coefficients will also be used for updating the node-averaged source for the next external iteration, which will check for convergence of k_{eff}, using the values obtained by the last inner iteration.

$$f_{g,i} = \frac{1}{k_{eff}} \nu\Sigma_{f,g} c_{g,i} \quad i = 0,1,2 .$$

At the end of an external iteration k_{eff} is calculated by means of:

$$k_{eff}^{ITN} = \frac{\sum_{n=1}^{N} \left(\sum_{g=1}^{2} \nu\Sigma_{f,g}^{n} \bar{\phi}_{g}^{n,(ITN)} \right)^2}{\sum_{n=1}^{N} \left(\sum_{g=1}^{2} \nu\Sigma_{f,g}^{n} \bar{\phi}_{g}^{n,(ITN)} \right) \cdot \left(\sum_{g=1}^{2} \nu\Sigma_{f,g}^{n} \bar{\phi}_{g}^{n,(ITN-1)} \right)} \tag{3.29}$$

In which N is the total number of nodes and ITN is the external iteration number.

For the transient case, an implicit finite difference scheme together with an exponential transformation technique is used. The flux is represented by:

$$\phi_g(\vec{r},t) = e^{\Omega(t'-t+\Delta t)} \phi'_g(\vec{r},t') \tag{3.30}$$

Using an implicit difference scheme the time derivative of the neutron flux at time t is replaced by:

$$\frac{d}{dt}\phi_g(\vec{r},t) \approx \frac{1}{\Delta t}\left((1+\Omega\Delta t)\phi_g(\vec{r},t) - e^{\Omega\Delta t}\phi_g(\vec{r},t-\Delta t) \right) \tag{3.31}$$

The exponent Ω can be estimated in every node, taking into account the change of the flux in the time interval by:

$$\Omega = \frac{1}{\Delta t} \ln \left(\frac{\sum_{g=1}^{G} \bar{\phi}_g(t)}{\sum_{g=1}^{G} \bar{\phi}_g(t-\Delta t)} \right) \tag{3.32}$$

The use of these approximations in the time derivative of the neutron flux will introduce terms that can be grouped in order to follow the procedure described above but in the transient case. A detailed description of the procedure is given in [Grundmann2005] and [Beckert2008].

3.4 Pin power reconstruction method used in DYN3D

To calculate the pin with the maximum power in an assembly, a two dimensional flux reconstruction of the nodal flux is performed in DYN3D based in the node homogenized cross sections. The time dependent system of two-group diffusion equation considering an exponential behaviour of the neutron fluxes during the time step and after some algebraic manipulation is given by:

$$-D_1\nabla^2\phi_1(x,y,t)+\Sigma_1\phi_1(x,y,t)=\sum_{g=1}^{2}\nu\Sigma_{fg}^{*}\phi_g(x,y,t)$$
$$-D_2\nabla^2\phi_2(x,y,t)+\Sigma_2\phi_2(x,y,t)=\Sigma_{s1}\phi_1(x,y,t) \tag{3.33}$$

with

$$\Sigma_1=\Sigma_{R,1}(\vec{r})+\frac{\Omega}{V_1}+D_1B_{z1}^2$$
$$\Sigma_2=\Sigma_{a,2}(\vec{r})+\frac{\Omega}{V_2}+D_2B_{z2}^2 \tag{3.34}$$

Where the distribution of delayed neutrons has been assumed proportional to the prompt neutrons and thereby [Grundmann2005]:

$$\nu\Sigma_{fg}^{*}=(1-\beta_g)\frac{\nu\Sigma_{fg}}{k_{eff}}+\sum_{i=1}^{I_p}\frac{\lambda_i\beta_{g,i}\nu\Sigma_{fg}}{\sum_{g'=1}^{2}\beta_{g',i}\nu\Sigma_{fg}\bar{\phi}_{g'}}C_j \tag{3.35}$$

and the transversal buckling B_{zg}^2 defined as:

$$B_{zg}^2=\frac{1}{D_g\bar{\phi}_g a_z}\left(J_{g,+z}^{+}-J_{g,+z}^{-}+J_{g,-z}^{+}-J_{g,-z}^{-}\right) \tag{3.36}$$

In matrix form, the system can be seen as:

$$\nabla^2\phi+A\phi=0 \tag{3.37}$$

where

$$A=\begin{pmatrix}\dfrac{\nu\Sigma_{f1}^{*}-\Sigma_1}{D_1} & \dfrac{\nu\Sigma_{f2}^{*}}{D_1}\\ \dfrac{\Sigma_{s1}}{D_2} & -\dfrac{\Sigma_2}{D_2}\end{pmatrix} \tag{3.38}$$

and

$$\phi=\begin{pmatrix}\phi_1(x,y,t)\\ \phi_2(x,y,t)\end{pmatrix} \tag{3.39}$$

A solution of the system can be obtained by the method of separation of variables. However it is first necessary to diagonalize the matrix *A*. Based in the fact that if a square matrix is diagonalizable, then there exists a non-singular matrix *P* such that:

$$B_m^2 = P^{-1} A P \tag{3.40}$$

where the columns of *P* are linearly independent eigenvectors of *A* and the diagonal elements of the diagonal matrix B_m^2 are the eigenvalues of *A* associated with these eigenvectors.

The eigenvalues of *A* can be found by solving the characteristic polynomial of *A* given by:

$$f_A = \det\left(A - IB^2\right) = 0 \tag{3.41}$$

where B^2 are the eigenvalues of *A*, and the characteristic polynomial results in:

$$f_A = \left(a_{11} - B^2\right)\left(a_{22} - B^2\right) - a_{12}a_{21} = \left(B^2\right)^2 - B^2\left(a_{11} + a_{22}\right) - \left(a_{12}a_{21} - a_{11}a_{22}\right) = 0 \tag{3.42}$$

The solutions are given by the next relation:

$$B^2 = -\frac{\mathrm{X}_1^2}{2}\left(1 \pm \sqrt{\Delta}\right) \tag{3.43}$$

where

$$\Delta = 1 + 4\frac{\mathrm{X}_2^2}{\left(\mathrm{X}_1^2\right)^2} \tag{3.44}$$

$$\mathrm{X}_1^2 = \frac{\Sigma_1 - \nu\Sigma_{f1}^*}{D_1} + \frac{\Sigma_2}{D_2} \tag{3.45}$$

$$\mathrm{X}_2^2 = \frac{\Sigma_{s1}\nu\Sigma_{f2}^* + \Sigma_2\left(\nu\Sigma_{f1}^* - \Sigma_1\right)}{D_1 D_2} \tag{3.46}$$

The system has two real solutions; the fundamental one that can be either positive or negative:

$$B_f^2 = -\frac{\mathrm{X}_1^2}{2}\left(1 - \sqrt{\Delta}\right) \tag{3.47}$$

and the transient one that it is always negative:

$$B_t^2 = -\frac{X_1^2}{2}\left(1+\sqrt{\Delta}\right) \tag{3.48}$$

the eigenvectors can now be calculated solving the system:

$$\left(A - IB^2\right)\kappa = 0 \tag{3.49}$$

for every eigenvalue B_f^2 and B_t^2. The following system of equations must be solved for each of them.

$$\begin{pmatrix} \frac{\nu\Sigma_{f1}^* - \Sigma_1}{D_1} - B^2 & \frac{\nu\Sigma_{f2}^*}{D_1} \\ \frac{\Sigma_{s1}}{D_2} & -\frac{\Sigma_2}{D_2} - B^2 \end{pmatrix} \begin{pmatrix} \kappa_1 \\ \kappa_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.50}$$

Choosing $\kappa_2 = 1$

$$\kappa_1 = \frac{\left(D_2 B^2 + \Sigma_2\right)}{\Sigma_{s1}} \tag{3.51}$$

Thus, the non singular matrix *P* used for the diagonalization of *A* will be given by:

$$P = \begin{pmatrix} \frac{\left(D_2 B_f^2 + \Sigma_2\right)}{\Sigma_{s1}} & \frac{\left(D_2 B_t^2 + \Sigma_2\right)}{\Sigma_{s1}} \\ 1 & 1 \end{pmatrix} \tag{3.52}$$

and the inverse by:

$$P^{-1} = \frac{\Sigma_{s1}}{D_2\left(B_f^2 - B_t^2\right)} \begin{pmatrix} 1 & -\frac{\left(D_2 B_t^2 + \Sigma_2\right)}{\Sigma_{s1}} \\ -1 & \frac{\left(D_2 B_f^2 + \Sigma_2\right)}{\Sigma_{s1}} \end{pmatrix} \tag{3.53}$$

Thus (3.37) can be rewritten as:

$$\nabla^2\phi + A\phi = \nabla^2\left(P^{-1}\phi\right) + \left(P^{-1}AP\right)\left(P^{-1}\phi\right) = \nabla^2\Psi + B_m^2\Psi = 0 \tag{3.54}$$

Where the modal fluxes have been defined as:

$$\Psi = \left(P^{-1}\phi\right) \tag{3.55}$$

Relation (3.55) establishes the transformation from normal fluxes to modal ones. In the subsequent, the method will be described for the modal fluxes with the understanding that the normal fluxes coming from the nodal solution are equivalent to the modal fluxes and thus the knowledge of normal fluxes implies the knowledge of the modal ones.

The problem has been reduced to solve the homogeneous Helmholtz equation by means of the separation of variables method. The general solution of such equation is an infinite series of cosines and sinuses or hyperbolic cosines and sinuses, depending on the signs of the eigenvalues. Thereby, on one hand, for the fundamental solution there exist two possibilities:

$$\begin{aligned}\Psi_f(x,y) &= \sum_{i=1}^{\infty}\left(a_{c,i}^f \cos\left(B_{f,i}\sigma_i\right) + a_{s,i}^f \sin\left(B_{f,i}\sigma_i\right)\right) \quad \rightarrow \quad B_f = \sqrt{B_f^2} \quad B_f^2 \geq 0 \\ \Psi_f(x,y) &= \sum_{i=1}^{\infty}\left(a_{c,i}^f \cosh\left(B_{f,i}\sigma_i\right) + a_{s,i}^f \sinh\left(B_{f,i}\sigma_i\right)\right) \quad \rightarrow \quad B_f = \sqrt{-B_f^2} \quad B_f^2 < 0\end{aligned} \tag{3.56}$$

On the other hand, for the transient solution, which is always negative, there exists just one possibility:

$$\Psi_t(x,y) = \sum_{i=1}^{\infty}\left(a_{c,i}^t \cosh\left(B_{t,i}\sigma_i\right) + a_{s,i}^t \sinh\left(B_{t,i}\sigma_i\right)\right) \tag{3.57}$$

The modal flux is then a function of the eigenvalues B_f^2 and B_t^2 and the arbitrary angle σ_i.

$$\sigma_i = x\cos\alpha_i + y\sin\alpha_i \tag{3.58}$$

In order to have a particular solution of the problem, the unknown coefficients $a_{c,i}^f$, $a_{s,i}^f$, $a_{c,i}^t$ and $a_{s,i}^t$ have to be determined from the boundary conditions (Dirichlet problem). Of course, the finite number of possible boundary conditions of the problem forces the truncation of the series for a specific number of arbitrary angles σ_i based on the number of boundary conditions that can be specified for the problem.

In rectangular geometry, the four surface averaged fluxes together with the 4 corner point fluxes constitute the minimum set of boundary conditions for the analytical flux reconstruction method [Boer1992]. The four surface averaged fluxes (or currents) come directly from the nodal solution. For the corner fluxes, the Method of Successive Smoothing (MSS) derived in [Fischer1981] is used.

The MSS can be illustrated by means of Figure 3–1.

It considers that the neutron modal flux value at a node vertex (grey box) can be approximated by the average (smoothing) of the vertex neutron modal fluxes of each adjacent node (yellow boxes) i.e.

$$\bar{\Psi}_C = \frac{1}{4}\sum_{i=1}^{4}\tilde{\Psi}^i \tag{3.59}$$

The vertex neutron modal flux $\tilde{\Psi}^i$ can be extrapolated as:

$$\tilde{\Psi}^i = A + Bx + Cy \tag{3.60}$$

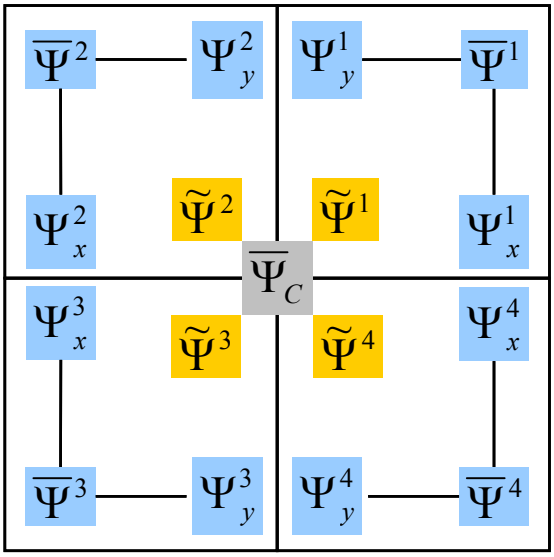


Figure 3–1 Four nodes with a common vertex.

The unknown parameters A, B and C can be found in terms of the known parameters (blue boxes). Thereby, for the vertex neutron of the first node $\tilde{\Psi}^1$ the next system of equations can be considered,

$$\begin{aligned}
\overline{\Psi}^1 &= \int_0^1\int_0^1 \tilde{\Psi}^i dxdy = \int_0^1\int_0^1 (A+Bx+Cy)\,dxdy = A+\frac{B}{2}+\frac{C}{2} \\
\Psi_x^1 &= \int_0^1 \tilde{\Psi}^i(x=0)\,dy = \int_0^1 (A+Cy)\,dy = A+\frac{C}{2} \\
\Psi_y^1 &= \int_0^1 \tilde{\Psi}^i(y=0)\,dx = \int_0^1 (A+Bx)\,dx = A+\frac{B}{2}
\end{aligned} \tag{3.61}$$

where normalization to the unitary square has been assumed for the sake of simplicity.

Solving the system (three equations with three unknowns), a general expression for the vertex neutron modal fluxes is found:

$$\tilde{\Psi}^1 = \overline{\Psi}^1(2x+2y-1) + \Psi_x^1(1-2x) + \Psi_y^1(1-2y) \tag{3.62}$$

Evaluating in (x,y) = (0,0)

$$\tilde{\Psi}^1 = \Psi_x^1 + \Psi_y^1 - \overline{\Psi}^1 \tag{3.63}$$

Similar expressions can be found for the other nodes.

Thus four surface averaged modal fluxes together with the four corner point modal fluxes in every energy group give a total number of 16 boundary conditions for the problem. The arbitrary angle given by (3.58) can have then four different values for the parameter α_i normally chosen to obtain uniformly distributed directions within the node namely (0°, 45°, 90° and 135°).

Defining:

$$CS_i = \begin{cases} \cos(B_i\sigma_i) \rightarrow B_i = \sqrt{B_i^2} & B_i^2 \geq 0 \\ \cosh(B_i\sigma_i) \rightarrow B_i = \sqrt{-B_i^2} & B_i^2 < 0 \end{cases}$$
$$SN_i = \begin{cases} \sin(B_i\sigma_i) \rightarrow B_i = \sqrt{B_i^2} & B_i^2 \geq 0 \\ \sinh(B_i\sigma_i) \rightarrow B_i = \sqrt{-B_i^2} & B_i^2 < 0 \end{cases} \tag{3.64}$$

The modal fluxes can be rewritten as:

$$\Psi_f(x,y) = \sum_{i=1}^{4}\left(a_{c,i}^f CS_i + a_{s,i}^f SN_i\right)$$
$$\Psi_t(x,y) = \sum_{i=1}^{4}\left(a_{c,i}^t CS_i + a_{s,i}^t SN_i\right) \tag{3.65}$$

A system of eight equations with eight unknowns can then be established for each one of the modal fluxes (fundamental and transient) by means of the evaluation of the modal flux in the four corners and the integration of it over each of the four surfaces. The system of equations for $u = f,t$ reads as follows:

$$\begin{aligned}
&\Psi_u(0,0) = \bar{\Psi}_1 \\
&\Psi_u(1,0) = \bar{\Psi}_2 \\
&\Psi_u(1,1) = \bar{\Psi}_3 \\
&\Psi_u(0,1) = \bar{\Psi}_4 \\
&\int_0^1 \Psi_u(x, y=0)dx = \Psi_{12} \\
&\int_0^1 \Psi_u(x=1, y)dy = \Psi_{23} \\
&\int_0^1 \Psi_u(x, y=1)dx = \Psi_{34} \\
&\int_0^1 \Psi_u(x=0, y)\,dy = \Psi_{41}
\end{aligned} \tag{3.66}$$

where the right hand values are the known neutron modal fluxes at the four vertexes calculated with the MSS (first four equations) and the surface modal fluxes (last four equations) coming from the nodal solution as illustrated in Figure 3–2.

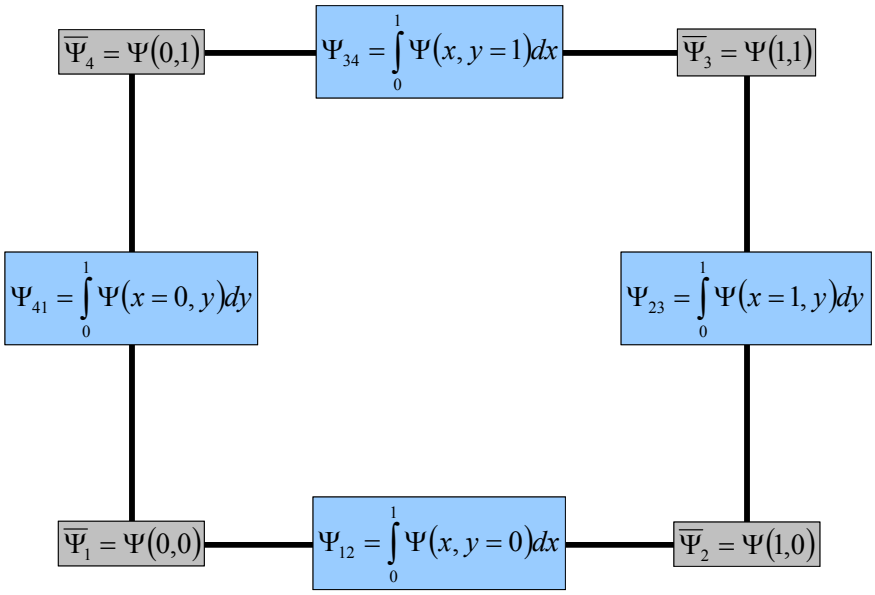


Figure 3–2 Boundary condition in the unitary square.

The solution of the system (3.66) will determine the coefficients $a_{c,i}^{f}$, $a_{s,i}^{f}$, $a_{c,i}^{t}$ and $a_{s,i}^{t}$.

By means of (3.52) and (3.55), the neutron fluxes can be calculated from the modal fluxes.

The method will give a continuous neutron flux function inside the node, and can be evaluated at every point of interest. The power inside the assembly can be estimated as usual:

$$P_{n,m}(x_n, y_m) = \sum_{g=1}^{2} \varepsilon_g \Sigma_{f,g} \phi_g(x_n, y_n) \tag{3.67}$$

where ε is the energy released by fission in the node considered.

It is important to notice that the semi-analytical neutron flux calculated inside the assembly is unable to take into account the internal heterogeneities, i.e., the influence of control rods, water holes, rods with gadolinium, among others would be not correctly represented. A correction of the method can be done by means of the introduction of form functions. These form functions are usually calculated with a lattice code and they can be defined as a global correction factor as:

$$ff_{n,m}(x_n, y_m) = \frac{\sum_{g=1}^{2} \Sigma_{f,g}^{nm} \varphi_g^{nm}(x_n, y_n)}{\sum_{g=1}^{2} \bar{\Sigma}_{f,g} \bar{\varphi}_g} \tag{3.68}$$

or as a group dependent correction factor as:

$$ff_{n,m}^{g}(x_n, y_m) = \frac{\Sigma_{f,g}^{nm} \varphi_g^{nm}(x_n, y_n)}{\bar{\Sigma}_{f,g} \bar{\varphi}_g} \quad g = 1,2 \tag{3.69}$$

where

$\Sigma_{f,g}^{nm}$ is the local fission cross section in the pin located at (x_n, y_m) for group g,
φ_g^{nm} is the local neutron flux in the pin located at (x_n, y_m) for group g,
$\overline{\Sigma}_{f,g}$ is the assembly averaged fission cross section for group g,
$\overline{\varphi}_g$ is the assembly averaged neutron flux for group g.

The introduction of the form functions in the calculation will lead to a new formulation of the intra-fuel assembly pin power distribution given in the global form by:

$$P_{n,m}(x_n, y_m) = ff_{n,m}\left(\sum_{g=1}^{2} \varepsilon_g \Sigma_{f,g} \phi_g(x_n, y_n)\right) \tag{3.70}$$

and in the group dependent form as:

$$P_{n,m}(x_n, y_m) = \sum_{g=1}^{2} ff_{n,m}^{g} \varepsilon_g \Sigma_{f,g} \phi_g(x_n, y_n) \tag{3.71}$$

In the case of DYN3D, the pin power reconstruction method has been implemented in the global i.e., pin power distribution reconstructed by means of (3.70).

3.5 Extensions to the pin power reconstruction method of DYN3D

The pin power reconstruction method in DYN3D was initially conceived for a hot channel analysis. Besides of the coolant channels connected to fuel elements of the reactor core, hot channels can be considered for analyzing the effect of local power peaks, coolant temperature, flow rate or fuel rod parameters variations. These hot channels are connected each to a certain core channel. A parallel hot channel is calculated with the same pressure drop and the same axial power distribution as for the related core channel, however, the total channel power, coolant inlet temperature, flow resistant coefficients at the channel inlet and fuel rod geometry and properties can be varied. The power for the hot channel can be estimated, then, from an intra-nodal pin power reconstruction calculation taking the power peak as a power value for the hot channel. Using these hot channels, DNB analysis for the hottest fuel rods can be performed on-line with the transient calculation [Grundmann2005]. The disadvantage of this method is the knowledge in advance of the hottest fuel assembly position, which must be indicated in the input deck for a DYN3D simulation with the pin power reconstruction option. This procedure implies a double execution of DYN3D one for finding the hottest fuel assembly and another one for the hot channel analysis using the pin power reconstruction method.

Although nowadays with the available computational power this procedure can be done without too much time cost, it is also true that additional problems may arise if the position of the hot fuel assembly changes during the transient evolution as may be the case in practical applications.

Moreover, the implementation of an automatic approach for the localization of the hottest fuel assembly within the core will only work if the pin power reconstruction is available in the whole core and not only in one fuel assembly. Hence a flexible fast running pin power reconstruction was developed for DYN3D that can be applied not just for one assembly but for several or, even, for an entire core calculation (except the ones of the periphery as will be later discussed). This option offers also the possibility to have DYN3D ready for the new developments related to local mesh refinements, thus paving the way for further optimization methods, such as the "non-conform geometry". The main idea of the "non-conform geometry" method is the refinement of the mesh and solver capabilities in the critical regions of the domain under study. Thus, the solution of a whole system can be done in a nodal base (assembly based) in regions away from local perturbations, and a local refinement can be utilized for the region with the largest perturbations.

The effect of an equivalent non-conform geometry in DYN3D can be useful for instance in the analysis of asymmetric core perturbation, e.g. a single control rod has to be moved, a boron dilution slug passing through one quarter of the core (in one of the water pumps), or due to changes in the operation of one of the pumps. Thus, for example, in Figure 3–3 a mesh and "solver" refinement has been used in the southwest quadrant of a PWR test core.

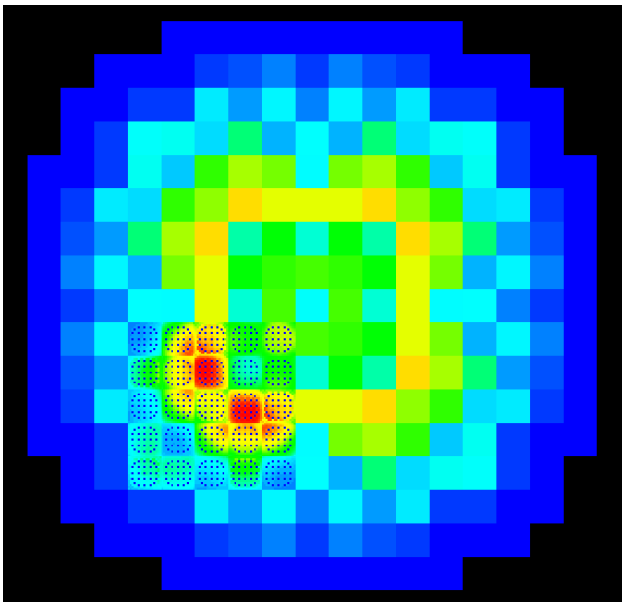


Figure 3–3 Southwest quadrant with non-conform geometry and pin power reconstruction.

Nowadays similar implementations are being developed in the frame of international projects like NURESIM [Cacuci2006], in which the subchannel code FLICA4 [Toumi2000] has been foreseen for having such kind of implementations.

The coupling of DYN3D and FLICA at nodal level was one of the tasks of the EU project which has been accomplished in the first part of the project [Gommlich2010]. The second part is related to the coupling in a non-conform base and is the main support for the extension done and described as a part of this dissertation. More details about the NURESIM platform will be given in a subsequent subsection.

3.5.1 Modifications to DYN3D source

The extension of the pin power reconstruction method required several modifications in the subroutines of DYN3D. The idea was also to implement such changes with less impact in the input file structure of DYN3D.

3.5.1.1 Extension of the semi-analytical pin power reconstruction

The modifications to the logic of the pin power reconstruction method in DYN3D can be grouped in two cases:

1. INITIALIZATION: Read input options, allocation of arrays and searching of assemblies for Pin Power Reconstruction (PPR).

2. CALCULATION: creation of arrays containing the cell lattice data for the fuel assemblies specified (assembly and pin pitch, number of pins per assembly, etc.), calculation of pin power reconstruction in selected assemblies either in steady state or in transient.

A simplified view of the changes related with the INITIALIZATION group can be observed in Figure 3–4. The pin power reconstruction calculation is an optional feature of DYN3D that will be done when the keyword 'FLUX RECONSTRUCTION' is present in the DYN3D input file [Grundmann2010].

Once the keyword is found in the input file, the reading of the necessary data for PPR calculation starts. Three cases are possible: (i) the input argument NREC can be zero, (ii) minus one or (iii) can have a value between one and the total number of assemblies for the problem under study (NASS).

The first case (NREC = 0) does not perform any PPR calculation, but an allocation of some arrays in the ***ndallocrec.f*** subroutine, that are needed later, is done.

The second case (NREC = -1) means that the flux reconstruction is carried out for all fuel assemblies with exception of the ones located at the core boundary. Such restriction can be explained by the fact that the PPR method requires the average of the neutron modal fluxes at the corners of each adjacent node for the calculation of the actual neutron modal flux value at these locations (see equation (3.59) and Figure 3–1). Thereby, an assembly located at the core boundary will not have all the necessary contributions for such calculation. For the automatic localization of the fuel assemblies for which PPR is possible, two new subroutines were developed. One of them deals with hexagonal geometry (***ndrectoth.f***) whereas the other one (***ndrectotr.f***) with rectangular geometry.

The last case ($1 \leq NREC \leq NASS$) is related with the new DYN3D capabilities in the direction of "non-conform geometry". A PPR calculation will be carried out for the assemblies selected in the input. Using this option, NREC additional lines have to be read from input file containing the coordinates of the assemblies chosen.

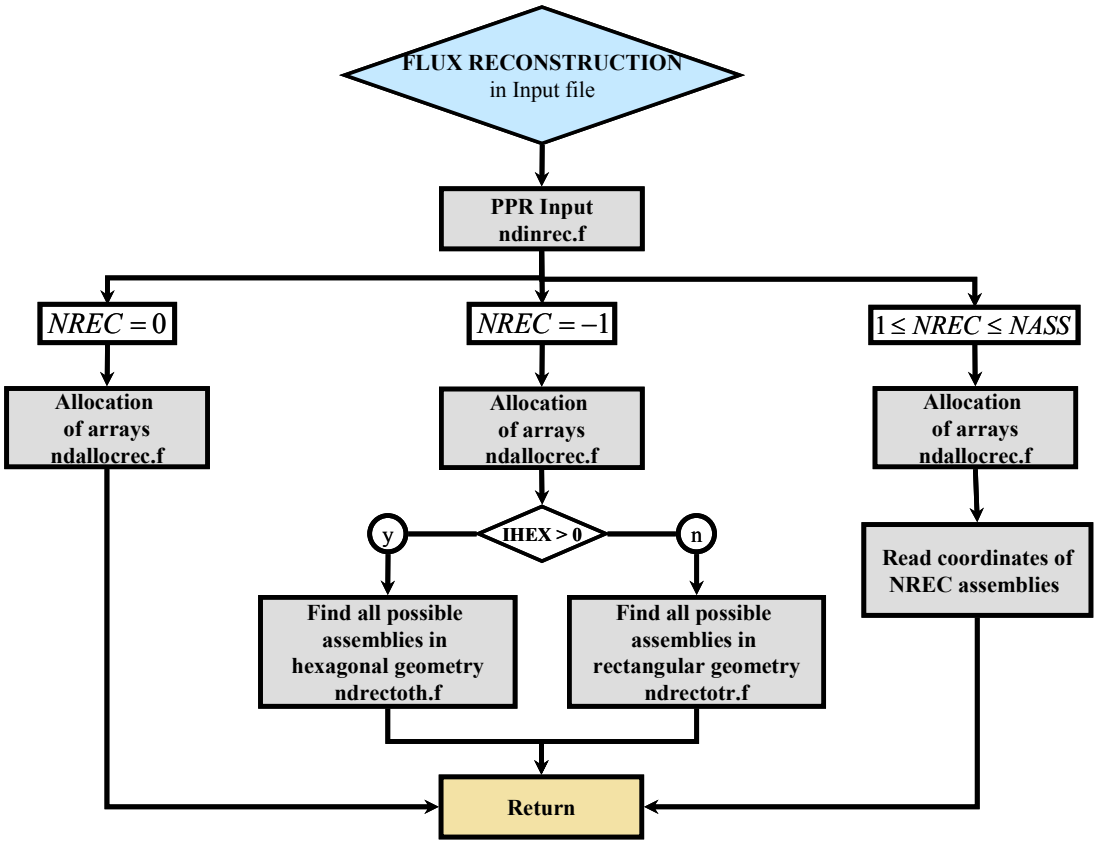


Figure 3–4 Simplified view of the flow diagram for the initialization of PPR in DYN3D.

In the Table 3-I, a summary of the subroutines involved in this first step, the status of changes (new or modified) and a brief description of the purpose and changes in the subroutine is given.

Table 3-I Subroutines used for INITIALIZATION of PPR.

Subroutine	Status	Description of changes
ndinrec.f	Modified	Input of new PPR calculation options
ndallocrec.f	Modified	Allocation of needed arrays e.g. storage of coordinates of assemblies to be reconstructed and for the calculated pin power distribution.
ndrectoth.f	New	For NREC = -1 performs a sweep in all the assemblies finding the ones in which the PPR can be done for hexagonal geometry.
ndrectotr.f	New	For NREC = -1 performs a sweep in all the assemblies finding the ones in which the PPR can be done for rectangular geometry.

The CALCULATION group implies two steps: a calculation of cell lattice data arrays and the PPR calculation.

In the previous version of DYN3D, as it was already mentioned, the PPR calculation was limited to just one assembly. The inner data of a fuel assembly (pin and assembly pitch) were taken directly from the input data for the selected assembly. For the new extension the problem must be solved in a different way because in theory it is always possible to have different fuel assemblies in a core load (especially in the case of BWR's). For this reason a new subroutine ***ndreclattice.f*** was developed and is in charge of building arrays containing the geometrical cell lattice data of each assembly type. The execution of this module is done after the reading of all input data of DYN3D. A previous allocation of the arrays used in this subroutine must be done. Due to this effect, modifications to the main subroutine for allocation of memory for the kinetic arrays (***ndallock.f***) and to the module containing such arrays (***cnd_salome.f***) were also done.

The pin power reconstruction calculation has two branches. In the left one of the flow diagram shown in Figure 3–5, the subroutine ***ndreconh.f*** is the calculation engine responsible for carrying out the PPR calculation in hexagonal geometry. On the other hand for rectangular geometry the subroutine ***ndreconr.f*** is the one dealing with the calculation. Both subroutines call different auxiliary subroutines in order to do the calculation. The PPR-solvers for each engine are respectively ***ndpinrch.f*** and ***ndflrecr.f***. In the case of hexagonal geometry, due to the more complex system of equations, two additional subroutines are used: ***ndgelgt.f*** dealing with the solution of the linear equation system and ***ndflux_h.f*** which calculates the fast and thermal neutron fluxes in the selected point of the hexagon of interest.

The two engines (***ndreconh.f*** and ***ndreconr.f***) work for both steady state and transient with the only difference that in transient the fission source must include the contribution of delayed neutrons which is assumed by the hypothesis to be proportional to the prompt fission source (see equation (3.35)).

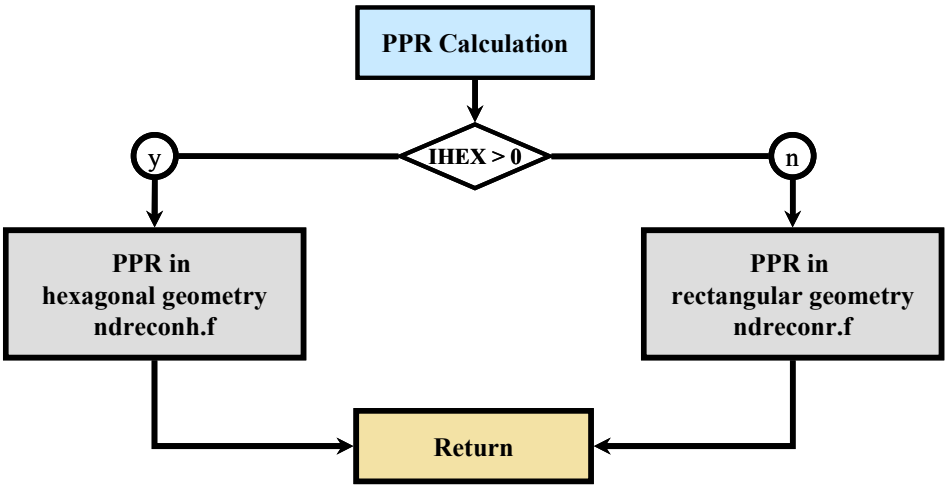


Figure 3–5 Simplified view of the flow diagram for the PPR calculation in DYN3D.

A brief description of all changes in the subroutines involved in the extension of the PPR method as well as the status of changes is presented in Table 3-II.

Table 3-II Subroutines used for CALCULATION of PPR.

Subroutine	Status	Description of changes
cnd_salome.f	Modified	Definition of arrays for geometrical cell lattice data.
ndallock.f	Modified	Allocation of arrays for holding the geometrical cell lattice data of each assembly type.
ndreclattice.f	New	Building of arrays containing the geometrical cell lattice data of each assembly type (assembly and pin pitch and number of pins per assembly)
ndreconh.f	Modified	Loop for several hexagonal fuel assemblies. Changes in the normalization for including the axial profile effect in each assembly.
ndreconr.f	Modified	Loop for several rectangular fuel assemblies. Changes in the normalization for including the axial profile effect in each assembly.
ndpinrch.f	Modified	Use of geometrical cell lattice data from new arrays. Double precision in all variables and intrinsic Fortran functions (due to numerical instabilities). Move of normalization to the main engine ***ndreconh.f***.
ndflrecr.f	Modified	Use of geometrical cell lattice data from new arrays. Double precision in all variables and intrinsic Fortran functions (due to numerical instabilities). Move of normalization to the main engine ***ndreconr.f***.
ndflux_h.f	Modified	Double precision in all variables and intrinsic Fortran functions (due to numerical instabilities).

3.5.1.2 Changes related with the use of form functions.

The use of form functions, discussed at the end of the previous section, for taking into account the inner heterogeneities of the fuel assemblies was also implemented. Although for the hexagonal geometry an implementation already existed, it was not fully usable and had to be almost totally rewritten.

Additional modifications were implemented not only in the subroutines related with the PPR method but also in the ones related with the reading and allocation of cross sections. As discussed above, the form functions must be calculated in advance by means of a lattice code and they must be part of the cross sections library. As a result, several changes were also done in the subroutines ***ndincrs22.f*** and ***ndall22_xs.f*** being the numbers 22 and 27 identification numbers for the cross sections libraries used in DYN3D based on the NEMTAB format [Ivanov1999].

The cross section library can include the form functions or not, even it can be a combination of materials with and without form functions. The form functions must be provided as a function of depletion at the end of each burn-up branch.

A summary of the modified subroutines is presented in Table 3-III.

Table 3-III Subroutines used for the inclusion of form functions.

Subroutine	Status	Description of changes
ndall22_xs.f	Modified	Allocation of arrays containing the form function values.
Ndincrs22.f	Modified	Read of form functions from the cross section library as a function of burn-up.
ndreconh.f	Modified	Use of form functions in hexagonal geometry.
ndreconr.f	Modified	Use of form functions in rectangular geometry.

The total extension of the pin power reconstruction method in DYN3D implied substantial modifications in 11 subroutines and the creation of three additional subroutines.

Before testing and verifying this extension, it is convenient to introduce another section dealing with the integration of DYN3D in the NURESIM platform [Cacuci2006] in order to get a better view of the potential of the developed PPR capabilities.

3.6 Integration of DYN3D into the SALOME platform

3.6.1 NURESIM platform

Within the FP7 Collaborative Project NURISP (**NU**clear **R**eactor **I**ntegrated **S**imulation **P**roject) [Chauliac2008] new and significant steps have been done towards a European Reference Simulation Platform for applications relevant to present PWR and BWR and to future reactors. The first step towards this target has been made during the FP6 NURESIM Integrated Project [Cacuci2006], where the already common and well-proven NURESIM computational platform has been developed. The common software platform used for the integration of codes is provided by SALOME [SALOME], which is an open-source set of tools for pre and post processing, and for integration and/or linkage of solvers.

One of the tasks of the NURISP project, in particular of the Multiphysics Subproject (SP3) is the full implementation of DYN3D into the NURESIM platform and the coupling with the thermal-hydraulic code FLICA-4 [Toumi2000] which is also part of the platform. The coupling will allow the advanced analysis of core transients.

Before the particular integration of DYN3D is described, a detailed explanation about the philosophy and general integration process in SALOME will be given.

3.6.2 SALOME philosophy

For the integration of a code in the SALOME platform it is necessary first of all to know the philosophy of the open source platform. SALOME is a generic development platform for pre/post-processing and code coupling for numerical simulation. The main objective of SALOME is to provide a generic user interface that is user friendly and efficient, and that contributes to reducing the research costs and calculation times. It facilitates the interoperation between CAD modelling and computing codes and the implementation of coupling between computing codes in a heterogeneous distributed environment.

SALOME is composed of several modules devoted to specific tasks, so it can be possible, in the broadest use of the platform, to define a geometry with the module ***GEOM*** then to define a mesh in the geometry with the module ***SMESH*** and to assign properties to the nodes resulting from the mesh definition (cross sections for instance). These steps related with the pre-processing will prepare the input data for the solvers integrated in the platform. The ***VISU*** module allows not only post-processing visualization but also on-line visualization. It is also possible to have total view and control of the flow chart under execution when the solvers are performing their tasks using the module ***SUPERV*** (or the equivalent ***YACS*** module introduced for the newest versions of SALOME). SALOME focuses on the external coupling of codes with a particular attention given to the exchange of structured and unstructured meshes and fields lying on those meshes.

The general platform architecture is shown in Figure 3–6 [Crouzet2009]. The solvers do not belong to the platform itself, but the platform provides automatic tools to generate interfaces encapsulating Python, C/C++ and FORTRAN (via a C interface) components in SALOME components [Bergeaud]. In Figure 3–7 an example of SALOME functionality is illustrated.

One key point related with the coupling of codes inside SALOME is the communication among them. If a common exchange format exists, all the codes will be capable of importing and exporting data in a common way. In the SALOME platform the MED/DEM (Data Exchange Model) has been chosen [Bouhamou2005]. Figure 3–8 shows an example of communication between two different codes (DYN3D and FLICA [GomezA2009]) via MED/DEM and an Application Programming Interface API. Details about the implementation will be given later.

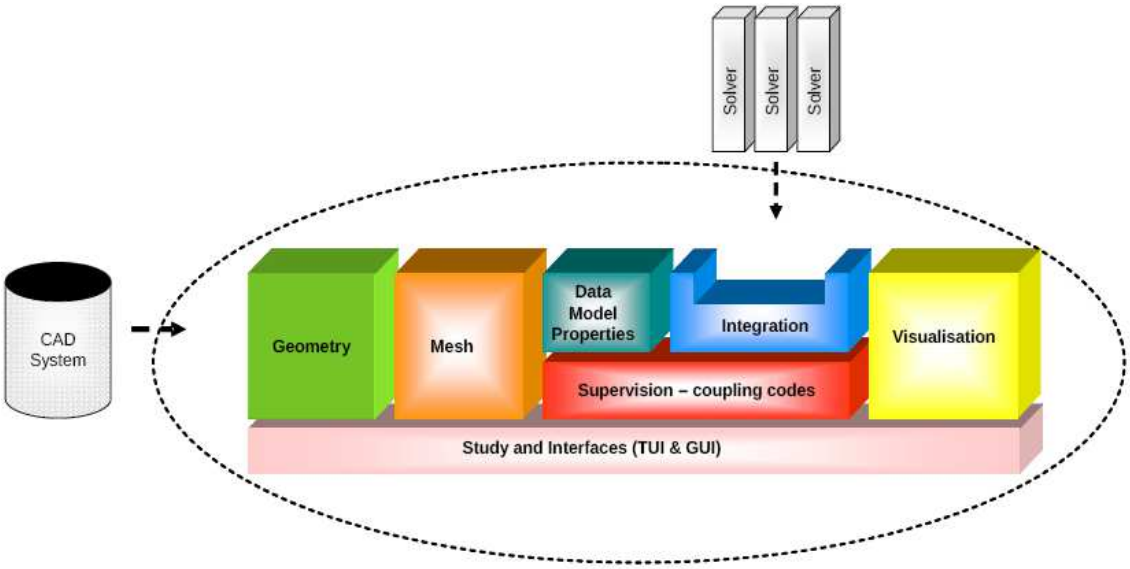


Figure 3–6 SALOME platform architecture.

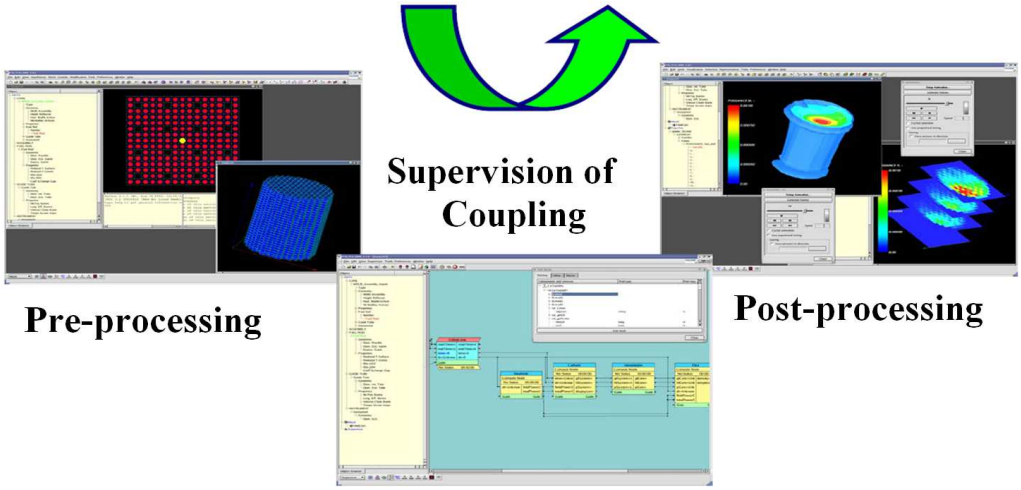


Figure 3–7 Overview of the SALOME platform.

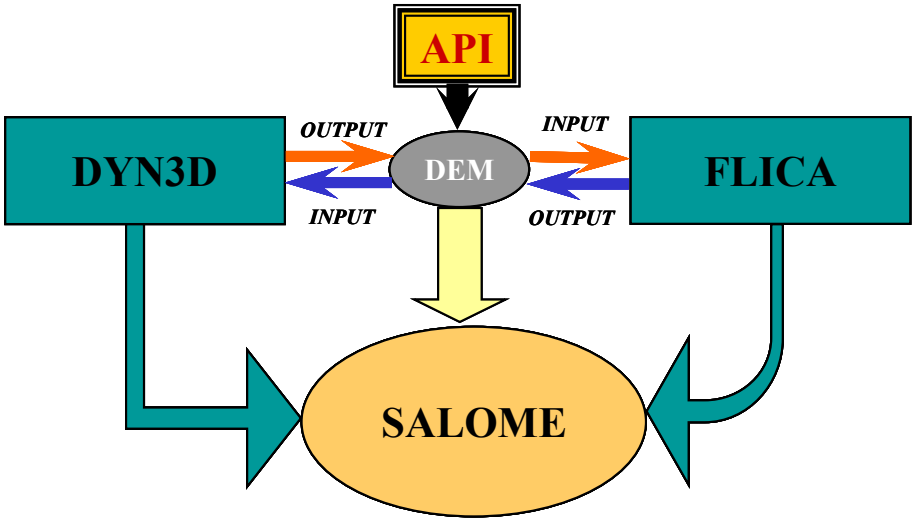


Figure 3–8 Two codes coupled via MED/DEM Data Exchange Model.

The SALOME platform is based on the concept of components. It is necessary to format a code as a component for integrating it in the platform. A component can have several "Services", that are operations or functions that the component can perform, for instance, to read input data, to compute output data or modify the component internal state among others. A component in SALOME follows the line illustrated in Figure 3–9 [Salome2005]. In the lowest level the original user code source is found. It can be written as usual in FORTRAN or in C/C++ and be packaged in the form of a static library or even as an external binary file "Black box". In the middle layer, the "C++ component" is located. In this point, the Application Programming Interface is defined in the form of a C++ class who will call and manipulate the functions, executables or libraries already defined in the lowest level. The final step is the integration of the class as a SALOME or Python component ready for being used in SALOME. This can be done automatically with the use of the SALOME tool ***hxx2salome***. This tool performs the wrapping to SALOME, making an automatic code generation, compilation

and update of the environmental variables, and will be discussed in detail in the next subsections. The final step is depicted in Figure 3–10.

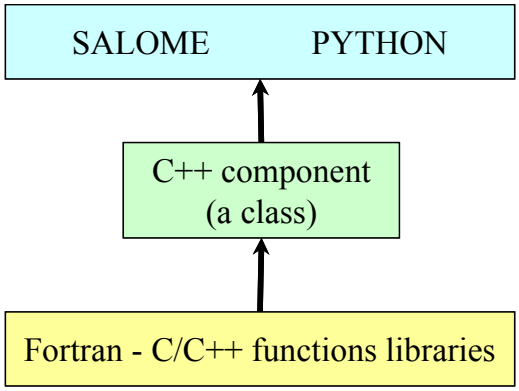


Figure 3–9 Code integration architecture.

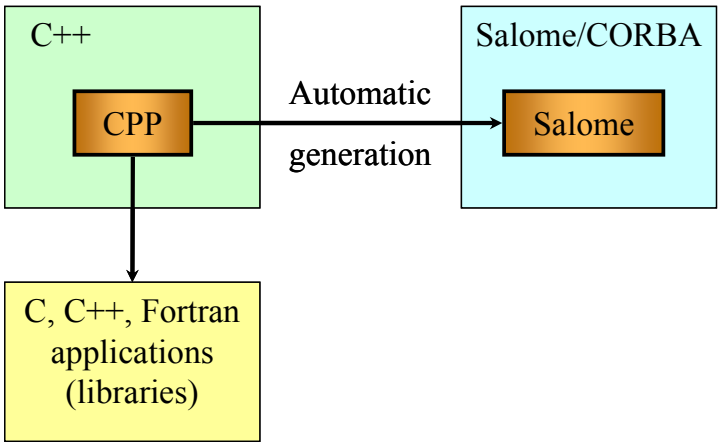


Figure 3–10 Automatic wrapping to SALOME.

3.6.3 Integration inside SALOME

The integration of a component inside SALOME follows some rules related to its structure and a formal procedure for building the application. SALOME platform is rather complex software, and its build procedure requires special attention. In the next subsections it will be roughly described the integration procedure [Salome2007].

3.6.3.1 General steps towards the integration of codes inside SALOME

The Build Procedure is based in the GNU ***autoconf*** tool [MacKenzie2008] and [GNUAutotools]. GNU ***autoconf*** tool helps writing Makefiles. The General build procedure with ***autoconf*** consists of the following steps:

1. Build configure: generate a configuration script ***configure*** from a ***configure.ac*** or ***configure.in*** pattern.
2. Configure: generate ***Makefile*** from each ***Makefile.in*** with ***configure*** script.

3. Make: compile the sources as specified in ***Makefiles***.
4. Install: place built binaries, resources and other files into a certain structure at a place specified on Configure step.

SALOME has several online and offline manuals and documentation [SALOME], which present the basis for the implementation of a Salome Component. Additionally SALOME has some scripts in order to make easier the integration.

Once SALOME platform is installed, a subdirectory HXX2SALOME is created. Several tools have been installed here, among them the ***hxx2salome*** script and some related tools like ***SA_new_cpp_component*** and ***SA_build*** that are useful for the integration.

The first thing to do is to implement the C++ engine that will perform the services. To do it the ***SA_new_cpp_component*** is used. This script creates a complete tree that will allow building a C++ component with a Python interface.

A Python interface can be obtained via SWIG. SWIG (Simplified Wrapper and Interface Generator) is a software development tool that simplifies the task of interfacing different languages to C and C++ programs. In summary, SWIG is a compiler that takes C declarations and creates the wrappers needed to access those declarations from other languages like Python [Beazley1997].

The ***SA_build*** script can be used for compilation and installation of the C++ component or engine. After running this script, it will be possible to run all the services or functions of the component outside the SALOME platform via normal binary file or via Python.

The most important tool for integration into the platform is the ***hxx2salome*** tool [Crouzet2]. The script ***hxx2salome*** is a prototype tool for automatic SALOME component generation. It makes the wrapping of a C++ standalone component, into a Salome component automatically. This tool takes as input the interface of a C++ component (an .hxx file), parses the public API, and uses the type information to generate the SALOME component (the Interface Definition Language, IDL and its implementation).

A brief explanation of each of the scripts with some examples related with the integration of DYN3D will be presented in order to clarify the procedure.

3.6.3.2 The ***SA_new_cpp_component*** script (C++ component implementation)

The ***SA_new_cpp_component*** takes as input the name of the component to be created. The result of running such command can be seen in Figure 3–11, where the automatically created *DYN3D* tree in the directory *DYN3DCPP_SRC* is shown.

All the files inside the tree directory must exist; however some of them are irrelevant for the integration process and can remain empty or unchanged (for instance: *AUTHORS* and *NEWS*). Some other files are scripts for finding and cleaning files and must not be modified (***root_clean***, ***rfind, archive***), they are used later on by the ***build_configure*** script.

```
[gomez@localhost salome]$ source env_products.sh
[gomez@localhost salome]$ SA_new_cpp_component DYN3D
[gomez@localhost salome]$ ls -la DYN3DCPP_SRC/
total 56
drwxr-xr-x   4 gomez gomez   4096  2010-06-07 12:23 ./
drwxrwxr--  89 gomez nurisp  4096  2010-06-07 12:23 ../
drwxr-xr-x   3 gomez gomez   4096  2005-12-12 11:29 adm/
-rwxr-xr-x   1 gomez gomez    552  2005-12-12 11:29 archive*
-rw-r--r--   1 gomez gomez     47  2005-12-12 11:29 AUTHORS
-rwxr-xr-x   1 gomez gomez   1268  2007-12-13 18:08 build_configure*
-rwxr-xr-x   1 gomez gomez   1262  2007-12-13 18:07 build_configure~*
-rw-r--r--   1 gomez gomez      0  2005-12-12 11:29 ChangeLog
-rw-r--r--   1 gomez gomez    332  2007-12-14 09:51 configure.in.base
-rw-r--r--   1 gomez gomez    331  2007-12-14 09:47 configure.in.base~
-rw-r--r--   1 gomez gomez     75  2005-12-12 11:29 Makefile.am
-rw-r--r--   1 gomez gomez      0  2005-12-12 11:29 NEWS
-rw-r--r--   1 gomez gomez    348  2005-12-12 11:29 README
-rwxr-xr-x   1 gomez gomez    878  2005-12-12 11:29 rfind*
-rwxr-xr-x   1 gomez gomez    703  2005-12-12 11:29 root_clean*
drwxr-xr-x   3 gomez gomez   4096  2010-06-07 12:23 src/
```

Figure 3–11 *DYN3D* component (*DYN3DCPP_SRC*).

