Coupling of Thermal Hydraulic and Neutron Physics Methods for Investigations of Fuel Assemblies in Pressurized Water Reactors

Diplomarbeit zum Erwerb des akademischen Grades "Diplom Ingenieur"
Fakultät für Maschinenbau der Universität Karlsruhe (TH)

Projet de fin d'études présenté pour obtenir le diplôme d'ingénieur E.N.S.A.M.
Ecole Nationale Supérieure d'Arts et Métiers

Performed at

Forschungszentrum Karlsruhe GmbH, Karlsruhe

Institute for Reactor Safety

Neutron Physics and Reactor Dynamics Division

by

Paul Vinson

March 2006

Acknowledgements

The time I spend at the Institute for Reactor Safety was very pleasant and instructive.

First I would like to thank Prof. Dr. Heinzel and Dr. C.H.M Broeders to offer me the opportunity to achieve this diploma work at the institute on a captivating subject. I met during this work high interesting people who welcomed me very friendly.

More particularly I want to thank gratefully Dr C.H.M Broeders and Dr. V. Sánchez for the support, the competences and the kindness I found every time. Thanks to them I have developed my knowledge and skills.

I want to thank M. Becker and P. Oberle for the helpful advices in many subjects and for the enjoyable coffee pauses, J. Aberle for solving my computer problems, and Dr. C. Homann for helping me with UNIX and for the advanced advices.

More generally I want to thank all other colleagues of the institute, who always called me for lunch and who beard kindly my own version of the German language…

Abstract

Coupled neutron physics and thermal hydraulic calculations are necessary to provide a better description of nuclear core behaviours. We assist today's at the development of such analysis tools to investigate different scales of the nuclear power plant. This report presents the realization of a code coupling COBRA-TF with KARBUS/DANTSYS for steady-state analysis on the pin level.

Important theoretical subjects are firstly presented, followed by discussions on the technical realization of the code written in FORTRAN-90. A last section describes the first results obtained with the coupling program on a benchmark proposal and discusses its capabilities and limits.

Table of content

List of Tables

List of Figures

1 Introduction

As a result of governmental directives, research relating to the development of new types of nuclear power plants is proscribed in Germany.
However, the improvement of the power generation inside the cores in operation, the burn-up optimization, and especially a better prediction of the safety criteria margins remain the focus of the research teams.
The Institute for Reactor Safety (IRS), part of the Research Centre of Karlsruhe (FZK), is involved in this domain as major actor.

Safety prediction improvements require the complete study of the whole power plant and necessitate adapting the methods of investigation for each component, with multi physics and multi scales approaches.

Neutron physics and thermal hydraulic analysis are not dissociable in the simulation of nuclear cores because of their interdependence for the solution determination. Namely, the neutronic calculation is strongly influenced by material properties such as coolant densities or fuel temperatures.
A possible approach consists of carry out separated neutronic and thermal neutronic calculations. Then, the influence of one part to the other is simulated by specifying the inputs. Such methods are not always satisfying, and a dynamic coupling between both domains is suitable for the update of dependent parameters, and in this way, to improve the accuracy of the results.

The use of the coupled codes RELAP5/PARCS [1] and TRACE/PARCS [2] at the institute showed good capabilities, but the description on the subchannel level is perfectible by improving the pin power reconstruction model implemented in PARCS. This pin power reconstruction model needs form functions from the thermal hydraulic module to describe the heterogeneities within a fuel assembly.

Material temperatures or survey of the departure from nucleate boiling ratio (DNBR) must be in 3-D investigated at the fuel pin scale instead of an assembly wise, to examine the critical regions and to compare the results with safety criteria. At this scale, Best-Estimate Codes are a good compromise between a correct description of the physical phenomena and a reasonable computing time. COBRA-TF [3] belongs to this category of tools and permits a 3-D pin-by-pin thermal hydraulic assembly calculation.

To improve the form functions, a coupling on the pin level of neutronic and thermal hydraulic calculations is required. The present work describes the coupling of COBRA-TF with the modular system KAPROS-E [4] and the transport code DANTSYS [5] for PWR steady-states investigations of fuel assemblies.
Chosen discussions on thermal hydraulic and neutronic calculations are presented and the main lines of the practical realization of the coupling are exposed.
The feedback process leads to an iterative approach to a stabilized solution and a method to accelerate the convergence is also implemented and discussed.

Finally the capabilities of the new coupled program are tested on the basis of a GRS PWR Benchmark assembly proposal [6].

2 Thermal hydraulic calculations

2.1 Introduction

The first aim of power plants is to furnish kinetic energy to a turbine to produce electricity. Steam is used for this transfer and the types of power plants only differ in the manner of turning water into steam. In nuclear PWR plants we use nuclear fission to produce heat in fuel elements. Then heat must be extracted as much as possible by the primary coolant loop.

The thermal hydraulic part is the science of extracting thermal power resulting of the combustion of fissile material. This extraction has to be optimized to have a better power transmission to the turbine, but also to avoid melting problems in the nuclear core. The cooling efficiency of the core also plays a relevant role for safety considerations, and flow investigations and optimizations are of high interest.

The thermal energy route from the birth to the core exhaust in a PWR is presented in Fig. 2-1. The figure frames the domains of investigation for plant analysis.