The ***build_configure*** script will take the ***configure.in.base*** script and will create the ***configure.in*** script for a subsequent run of ***autoconf***. Additionally it will take ***Makefile.am*** and via ***automake*** [MacKenzie22008] will create the ***Makefile.in*** file in every directory in which it finds a ***Makefile.am***.

The script ***configure.in.base*** defines the prerequisites (required products) for compilation of the component. It is automatically created; however some additional lines must be added in case of particular prerequisites for the component. The script defines the following main things:

- Path to ***m4*** [Seindal2010] macros with products definition.
- General files which could be included in every ***Makefile***.
- Source and build root directories.
- Check procedure for common tools, such as libtool, C/C++ compiler and others.
- Some additional actions to be done on configure step: create directories, copy files, etc.

In the case in which the component is based on a FORTRAN source code (like in the case of DYN3D), some lines related with the FORTRAN compiler desired for compilation of the source must be added to the ***configure.in.base*** script.

The script ***Makefile.am*** must be included in all the subdirectories in which a compilation, a library, documentation or an executable creation is needed. Each ***Makefile.am*** is basically a series of make variable definitions with rules. The ***Makefile.am*** in this level must usually contain just the name of the directory *src* where the sources of the component are found.

The folder *adm* includes a subdirectory called *unix* in which it is possible to define check procedures via m4 files with flags, global variables, and check procedure for the new product. It is also possible to check if needed libraries or programs are installed and can work. Examples of this options can be found in [Salome2007] and in [Seindal2010].

Finally the *src* subdirectory is the place in which all the sources are contained. The content of this subdirectory is listed in Figure 3–12.

```
[gomez@localhost DYN3DCPP_SRC]$ cd src
[gomez@localhost src]$ ls -la
total 16
drwxr-xr-x  5 gomez gomez  4096 2010-06-07 12:23 ./
drwxr-xr-x  3 gomez gomez  4096 2010-06-07 12:23 ../
drwxr-xr-x  2 gomez gomez  4096 2010-06-07 12:23 DYN3D/
-rw-r--r--  1 gomez gomez    17 2010-06-07 12:23 Makefile.am
[gomez@localhost src]$ cd DYN3D
[gomez@localhost DYN3D]$ ls -la
total 24
drwxr-xr-x  5 gomez gomez  4096 2010-06-07 12:23 ./
drwxr-xr-x  3 gomez gomez  4096 2010-06-07 12:23 ../
drwxr-xr-x  2 gomez gomez  4096 2010-06-07 12:23 DYN3D_CXX/
drwxr-xr-x  2 gomez gomez  4096 2010-06-07 12:23 DYN3D_SWIG/
drwxr-xr-x  2 gomez gomez  4096 2010-06-07 12:23 DYN3D_TEST/
-rw-r--r--  1 gomez gomez    44 2010-06-07 12:23 Makefile.am
```

Figure 3–12 *src* and *src/DYN3D* directories.

This tree was automatically created for a C++ component. It can be seen that the structure does not include a subdirectory for the FORTRAN source. Such subdirectory (*DYN3D_FORTRAN/*) can be created by hand and the source of the code (with the respective ***Makefile.am***) or even just the library (resulted from a previous compilation of the source) must be placed here.

It is sometimes also useful to have an additional subdirectory *DYN3D_PYTHON/* where some Python scripts (related for instance with online visualization) can be situated. The ***Makefile.am*** in this case must have the rules for copying the python scripts to the final installation path. All the subdirectories, automatically created, contain templates that can be adapted to the specific component. Like an example, the *DYN3D_CXX* subdirectory is listed in Figure 3–13.

The templates can be edited and adapted to the necessities. The C++ component here defined must contain the *DYN3D* C++ class functions and header that will then call the FORTRAN subroutines of the library located in *DYN3D_FORTRAN* (obtained in advance).

```
[gomez@localhost DYN3D_CXX]$ ls -la
total 28
drwxr-xr-x   2 gomez gomez  4096  2010-06-07 12:23 ./
drwxr-xr-x   5 gomez gomez  4096  2010-06-07 12:23 ../
-rw-r--r--   1 gomez gomez   209  2010-06-07 12:23 DYN3D.cxx
-rw-r--r--   1 gomez gomez   271  2010-06-07 12:23 DYN3D.hxx
-rw-r--r--   1 gomez gomez   255  2010-06-07 12:23 main.cxx
-rw-r--r--   1 gomez gomez   589  2010-06-07 12:23 Makefile.am
```

Figure 3–13 *DYN3D_CXX* directory.

A key point in the class generation is the correct use of FORTRAN variables and functions inside the C++ class. Figure 3–14 shows an example in which a FORTRAN variable and a function can be used inside the C++ class.

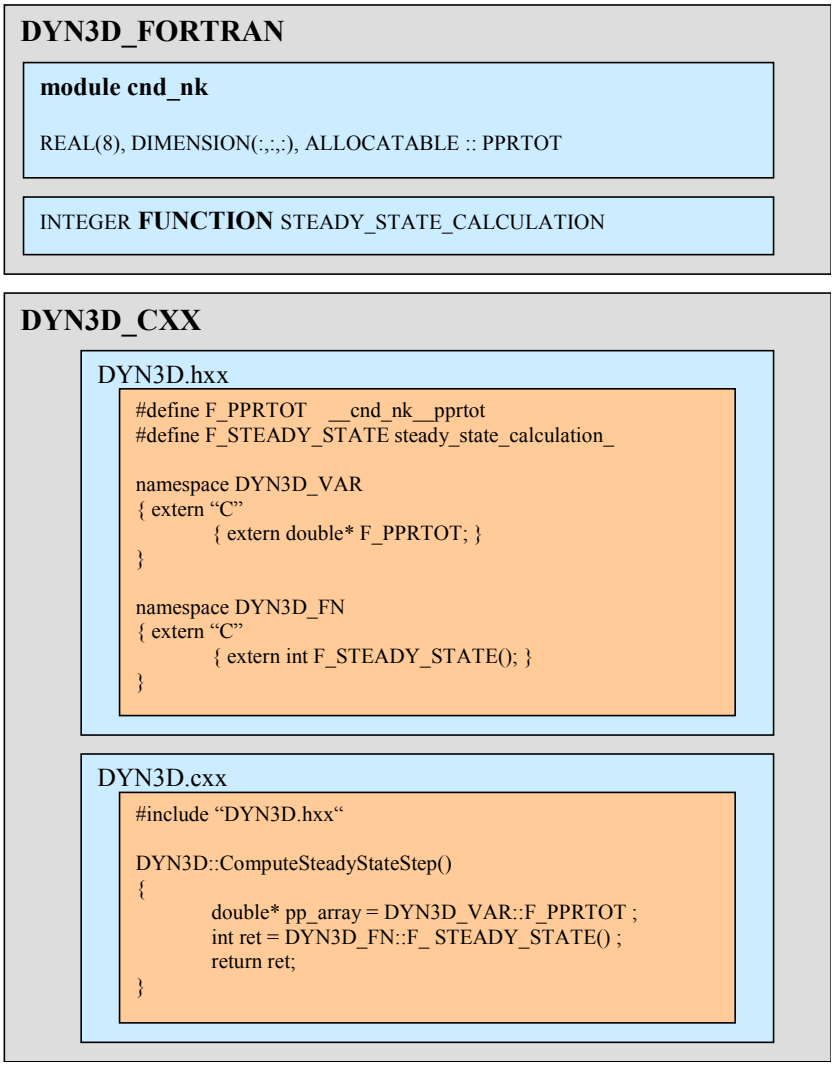


Figure 3–14 Use of FORTRAN variables and functions in the C++ class.

Inside the *DYN3D_FORTRAN* subdirectory the module ***cnd_nk*** is located (among others) in which the three dimensional variable ***PPRTOT***, containing the pin power distribution coming from the pin power reconstruction method, is defined and stored and the function

STEADY_STATE_CALCULATION, which performs a DYN3D steady state step, is also contained in this folder. These variable and function can then be used in ***DYN3D.cxx*** once they have been declared in ***DYN3D.hxx*** as external variables, as it is shown in Figure 3–14.

The ***main.cxx*** is the main program used for testing the correct compilation of the class, and can contain a call to the different functions of the class. Figure 3–15 shows an example of a main program calling the function defined in Figure 3–14. A similar test can be done by means of a Python script as it is discussed below.

DYN3D_CXX

main.cxx

```
#include "DYN3D.hxx"
#include <stdlib.h>

using namespace std;
int main(int argc, char ** argv)
{
   if (getenv("SALOME_trace") == NULL )
          setenv("SALOME_trace","local",0);
   DYN3D myDYN3D;
   int flag = myDYN3D.ComputeSteadyStateStep();
   if (flag == 1 )
   {
          cout<<"---- Error while execution ----"<<"\n";
    }
}
```

Figure 3–15 The main program ***main.cxx***.

Finally the ***Makefile.am*** of the folder *DYN3D_CXX* must define the rules and flags for compiling the class. It is important here to include a compilation rule with the path of the FORTRAN library contained in the *DYN3D_FORTRAN* folder. An example of this ***Makefile.am*** can be seen in Figure 3–16.

DYN3D_CXX

Makefile.am

```
include $(top_srcdir)/adm/unix/make_begin.am

lib_LTLIBRARIES = libDYN3DCXX.la
libDYN3DCXX_la_SOURCES = DYN3D.cxx
libDYN3DCXX_la_LIBADD   = -L$PATH/TO/DYN3D_FORTRAN -lDYN3DF

# exported headers
library_includedir=$(includedir)
library_include_HEADERS = DYN3D.hxx

bin_PROGRAMS = DYN3D_test
DYN3D_test_SOURCES = main.cxx
DYN3D_test_LDADD   = -L. -lDYN3DCXX

include $(top_srcdir)/adm/unix/make_end.am
```

Figure 3–16 ***Makefile*** for the C++ class.

There are still two more folders that contain templates and that must be modified according to the necessities. They are *DYN3D_SWIG* and *DYN3D_TEST*.

The first one (*DYN3D_SWIG*) is devoted to take the C++ declarations and create wrappers needed to access the C++ class functions from Python. As its name indicates it uses the software SWIG. SWIG receives as an input an interface file (***DYN3D.i***), then produces a file ***DYN3D_wrap.cxx*** that should be compiled and linked to the rest of the program [Beazley1997]. The interface file is automatically created by the ***SA_new_cpp_component*** script. Figure 3–17 shows an example of the content of the *DYN3D_SWIG* folder.

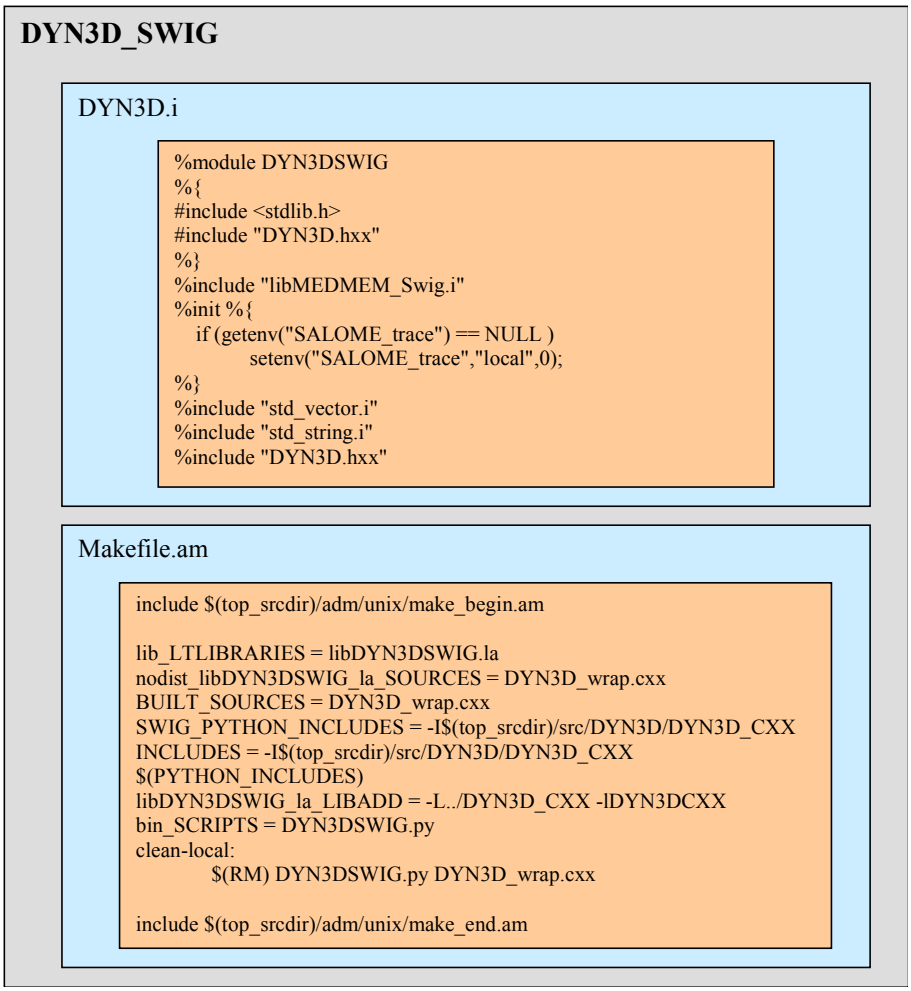


Figure 3–17 *DYN3D_SWIG* directory: input file for SWIG and ***Makefile***.

Finally the subdirectory *DYN3D_TEST* includes a Python script as an additional method for testing the correct compilation of the C++ class. This Python script uses the ***DYN3DSWIG*** module obtained via SWIG in the compilation of *DYN3D_SWIG* subdirectory. Figure 3–18 shows the content of *DYN3D_TEST*.

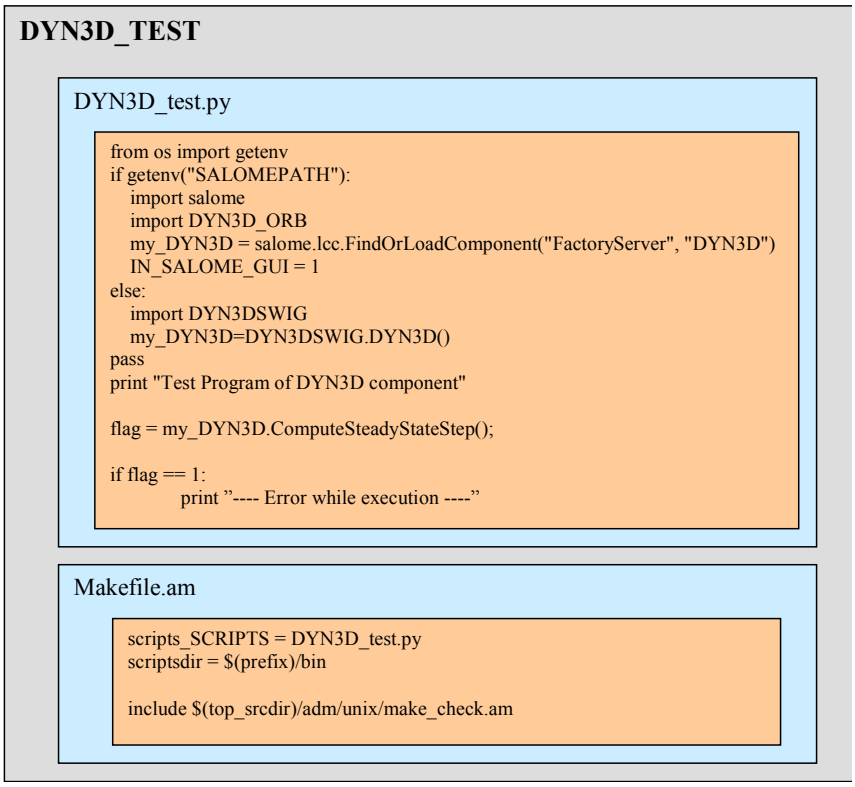


Figure 3–18 *DYN3D_TEST* directory: Python script and ***Makefile***.

3.6.3.3 The ***SA_build*** script (Compilation and Testing)

Once the DYN3D C++ class is completely defined it is time to compile the component and test it. As it was discussed before, the component is built under ***Automake*** so the build procedure must be done as described in section 3.6.3.1 or in [GNUAutotools]. The procedure is, if not complicate, at least laborious and the best way for doing it is via a script. SALOME has a script called ***SA_build*** that performs the build procedure. The use of this script is illustrated in Figure 3–19.

```
[gomez@localhost salome]$ SA_build DYN3DCPP_SRC
…
…
  [OK] Compile :  /DYN3DCPP_SRC
  [OK] Install  :  /DYN3DCPP_SRC
[gomez@localhost salome]$
```

Figure 3–19 The ***SA_build*** script.

At the end if there were no problems in the compilation and installation process the status of both processes must be OK. In case of problems the compilation process will break and a compilation message will be showed with a respective KO Compile and KO Install.

The script creates two subdirectories, one for building (*DYN3DCPP_BUILD*) and the other one for installing (*DYN3DCPP_INSTALL*) and follows the usual building procedure (configure, make and make install). All the libraries, binaries and includes must be locate in the *DYN3DCPP_INSTALL* directory.

```
[gomez@localhost salome]$ ls -la
...
drwxr-xr-x   3 gomez gomez   4096   2010-06-07 12:29 DYN3DCPP_BUILD/
drwxr-xr-x   5 gomez gomez   4096   2010-06-07 12:29 DYN3DCPP_INSTALL /
drwxr-xr-x   5 gomez gomez   4096   2010-06-07 12:29 DYN3DCPP_SRC /
...
[gomez@localhost salome]$ ls –la DYN3DCPP_INSTALL
total 20
drwxr-xr-x   5 gomez gomez   4096   2010-06-07 12:29 ./
drwxr-xr-x   91 gomez gomez  4096   2010-06-07 12:29 ../
drwxr-xr-x   2 gomez gomez   4096   2010-06-07 12:23 bin/
drwxr-xr-x   2 gomez gomez   4096   2010-06-07 12:29 include/
drwxr-xr-x   2 gomez gomez   4096   2010-06-07 12:29 lib/
```

Figure 3–20 Structure of the compiled component.

At this point is possible to test the class. All the executables or Python scripts devoted to test the class (described in the past subsection) must be included in the bin subdirectory of the *DYN3D_CPP_INSTALL* folder. There are two ways for testing: via binary (***DYN3D_test***) or via Python script (***DYN3D_test.py***).

The binary option (created by means of ***Makefile.am*** from Figure 3–16) can be run as usual:

```
[gomez@localhost salome]$ DYN3DCPP_INSTALL/bin/DYN3D_test
---- Error while execution ----
[gomez@localhost salome]$
```

Figure 3–21 DYN3D running via binary.

If there were problems, the error messages defined in the ***main.cxx*** file will be here shown.

The Python option requires additional update of paths:

```
[gomez@localhost salome]$ export
PYTHONPATH=$PYTHONPATH:DYN3DCPP_INSTALL/bin:DYN3DCPP_INSTALL/lib
[gomez@localhost salome]$ export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:DYN3DCPP_INSTALL/lib
```

Figure 3–22 Update of paths for running DYN3D via Python.

For testing via python script:

```
[gomez@localhost salome]$ python
>>> import DYN3D_test
Test Program of DYN3D component
---- Error while execution ----
[gomez@localhost salome]$
```

Figure 3–23 DYN3D running via Python.

These are the two options for debugging and testing the class implementation outside SALOME. Once everything is working fine, it is time for integrating the component.

3.6.3.4 The ***hxx2salome*** script (SALOME component generation)

When the C++ engine is finished and tested, the final step is the integration inside SALOME. This is done using the ***hxx2salome*** tool. The generator is a script written in bash, and it manages:

- the code generation,
- the compilation of the generated module and
- the update of SALOME environment file.

A Graphical User Interface, named ***ghxx2salome*** was also developed to wrap the script. Before using the ***hxx2salome*** script some changes can be done in order to facilitate the usage (although they are not mandatory because the arguments can be passed also as options). For configuring the script, the following two variables can be set:

- ENVIRON_FILE: Is the SALOME environment file used for compilation. If present, ***hxx2salome*** will propose to compile the new module (by sourcing ENVIRON_FILE, and executing ***build_configure***, ***configure***, ***make*** and ***make install***). It will also update this file with the new environment variables necessary for running the new generated module. This environment file can also be passed using the –e option.

- CONFIGURE_OPTION: define options passed to configure (for example –disable-debug or –enable-production). This one cannot be passed by argument to the script and by default is no option.

The command for running the script is shown in Figure 3–24.

```
[gomez@localhost salome]$ hxx2salome [options] CPPdir CPP.hxx libCPP.so SALOMEdir
```

Figure 3–24 The hxx2salome script.

where the mandatory components are:

- *CPPdir*: the installation directory (absolute path) of the C++ standalone component,
- ***CPP.hxx***: the header name of the component,

- ***libCPP.so***: the name of the shared library,
- *SALOMEdir*: the directory where the generated SALOME component will be installed.

The ***CPP.hxx*** and ***libCPP.so*** have to be found in *CPPdir*. In addition, the following options can be used for transmitting information to the generator:

- -h: help
- -c: to compile the component after code generation,
- -l: to launch SALOME with the component after compilation,
- -e: to specify a SALOME environment file to source (for compiling).
- -s: script extension (sh or csh)
- -g: to create a gui part in the component building tree

The script gives user information on what has been done (checking arguments, extraction of public function, the generated IDL, etc…). The use of this script for the DYN3D component is depicted in Figure 3–25.

```
[gomez@localhost salome]$ hxx2salome –c –s sh –e ${SALOME}/env_products.sh
${SALOME}/DYN3DCPP_INSTALL/ “DYN3D.hxx” “libDYN3DCXX.so” ${SALOME}/
…
…
Generation done
[gomez@localhost salome]$
```

Figure 3–25 ***hxx2salome*** for DYN3D.

in which ${SALOME} is the path for SALOME.

The ***ghxx2salome*** GUI allows selecting the arguments with a file browser, avoiding spelling mistakes in file names that could cause script abortion. Details about the process done by the ***hxx2salome*** script (out of the scope of this document) can be found in reference [Crouzet2].

Once the generation has been done, the component can be used inside SALOME, from the Python embedded console, or from ***Supervision*** module. For doing that it is necessary first to source the environment file (with the new changes already added by ***hxx2salome***) and then to run the script ***runSalome*** for starting SALOME:

```
[gomez@localhost salome]$ source env_products.sh
[gomez@localhost salome]$ runSalome --modules=DYN3D
```

Figure 3–26 Starting of SALOME using the ***runSALOME*** script.

3.6.4 The Data Exchange Model (DEM/MED)

The coupling of two or more different codes inside SALOME is strongly dependent on the way in which these codes manage themselves to receive and transmit data. In SALOME the Data Exchange Model (DEM/MED) has been chosen as a reference format for mesh and fields manipulation [MED2003]. This data exchange can be achieved either through files using the MED – file formalism or directly through memory with the MED – memory (MEDMEM) library [Godbronn2006].

The Med – libraries are organized in multiple layers:

- The MED–file layer: C and FORTRAN API to implement mesh and field objects.
- The MED–memory level: C++ API to create and manipulate mesh and field objects in memory.
- Python API: generated using SWIG, it wraps the complete C++ API of the MED – Memory.
- CORBA API: used to simplify distributed computation inside SALOME (Server Side).
- MED–Client classes: used to simplify and optimize interaction of distant objects within the local solver.

Thanks to the MED–memory, any component can access a distant mesh or field object. Two codes running on different machines can thus exchange meshes and fields. These meshes and fields can easily be read or written in a MED–file format enabling access to the whole SALOME suite of tools (CAD, meshing, visualization, etc…).

Although the creation of MED–files can be done directly in the FORTRAN or C source, the implementation of MED–memory can be only performed in the C++ class definition. Thus, assuming that the original source is written in FORTRAN (as is the case of DYN3D), at least two strategies can be explored:

- Direct creation of MED–files in the source with the MED FORTRAN API functions, followed by a read and dump in MED–memory (in the C++ class) for a final exchange of data and write of results in files, or
- Creation of MED–memory direct in the C++ class for exchange of data followed for a later write of MED–files for post-processing.

Both options have advantages and disadvantages. The first option permits the use of SALOME as a post-processor without having the code integrated in the platform (just by opening the MED–files already created for visualization purposes), however it implies a redundant effort (writing MED–files in FORTRAN and in C++). The second option is more direct, the creation of MED–memory in the C++ API for exchanging and online visualization and the writing of MED–files for post-processing purposes, but with the disadvantage that the code must be integrated or at least compiled through a C++ API in order to have MED–files with results.

Since in this work the exchange of data online (coupling of codes) is the main task, the second one sounds like the most appropriate option to be used and the following discussion will be focused on it (MED–memory C++ API).

3.6.4.1 MED–memory API

The MED–memory library (available in C++ and Python) uses two namespaces: ***MEDMEM*** which is the general namespace where the main classes are defined and the equivalent in English ***MED_EN*** (***MEDMEM*** will be used). At a basic usage level, the API consists in few classes which are located in the ***MEDMEM*** C++ namespace (see Figure 3–27).

- ***MED***: the global container;
- ***MESH***: the class containing 2D or 3D mesh objects;
- ***SUPPORT***: the class containing mainly a list of mesh elements;
- ***FIELD***: the class template containing mainly a list of values lying on a particular support.

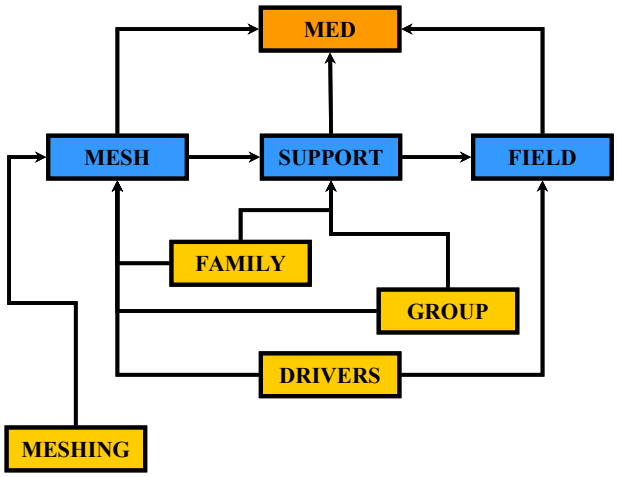


Figure 3–27 Structure of MED–memory API classes.

The API of those classes is quite sufficient for most of the component integrations in the SALOME platform. The use of the MED–memory libraries may make easier the code coupling in the SALOME framework. With these classes it is possible to:

- Read/write meshes and fields from MED–files;
- Create fields containing scalar or vectorial values on list of elements of the mesh;
- Communicate these fields between different components;
- Read/write such fields.

A more advanced usage of MED–memory is possible through other classes that include:

- ***GROUP***: a class inherited from ***SUPPORT*** used to create supports linked to mesh groups. It stores restricted list of elements used to set boundary conditions or initial values.

- ***FAMILY***: a class used to manipulate a certain kind of support and does not intersect with each other.
- ***MESHING***: devoted to build meshes from scratch, it can be used to transform meshes from a specific format to the MED format or to integrate a mesher within SALOME platform.
- ***DRIVERS***: enable the user to get a fine control of the I/O operations.

Figure 3–27 shows how the ***MED*** container controls the life cycle of all the objects it contains: its destructor will destroy all the objects it aggregates. On the other hand, the life cycle of ***MESH***, ***SUPPORT*** and ***FIELD*** objects are independent. Destroying a ***SUPPORT*** for instance, will have no effect on the ***MESH*** which refers to it. But it is important to maintain the link: a ***MESH*** aggregates a ***SUPPORT*** which aggregates a ***FIELD***. If it is necessary to delete ***MED*** – memory objects, the ***FIELD*** has to be deleted first, then the ***SUPPORT*** and finally the ***MESH***.

3.6.4.2 Create a ***MESHING*** object

The class ***MESHING*** is the one useful when it is necessary to create a ***MESH*** from scratch. This process is usually done for every code after reading of input data. The code must define its domain and the ***MESH*** for a later dump of results as a ***FIELD*** representation. The creation of a ***MESHING*** object implies the definition of coordinates and connectivities:

Coordinates

The first step is the definition of coordinates for the ***MESH***. The method ***setCordinates*** deals with this task. The use of this method is illustrated in Figure 3–28 after the creation of a variable myMeshing of the class ***MESHING***.

```
MESHING myMeshing ;
myMeshing.setCoordinates(SpaceDimension, NumberOfNodes, Coordinates, System, Mode)
```

Figure 3–28 The ***setCoordinates*** method.

The input arguments are:

- SpaceDimension: Type integer with the dimension of the domain considered.
- NumberOfNodes: Type integer with the total number of nodes considered.
- Coordinates: Type array of double with the coordinates of the points defining the mesh.
- System: Type string with the system of coordinates desired: “CARTTESIAN”, “CYLINDRICAL”, “ESPHERICAL”.
- Mode: Type string defining the way for input the coordinates: The coordinates can be given in a full interlace way “MED_FULL_INTERLACE” (x1, y1, z1, x2, y2, z2 …) or without interlace (x1, x2… y1, y2… z1, z2 …) “MED_NO_INTERLACE”.

Connectivities

Once the coordinates have been defined, the next step is to define the connectivity of the coordinates. For each element type, the reference connectivity is used to recreate cells, edges or faces of elements. First of all it is necessary to define connectivity of cell elements. After that, it is possible to define constituent connectivity (if necessary) for faces and/or edges. There are several kinds of connectivities based in the kind of cell desired. For instance, a one dimensional cell (a segment) can be created with two vertices a triangle with three and a quadrangle with four. The case with polyhedrons is more complex. The connectivities must follow a rule for building the polyhedrons based on the coordinates of the vertices. In case of a core with rectangular geometry, for instance, the hexahedrons are the best option for meshing. Figure 3–29 shows a hexahedron (cube) and the order in which the connectivities must be given, based in the vertices (coordinates defined in the past subsection) [MED2003]. For each kind of connectivity considered, the methods below could be used in the following order:

- ***setNumberOfTypes***: sets the number of different geometric types.
- ***setTypes***: sets the different geometric types (MED_TETRA4, MED_PYRA5, MED_HEXA8, etc… [medMEM]). Types must be given in increasing order of number of nodes for this type.
- ***setNumberOfElements***: sets the number of elements for each geometric type.
- ***setConnectivity***: sets the connectivity for each geometric type.

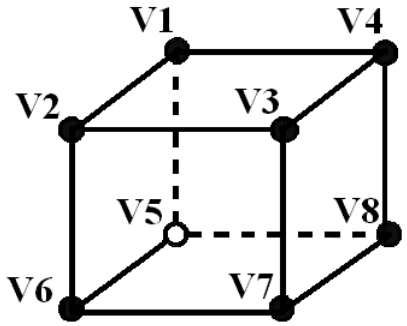


Figure 3–29 Order of connectivities for a hexahedron (cube).

Another example can be shown in the Figure 3–30. A mesh with 2 different kinds of geometric types (triangle and quadrangle) has been defined. The number of elements for each type (3 and 2 respectively) and then the connectivities for each geometric type, based in the coordinates previously defined. As it is expected, for creation of a MED_TRIA3 it is needed 3 connectivities, having 3 MED_TRIA3 results in 12 connectivities. Analogous, for the MED_QUAD4 geometric type, 4 connectivities are needed for each of the two quadrangles considered, resulting in 8 connectivities passed in the last line of the example.

```
MESHING myMeshing ;
myMeshing.setCoordinates(SpaceDimension, NumberOfNodes, Coordinates, System, Mode);
myMeshing.setNumberOfTypes(2,MED_CELL);
myMeshing.setTypes({MED_TRIA3, MED_QUAD4}, MED_CELL);
myMeshing.setNumberOfElements({3,2}, MED_CELL);
myMeshing.setConnectivity({1,2,3,6,8,9,4,5,6}, MED_CEL, MED_TRIA3);
my.Meshing.setConnectivity({1,3,4,5,4,5,7,8}, MED_CELL, MED_QUAD4);
```

Figure 3–30 The setConnectivity method.

3.6.4.3 Creation of a Mesh for an assembly in a pin-by-pin resolution

For the DYN3D integration and, especially, for the pin power reconstruction method, the creation of a ***MESHING*** for a pin-by-pin distribution inside a PWR fuel assembly (17 X 17) like the one showed in Figure 3–31 must be considered.

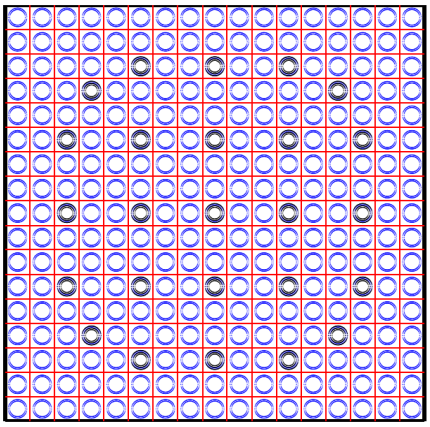


Figure 3–31 Radial view of a PWR Fuel Assembly.

Assuming that the assembly is centred in the origin, it has a pitch of 21.4 cm, and an active height of 360 cm, part of the code for the creation of the MESHING object representing this assembly can be seen in Figure 3–32. There it is assumed that the vectors *x_coor*, *y_coor* and *z_coor*, contain the coordinates of each direction, and it is necessary to create the structured coordinates (*x1, y1, z1, x2, y2, z2* …) in the vector *coor_ xyz*. The creation of connectivities is the next step. At the end, the creation of ***MESHING*** is shown.

Some useful variables for understanding the example code are: *n_pin* = 17, *n_z* = number of axial layers, *COORD_DIM* = 3, *HEXA_DIM* = 8, *MESH_DIM* = 3, *MESH_NAME_PPR* = ‘DYN3DMESHPPR’.

DYN3D_CXX

DYN3D.cxx

```
void DYN3D::_Generate_HEXA8_pprmesh()
{
…
// Creation of structured coordinates
   for (int iz = 0; iz<z_coor.size(); iz++) {
      for (int iy = 0; iy<y_coor.size(); iy++){
         for (int ix = 0; ix<x_coor.size(); ix++){
                  coor_xyz.push_back(x_coor[ix]);
                  coor_xyz.push_back(y_coor[iy]);
                  coor_xyz.push_back(z_coor[iz]); } } }

// Creation of connectivities
            np1 = n_pin + 1 ;
            np1_2 = np1 * np1 ;
            index = 0 ;
   for (int inz = 1; inz<=n_z; inz++){
      for (int iny = 1; iny<=n_pin; iny++){
         for (int inx = 1; inx<=n_pin; inx++){
            index = np1_2 * (inz-1) + np1 * (iny-1) + inx ;
            connec.push_back(index + np1_2);
            connec.push_back(index + np1_2 + 1);
            connec.push_back(index + np1_2 + np1 + 1);
            connec.push_back(index + np1_2 + np1);
            connec.push_back(index);
            connec.push_back(index + 1);
            connec.push_back(index + np1 + 1);
            connec.push_back(index + np1); } } }

// Definition of mesh for ppr
            int n_coor=(int)(coor_xyz.size()/COORD_DIM);
            int n_elem=(int)(connec.size()/HEXA_DIM);
            string names[COORD_DIM]={"X","Y","Z"};
            medGeometryElement types[1] = {MED_HEXA8}
            MESHING* my_meshing = new MEDMEM::MESHING();
            my_meshing->setName(MESH_NAME_PPR);
            my_meshing->setSpaceDimension(MESH_DIM);
            my_meshing->setCoordinates(COORD_DIM, n_coor, coor_xyz.data(), "CARTESIAN",
MED_FULL_INTERLACE);
            my_meshing->setCoordinatesNames(names);
            for(int i=0;i<COORD_DIM;i++) my_meshing->setCoordinateUnit(m_mesh_unit,i);
            my_meshing->setNumberOfTypes(1 , MED_CELL);
            my_meshing->setTypes(types , MED_CELL);
            int an_elem[1]={n_elem};
            my_meshing->setNumberOfElements(an_elem,MED_CELL);
            my_meshing->setMeshDimension(MESH_DIM);
            my_meshing->setConnectivity(connec.data(),MED_CELL,MED_HEXA8);
m_ppr_mesh = my_meshing ; …
}
```

Figure 3–32 Part of the ***DYN3D.cxx*** class for creation of PPR ***MESHING***.

3.6.4.4 Creation of ***SUPPORT*** and ***FIELD*** objects.

As shown in Figure 3–27, once the ***MESH*** is created, the dump of data inside the ***MED*** field is done through the ***FIELD*** class based in a ***SUPPORT***.

To create a ***SUPPORT***, it is necessary to give a reference to a ***MESH*** object, its name and to specify on which entity it applies (MED_CELL, MED_FACE, MED_NODE, etc…). Once the support has been created, the method ***FIELD*** with some specific data can be used for dumping data in the ***MESH***.

A ***FIELD*** is characterized by its name, an operational description, and its calculation time, i.e., an iteration number (or time step number), an order number (used when there are internal iterations in a time step) and the time corresponding to this iteration number. For the creation of a ***FIELD*** it is necessary to know its ***SUPPORT*** and the number of components. The ***FIELD*** class has several methods for giving a name or an optional description of the ***FIELD***, for setting the iteration number and time and of course for setting values. The best way for having a feeling of a ***FIELD*** class is to follow the example already discussed in the previous subsection.

Figure 3–33 shows part of the example code for generating a ***SUPPORT*** for the PWR assembly.

DYN3D_CXX

DYN3D.cxx

```
void DYN3D::_GenerateSupport_ppr()
{
        _DeleteSupport_ppr(); // deletes previously defined SUPPORT

        if(!m_ppr_mesh) return; // if there is no mesh returns

        SUPPORT* support=new SUPPORT((MESH*)m_ppr_mesh,"On_Cells",MED_CELL);

        if(!support) return; // if there were problems creating the support returns

        m_ppr_support=support;

        return;
}
```

Figure 3–33 Creation of a ***SUPPORT*** for the ***MESH***.

In Figure 3–34, the ***FIELD*** construction and its use are illustrated assuming that there exists a structure *datappr* with name, description and unit.

Details about all the methods available can be found in [Godbronn2006] and [MED2003].

3.6.4.5 Writing ***MESH*** and ***FIELDS*** on files

Finally it is also useful to be able to have access to the calculated data as a MED-file, for post-processing purposes. To write a complete ***MESH*** object in an available writing format, the method ***addDriver*** must be used followed by the method ***write***.

The Figure 3–35 shows the procedure for writing the resulting ***MESH*** and fields created in the previous subsections.

DYN3D_CXX

DYN3D.cxx

```
FIELD<double>* DYN3D::GetPinPowerField(const bool norm)
{
        double* pp_array=DYN3D_VAR::F_PPRTOT ;
        if(pp_array) _GetPPRDataField(pp_array);
        else _DeletePPRField();
        return m_ppr_field;
}

void DYN3D::_GetPPRDataField(const double* data)
{
        _DeletePPRField();// deletes previously defined FIELD
        FIELD<double>* field=new FIELD<double>(m_ppr_support,1);
        if(!field) return; // if there were problems creating the field returns

        field->setName(datappr.name);
        field->setComponentName(1,datappr.name);
        field->setComponentDescription(1,datappr.description);
        field->setMEDComponentUnit(1,datappr.unit);
        field->setTime(_GetCurIterationTime());
        field->setIterationNumber(iter);
        field->setOrderNumber(1);

        double val ;
        int index=0;

        for(int iza=0;iza<n_z;iza++)
        {
                for(int il=0; il<n_pin * n_pin; il++)
                {
                        val = data[index];
                        field->setValueIJ(index+1,1,val);
                        index = index + 1;
                }
        }
        m_ppr_field=field;
return;
}
```

Figure 3–34 ***FIELD*** creation and use.

DYN3D_CXX

DYN3D.cxx

```
void DYN3D::_Writeppr_mesh()
{
…
        // creation of the mesh in a file
int id=m_ppr_mesh->addDriver(MED_DRIVER,"ppr.med",MESH_NAME_PPR);

        m_ppr_mesh->write(id);

        // writing a field in the file
int idf=m_ppr_field->addDriver(MED_DRIVER,"ppr.med",m_ppr_field->getName());

        m_ppr_field->write(idf);
…
}
```

Figure 3–35 Writing a ***MESH*** into a file.

3.6.5 DYN3D integrated in SALOME

Based in the discussion of the past subsection, an Application Programming Interface (API) for DYN3D has been developed [Gommlich2010]. This API contains a C++ class with 34 functions and it is responsible for the creation of the data exchange meshes, for the initialization and finalization of runs, for the data exchange (sending power distributions and receiving the feedback parameters) and for the assessment of the convergence. As was discussed in the previous section, special attention must be put on the reading and writing in the MED format.

3.6.5.1 Main Steps for the Integration of DYN3D

In order to perform the integration, the following steps were identified, constituting the integration strategy:

- For the communication within the NURESIM platform, changes had to be introduced into the FORTRAN source code of DYN3D. It was needed to modularize the code. The original code structure of DYN3D was preserved during the work on the modularization. The necessary modifications were introduced in such a way that the stand-alone execution of DYN3D is not affected. It should be noted that all these modifications are part of the new standard version of DYN3D.

- The FORTRAN source of DYN3D with the mentioned changes was compiled and merged into a static library.

- The C++ class was developed. The C++ functions in the class are devoted to perform different services calling the FORTRAN subroutines contained in the static library. Most of the effort was focused in the generation of meshes in MED format. After compilation a dynamic library with all the DYN3D functionalities is available. At this point DYN3D can be executed using Python scripts outside SALOME.

- The final step is the integration of the class into SALOME using the ***hxx2salome*** tool.

DYN3D as integrated in SALOME is able not only to perform stand alone calculations with its internal thermal-hydraulic model FLOCAL, but also it is ready for interaction with the thermal-hydraulic code FLICA4 already integrated in the platform or with any other thermal-hydraulic code that would be integrated. Thereby it is possible to choose between two options of calculation:

- Coupling with the internal thermal-hydraulic model (FLOCA) or

- Coupling with the external thermal-hydraulic code FLICA4.

The DYN3D services developed for the integration obeys the typical Neutronic – Thermal-hydraulic coupling strategy. The power obtained from the neutronic code in every node of the mesh, after internal convergence, is given to the thermal-hydraulic model. The thermal-hydraulic model, then, uses this power distribution, as a heat source, to perform its calculation. After internal convergence, the typical feedback parameters (moderator temperature,

boron concentration, fuel temperature and moderator density), are returned to the neutronic solver. With these new values the neutronic solver updates the cross sections, for another neutronic calculation until some criteria in the feedback parameters and in the k_{eff} or power are reached. In Table 3-IV a brief description of the 34 DYN3D functions available inside SALOME and grouped by functionality is given. The names of the DYN3D component's services are self explaining.

Table 3-IV Description of the DYN3D functions.

Group	Functions	Description
1	***SetUseFLOCAL*** ***SetUsePPR*** ***SetResultParameter*** ***InitCalc*** ***Finalize***	**Control Functions**: set the use of FLOCAL, the use of Pin Power Reconstruction, and the parameters to be stored into the MED file. Set of input path and name of the problem and close and erase of files at the end of calculation.
2	***SetCoreMeshExtended*** ***SetCoreMeshRotation*** ***SetCoreMeshTranslation*** ***GetCoreMesh*** ***ReadCoreMeshFromFile***	**Mesh and Coordinate Functions**: devoted to get the meshes and coordinates from the input or to set external core meshes.
3	***ComputeSteadyState*** ***ComputeTransientStep***	**Calculation Functions**: perform a Steady State or Transient Step depending on the problem.
4	***GetTotalPower*** ***GetKeff*** ***GetReactivity*** ***GetReactivityByPT*** ***GetAverageBoronConcentration*** ***GetAverageFuelTemperature*** ***GetAverageModeratorTemperature*** ***GetAverageModeratorDensity*** ***GetMaxFuelTemperature*** ***GetMaxModeratorTemperature***	**Scalar Functions**: extraction of scalar results as: total power, k effective, total reactivity or per coefficient, average global values of the feedback parameters, and maximal values of safety parameters.
5	***GetCorePowerField***	**Field function**: extraction of power from DYN3D.
6	***GetFuelTemperatureField*** ***GetModeratorDensityField*** ***GetBoronConcentrationField*** ***GetModeratorTemperatureField***	**Field Get Feedback functions**: extraction of feedback parameters when they are calculated with the internal FLOCAL model.
7	***SetFuelTemperatureField*** ***SetModeratorDensityField*** ***SetBoronConcentrationField*** ***SetModeratorTemperatureField***	**Field Set Feedback functions**: set of feedback parameters when the coupling is performed with the external code like FLICA.
8	***WriteCoreMeshInFile*** ***WriteCorePowerFieldInFile*** ***WriteResultsInFile***	**Writing Functions**: write of results in MED files.

3.6.5.2 Use of DYN3D inside SALOME

DYN3D can be run inside SALOME by means of several combinations of the services. The simplest case that can be calculated with the DYN3D module is a steady state case coupled with the internal thermal-hydraulic model (FLOCAL). For such calculation four functions are needed:

- ***SetUseFLOCAL***
- ***InitCalc***
- ***ComputeSteadyState***
- ***Finalize***

The linkage of the functions can be done graphically using the YACS component of SALOME, as depicted in Figure 3–36 or via Python scripts.

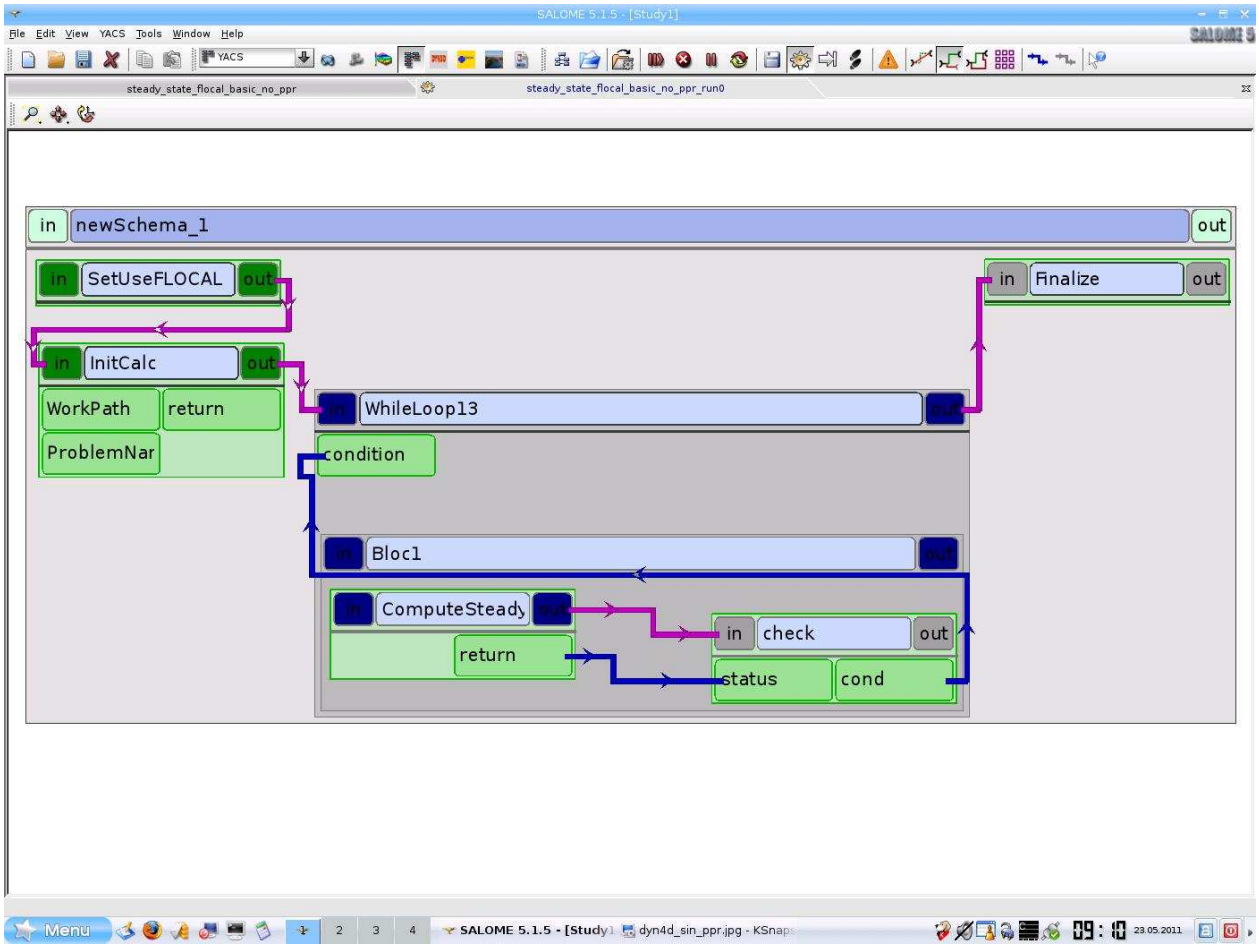


Figure 3–36 Basic Graph for computing steady state problems.

The arguments of the ***InitCalc*** function are the path of the input file and the name of the problem (usual arguments for DYN3D). The ***SetUseFLOCAL*** function will turn to true the flag for using FLOCAL and the ***ComputeSteadyState*** will perform the usual internal iteration step of DYN3D. The steady state loop (central box in Figure 3–36) can be built using a SALOME predefined computational node ***WhileLoop***, and a Python function (***check***) that will check if

the convergence criteria are met. Once the steady state is reached, the function ***Finalize*** will close and erase temporary files for ending the calculation. With this YACS graph all the steady state cases (input files for DYN3D) can be run. A more complex graph is shown in Figure 3–37, in which additional functions of the DYN3D component are used. The initial ***WriteResultsInFile*** function will define the path and name of the MED file containing the results (fields) specified by one or several functions ***SetResultParameter***, the use of FLOCAL and pin power reconstruction are set by means of ***SetUseFLOCAL*** and ***SetUsePPR*** respectively. ***InitCalc*** will start the calculation as previously described and it is followed by the function ***WriteCoreMeshInFile*** which gives the possibility to have an additional MED file with just the mesh used in the calculation. Two more functions are placed in the ***WhileLoop*** box, ***GetCorePowerField*** and ***GetFuelTemperatureField***; they are used for extracting the power distribution and fuel Doppler temperature and dumping the results in the results MED file. In a similar way, in the case of a transient calculation an additional loop for the transient step must be implemented.

With the graph mode it is easy to monitor the running code on-line. In Figure 3–36 and Figure 3–37 three different colours in the function-boxes indicate the status of the run: green (already finished), blue (running) and grey (waiting).

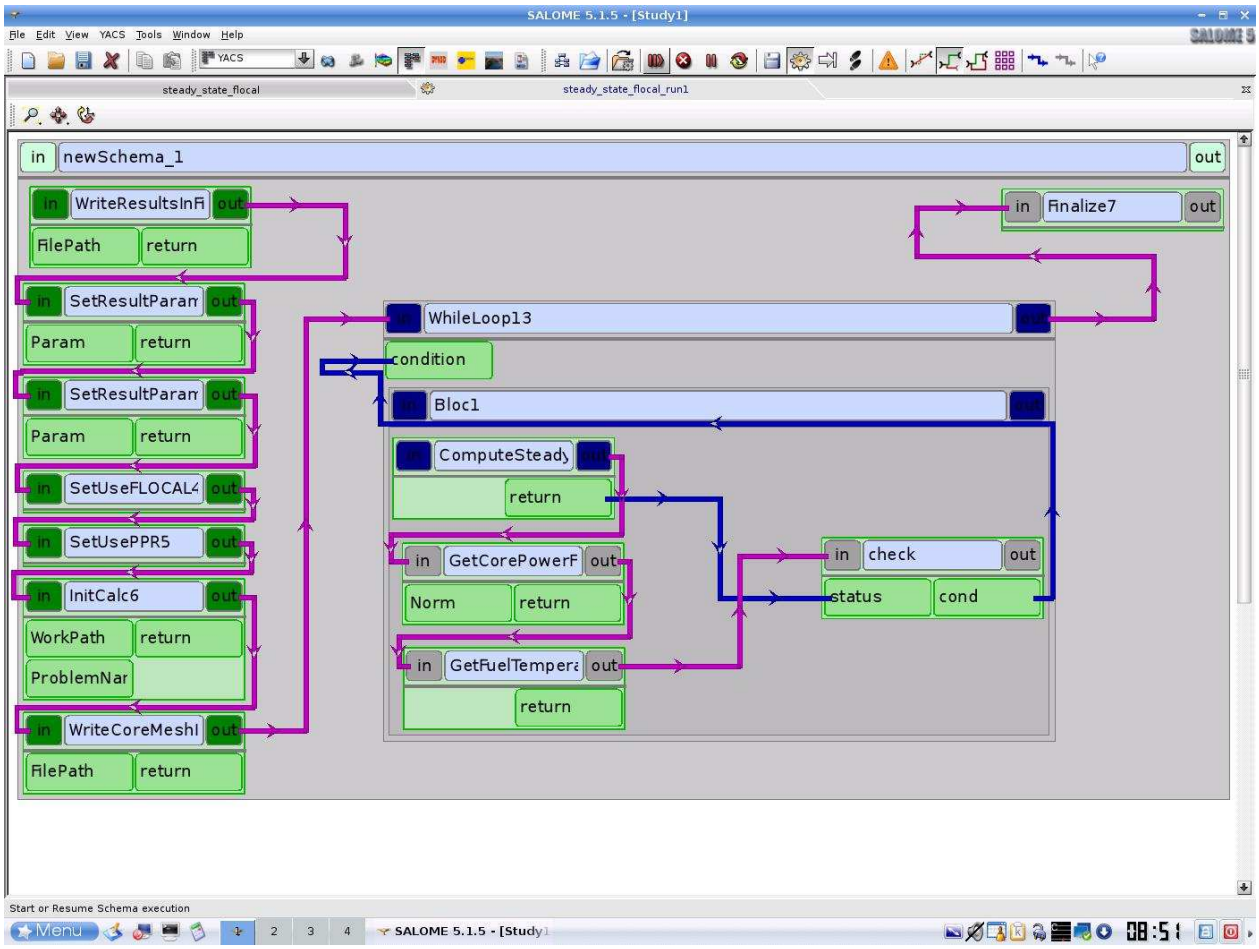


Figure 3–37 Graph for computing Steady State problems with PPR and med files creation.

3.6.6 Verification of the pin power reconstruction extension of DYN3D

DYN3D with its internal thermal-hydraulic model FLOCAL has been extensively validated [Grundmann1997], [Grundmann2006], [Grundmann2003], [Kliem2008] among others. Furthermore, the verification of the integration of DYN3D in SALOME and the coupling with the already integrated thermal-hydraulic code FLICA has been done as a part of the tasks of the NURISP project and reported in [Gommlich2010], [GomezA2010] and [Kliem22011]. As previously discussed, the pin power reconstruction extension of DYN3D it is necessary for a later coupling with the under-development version of FLICA with non-conform geometry. Thus, in this section just the verification of the pin power reconstruction extension will be presented.

3.6.6.1 New pin-by-pin mesh generator

In order to verify that the algorithm for creating the mesh works correctly, a 3x3 minicore surrounded by a row of reflector nodes was used (Figure 3–38). Such minicore has been included as a part of the definition of a boron dilution benchmark in the NURISP project [Kliem2011] and is based in the OECD/NRC PWR MOX Benchmark [Kozlowski2006]. The central assembly is composed of fresh UO_2 at 4.5% and the ones surrounding it are fresh MOX at 4.3%.

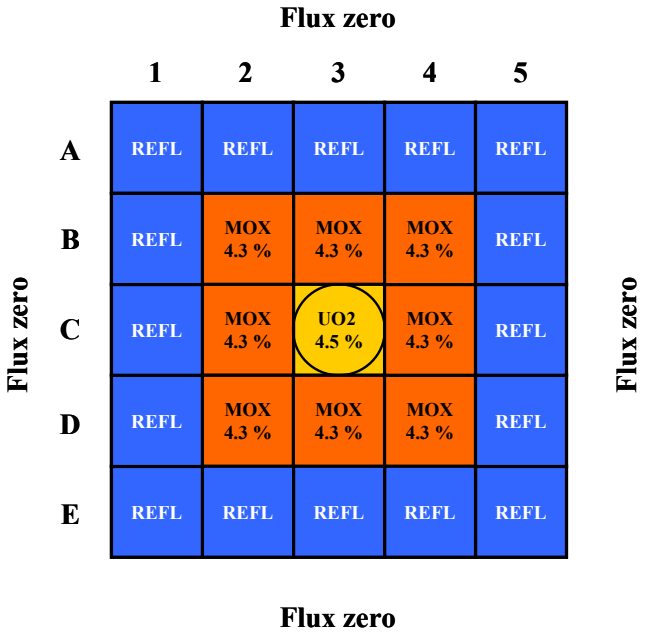


Figure 3–38 3x3 minicore based in the OECD/NRC PWR MOX benchmark.

The meshing algorithm performs a sweep over all the nodes and introduces a refinement in the ones selected for having pin power reconstruction. Figure 3–39 shows in the upper-left quadrant the nodal mesh without the use of pin power distribution. In the other quadrants, three different cases with refinement in one or several assemblies are shown. Case (b) considers a refinement in the central assembly (containing the control rod). In case (c), five sepa-

rated assemblies have been refined and, finally, in case (d) all the possible assemblies (the 9 internal assemblies) are ready for having pin power reconstruction.

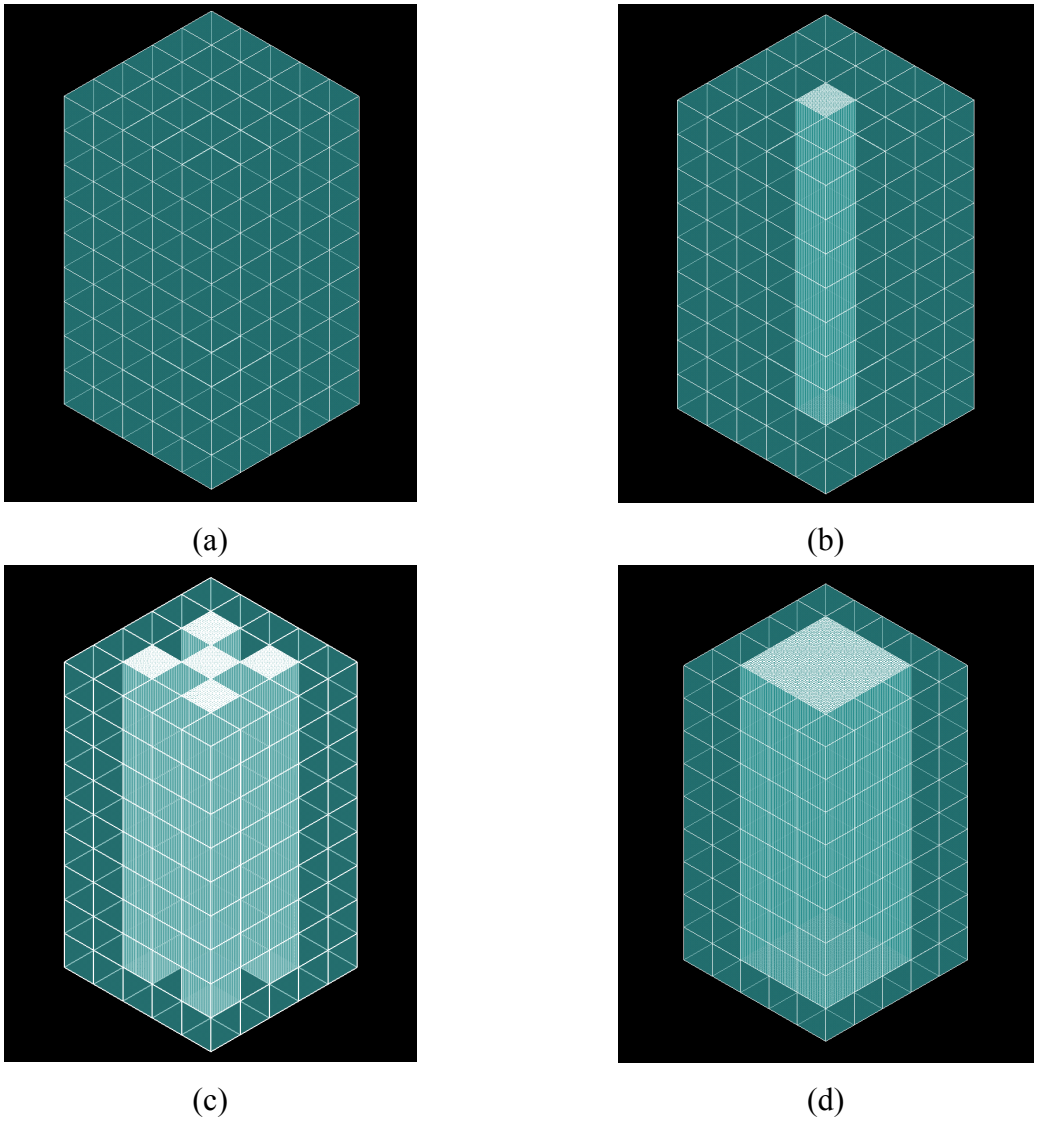


Figure 3–39 DYN3D mesh generator, (a) nodal base, (b) central assembly with PPR, (c) 5 different assemblies with PPR, (d) all possible assemblies with PPR.

3.6.6.2 Calculations with the new version of DYN3D

Once the meshing algorithm is working correctly the next step is to verify the pin power calculation.

The section 5.2 of the boron dilution Benchmark [Kliem2011] defines two calculations with the minicore configuration: a steady state hot full power (HFP) calculation and one transient calculation starting from hot zero power (HZP).

In the transient calculation a control rod partially inserted in the central assembly has to be fully ejected in 0.1 seconds leading to a very fast insertion of reactivity in the system. Such local perturbations justify the intention of having mesh refinements focused on the affected zone. The operational conditions for the minicore in the transient case are shown in Table 3-V.

Table 3-V Operational conditions for the minicore HZP calculation.

Operational condition	**Value**
Core power	1 W
Burn-up	0.0 MWd/t
Mass flow rate (core + reflector)	2053 kg/s
Mass flow rate per assembly	82.12 kg/s
Core outlet pressure	15.4 MPa
Coolant inlet temperature	286.85 °C
Control rod insertion	232.433 cm
Boron concentration	486.7 ppm
Neutronic boundary conditions (radial)	Zero flux
Neutronic boundary conditions (axial)	Zero flux

A central cut of the steady state nodal power distribution (normalized power density) can be seen in Figure 3–40. The “hottest region” of the core has been shifted to the lower part of the core due to the effect of the partially inserted control rod. Since the scenario starts with a HZP state, the power distribution is almost zero in the whole configuration and 1 W is distributed in the region without the influence of the control rod.

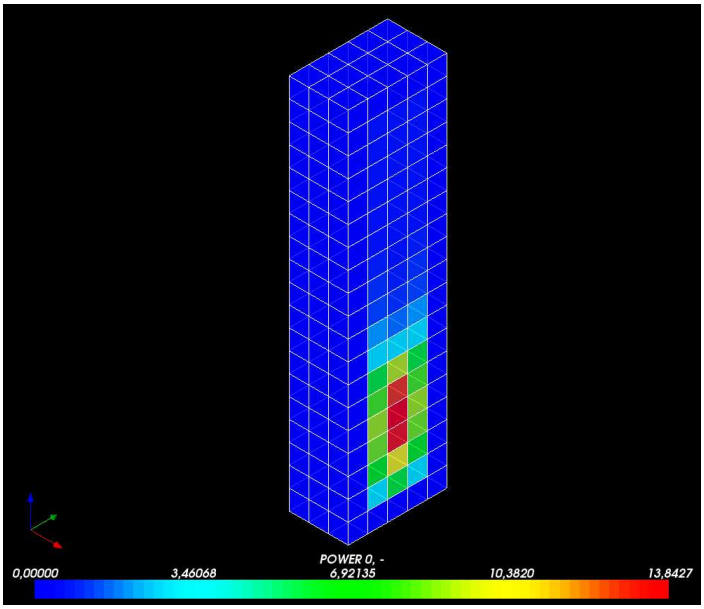


Figure 3–40 Normalized nodal power distribution for the steady state HZP scenario for the minicore.

A radial cut of the steady state in the hottest layer of the core at assembly base can be seen in the upper-left quadrant of Figure 3–41. A more detailed description of the power distribution

can be obtained using the pin power distribution method. The upper-right and lower left and right quadrants of Figure 3–41 show the same calculation but considering the different mesh refinements as presented in Figure 3–39. The internal assembly heterogeneities like the control rods can be clearly distinguished.

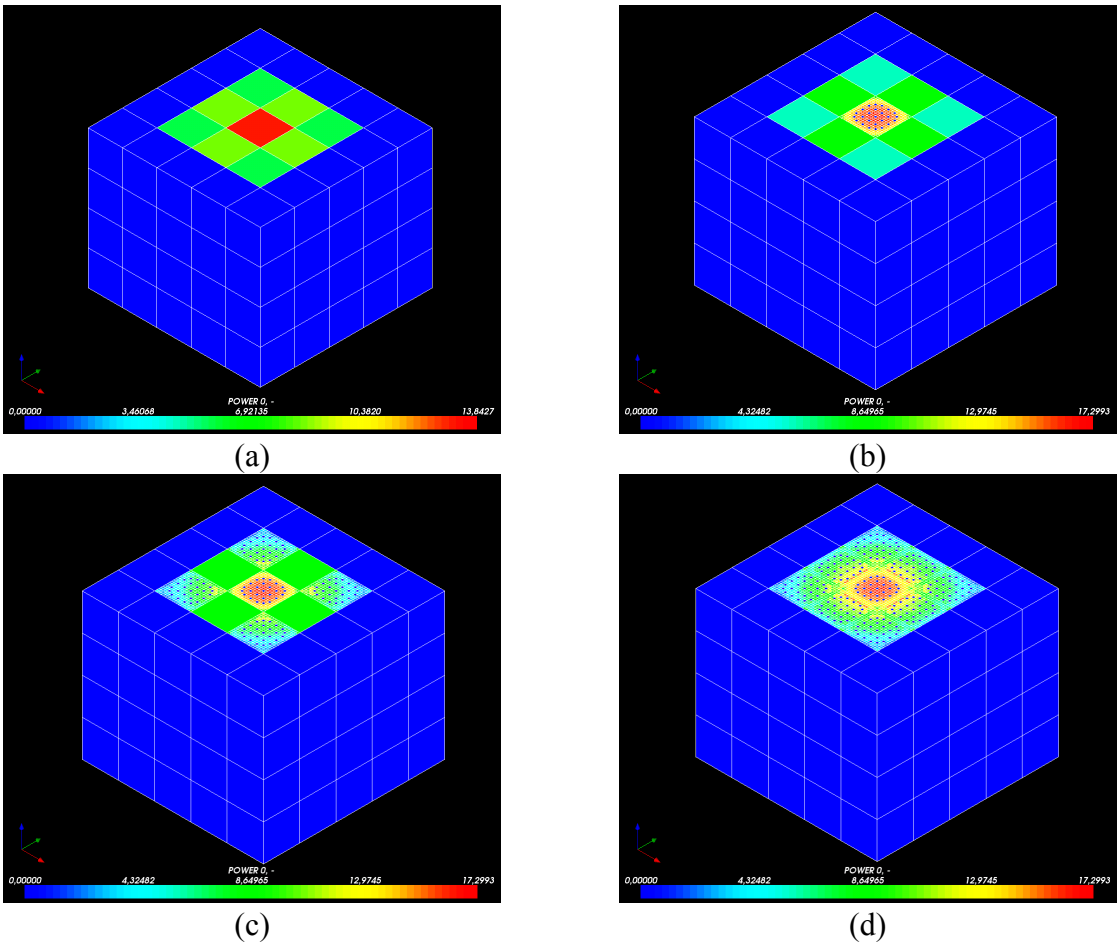


Figure 3–41 Normalized radial power distribution for the hottest layer, (a) nodal base, (b) central assembly with PPR, (c) 5 different assemblies with PPR, (d) all possible assemblies with PPR

The reallocation of power inside the assembly, due to the cell lattice structure of the fuel assemblies (containing elements that do not produce power, like guide tubes full of water or control rods), tends to shift the power density, in this symmetric case, to the centre. For this reason, a better and more detailed prediction of the safety parameters could be done by means of a hot region calculation (and not just hot channel) in which the pin power distribution can be given to a subchannel code for the calculation of local safety parameters. Such extension of the code is under-development and a brief discussion about that will be included in the outlook of this dissertation.

A detailed normalized pin power distribution with a central axial cut like the one presented in Figure 3–40 but limited to the bottom half of the core is shown in Figure 3–42.

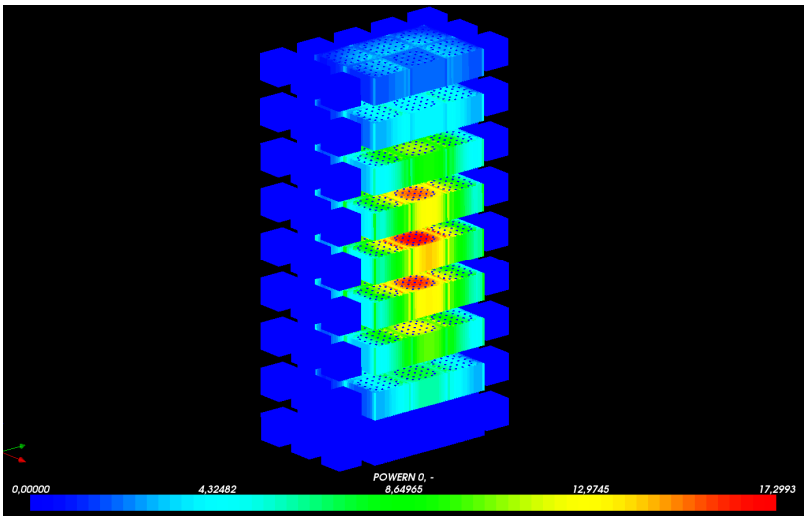


Figure 3–42 Normalized pin power distribution for the steady state HZP scenario for the minicore.

Figure 3–43 shows the normalized pin power distribution at different times during the transient calculation. The effect of the control rod withdraw can be clearly observed. The power concentrated originally at the bottom of the reactor starts moving to the top as a consequence of a reduction of the absorber (control rod ejection). After 0.1 seconds, the control rod is fully ejected and then the power stabilizes to typical axial cosine behaviour after experiencing a dramatic power peak due to the very fast reactivity insertion.

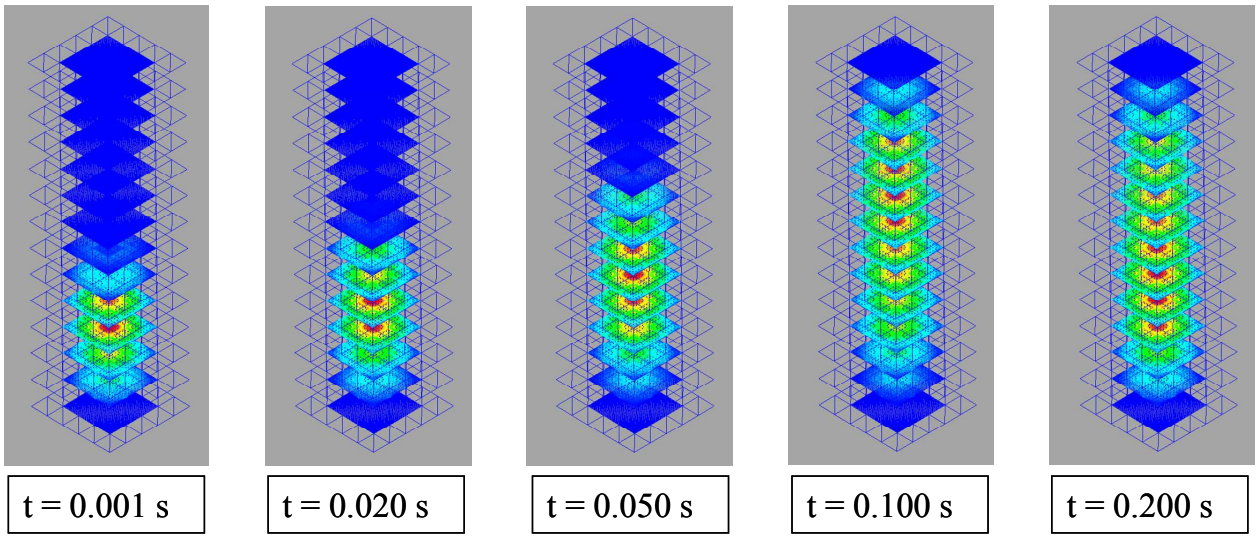


Figure 3–43 Normalized pin power distribution at different time steps during the control rod ejection.

3.6.6.3 Application to whole core problems

In order to test the extension in a more practical application, the full core based in the boron dilution benchmark [Kliem2011] has been used. The PWR core contains 193 fuel elements and the corresponding core loading is based on the OECD/NRC PWR MOX benchmark.

The material compositions and the pin and fuel assembly geometries defined in this benchmark have been used to create a new cross section library. This new library contains a wide range of feedback parameter points and additionally pin power form function data that can be used for the pin power reconstruction extension.

The core configuration, control rod banks, burn-up states and thermo-physical data is identical to that of the OECD/NRC PWR MOX benchmark [Kozlowski2003] and it is shown in Figure 3–44. The active core is surrounded by a row of reflector elements with the same width as the active fuel elements. Beside the radial row of reflector elements one reflector layer at the bottom and the top of the core is used in the calculations.

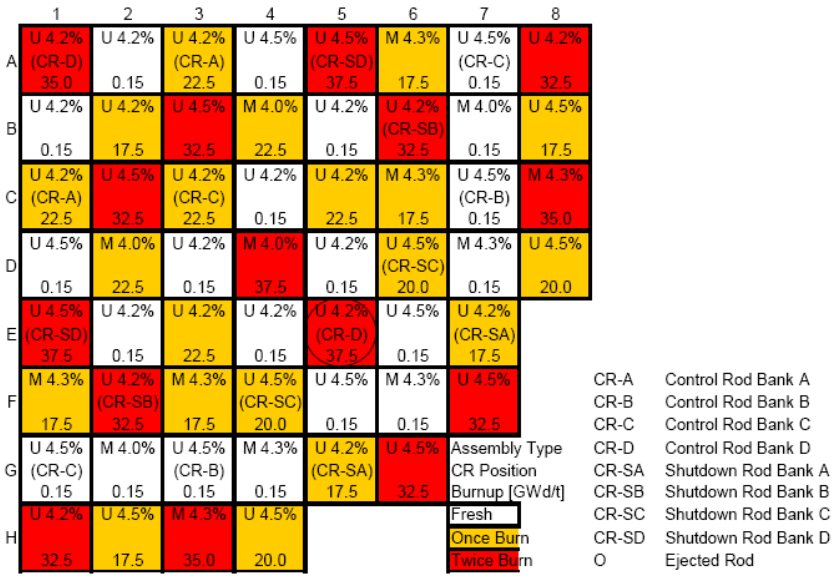


Figure 3–44 Core configuration used in the boron dilution benchmark (1/4th symmetry).

The initial state of the reactor is at shutdown conditions with a boron concentration of 2000 ppm and all control and shutdown rods inserted. It is assumed that a slug of deborated water has been accumulated in one of the cold legs of the reactor during outage. The slug has not been recognized and the normal start-up procedure has been initiated. At a system pressure of 3.0 MPa the main coolant pump is switched on in the loop with the slug. The coolant pump reaches its full flow rate within 14 s. The slug is transported into the reactor pressure vessel (RPV). Here it mixes with the highly borated coolant. Due to this mixing, a heterogeneous boron concentration is observed at the core inlet. During the transient the core inlet flow rate and the core inlet coolant temperature remain constant. The operational conditions for the problem are listed in Table 3-VI. The boron slug for the core inlet in the first 10 seconds is showed in Figure 3–45.

Table 3-VI Operational conditions for the boron dilution benchmark

Operational condition	Value
Core power	1 W
Mass flow rate (core + reflector)	6106.32 kg/s
Mass flow rate per assembly	23.76 kg/s
Core outlet pressure	3.0 MPa
Coolant inlet temperature	200.0 °C
Control rod insertion	365.755 cm (fully inserted)
Boron concentration	2000 ppm
Neutronic boundary conditions (radial)	Zero flux
Neutronic boundary conditions (axial)	Zero flux

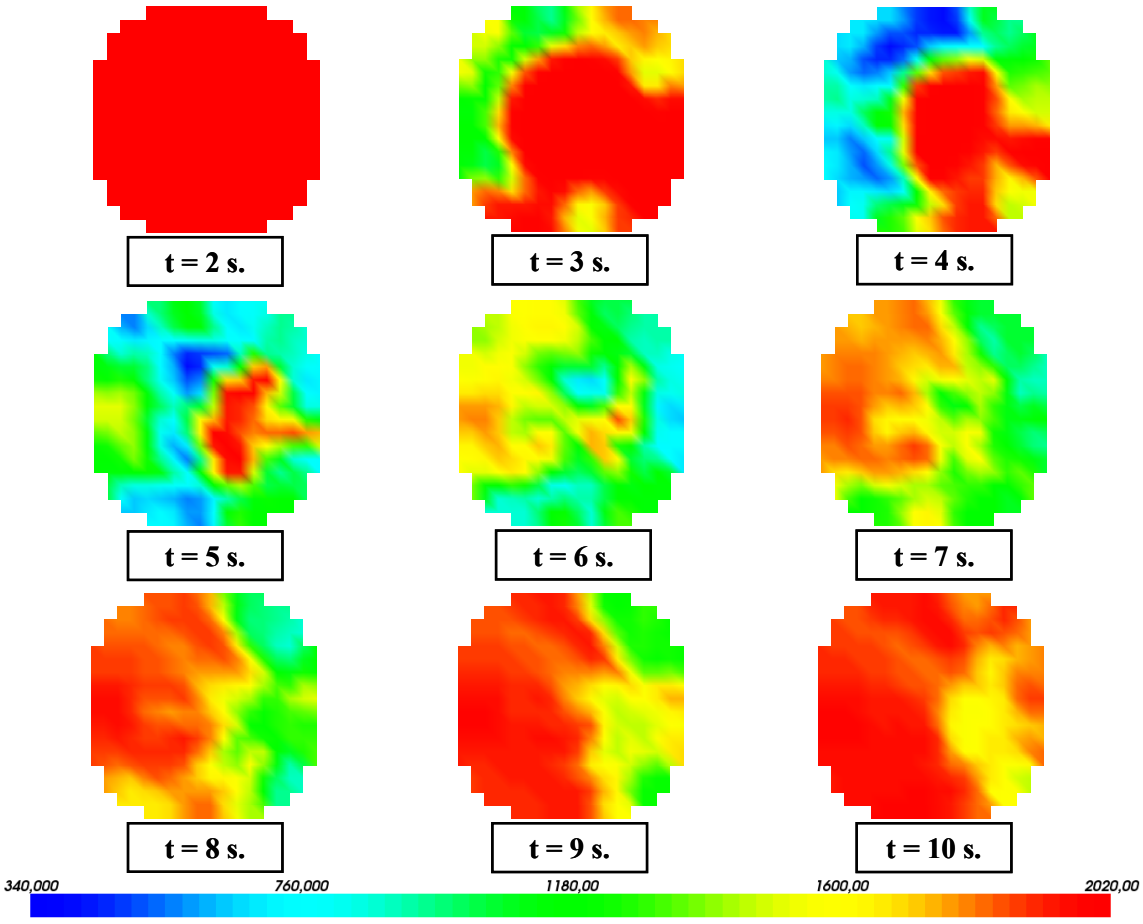


Figure 3–45 Boron slug at the inlet of the core for the first 10 seconds of the transient calculation.

The totally symmetric initial steady state in a pinwise configuration is shown in Figure 3–46.

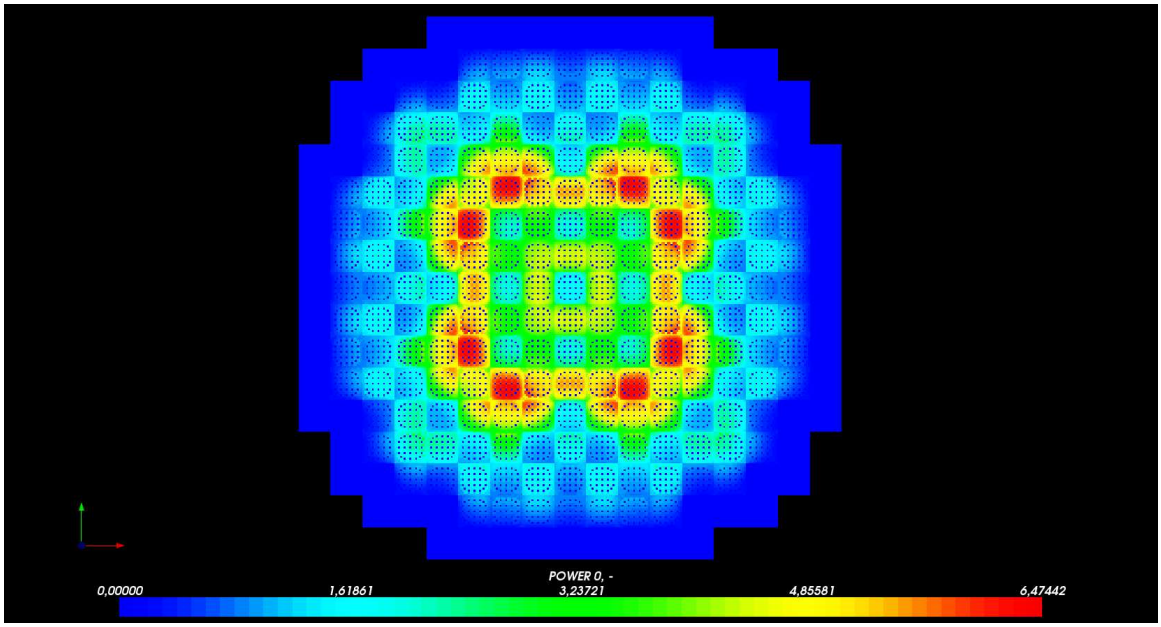


Figure 3–46 Steady state of the boron dilution benchmark in a full core pinwise configuration.

The pin power reconstruction method allows a fast whole core prediction of pin power distribution; however, sometimes could be interesting to focus on smaller regions, e.g., in the quadrant with the more impact due to the boron dilution (where the pump with the deborated slug is located), or where the control rod has to be withdraw. The non-conform geometry allows such kind of refinements that can be very useful in cases in which, based in the more detailed pin power distribution, a faster best estimate evaluation of local safety parameters is desired. This better estimation may be possible, for instance, by means of a subchannel code with non-conform geometry, or doing a parallel calculation of the refined region as in TRAC-PF1/NEM/COBRA-TF [Solis2002], [Ziabletsev2004] and [Solis2004].

It is also important to note that, although the pin power reconstruction method is very fast and does not imply a very large increase in calculation time, the 3D thermal-hydraulic subchannel code, using the power distribution for the estimation of local safety parameters, has to solve the whole problem (pin-by-pin) resulting in a very time consuming process.

Finally, some problems can arise while storing pin power data and thermal hydraulic feedback in the whole domain for post processing tasks. Thus, in a transient in which the time step is small (or becomes small due to automatic time step algorithms like in DYN3D), storing and manipulating the data of every pin could be very computationally challenging. For example, the required space for the 3D pin power distribution for solving the deborated slug in the first 10 seconds with the automatic time step size option of DYN3D will require more than 70 Gb of free disc space for the creation of the MED file containing just the pin power data at every time step.

Thereby, taking advantage of the versatility of the new pin power reconstruction extension, a detailed zone around a region (bottom-left quadrant of the core) was defined for the boron dilution Benchmark (Figure 3–47).

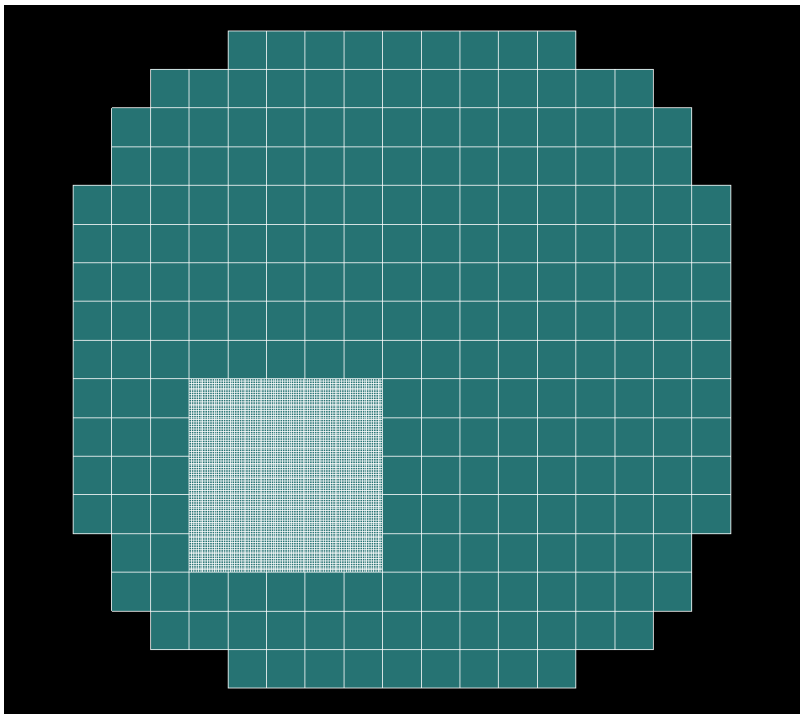


Figure 3–47 Mesh refinement for the boron dilution benchmark.

The steady state for this new configuration is presented in Figure 3–48.

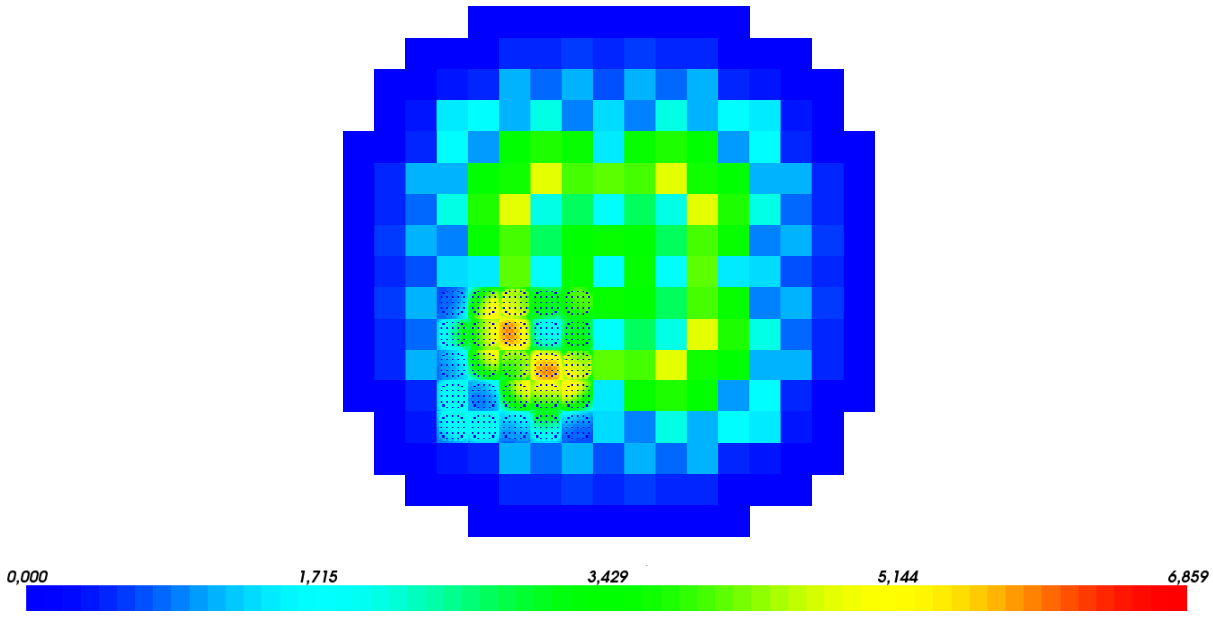


Figure 3–48 Steady state of the boron dilution benchmark in a zone with pinwise configuration (central layer).

The behaviour of the normalized radial pin power distribution for the central axial layer during transient is shown in Figure 3–49. It is interesting to see the effect of the deborated slug, in this case in the central layer. It takes approximately five seconds until the effect of the slug causes changes in the central layer. Referring to the deborated slug shown in Figure 3–45, it is at 4 seconds when the most deborated water (colour blue in the Figure 3–45 at 4 seconds) enters to the core through the region of interest. The thermal-hydraulic feedback of such effect is reflected two seconds after in the central layer, i.e. at 6 seconds the maximal power density is reached in the refined section.

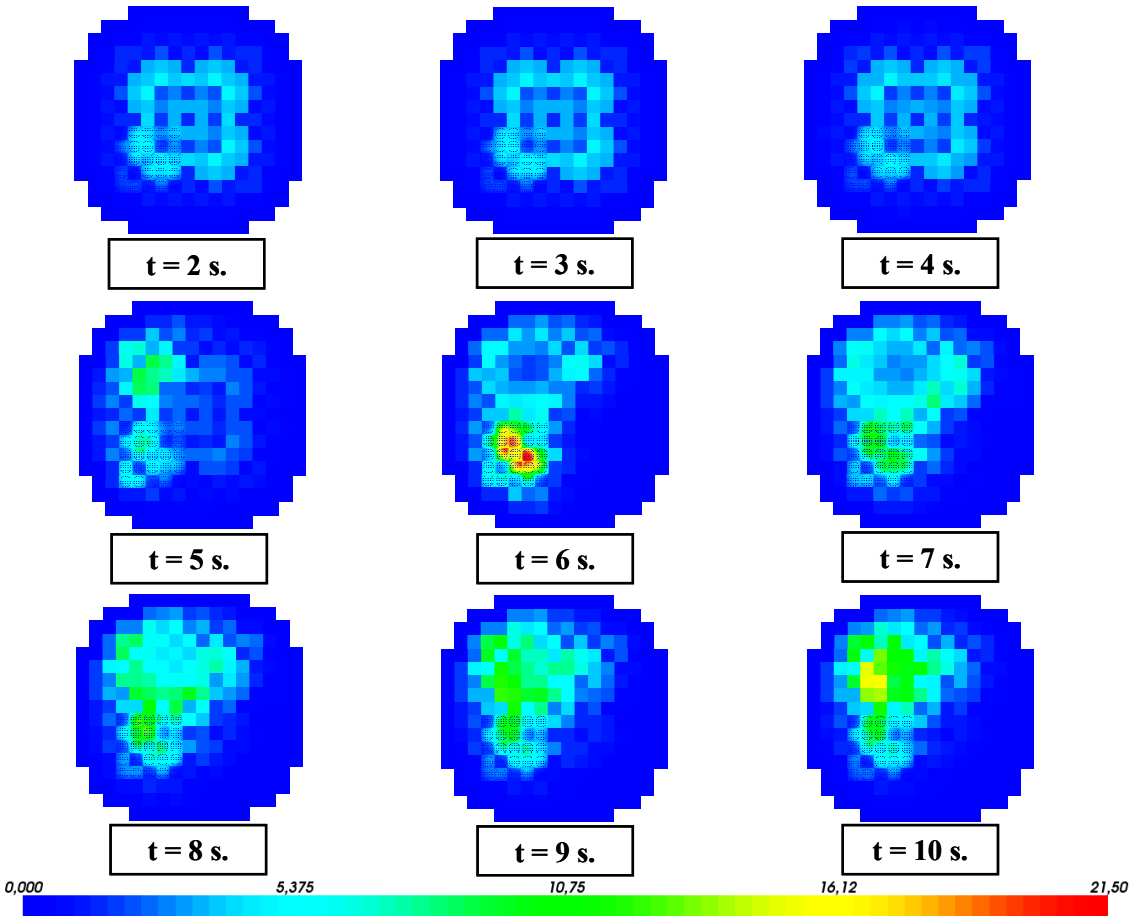


Figure 3–49 Transient of the boron dilution benchmark in a zone with pinwise configuration (central axial layer).

3.6.6.4 Applications to hexagonal geometries

As described before, the new pin power reconstruction method works not only with Cartesian geometries but also with hexagonal. In order to illustrate the capabilities of the new extension, a full core of a VVER-1000 reactor in hexagonal geometry was modelled. The cross sections including the form functions are based in the VVER-1000 reactor from the beginning of first cycle with only TVSA fuel assemblies in the core [Lötsch2009]. They were produced with HELIOS [Pralong2005] at HZDR based in a HELIOS model developed at the “State Scientific and Technical Centre for Nuclear and Radiation Safety” at Kiev (SSTC-NRS). At the end were transformed in NEMTAB format for being able to be used in DYN3D.

The non-conform capability and all the details already described for the Cartesian geometry were also implemented and are available in hexagonal geometry.

In Figure 3–50 a whole core in a pinwise resolution is presented for a lower layer of the core.

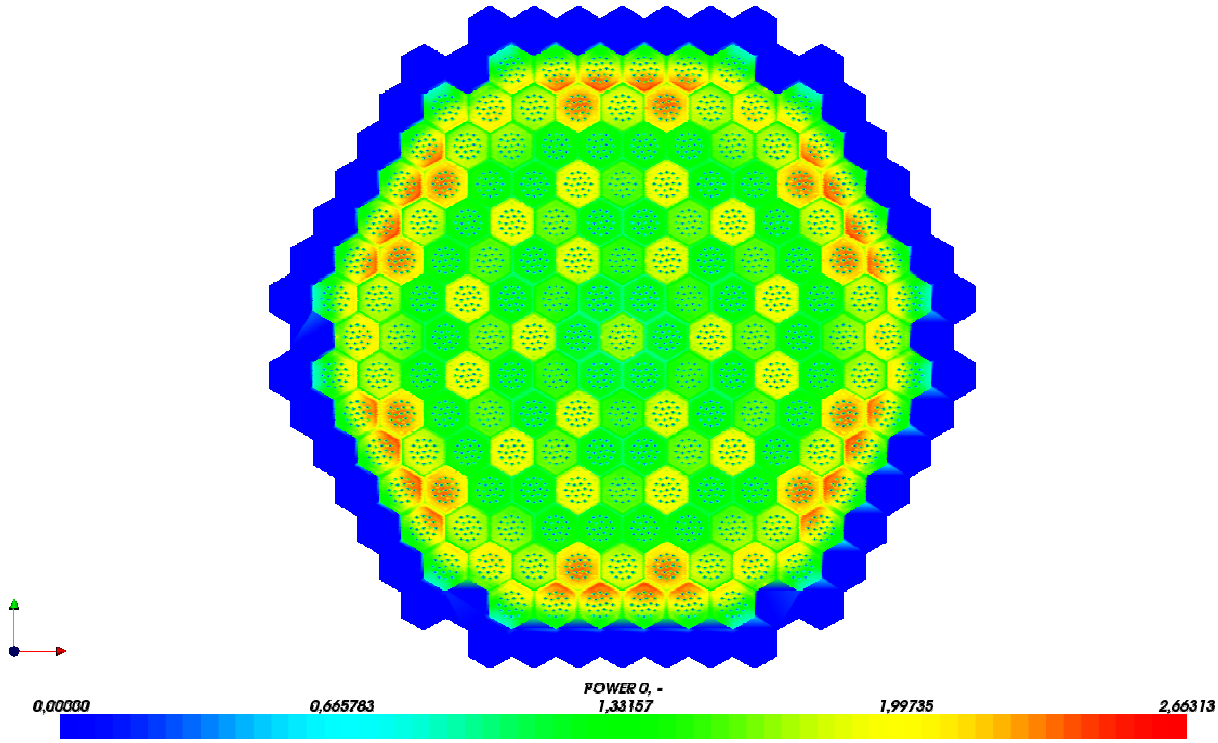


Figure 3–50 Steady state of a WWER core in a pinwise configuration with 211 nodal elements (163 FA + 48 reflector)

3.7 Potential application of the new DYN3D PPR capability

As previously discussed, a potential application for the extension of the pin power reconstruction (PPR) method is related with a more detailed prediction of local safety parameters. The capability to have pin power reconstruction in a zone or even in the whole geometry match very well with the new developments of the subchannel codes in the direction of non-conform geometry. The implementation of a non-conform solution in the code FLICA4 [Toumi2000] is under development as a part of NURISP project [Chauliac2008]. Furthermore, in the subchannel code SUBCHANFLOW [Imke2010] such implementation is foreseen. These conditions and previous experiences on multilevel coupling algorithms using pin power reconstruction, implemented for instance in TRAC-PF1/NEM/COBRA-TF [Ziabletsev2004] will make possible to implement coupling approaches for having a fast and more accurate prediction of local safety parameters not just in one assembly but in a bigger region of interest.

Additionally, both subchannel codes (FLICA4 and SUBCHANFLOW) are already integrated in the SALOME platform as it is the case of DYN3D making possible to take advantage of all the capabilities of the platform.

4 Development of an advanced coupling code based on DYN3D-SP3 and SUBCHANFLOW

4.1 Introduction

Several neutronics codes are able to solve the diffusion equation in multi-groups, considering feedback parameters coming from the thermal-hydraulic core model. As discussed before, these codes usually contain a two phase flow thermal-hydraulic model able to calculate and exchange assembly averaged feedback parameters. The thermal-hydraulic model can be a 1-D two phase flow model assigning parallel channels to each assembly (or group of assemblies) like FLOCAL in DYN3D [Grundmann2005]. For a more detailed analysis, the thermal-hydraulic calculation can be done via an internal coupling with a thermal-hydraulic subchannel code considering cross flow, which is important for PWR's, e.g. COBRA-TF/QUABOX-CUBBOX [Perin2010]. A system code with a three dimensional flow description like TRAC-PF1/NEM [IvanovK1999] can be used for a more simplified investigation. The coupling is done in all mentioned cases at a fuel assembly scale. Important efforts have been done in past years to refine the computational domain of coupled solutions from the fuel assembly scale to the subchannel/pin level. A new generation of high fidelity codes is under development that will allow a better prediction of local safety parameters. Several strategies have been investigated to extend the coupled methodologies for a more detailed and physical prediction of the pin power. With solvers based on the numerical solution of the transport equations (Boltzmann equation) direct predictions of the local power at pin level can be done avoiding the approximations of pin power reconstruction, for instance the transport approximation SP_3 used in DYN3D-SP3 [Beckert22008] and [Grundmann2009] and PARCS [Downar2006] or the discrete ordinates method used in TORT [Azmy1996]. In DYN3D-SP3 the coupling with the thermal-hydraulic model is performed in a so-called one-and-a-half way, i.e. the neutronic solver calculates a detailed pin power distribution but the thermal-hydraulic model FLOCAL predicts nodal averaged values (fuel assembly level) for the calculation of feedback parameters. In so doing, important local information is lost. Consequently, a more precise and realistic description of both the thermal-hydraulics and the neutronics domains at the same spatial scale is needed in order to consider the increasing heterogeneities of modern core loadings in LWR and future reactor designs.

A new coupled system, DYNSUB, has been developed by coupling DYN3D-SP3 and SUBCHANFLOW [Imke2010] at pin level and as the main topic of this dissertation.

In this chapter, a brief description of the two codes DYN3D-SP3 and SUBCHANFLOW followed by the coupling approach and the main features of the coupled system DYNSUB are presented. Several results for the OECD/NRC PWR MOX benchmark [Kozlowski2003] are also shown.

4.2 Description of DYN3D-SP3

Although the first version of DYN3D already discussed in the past chapter (two-group neutron diffusion equation) has been extensively validated [Grundmann1997], [Grund-

mann2003], [Grundmann2006], [Kliem2008] among others, it is also true that, modern core loading including MOX fuels, larger fuel cycle length, higher burn-up, part-length rods, water rods, etc. are very challenging for nodal diffusion codes. Thus, the trend of moving towards best estimate codes resulted in a new version of DYN3D able to deal with nodal diffusion methods in many energy groups (Multigroup), and in including a simplified transport approximation SP_3 [Grundmann2009], [Beckert22008] and [Beckert2008]. Thereby, DYN3D was extended to its new version DYN3D-SP3 giving the possibility to perform pin by pin calculations based on a transport approximation.

In DYN3D-SP3 the thermal-hydraulic model FLOCAL is still in charge of the thermal-hydraulic feedback. The neutronic solver calculates a detailed pin power distribution but the thermal-hydraulics model FLOCAL predicts nodal averaged values (fuel assembly level) for the calculation of feedback parameters.

In order to illustrate the main differences between the diffusion approximation described in the past chapter and the new SP_3 approximation, a brief description of the SP_3 method will be given in the next subsections.

4.2.1 Theoretical basis

When a reactor has been described in terms of its geometry, composition, and cross sections, the purpose of a neutron physics calculation is to determine the reaction rates and therefore the neutron density or the angle-dependant neutron flux $\Psi(\mathbf{r},\mathbf{\Omega},E,t)$, in which, contrary to the diffusion theory discussed in the past chapter, the dependence of the neutron flux in the angular direction of the neutrons is explicitly treated.

The goal of every reactor dynamics code is the solution of the time dependent Boltzmann transport equation which arises from a balance of neutrons in a specific volume dV of a nuclear reactor, as a function of the position $\vec{r}$, angular direction $\mathbf{\Omega}$, energy E, and time t. Hence, the total balance equation is [Beckert2008].

$$\frac{1}{V(E)}\frac{\partial}{\partial t}\Psi(\vec{r},\mathbf{\Omega},E,t)=Q(\vec{r},\mathbf{\Omega},E,t)-\Omega\cdot\nabla\Psi(\vec{r},\mathbf{\Omega},E,t)-\Sigma_t(\vec{r},E,t)\Psi(\vec{r},\mathbf{\Omega},E,t) \tag{4.1}$$

Where for a specific volume dV, the variation in time of the angular flux, at the left side of the equation (4.1) (divided by the mean neutron velocity $V(E)$), is equal to the production of neutrons $Q(\vec{r},\mathbf{\Omega},E,t)$ (source of neutrons), minus the leakage of neutrons $\Omega\cdot\nabla\Psi(\vec{r},\mathbf{\Omega},E,t)$, minus the missing neutrons due to absorptions and scatterings in the volume $\Sigma_t(\vec{r},E,t)\Psi(\vec{r},\mathbf{\Omega},E,t)$, being $\Sigma_t(\vec{r},E,t)$ the total macroscopic cross section.

As previously discussed, the production of neutrons can have three contributions: the neutron source due to fissions taking part in the reactor $Q_f(\vec{r},\mathbf{\Omega},E,t)$, the scattered neutrons in the volume coming from other energy levels and other angular directions $Q_S(\vec{r},\mathbf{\Omega},E,t)$ and finally an external neutron's source $Q_{EXT}(\vec{r},\mathbf{\Omega},E,t)$. For critical reactors the external source is usually set to zero assuming that all the neutrons come from fission. However, in the analysis

of subcritical systems this term is very important, because, thanks to the external source, criticality is reached.

$$Q(\vec{r},\mathbf{\Omega},E,t)=Q_f(\vec{r},E,t)+Q_S(\vec{r},\mathbf{\Omega},E,t)+Q_{EXT}(\vec{r},\mathbf{\Omega},E,t) \tag{4.2}$$

Taking into account that the angular neutron flux is described with 7 independent variables (3 in space, 2 in angle, 1 in energy and 1 in time), several assumptions are usually made in order to come to practical solutions. Deterministic methods are the most used in reactor calculations. These methods reformulate the Boltzmann equation by means of a numerical discretization of the variables and differential and integral operators in space, angular direction, energy and time.

As discussed in Chapter 2, two main classes of numerical methods can be distinguished to solve discretized integro-differential form of the Boltzmann equation: the method of "Spherical Harmonics" (P_N) and the "Method of Discrete Ordinates" (DOM).

The increase in complexity compared with the diffusion method described in Chapter 3 is the treatment of the angular dependency of the flux, which has been simplified in the diffusion approximation by making use of the Fick's law to relate the neutron net current with the gradient of the scalar neutron flux, as discussed in Chapter 3.

4.2.2 The Spherical Harmonic Method (P_N-Method) and the SP_N used in DYN3D-SP3

In the P_N-Method, the angular dependence of the neutron flux is expanded in spherical harmonic functions up to order N. The exact transport solution is recovered as N tends to infinity. In three dimensional geometries, the number of P_N equations grows like $(N+1)^2$; in one dimensional planar geometry, the number of P_N equations is only $(N+1)$. The P_N equations in one dimensional planar geometry are relatively simple and can be reformulated in second order form as $(N+1)/2$ equations similar to the diffusion ones but coupled through the angular moments [Brantley2000].