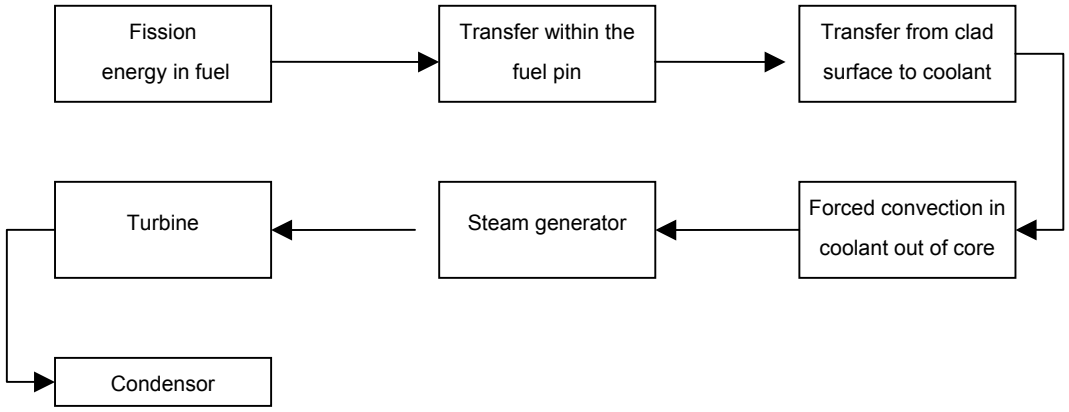


Fig. 2-1 Heat path within a nuclear power plant

There are several codes dedicated to different types of thermal hydraulic studies. The investigated scale goes from whole system analysis up to subchannel calculation as it will be shown in this section.

Analytical solutions for thermal hydraulic problems are often difficult to solve and numerical resolutions have to be applied. For such calculations, the dependencies in space and time for transient investigations must be treated by discretization and this step leads to a system of equations.

Computer improvements permit to solve such kind of system, but the memory size is an obstacle to the treatment of large 3-dimensional problems with a fine spatial discretization. The solution consists in a multi scale approach of the components to apply on each level the con-

venient methods of resolution with a reasonable discretization. Moreover and because of the complexity of the physical models, simplifications and assumptions can be adapted to each level, and more radically, some physical phenomena can simply be neglected if they are of less importance.

The following part of the section presents briefly different levels of investigation on nuclear power plant, with their associated calculation tools. After that the subject are subchannel tools, which are involved in the thermal hydraulic part of the coupling problem.

2.1.1 Containment codes

These tools simulate the augmentation of the pressure within the external containment of the plant to study explosion risks. They are at the top of the discretization scale. GASFLOW [7] and COCOSYS [7] belong to this category.

2.1.2 Reactor system codes

The next level of study for thermal hydraulic tools is the consideration of the reactor system. This level describes the core, the reactor pressure vessel, the primary and secondary circuit including the steam lines and turbine. In addition the volume control systems as well as the emergency core cooling systems are included in the simulation.

For PWR the fluid path is divided into two circuits. The primary one filled with water which extracts the heat of the core and plays the role of moderator too. The secondary loop coupled with the first by a heat exchanger guides the steam to the turbine. The reactor system thermal hydraulic analysis tools describe this whole cycle of heat generation and transfer. They permit to simulate and to study consequences of disturbances appearing in the primary or the secondary loop on the rest of the system (e.g. loss of coolant accident due to breaches in the loops).

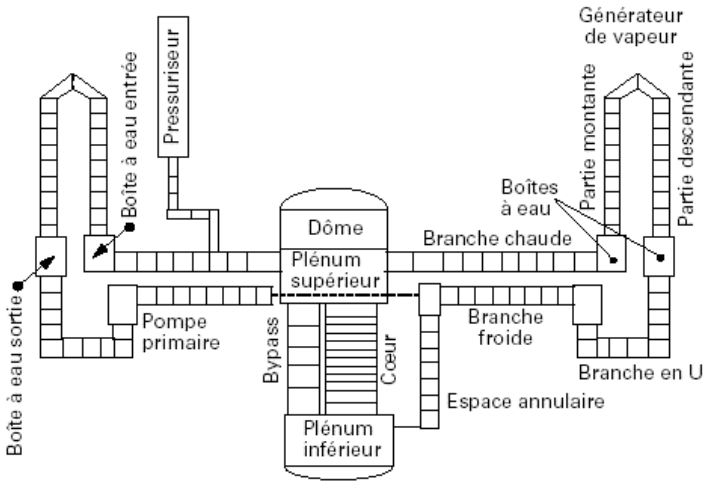


Fig. 2-2 Example of system discretization for the primary loop in CATHARE

Since this type of code handles with a high variety of elements, most of the codes use 1-D hydrodynamic models. The improvements of these codes allow nowadays the utilisation of 2-D and 3-D descriptions. The Fig. 2-2 illustrates the model of the primary loop of a PWR with CATHARE [8]. CATHARE is a French system analysis tool developed in collaboration by FRAMATOME, CEA, and EDF.

2.1.3 Core analysis codes

The heat generation takes place in the core of the nuclear plant. The heat must be extracted here and special tools exist to study this aspect within the core.

The core consists in the repetition of fuel assemblies in accordance with predefined spatial disposition established by neutronic calculations. The Fig. 2-3 represents a PWR core with its components and containments. The right part of the figure shows in details the conception of fuel assembly as bundle of fuel pins and moderator rods (black channels) maintained by grid spacers axially disposed.

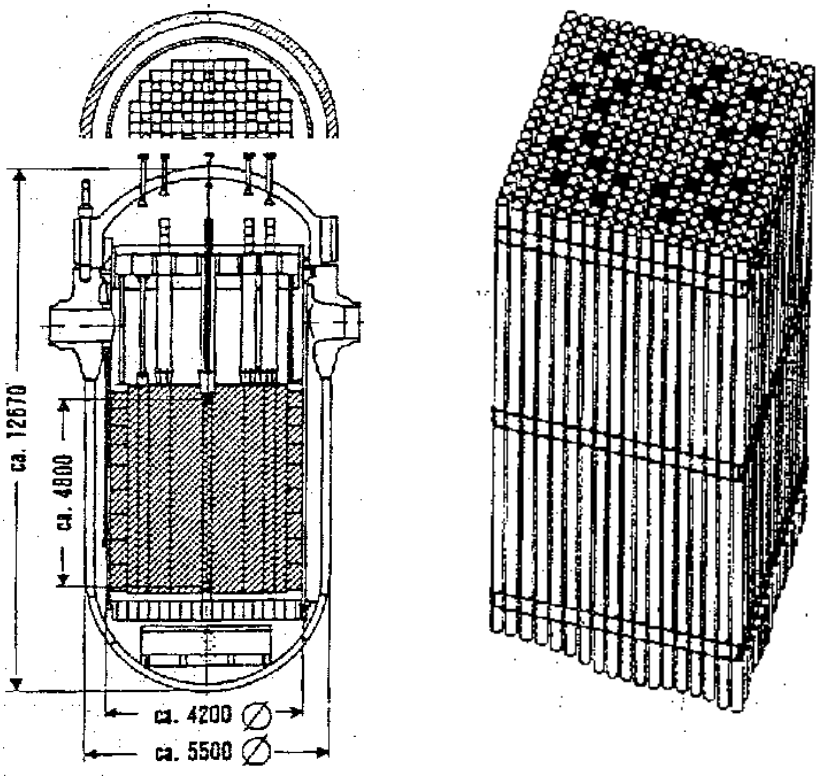


Fig. 2-3 PWR Core and assembly representation [9]