In multidimensional geometries the case is too complicated. Although the problem can be formulated as a set of second order equations, in this case the number of equations increases and the coupling involves not only the angular moments but also mixed spatial derivatives of these moments. This increase in complexity led to the proposal of the Simplified P_N approximation [Gelbard1960]. In the SP_N method, the second order derivatives in one dimensional planar geometry of the P_N equations are replaced or "generalized" by means of the three dimensional Laplacian operator leading to a multidimensional generalization of the planar geometry P_N equations that avoids the complexities of the full spherical harmonics approximation.

For one dimension, considering for instance an infinite plate, the angular neutron flux depends only on one spatial coordinate x, and one angular direction expressed through the angle θ. Thus, the angular flux can be described as a function of $\mu=\cos\theta$ and the Boltzmann equation (4.1) under the actual conditions has the form:

$$\left[\frac{1}{V(E)}\frac{\partial}{\partial t}+\mu\frac{\partial}{\partial x}+\Sigma(x,E,t)\right]\Psi(x,\mu,E,t)=Q(x,\mathbf{\Omega},E,t) \tag{4.3}$$

By means of the Multigroup approximation [Glasstone1970], and considering, for the sake of simplicity, just the stationary case, equation (4.3) can be rewritten as follows:

$$\left[\mu\frac{\partial}{\partial x}+\Sigma_g(x)\right]\Psi_g(x,\mu)=Q_{f,g}(x)+Q_{s,g}(x,\mu)+Q_{EXT,g}(x,\mu) \tag{4.4}$$

For each group g with $g=1,\ldots,G$, and with the group integrated flux defined as:

$$\Psi_g(x,\mu)=\int_g\Psi(x,\mu,E)\,dE \tag{4.5}$$

Taking into account that, in one dimension the Spherical Harmonics are the Legendre Polynomials, the scattering term can be expanded:

$$Q_{S,g}(x,\mu)=\sum_{l=0}^{\infty}(2l+1)P_l(\mu)\sum_{g'=1}^{G}\Sigma_{Sl,gg'}(x)\Phi_{l,g'}(x) \tag{4.6}$$

where $\Sigma_{Sl,gg'}(x)$ is the l-moment of the scattering cross section from group g' to group g , and the Legendre moments for every group are defined as:

$$\Phi_{l,g}(x)=\frac{1}{2}\int_{-1}^{1}P_l(\mu)\Psi_g(x,\mu)\,d\mu \tag{4.7}$$

Analogous, considering an isotropic fission source:

$$Q_{f,g}(x)=\frac{1}{k_{eff}}\chi_g\sum_{l=0}^{\infty}(2l+1)P_l(\mu)\sum_{g'=1}^{G}\nu\Sigma_{fl}^{g}(x)\Phi_l(x) \tag{4.8}$$

with Σ_{fl}^{g} representing the fission moment.

The group angular flux can also be expanded in a Legendre series:

$$\Psi_g(x,\mu)=\sum_{l=0}^{\infty}(2l+1)P_l(\mu)\Phi_{l,g}(x) \tag{4.9}$$

Substituting (4.9) in (4.4), multiplying by $\frac{1}{2}P_m(\mu)$, integrating over the interval $-1\le\mu\le1$ and making use of the orthogonal properties of the Legendre Polynomials, the following set of equations can be obtained:

$$\frac{m}{2m+1}\frac{\partial}{\partial x}\Phi_{m-1,g}(x)+\frac{m+1}{2m+1}\frac{\partial}{\partial x}\Phi_{m+1,g}(x)+\Sigma_g(x) = \sum_{g'}^{G}\Sigma_{sm,gg'}(x)\Phi_{m,g'}(x)+S_{m,g}(x) \tag{4.10}$$

with

$$S_{m,g}(x)=\begin{cases}\frac{1}{k_{eff}}\chi_g^i\sum_{g'=1}^{G}\nu\Sigma_{f,g}^i\Phi_{0,g'}^i(x)+Q_{EXT,g}^i(x) & m=0\\ 0 & m>0\end{cases} \tag{4.11}$$

where an isotropic external source was considered (no dependency on μ).

The equations (4.10), considering $m=0,\ldots,N$ are the P_N equations for the group g and lead to a system of $N+1$ equations with $N+1$ unknowns (the $\Phi_{-1,g}(x,t)$ and $\Phi_{N+1,g}(x,t)$ terms are neglected).

The system of differential equations P_3 can be obtained considering m=0, 1, 2, 3.

$$\begin{aligned}
&\frac{d}{dx}\Phi_{1,g}^i(x)+\Sigma_g^i\Phi_{0,g}^i(x)=\sum_{g'=1}^{G}\Sigma_{s0,gg'}^i\Phi_{0,g'}^i(x)+S_{0,g}^i(x)\\
&\frac{1}{3}\frac{d}{dx}\Phi_{0,g}^i(x)+\frac{2}{3}\frac{d}{dx}\Phi_{2,g}^i(x)+\Sigma_g^i\Phi_{1,g}^i(x)=\sum_{g'=1}^{G}\Sigma_{s1,gg'}^i\Phi_{1,g'}^i(x)\\
&\frac{2}{5}\frac{d}{dx}\Phi_{1,g}^i(x)+\frac{3}{5}\frac{d}{dx}\Phi_{3,g}^i(x)+\Sigma_g^i\Phi_{2,g}^i(x)=\sum_{g'=1}^{G}\Sigma_{s2,gg'}^i\Phi_{2,g'}^i(x)\\
&\frac{3}{7}\frac{d}{dx}\Phi_{2,g}^i(x)+\Sigma_g^i\Phi_{3,g}^i(x)=\sum_{g'=1}^{G}\Sigma_{s3,gg'}^i\Phi_{3,g'}^i(x)
\end{aligned} \tag{4.12}$$

Considering that there is no anisotropic scattering between different groups for the high order Legendre moments [Brantley2000], i.e.

$$\Sigma_{s,m,gg'}=0 \qquad g'\neq g,\quad 1\le m\le 3 \tag{4.13}$$

and defining the removal cross section as:

$$\Sigma_{R,m,g}=\Sigma_g-\Sigma_{s,m,gg} \qquad 1\le m\le 3 \tag{4.14}$$

The system of equations (4.12) can be rewritten as:

$$\begin{aligned}
&\frac{d}{dx}\Phi^i_{1,g}(x)+\Sigma^i_{R,0,g}\Phi^i_{0,g}(x)=\sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma^i_{s0,gg'}\Phi^i_{0,g'}(x)+S^i_{0,g}(x)\\
&\frac{1}{3}\frac{d}{dx}\Phi^i_{0,g}(x)+\frac{2}{3}\frac{d}{dx}\Phi^i_{2,g}(x)+\Sigma^i_{R,1,g}\Phi^i_{1,g}(x)=0\\
&\frac{2}{5}\frac{d}{dx}\Phi^i_{1,g}(x)+\frac{3}{5}\frac{d}{dx}\Phi^i_{3,g}(x)+\Sigma^i_{R,2,g}\Phi^i_{2,g}(x)=0\\
&\frac{3}{7}\frac{d}{dx}\Phi^i_{2,g}(x)+\Sigma^i_{R,3,g}\Phi^i_{3,g}(x)=0
\end{aligned} \tag{4.15}$$

Solving the second and the fourth equation for $\Phi_{1,g}(x)$ and $\Phi_{3,g}(x)$, and substituting in the first and third in order to eliminate the odd moments (1 and 3), a system of two equations with two unknowns is obtained:

$$\begin{aligned}
&-\frac{1}{3\Sigma^i_{R,1,g}}\frac{d^2}{dx^2}\left[\Phi^i_{0,g}(x)+2\Phi^i_{2,g}(x)\right]+\Sigma^i_{R,0,g}\Phi^i_{0,g}(x)=\sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma^i_{s0,gg'}\Phi^i_{0,g'}(x)+S^i_{0,g}(x)\\
&-\frac{2}{15\Sigma^i_{R,1,g}}\frac{d^2}{dx^2}\left[\Phi^i_{0,g}(x)+2\Phi^i_{2,g}(x)\right]+\frac{9}{35\Sigma^i_{R,3,g}}\frac{d^2}{dx^2}\Phi^i_{2,g}(x)+\Sigma^i_{R,2,g}\Phi^i_{2,g}(x)=0
\end{aligned} \tag{4.16}$$

New definitions can be introduced in order to simplify the system:

$$D_{0,g}=\frac{1}{3\Sigma_{R,1,g}};$$

$$D_{2,g}=\frac{9}{35\Sigma_{R,3,g}};$$

$$\tilde{\Phi}_{0,g}=\Phi_{0,g}+2\Phi_{2,g};$$

$$\tilde{\Phi}_{2,g}=\Phi_{2,g}$$

Leading to:

$$-D_{0,g}\frac{d^2}{dx^2}\tilde{\Phi}^i_{0,g}(x)+\Sigma^i_{R,0,g}\tilde{\Phi}^i_{0,g}(x)-2\Sigma^i_{R,0,g}\tilde{\Phi}^i_{2,g}(x)=\sum_{\substack{g'=1\\g'\neq g}}^{G}\Sigma^i_{s0,gg'}\left[\tilde{\Phi}^i_{0,g'}(x)-2\tilde{\Phi}^i_{2,g'}(x)\right]+\tilde{S}^i_{0,g}(x) \tag{4.17}$$

and

$$-D_{2,g}\frac{d^2}{dx^2}\tilde{\Phi}^i_{2,g}(x)-\frac{2}{5}D_{0,g}\frac{d^2}{dx^2}\tilde{\Phi}^i_{0,g}(x)+\Sigma^i_{R,2,g}\Phi^i_{2,g}(x)=0 \tag{4.18}$$

With the new source term defined as:

$$\tilde{S}_{0,g}(x)=\frac{1}{k_{eff}}\chi_g^i\sum_{g'=1}^{G}\nu\Sigma_{f,g}^i\left[\tilde{\Phi}_{0,g'}^i(x)-2\tilde{\Phi}_{2,g'}^i(x)\right]+Q_{EXT,g}^i(x) \tag{4.19}$$

Additionally, equation (4.17) can be used to substitute the second order derivative of the zero flux moment in (4.18) leading to:

$$\begin{aligned}-D_{2,g}\frac{d^2}{dx^2}\tilde{\Phi}_{2,g}^i(x)+\left[\frac{4}{5}\Sigma_{R,0,g}^i+\Sigma_{R,2,g}^i\right]\Phi_{2,g}^i(x)-\frac{2}{5}\Sigma_{R,0,g}^i\Phi_{0,g}^i(x)=\\ -\frac{2}{5}\left[\sum_{\substack{g'=1\\ g'\neq g}}^{G}\Sigma_{s0,gg'}^i\left[\tilde{\Phi}_{0,g'}^i(x)-2\tilde{\Phi}_{2,g'}^i(x)\right]+\tilde{S}_{0,g}^i(x)\right]\end{aligned} \tag{4.20}$$

Equations (4.17) and (4.20) form the P_3 system of equations in one dimension.

If the second order derivatives in the one dimensional planar geometry of the P_3 equations are replaced by means of the three dimensional Laplacian operator, as previously discussed, the general expression of the SP_3 equations is obtained [Gelbard1960] and [Brantley2000].

$$\begin{aligned}&-D_{0,g}\Delta\tilde{\Phi}_{0,g}^i(x)+\Sigma_{R,0,g}^i\tilde{\Phi}_{0,g}^i(x)-2\Sigma_{R,0,g}^i\tilde{\Phi}_{2,g}^i(x)=\sum_{\substack{g'=1\\ g'\neq g}}^{G}\Sigma_{s0,gg'}^i\left[\tilde{\Phi}_{0,g'}^i(x)-2\tilde{\Phi}_{2,g'}^i(x)\right]+\tilde{S}_{0,g}^i(x)\\ &-D_{2,g}\Delta\tilde{\Phi}_{2,g}^i(x)+\left[\frac{4}{5}\Sigma_{R,0,g}^i+\Sigma_{R,2,g}^i\right]\Phi_{2,g}^i(x)-\frac{2}{5}\Sigma_{R,0,g}^i\Phi_{0,g}^i(x)=\\ &\qquad -\frac{2}{5}\left[\sum_{\substack{g'=1\\ g'\neq g}}^{G}\Sigma_{s0,gg'}^i\left[\tilde{\Phi}_{0,g'}^i(x)-2\tilde{\Phi}_{2,g'}^i(x)\right]+\tilde{S}_{0,g}^i(x)\right]\end{aligned} \tag{4.21}$$

The final form of the equations (4.21) for every flux moment is very similar to the diffusion equation, thus, the solution of the SP_3 approximation can be obtained with a diffusion solver introducing minor modifications. A similar procedure like the one described in the Chapter 3 (extended to several energy groups) with the NEM method can be implemented for the solution of the SP_3 equations.

4.3 SUBCHANFLOW

The analysis of two phase flow behaviour is of great interest in many fields of science and engineering. However, an accurate prediction of the behaviour is not easy due to the presence of internal interfaces constantly moving (for instance, interfaces of vapour and liquid phases). The conservation equations of mass, momentum and energy can define the instantaneous microscopic behaviour of the fluid under study and codes like DNS, previously discussed, can be used. However, for practical purposes the microscopic behaviour can be very difficult to determine and then simplifications must be introduced by means of integrations and averages

in order to be able to predict the macroscopic behaviour of the fluid so large flow systems can be analyzed.

There exist several possibilities for representing the macroscopic behaviour of the fluid, ranging from description of the fluid as a one phase pseudo-fluid (mixture) until having a full multiphase representation with more than one form to represent the same fluid and phase depending on the system behaviour. The more complex the two phase flow model is, the more conservation equations are required to represent it.

SUBCHANFLOW [Imke2010] is a subchannel code based on the COBRA family of computer programs [Wheeler1976] and [Basile1999] but with improved capabilities. In opposite to the COBRA family of subchannel codes, SUBCHANFLOW uses rigorously SI units internally in all modules. Coolant properties and state functions are implemented for water using the IAPWS-97 formulation (The International Association for the Properties of Water and Steam). In addition, property functions for liquid metals (sodium and lead) are available, too.

A detailed description of the code and the specific models used in it can be found in [GomezR2010], or [Berkhan2011], for the purpose of this thesis only a brief description of the main physical models implemented in SUBCHANFLOW is given.

4.3.1 Heat conduction model

SUBCHANFLOW calculates heat conduction in the fuel pellet and within the cladding material by means of a fully implicit finite difference method.

The heat balance equation solved is given by:

$$\rho(\vec{r},T)C_P(\vec{r},T)\frac{\delta}{\delta t}T(\vec{r},t)=\nabla\cdot k(\vec{r},T)\nabla T(\vec{r},t)+q'''(\vec{r},t) \tag{4.22}$$

where

$\vec{r}$	is the position vector,
$T(\vec{r},t)$	is the temperature as a function of time and space,
$k(\vec{r},T)$	is the thermal conductivity of the considered material as a function of position and temperature,
$q'''(\vec{r},t)$	is the heat generation rate per unit volume (volumetric heat source),
$C_P(\vec{r},T)$	is the specific heat of the considered material as a function of position and temperature,
$\rho(\vec{r},T)$	is the density of the material function of position and temperature.

Several simplifications are usually done in order to solve this equation. If it is assumed, for instance, that the heat is transferred mainly in the radial direction (axial heat transfer neglected) and that the temperature distribution is isotropic in the θ direction, ($\delta T / \delta\theta = 0$), the heat balance equation in cylindrical coordinates can be rewritten as:

$$\rho(r,T)C_P(r,T)\frac{\delta}{\delta t}T(r,t)=\frac{1}{r}\frac{\delta}{\delta r}rk(r,T)\frac{\delta}{\delta r}T(r,t)+q'''(r,t) \tag{4.23}$$

Temperature dependent material properties for different fuels (UO_2 and UO_2PuO_2) and fuel-clad (Zircaloy and stainless steel) are implemented in SUBCHANFLOW. Fixed values can also be chosen if needed.

4.3.2 Heat transfer

The heat flux transferred from the external clad wall to the fluid q'' is governed by:

$$q''=h_{coef}\left(T_w-T_b\right) \tag{4.24}$$

where T_w is the wall temperature and T_b is the temperature of the fluid.

The heat transfer coefficient h_{coef} is determined by using empirical correlations depending on the heat transfer mode. The heat transfer mode along the boiling curve is described by a combination of different heat transfer models. Thus for instance, for the single-phase liquid force convection, the Dittus Boelter or the Gnielinski correlation can be used [Imke2010]. The sub-cooled nucleate boiling region relays on the Thom model [GomezR2010] or Schrock-Grossman correlation [Berkhan2011], etc. A full description of the models used in SUB-CHANFLOW can be found in the references mentioned in this paragraph.

4.3.3 Fluid Dynamics model

The two phase flow model used in SUBCHANFLOW considers the fluid as a mixture. Based on three mixture conservation equations, a four equations formulation is established: one equation for conservation of mass, one for conservation of energy, and two for conservation of momentum (one in axial direction and the other for lateral direction in order to take into account the cross flow).

If it is assumed that the velocities and the pressures of the two phases are the same (denoted as $\vec{v}$ and p respectively) and that there exists thermodynamic equilibrium, the three conservation equations can be written as follows:

Mass conservation:

$$\frac{\delta\rho}{\delta t}+\nabla\cdot\left(\rho\vec{v}\right)=0 \tag{4.25}$$

Energy conservation:

$$\frac{\partial(\rho h)}{\partial t}+\nabla\cdot\left(\rho h\vec{v}\right)=-\nabla\cdot\left(q''+q''^{T}\right)+\frac{\partial p}{\partial t}+q'''-\left(q'''_{wv}+q'''_{wl}\right) \tag{4.26}$$

Momentum conservation:

$$\frac{\partial(\rho\vec{v})}{\partial t}+\nabla\cdot(\rho\vec{v}\vec{v})=-\nabla p+\rho g+\nabla\cdot\left(\underline{\underline{\tau}}+\underline{\underline{\tau}}^{T}\right)+\underline{\underline{\tau}}''' \tag{4.27}$$

where ρ and h are the density and enthalpy of the mixture respectively, q'' and q''^{T} are the molecular and turbulent heat fluxes, and q''' is the energy generated in the fluid.

Finally $\underline{\underline{\tau}}$ and $\underline{\underline{\tau}}^{T}$ are the molecular and turbulent stress tensors and $\underline{\underline{\tau}}'''$ is the interfacial wall drag tensor.

4.3.3.1 Iterative procedure

A brief description of the iterative procedure implemented in SUBCHANFLOW for the solution of the fluid equations is as follows:

1. The mixture enthalpy h is obtained by means of the energy conservation equation (4.26).

2. With the enthalpy h, the quality x can be obtained in terms of the liquid and evaporation enthalpy:

$$x=\frac{h-h_{liq}}{h_{evap}} \tag{4.28}$$

3. The vapour volume fraction $\alpha_v(x,slip)$ is obtained as a function of the quality and the slip velocity model.

4. The effective mixture density ρ is calculated.

5. The momentum and mass conservation equations are solved for the pressure drop and momentum.

6. Repeat from 1 with actualized parameters.

A Successive Over-Relaxation (SOR) method is used for the solution of the energy balance matrices for the enthalpy, whereas a Gauss reduction procedure is used for the solution of the momentum balance matrices for the pressure drop and momentum. Details about the solution procedure can be found in [GomezR2010] and [Berkhan2011].

4.4 DYNSUB: A best estimate coupled code for the evaluation of local safety parameters

Based in the recent best estimate developments aiming at predicting local safety parameters in a more accurate manner, a new code system DYNSUB has been developed by coupling DYN3D-SP3 and SUBCHANFLOW at pin level.

With the development of DYN3D-SP3 important steps were done aiming at getting a more detailed flux description that can address the increasing heterogeneities in current and under development core designs. However, as previously discussed, the one dimensional thermal-hydraulic module FLOCAL included in DYN3D-SP3, is able to predict the feedback parameters just at a nodal level, i.e. averaged over one fuel assembly. A schematic view of the coupling performed in DYN3D-SP3 stand alone can be seen in Figure 4–1. The neutronic solution is a detailed one, while the thermal-hydraulic solution calculates nodal averaged values at fuel assembly level. The actualization of cross sections, at every iteration or time step, is done at pin level but with nodal averaged feedback parameters.

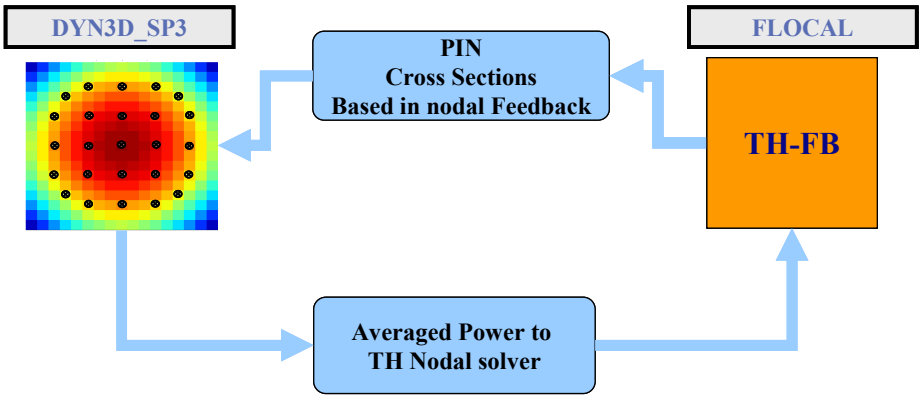


Figure 4–1 One and a Half Way Coupling in DYN3D_SP3 stand alone.

To increase the degree of detail in the thermal-hydraulic solution, a transition from a one dimensional to a multidimensional solver at pin level using subchannel codes is needed. Considering a thermal hydraulic subchannel code, a real two-way coupling with a detailed neutronic solver may be possible. Hence, the subchannel code SUBCHANFLOW has been selected for the coupling with DYN3D-SP3. By following this approach, the goal is to calculate and store the pin power in every axial node of each rod for a later calculation of the pin based cross sections with pin based thermal-hydraulic feedback parameters for every node avoiding the averaging process.

In Figure 4–2, the two-way coupling scheme implemented in the new system DYNSUB is illustrated for a fuel assembly. The pin power distribution is calculated with DYN3D-SP3 for every node in the geometry. The calculated pin power is transferred to the thermal-hydraulic solver, where it is allocated to each axial node of the fuel assembly and of its subchannels as considered in the SUBCHANFLOW fuel assembly representation.

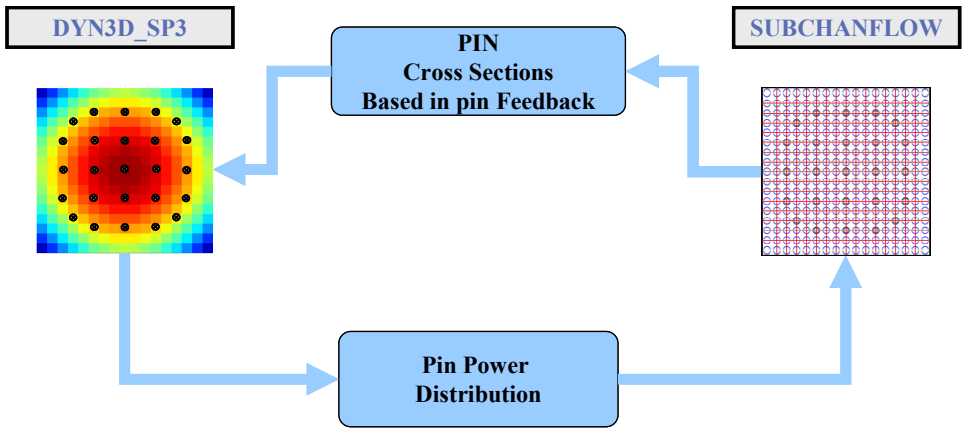


Figure 4–2 Two-Way Coupling in DYNSUB.

In an internal coupling, each neutron kinetics node is coupled directly to a thermal-hydraulic cell in the subchannel code. Although this method requires the exchange of a significant amount of information between the two codes (power and thermal-hydraulic feedback TH-FB for every single node), it also allows detailed and direct system calculations [Ivanov2007].

4.4.1 Coupling approach

For replacing FLOCAL with SUBCHANFLOW several changes were required. The main idea was to use SUBCHANFLOW as a slave compiled library linked to the DYN3D-SP3 source, as it is shown in Figure 4–3.

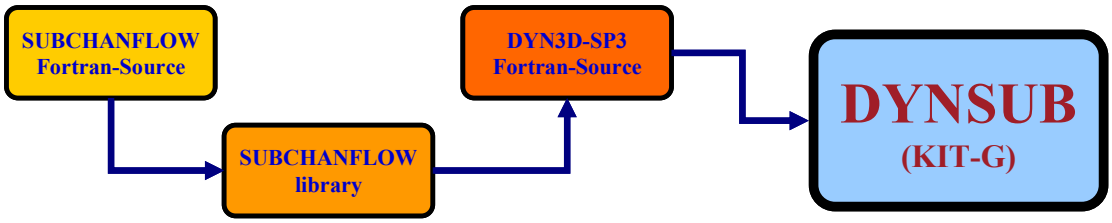


Figure 4–3 DYNSUB Coupling scheme.

The initial Windows version of the code used a Dynamic Link Library (dll) of SUBCHANFLOW linked to the FORTRAN source of DYN3D-SP3. The later implementation under LINUX was done by means of static libraries.

The coupling process resulted in several improvements in the standalone version of SUBCHANFLOW aiming to its use in a coupled system but without affecting the standalone functionality of the code.

SUBCHANFLOW was original conceived for standalone calculations. An internal coupling with a neutronics solver was not foreseen. In SUBCHANFLOW, a radial and axial averaged power profile had to be given by input in order to perform a calculation. Several transient cases could be analyzed by defining time dependent curves in the behaviour of certain parameters including power and for instance inlet temperature or flow.

Due to the desired coupling, three main SUBCHANFLOW updates were done in a short period (from version 1.5 to version 1.8) trying to address mainly the allocation of power coming from the neutronics code.

Figure 4–4 shows the evolution of DYNSUB and its influence in the upgrade of SUBCHANFLOW. Together with SUBCHANFLOW a Pre-processor devoted to generate automatically the input tables of SUBCHANFLOW for practical problems (hundreds of subchannels may be modelled in a pin by pin scale) was created, and will be described in a posterior subsection. Such Pre-processor is able to run as standalone code but it was also integrated in the coupled system. The neutronic solver of DYN3D-SP3 has not been modified, however new subroutines devoted to the communication with SUBCHANFLOW, the leading through the logic of the coupled system and the output of information were developed and will be later described. Additionally, a Graphical User Interface (GUI) for DYNSUB was created based in Qt and Python. The GUI allows not only some pre-processing tasks and executing the code in a user friendly way but also the most important application is the post processing options in which the powerful capabilities of SALOME can be used by means of the creation of MED files with the desired results. Details about the API will be given in Annex A.

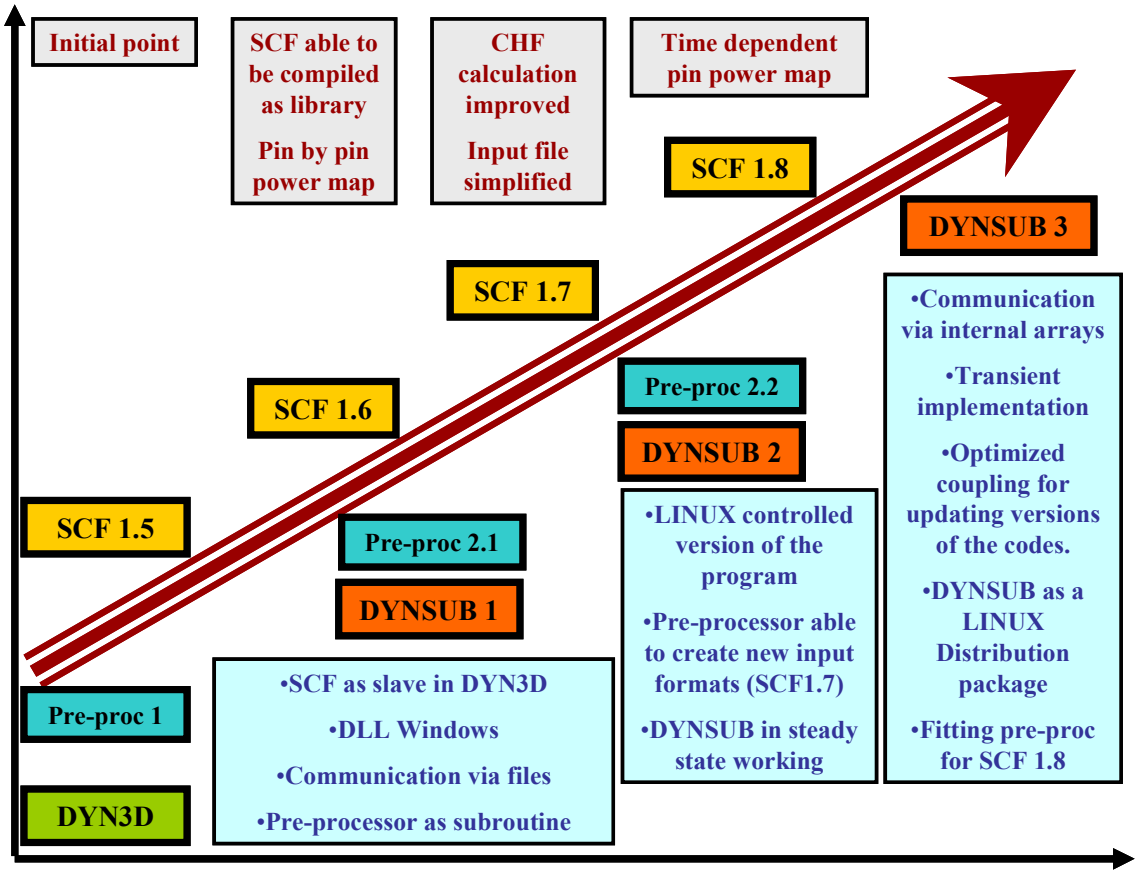


Figure 4–4 Evolution scheme of SUBCHANFLOW, Pre-processor and DYNSUB.

4.4.1.1 SUBCHANFLOW modifications and extensions

As discussed above, the goal of the coupled system is the replacement of the FLOCAL with SUBCHANFLOW. The use of SUBCHANFLOW as a library or slave program required several changes. A description of the three main modifications implemented in the stand alone version of SUBCHANFLOW is listed hereafter:

1) A slave program working as a library is not allowed to break a run but to send an error message once a problem is founded. Any "stop" in the source of SUBCHANFLOW would break the run. For this reason the replacement of "stops" by "returns" in the source of SUBCHANFLOW with the respective error message was performed.

2) The thermal power fraction in every node of SUBCHANFLOW was calculated based in an axial profile combined with a radial profile. Such profiles are usually the result of a radial and axial average of the power distribution coming from the neutronic code. This was not favourable for the implementation of the two-way coupling in which the main idea is to avoid such average processes. Starting with SUBCHANFLOW version 1.6, changes were implemented in order to have an additional option for direct allocation of the power distribution in every single node of the 3D problem representation allowing a different axial profile for every single rod in the configuration under study.

3) Furthermore, a time dependent pin power map was also implemented in the version 1.8. In previous versions, the power transients were performed in a global way, i.e., the shape of the pin power distribution remained constant during the time interval and the change in power was taken into account as a global change (the original power multiplied by the fractional change in power). Doing just like this, a local change in shape due to, for example, withdraw of a control rod was not fully represented in the time step. In SUBCHANFLOW version 1.8 (V.1.8) is possible to give a power map dependent on time.

   An example of this extension is presented in Figure 4–5. One axial layer with 4 nodes is considered. The transient implies a change in power of 10% due to a control rod movement (withdraw) in the southwest quadrant of the geometry considered. The left side describes the transient treatment in SUBCHANFLOW version 1.7 (V.1.7). The right side shows the effect of the new time dependent pin power map implemented in SUBCHANFLOW V.1.8. As it can be seen in the example, whereas in V.1.7 the effect of the temporal change in power is done globally (same power shape at the beginning and at the end of the time step leading to a 10% increase uniformly distributed in the four nodes), in V.1.8, the local influence in the power distribution as needed in a coupling with neutronics codes is correctly taken into account (a different power shape is considered at the end of the time step, the 10% increase is distributed in agreement with the local perturbation).

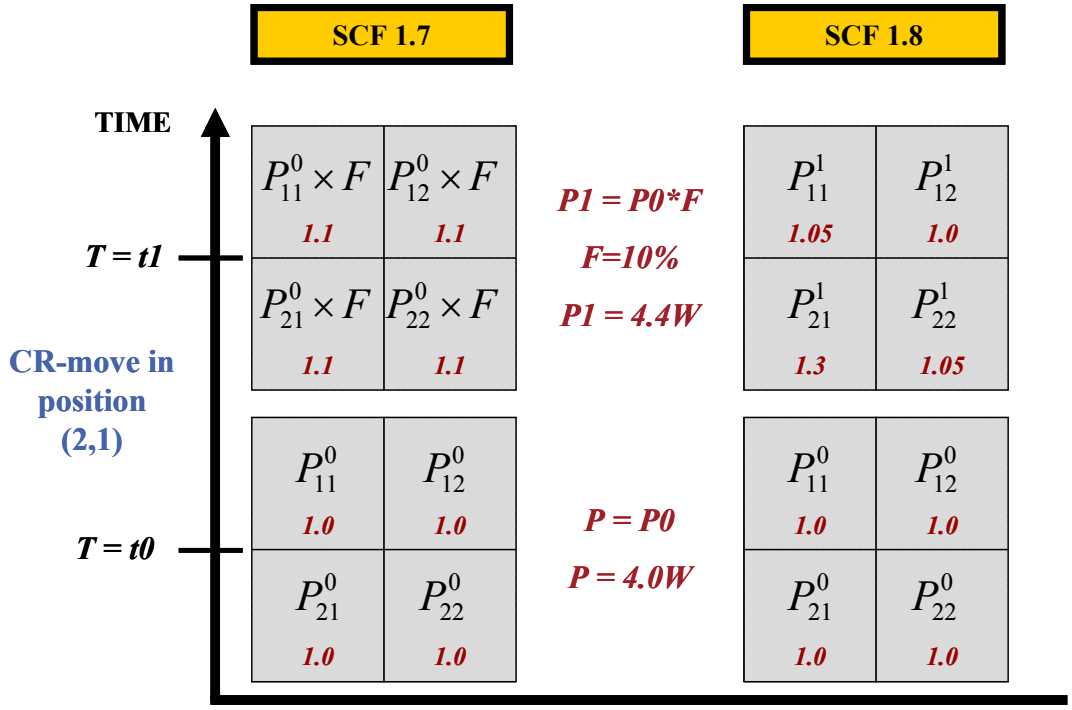


Figure 4–5 Transient treatment in SUBCHANFLOW 1.7 and 1.8.

4.4.1.2 Creation of a Pre-processor for SUBCHANFLOW

A Pre-processor for SUBCHANFLOW is a requisite when dealing with very big problems in which a great amount of channels and rods has to be created. SUBCHANFLOW had a very basic pre-processor able to create automatic input files for simple regular geometries (square bundles of several pins). However, the use of SUBCHANFLOW integrated in DYN3D-SP3 needed a more general and flexible tool in order to represent accurately a core configuration. Thus, even the simplest configuration to be solved with the coupled system (one PWR fuel assembly for instance) may need the creation of tables with hundreds of rows (one for each subchannel considered).

The SUBCHANFLOW pre-processor was extended and almost totally rewritten in order to be able to generate all the tables needed by SUBCHANFLOW with all the possible details. With the new SUBCHANFLOW pre-processor (Pre-processor version 2.2) is now possible, for instance, to define an irregular cluster of assemblies each one with different inner configurations (namely: type of rods with different thermal properties and control rod positions, with or without wetted boundary, etc.).

Figure 4–6 shows an example of geometry and input data for the Pre-processor working in its stand alone version. A PWR minicore with quarter symmetry for a total of 8 assemblies is represented. The number of subchannels in such geometry is 2592 (324 for each fuel assembly) and the number of rods is 2312 (289 for each fuel assembly). Considering 21 axial levels in the discretization, the number of nodes considered in this representation account for 54432 (number of subchannels times axial levels). Thus, tables up to 54432 rows have to be created in order to perform a SUBCHANFLOW run. These tables are automatically created with the Pre-processor. A detailed description of the input file needed and some examples are given in Annex B.

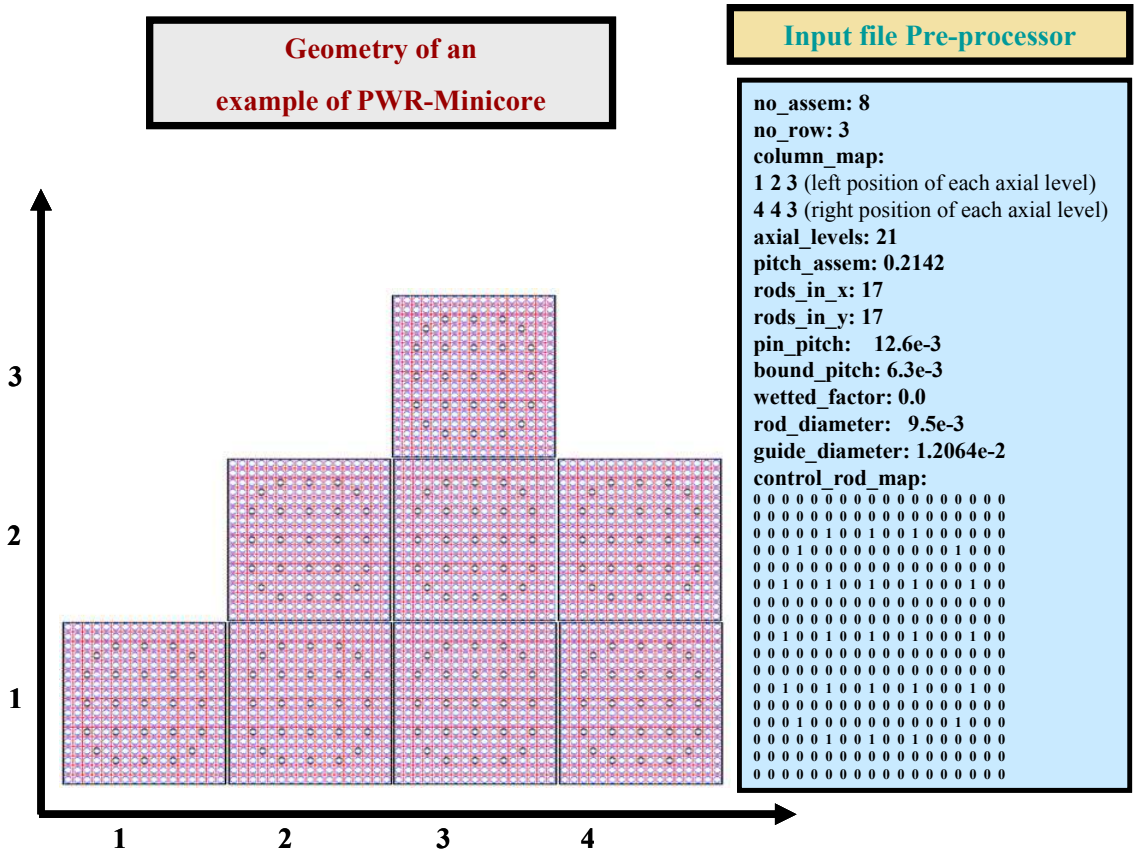


Figure 4–6 SUBCHANFLOW Pre-processor, an example of geometry and input file.

The Pre-processor working for the stand alone version of SUBCHANFLOW was included in the coupling as a subroutine. For this purpose, several modifications were implemented. First of all, the Pre-processor has to be able to identify the geometrical variables already described in the input file of DYN3D-SP3 and stored in internal arrays. Such variables describe the nodal configuration of the problem to be solved (the coordinates and size of the assemblies considered). Additional information must be provided via a simple input file (that must be called ***preproc.dat***) in order to define the internal configuration of the fuel assemblies. In Figure 4–7, the Pre-processor input file of the geometry described in Figure 4–6 for its use in the coupled system is shown. It is possible in principle to define different fuel assemblies with different intra-nodal configurations (useful, for instance, while analyzing BWR problems in which several assembly types are considered). Because SUBCHANFLOW does not have a boron transport model, a fixed boron concentration can also be given here by input. Further details and examples are given in Annex B.

Input file Pre-processor as a subroutine in DYNSUB

```
! Number of different assemblies
 1
! Rod diameter
 9.5e-3
! Guide diameter
 1.2064e-2
! Fuel assembly map
1  1  1  1
   1  1  1
      1
! Control rod map:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
!Fixed Boron concentration:
0.0
```

Figure 4–7 SUBCHANFLOW Pre-processor as a subroutine in DYNSUB.

4.4.1.3 DYN3D-SP3 modifications and extensions

The changes related with DYN3D-SP3 were mainly devoted to the replacement of the thermal-hydraulics module FLOCAL by SUBCHANFLOW. The general control of the iteration processes follows almost the same logic described in the description of models and methods of DYN3D [Grundmann2005]. However, several modifications were done while replacing FLOCAL by SUBCHANFLOW. The changes can be listed as follows:

1) New subroutine for the allocation of detailed 3D feedback parameters (at pin scale). In the original DYN3D-SP3 the memory allocation is done for the nodal i.e. assembly-wise feedback parameters (same value for every node inside the assembly). With SUBCHANFLOW instead of FLOCAL, the allocation must be done in a pin based scale (different values for every node in the assembly).

2) Modifications in the subroutines used for calculation of pin by pin cross sections by interpolation with the new pin based feedback parameters.

3) Creation of subroutines in order to manage the results. The replace of FLOCAL by SUBCHANFLOW was done trying to maintain the output structure of DYN3D-SP3; however, due to the more detailed output coming from SUBCHANFLOW, several modifications were implemented.

4) Modifications in the main subroutines performing the steady state and transient calculation in order to include the new thermal-hydraulic model. Initialization of new control variables and arrays, creation of SUBCHANFLOW input files by means of calling the Pre-processor subroutine, storage of old thermal-hydraulic values in case it is needed to iterate inside the time step, etc.

A detailed description about all the new and modified subroutines will be presented and discussed with more detail in the next subsection.

4.4.2 Spatial coupling

The spatial mapping between the neutronics and thermal-hydraulic domains at a pin/subchannel scale as well as the proper information exchange between these domains is one of the most challenging parts of the new coupled system. The mesh usually used in the subchannel codes has normally a different nodalization compared with the neutronics one. In the case of SUBCHANFLOW, the power coming from the neutronic code can be automatically allocated in the radial and axial rod power distribution. On the other hand, the thermal-hydraulic feedback parameters (TH-FB) are calculated in every subchannel making necessary the use of a weighting method for having just one feedback value for every pin.

As an example, in Figure 4–8 a bundle of four rods with 9 subchannels (SC) has been defined; furthermore the neutronic mesh (black full line) and the thermal-hydraulic mesh (orange dotted line) are shown. The radial mapping between the neutronic and thermal-hydraulic relates 4 subchannels to every neutronic node. Hence the TH feedback parameters for each of the neutronic nodes must be calculated based on the corresponding 4 subchannels before they are transferred to the neutronic part.

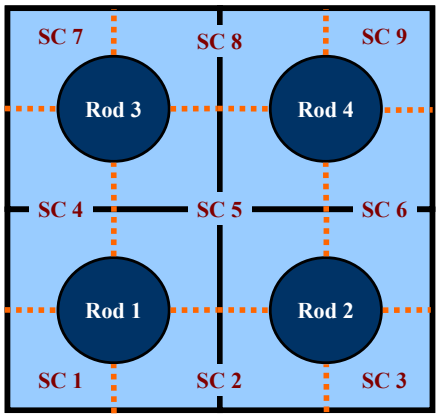


Figure 4–8 Bundle of 4 fuel rods with 9 subchannels.

The thermal-hydraulic feedback for every neutronic node "*i*" is given by means of the equation (4.29) :

$$\overline{FP}_i = \frac{\sum_{k=1}^{4} FP_k * W(SC_k)}{\sum_{k=1}^{4} W(SC_k)} \tag{4.29}$$

where:

FP_k : is a given thermal-hydraulic feedback parameter calculated in the subchannel SC_k (moderator density or temperature, void fraction, etc…)

$W(SC_k)$: Is the weighting factor that depends on the thermal-hydraulic feedback parameter to be calculated. Thus, if the moderator density must be calculated, the

weighting parameter will be the area of the subchannel. If the temperatures have to be calculated, the mass of the subchannels yields more information (comparing with just the volume) therefore they are used as a weighting parameter.

$\overline{FP_i}$: is the mean value of the Feedback Parameter transferred to the neutronic mesh.

For the axial mapping, a similar weighting method can be applied. Axial weighting functions based in the geometry have to be calculated and used. This option is not yet implemented in DYNSUB but it is foreseen and will be briefly discussed in the outlook. Thus, all the results presented in this thesis consider a one-to-one axial coupling (the same axial mesh must be considered in both codes).

Once the feedback parameters are ready to be transferred, the communication process has to be defined. As a first try, the interchange of data was developed via files. For the first version of DYNSUB (Windows version) subroutines for writing and reading were developed [GomezA2011]. Although the experience was useful in order to verify that the exchange of data and the weighting process were done correctly, the process is not efficient (reading and writing can take a significant amount of time while analyzing complex problems).

Later on, the interchange of feedback parameters was realized via memory arrays. Together with this implementation, the migration from Windows to LINUX version was a requirement in order to be able to use powerful LINUX clusters to address practical problems with important memory requirements.

4.4.3 Temporal coupling

The temporal coupling and time step selection plays a very important roll in the coupling. The most typical and straightforward approach is to use a one-to-one time step selection. One of the codes acts as the master and the other as slave code and both codes use the same time step (controlled by the master). However, this method is not always the best approach. The selection of the time step is done by the master program based in the convergence of its own local parameters. Thus in coupled codes in which the thermal-hydraulic code plays the roll of master, the time step selection is based on the convergence of the thermal-hydraulics parameters and global power but not the local fluxes [Watson2010]. A transient with a fast increase of neutron flux could be very challenging for the coupled system.

A variable time step control algorithm is implemented in DYN3D-SP3. Thereby in a fast increase of neutron flux, the neutronics (NK) iteration can select a small time step. On the other hand, the thermal-hydraulic (TH) processes are often slower and a large time step can be used. Therefore DYN3D-SP3 considers the use of separate time steps for NK and TH, with the only condition that the NK time step is less or equal the TH one [Grundmann2005]. A whole number of NK steps is included in the TH step.

Beyond the time step size, the point at which data is exchanged between the two codes is important and is normally classified in three types of couplings namely explicit, implicit and semi-implicit. The three of them have advantages and disadvantages. A comprehensive and short description of them can be found in [Watson2010]. Among these methods, the explicit coupling is the simplest one and probably the most widely used method and is the one implemented in DYN3D-SP3. To obtain converged results with this coupling scheme, very small time steps are required. With this method the master code converges first and then sends its

parameters to the slave code, the slave code in turn converges and sends data back to the master process. A new time step is selected and the process is repeated for each new time step.

In DYN3D-SP3, the neutronics solver plays the roll of master and the thermal-hydraulics model (FLOCAL) the salve. A simplified view of the explicit coupling in DYN3D-SP3 standalone is presented in Figure 4–9 (for an one-to-one time step coupling).

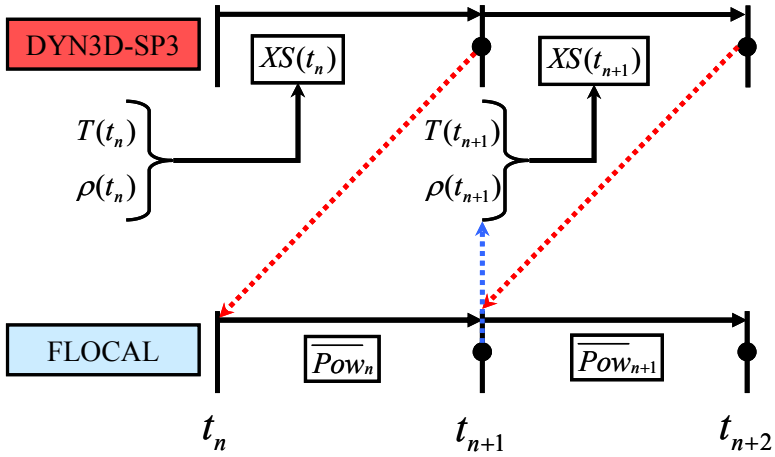


Figure 4–9 Explicit coupling of DYN3D-SP3 with its thermal-hydraulic model FLOCAL

The new power distribution is first calculated by means of the neutronics solver. In the case of the thermal-hydraulic model (FLOCAL), the power densities at begin and end of the thermal-hydraulic step are needed as an input, however the average value of the two densities is used in the numerical equations (equation (4.30)).

$$\overline{Pow_n} = \frac{Pow(t_n) + Pow(t_{n+1})}{2} \tag{4.30}$$

When several neutronic steps are inside the thermal-hydraulic step, the average value of the power densities at each neutronics step is estimated. The average value of power density used in FLOCAL results from the average over the different neutronic steps inside the thermal-hydraulic step.

The length of the neutronics time steps can be adjusted at the end of each neutronics time step. They depend on the changes of the neutron distribution during the previous step. The changes of the averaged spatial distribution given by the average exponent of the exponential transformation (see equation (3.32)), the exponent itself, the changes of the shape of the flux distribution and the number of neutronics iterations are used for the neutronics time step control. The neutronics time steps can be doubled or halved depending on the given criteria of the changes. At the end of the thermal-hydraulics time step the estimated neutronics time step is changed to meet the end of the thermal-hydraulics step [Grundmann2005].

In DYNSUB the time integration follows the same logic but with some variations. The one-to-one time coupling in DYNSUB is shown in Figure 4–10. DYN3D-SP3 solves the time dependent problem with fixed cross sections (initial values or coming from the previous step). The power at the end of the time step is transferred to SUBCHANFLOW for its transient calculation. In contradistinction of FLOCAL, SUBCHANFLOW has a time dependent power

distribution map (see Figure 4–5). An average power density is in this case not required. SUBCHANFLOW uses the two power maps (at the beginning and at the end of the time step), for the time step calculation. In this way local changes of power inside the neutronics interval can be taken into account. At the end of the thermal-hydraulic step, new feedback parameters are used for actualization of the cross sections of the next neutronics step.

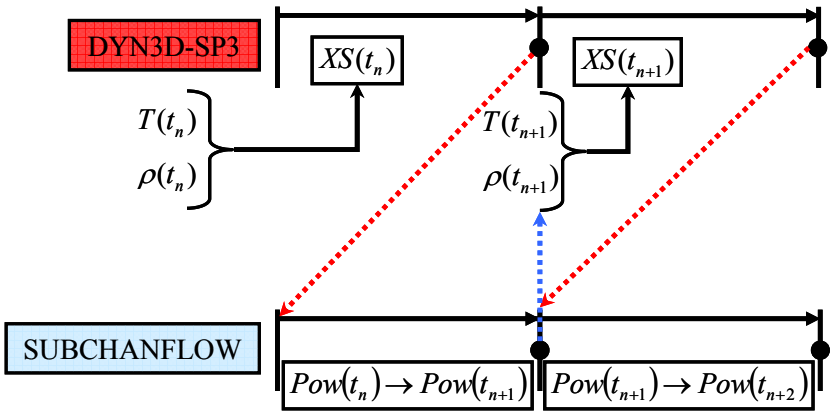


Figure 4–10 Explicit coupling of DYN3D-SP3 with SUBCHANFLOW

Additionally to the explicit coupling, a nested loop iteration or fixed point iteration (FPI) [Xu2005] is implemented in DYN3D-SP3 and consequently in DYNSUB. A FPI is not an implicit scheme but approximates it by adding a loop to the current marching scheme [Watson2010]. This method has been already investigated using TRACE and PARCS in [Gan2003] where it was shown that for a small penalty in accuracy the method allows larger time steps.

In DYN3D-SP3 the method is conceived for dealing with the problem of facing biggest jumps in power distribution that may lead to non converged results when the time step is not small enough. In Figure 4–11 the FPI coupling diagram of DYNSUB is depicted.

The process starts similar to the one of Figure 4–10; however at the end of the SUBCHANFLOW transient calculation, the final feedback parameters may be used to update the cross sections for a new neutronics calculation starting again the time step. If the convergence criterion, given by equation (4.31), is not satisfied, the iteration process is repeated.

$$Dev\_Pow^{t} = \frac{Pow_n^{t} - Pow_{n-1}^{t}}{Pow_n^{t}} \leq \varepsilon_{POW} \tag{4.31}$$

where the convergence criterion ε_{POW} is defined by the user.

Additionally, a number of iterations can be given by input in order to stop the iteration process and to go to the next step although no convergence is satisfied.

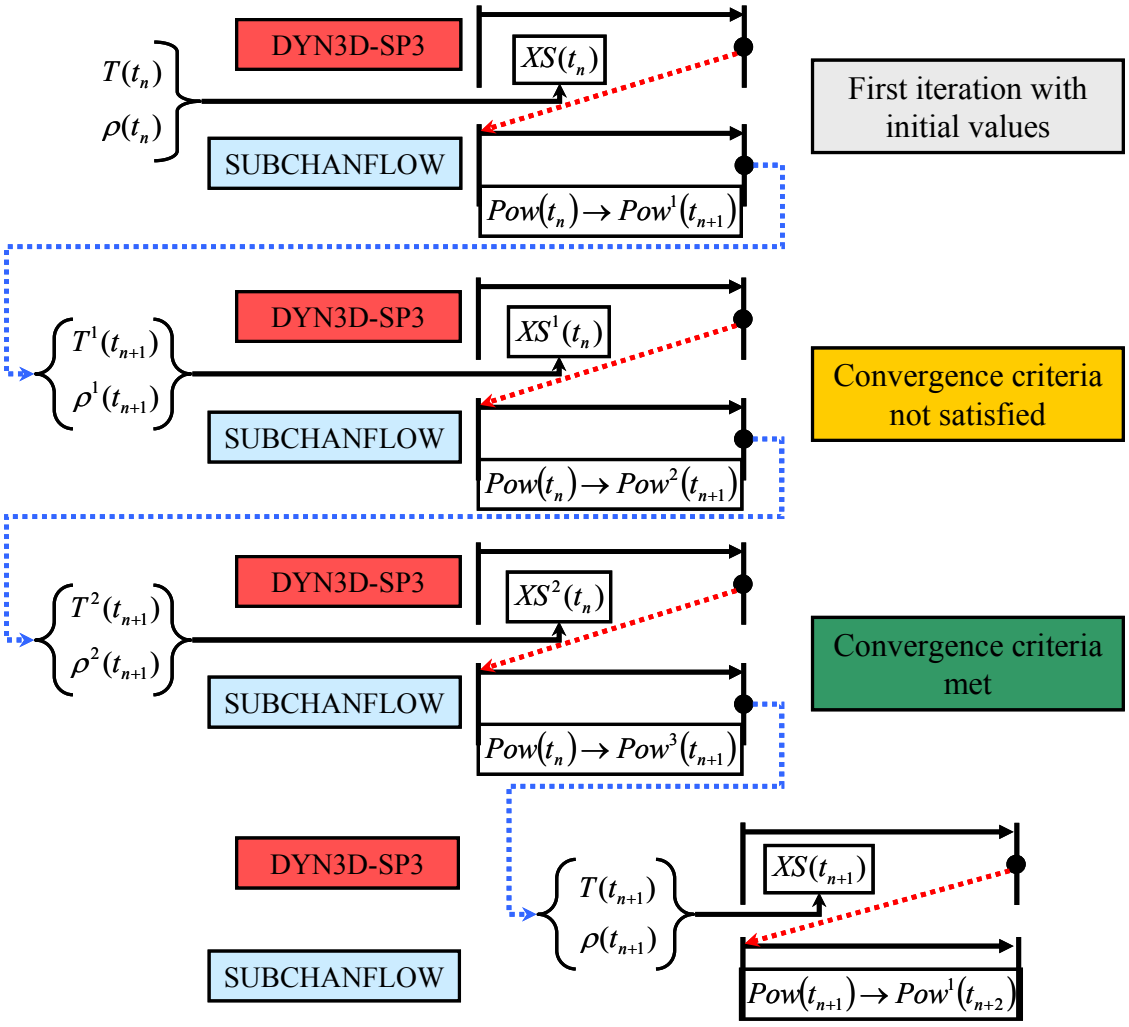


Figure 4–11 Coupling diagram with iterations inside a time step

Moreover, at the first steps the changes of the fuel temperature against the previous step are used for limiting the iteration. If the power level is low, for example at hot zero power, an increase of power during the time step does not influence the fuel temperatures. An iteration is not necessary in this case. More details will be given while describing the flow diagrams in the next subsection.

The coupling with iterations, i.e. nested loop, (Figure 4–11) could be very useful in some cases because, as previously mentioned, it allows the use of “bigger” time steps than in the case without iterations. Furthermore it results in an appropriate approximation to a reference solution. A FPI becomes unnecessary if, for instance, very small time steps in the critical interval of time are used. However a coupling strategy must be analyzed in order to take advantage of the method. Some comparisons about the two methods and a discussion about the advantages and disadvantages of each of them are presented in the results section and in the conclusions.

4.4.4 DYNSUB Flow diagrams

4.4.4.1 General flow diagram

The general control of DYNSUB is done based in the main program of DYN3D-SP3, however several modifications had to be introduced related with the replacement of FLOCAL with SUBCHANFLOW. Although the modifications in both codes, the input of DYNSUB was done in such a way in which no changes are needed in the stand alone inputs of DYN3D-SP3 and SUBCHANFLOW however they must be coherent to each other. As previously discussed, an additional input file with details about the internal assembly configuration is needed for letting the pre-processor know how to build the inner assembly geometry. The general flow diagram of the main program of DYNSUB is depicted in Figure 4–12.

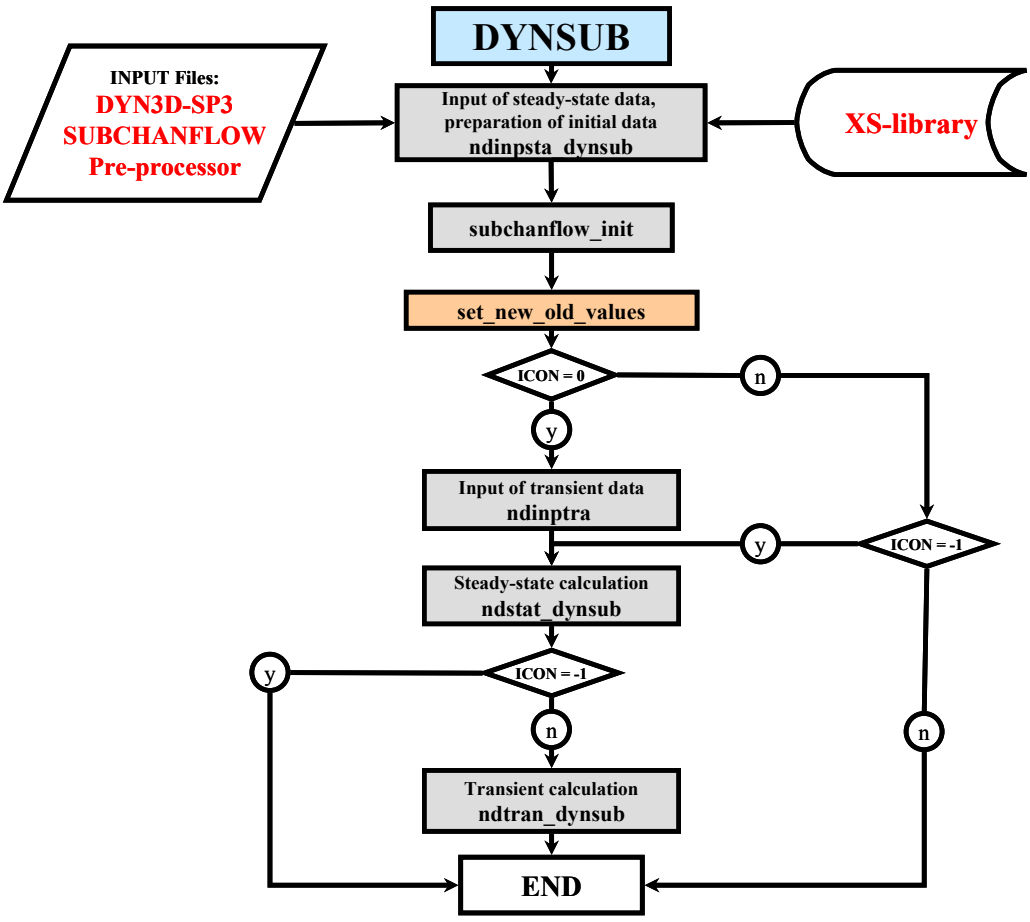


Figure 4–12 DYNSUB general flow diagram

The three input files in the left and the cross sections library in the right are the set of input files for the coupled code. Two options are available. The transient calculation (ICON = 0) and a steady state calculation (ICON = -1). The transient calculation is always preceded by a steady state calculation. An extended capability related with a transient calculation starting from a restart file is under development.

At the beginning of the calculation the initial values of the thermal-hydraulic calculation (namely initial flows, enthalpies, densities, etc…) are stored by means of the subroutine: ***set_new_old_values.f*** for repeating the steady state calculation of SUBCHANFLOW if needed (as a part of the iteration process). A brief description and summary of all the new and modified subroutines implemented in the coupling will be given in a later subsection.

4.4.4.2 Steady state flow diagram

The flow diagram inside the steady state calculation (***ndstat_dynsub.f***) is shown in Figure 4–13. An eigenvalue iteration is performed in the solution of the nodal equation system. The iteration starts with fixed thermal-hydraulic parameters (given by input) in all the nodes considered and initial guesses for the fluxes. A first estimation of power with these fixed thermal-hydraulic conditions is done. The power is transferred to SUBCHANFLOW (***dyn2sub.f***) for a first thermal-hydraulic calculation. After that, the thermal-hydraulic parameters are transferred back (***sub2dyn.f***) for an updating of cross sections and the start of the Neutronics–Thermal-hydraulics iteration process (NK-TH).

The convergence criteria for ending the NK-TH iteration process are based in the change in k_{eff} which is determined in the NK calculation (***nditer_r.f***) and the maximal deviations (over all the nodes *i* of the system) for the fuel Doppler temperatures and coolant densities (***subchan_dev.f***) in the TH calculation. The calculation of deviations is done based in equations (4.32) for the k_{eff}, (4.33) for the fuel Doppler temperature and (4.34) for the moderator density.

$$Dev\_k_{eff} = \frac{k_{eff}^{n} - k_{eff}^{n-1}}{k_{eff}^{n}} \leq \varepsilon_k \tag{4.32}$$

$$Dev\_TF_{DOPPLER} = \max_i \left(\frac{TF_{DOPPLER,i}^{n} - TF_{DOPPLER,i}^{n-1}}{TF_{DOPPLER,i}^{n}} \right) \leq \varepsilon_{TF} \tag{4.33}$$

$$Dev\_TF_{DOPPLER} = \max_i \left(\frac{TF_{DOPPLER,i}^{n} - TF_{DOPPLER,i}^{n-1}}{TF_{DOPPLER,i}^{n}} \right) \leq \varepsilon_{TF} \tag{4.34}$$

The convergence criteria ε_k, ε_{TF} and ε_{DM} are defined by the user.

The ***set_new_old_values.f*** subroutine has two functions. During the iteration process (green box in Figure 4–13), the initial thermal-hydraulic values (stored in the initialization process showed in Figure 4–12) are used together with the new power distribution for a new SUBCHANFLOW calculation. Once the NK-TH iteration process reaches convergence, the ***set_new_old_values.f*** (orange box in Figure 4–13) will replace the stored values with the new data (converged data) for starting a possible transient calculation. Thus the following relations are valid:

$$TH\_conditions_{OLD} = TH\_conditions_{ACTUAL} : \text{Orange box.}$$

$$TH\_conditions_{ACTUAL} = TH\_conditions_{OLD} : \text{Green box.}$$

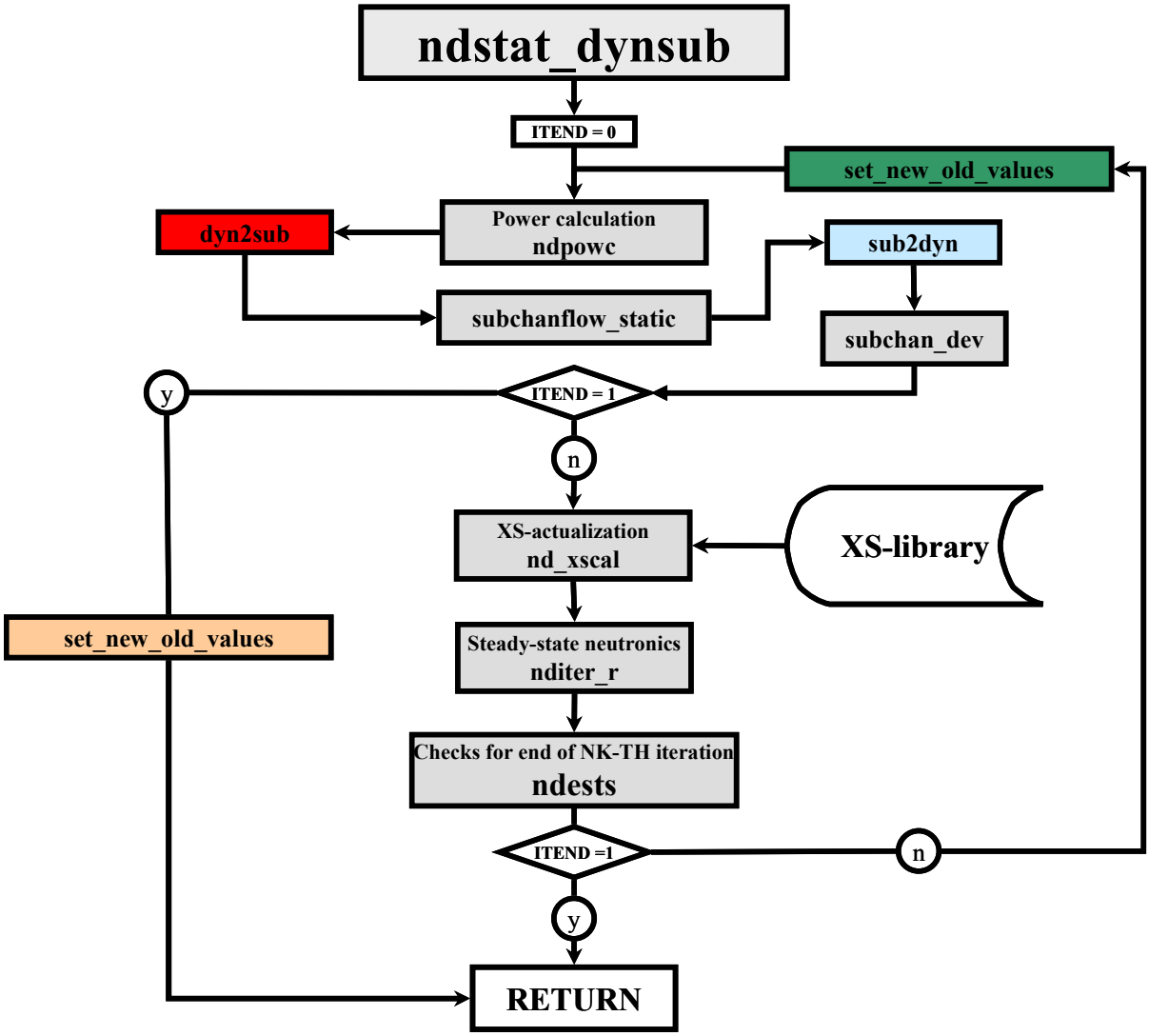


Figure 4–13 Flow diagram for the steady state calculation of DYNSUB

4.4.4.3 Transient flow diagrams

In the transient iteration, the calculation is similar to the steady state but instead of an eigenvalue calculation, the solution of the system is done with a given source obtained from the integration of the precursor equations.

Figure 4–14 shows the general flow diagram of the transient calculation.

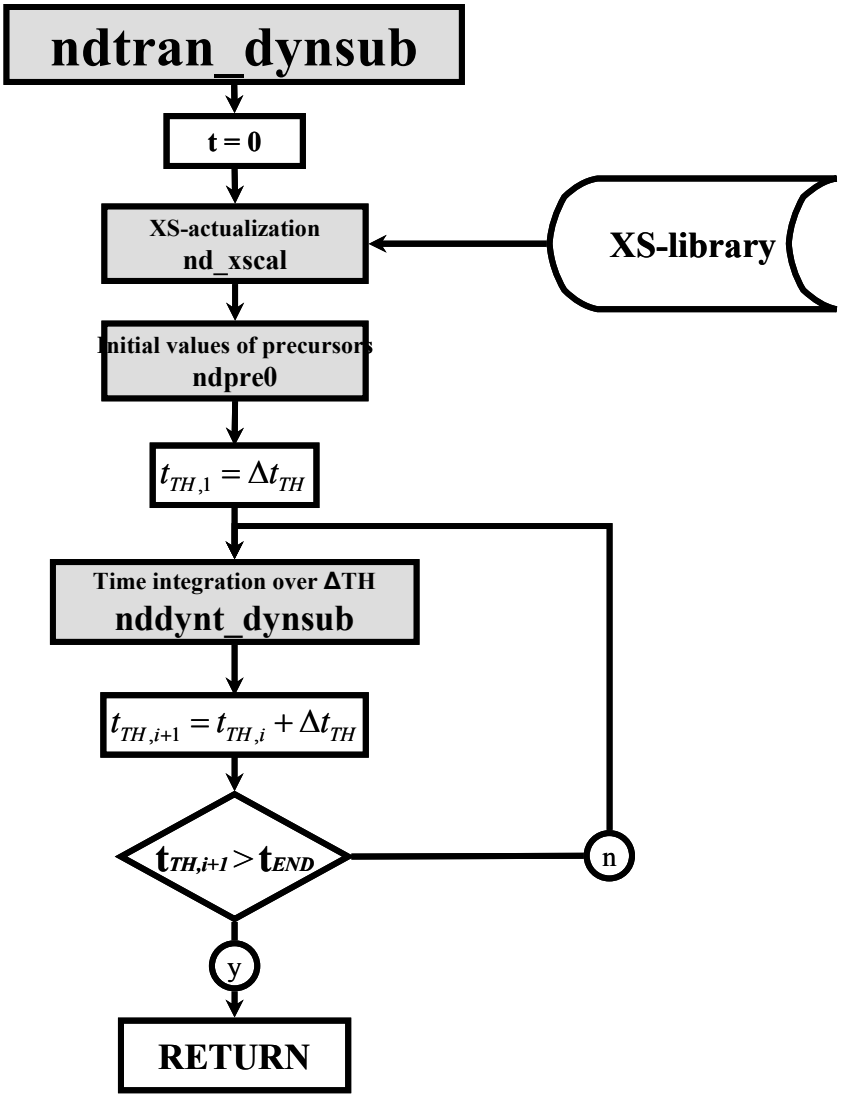


Figure 4–14 Flow diagram for the transient calculation of DYNSUB

The integration over the TH step is done in subroutine ***nddynt_dynsub.f*** and is depicted in Figure 4–15.

The iteration logic is similar to the steady state calculation. The subroutine ***set_new_old_values_t.f*** stores values either to repeat an internal iteration step (green box) in the FPI scheme or to save the new values for the next time step (orange box) in the normal explicit scheme. Additionally to the thermal-hydraulic data, the initial pin power distribution is also stored. The subroutine ***subchanflow_transient.f*** solves in every call its own transient calculation moving from the initial power map (previously stored) to the new calculated power map, as showed in Figure 4–5. In ***ndtrit.f*** the neutronic integration is done.

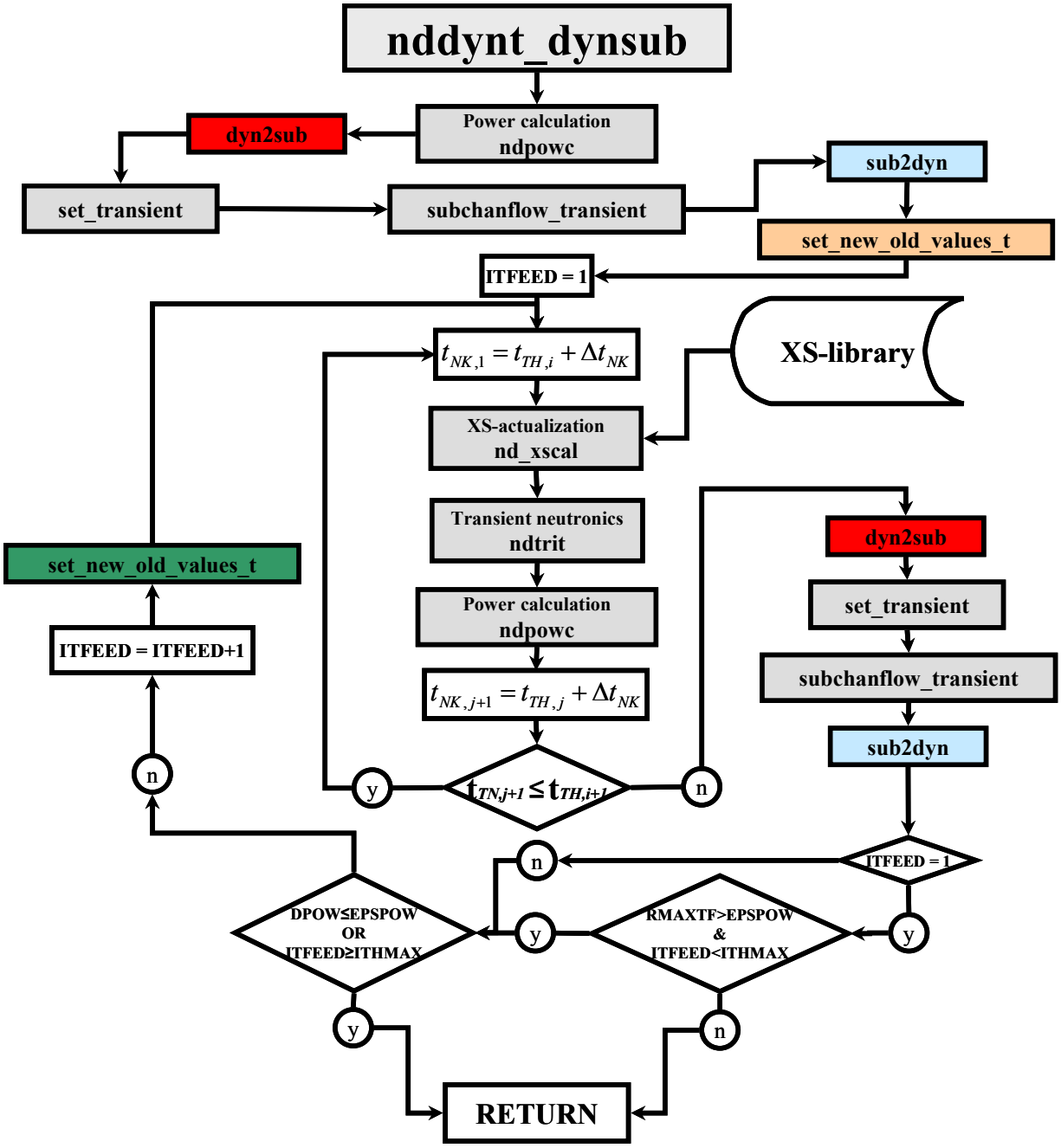


Figure 4–15 Flow diagram for the time integration in DYNSUB

4.4.5 Structure of the DYNSUB distribution package

The structure of DYNSUB is based in the GNU Autotools [GNUAutotools] for distributing projects. It is also extended and maintained by means of a controlled version using SUB-VERSION [Collins2008].

The source structure of DYNSUB was built in an efficient way isolating the subroutines of SUBCHANFLOW and DYN3D-SP3 standalone where changes related with the coupling were done. In this way an update of a new version of either DYN3D-SP3 or SUBCHAN-FLOW can be done without too many changes. In such case, the subroutines that have been

modified and have nothing to do with the coupling can be simply substituted. However special attention must be pay to the subroutines involved in the coupling. For this reason such subroutines are stored in a separate folder and thus if the new version of the standalone codes involves changes in these subroutines a special treatment has to be done in order to take into account the new capabilities without damaging the coupling process.

Figure 4–16 shows the structure of the distribution package of DYNSUB. The main directory ***DYNSUB_SRC*** contains several files and subdirectories together with the *Makefile.am* and *Configure.ac* files devoted to perform the automatic compilation and linkage of libraries. Inside the ***src*** directory, the subdirectory ***DYNSUB*** is located with four main branches namely: ***TEST***, ***DYN3D-SP3***, ***SUBCHANFLOW*** and ***PHYTON_TOOLS***.

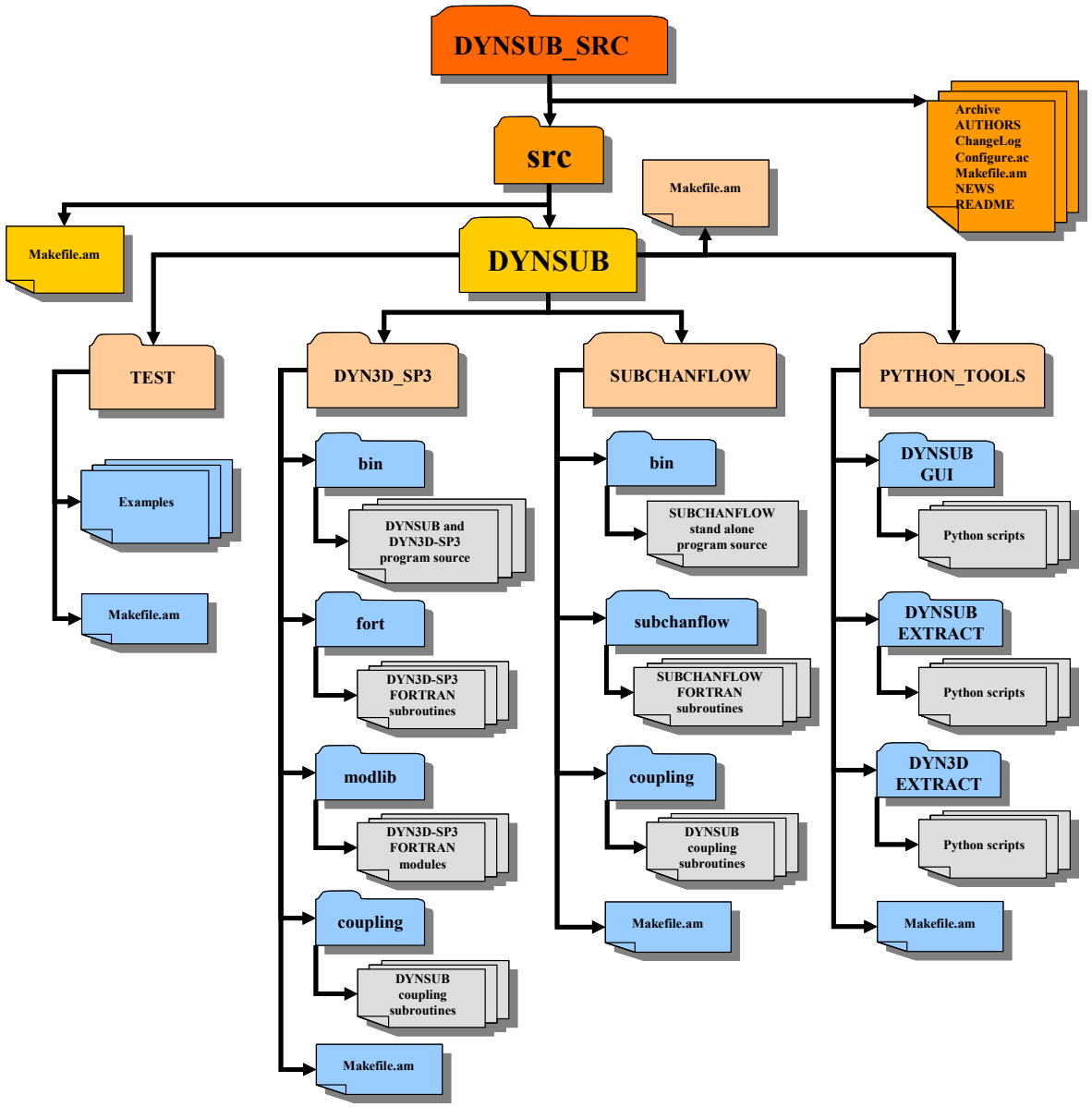


Figure 4–16 Structure of DYNSUB as a distributed project.

The ***TEST*** subdirectory contains a complete set of predefined examples that can be run after compilation and installation, its Makefile is in charge of install them in the appropriate path. On the other hand the central subdirectories ***DYN3D-SP3*** and ***SUBCHANFLOW*** have a similar structure. Additionally to the source files of each program, each of them has a ***coupling*** subfolder in which all the new subroutines as well as the modified subroutines of the original sources are stored. In this way, as previously mentioned, a new release of ***SUBCHANFLOW*** or ***DYN3D-SP3***, can be easily updated by substitution of the original sources in the corresponding directories (***fort*** and ***modlib*** for DYN3D-SP3 and ***subchanflow*** in the case of SUBCHANFLOW) and modifying the subroutines stored in ***coupling*** (just in case the new release includes changes in them). The standalone version of each code is additionally compiled and linked for getting the executable (binary file).

The installation of DYNSUB is performed using an automatic installation script in which just the path of the source and of the installation directory must be given. The use of such script together with the instructions for making working copies from the DYNSUB repository are described in Annex C. In Figure 4–17, the structure of the installed package (after running the installation script) is shown.

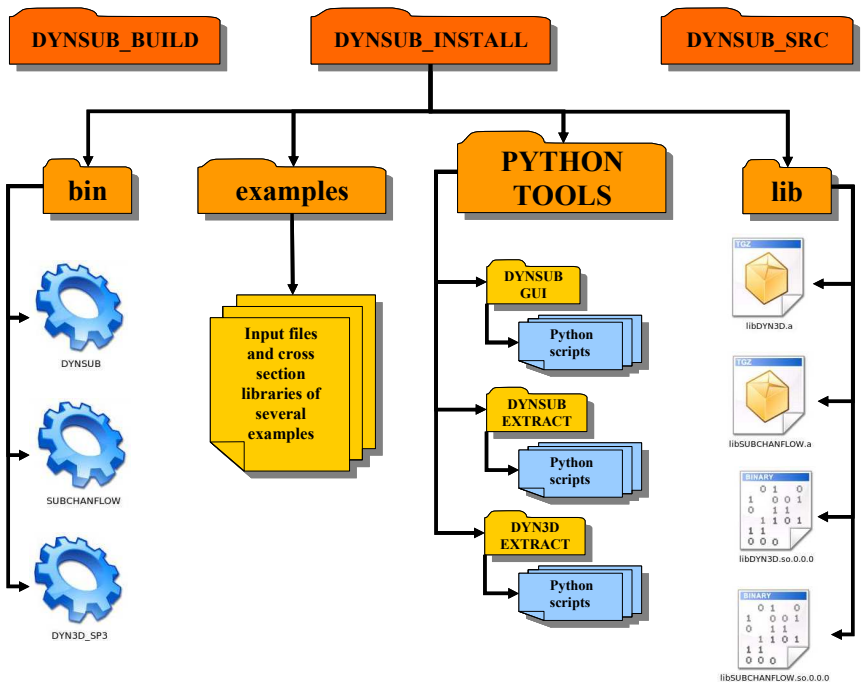


Figure 4–17 DYNSUB as an installed program

The ***DYNSUB_SRC*** directory is a copy of the original source, ***DYNSUB_BUILD*** is where the compilation was done and ***DYNSUB_INSTALL*** contains a ***bin*** directory with the executables of DYNSUB and SUBCHANFLOW and DYN3D-SP3 standalone, a ***lib*** directory with the needed libraries coming from the compilation, an ***example*** folder with several problems ready to run and a ***PYTHON_TOOLS*** folder with python scripts for extracting the data of DYN3D_SP3 standalone and DYNSUB (described in Annex C), as well as the DYNSUB Graphical User Interface described in Annex A.

4.4.6 Description of new subroutines

All new subroutines involved in the coupling are stored in the ***coupling*** subfolders of DYN3D-SP3 and SUBCHANFLOW. In Table 4-I and Table 4-II, the new subroutines for DYN3D-SP3 (15 subroutines) and SUBCHANFLOW (12 subroutines) respectively are listed and briefly described.

Table 4-I Description of new subroutines in the *coupling* folder of DYN3D-SP3

Subroutine	Description
dyn2sub.f	The pin power distribution calculated with DYN3D-SP3 is transferred to SUBCHANFLOW at every iteration step or time step. Storage of pin power data in the binary file DYN3D.bin.
sub2dyn.f	The feedback parameters from SUBCHANFLOW are transferred to DYN3D-SP3. The weighting of moderator temperature and density by means of equation (4.29) is done. Safety parameters are calculated (rod with the maximal fuel temperature, rod with the hottest clad temperature, location of the hottest point in the reactor, etc.). Storage of TH-FB parameters in the binary files SCF_DM.bin, SCF_TF.bin, SCF_TM.bin, MAX_FUEL.bin.
rectbundle.f	The definition of subchannels and input tables for SUBCHANFLOW is done by means of the Pre-processor. It reads a reduced input file (Figure 4–7) and the rest of the geometrical data are taken directly from the values coming from the geometry definition in the normal input file of DYN3D-SP3.
clean_rectbundle.f	Auxiliary subroutine for removing the comments (lines starting with "!") in the Pre-processor input file.
ndallint_subchanflow.f	Performs the allocation of arrays for the pin based feedback parameters coming from SUBCHANFLOW and used in DYN3D-SP3.
nd_xscal_sub.f	Calls the different subroutines for calculation of cross sections using feedback from SUBCHANFLOW.
ndfba_22_sp3_sub.f	Calculates (interpolates) the actual cross sections from pin based feedback coming from SUBCHANFLOW for a given thermal-hydraulic state of the core.
set_transient.f	Sets the transient conditions for SUBCHANFLOW based in power changes coming from the neutronic part.
subchan_dev.f	Calculates the deviations of the TH-FB parameters needed as criteria for stopping the steady state iteration process. The core average feedback properties (Doppler temperature and moderator temperature and density) are also calculated.
ndinpsta_dynsub.f	Input of data for stationary calculation with DYNSUB.
ndstat_dynsub.f	Performs the steady state calculation of DYNSUB.
ndtran_dynsub.f	Performs the transient calculation of DYNSUB.
nddynt_dynsub.f	Performs the time integration over the thermal-hydraulic step.
ndallxs_sp3.f	Subroutine with compiler dependant changes mainly related with writing format or initialization of variables. Changes are not related directly with the implementation.
ndset_sp3.f	Subroutine with compiler dependant changes mainly related with writing format or initialization of variables. Changes are not related directly with the implementation.

Table 4-II Description of new subroutines in the *coupling* folder of SUBCHANFLOW

Subroutine	Description
subchanflow_init.f90	Initialization of variables and arrays of SUBCHANFLOW.
subchanflow_static.f90	Performs a static calculation in every iteration step.
subchanflow_transient.f90	Performs a transient calculation for a given time step.
set_new_old_values.f90	In the steady state iteration process, the initial values of several arrays of SUBCHANFLOW are stored. If it is necessary to repeat the iteration, with the original values (stored) plus the new power distribution a new static calculation can be done. When the convergence criteria are met, the final values are the ones stored for starting the transient.
set_new_old_values_t.f90	The same logic as above but in every iteration step during transient (when necessary or desired). Some other values are additionally stored in case of transient, for instance the power distribution and the fuel rod temperature profile.
set_transient_variables.f90	Allocation of SUBCHANFLOW variables for performing a transient calculation.
dyn_average.f90	Writes the radial-averaged thermal-hydraulic values of the system.
dyn_results.f90	Writes some detailed information in the output file, for instance, the temperature profile of the hottest rod in the system.
close_units.f90	Close of files used by SUBCHANFLOW.
a2_var_global.f90	Original SUBCHANFLOW module with changes related with the coupling. New definition of arrays for saving old values.
setup.f90	Original SUBCHANFLOW subroutine with changes related with the coupling. Allocation of arrays for saving old values.
solution.f90	Original SUBCHANFLOW subroutine with changes related with writing of output data. Some output data is written also in the general DYNSUB output file.

4.5 Investigations with DYNSUB

4.5.1 Introduction

For testing the performance of DYNSUB two cases were defined.

- Case 1: A fast running test case (2x2 minicore) based in the "OECD/NEA and U.S. NRC PWR MOX/UO2 core transient Benchmark" [Kozlowski2003] and previously defined and studied in [Seubert2008] and [Christienne2010] was chosen. The steady state and a control rod ejection were calculated and compared with the DYN3D-SP3 stand alone results. Comparisons with the explicit and the FPI temporal coupling options are also explored.

- Case 2: In order to evaluate the practical feasibility of DYNSUB, a control rod ejection in an eighth core geometry (for the same Benchmark) is presented. Comparisons with the DYN3D-SP3 standalone are also done.

The Cross Sections library with 8 Group pinwise homogenized XS's used in the two cases is provided in the NEMTAB-like format used in the OECD MSLB Benchmark [Ivanov1999]. The original library of the Benchmark problem was replaced with the library generated and validated in [Beckert2008] and [Beckert22008] for the development of DYN3D-SP3, which is based in the same operational conditions but differs in some of the lower energy cut-off of the 8 – group structure. Furthermore includes the P_1 scattering table used in the SP_3 transport approximation. In general, the 8 Group structured library contain branches in fuel temperature (3), moderator density (3) and boron density conditions (3) as well as 7 burn-up steps, to cover the expected range of core operating conditions. The moderator temperature effect is treated implicitly in the moderator density term assuming constant pressure of 15.5 MPa as described in [Kozlowski2003].

The configuration for the UO_2 and MOX fuel assemblies is shown in Figure 4–18.

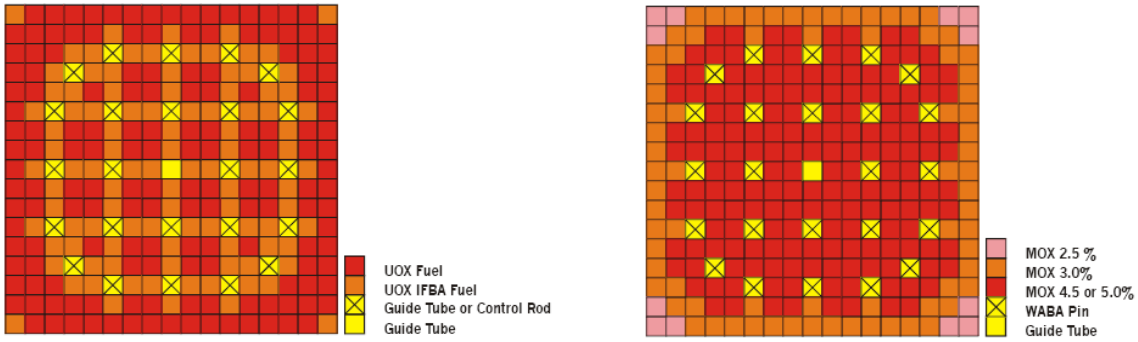


Figure 4–18 Configurations for the UO_2 (left) and MOX (right) fuel assemblies.

For control of reactivity, pins with Integral Fuel Burnable Absorber (IFBA) and Wet Annular Burnable Absorbers (WABA) are used in the UO_2 and MOX fuel assembles respectively. The

IFBA is a coating of zirconium diboride (ZrB_2) on the fuel pellets and provides reactivity control over a relatively short burn-up period. All UO_2 assemblies contain 104 IFBA pins located in the highest worth regions (vicinity of the guide tubes) and in the corners. The WABA is an annular pellet of Al_2O_3-B_4C with wet (water filled) central region and Zircaloy cladding. In contrast to IFBA, WABA provides relatively long term reactivity control. The MOX fuel assemblies are designed with 24 WABA pins inserted in the guide tube locations [Kozlowski2003].

4.5.2 Case 1: Fast running minicore

4.5.2.1 Definition of the problem for the Case 1

Figure 4–19 shows the minicore arrangement. It consists of a bundle of 2x2 fuel assemblies of UO2 at 4.2%. Reflective (North and East faces) and vacuum (South and West faces) boundary conditions are imposed.

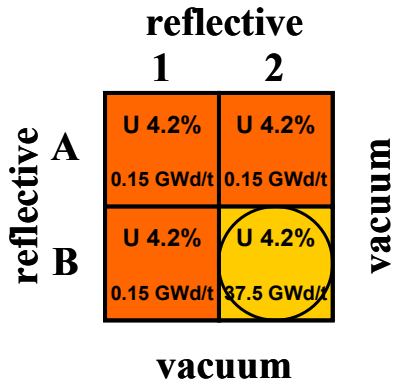


Figure 4–19 Geometrical configuration of the 2 x 2 minicore

The initial operational conditions used in this case are presented in Table 4-III. For the upper and lower boundary conditions, in [Seubert2008] and [Christienne2010], water reflectors together with vacuum boundary conditions at the end were used in both sides. In this case, the lower and boundary conditions were adjusted by means of axial albedos in order to get a similar k_{eff} as reported in [Christienne2010].

At the initial steady state condition, the B2 assembly has a control rod fully inserted. Furthermore, such assembly has the biggest burn-up (37.5 GWd/t).

After a steady state calculation, the transient starts at time zero with a fast control rod ejection in assembly B2. The control rod must be fully ejected after 0.1 seconds. The transient continues until 1 second.

Table 4-III Operational conditions for the 2 x 2 minicore

Operational condition	value
Number of assemblies	4
Power (MWth)	22.9
Inlet Temperature (°C)	287.0
Inlet Pressure (MPa)	15.5
Pressure drop over the core (kPa)	125.0
Active Flow (kg/sec)	328.4849
Fuel lattice, fuel rods per assembly	17 x 17, 264
Active length (cm)	365.76
Assembly pitch (cm)	21.41
Pin pitch (cm)	1.26
Axial boundary conditions	α = 0.75
Number of axial nodes	17 (21.515 cm)
Boron concentration (ppm)	1015.0

4.5.2.2 Steady state results for the Case 1

In order to evaluate the rod worth of the assembly B2, two steady state calculations were performed with DYN3D-SP3 standalone and DYNSUB, all rods out (ARO) and all rods in (ARI). The rod worth is calculated by means of equation (4.35) [Christienne2010] and the insertion of reactivity in Dollars ($) by means of (4.36).

$$rod\_worth = \frac{k_{eff}^{ARO} - k_{eff}^{ARI}}{k_{eff}^{ARO} k_{eff}^{ARI}} \tag{4.35}$$

$$\rho = \frac{rod\_worth}{\beta_{eff}} \tag{4.36}$$

The beta effective value ($\overline{\beta_{eff}}$) for this core configuration is 695.7203 pcm and it was obtained by means of equation (4.37).

$$\overline{\beta_{eff}} = \frac{\sum_{m=1}^{NASS} \beta_{eff}^{m}}{NASS} \tag{4.37}$$

where β_{eff}^{m} is the beta effective for material *m* coming from the two-group diffusion cross sections library of the OECD NEA PWR MOX Benchmark [Kozlowski2003], and *NASS* is the number of assemblies in the configuration.

The results are presented in Table 4-IV. Differences in k_{eff} up to 73.9 pcm for the ARO configuration and up to 88.2 for the ARI configuration were found, taken as a reference the DYN3D-SP3 calculation.

Although the k_{eff} for DYN3D-SP3 is in both cases bigger than the one calculated with DYNSUB, the worth of the rod for DYNSUB (944.95 pcm) is greater than the rod worth calculated with DYN3D-SP3 (928.38 pcm). Furthermore, the beta effective value (β_{eff}) is in both cases smaller than the rod worth. Therefore a fast reactivity insertion leading the minicore to a super prompt critical state is expected.

Table 4-IV k_{eff}, rod worth and reactivity (ρ) comparison between DYN3D-SP3 and DYNSUB for the 2 x 2 minicore.

	DYN3D-SP3	**DYNSUB**	**Deviation (pcm)**
k_{eff}^{ARO}	0.980843	0.980104	73.9
k_{eff}^{ARI}	0.971992	0.971110	88.2
Rod worth (pcm)	928.38	944.95	-16,57
ρ **($)**	1.3344	1.3582	1.78 %

Figure 4–20 shows for the ARI configuration, the convergence of k_{eff} (top-left), and how the deviations of the convergence parameters behave in the iteration process (the convergence criteria are defined by the user via input and calculated by means of equations (4.32), (4.33) and (4.34)).

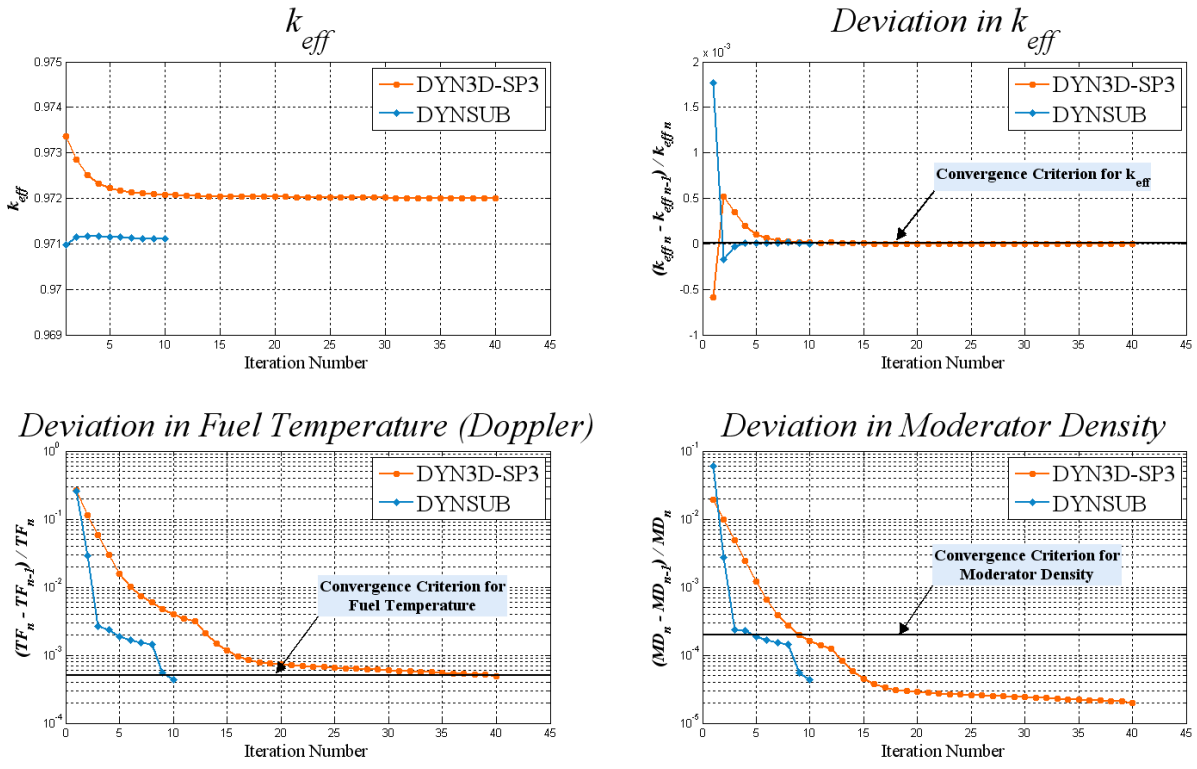


Figure 4–20 k_{eff} and deviations of the convergence criteria in the steady state calculation ARI of Case 1.

Although the convergence criteria in k_{eff} (top-right) and in moderator temperature (bottom-right) were met relatively easy by the two codes, for DYN3D-SP3 standalone it was necessary much more iterations (41 in total) in order to fulfil the fuel Doppler temperature requirement (bottom-left). On the other hand, the thermal-hydraulic behaviour of SUBCHANFLOW allowed DYNSUB to finish in just 10 iterations. It is important to mention that in order to validate the coupling very strict criteria were used specially in the fuel temperature ($5x10^{-4}$), relaxing the convergence criteria would lead to a faster fulfilment of all the criteria by DYN3D-SP3 as it can be inferred in the trend of the fuel temperature deviation for DYN3D-SP3 (bottom-left plot).

The axial power distribution of Figure 4–21 shows very good agreement between the two codes for the ARI configuration (Figure 4–21 top). The maximum is reached in the same axial node and only small differences between the two profiles can be distinguished. The ARO configuration (Figure 4–21 bottom) shows also good agreement in the lowest part of the reactor however greater deviations are distinguished in the upper part.

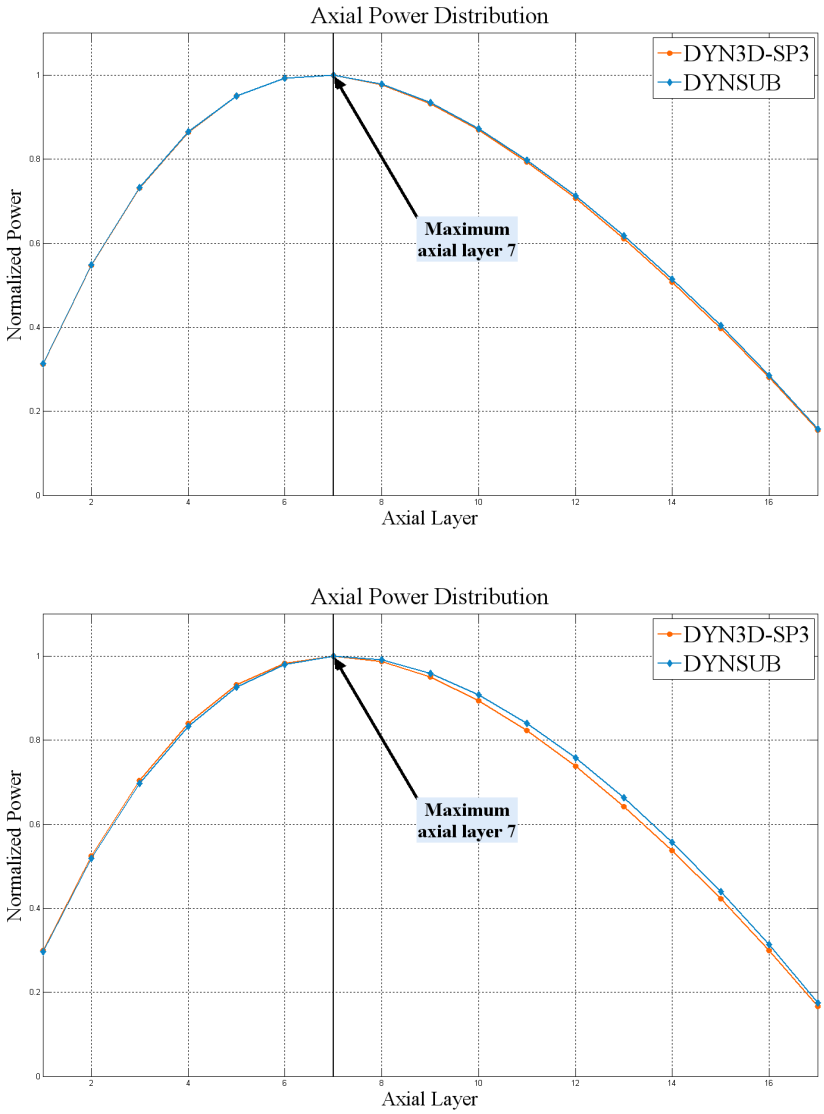


Figure 4–21 Comparison of the normalized axial power profile.

Top: ARI configuration, Bottom: ARO configuration.

The radial pin power distribution for DYNSUB in the hottest layer (layer 7) for the ARI configurations is shown in Figure 4–22. The effect of the control rod in the B2 assembly can be clearly observed

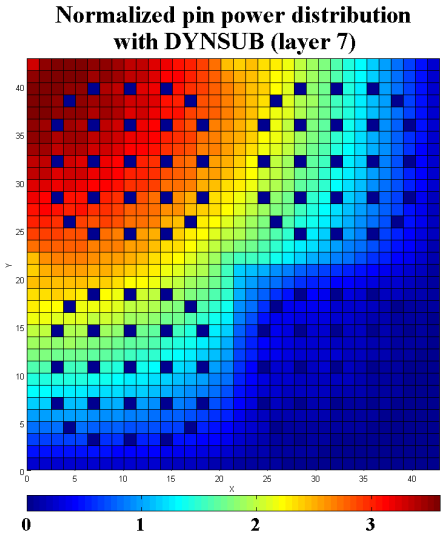


Figure 4–22 Normalized pin power distribution calculated with DYNSUB on the hottest layer (layer 7) for the ARI configuration.