Thermal hydraulic codes for core description deal with the heat transfer between fuel assemblies and the coolant. In this kind of codes, the assembly is considered as a single element and is not detailed more close. Codes allow also the gathering of assemblies for accelerating the calculations but also permit an assembly-by-assembly description. The fluid spaces be-

tween assemblies or assembly bundles are split up into channels. Physical models permit the treatment of heat exchange between assemblies and channels, and between channels together. In this way it is possible to find out the hot spots within the core and to begin more accurate investigations with subchannel analysis tools. The results of this calculation step can be used as boundary condition for the sub assembly analysis.

In fact core codes are also subchannel codes allowing the gathering of subchannels to describe the assembly with average values. COBRA-TF, a subchannel analysis tool, permits this type of core calculation as well as COBRA-EN [10].

2.1.4 Subchannel analysis codes

For this level of geometry discretization, assemblies are detailed pin wise until consideration of pin internal structures. The heat exchange is now investigated directly within the pin and between the fuel pin clad surfaces and the water flowing along the axial direction of the assembly (Fig. 2-4) [11]. The Fig. 2-5 represents a view from the top of the fuel bundle and explains the subchannel control model. The space filled with water between 4 rods is called subchannel. It is the typical domain of investigation of thermal hydraulic subchannel analysis tool. This pattern is not reproducible within the whole assembly due to the control rods presence. Another type of subchannel is so defined to model this domain.

Then the subchannels are nodalized in the axial direction to observe the axial variation of the coolant, and fuel properties. The precision of the nodalization can be adapted to the problems to have accurate results, but naturally the more numerous are the channels and the nodes, the more expensive will be the computer costs.
In each subchannel, computed properties like temperature, density or pressure depend only on the axial height.

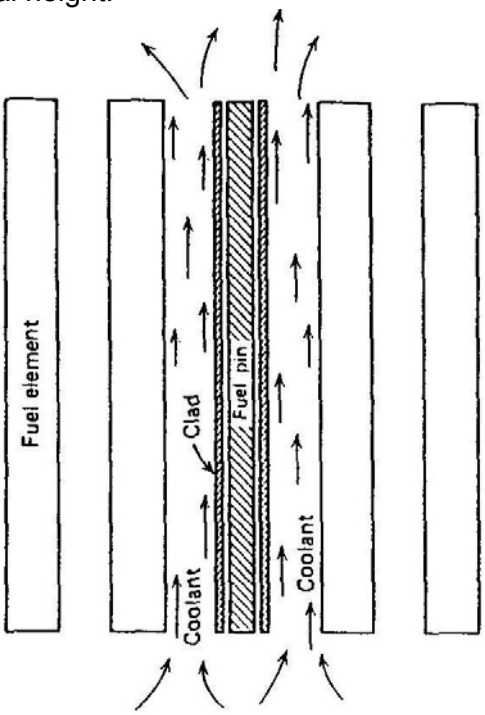


Fig. 2-4 Reactor core cooling

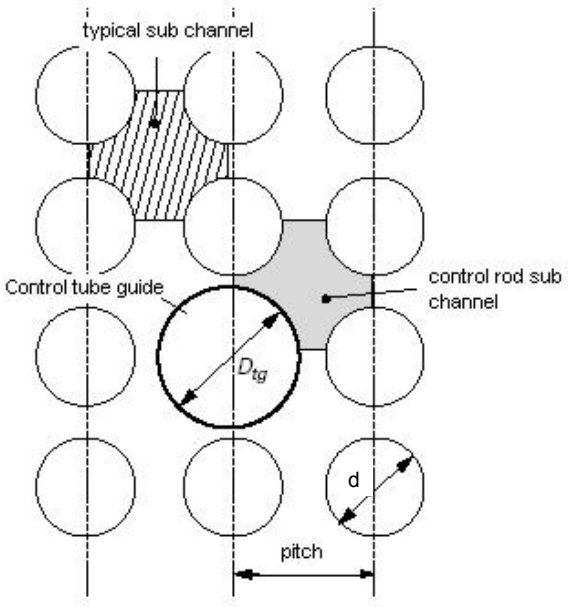


Fig. 2-5 Subchannels illustration

2.2 Codes overview

The next table summarizes the simulation scope with some examples for each category.

Appellation	Containment	System	Subchannels
Domain of study	Containment	Core, coolant loops, valves, pumps…	Core and/or assembly or assembly part
Examples	GASFLOW COCOSYS	RELAP (1D,3D) CATHARE (1D,3D) [8] TRACE (1D,3D) ATHLET (1D)	COBRA [3],[10] FLICA [12] NASCA

Table 2-1 Codes overview

Details on the code mentioned in the table without reference can be found in [7].

For a fuel pin basis approach, tools belonging to the subchannel category are a convenient choice. The corresponding modelization allows a single description of each pin and pin environment within an assembly on different axial layers. The physical models deployed contain physical approximations which lead to a satisfying precision without spending as much time as CFD tools. These theoretical models are introduced in the following part. The references [14] and [17] provide a complete description of the subject, and only selected topics are presented in the next section.

2.3 Physical models involved in subchannel tools

A thermal hydraulic subchannel code consists of a hydrodynamic model which describes the mass flow of liquid and vapor through the system, and a heat transfer model. This last part handles the heat exchange inside the fuel pin and the interfacial transfer behaviour between pin cladding and coolant.