On Figure 4–23, the deviation from the DYN3D-SP3 calculation in percent is shown. The differences range from approx. -1.0% in the coldest regions until +1.5% in the hottest regions of each assembly, with the understanding that a negative value implies an over estimation of the power by DYNSUB and therefore, a positive value results in an under estimation.

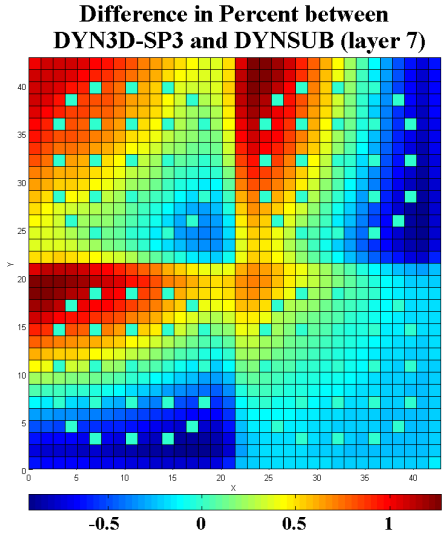


Figure 4–23 Percentual difference between DYN3D-SP3 and DYNSUB for the hottest layer (layer 7).

Such differences arise due to the more detailed description of the thermal-hydraulic parameters coming from SUBCHANFLOW and are explained hereafter. Whereas DYN3D-SP3 pre-

dicts one value for each axial node (Figure 4–24), DYNSUB is able to predict a 2-D distribution inside each axial node (Figure 4–25).

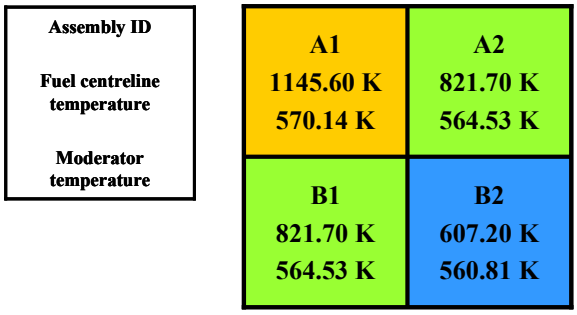
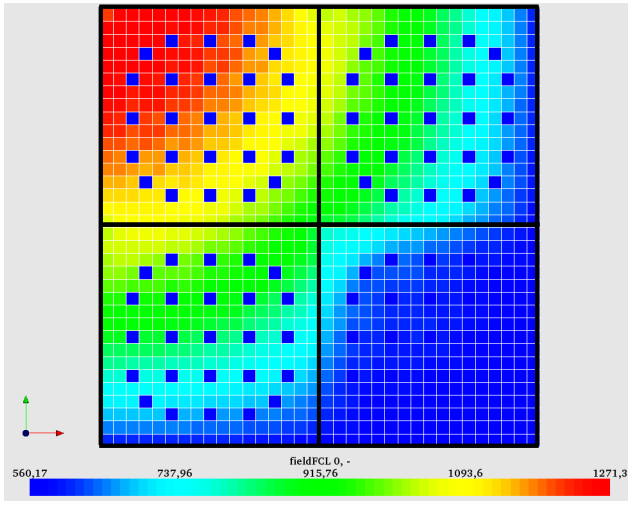
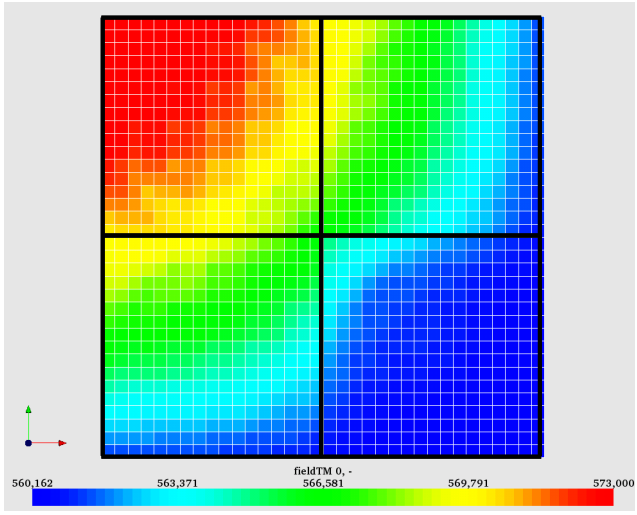


Figure 4–24 Thermal-hydraulic parameters on the hottest layer for the ARI configuration with DYN3D-SP3.



(a)



(b)

Figure 4–25 Thermal-hydraulic parameters on the hottest layer for the ARI configuration with DYNSUB: (a) Fuel centreline temperature; (b) Moderator temperature.

The fuel centreline temperature for the hottest assembly (A1 on the top-left) is in the case of DYN3D-SP3 1145.60 K. DYNSUB however, predicts the hottest temperature ranging from approx. 915 K in the coldest region of the assembly (bottom-right corner) until 1271 K in the hottest region (top-left corner). The bigger fuel centreline temperature predicted by DYNSUB in this assembly (top-left corner) results as an underestimation of the power in DYNSUB (due to the Doppler effect as well as the decrease in moderator density), likewise, the smaller temperature prediction (bottom-right corner) leads to a DYNSUB overestimation of power (see Figure 4–23).

Similar analysis can be done in the other fuel assemblies. It is important to stress that the small deviations in the normalized radial power distribution (from approx. -1.0% to +1.5%) shown in Figure 4–23 resulted in 11% difference in the hottest centreline temperature predicted by DYNSUB (1271.0 K) in comparison with DYN3D-SP3 (1145.6 K). Further discussion related with these differences will be presented in the transient analysis results of the next subsection.

4.5.2.3 Transient results for the Case 1

As previously mentioned, the transient under study corresponds to a very fast rod ejection. As a consequence, the rapid increase in reactivity brings the system to a prompt critical state and a significant power rise is expected in the initial milliseconds. The strong negative fuel temperature (Doppler) feedback reactivity is the only inherent reactivity feedback responsible to avoid a bigger power excursion and to mitigate the power peak. The fuel temperature reactivity feedback has to do with the Doppler broadening of resonances and is directly associated with the fuel temperature.

In order to evaluate the impact of the detailed thermal-hydraulic model several assumptions were made in order to be able to perform comparisons between DYN3D-SP3 and DYNSUB. The thermal-hydraulics models FLOCAL and SUBCHANFLOW have different models implemented for the heat conduction in the fuel pin as well as for the heat transfer from the cladding to the fluid. However, for the sake of comparison, fixed and idealized properties were established for the heat conduction in the fuel for the two codes. Table 4-V shows a resume of them.

Table 4-V Heat conduction properties for the fuel element.

Heat conduction properties	**value**
Fuel thermal conductivity (kW/(m*K))	3.0×10^{-3}
Fuel density (kg/m^3)	10410.0
Fuel specific heat (kJ/(kg*K))	3.0×10^{-1}
Cladding thermal conductivity (kW/(m*K))	1.95×10^{-2}
Cladding density (kg/m^3)	6504.0
Cladding specific heat (kJ/(kg*K))	3.0×10^{-1}
Number of radial nodes in the fuel	10
Heat transfer coefficient for the gas gap (kW/(m^2*K))	10.0
Fraction of heat released in the fuel	1.0

The heat transfer coefficient between the cladding and moderator is to be calculated using the own internal correlation of the both codes. Because this is a very fast transient, any variation may not have a great impact at the initial part of the transient. However, after the peak such variations may affect the asymptotic behaviour of the power.

The effective Doppler temperature $T_{DOPPLER}$ is determined from the fuel rod centreline temperature $T_{f,C}$ and the rod surface temperature $T_{f,S}$ by means of:

$$T_{DOPPLER} = (1-\alpha)T_{f,C} + \alpha T_{f,S} \tag{4.38}$$

For the temporal scheme, calculations with FPI or nested loop iteration together with a small and fixed time step were done with DYN3D-SP3 and DYNSUB. As previously mentioned, the FPI is an approximation to an implicit coupling scheme [Watson2010]. Thus, such calculations can be seen as a "reference calculation". Details about the temporal parameters used by the two codes are presented in Table 4-VI. A comparison among the different temporal approaches is given later.

Table 4-VI Temporal parameters for the calculation of Case 1.

Temporal parameters	**value**
Model	FPI
Time step (milliseconds)	1
Maximal number of internal iterations at each time step	10
Convergence criterion for power deviation	1.0×10^{-3}

The global behaviour of the power for the two calculations is presented in Figure 4–26.

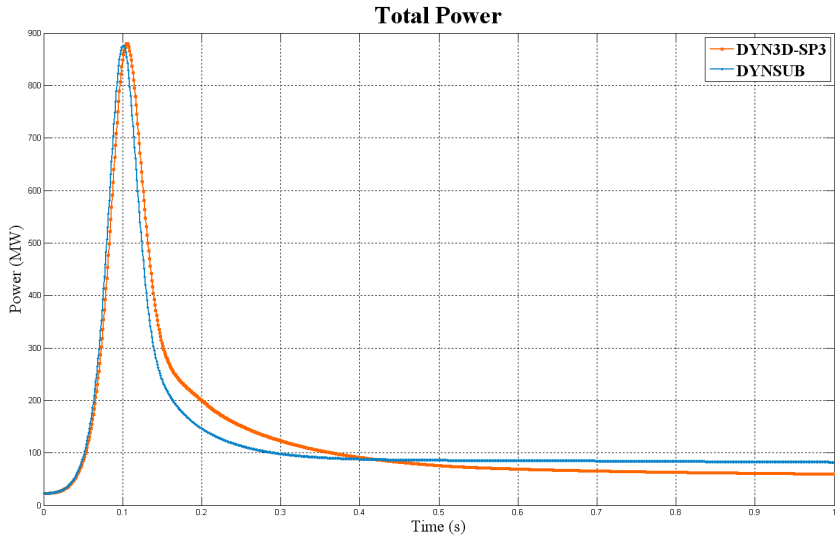


Figure 4–26 Global behaviour of total power during the transient.

As expected, a power peak was reached after the total ejection of the control rod. In Table 4-VII, details about the results are given. The deviations were calculated taken DYN3D-SP3 calculation as reference with:

$$Deviation = \left(\frac{Val_{DYNSUB} - Val_{DYN3D-SP3}}{Val_{DYN3D-SP3}} \right) * 100\% \tag{4.39}$$

Table 4-VII Comparison of DYN3D-SP3 and DYNSUB

	DYN3D-SP3	**DYNSUB**	**Deviation**
Power peak (MW)	879.651	875.543	- 0.46 %
Peak time (milliseconds)	105	102	-2.85 %
Asymptotic power at the end of the transient (MW)	59.5304	82.2091	38 %
Relative CPU time	1 (1530 min)	1.26 (1939 min)	26 %

The 3D thermal-hydraulic effect in DYNSUB calculation increased the computational time in 26% comparing with DYN3D-SP3 calculation with its 1D thermal-hydraulic model FLO-CAL.

On the other hand, although the rod worth calculated with DYNSUB was 1.78 % bigger than the one calculated with DYN3D-SP3, a power peak 0.46% smaller was predicted with DYN-SUB. The reason can be explained by analyzing the Figure 4–27 where the global behaviour of the Doppler temperature and moderator density are presented as a function of time.

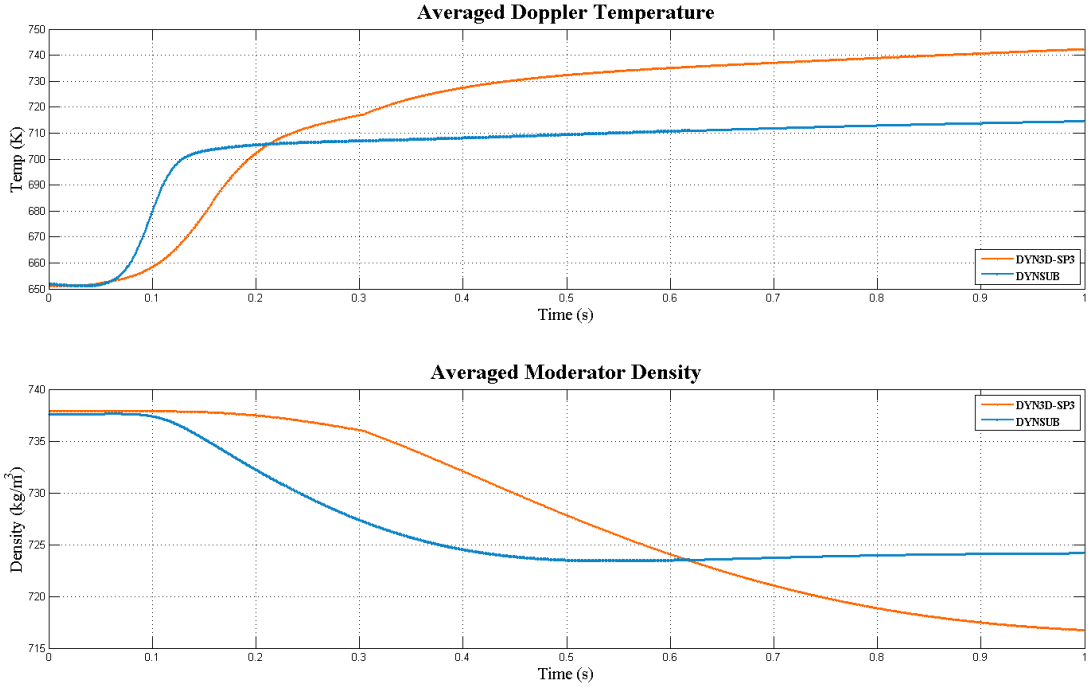


Figure 4–27 Global behaviour (averaged over the whole minicore) of the Doppler temperature and moderator density during the transient.

It can be seen in the upper graph of Figure 4–27 that the averaged Doppler temperature predicted with DYNSUB increases faster than the one calculated with DYN3D-SP3. Thus, a faster insertion of negative reactivity is given in the DYNSUB case. After reaching the power peak (102 milliseconds), the decrease on moderator density, as a consequence of the fuel temperature increase, appears as an additional negative contribution to the reactivity. The moderator density predicted with DYNSUB decreases also faster than the one predicted with DYN3D-SP3. These two effects are the responsible of the narrower shape of the DYNSUB peak in comparison with the DYN3D-SP3 peak.

The discrepancies arising after the power peak and in the asymptotic part are due to the different heat transfer and thermal-hydraulics models and correlations used in SUBCHANFLOW and FLOCAL.

In Figure 4–28, the maximal fuel temperature (centreline) as a function of time for the two calculations is shown. Whereas DYN3D-SP3 calculates the assembly-averaged maximal fuel temperature, DYNSUB is able to predict the rod with the maximal (rod 241) and the minimal (rod 17) fuel temperature within the hottest fuel assembly. Differences over 100 K between the assembly-averaged maximal fuel temperature (predicted with DYN3D-SP3) and the maximal fuel temperature of the hottest rod (predicted with DYNSUB) can be observed. However in the asymptotic part the deviations decrease.

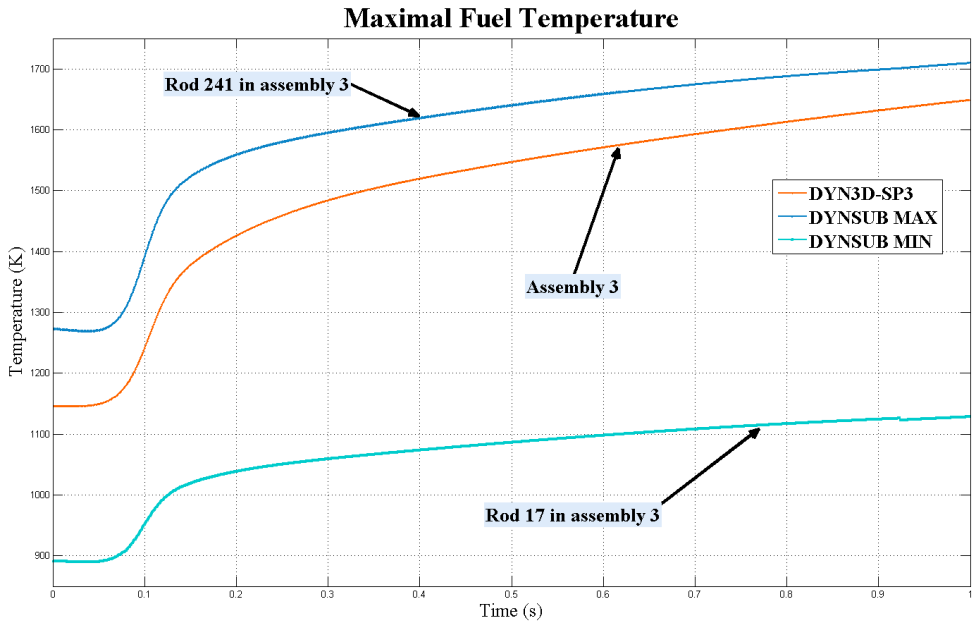


Figure 4–28 Maximal fuel temperature during the transient.

Furthermore, details about the location of the axial level with the hottest rod as a function of time are presented in Figure 4–29. It can be seen that although the same hottest assembly is predicted by the two codes, there are differences between the two calculations. DYNSUB prediction moves faster from the axial level 7 at the beginning of the transient until the axial level 5 at the end of the transient. In the case of DYN3D-SP3, the hottest axial layer moves from 7 to 6 and stabilizes.

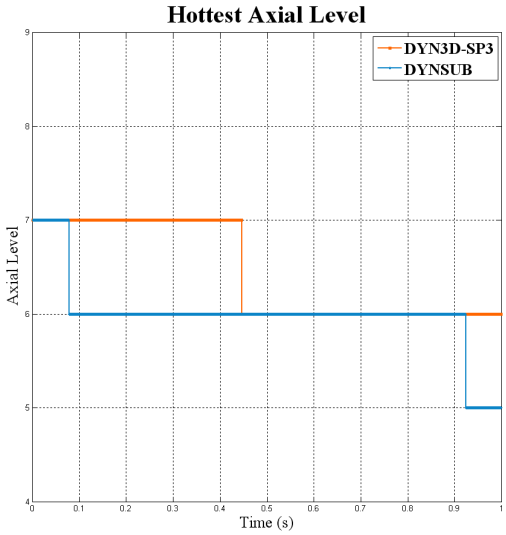


Figure 4–29 Axial position of the hottest assembly during the transient.

Figure 4–30 shows the fuel centreline, the clad surface and the moderator axial temperature profiles at the DYNSUB peak time (Figure 4–30 (a), (b) and (c)) and at the end of the transient (Figure 4–30 (d), (e) and (f)) respectively. The axial shift presented in Figure 4–29 can be seen by comparing the maximal fuel temperature profiles of Figure 4–30 cases (a) and (d).

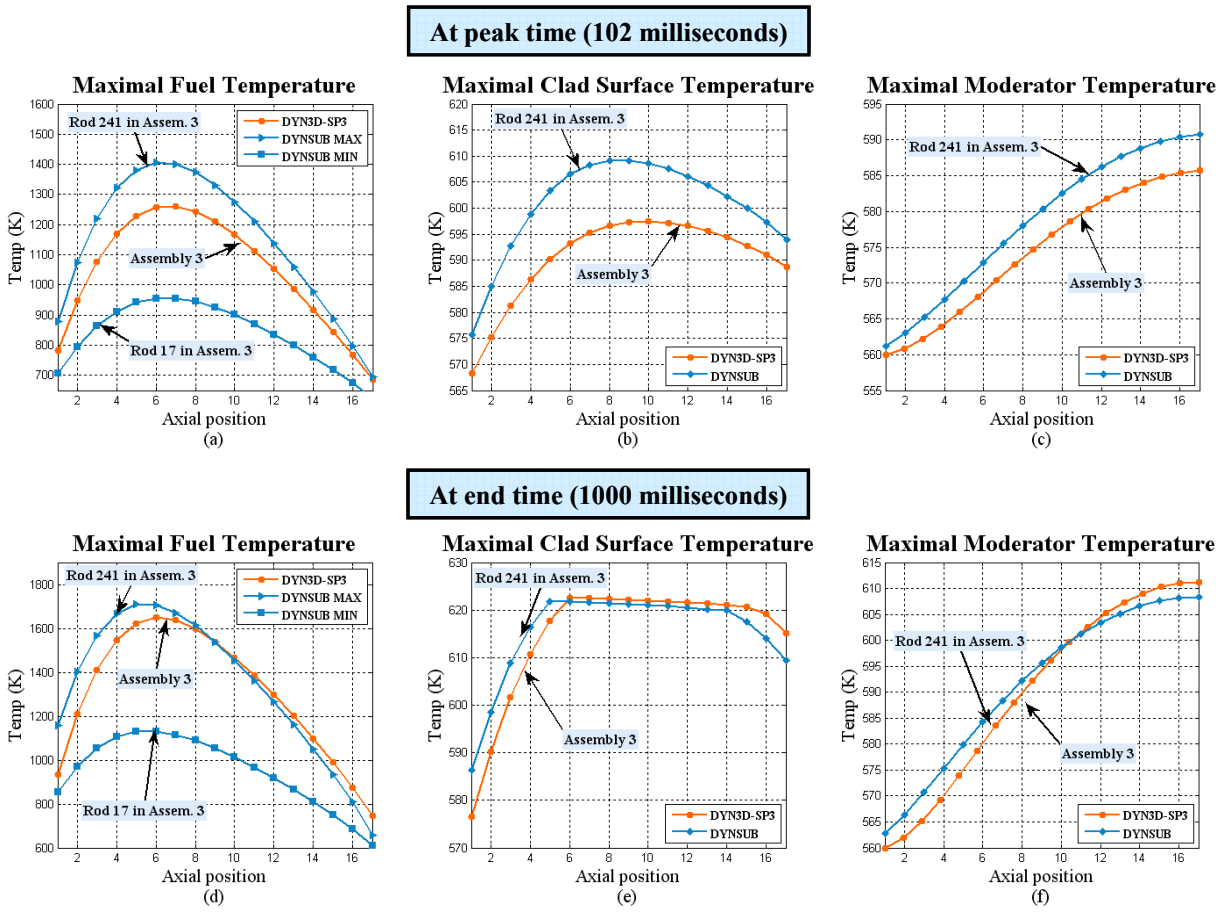


Figure 4–30 Axial temperature distribution for the fuel centreline, the clad surface and the moderator at peak time (102 milliseconds) and at the end of the transient (1000 milliseconds).

At the peak time, differences up to 150 K in the centreline fuel temperature and up to 12 K in the cladding temperature are observed in all cases with an overestimation of DYNSUB. These differences become smaller as the transient progresses. At the end of the time there is still a DYNSUB overestimation of the fuel centreline temperature (about 60 K) but it is not the case of the maximal cladding temperature and the moderator temperature.

A chronologic analysis summarizing the progress of the transient and the main effects is presented in Table 4-VIII.

Table 4-VIII Chronologic analysis of the transient calculation.

Time (ms)	Description	Comparison
0.0	The control rod worth at steady state has been calculated.	$\frac{rod\_worth_{DYNSUB}}{rod\_worth_{DYN3D-SP3}} = 1.017$
0 – 105	Fast insertion of reactivity due to the fully ejection of the control rod. Excursion of power.	$POWER_{DYNSUB} > POWER_{DYN3D-SP3}$
	Fuel temperature starts increasing.	$Dopp\_temp_{DYNSUB} > Dopp\_temp_{DYN3D-SP3}$
	The negative Doppler reactivity counteracts the positive reactivity.	$Dopp\_reac_{DYNSUB} > Dopp\_reac_{DYN3D-SP3}$
	A power peak is reached.	$\frac{Peak\_time_{DYNSUB}}{Peak\_time_{DYN3D-SP3}} = 0.97$ $\frac{Peak\_pow_{DYNSUB}}{Peak\_pow_{DYN3D-SP3}} = 0.995$
105 – 400	Moderator density starts decreasing	$Mod\_den_{DYNSUB} < Mod\_den_{DYN3D-SP3}$
	Decrease of moderator density becomes negative reactivity.	$Mod\_reac_{DYNSUB} > Mod\_reac_{DYN3D-SP3}$
	The power decreases fast due to both negative effects.	$POWER_{DYNSUB} < POWER_{DYN3D-SP3}$
400 - 1000	All reactivity mechanisms stabilize the power in an asymptotic behaviour. The change in power and thermal-hydraulic behaviour is due to different thermal-hydraulic models.	$POWER_{DYNSUB} > POWER_{DYN3D-SP3}$

4.5.2.4 Comparison of different temporal schemes of DYNSUB for the Case 1

The main goal of this subsection is to assess the impact of the two temporal schemes implemented in the coupling. On one hand, it is well known that an explicit scheme must use small time steps in order to get converged results. On the other hand the fixed point iteration method (FPI or nested loop), being an approximation to an implicit scheme, may allow the use of bigger time steps but with some penalty in the computational time due to the internal iterations at every time step.

In order to evaluate the impact of the use of bigger time step in order to decrease the computational time without compromising the accuracy of results, several cases were evaluated. The reference solution used in the comparison corresponds to the case presented in the previous section where a FPI scheme with a small time step (1 millisecond) and a strict convergence criterion in power deviation was chosen.

In all cases the operational conditions and heat transfer properties are the same as previously defined. A description of the cases analyzed and the differences in the temporal coupling schemes are presented in Table 4-IX.

Table 4-IX Cases analyzed with different temporal schemes.

	FPI_{ref}	**FPI_1**	**FPI_2**	**EXP_{ref}**	**EXP_1**
Temporal scheme	FPI	FPI	FPI	Explicit	Explicit
Time step (milliseconds)	1	4	6	1	4
Max. No. iterations	10	10	10	1	1
Convergence criterion for power deviation	$1.0x10^{-3}$	$5.0x10^{-3}$	$5.0x10^{-3}$	---	---

Additionally to the reference solution (FPI_{ref}), two more calculations with FPI were done. The FPI_1 considers a time step four times greater (4 milliseconds) than the reference solution. The second FPI case (FPI_2) used a time step six times greater (6 milliseconds). Furthermore, two calculations with an explicit scheme were chosen. The EXP_{ref} is the explicit version of the reference solution. A time step of 1 millisecond was used. The final case EXP_1 is also an explicit scheme with a time step 4 times bigger (4 milliseconds).

For the FPI cases, the maximum number of internal iterations per time step is set to 10. The convergence criterion in the power deviation was $5.0x10^{-3}$ for FPI_1 and FPI_2.

In order to provide an accurate description of the error as a function of time two metrics were used: a Power-Weighted Error (PWE) and an Error-Weighted Error (EWE). Both are defined as a weighted average of the error by equations (4.40) and (4.41) respectively. The PWE is similar to absolute error. It weights the percent error with the reference power, therefore the PWE diminishes the importance of error where low values of the power are presented and amplifies the error in the highest regions (for instance in the power peak). The EWE is similar to RMS (Root Mean Square) error. It weights the largest percent errors more than the small ones and is not linked to any reference parameter like power. [Kozlowski2006].

$$PWE = \frac{\sum_t |e_t| ref_t}{\sum_t ref_t} \tag{4.40}$$

$$EWE = \frac{\sum_t |e_t||e_t|}{\sum_t |e_t|} \tag{4.41}$$

with

$$|e_t| = \frac{calc_t - ref_t}{ref_t} \times 100\% \tag{4.42}$$

The global behaviour of the power (at the first 300 milliseconds) for the five cases is presented in Figure 4–31. At the peak time just the FPI_1, with a 0.69 % and EXP_{ref} with -0.92 % deviation from the reference solution (FPI_{ref}) were able to adequately represent the power peak. However, the explicit solution (EXP_{ref}) predicted a narrower shape of the peak whereas with the FPI_1 the power peak's shape is broadened but closer to the reference solution (the PWE for FPI_1 is 5.1257% comparing with the PWE of the EXP_{ref} 8.5682%). It is important to stress the great impact of the nested loop in the FPI_1 case in which the time step (4 milliseconds) is 300% bigger than the time step of EXP_{ref} (1 millisecond).

The other two calculations, FPI_2 and EXP_1, overestimated the power peak in 10.96 % and 13.61 % respectively. However it is also important to notice that FPI_2 considered a time step 50% bigger than the EXP_1 time step. A summary of the PWE and EWE over the whole time interval is presented in Table 4-X.

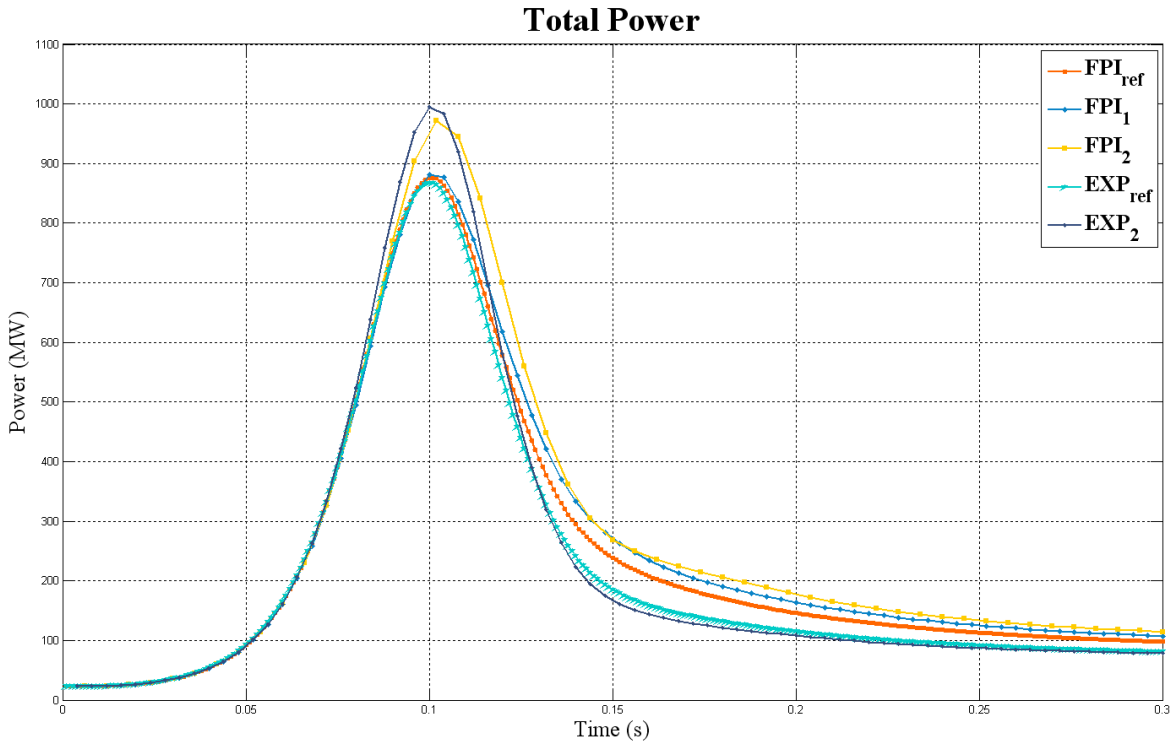


Figure 4–31 Total global power during transient for the different temporal schemes.

Figure 4–32 shows a similar situation for the maximal fuel temperature. FPI_1 and EXP_{ref} are able to represent the shape of the curve in a more accurate way. Also in this case, the FPI_1 calculation (EWE_TF = 1.6691%) is closer to the reference solution than the EXP_{ref}

(EWE_TF = 2.9398%). FPI_2 presents greater deviations (EWE_TF = 3.6308%) but the shape of the curve remains. On the counter case, the EXP_1 has a small error weighted for the fuel temperature (EWE_TF = 2.4584%) even smaller than the EXP_{ref} calculation, however the shape of the curve is clearly different.

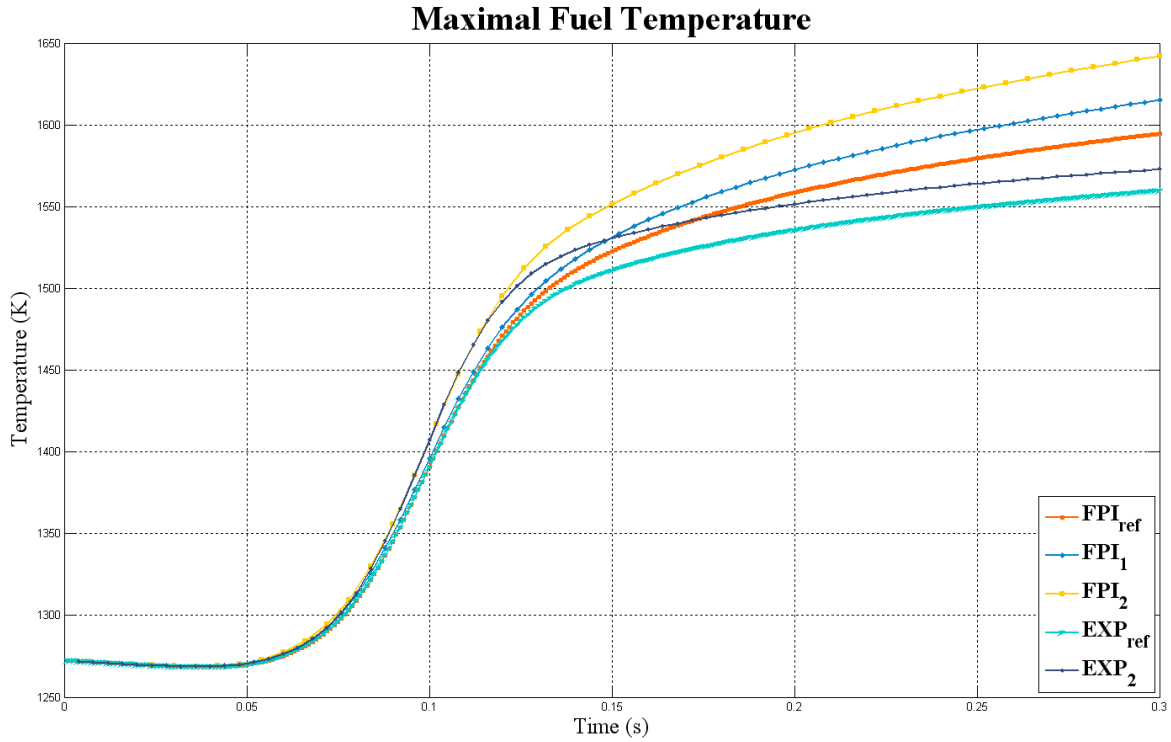


Figure 4–32 Maximal fuel temperature during transient for the different temporal schemes.

Finally in Figure 4–33 the time step neutronics – thermal-hydraulics (NK-TH) internal iterations for the FPI cases can be seen. As expected, the bigger the time step, the more internal iterations in every time step have to be done. In the three cases, as soon as the power increases significantly, the internal iterations in every time step start. For the reference solution (FPI_{ref}), one additional NK-TH iteration in every time step was needed. On the other hand, FPI_1 needed two additional and FPI_2 up to three additional iterations. The explicit schemes do not perform any NK-TH iteration inside the time step. For this reason they move from one time step to the other without a converged solution on exchanged parameters between the two fields.

The total number of NK-TH iterations and the computational times are also presented in Table 4-X. Comparing the two reference calculations, the FPI_{ref} needed 81 additional iterations than EXP_{ref}, all of them within the power peak. These additional iterations had an impact in the calculation time of 18% more time.

On the other hand, comparing the two calculations with a time step of 4 milliseconds, FPI_1 did 28 iterations more than EXP_1, however its performance was much more accurate than the explicit scheme, and the CPU time was very acceptable.

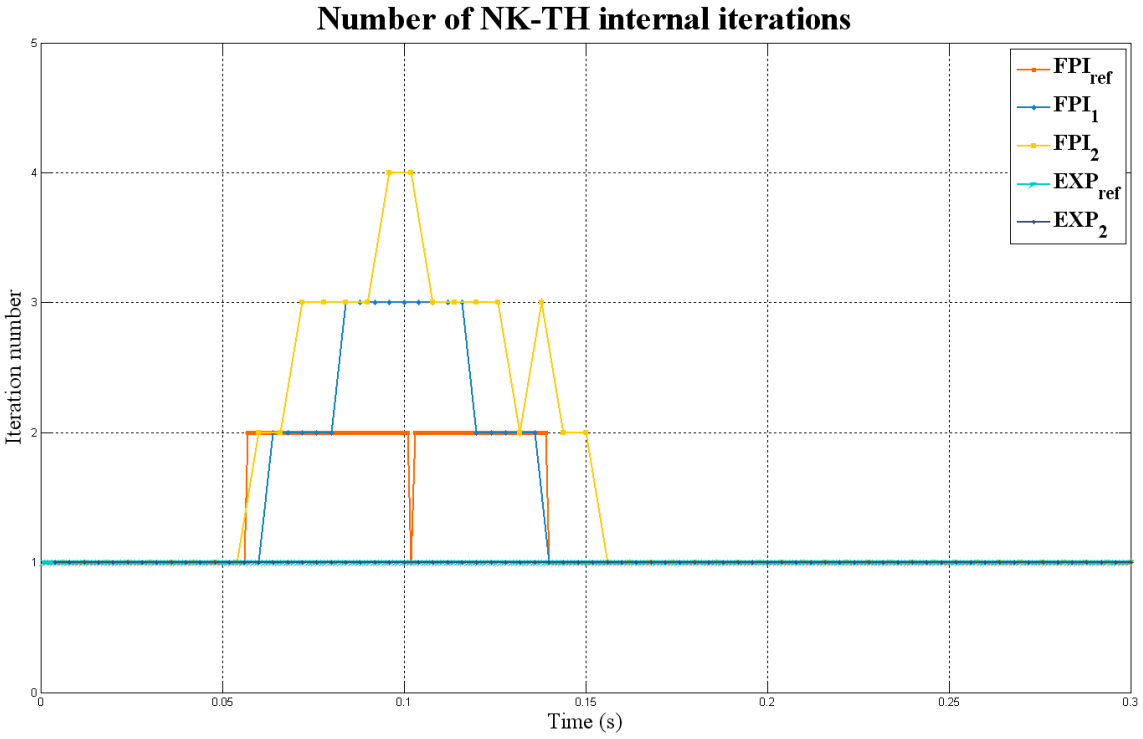


Figure 4–33 Internal iterations during transient for the different temporal schemes.

Table 4-X Cases analyzed with different temporal schemes.

	FPI_{ref}	FPI_1	$FP1_2$	EXP_{ref}	EXP_1
Power peak (MW)	875.54	881.60	971.5260	867.40	994.76
Difference in power peak (%)	---	0.69	10.96	-0.92	13.61
PWE (%)	---	5.1257	11.2650	8.5682	12.1694
EWE_P for Power (%)	---	0.3329	3.6643	0.9626	1.4624
EWE_TF for Fuel Temp (%)	---	1.6691	3.6308	2.9398	2.4584
Total number of iterations	1081	278	167	1000	250
CPU time (minutes)	1939	1343	1372	1589	1062
CPU time ratio (comparing with reference)	1.0	0.69	0.70	0.82	0.55

As a conclusion, it is clear that the effect of the nested loop (FPI) can play a very important roll in the calculations. The FPI allows moving from one time step to the other with converged solution. It also allows the use of bigger time steps without compromising the results, and even with a better prediction than the one calculated with an explicit scheme with very small time step (FPI_1 vs. EXP_{ref}).

Furthermore, comparing FPI_1 with FPI_2, it is important to notice the differences in CPU time between these two cases. Although the FPI_2 case needed less NK-TH iterations (167) than the FPI_1 (278), the calculation time was bigger (1372 minutes vs. 1343 of FPI_1). This fact is due to the NK internal iterations. When the time step is bigger, the differences in fission source and power are also bigger at each time step, thus, the internal convergence of the neutronics calculation is more difficult to reach and more internal iterations are needed.

From the four cases compared with the reference solution, the FPI_1 calculation had, in general, the better performance. The errors can be even decreased by choosing more strict power deviation criterion for the time step internal iteration. However, a more strict power deviation criterion will increase the number of internal iterations and therefore the CPU time. Therefore, it becomes an issue of optimization and more studies have to be done in order to define an optimal strategy. A complete implicit coupling either via Jacobian free Newton Krylov methods or direct solving of full Jacobian of coupled fields can be also investigated.

4.5.3 Case 2: Eighth of PWR core.

4.5.3.1 Definition of the problem for the Case 2

For a more realistic application, the core configuration of the OECD/NEA and U.S. NRC PWR MOX/UO2 core transient Benchmark [Kozlowski2003], assuming one-eighth symmetry, was modelled (Figure 4–34). The benchmark was designed to provide the framework to asses the ability of modern reactor kinetic codes to predict the transient response of a core partially loaded with MOX fuel. A rod ejection may occur as a consequence of the rupture of the drive mechanism casing located on the reactor pressure vessel. This event is particular interesting for MOX fuelled cores since the delayed neutron fraction in MOX fuel is significantly smaller than in UO_2 cores.

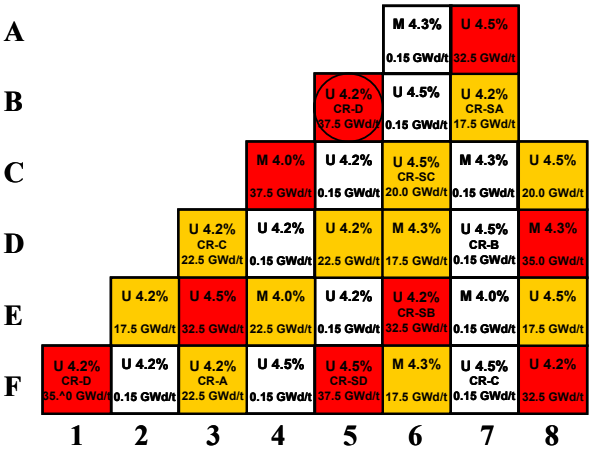


Figure 4–34 Geometrical configuration of the PWR core with one-eighth symmetry.

The core has uniform fuel composition in axial direction. The original benchmark core considers axial reflectors (top and bottom) with the same width as the fuel assembly pitch and in the radial direction, the core is surrounded by a single row of reflector assemblies. In DYNSUB, since the use of reflector has not been implemented, axial boundary conditions by means of Albedos where chosen. The initial operational conditions used in this case are presented in Table 4-XI.

Table 4-XI Operational conditions for eighth of core

Operational condition	**value**
Number of assemblies	31
Power (MWth)	4.456×10^{-4} (10^{-4}% rated power)
Inlet Temperature (°C)	287.0
Inlet Pressure (MPa)	15.5
Pressure drop over the core (kPa)	125.0
Active Flow (kg/sec)	2545.69
Fuel lattice, fuel rods per assembly	17 x 17, 264
Active length (cm)	365.76
Assembly pitch (cm)	21.41
Pin pitch (cm)	1.26
Axial boundary conditions	$\alpha_{AX} = 0.5$
Radial boundary conditions	$\alpha_{R} = 0.5$
Number of axial nodes	10 (36.576 cm)
Boron concentration (ppm)	1605.0

4.5.3.2 Steady state results for the Case 2

Similarly to the minicore, the rod worth of the assembly B5 was calculated with DYN3D-SP3 and DYNSUB. The average beta effective value ($\overline{\beta_{eff}}$) for this core configuration (eighth of core) is 559.3031 pcm and it was obtained by means of equation (4.37).

The results are presented in Table 4-XII. Differences in k_{eff} up to 2.7 pcm for the ARO configuration and up to 3.2 pcm for the ARI configuration were found, taken as a reference the DYN3D-SP3 calculation. These small differences, in comparison with the minicore, are due to the HZP condition of the actual configuration. Therefore, it is clear that the thermal-hydraulic conditions do not play an important roll at the steady state.

The worth of the rod for DYNSUB (793.22 pcm) and the one of DYN3D-SP3 (792.67 pcm) are greater than the beta effective value. Thereby a fast reactivity insertion bringing the minicore to a super prompt critical state is expected.

Table 4-XII k_{eff}, rod worth and reactivity (ρ) comparison between DYN3D-SP3 and DYNSUB for the eighth of core.

	DYN3D-SP3	DYNSUB	Deviation (pcm)
k_{eff}^{ARO}	0.998378	0.998351	2.7
k_{eff}^{ARI}	0.990539	0.990507	3.2
Rod worth (pcm)	792.67	793.22	-0.55
ρ **(\$)**	1.4172	1.4182	0.07%

Figure 4–35 shows for the ARI configuration, the convergence of k_{eff} (top-left), and how the deviations of the convergence parameters behave in the iteration process.

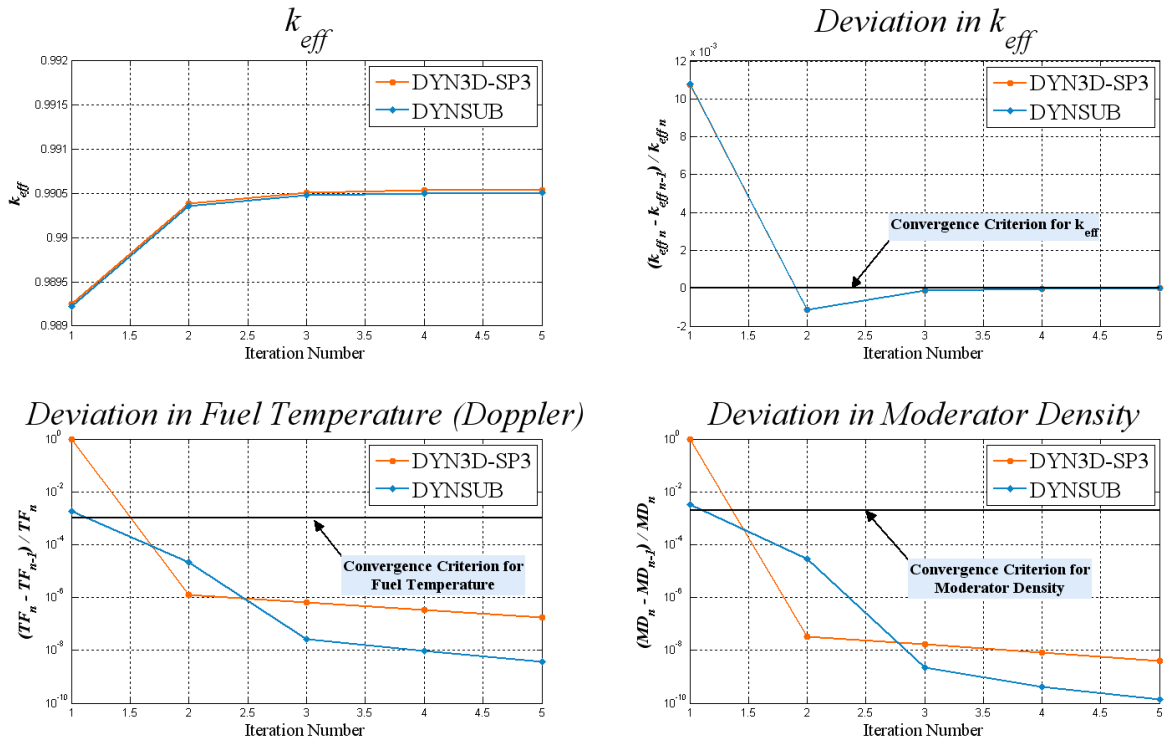


Figure 4–35 k_{eff} and deviations of the convergence criteria in the steady state calculation ARI of Case 2.

In this case both calculations got convergence in just 5 iterations. It can be seen that the iterative process is strongly related with the neutronics behaviour (deviations in k_{eff}) and not due to the thermal-hydraulics. This is explained by the fact that the core is at HZP state and thus, the thermal-hydraulics conditions do not play an important roll at steady state.

The axial power distribution for the ARI configuration shows very good agreement between the two codes. The two curves in Figure 4–36 reached the peak in the same axial position and only small differences between the two profiles can be distinguished.

Furthermore, in Figure 4–37, the percentage difference between DYN3D-SP3 and DYNSUB, for the radial power distribution in the axial layer 5, is shown. Just small differences ranging from -0.08 % in the innermost region of the core (centre of the core) until 0.04 % in the outer region of the core appeared.

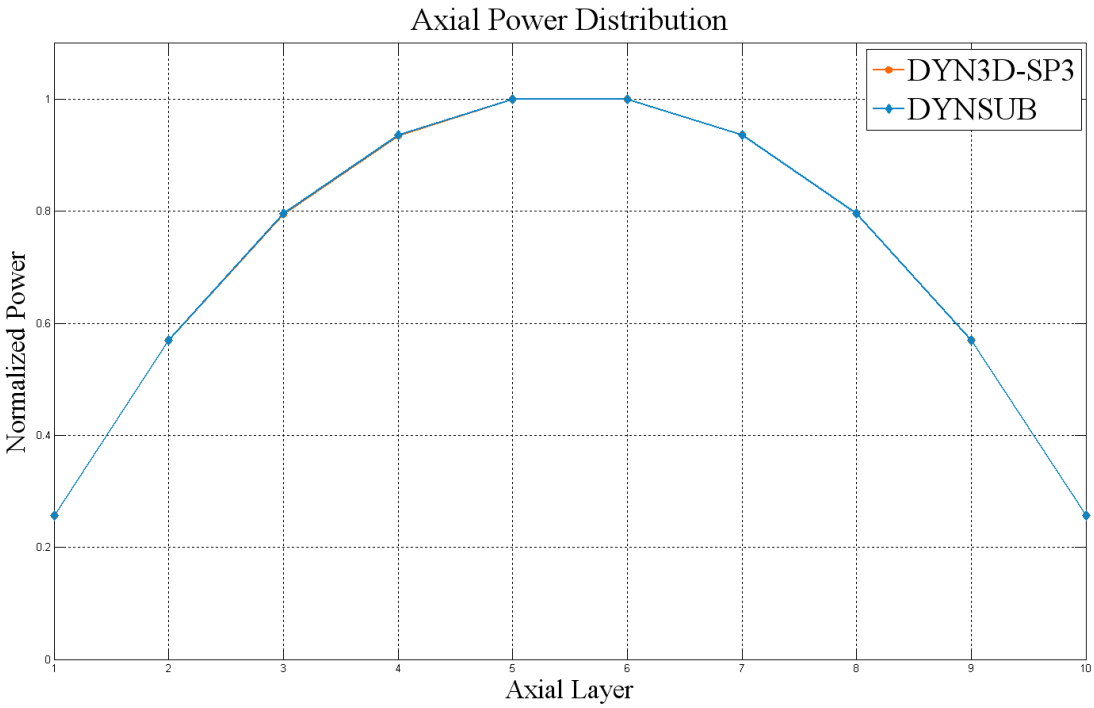


Figure 4–36 Comparison of the normalized axial power profile for the ARI configuration of the eighth of core.

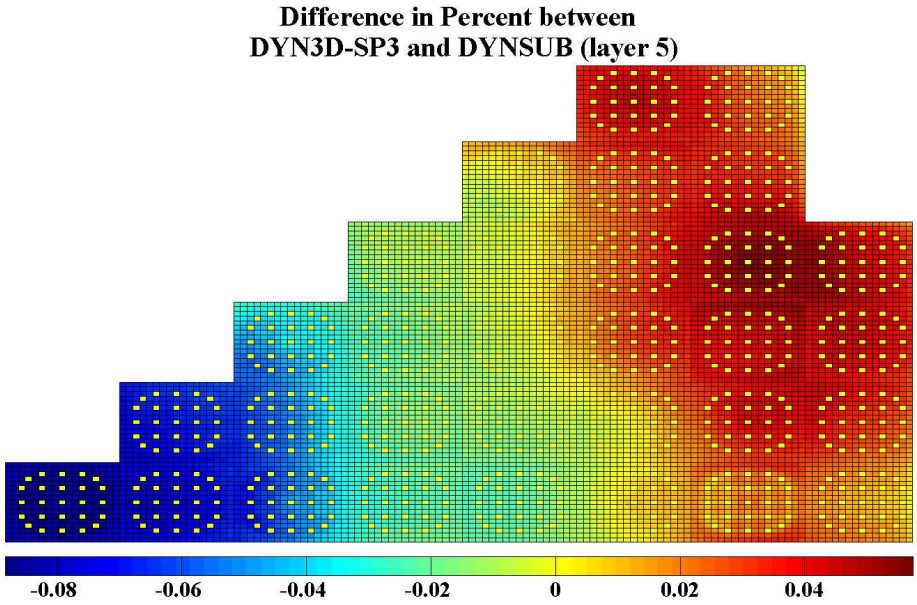


Figure 4–37 Pin power distribution, difference in percentage between DYN3D-SP3 and DYNSUB for the hottest layer (layer 5).

4.5.3.3 Transient results for the Case 2.

In the same way that it was done for the minicore, several assumptions were made in order to be able to perform comparisons between DYN3D-SP3 and DYNSUB. The heat conduction properties in the fuel pin were also fixed in agreement with Table 4-V (previously defined for the Case 1).

For the temporal scheme, details about the temporal parameters used by the two codes are presented in Table 4-XIII.

Table 4-XIII Temporal parameters for the calculation of Case 2.

Temporal parameters	**value**
Model	FPI
Time step (milliseconds)	2
Maximal number of internal iterations at each time step	10
Convergence criterion for power deviation	5.0×10^{-3}

The global behaviour of the power as well as for the Doppler temperature and moderator density and temperature are presented as a function of time in Figure 4–38 .

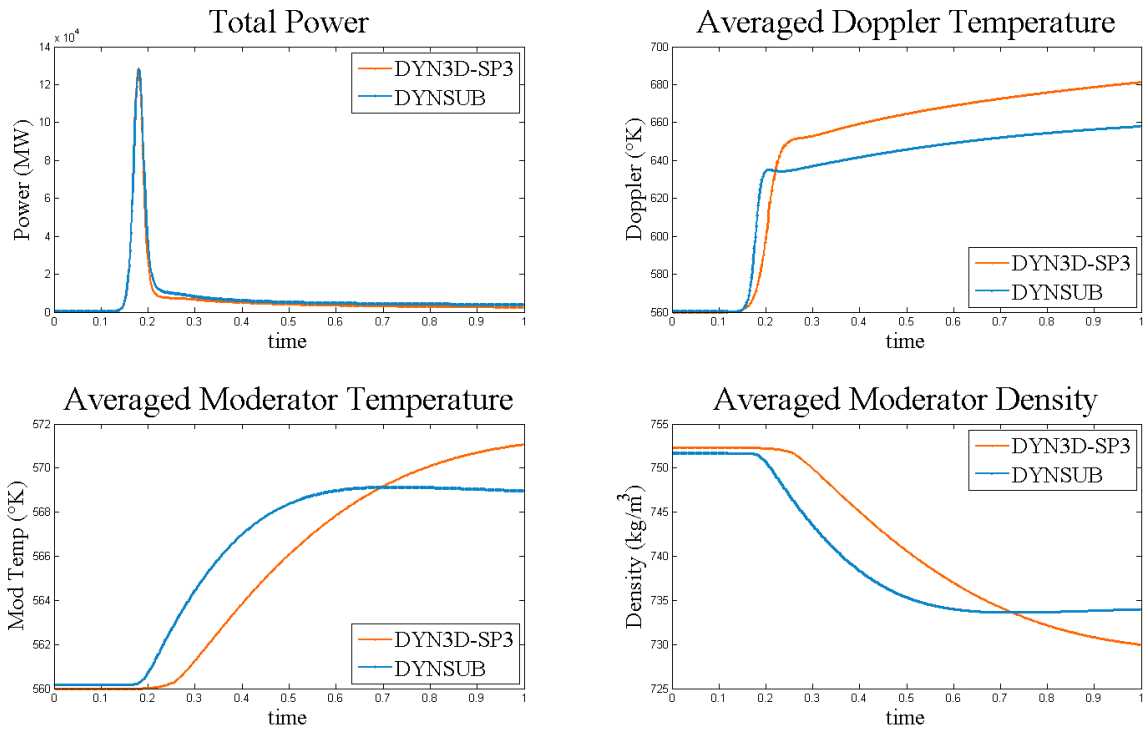


Figure 4–38 Global behaviour of total power and thermal-hydraulics parameters during the transient.

In Table 4-XIV, details about the results are given. The deviations were calculated taken DYN3D-SP3 calculation as reference.

Table 4-XIV Comparison of DYN3D-SP3 and DYNSUB for the Case 2.

	DYN3D-SP3	DYNSUB	Deviation
Power peak (MW)	1.28523×10^5	1.27312×10^5	-0.94 %
Peak time (milliseconds)	180	180	0.0
Asymptotic power at the end of the transient (MW)	2.55044×10^3	3.74688×10^3	46.9%
NK-TH iterations	524	524	0
NK internal iterations	108209	103242	-4967
Relative CPU time	1 (9556 min)	0.73 (7006 min)	27%

Similarly to the Case 1, a power peak slightly smaller was predicted with DYNSUB (-0.94%). The averaged Doppler temperature in the first 200 milliseconds increases faster with DYNSUB than with DYN3D-SP3 (top-right plot of Figure 4–38). For this reason, the moderator temperature predicted with DYNSUB start also increasing before the one predicted with DYN3D-SP3 (bottom-left plot of Figure 4–38). After the first 200 milliseconds, the Doppler temperature predicted by DYN3D-SP3 becomes greater than the one predicted with DYNSUB, thus this negative reactivity brings the power below the one predicted by DYNSUB.

Like in the case of the minicore (Case 1), the discrepancies arising after the power peak and in the asymptotic part are due to the different heat transfer and thermal-hydraulics models and correlations used in SUBCHANFLOW and FLOCAL. These differences resulted in a deviation of 46.9% in the power at the asymptotic part of the transient (from 250 until 1000 milliseconds).

Figure 4–39 shows the number of time step internal iterations at the moment of the peak.

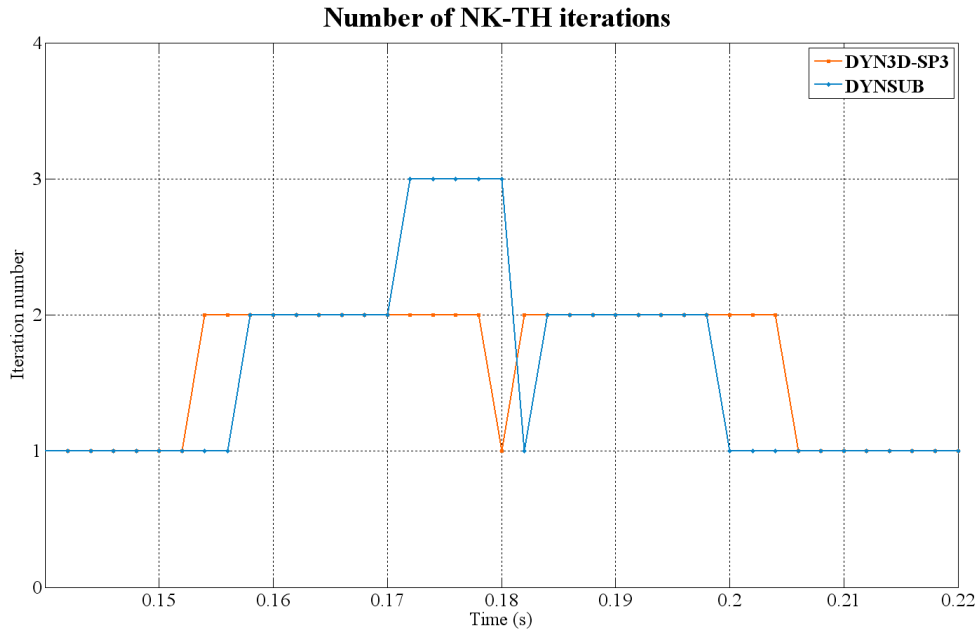


Figure 4–39 NK-TH iterations during the power peak time interval.

Important is to notice the impact of SUBCHANFLOW in the iteration process. Although the same number of NK-TH iterations was needed in the two cases (524 iterations). The computational time was 27% larger in the case of DYN3D-SP3. The reason is that DYNSUB needed less internal iterations for the neutronic calculation (103242) than DYN3D-SP3 (108209) specially in the asymptotic region in which the power has already stabilized in its asymptotic value (Figure 4–40). Similarly to the Case 1, in which a HFP state is analyzed, it can be seen that the 3D thermalhydraulic influence in DYNSUB (by means of SUBCHANFLOW) has an impact in the CPU time. Further comparisons can be found in [GomezA2011].

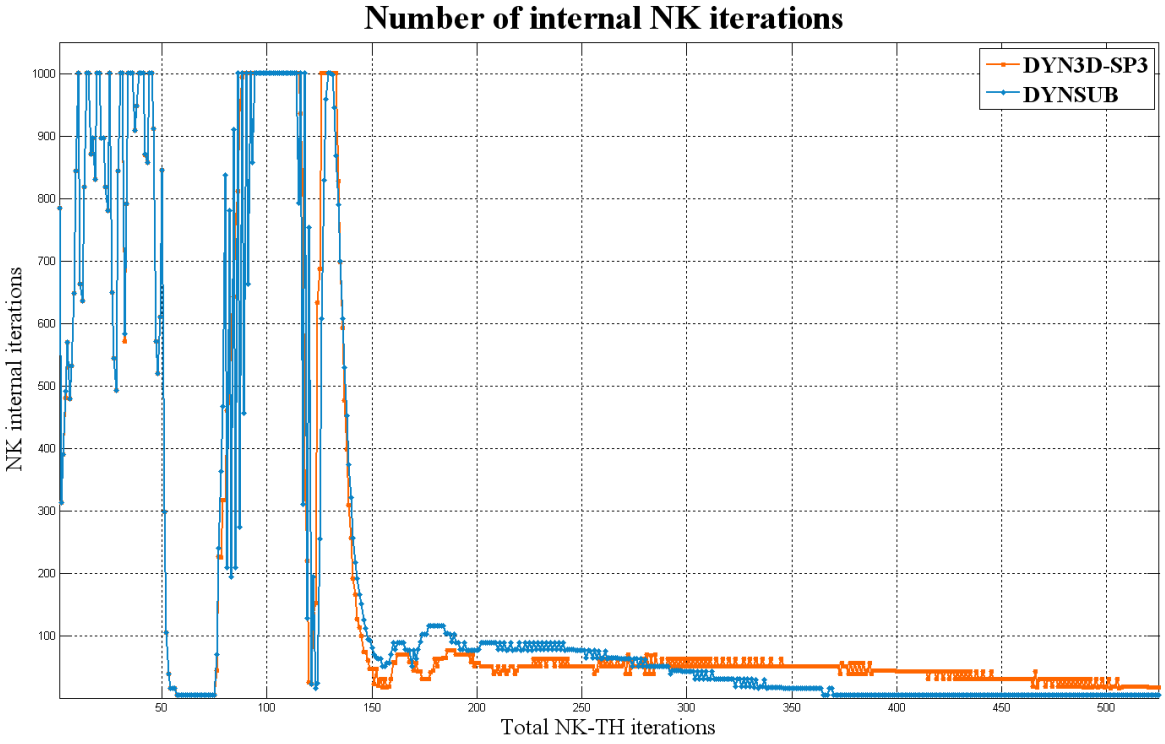


Figure 4–40 NK internal iterations in every NK-TH iteration.

For the prediction of the hottest point in the reactor configuration, discrepancies between the two codes were found. In the top-left graph of the Figure 4–41, the maximal fuel temperature (centreline) as a function of time for the two calculations is presented. Whereas DYN3D-SP3 calculates the position of the hottest fuel assembly (assembly B6 at the beginning and C5 at the end), DYNSUB is able to predict the rod with the maximal (rod 6989) and the minimal (rod 7225) fuel temperature within the hottest fuel assembly (assembly C7). DYNSUB predicted a maximal fuel temperature up to 62 K larger than DYN3D-SP3. The axial level with the hottest temperature is the same after the power peak with the two codes (top-right Figure 4–41); however the radial location of the hottest fuel assembly did not match. DYNSUB predicted always the same fuel assembly (C7), whereas DYN3D-SP3 moved from assembly D6 until C5 (bottom-left Figure 4–41). Finally, with DYNSUB is also possible to know exactly the hottest rod in the hottest assembly (bottom-right plot of Figure 4–41).

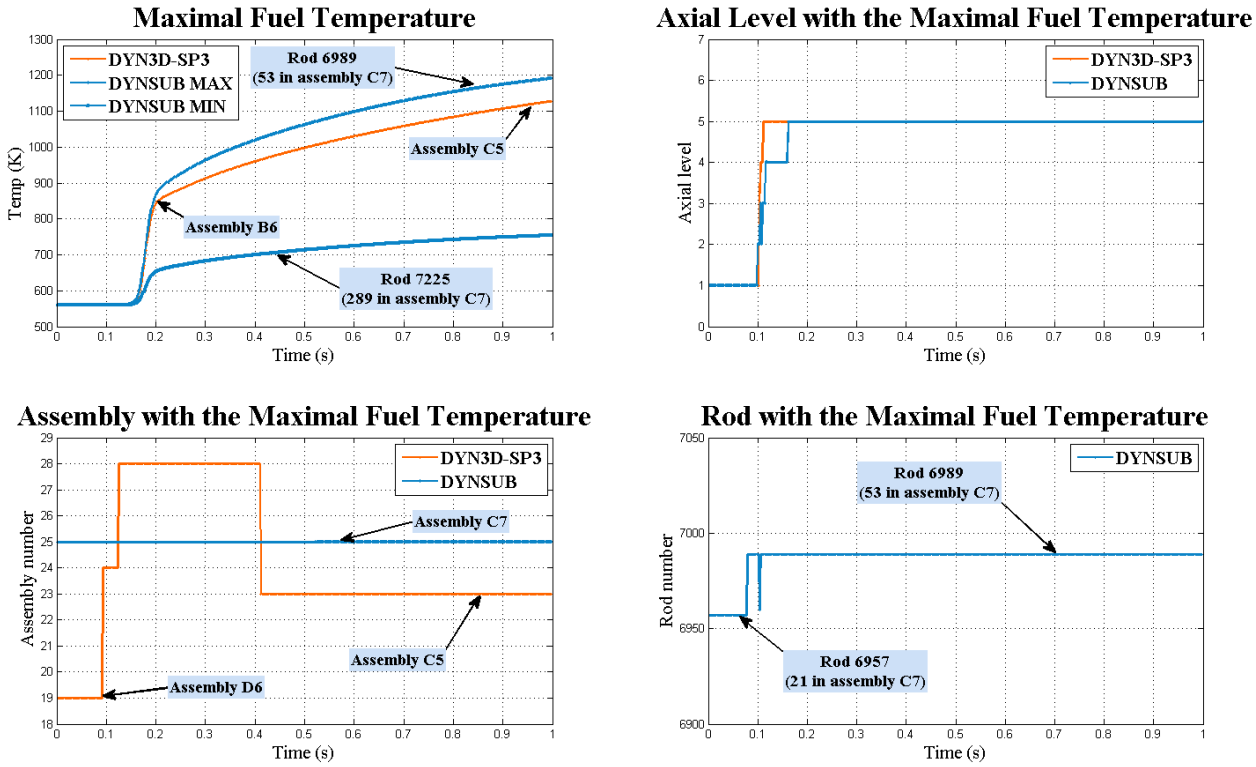


Figure 4–41 Location of the hottest point in the reactor at assembly base and pin base with DYN3D-SP3 and DYNSUB respectively.

Figure 4–42 shows the fuel centreline, the clad surface and the moderator axial temperature profiles at the peak time (Figure 4–42 (a), (b) and (c)) and at the end of the transient (Figure 4–42 (d), (e) and (f)) respectively.

At the peak time, differences up to 15 K in the centreline fuel temperature and up to 4.3 K in the cladding temperature are observed in all cases with an overestimation of DYNSUB. The differences in the fuel centreline temperature, in opposite to the Case 1, become larger as the transient progresses (62 K at the end of the transient). However, the cladding temperature predicted by DYNSUB at the end of the transient is 5 K smaller.

The differences in both cases are due to the thermal-hydraulic models. In DYNSUB a faster increase of global Doppler and moderator temperature can be observed in the first milliseconds of transient (Figure 4–38). However at the end of the transient, the global temperatures predicted by DYN3D-SP3 are larger and although the fuel centreline temperature for the hottest rod remains bigger than the hottest assembly is important to notice that the difference between the hottest assembly temperature predicted by DYN3D-SP3 and the coldest rod in this assembly predicted by DYNSUB increased from approximately 100 K at peak time until 350 K at end time. Thus, at the end, although the temperature of the hottest rod (predicted by DYNSUB) remains over the hottest assembly, the average core centreline temperature is smaller.

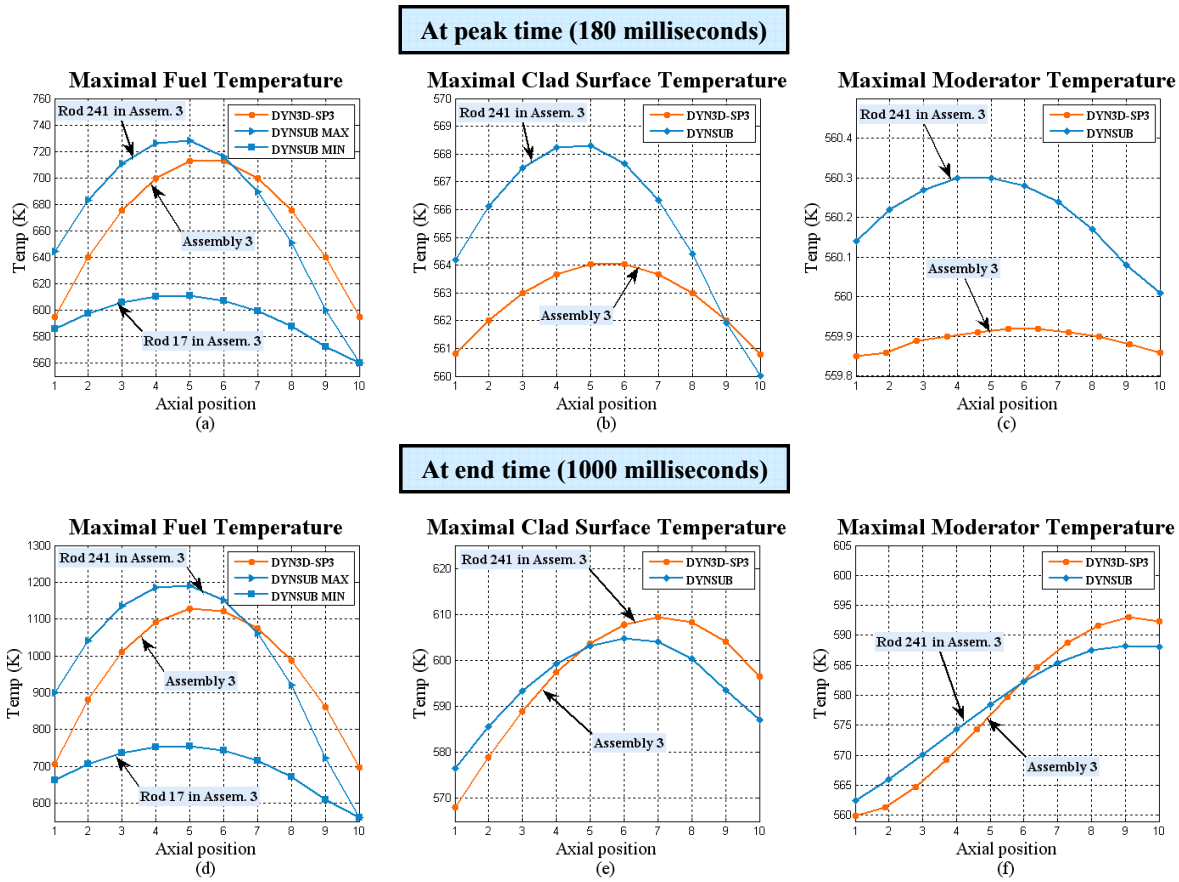


Figure 4–42 Axial distribution for the maximum fuel centreline, the clad surface and the moderator temperatures at peak time (180 milliseconds) and at the end of the transient (1000 milliseconds).

5 Conclusions

Two different multiphysics and multiscale approaches were investigated in this dissertation: the extension of the pin power reconstruction method of DYN3D, and a pin level coupling scheme between DYN3D-SP3 and SUBCHANFLOW.

5.1 Conclusions for the extension of the pin power reconstruction capability of DYN3D

An extension of the pin power reconstruction method of the reactor dynamics code DYN3D was done. This extension, is able to calculate pin power distribution not only in one assembly (useful for hot channel analysis) but also in several fuel assemblies (a region of interest) or even in the whole core. The analysis of a region of interest will allow the coupling with subchannel codes able to consider local refinements in a non-conform geometry. One of the main advantages of considering regions and not only a hot channel is that the cross flow inside the region of interest can be taken into account more accurately (specially useful in PWR analysis). Such kinds of coupling improvements have been investigated and are under development as a part of the new European Nuclear Reactor Simulation platform NURESIM. In order to illustrate the capabilities of the NURESIM platform a detailed description of the integration of this extended version of DYN3D in the European Nuclear Reactor Simulation Platform (NURESIM) was also presented. This integration is a step forward in the direction of two-level coupling with a subchannel code being one of the major objectives of NURESIM platform.

In order to test the extension of DYN3D integrated in the NURESIM platform, three test cases were presented:

- A control rod ejection in a 5 x 5 minicore: The use of pin power reconstruction (refinement) was performed in all the fuel assemblies. The effect of the control rod ejection could be clearly observed. The more detailed pin power distribution inside the assembly, will allow a better and more detailed prediction of the safety parameters by means of a hot region calculation in which the pin power distribution can be given to a subchannel code for the calculation of local safety parameters.

- A boron dilution transient in a PWR core: In order to test the extension in a more practical application, a slug of deborated water transported into the reactor pressure vessel of a full PWR core was modelled. A local refinement in a “region of interest” was considered. A detailed pin power distribution was calculated for this region avoiding problems that can arise while storing and manipulating the output data.

- A steady state full WWER core: The DYN3D extension in hexagonal geometry was tested. The pin power distribution of the whole WWER core (211 assemblies) was calculated.

.

5.2 Conclusions for the development of an advanced coupling code based on DYN3D-SP3 and SUBCHANFLOW

DYNSUB is a new best estimate coupled system, based in well validated stand alone codes. Due to the replacement of the 1D- FLOCAL model by the 3D SUBCHANFLOW code, DYNSUB avoids the loss of information due to averaging at assembly level, making possible to predict safety parameters more accurately and without the use of hot channel factors.

With the LINUX version of the code it is possible to solve practical problems like the eighth of core involving a large amount of nodes.

In the next subsections remarks and conclusions about the studied cases will be presented.

5.2.1 Case 1: minicore 2 x 2

In order to verify the implementation, a minicore 2 x 2 was extensively investigated. Steady state at hot full power (HFP), transient (control rod ejection) and different temporal schemes (explicit and FPI) were analyzed. Code-to-code comparisons with DYN3D-SP3 standalone were done.

5.2.1.1 Case 1: Steady state

The use of a more refined thermal-hydraulic model in DYNSUB resulted in deviation in k_{eff} up to 88.2 pcm Table 4-IV and differences in the pin power distribution ranging from -1% until 1.5% in the hottest axial layer Figure 4–23. It was shown that these deviations are due to the impact of the more detailed thermal-hydraulic model. Whereas DYN3D-SP3 predicts just one thermal-hydraulic parameter per fuel assembly Figure 4–24; DYNSUB is able to give a thermal-hydraulic distribution Figure 4–25. This detailed calculation implies differences in the prediction of the maximal fuel centreline temperature and moderator density and temperature which impact the updating of cross sections mainly in the hottest regions. Differences up to 125 K in the maximal fuel centreline temperature between DYNSUB (1271 K) and DYN3D-SP3 (1145.6) were found.

Furthermore, it was showed that the use of SUBCHANFLOW in DYNSUB has a great impact in the iterative process. The convergence was achieved in just 10 NK-TH iterations in contrast with the 40 iterations needed by DYN3D-SP3 (Figure 4–20).

5.2.1.2 Case 1: Transient

In the transient analyzed, the very fast control rod ejection was correctly reproduced by DYNSUB. The global behaviour of the power showed very good agreement in the first 100 milliseconds of the transient, where just the Doppler Effect plays an important roll (Figure 4–26). The power peak predicted by DYNSUB was 0.46% smaller than the one of DYN3D-SP3 (Table 4-VII). As soon as the heat is transported to the moderator (after the power peak is achieved), the different heat transfer models together with the degree of detail in the thermal-hydraulics models (SUBCHANFLOW and FLOCAL), start affecting the global behaviour and different asymptotic curves are found (Figure 4–26 and Figure 4–27).

The evolution of the maximal fuel temperature showed also important differences Figure 4–28. The maximal fuel temperature in the hottest rod calculated with DYNSUB was always bigger than the maximal fuel temperature of the hottest assembly predicted by DYN3D-SP3. Differences up to 152 K at the peak time were reduced until 60 K at the end of the transient. For the maximal cladding temperature and coolant temperature a similar behaviour is predicted (Figure 4–30). Such differences can be very significant while approaching to the operational limits.

5.2.1.3 Case 1: Temporal schemes

The two time schemes implemented in DYNSUB (explicit and Fixed Point Iteration, FPI) were compared. It was shown that the effect of the FPI can play a very important roll in the accuracy of the calculations. The FPI scheme, as an approximation to an implicit scheme, allows moving from one time step to the other with converged solution. It also allows the use of bigger time steps without compromising the accuracy of the results Table 4-X.

From the four cases compared with the reference solution, the FPI_1 calculation had, in general, the better performance. The errors can be even decreased by choosing more strict power deviation criterion for the time step internal iteration. However, a stricter criterion will increase the number of internal iterations and therefore the CPU time. The results using FPI are very promising and represent a very good option in order to optimize computational times without loosing accuracy. Further comparisons with the automatic time step algorithms already implemented in DYN3D-SP3 but not tested with DYNSUB may be also done.

5.2.2 Case 2: PWR core assuming one-eighth symmetry

For a more realistic application, a PWR core with one-eighth geometry was modelled. It also included an increase in complexity due to the use of MOX fuel elements and different burn-up states. A steady state from hot zero power (HZP) and a fast control rod ejection were calculated. Code-to-code comparisons between DYNSUB and DYN3D-SP3 were also done.

5.2.2.1 Case 2: Steady state

Due to the HZP condition, the steady state reduces almost to a merely neutronics problem. The thermal-hydraulics does not have a great impact in the steady state calculation. Thereby, small differences on the keff are reported in Table 4-XII. The NK-TH iterative process got convergence in just 5 iterations in both codes. The axial power distribution (Figure 4–36) and the radial pin power distribution for the hottest layer (Figure 4–37) showed also very good agreement.

5.2.2.2 Case 2: Transient

In the transient case, the prediction of the power peak by means of DYNSUB showed very good agreement comparing with DYN3D-SP3 (Figure 4–38). The power peak was reached at the same time and with a deviation smaller than 1% Table 4-XIV. The use of SUBCHANFLOW instead of FLOCAL had a great impact in the computational time. The more detailed

description of the thermal-hydraulics parameters (sub-channel based) allowed DYNSUB to have a faster convergence process (Figure 4–40) leading to considerable CPU time savings.

The different thermal-hydraulic models played also an important roll after the power peak. In DYNSUB, a faster increase of temperatures was predicted in the first milliseconds of the transient. The more detailed information, that it is possible to obtain by means of DYNSUB, makes possible to see that although the global fuel centreline temperature at the end of the transient predicted by DYNSUB are smaller than the ones predicted by DYN3D-SP3, in the local level is the opposite (Figure 4–42).

5.2.3 General remarks

The results obtained with DYNSUB presented a good agreement in the code to code comparisons. The degree of detailed obtained with DYNSUB may allow a more accurate estimation of local safety parameters. Thus, the use of hot factors may be avoided in the analysis of safety margins. Further verifications and validations must be however done and the impact of the thermal-hydraulics models and correlations and the influence of cross flow in the calculation have to be assessed by means of uncertainty and sensitivity evaluations.

6 Outlook

The future work is also divided in the two different multiphysics and multiscale approaches presented in this dissertation.

6.1 Future work related with the Pin Power Reconstruction Extension of DYN3D

For the DYN3D pin power reconstruction extension the clearest objective will be the coupling with subchannel codes able to consider non-conform geometries. However, a formal and extensive validation of the pin power reconstruction method must be done. For this purpose, code to code comparisons with advanced lattice codes or 3D transport codes (S_N or SP_3 methods) are required.

Although the foreseen coupling seems to be promising, it is also true that the coupling in a pin by pin scale is just in one direction. The pin power distribution can be allocated in the subchannel code; however, the thermal-hydraulic feedback must be averaged and passed to the diffusion solver as a nodal base feedback. Nevertheless, previous works have shown that an approach to a two way coupling scheme can be done with a local feedback coming from an appropriate set of form functions depending on the thermal-hydraulic branches. Therefore, methodologies for an appropriate calculation of form functions must be established.

Finally, the use of the neutronics diffusion codes with pin power reconstruction methods as an accelerator technique integrated in a pin based two way coupled system may be interesting to explore. The fast pin power reconstruction method can provide initial power shape to more complex systems, for instance, to a MCNP/subchannel-code coupled system or to DYNSUB. Appropriate shapes can accelerate the convergence process of such coupled codes.

6.2 Future work related with the coupling code DYNSUB

The list of improvements that can be done in DYNSUB could be a never ending list. There are always new techniques and new ideas that can be implemented in a numerical simulator. A reduced and finite list is presented hereafter.

1. A further verification of DYNSUB is foreseen. The two benchmark problems here presented covered just the part of adiabatic transients leaded by neutronics reactivity changes. Further analysis and modifications of the code have to be done for being able to analyze transients relaying in thermal-hydraulics changes, like for instance MSLB.

2. A more general mesh mapping scheme has to be implemented in DYNSUB. Although the radial mapping is somehow flexible, the axial mapping is based in a tight scheme; the same axial nodes have to be described in both codes.

3. The treatment of reflectors in DYNSUB is also foreseen. Until now, the axial reflectors have been simulated by means of neutronics boundary conditions (axial albedos).

In order to be able to do a better verification and a subsequent validation the inclusion of reflectors must be programmed.

4. The dynamic development of SUBCHANFLOW and the increasing user's group feedback results in a rapid move from one version to other. In this dissertation the last version of SUBCHANFLOW used was 1.8. Version 2.0 has been extended in order to calculate more local safety parameters like DNB ratios. DYNSUB has to move at least to this version in order to fulfil the objectives of the coupled system: "best estimate code for the evaluation of local safety parameters"

5. Dynamic modifications to the SUBCHANFLOW Pre-processor as standalone and as a subroutine have to be done in order to address the requirements of the new SUBCHANFLOW versions. A GUI (Graphical User Interface) could also be scheduled.

6. The two-way-coupling of DYNSUB requires great amounts of memory and computational power. Practical applications, like a whole core representation with reflectors and several axial layers, may be not possible to treat. Optimization methods or a possible parallelization of the code have to be explored in order to tackle these obstacles.

7. The treatment of Burn-up capability at pin level is also an interesting and important issue in the modern coupled systems that can be implemented.

8. Integration of DYNSUB in the SALOME platform is also foreseen.

9. The user friendly interfaces with pre and post processing capabilities as well as for executing the code have an increasing use in the modern technological era. Thus, further developments and improvements of the DYNSUB Graphical User Interface are also planned tasks.

10. The evolution of all codes is only possible if the people use the code and find "bugs" or need to model a new design that it is not supported by the actual version of the code. Therefore, the establishment of a working group for further validation and improvement of DYNSUB results an essential task.

7 References

[ANSYS2009] ANSYS CFX-Solver Theory Guide, Release 12.0, ANSYS, Inc. April 2009.

[Azmy1996] Y. Y. Azmy, The Three-Dimensional, Discrete Ordinates Neutral Particle Transport Code TORT: An Overview, OECD/NEA meeting on 3D Deterministic Radiation Transport Computer Programs, Feature, Applications and Perspectives, Paris France, December 2 – 3, 1996.

[Bahadir2006] Tamer Bahadir and Sten-Orjan Lindahl, SIMULATE-4 Pin power calculations, PHYSOR-2006, ANS Topical meeting on Reactor Physics, Vancouver, Canada, September 10-14, 2006.

[Bandini1990] B. R. Bandini, A Three Dimensional Transient Neutronics Routine for the TRAC-PF1 Reactor Thermal Hydraulic Computer Code, PhD Thesis, The Pennsylvania State University,Department of Nuclear Engineering 1990.

[Barber1998] Barber, D.A., Downar, T.J., and Wang, W., Final Completion Report for the Coup led RELAP5/PARCS, PU/NE-98-31, Purdue University, 1998.

[Barre1990] F. Barre and M. Bernard, The CATHARE code strategy and assessment, Nuclear Engineering and Design 124 (1990) 257-284.

[Barthel2009] Barthel V., Vandroux S.: Validation of the Trio_U code - October 2009, CEA Grenoble, DEN/CAD/DER/SSTH/LDAL/NT/2009-008/A 2009.

[Basile1999] D. Basile, R. Chierici, M. Beghi, E. Salina and E. Brega, COBRA-EN, an Updated Version of the COBRA-3C/MIT Code for Thermal-Hydraulic Transient Analysis of Light Water Reactor Fuel Assemblies and Cores, Report 1010/1, ENEL-CRTN Compartimento di Milano, 1999.

[Beam1999]Tara M. Beam, Kostadin N. Ivanov, Anthony J. Baratta, Herbert Finneman, Nodal kinetics model upgrade in the Penn State coupled TRAC/NEM codes, Annals of Nuclear Energy 26 (1999) 1205-1219.

[Beazley1997] David M. Beazley, SWIG, A Simplified Wrapper and Interface Generator: Users Manual, Department of Computer Science, University of Utah, 1997.

[Beckert2008] Carsten Beckert, Ulrich Grundmann, Entwicklung einer Transportnährung für das reaktordynamische Rechenprogramm DYN3D, Forschungzentrum Dresden-Rossendorf, Institut für Sicherheitsforschung, Berichts-Nr. FZD-497, Juni 2008.

[Beckert22008] Beckert, C., Grundmann, U. Development and verification of a nodal approach for solving the multigroup SP3 equations, Annals of Nuclear Energy 35 (2008) 75-86.

[Bergeaud] V. Bergeaud, M. Tajchman, Application of the SALOME software architecture to Nuclear Reactor Research, SpringSim Vol. 1, p. 383-387.

[Berkhan2011] Ana Kate Cecilia Berkhan Cazón, Qualifizierung des Thermohydraulik-Unterkanalcodes SUBCHANFLOW für die Anwendung auf Leichtwasserreaktoren, Diplomarbeit at KIT-INR, 2011.

[Berna1997] G. A. Berna, C. E. Beyer, K. L. Davis, D. D. Lanning, FRAPCON-3: A computer code for the calculation of Steady-state, Thermal-Mechanical behaviour of oxide fuel rods for high burnup, U.S. NRC NUREG/CR-6534, Vol. 2, December 1997

[Boer1992] R. Boer, H. Finnemann, Fast analytical flux reconstruction method for nodal space-time nuclear reactor analysis, Annals of Nuclear Energy 19 (1992) 617-628.

[Bouhamou2005] N. Bouhamou, Salome Training Session, June 2005.

[Bousbia2007] Anis Bousbia-Salah, Francesco D'Auria, Use of coupled code technique for Best Estimate safety analysis of nuclear power plants, Progress in Nuclear Energy 49 (2007) 1-13.

[Bowman2007] S. M. Bowman, Overview of the SCALE code system, American Nuclear Society 2007 Winter Meeting, Washington, DC, November 11–15, 2007.

[Brantley2000] Patrick S. Brantley, Edward W. Larsen, The Simplified P3 Approximation, Nuclear Science and Engineering 134 (2000), 1-21.

[Brun2009] E. Brun, Y. Cobigo (CS), E. Dumonteil, F.-X. Hugot, N. Huot, C. Jouanne, Y.-K. Lee, F. Malvagi, A. Mazzolo, O. Petit, J.-C. Trama, T. Visonneau (CS), TRIP-OLI-4 Version 6 User Guide, CEA REPORT SERMA/LTSD/RT/08-4557/A, May 2009.

[Cacuci2000] D.G. Cacuci, Dimensionally adaptive dynamic switching and adjoint sensitivity analysis: new features of the RELAP5/PANBOX/COBRA code system for reactor safety transients. Nuclear Engineering and Design 202 (2000) 325-338.

[Cacuci2006] Cacuci D. G., J. M. Aragones, D. Bestion, P. Coddington, L. Dada, and C. Chauliac, NURESIM: A European Platform for Nuclear Reactor Simulation, Proceedings of FISA, Conference on EU Research and Training in Reactor Systems, Luxembourg, 2006.

[Chauliac2008] C. Chauliac, Overview on the Collaborative Project NURISP, Presentation at the NURESIM General Seminar, Madrid, Nov. 27-28, 2008.

[Christienne2010] M. Christienne, M. Avramova, Y. Périn, A. Seubert, Coupled TORT-TD/CTF capability for high-fidelity LWR core calculations, PHYSOR 2010, Pittsburg, Pennsylvania, USA, May 9 -14, 2010.

[Collins2008] Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, Version Control with Subversion: For Subversion 1.5: (Compiled from r3305), 2008.

[CRISSUEV1] Neutronics/Thermal-hydraulics Coupling in LWR Technology, Vol. 1, CRISSUE-S I WP1. NEA No. 4452, ISBN 92-64-02083-7, 2004.

[CRISSUEV2] Neutronics/Thermal-hydraulics Coupling in LWR Technology: State-of-the-art Report (REAC-SOAR), Vol. 2, CRISSUE-S I WP2, NEA No. 5436, ISBN 92-64-02084-5, 2004.

[Crouzet2] N. Crouzet, hxx2salome: a Salome component generator.

[Crouzet2009] N. Crouzet, Code integration in Salomé, NURISP General Seminar, Dresden, Germany, November 30, 2009.

[Cuervo2007] Diana Cuervo Gómez, Análisis termohidráulico de núcleos de PWR con modelización de flujo bifásico para acoplamiento con la neutrónica, Ph.D Thesis, Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingenieros Navales, Departamento de Ingeniería Nuclear, Febrero 2007.

[Cunningham] M. E. Cunningham, C. E. Beyer, P.G. Medvedev, G. A. Berna, FRAPTRAN: A computer code for the Transient analysis of oxide fuel rods, U.S. NRC NUREG/CR-6739, Vol. 1.

[Downar2006] T.J. Downar, Y. Xu, V. Seker, PARCS v2.7U.S.NRC Core Neutronics Simulator, User Manual. University of Purdue, November 2006.

[Federici] E. Federici, F. Lamare, V. Bessiron, J. Papin, The SCANAIR code version 3.2: Main features and status of qualification, Institut de Protection et de Surete Nucleaire (IPSN), Cadarache, France.

[Fischer1981] Hans Dieter Fischer and Herbert Finnemann, The nodal integration method - A diverse solver for neutron diffusion problems, Atomkernenergie Kerntechnik Bd. 39, 1981.

[Gan2003] J. Gan, Y. Xu, T. Downar, A Matrix-Free Newton method for coupled neutronics thermal-hydraulics reactor analysis, Nuclear Mathematical and Computational Science, American Nuclear Society, Gatlinburg, Tennessee, April, 2003.

[Gelbard1960] E. M. Gelbard, Application of Spherical Harmonics Methods to Reactor Problems, WAPD-BT-20, Bettis Atomic Power Laboratory, 1960.

[Glasstone1958] Samuel Glasstone and Milton C. Edlund, The elements of Nuclear Reactor Theory, D. Van Nostrand Company, Inc., Princeton, New Jersey, August 1958.

[Glasstone1970] George I. Bell, Samuel Glasstone, Nuclear Reactor Theory, D. Van Nostrand Company, Inc., Princeton, New York, 1970.

[GNUAutotools] http://inti.sourceforge.net/tutorial/libinti/autotoolsproject.html , Init Tutorial: Building a GNU Autotools Project.

[Godbronn2006] P.Godbronn, E. Fayolle, N. Bouhamou, J. Roy and N. Crouzet, MEDMEM user's guide, CEA, Rapport DM2S, July 2006.

[GomezA2009] A. Gomez, Coupling of Neutronics and Thermal-hydraulics in the NURESIM platform, Presentation at the IKET Winter Seminar on Energy Technologies, Schruns, Austria, 15-18 February, 2009.

[GomezA2010] Armando M. Gomez Torres, Victor H. Sanchez Espinoza, Sören Kliem, Andre Gommlich and Ulrich Rohde, Integration of DYN3D inside the NURESIM Platform, 17th Pacific Basin Nuclear Conference, Cancun Q.R. Mexico, October, 2010.

[GomezA2011] Armando Gomez Torres, Victor Sanchez Espinoza, Uwe Imke and Rafael Macian Juan, Pin level neutronic-thermal-hydraulic two-way-coupling using DYN3D-SP3 and SUBCHANFLOW, International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Rio de Janeiro, Brazil, 2011.

[GomezR2010]Rafael Gomez, Validation of the Two-Phase Flow Models of the 3D Subchannel code SUBCHANFLOW using the NUPEC BFBT data, Technical report, FZK-INR, 2010.

[Gommlich2010] A. Gommlich, S. Kliem, U. Rohde, A. Gomez, V. Sanchez, Coupling of the Neutron Kinetic Core Model DYN3D with the Thermal Hydraulic Code FLICA-4 within the NURESIM Platform, Proceedings of Annual Meeting on Nuclear Technology, Berlin, May 4-6, 2010.

[Gouja2010] Ilyes Gouja, Maria Avramova, Adam Rubin, Development and optimization of coupling interfaces between reactor core neutronics and thermal-hydraulic codes, PHYSOR 2010, Pittsburg, Pennsylvania, USA, May 9 -14, 2010.

[Grossman2007] Lawrence M. Grossman, Jean-Pierre Hennart, Nodal diffusion methods for space-time neutron kinetics, Progress in Nuclear Energy 49 (2007) 181-216.

[Grötzbach1977] G. Grötzbach, Direkte numerische Simulation turbulenter Geschwindigkeits-, Druk- und Temperaturfelder bei Kanalströmungen, Dissertation von der Fakultät für Maschinenbau der Universität Karlsruhe, October 1977.

[Grundmann1997]U. Grundmann, U. Rohde, Verification of the code DYN3D/R with the help of international Benchmarks, Internal report Forschungszentrum Rossendorf 195, October 1997.

[Grundmann1998]U. Grundmann, S. Kliem, E. Krepper, S. Mittag, U. Rohde, F. Schäfer, A. Seidel, Qualifizierung des Kernmodells DYN3D im Komplex mit dem Störfallcode ATHLET als fortgeschrittenes Werkzeug für die Störfallanalyse von WWER-Reaktoren, Report FZR-216 Rossendorf, 1998.

[Grundmann2003]U. Grundmann, S. Kliem, Analyses of the OECD main steam line break benchmark with the DYN3D and ATHLET codes, Nuclear Techn., 142 (2003) 146-153.

[Grundmann2005]Grundmann, U., Rohde U., Mittag S., Kliem S., DYN3D Version 3.2 Code for Calculation of Transients in LWR with Hexagonal or Quadratic Fuel Elements – Description of Models and Methods, Research Centre Rossendorf Inc., August 2005.

[Grundmann2006]U. Grundmann, Calculations of a steady state of the OECD/NRC PWR MOX/UO2 transient benchmark with DYN3D, Proceedings of Annual Meeting on Nuclear Technology, Aachen, May 16-18, 2006.

[Grundmann2009]U. Grundmann, "DYN3D – MG – V2.0: Code for Calculation of Steady States and Transients of Reactors by using the Multigroup Neutron Diffusion approximation for hexagonal or quadratic fuel Assemblies or the Multigroup SP3 approximation for quadratic fuel assemblies", Institute of Safety Research, Forschungszentrum Dresden-Rossendorf, December 2009.

[Grundmann2010]U. Grundmann, S. Mittag, U. Rohde and S. Kliem, DYN3D Version 3.2, Code for Calculation of Transients in Light Water Reactors (LWR), with Hexagonal or Quadratic Fuel Elements, Code Manual and Input Data Description for Release, September 2010.

[Guelfi2005] A. Guelfi, M. Boucker, J-M. Hérard, P. Péturaud, D. Bestion, P. Boudier, P. Fillion, M. Grandotto and E. Hervieu, A new multi-scale platform for advanced nuclear thermal-hydraulics Status and prospects of the NEPTUNE project, NURETH-11 Avignon, France, October 2-6, 2005.

[Han2004] Han Gyu Joo, Jin Young Cho, Kang Seog Kim, Chung Chan Lee and Sung Quun Zee, Methods and Performance of a Three-Dimensional Whole-Core Transport Code DeCART, PHYSOR 2004 -The Physics of Fuel Cycles and Advanced Nuclear Systems: Global Developments Chicago, Illinois, April 25-29, 2004.

[Han2009] Han Gyu Joo, Joo Il Yoon, Seung Gyou Baek, Multigroup pin power reconstruction with two dimensional source expansion and corner flux discontinuity, Annals of Nuclear Energy 36 (2009) 85-97.

[Hoffmann2000] Klaus A. Hoffmann, Steve T. Change, Computational Fluid Dynamics, Volume I. ISBN 0-9623731-0-9, Enginnering Education System, Wichita, Kansas, USA, 2000.

[Hollenbach2005] Hollenbach D.F., Dunn M., Status and Preliminary Testing of Continuous-Energy KENO V.a and KENO-VI, Proceedings of the 2005 NCSD Topical Meeting, Knoxville, Tennesse, 2005.

[Imke2010]Imke, U., Sanchez, V., and Gomez, R. Subchanflow: A new empirical knowledge based subchannel code, Annual Meeting on Nuclear Technology, May 2010.

[Imke2011]U. Imke, Input instructions for SUBCHANFLOW 1.8, KIT-INR, January 2011.

[Ivanov1999] K. N. Ivanov, T. M. Beam, A. J. Baratta, PWR Main Steam Line Break (MSLB) Benchmark, Volume I: Final Specifications, NEA/NSC/DOC(99)8, April 1999.

[Ivanov2007] Kostadin Ivanov, Maria Avramova, Challenges in coupled thermal-hydraulics and neutronics simulations for LWR safety analysis, Annals of Nuclear Energy 34 (2007) 501-513.

[Ivanov21999] Kostadin N. Ivanov and Anthony J. Baratta, Coupling Methodologies for Best Estimate Safety Analysis, Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications, Madrid, Spain, September 1999.

[IvanovK1999]K. Ivanov, R. Macian, A. Irani, A. Baratta, Features and Performance of a Coupled Three-Dimensional Thermal Hydraulic/Kinetics PWR Analysis Code TRAC-PF1/NEM, Annals of Nuclear Energy, 26 (1999) 1407-1417.

[Iwamoto1999]Tatsuya Iwamoto and Munenari Yamamoto, Pin power reconstruction methods of the few-group BWR core simulator NEREUS, Journal of Nuclear Science and Technology 36 (1999) 1141-1152.

[Jackson1995] C. J. Jackson and H. Finneman, Verification of the Coupled RELAP5/PANBOX2 System with the NEACRP LWR Core Transient Benchmark, Proc. Int. Conf. Mathematics and Computations, Reactor Physics, and Environmental Analyses, Portland, Oregon, April 30–May 4, 1995.

[Jackson1999] C. J. Jackson, D. G. Cacuci, H. B. Finnemann, Dimensionally Adaptative Neutron Kinetics for Multidimensional Reactor Safety Transients - I: New Features of RELAP5/PANBOX, Nuclear Science and Engineering 131 (1999) 143-163.

[Jamet2010]D. Jamet, O. Lebaigue, C. Morel and B. Arcen, Towards a Multiscale Approach of the Two-Phase Flow Modeling in the Context of DNB Modeling, Nuclear Engineering and Design, 240 (2010), 2131-2138.

[Jimenez2010] Javier Jimenez Escalante, Desarrollo e implementación de la descomposición en subdominios del acoplamiento neutrónico-termohidráulico mediante disecciones alternadas con un sistema de cálculo multiescala, PhD. Thesis, Universidad

Politécnica de Madrid, Escuela Técnica Superior de Ingenieros Industriales, Septiembre, 2010.

[Joon1989] Won Joon Na and Nam Zin Cho, A new formulation of the reconstruction problem in neutronics nodal methods based on maximum entropy principle, Journal of the Korean Nuclear Society, Volume 21, Number 3, 1989.

[Kennard1938] Earle H. Kennard, Kinetic Theory of Gases, with an Introduction to Statistical Mechanics, McGraw-Hill book company, 1938.

[Kliem2008] S. Kliem, DYN3D - Overview on code validation for VVER reactors, EU TACIS project, DYN3D training course, Kiev, November 3 - 7, 2008.

[Kliem2009] S. Kliem, A. Gommlich, Actual Status of the DYN3D/FLICA Coupling inside SALOME, 1st NURISP-SP3 Meeting, PSI, 5-6 November, 2009.

[Kliem2010] Sören Kliem, Ein Model zur Beschreibung der Kühlmittelvermischung und seine Anwendung auf die Analyse von Borverdünnungstransienten in Druckwasserreaktoren, Ph.D Thesis, TU Dresden, July 2010.

[Kliem2011] S. Kliem, S. Mittag, A. Gommlich, P. Apanasevich, D3.1.2.2: Definition of a boron dilution benchmark, NURISP Project, 2011.

[Kliem22010] S. Kliem, U. Rohde, J. Schütze, Th. Frank, Prototype coupling of the CFD software ANSYS CFX with the 3D neutron kinetic core model DYN3D, Proceedings of the PHYSOR 2010, Pittsburgh, USA, 2010.

[Kliem22011] S. Kliem, A. Gommlich, A. Grahn, U. Rohde, J. Schütze, T. Frank, A. Gomez and V. Sanchez, Development of multi-physics code systems based on the reactor dynamics code DYN3D, Kerntechnik 76 (2011) 160-165.

[Koebke1977] Klaus Koebke, Manfred R. Wagner, The determination of the pin power distribution in a reactor core on the basis of nodal coarse mesh calculations, Atomkernenergie Bd. 30 (1977) 136-142.

[Kozlowski2003] Kozlowski, T., Downar, T., OECD/NEA AND U.S. NRC PWR MOX/UO2 CORE TRANSIENT BENCHMARK – Final Specification, OECD Nuclear Energy Agency/Nuclear Science Committee, 2003.

[Kozlowski2004] Tomasz Kozlowski, R. Matthew Miller, Thomas J. Downar, Douglas A. Barber, Han Gyu Joo, Consistent comparison of the codes RELAP5/PARCS and TRAC-M/PARCS for the OECD MSLB coupled code Benchmark, Nuclear Technology, 146 (2004) 15-28.

[Kozlowski2006] T. Kozlowski, T. J. Downar, PWR MOX/UO2 Core Transient Benchmark (Final Report), OECD Nuclear Energy Agency, Rep. NEA/NSC/DOC(2006)20, 2007.

[Kozmenkov2007] Y. Kozmenkov, S. Kliem, U. Grundmann, U. Rohde, F.-P. Weiss, Calculation of the VVER-1000 coolant transient benchmark using the coupled code systems DYN3D/RELAP5 and DYN3D/ATHLET, Nucl. Eng. Design, 237 (2007) 1938-1951.

[Kuzmin2010] Dmitri Kuzmin, A Guide to Numerical Methods for Transport Equations, Lecture, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2010.

[Langenbuch] S. Langenbuch, K. Velkov, S. Kliem, U. Rohde, M. Lizorkin, G. Hegyi, A. Kereszturi, Development of coupled systems of 3D neutronics and fluid-dynamic system codes and their application for safety analysis.

[Langenbuch1984] Langenbuch, S., QUABOX/CUBBOX-HYCA, Ein Dreidimensionales Kernmodell mit parallelen Kühlkanälen für Leichtwasser-reaktoren, GRS-A-926, Garching, Germany, 1984.

[Lassmann1992] K. Lassmann, TRANSURANUS: a fuel rod analysis code ready to use, Journal of Nuclear Materials 188 (1992) 295-302.

[Lautard1999] J. J. Lautard, D. Schneider, and A. M. Baudron, Mixed Dual Methods for Neutronic Reactor Core Calculations in the CRONOS System, Proc. Mathematics and Computation, Madrid, Spain, September 27–30, 1999.

[Leppanen2009] Leppänen J., Pusa M., Burnup Calculation Capability in the PSG2 / SERPENT Monte Carlo Reactor Physics Code, M&C 2009, Saratoga Springs, New York, May 3 – 7, 2009.

[Lerchl1998] Lerchl, G. and H. Austregesilo, ATHLET Mod 1.2 Cycle A – User's Manual, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, GRS-P-1, Vol. 1, October 1998.

[Lötsch2009] Lötsch T., Khalimonchuk V., Kuchin A., (2009), Proposal Of A Benchmark For Core Burnup Calculations For A VVER-1000 Reactor Core, Proceedings of the 19th Symposium of AER on VVER Reactor Physics and Reactor Safety, Varna, Bulgarien, 2009.

[Lozano2008] Juan Andrés Lozano, José María aragonés, Nuria García-Herranz, Transient analysis in the 3D nodal kinetics and thermal-hydraulics ANDES/COBRA coupled system, International Conference on the Physics of Reactors, Casino-Kursaal Conference Center, Interlaken, Switzerland, September 14-19, 2008.