2.3.1 Fluiddynamic models

The hydrodynamic part describes the behaviour of single or multiphase flows in channels. Each flow has to be characterized as well as their interactions by exchanging mass and energy at their interfaces.

Exact conservation equations of energy, mass and momentum can be written for each phase and interface. Theoretically these equations could be solved but the chaotic nature of the topic makes the numerical solution difficult and time consuming. Fortunately the exact motion of each bubble in the flow is not really interesting and it is more pertinent to have an overview of the behaviour of each phase. A special set of time averaged conservation equations is used and is presented in the following section [15].

2.3.1.1 Two fluids flow conservation equations

The two-phase flow model was developed in the seventies and integrated in simulation codes since the eighties. This model treats two fluids existing in the same channel and exchanging energy, mass and velocity at their interfaces. This is possible by resolving 6 constitutive equations (3 for each phase). The drift flux model of Zuber and Findlay [14] can be chosen to reduce the equation number [16].

This kind of model is particularly appropriate when the two fluids have distinct densities and that is typically the case for water and its vapour for PWR investigations. The following equations [15] are written for the phase k, notations and assumptions relative to these equations are reported in Annex A.

Conservation of Mass:

$$\underbrace{\frac{\partial}{\partial t}(\alpha_k\ \rho_k)}_{\text{Rate of change of mass}} + \underbrace{\nabla.(\alpha_k\ \rho_k\ \underline{U}_k)}_{\text{Rate of mass loss}} = \underbrace{\Gamma_k}_{\text{Rate of mass transfer to phase k from the other phases}} \tag{2.1}$$

Conservation of Momentum:

$$\underbrace{\frac{\partial}{\partial t}(\alpha_k\ \rho_k\ \underline{U}_k)}_{\text{Rate of change of momentum}} + \underbrace{\nabla.(\alpha_k\ \rho_k\ \underline{U}^2_k)}_{\text{Rate of momentum loss}} = \underbrace{\alpha_k\ \rho_k\ \underline{g}}_{\text{Gravity force}} - \underbrace{\alpha_k\ \nabla \mathrm{P}}_{\text{Pressure gradient force}}$$
$$+ \underbrace{\nabla .\left[\alpha_k(\underline{\underline{\tau}}_k + \underline{\underline{T}}^T_k)\right]}_{\text{Viscous and turbulent forces}} + \underbrace{\underline{M}^{\Gamma}_k}_{\text{Momentum exchange due to mass transfer to phase k}} + \underbrace{\underline{M}^{d}_k}_{\text{Interfacial drag force}} \tag{2.2}$$

Conservation of Energy:

$$\underbrace{\frac{\partial}{\partial t}(\alpha_k\ \rho_k\ h_k)}_{\text{Rate of change of enthalpy}} + \underbrace{\nabla.(\alpha_k\ \rho_k\ \underline{U}_k\ h_k)}_{\text{Rate of enthalpy loss}} = \underbrace{-\nabla.\left[\alpha_k(\underline{Q}_k + \underline{\underline{q}}^T_k)\right]}_{\text{Conduction and turbulent heat flux}} +$$
$$\underbrace{\Gamma_k\ h^i_k}_{\text{Energy exchange due to mass transfer to phase k}} + \underbrace{q^{'''}_{I_k}}_{\text{Interfacial heat transfer}} \tag{2.3}$$

These equations are not sufficient. Closure laws are necessary to close and thus enable the determination of all unknown quantities embedded in the set of balance equations (field equations and discontinuity jump conditions) with the respective set of boundary and initial conditions (see [34] for more details). Due to the complexity of the two phase flows, these closure laws are empirical.

2.3.1.2 Three-field conservation equations

In some thermal hydraulic subchannel codes a two-phase, three-field formulation is realized. The three fields are vapour, continuous liquid and entrained liquid. The division of liquid into two fields is the only way to treat the different behaviour of liquid film and liquid droplets.

For the three-field description, codes work with 8 equations: 3 for each equation (2.1) and (2.2), and only two energy equations since the continuous liquid and the entrained liquid are assumed in thermal equilibrium. Additional assumptions and notations are required, that is why the detailed expressions of the resulting three-field equations are not given here but can be found in [15].

2.3.1.3 Control volumes

To solve the system equations, a space discretization is necessary. The degree of detail of the nodalization is governed by the user's wishes. The splitting leads to the representation of the geometry by a sequence of contiguous meshes, where the equations are solved using finite difference methods. Different meshes are used for the equation (2.1), (2.2) and (2.3). The Fig. 2-6 gives a plane representation of the different meshes. The central one is the mesh for the mass and the energy. The momentum equations are solved on staggered meshes where the momentum mesh cell is centred on the scalar mesh cell surface [3], [14]. This is also the case for the axial momentum which is staggered axially.

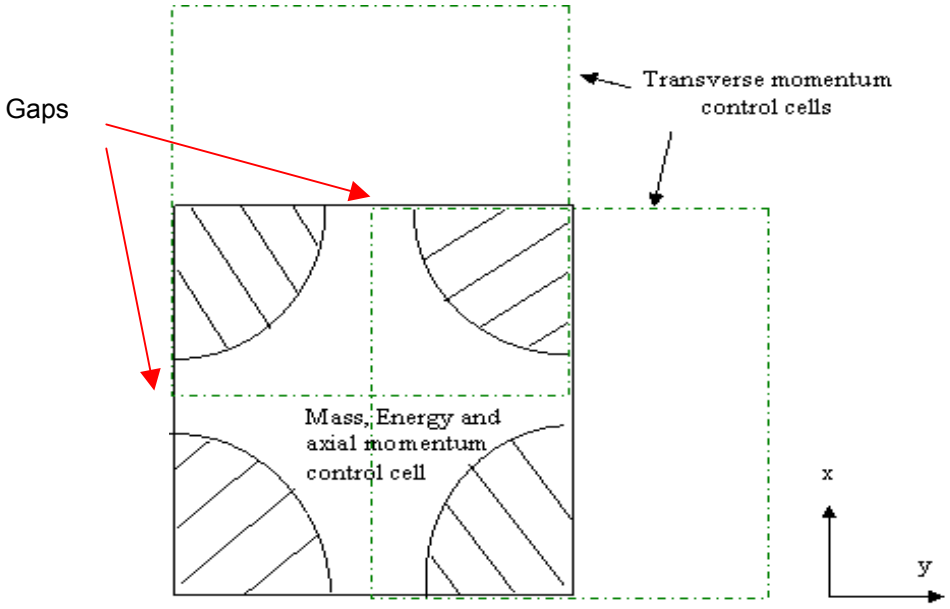


Fig. 2-6 Control Volumes

2.3.1.4 Subchannel approach

The finite-difference equations are written such that they may be solved on Cartesian coordinates or using the subchannel formulation. Codes such as COBRA-TF let the user the possibility to choose between both systems of description, but the subchannel approach is historically of high importance.

While choosing a control volume is not an approximation, the choice between 3-D and subchannel approach leads to approximations when selecting the second one. All transverse flows are assumed occurring trough gaps between fuel rods and regardless of the gap orientation. Thus, one transverse momentum equation is applied to all gaps, regardless of the gap orientation. This reduces the number of component momentum equations to only two: vertical and transverse. For this reason transverse momentum flux contributions in a 3-D application are not completely represented [17]. A discussion on the equivalence or divergence of the 3-D or subchannel approaches is presented in [15].

2.3.2 Heat transfer model

As already seen in Fig. 2-1, thermal hydraulic considerations begin from the energy production in the fuel material. This production is determined by neutronic calculations. However the fuel material is not put directly into water but it is enclosed by a zirconium clad. The architecture of a fuel rod is described in Fig. 2-7 and is determinant for the heat exchange.

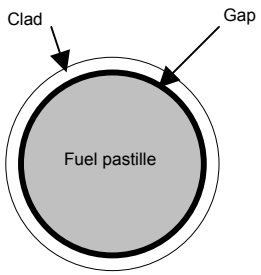


Fig. 2-7 Radial section of a fuel rod

The fuel consists usually of uranium oxide UO_2 and eventually some percents of plutonium oxide PuO2 (MOX). The clad is usually made of zirconium.

The heat transfer with the coolant depends mainly on 4 things:

- The energy production within the fuel
- The properties of the gap (material and geometry)
- The properties of the clad (material and geometry)
- The coolant itself (element, state) and the flow regime

2.3.2.1 Radial heat conduction in reactor fuel elements

A basic energy balance for a volume V at the place r in an arbitrary medium gives the equation (2.4) [11].

$$\underbrace{\frac{\partial(e)}{\partial t}}_{\substack{\text{time rate of}\\ \text{energy change}}} = \underbrace{q_{\text{int}}}_{\substack{\text{energy addition}\\ \text{due to heat sources}}} - \underbrace{div\,\varphi}_{\substack{\text{energy loss due}\\ \text{to heat transfer}}} \qquad (2.4)$$

e :energy density at the place r $\left[\frac{J}{cm^3}\right]$

q_{int} (r,t) volumetric fission heat source $\left[\frac{W}{cm^3}\right]$

φ (r,t) represents the heat flux density depending on the temperature $\left[\frac{W}{cm^2}\right]$

For steady-state considerations, the first term of (2.4) disappears and the other variables become time independent. Energy production and dissipation are equal and this leads to:

$$q_{\text{int}} = div\,\varphi \qquad (2.5)$$

We will now introduce an approximation (2.6) which is the direct analogue to the diffusion approximation used for simplification of the transport equation (see section 3.2).
The conduction within a solid follows the Fourier Law with a good approximation:

$$\varphi = -\lambda \nabla T \tag{2.6}$$

λ is the thermal conductivity $\left[\frac{W}{cm\,K}\right]$ T (r,t) is the temperature $[K]$

Due to the pastille geometries, the cylindrical coordinate system is the most appropriate to describe the conduction. The equations (2.5) and (2.6) in (2.4) leads to (2.7) [19].

$$\frac{1}{r}\frac{d}{dr}\left(\lambda r \frac{dT}{dr}\right) + q_{\text{int}} = 0 \tag{2.7}$$

q_{int} (or equivalent to the linear power) is input to thermal hydraulic software from neutronic codes for each fuel pin at each axial node.

The heat conduction model solves the equation (2.7) to determine the radial distribution shape of temperature within the fuel. Different approximations can be used and it is first important to remark that the thermal conductivity coefficient is not temperature independent. This consideration has particularly sense within the fuel where high temperature variations take place.

The equation (2.7) is solved also for the conduction domain within the clad thickness (q_{int} =0) and for the gap domain where q_{int} is zero and heat radiation models are added [15]. A typical radial temperature distribution within a cylindrical fuel element is shown in Fig. 2-8.

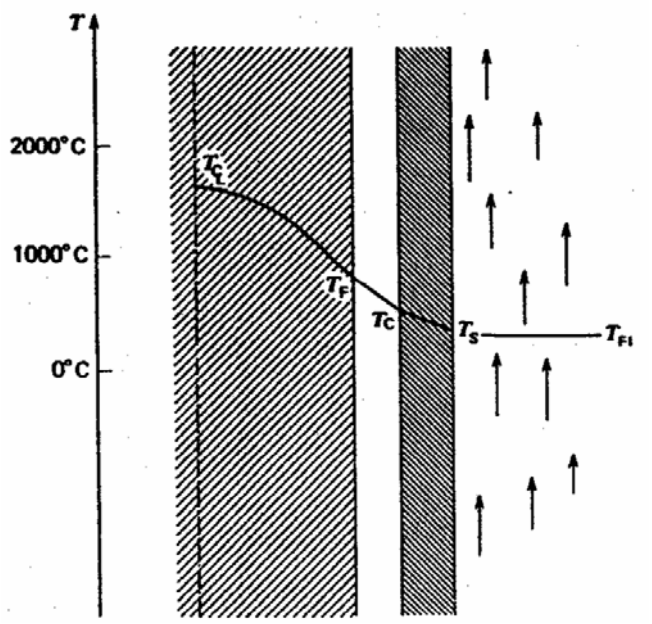


Fig. 2-8 Temperature distribution in a cylindrical fuel pin [11]

2.3.2.2 Single phase forced convection heat transfer

A forced convection of the primary fluid is the better way to have a high heat transfer between the pin cladding and the coolant. The higher is the velocity of the fluid the higher is the

heat transfer. The fluid flows are usually turbulent and the physical laws related to these are often empiric due to the complexity of the phenomenon.