[Lulin2010] Lulin Yu, Dong Lu, Shaohong Zhang and Yung-an Chao, Group decoupled multi-group pin power reconstruction utilizing nodal solution 1D flux profiles, PHYSOR 2010, Pennsylvania, USA, May 9-14 May, 2010.

[MacKenzie2008] David MacKenzie, Ben Elliston Akim Demaille, Autoconf, Creating Automatic Configuration Scripts. Free Software Fundation, Inc. September 2008.

[MacKenzie22008] David MacKenzie, Tom Tromey, Alexander Duret-Lutz, GNU Automake, A program that creates GNU standards-compliant Makefiles from template files, Free Software Fundation, Inc. November 2008.

[Marleau2008] G. Marleau, A. Hebert and R. Roy, A User Guide for DRAGON 3.05F, TECHNICAL REPORT IGE–174 Rev. 6F, Institut de genie nucleaire, Ecole Polytechnique de Montreal, May 2008.

[MATRA1998] Development of a Subchannel Analysis Code MATRA α, Korean Atomic Energy Research Institute, KAERI, April 1998.

[MCNP2006] Monte Carlo N-Particle Transport Code System, MCNP5 1.4, OAK Ridge National Laboratory, January 2006.

[MED2003] EDF R&D, MED/DEM data exchange model definition, version 2.2. 2003.

[Mignot2004] G. Mignot, E. Royer, B. Rameau and N. Todorova, Computation of a BWR Turbine Trip with CATHARE-CRONOS2-FLICA4 coupled codes, Nuclear Science and Engineering 148 (2004) 235-246.

[Nam2005] Nam Zin Cho, Fundamentals and Recent Developments of Reactor Physics Methods, Nuclear Engineering and Technology, 37 (2005) 25-78.

[Nuttin2004] A. Nuttin, Eatat d'avanvement du couplage neutronique/termohydraulique MCNP/TRIO-U. Atellier GADEPEON. Paris 5-6 October 2004.

[OPENFOAM2010] Open FOAM, The open source CFD Toolbox, Users Guide Version 1.7.1, August 2010.

[Papin2006] J. Papin, M. Petit, C. Grandjean, V. Georgenthum, IRSN R&D studies on high burn-up fuel behaviour under RIA and LOCA conditions, International Meeting on LWR fuel performance, Salamanca Spain, 22-26 October 2006.

[Papukchiev2009] Papukchiev, A., Coupling of ANSYS CFX and ATHELT: Progress Report, Presentation at the 16th meeting of German CFD Network, Garching, Germany, 7 - 8 October, 2009.

[Pengelly2002] Jonathan Pengelly, Monte Carlo Methods, Student tutorial at the Computer Science department of the University of Otago, New Zeland, February 26, 2002.

[Perin2010] Périn Y., Velkov K., Pautz A., COBRA-TF / QUABOX-CUBBOX: Code system for coupled core and subchannel analysis, PHYSOR 2010, Pittsburg, Pennsylvania, USA, May 9-14, 2010.

[Pralong2005] C. Pralong Fauchere, M. Murphy, F. Jatuff, R. Chawla, Within-Pin Reaction Rate Distributions: CASMO-4 and HELIOS Compared Against Tomographic

Measurements at the PROTEUS Reactor, Nuclear Science and Engineering, 150 (2005) 27-36.

[PyQtRiverbank] PyQt Reference Guide, Riverbank Computing Limited, http://www.riverbankcomputing.com/static/Docs/PyQt4/html/#

[Rabiti2006] C. Rabiti, M.A. Smith, G. Palmiotti, The Method of Characteristics for the Neutron Transport Equation, Seminar talks, Argonne National Laboratory, Nuclear Energy Division, 2006.

[RELAP2000] RELAP5-3D© Code Development Team, RELAP5-3D© Code Manual, INEEL-EXT-98-0083 – Rev.1.2, May 2000.

[Reus2008] Paul Reuss, Neutron Physics, Institut national des sciences et techniques nucleaires, ISBN: 978-2-7598-0041-4, EDP Sciences 2008.

[Rhodes2006] Joel Rhodes, Kord Smith, Deokjung Lee, CASMO-5 Development and Applications, PHYSOR-2006, ANS Topical Meeting on Reactor Physics, Vancouver, BC, Canada, September, 2006.

[Royer2008] Eric Royer, Multiphysics and LWR transient analysis, International Conference on the Physics of Reactors, Casino-Kursaal Conference Center, Interlaken, Switzerland, September 14-19, 2008.

[Sadatomi1994] Michio Sadatomi, Akimaro Kawahara, Yoshifusa Sato, Flow redistribution due to void drift in two-phase flow in a multiple channel consisting of two subchannels, Nuclear Engineering and Design 148 (1994) 463 – 474.

[SALOME] The open source Integration Platform for Numerical Simulation. http://www.salome-platform.org

[Salome2005] Salome Training Session, Sept. 19 - 21, 2005.

[Salome2007] JFA, OPEN CASCADE, SALOME Tutorial User Guide. February 2007.

[SanchezR2003] Richard Sanchez et. al., APOLLO2: Reference Manual for Version 2.8-2.E, CEA DM2S REPORT SERMA/LLPR/RT/10-4853/A, March 2003.

[SanchezV2009] V. Sanchez and A. Al-Hamry, Development of a coupling scheme between MCNP and COBRA-TF for the prediction of the pin power of a PWR fuel assembly, M&C 2009, Saratoga Springs, New York, May 3-7, 2009.

[Seindal2010] René Seindal, Francois Pinard, Gary V. Vaughan and Eric Blake. GNU M4, A powerful macro processor. Free Software Fundation, Inc. February 2010.

[Seker2007] V. Seker, J. Thomas and T.J. Downar, Reactor simulations with coupled Monte Carlo and computational fluid dynamics. Joint International Topic Meeting on

Mathematics & Computation and Supercomputing in Nuclear Applications, M&C + SNA 2007 Monterey California, April 15-19, 2007.

[Seubert2007] A. Seubert, S. Langenbuch, K. Velkov, W. Zwermann, Applicability of deterministic and Monte Carlo neutron transport models coupled with thermo-fluiddynamics, EUROSAFE 2007, Towards Convergence of Technical Nuclear Safety Practices in Europe, Berlin, 2007.

[Seubert2008] A. Seubert, K. Velkov, S. Langenbuch, The time-dependent 3D discrete ordinates code TORT-TD with thermal-hydraulic feedback by ATHLET models, International Conference on the Physics of Reactors, Casino-Kursaal Conference Center, Interlaken, Switzerland, September 14-19, 2008.

[Seubert2011] A. Seubert, The Time-Dependent 3-D Transport Code TORT-TD: Recent Advances and Applications, Transactions of the American Nuclear Society, Vol. 104, Hollywood, Florida, June 26–30, 2011.

[Sofu2007] Sofu T., et al., Coupled BWR calculations with the Numerical Nuclear Reactor Software System, Joint International Topic Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications, M&C + SNA 2007 Monterey California, April 15 – 19, 2007.

[Solis2001] Jorge Solis, Kostadin N. Ivanov, Baris Sarikaya, Andy M. Olson and Kenneth W. Hunt, Boiling Water Reactor Turbine Trip (TT) Benchmark, US Nuclear Regulatory Commission and OECD Nuclear Energy Agency, February 2001.

[Solis2002] Jorge Solís, Maria N. Avramova, Kostadin N. Ivanov, Temporal adaptative algorithm for TRAC-BF1/NEM/COBRA-TF coupled calculations in BWR safety analysis, Annals of Nuclear Energy 29 (2002) 2127-2141.

[Solis2004] Solis, J., Avramova, M., Ivanov, K., Multi-level methodology in parallel computing environment for evaluating BWR safety parameters, Nuclear Technology 146 (2004), 267-278.

[Taylor2007] J. Bryce Taylor, Dave Knott, Anthony J. Baratta, A method of Characteristics solution to the OECD/NEA 3D neutron transport benchmark problem, M&C + SNA 2007, Monterey, California, April 15-19, 2007.

[Tohjoh2006] Masayuki Tohjoh, Masato Watanabe, Akio Yamamoto, Three-dimensional pin power reconstruction for the axially heterogeneous region in BWR, Annals of Nuclear Energy 33 (2006) 242-251.

[Toumi2000] I. Toumi, A. Bergeron, D. Gallo, E. Royer, D. Caruge, FLICA-4: a three-dimensional two-phase flow computer code with advanced numerical methods for nuclear applications, Nuclear Engineering and Design 200 (2000) 139-155.

[TRACE2008] TRACE V5.0 Theory Manual - Field Equations, Solution Methods, and Physical Models, U.S. Nuclear Regulatory Commission, 2008.

[Turinsky1994] Turinsky, P.J., et al., NESTLE: A Few-group Neutron Diffusion Equation Solver Utilizing the Nodal Expansion Method for Eigenvalue, Adjoint, Fixed-source Steady State and Transient Problems, EGG-NRE-11406, Idaho National Engineering Laboratory, June 1994.

[Vyskocil2011] Ladislav Vyskocil, Simulation of MSLB Case at VVER-1000 Primary Circuit by Coupled CATHARE and TRIO_U Codes, Deliverable report, NURISP, 2011.

[Watson2010] Justine Kyle Watson, Implicit time-integration method for simultaneous solution of a coupled non-linear system, PhD Thesis, The Pennsylvania State University, May 2010.

[Weber2005] Weber D.P., et al., The Numerical Nuclear Reactor – A high Fidelity, Integrated Neutronic, Thermal- Hydraulic, and Thermo-Mechanical Code., Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications Palais des Papes, Avignon, France, September 12-15, 2005.

[Wheeler1976] C. L. Wheeler, et al., COBRA-IV-I: An Interim Version of COBRA for Thermal Hydraulic Analysis of Rod Bundle Nuclear Fuel Elements and Cores, BNWL-1962, Pacific Northwest Laboratory (1976).

[Williams1971] M. M. R. Williams, Mathematical Methods in Particle Transport Theory, Butterworths, London 1971.

[Williams2007] Thomas Williams and Colin Kelley, gnuplot An interactive Plotting Program, March 2007.

[WNA20101] Outline History of Nuclear Energy, World Nuclear Association, http://world-nuclear.org/info/inf54.html , June 2010.

[WNA22010] Safety of Nuclear Power Reactors, World Nuclear Association, http://world-nuclear.org/info/inf06.html , September 2010.

[Xu2005] Y. Xu, T. Downar, The implementation of Matrix Free Newton/Krilov Methods Based on a Fixed Point Iteration, Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Avignon, France, Semptember 2005.

[Xu2009] Yunli Xu, Thomas Downar, R. Walls, K. Ivanov, J. Staudenmeier, J. March-Lueba, Application of TRACE/PARCS to BWR stability analysis, Annals of Nuclear Energy 36 (2009) 317-323.

[Ziabletsev2000] Dmitri N. Ziabletsev, Kostadin N. Ivanov, Improved verification methodology for TRAC-PF1/NEM using NEA/OECD core transient benchmarks, Annals of Nuclear Energy 27 (2000) 1319-1331.

[Ziabletsev2004] Ziabletsev, D., Avramova, M., Ivanov, K., Development of PWR integrated safety analysis methodology using multi-level coupling algorithm. Nuclear Science and Engineering, 148 (2004) 414-425.

Annex A DYNSUB Graphical User Interface

Annex .A.1 Introduction

A Graphical User Interface (GUI) is nowadays a must in the development of new codes. GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. Tasks like pre and post processing are done easily with these tools and the typical way of making input files (text mode) can be checked in a visual way. At the end, the idea of having a GUI is not just to have a better presentation of the code but also to save time in the design of models and in the analysis of output data.

Very big efforts have been done worldwide in order to provide GUI in all the fields. A clear example in the nuclear field is the NURESIM platform [Cacuci2006]. It is aimed not only to provide a set of multiphysics and multiscale calculation tools in a graphical environment easy to manipulate, but also to have an easy way to analyze results either online or offline.

Based in this principle, a GUI for DYNSUB was developed. The open source PyQt package [PyQtRiverbank] based in Qt and Python was used. The DYNSUB GUI can be currently used as a post processing tool and is under development for pre-processing tasks. In the next section the GUI will be introduced.

Annex .A.2 DYNSUB GUI description

In the main window of the DYNSUB GUI, shown in Figure 7–1, the path with the input files and the problem name must be provided.

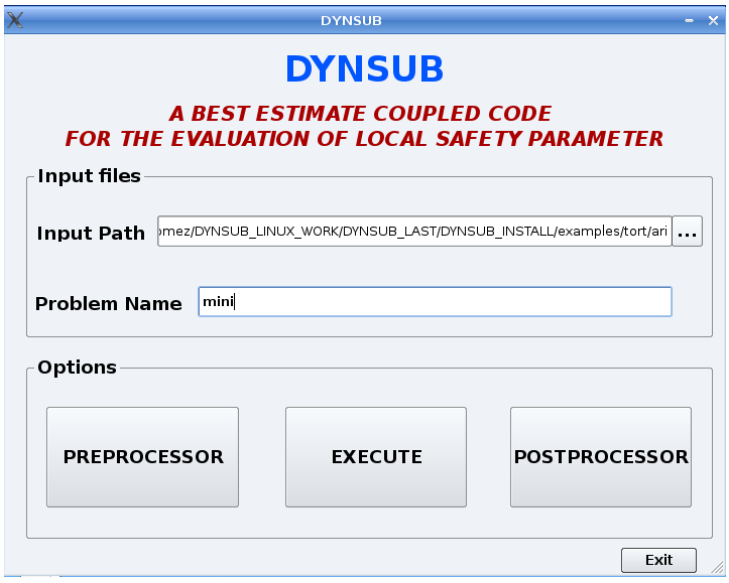


Figure 7–1 DYNSUB GUI main window

Three options are then available:

- The preprocessor option (under development) will provide the possibility to check coherence between the input files. It is also forseen to provide templates for the creations of input files for SUBCHANFLOW and DYN3D-SP3 in order to be used either as a standalone calculation or in a DYNSUB calculation.

- The execute option is a shortcut for executing the code in a visual way and not in command line. It takes control of the system and executes the chosen case. It may be not very practical in cases in which the problem is very big and will take a considerable time running; however some capabilities are also foreseen in order to get online information in an output panel.

- The postprocessor option is the most developed part and provides a very useful tool for a fast analysis of the main output data. A description of this option is given hereafter.

Annex .A.3 Postprocessor tools

Clicking in the POSTPROCESSOR buttom will show a new window (Figure 7–2).

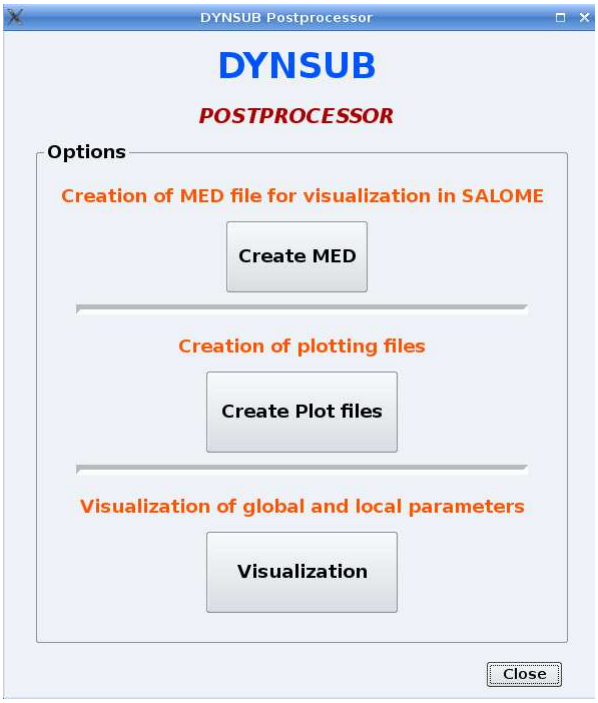


Figure 7–2 DYNSUB GUI Postprocessor

Three options are again available:

- Create MED: With this option it is possible to take the output files of DYNSUB and create med files. Thus, all the powerful post-processing odds of the SALOME platform can be used. The Create MED buttom will activate the “Creation of MED mesh and fields” window (Figure 7–3). The creation of a med file containing all the results is done after cklicking start button.

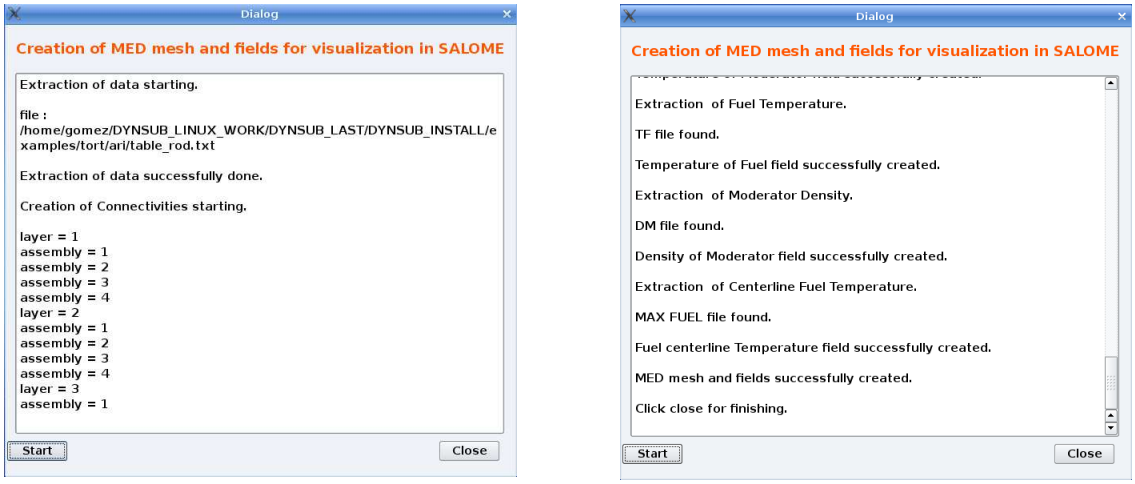


Figure 7–3 DYNSUB GUI Creation of MED files.

The SALOME platform can now be used for visualization of results.

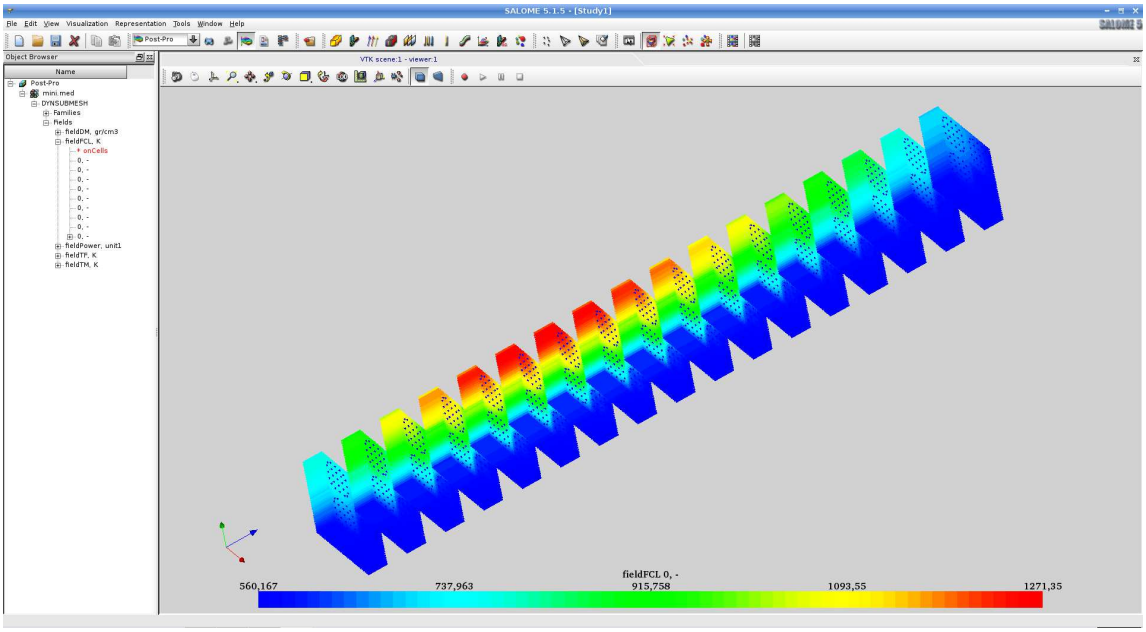


Figure 7–4 Maximal Fuel Temperature field in the MED file created with DYNSUB GUI

- Create plot files: With this option it is possible to create ASCII files for plotting. A detailed description of the functions available and files that can be created is done in Annex C.3. The same Python functions can be executed but in a graphical environment

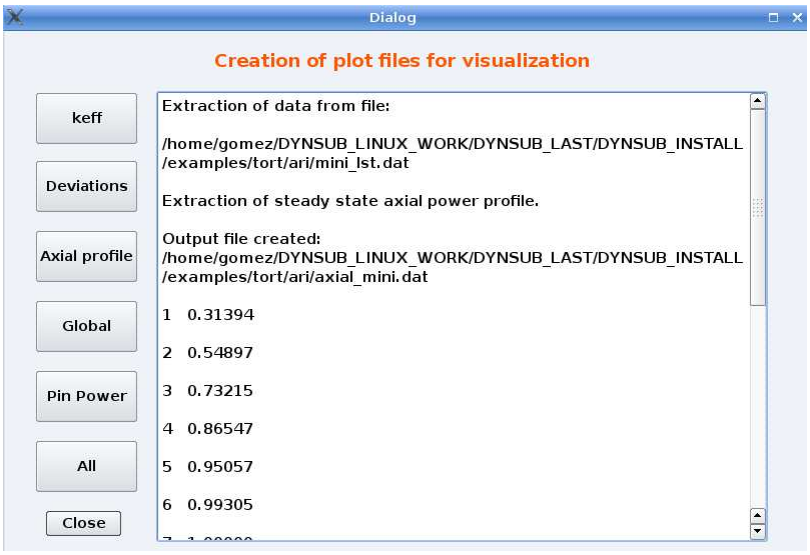


Figure 7–5 DYNSUB GUI Creation of plot files for visualization.

- Visualization: this option gives a faster access to the visualization of results. It is possible to obtain plots of the keff, axial power distribution at steady state, and the transient behaviour of global and local parameters. In Figure 7–6 the total power of the transient for the Case 1 is shown. Figure 7–7 shows the maximal clad temperature.

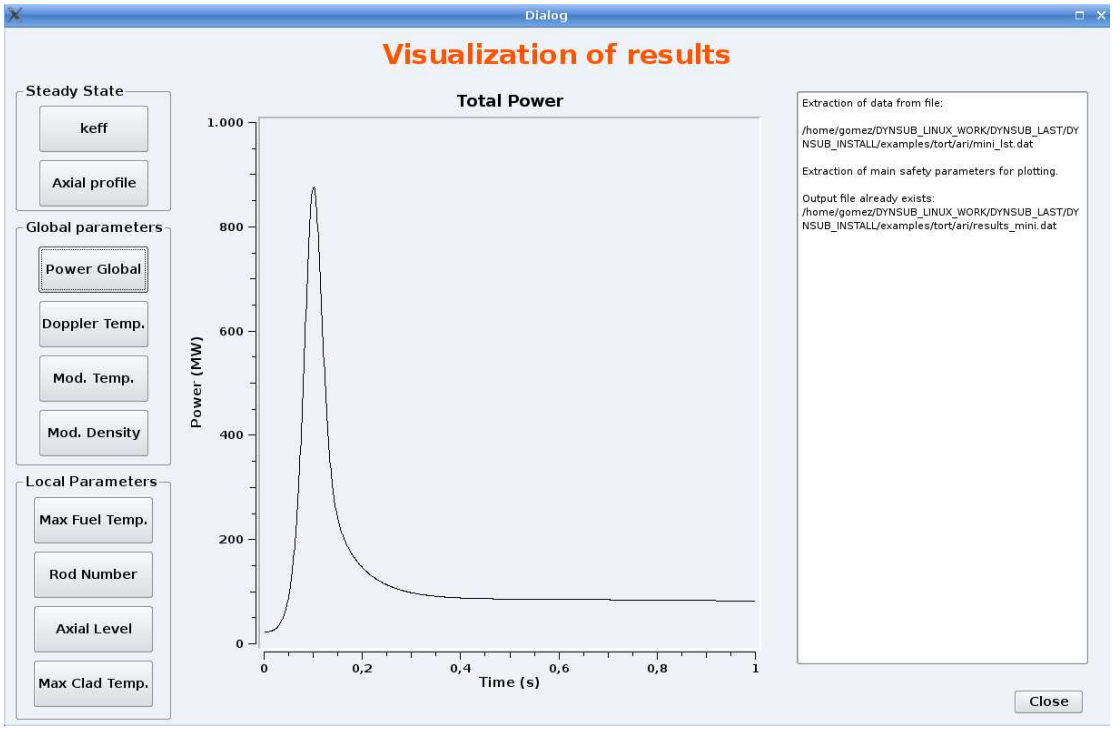


Figure 7–6 DYNSUB GUI Visualization of total power as a function of time.

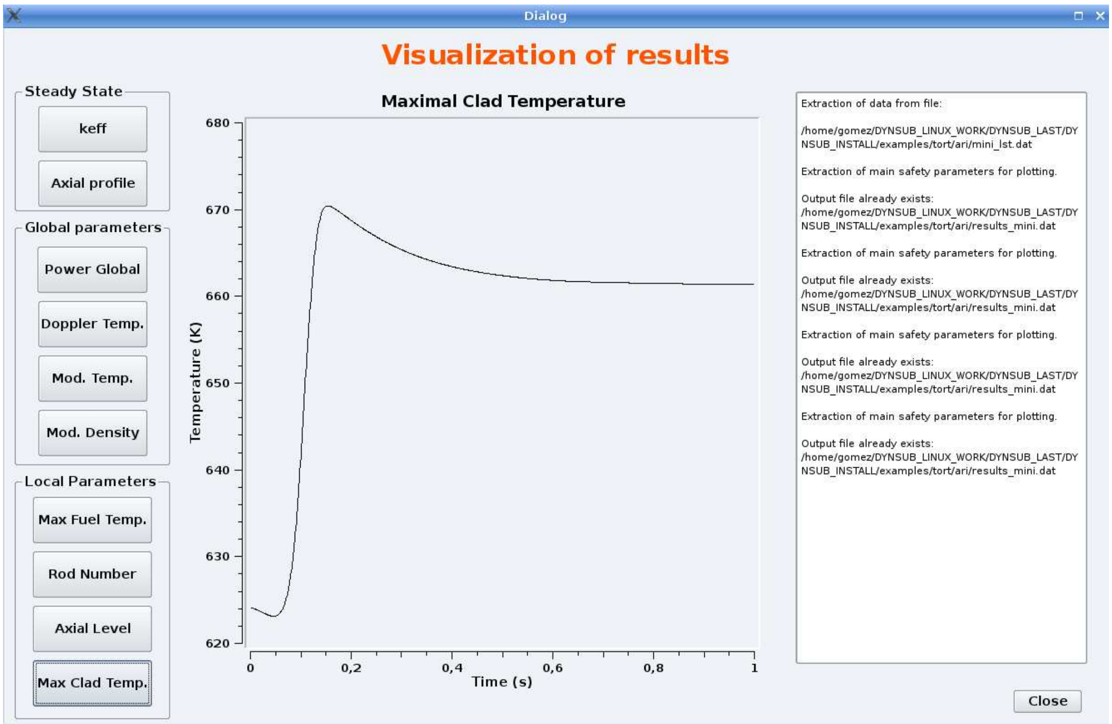


Figure 7–7 DYNSUB GUI Visualization of the maximal clad temperature as a function of time.

The visualization of data can be done either at the end of the transient or at every moment. In this way an "online" analysis of results can be done.

Annex B SUBCHANFLOW Pre-processor

Annex .B.1 Introduction

The use of subchannel codes is intrinsic linked to the creation of tables with information defining the subchannels and fuel rods. Thus, tables for the channel and rod layout specifying geometrical configuration, material composition and neighbourhood have to be created. Additionally, some tables with operational conditions or defining transients are some times also needed. For small clusters or fuel bundles, these tables can be created by hand without too many problems. However, an automatic tool for the creation of such tables is a requisite when dealing with very big problems in which a great amount of channels and rods has to be created.

SUBCHANFLOW had a very basic Pre-processor able to create automatic input files for simple regular geometries (square bundles of several pins). However, it was not able to model several fuel assemblies containing hundreds of channels and with a non-uniform configuration (core-like configuration like the one presented in Figure 7–8).

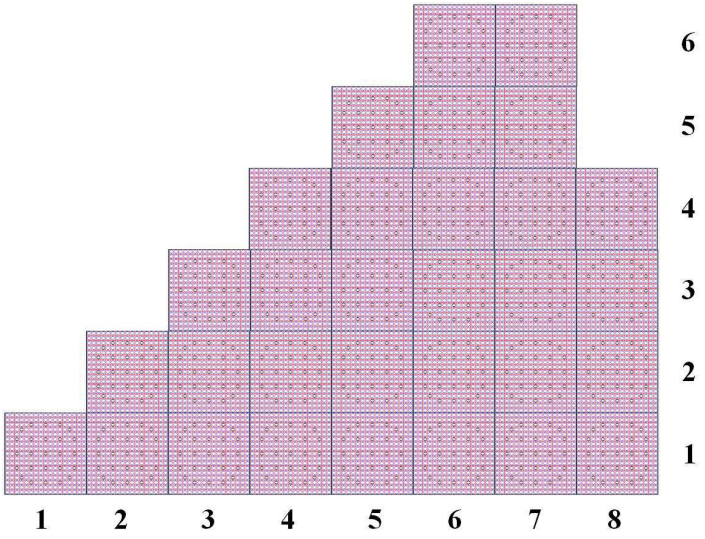


Figure 7–8 Subchannel representation of an eighth of PWR core.

The coupling developed in this dissertation needed a more general and flexible tool in order to represent accurately a core configuration. Thus, even the simplest configuration to be solved with the coupled system (one fuel assembly for instance) may need the creation of tables with hundreds of rows.

Annex .B.2 Input data description for the standalone version of the SUBCHANFLOW Pre-processor

The input data follows the same format as SUBCHANFLOW [Imke2011]. It has to be typed in a text file ***preproc.dat***. The input is organized by keywords. All input data is mandatory, there are no default values. All records starting with "!" are defined as comments, a comment

can be used at the end of each record starting with "!". A totally blank line is also handled as comment line. The input file without comments is found in ***clean_preproc.dat***.

The input data description for the Pre-processor is listed hereafter.

Table 7-I Description of input data for the Pre-processor

Keyword	Variable	Type	Unit	Comment
no_assem	ncas	integer	---	Number of total assemblies in the configuration.
no_row	jup	integer	---	Number of horizontal rows in the configuration.
column_map	icl(j)	integer	---	x-coordinates of the assemblies on the left side of the configuration. $1 \leq j \leq jup$
	icr(j)	integer	---	x-coordinates of the assemblies on the right side of the configuration. $1 \leq j \leq jup$
axial_levels	nz	integer	---	Axial levels in the configuration.
pitch_assem	pitch_ass	real	m	Fuel assembly pitch.
rods_in_x	nrodx	integer	---	Number of rods in the x direction inside the fuel assembly.
rods_in_y	nrody	integer	---	Number of rods in the y direction inside the fuel assembly.
pin_pitch	pin_pitch	real	m	Pitch of the central fuel rod.
bound_pitch	bou_pitch	real	m	Pitch of the fuel rod located in the boundary of the fuel assembly.
wetted_factor	wet_bou	real	---	0.0 if it is an open fuel assembly (PWR). 1.0 if it is a closed fuel assembly (BWR).
rod_diameter	rod_diam	real	m	Diameter of the fuel rod.
guide_diameter	guide_diam	real	m	Diameter of the guide tube.
control_rod_map	control_r(i)	integer	---	The position of the control rods or guide tubes inside the assembly has to be given: 1 in the position of a control rod, 0 otherwise. $1 \leq i \leq nrodx*nrody$

Annex .B.3 Output description for the standalone version of the SUBCHANFLOW Pre-processor

The output is a set of five files containing some of the tables used as input in SUBCHANFLOW. The last version of SUBCHANFLOW used in this dissertation was version 1.8. Newest versions may need some changes in the table configuration. A further extension of the pre-processor must be done for taking into account the possible changes in the configuration of the tables.

A brief description of the each of the output files is found in the next tables (Table 7-II to Table 7-III). Details about the column headers of each table (definition, units, etc.) are given in the input instructions for SUBCHANFLOW 1.8 [Imke2011].

Table 7-II Channel layout tables.

File name	Description	Column headers
table_channel.txt	Defines the layout of each subchannel.	1. channel number 2. channel area 3. wetted perimeter 4. heated perimeter 5. coordinate x 6. coordinate y
table_channel_neighbors.txt	Defines the neighbourhood of each subchannel.	1. channel 2. neighbour 1 (n1) 3. gap to n1 4. distance to n1 5. neighbour 2 (n2) 6. gap to n2 7. distance to n2 8. neighbour 3 (n3) 9. gap to n3 10. distance to n3 11. neighbour 4 (n4) 12. gap to n4 13. distance to n4

Table 7-III Rod layout tables.

File name	Description	Column headers
table_rod.txt	Defines the layout of each rod.	1. rod number 2. material type 3. outer diameter 4. power fraction 5. coordinate x 6. coordinate y
table_rod_channel.txt	Defines the relation between the rods and the neighbouring channels.	1. channel 2. neighbour 1 (n1) 3. gap to n1 4. distance to n1 5. neighbour 2 (n2) 6. gap to n2 7. distance to n2 8. neighbour 3 (n3) 9. gap to n3 10. distance to n3 11. neighbour 4 (n4) 12. gap to n4 13. distance to n4

Table 7-IV Power distribution map in axial and lateral direction

File name	Description	Column headers
table_power_map.txt	Defines the 3D power distribution for all the rods.	1. axial cell number 2. rod number 3. power value

Annex .B.4 Input Example for the standalone version of the SUBCHANFLOW Pre-processor

As example the configuration showed in Figure 7–8 will be used. An eighth of PWR core based in the OECD/NEA and U.S. NRC PWR MOX/UO2 core transient Benchmark [Kozlowski2003] with 21 axial levels is considered.

The input file ***preproc.dat*** for such configuration is shown in Figure 7–9:

```
no_assem: 31
no_row: 6
column_map:
1 2 3 4 5 6
8 8 8 8 7 7
axial_levels: 21
pitch_assem: 0.2142
rods_in_x: 17
rods_in_y: 17
pin_pitch:    12.6e-3
bound_pitch: 6.3e-3
wetted_factor: 0.0
rod_diameter:   9.5e-3
guide_diameter: 1.2064e-2
control_rod_map:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 7–9 Input file for the standalone version of the SUBCHANFLOW Pre-processor.

Annex .B.5 Input data description for the SUBCHANFLOW Pre-processor integrated in DYNSUB

The Pre-processor working for the stand alone version of SUBCHANFLOW was included in the coupling as a subroutine. Several modifications were implemented. Due to the fact that the Pre-processor acting as a subroutine has to be able to identify the geometrical variables already described in the input file of DYN3D-SP3 and stored in internal arrays, simplifications in the input file are found. Only additional information must be provided via a simpler input file (also called ***preproc.dat***).

Some differences can be however distinguished:

- The use of keywords has been suppressed.
- It is possible to choose different materials in the fuel assemblies. Thus, a core with MOX and UO_2 can be represented giving appropriate material parameters of the different fuels in the SUBCHANFLOW input file [Imke2011].
- The FLOCAL model of DYN3D-SP3 has a boron transport model and the data for the boron concentration is given via input of the thermal-hydraulics data. In the case of SUBCHANFLOW, until now, it has no boron transport model. Thereby, only a fixed boron concentration can be considered with DYNSUB and is given at the end of the ***preproc.dat*** file.

The input data description for the Pre-processor integrated as a DYNSUB subroutine is listed in Figure 7–10.

Table 7-V Description of input data for the Pre-processor as a DYNSUB subroutine.

Variable	Type	Unit	Comment
diff_assem	integer	---	Number of different assemblies in the configuration.
rod_diameter	real	m	Diameter of the fuel rod.
guide_diameter	real	m	Diameter of the guide tube.
ass_type(jup,nimax)	integer	---	The fuel assembly map for each of the different materials considered and given by input in the first card. $1 \leq$ ass_type(j,i) $\leq$diff_assem with $1 \leq j \leq$ jup and icl(j)$\leq i \leq$icr(j) (with the internal DYN3D-SP3 variables: jup: Number of horizontal rows, nimax: maximal number of columns, icl and icr: the x-coordinates of the assemblies on the left and right side of the configuration respectively.)
control_rod(i)	integer	---	The position of the control rods or guide tubes inside the assembly has to be given: 1 in the position of a control rod, 0 otherwise. $1 \leq i \leq$ ndiv*ndiv (ndiv is an internal DYN3D-SP3 variable with the number of rods in the x or y direction inside the fuel assembly)
FIX_BORON	real	ppm	Fixed boron concentration.

Annex .B.6 Output description for the SUBCHANFLOW Pre-processor integrated in DYNSUB

The output tables are the same as the standalone version of the SUBCHANFLOW Pre-processor, described in Annex .B.3

Annex .B.7 Input Example for the SUBCHANFLOW Pre-processor integrated in DYNSUB

Continuing with the same example of the configuration showed in Figure 7–8, the input file ***preproc.dat*** for such configuration is shown in Figure 7–10.

```
! Number of different assemblies
 1
! Rod diameter
 9.5e-3
! Guide diameter
 1.2064e-2
! Fuel assembly map
1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1
    1 1 1 1 1 1
      1 1 1 1 1
        1 1 1
          1 1
control_rod_map:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
!Fixed Boron concentration:
1605.0
```

Figure 7–10 Input file of the SUBCHANFLOW Pre-processor as a subroutine in DYNSUB.

A fixed boron concentration of 1605 ppm has been considered. A more representative input file can be done if the different fuel types are taken into account. In Figure 7–11 a core configuration as a mixture of UO_2 and MOX fuel assemblies (as it is exactly described in [Kozlowski2003]) has been considered.

```
! Number of different assemblies
 2
! Rod diameter
 9.5e-3
! Guide diameter
 1.2064e-2
! Fuel assembly map
1  1  1  1  1  2  1  1
   1  1  2  1  1  2  1
      1  1  1  2  1  2
         2  1  1  2  1
            1  1  1
               2  1
control_rod_map:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
!Fixed Boron concentration:
1605.0
```

Figure 7–11 Input file of the SUBCHANFLOW Pre-processor as a subroutine in DYNSUB with different fuel assemblies.

Although different materials can be used, the inner configuration can not be different in this version of the Pre-processor. Extensions are foreseen for the analysis of BWR with assemblies with different internal configurations. In such cases, a loop over the different materials has to be included.

Annex C DYNSUB Working copies and Installation

The distribution package of DYNSUB is based in Autotools [GNUAutotools] and the control is done via SUBVERSION [Collins2008]. In this Annex, the procedure for doing DYNSUB working copies, for installing DYNSUB and for using the post-processing Python scripts for analysing results in console are described.

Annex .C.1 Working copies of DYNSUB

DYNSUB Repository is installed in the INRSIM01 server at the Institute for Neutron Physics and Reactor Technology (INR) in the Karlsruhe Institute of Technology (KIT) under:

/path/to/repository/DYNSUB_LINUX_SVN

The main branch of the code (under developing version) is located in:

/path/to/repository/DYNSUB_LINUX_SVN/DYNSUB_LINUX/TRUNK

For making a working copy in a remote computer use:

```
[gomez@localhost home]$ svn checkout \
svn://path/to/repository/DYNSUB_LINUX_SVN/DYNSUB_LINUX/TRUNK
```

For a working copy in an account in the server use:

```
[gomez@localhost home]$ svn checkout \
file:///path/to/repository/DYNSUB_LINUX_SVN/DYNSUB_LINUX/TRUNK
```

Annex .C.2 DYNSUB Installation and execution

The structure of the source directory is shown in Figure 4–16. In the ***DYNSUB_SRC*** folder, an installation script (***install_dynsub.sh***) is found. For making an installation, in the upper part of this script is necessary to modify the path where the source is located (usually the local working copy) and the path where the package is desired to be installed. Additionally it is possible to choose the FORTRAN compiler to be used in the installation.

DYNSUB has been developed and checked under LINUX distribution MANDRIVA 2008 using the **gfortran** compiler. Slightly changes (mainly related with writing formats) have been introduced in the DYN3D-SP3 source in order to get compilation and linkage with **gfortran**. The use of other compilers may produce some warning or errors intrinsically associated with each compiler and thus modifications to the source may be necessary in order to get a successful compilation.

After modifying the paths it is just necessary to run the script by means of:

```
[gomez@localhost DYNSUB_SRC]$ ./install_dynsub.sh
```

The ***install_dynsub.sh*** script is here listed with the lines where changes must be done in red (paths of DYNSUB source and of the destination directory where the package must be installed.)

```
#!/bin/bash
#*****************************************************************************
# This script:
# - prepares the environmental variables for the installation of DYNSUB
# - copy the DYNSUB sources in the directory DYNSUB_SRC
# - creates the DYNSUB_INSTALL and DYNSUB_BUILD used in the installation process
# - compiles and installs the DYNSUB package
#*****************************************************************************
# Author: Armando Miguel Gomez Torres
# Date: 01.12.2010
# Place: Karlsruhe Institute of Technology
#*****************************************************************************
# User's space:
# Set where are placed the sources
export SOURCES_DIR=${HOME}/DYNSUB_SRC_backup
# Set the installation directory to copy source, build and install
export INSTALL_DIR=${HOME}
# Chose compiler
export FORTC=gfortran
export FC=${FORTC}
# Fortran compilers for the wrapping
export F77=${FORTC}
#*****************************************************************************
# Installation space:
#
# remove previous installation of DYNSUB
cd ${INSTALL_DIR}
rm -rf DYNSUB_SRC
rm -rf DYNSUB_INSTALL
rm -rf DYNSUB_BUILD
# copy the new sources
echo 'Compiling using' $FORTC
cp -rp ${SOURCES_DIR} DYNSUB_SRC
cd DYNSUB_SRC/
cd src/
cd DYNSUB/
cd DYN3D_SP3/
# export of sources for DYN3D
```

```
for f in modlib/*.f; do
    SRC_DYN3D="${SRC_DYN3D} ${f}"
done
for f in fort/*.f; do
    SRC_DYN3D="${SRC_DYN3D} ${f}"
done
export SRC_DYN3D=${SRC_DYN3D}
cd ..
cd SUBCHANFLOW/
# export of sources for SUBCHANFLOW
for f90 in *.f90; do
    SRC_SUBCHAN="${SRC_SUBCHAN} ${f90}"
done
export SRC_SUBCHAN=${SRC_SUBCHAN}
cd ..
cd ..
cd ..
echo ${PWD}
# creating configure and Makefiles
libtoolize
aclocal
automake --add-missing
autoconf
# creating folders for BUILD and INSTALL
mkdir ../DYNSUB_BUILD
mkdir ../DYNSUB_INSTALL
cd ..
# changing permisions
chmod 776 -R DYNSUB_SRC DYNSUB_INSTALL DYNSUB_BUILD
cd DYNSUB_BUILD
# configure
../DYNSUB_SRC/configure --prefix=${INSTALL_DIR}/DYNSUB_INSTALL
# make and make install
make
make install
#*****************************************************************************
# End of script
#*****************************************************************************
```

Note: This script is based in previous developments related with the compilation and integration of components into the NURESIM platform [Cacuci2006], and in particular related with the integration of COBAYA [Jimenez2010] and DYN3D [Gommlich2010].

After installation the structure of the installed package will be the one shown in Figure 4–17.

The examples directory contains several tests cases ready to run. For running them, it is necessary to copy the ***DYNSUB*** binary file in the test-directory that will be calculated and run the code with the next command:

```
[gomez@localhost example_1]$ ./DYNSUB path/to/test/directory/problem_name
```

Running a new case may require a similar structure as the one of the test cases, i.e. a new folder with the input files of SUBCHANFLOW, DYN3D-SP3 and Pre-processor.

Annex .C.3 Post-processing in console via Python scripts

After running a case with DYNSUB, an output file ***problem_name_lst.dat*** containing the output data of DYNSUB is created. For a quickly analysis of results it is possible to use a command post-processing tool based in Python. In the ***PYTHON_TOOLS*** directory two folders with Python post-processing scripts for extracting data from the DYNSUB output or even from the DYN3D-SP3 standalone output file are found (***DYNSUB_EXTRACT*** and ***DYN3D_EXTRACT*** respectively). In the ***DYNSUB_EXTRACT*** folder there is a script ***dynsub.py*** (or ***dyn3d.py*** in the case of DYN3D-SP3 standalone). This script has several functions in order to extract data from the DYNSUB output file. These functions can be called in a Python console using the following commands sequence:

```
[gomez@localhost example_1]$ cp
    ../../PYTHON_TOOLS/DYNSUB_EXTRACT/dynsub.py .
[gomez@localhost example_1]$ python
>>> import dynsub
Functions ready to use.
>>> dynsub.keff("/path/to/test/directory","problem_name")
START KEFF EXTRACTION
The keff convergence is printed in:
/path/to/test/directory/keff_problem_name.dat
1    1.2351
2    1.2354
…
N    1.2361
END KEFF EXTRACTION
>>>
```

The previous script will extract the k_{eff} convergence iterations and will create a file keff_*problem_name*.dat with the data shown also as output in the console. All the functions available in the Python script and a short description are shown in Table 7-VI. The headers of the columns in the output files created with these functions are presented in Table 7-VII.

Table 7-VI Python functions for extraction of data in Python console

Function's name	Arguments	Description
deviations	*path/to/test/directory* *problem_name*	Extracts the deviation criteria used in the steady state convergence process (deviation in k_{eff}, fuel temperature and moderator density) as a function of the iteration number. The output file is: dev_*problem_name*.dat
keff	*path/to/test/directory* *problem_name*	Extracts the k_{eff} as a function of the iteration number. The output file is: keff_ *problem_name*.dat
axial	*path/to/test/directory* *problem_name*	Extracts the normalized axial power profile for the converged steady state. The output file is: axial_ *problem_name*.dat
extract_global	*path/to/test/directory* *problem_name* Boron_flag (yes/no)	Extracts the global behaviour (average moderator density and temperature, fuel Doppler temperature and boron concentration if it is considered in the calculation) and local safety parameters (maximal fuel and cladding temperatures, as well as their location in the configuration) as a function of time. The output file is: global_param_ *problem_name*.dat
pin_power_distribution	*path/to/test/directory* *problem_name* No. of axial layers No. of assemblies nas nax	Extracts the normalized pin power distribution at the converged steady state for the Assembly assem at the Axial level axi The output file is: ppdata_ *problem_name*_ass_*nas*_ax_*naxi*.dat
plot	*path/to/test/directory* *problem_name*	Same as extract global but without repetition of time steps in case of iteration defined in every time step is desired. The output file is: results_ *problem_name*.dat

Table 7-VII Python functions for extraction of data in Python console

Function's name	Columns	Headers
deviations	4	1. Iteration step. 2. Deviation in $k_{eff.}$ 3. Deviation in fuel Doppler temperature. 4. Deviation in moderator density.
keff	2	1. Iteration step. 2. k_{eff} value.
axial	2	1. Axial layer. 2. Normalized power.
extract_global	21	1. Time. 2. Initial Power. 3. Maximal fuel temperature. 4. Rod number with the maximal fuel temp. 5. Minimal fuel temperature in the same assembly where the maximal fuel temperature occurs. 6. Rod number with the minimal fuel temp. 7. Axial level with the maximal fuel temp. 8. Assembly number with the maximal fuel temp. 9. Maximal clad temperature. 10. Surface clad temperature. 11. Rod number with the maximal clad temp. 12. Axial level with the maximal clad temp. 13. Assembly number with the maximal clad temp. 14. Averaged (global) Doppler temperature. 15. Averaged (global) moderator temperature. 16. Averaged (global) moderator density. 17. Averaged (global) boron concentration (if Boron_flag = 'yes'). 18. Internal iteration number (useful in case of iterations inside the time step are used). 19. Average power in the time step. 20. Actual power at the end of the time step. 21. Deviation in power (useful in case of iterations inside the time step are used)
pin_power_distribution	Number of rods	Matrix containing the pin power distribution in every rod inside the fuel assembly.
plot	21	Same as extract global but without repetition of time steps in case of iteration defined in every time step is desired.

In order to facilitate even more the extraction of data, a Python script ***extract_dynsub.py*** using the functions of **Table 7-VI** for extracting all the possible data is also included in the ***DYNSUB_EXTRACT*** subfolder. It is only necessary to modify the path of the ***extract_dynsub.py*** script in order to use it. Also it is possible to modify the axial position for extracting pin power data at the needed level. This Python script is here listed with the user's defined section in red.

```
# -*- coding: cp1252 -*-
#*************************************************************************
# Main script for extracting from dynsub output file: problem_name _lst.dat
#*************************************************************************
# Author: Armando Miguel Gomez Torres
# Date: 18.02.2011
# Place: Karlsruhe Institute of Technology
#*************************************************************************
#
import os
import dynsub
#*************************************************************************
# User's space:
# Set path of example's folder and name
root_dir = "/path/to/example/directory"
problem_name = "Problem_name"
total_axial = 17
total_assemblies = 4
axial_level = 7
#*************************************************************************
dynsub.deviations(root_dir,problem_name)
dynsub.keff(root_dir,problem_name)
dynsub.axial(root_dir,problem_name)
dynsub.extract_global(root_dir,problem_name,'yes')

for klm in range(total_assemblies):
    dyn-
sub.pin_power_distribution(root_dir,problem_name,total_axial,total_assemblies,klm+1,axi
al_level)

dynsub.plot(root_dir,problem_name)

print 'FIN EXTRACTION'
#*************************************************************************
# End of script
#*************************************************************************
```

Running the script in console can be done by:

```
[gomez@localhost example_1]$ python dynsub_extract.py
```

The script creates automatically all the output files listed in Table 7-VI. A quick visual access to the data in the output files can be done by means of, for instance, gnuplot [Williams2007]. Thus for example, the axial power profile in "axial_*name_problem*.dat" file can be plotted using gnuplot by means of:

[gomez@localhost example_1]$ ***gnuplot***

gnuplot> plot 'axial_name_problem.dat' with linespoints

gnuplot>

In Figure 7–12 the result of this for an example output file 'axial_uo2_2000.dat' is shown.

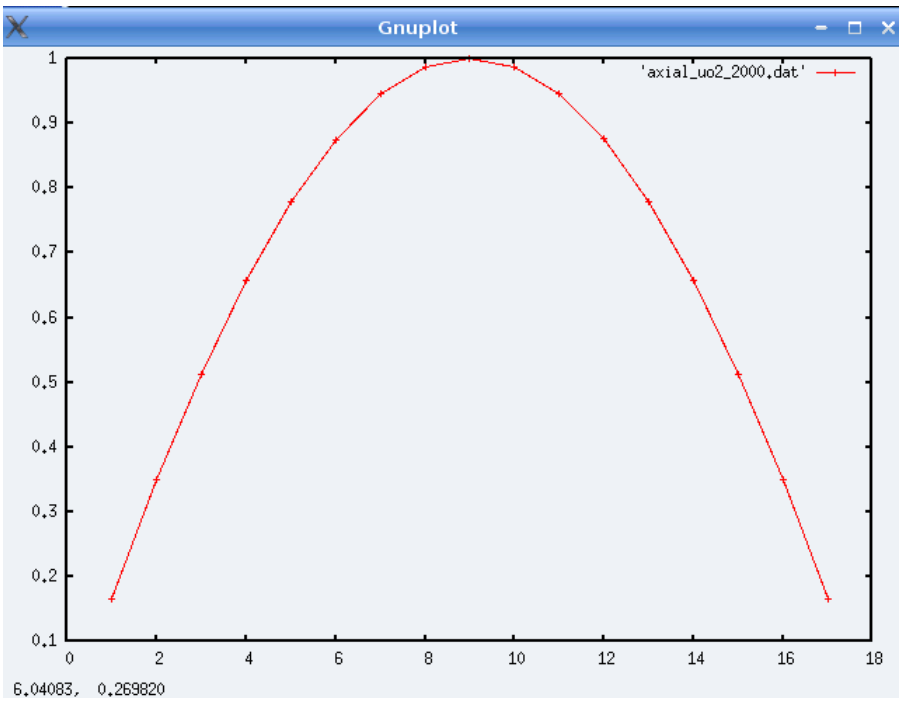


Figure 7–12 Example of axial power distribution plot using gnuplot.