The heat transfer is described by Newton's law of cooling [11]:

$$\varphi = h_s \left(T_s - T_{fl} \right) \tag{2.8}$$

h_s is the coefficient of convective heat transfer. Its magnitude varies strongly for different types of coolants and flow conditions $\left[\frac{W}{cm^2 K} \right]$

T_s: Temperature of the clad surface

T_{fl}: Temperature of the fluid

The objectives are now to determine the expression of h_s in the case of a forced fluid transfer (mass coolant motion induced by pumps in the primary loop). The fluid is assumed incompressible. The physical laws for flow studies and h_s determination were established for pipes with cylindrical cross section (diameter D). Frequently, and particularly for this case, it becomes necessary to extend these results to flow in channels of non-circular cross section. Then an equivalent hydraulic diameter D_h is defined by:

$$D_h = 4S / Z \tag{2.9}$$

S is the passage flow area
Z is the wetted perimeter of the flow

For the geometry of Fig. 2-5 we find:

$$D_h = d \left[\frac{4}{\pi} \left(\frac{p}{d} \right)^2 - 1 \right] \tag{2.10}$$

p is the square pitch

h_s is usually expressed in terms of the thermal conductivity of the fluid λ, D_h and the Nusselt number Nu as expressed in (2.11):

$$h_S = \left(\frac{\lambda}{D_h} \right) Nu \tag{2.11}$$

The Nusselt number can not be immediately deduced and is a function of two other dimensionless parameters:

- The Reynolds number (Re), which characterizes the conditions of the flow
- The Prandtl number (Pr) which characterizes the properties of the coolant fluid

Formula and explanations for these three numbers are not explained in more details here. The full expressions are detailed in appendix of [17].

Several of the more common correlations for the Nusselt number (and then h_s) are tabulated in Annex B for various ranges of Prandtl number as well as several physical properties of some typical coolants. The most common correlations used in reactor analysis is the *Dittus-Boelter correlation* [16] (for fluids such as water or helium where Pr≈1) directly introduced in (2.11) to obtain (2.12):

$$hs_{D-B} = 0.023 \frac{\lambda}{D_H} (\mathrm{Re})^{0.8} (\mathrm{Pr})^{0.4} \tag{2.12}$$

2.3.2.3 Boiling heat transfer

The precedent case was adapted to single phase liquid considerations. On the other hand the change of state of the fluid introduces another part in the heat transfer due to the nucleation. There are different methods to analyse this boiling heat transfer.

Depending on the temperature of rod and fluid different domains are encountered: single-phase forced convection, nucleate boiling, transition and film boiling. For two-phase flows, different flow pattern and hence heat transfer models exist as shown in the Fig. 2-9 and in Fig. 2-10 [18]. The domain of nucleate boiling can be divided into subcooled nucleate boiling, and saturated nucleate boiling. For normal working conditions in PWR, only the first two regimes are interesting. The maximal heat flux point is called critical heat flux. After this point the heat transfer coefficient decrease strongly and because of this it results a large augmentation of the surface temperature. This can lead to instable behavior and material damage.

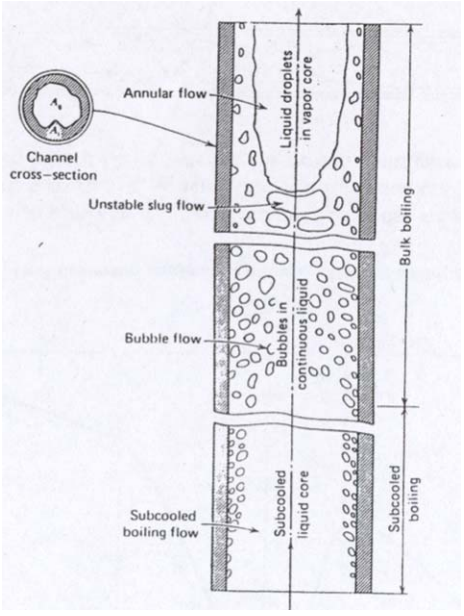


Fig. 2-9 Flow pattern in a heated channel

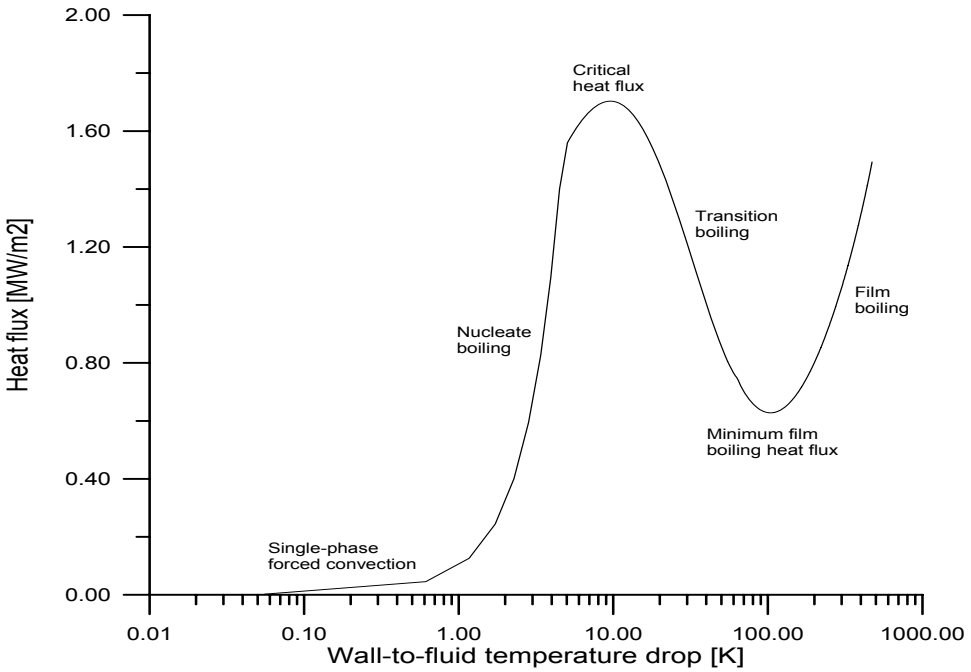


Fig. 2-10 Heat flux in function of the temperature difference

- Subcooled boiling

This flow regime is characterized by: $T_s \geq T_{sat}$ but $T_{fl} \leq T_{sat}$. T_{fl} is naturally a mean temperature across the section and the fluid temperature could be higher than T_{sat} in several places.

The equations of Jens and Lottes (2.13) or Thom (2.14) [20] can be chosen to describe the heat transfer in this domain of nucleate boiling even if the Thom's must be preferred [16]:

Jens and Lottes:

$$T_s - T_{sat} = 25\, e^{\frac{-p}{6,2}}\, \varphi^{\frac{1}{4}} \tag{2.13}$$

Thom:

$$T_s - T_{sat} = 22,7\, e^{\frac{-p}{8,7}}\, \varphi^{\frac{1}{2}} \tag{2.14}$$

T_{sat} is the fluid saturation temperature and p the system pressure (psia).

- Saturated or bulk boiling

Eventually sufficient heat is transferred to the coolant that reaches the saturation and begins bulk boiling. The equations of Jens and Lottes (2.13) or Thom (2.14) can still be used for the description of begin of the phenomena. The Chen Correlation is however more adapted: It considers a dual contribution of nucleate boiling and convection in the heat transfer coefficient [20].

The nucleate boiling part is based on the Forster-Zuber equation [16] and the convective part is based on the Dittus-Boelter equation (2.12). The Chen correlation, though developed for saturated boiling, may be extended into the subcooled region too, as it is the case in COBRA-TF [15].

2.4 Choice of code for PWR subassembly calculations

Safety relevant parameters such as clad temperature or coolant properties have to be investigated pin wise instead of assembly wise. This allows an accurate determination of critical domains within the assemblies. A pin by pin feedback modelling of the assembly in steady-state requires the choice of codes describing equivalent scale for the thermal hydraulic and the neutronic parts.

According to [18], the choice of COBRA-TF for the thermal hydraulic part of the coupling is convenient since it is a powerful subchannel tool with enhanced physical models for flow regime investigations. At contrary the configuration of COBRA-TF is quite difficult due to the size and the rigidity of its input file.

It is relevant to remark that COBRA-TF does not directly perform steady-state calculations, and that seems to be contradictory with our problem. On the other hand it was shown in [18] that after a certain time, results can be assumed as stationary. All COBRA-TF runs were performed with a calculation time of five seconds to reach this stationary state.

3 Neutron physics calculations

3.1 Introduction

Neutron physics calculations are the basis of the geometry determination for the core and the components. They also make it possible to study the limits of exploitation according to safety criteria. The actual methods of calculation use advanced modern calculation resources. This is only possible since the development of the computer power.

The state of a nuclear core is described by two main relevant parameters [21]:

- The multiplication factor: this is a mathematical tool to express the divergence of the neutron population to an equilibrium situation.

- The power distribution: it mainly describes macroscopically the repartition of fission events in the multiplication domain at every time (a little part of the power production is also due to other phenomenon such as scattering). This parameter depends on material properties and on the neutron distribution (space and energy).

Actually the power distribution is the central preoccupation of nuclear reactor theory, and moreover in this work since it is the principal interface for the coupling with the thermal hydraulic module. The knowledge of the repartition of the neutron population is of primary importance to be able to deduce the power distribution. The way to calculate this repartition is presented in the following part. This section was written on the basis of [9], [11] and [21].

3.2 The Boltzmann Equation

Since they are very numerous, the neutron population can be described by a density. This density depends on the parameters space ($\vec{r}$), speed (value and direction) and the time. For practical applications the neutron flux defined in (3.1) is used instead of the neutron density.

$$\Phi\left(\vec{r}, v, \vec{\Omega}, t\right) = v * n\left(\vec{r}, v, \vec{\Omega}, t\right) \tag{3.1}$$

Φ is the neutron flux and n the neutron density dependent on 7 independent variables : 3 space variables, two solid angle variables ($\vec{\Omega}$), the speed module v, and the time t.

The Boltzmann equation (3.2) [21] or neutron transport equation describes the behaviour of nuclear systems. It can be derived by considering an arbitrary volume and balancing the various mechanisms by which neutrons can be gained or lost within the volume.

$$\frac{1}{v}\frac{\partial\Phi(r,E,\Omega,t)}{\partial t} = \underbrace{-\Omega\nabla\Phi(r,E,\Omega,t)d^3rdEd^2\Omega - \Sigma_t(r,E,\Omega,t)\Phi(r,E,\Omega,t)d^3rdEd^2\Omega}_{A_0\Phi}$$
$$+\underbrace{\left[\int_0^\infty dE'\int_{4\pi}\int d^2\Omega'\Sigma_s(r,E'\to E,\Omega'\to\Omega,t)\Phi'(r,E',\Omega',t)\right]d^3rdEd^2\Omega}_{P_0\Phi} \qquad (3.2)$$
$$+\underbrace{P(r,E,\Omega,t)+Q(r,E,\Omega,t)}_{\xi}$$

This equation contains both derivatives in time and space as well as integrals over angle and energy and is so qualified as integrodifferential.

$\Sigma_s(r,E'\to E,\Omega'\to\Omega,t)$	differential transfer cross section
$\Sigma_t(r,E,\Omega,t)$:	total cross section
$P(r,E,\Omega,t)$:	fission sources
$Q(r,E,\Omega,t)$:	other sources

A_0:	Absorption and leakage operator
P_0:	Production operator within the multiplicator medium.
ξ :	Source term

For steady-state conditions, the neutron balance equation becomes (3.3) and is inhomogeneous.

$$A_0\Phi + P_0\Phi = -\xi \qquad (3.3)$$

With appropriate simplifications and hypothesis the transport equation can be simplified in a homogenous one (3.5). The transformation is based on the supposition that the imbalance between the absorption and production terms (the inhomogeneous term ξ) can be expressed proportionally to the flux (3.4).

$$\xi = \rho\, P_0\, \Phi_0 \qquad (3.4)$$

ρ is the reactivity

$$A_0\,\Phi_0 + \lambda_0 \mathrm{P}_0\,\Phi_0 = 0 \qquad (3.5)$$

$\lambda_0 = 1-\rho$ where ρ is the reactivity
Φ_0 is the fundamental mode of the neutron flux Φ

3.3 Methods of solution

It is usually impossible to solve analytically the Boltzmann equation due to the dependence in space, energy and angular direction. The only way is to look for approximate methods to obtain convenient results for specified aims. Several numeric methods provide this possibility and the two classes are the deterministic methods and the Monte Carlo methods. These methods are presented in the current section.

3.3.1 Monte Carlo methods

Monte Carlo methods are based on the study of stochastic phenomena. The behaviour of a population of neutrons is simulated and observed from the birth to the disappearance. The journey (route) of each neutron follows the probabilistic laws of collisions and free flights. These observations are conducted for a huge number of neutrons lives and the value we want to determine becomes the mathematical expectation of a random variable. The large number theory proved that the accuracy of results increase with an increase of number of histories. MCNP [22] for example is a Monte Carlo analysis code used in FZK. Tripoli [23] is used in France while VIM [24] and MCNP are frequently used in the USA.

The approximation of this method consists in the probabilistic approach of the results, but at contrary, the dependencies towards energy, space and angular direction are considered continuously. The geometry can be exactly described even for the most complicated ones and these are the major advantages of this method.

On the other hand their computation times are quite expensive. Moreover, some problems can occur when trying to handle large cores in 3 dimensions where convergence problems were detected [7].

Monte Carlo calculations are frequently considered as reference calculation and are applied for verifying results from deterministic calculations and for the validation of new codes.

3.3.2 Deterministic methods

As already said it is impossible to solve directly the Boltzmann equation. Deterministic methods are an alternative to Monte Carlo methods and are in this section briefly presented. The concept is to obtain an equation system from the continuous Boltzmann equation. This is can be carried out by discretization in space, angular direction and energy.

3.3.2.1 Energy discretization

Problems of the shape of neutronic spectra require the splitting of the energy range covered by neutrons in some discretized energy groups. This discretization is based on the multi-group theory and treats usually energies from 10-20 MeV to 10^{-3} eV in KAPROS-E for example. This range is split into N groups of decreasing energies where the N number depends on the type of calculation and of the required accuracy. An example of this splitting is given in (3.6).

$$E_1 \geq E_2 \geq ... \geq E_g \geq e \geq E_{g+1} \geq ... \geq E_{N+1} \quad (3.6)$$

E_i are the boundary energies of the range discretization
e is the energy of considered neutrons

The neutrons of energy e are described being a part of the group g and are then considered as monoenergetic neutrons. The energy dependant problem is in this way transposed into a system of coupled equations. Each equation is now energy independent.

3.3.2.2 Angular direction discretization

The Boltzmann equation can be expressed in two equivalent formulations: integral or integrodifferential (3.2) [25] .From these two forms it is possible to treat the angular discretization in different ways.

- Integral approach: a common method is the “first collision probability method”. The space is cut into pieces of volume Vi and these volumes are considered homogeneous. The probability that a collision occurs in the Vj volume is then calculated, for neutron born in Vi or which energy or direction was modified in Vi due to a past collision. A mean flux can next be calculated in each mesh. A complete development is available in [26].

- Integral differential approach: the space is cut into many sectors taking in count of the neutron speed distribution after a collision or their birth. This method is used by the neutronic module and it will be explained in the following development.

  - Development on spherical harmonics or P_N method:

    With this method the flux is expanded on a base of appropriate functions (Legendre polynomials), the spherical harmonics, to obtain a system of differential equations, as for the discretization in energy. The spherical harmonics are an orthogonal set of solutions to the Laplace equation represented in a system of spherical coordinates. These are the three-dimensional equivalents of the trigonometric functions used in the development in Fourier series. The P0 approximation is in fact the diffusion approximation.

  - Discrete ordinates approximation

    Still belonging to integrodifferential methods, the discrete ordinates approximation is another solution for treating the angular variation of the particle distributions. The infinite direction represented by $\vec{\Omega}$ in the balance equation is replaced by a finite number of chosen directions (N). Each direction is not equivalent due to the geometry of the problem, thus weighting parameters as ω_i for the ith direction are used to weight the part of the flux in the ith direction Φ_i to the whole flux.

$$\int_{4\pi} \Phi(r,E,\Omega)\, d\Omega = \sum_{i=1}^{N} \omega_i \Phi_i (r,E,\Omega_i)$$

The evolution of the flux is assumed linear between the discretized directions for such kind of methods. The accuracy grows up with an increasing direction number N, but the complexity grows at the same time. Generally these methods are chosen with N= 4 to 16. If a linear evolution of the flux for the space and for the angular direction is assumed, on each discretization range, the numerical diamond scheme method can be employed [13]. The Chebyshev acceleration method is often used to accelerate the inner and outer iterations.

3.3.2.3 Space discretization

With deterministic methods the distribution of neutrons is calculated by transforming the transport equation into an equation system. The model area has to be divided spatially in sub regions defining calculation meshes or calculation nodes. This space splitting process could be done more or less roughly and offers then a large panel of possibilities to optimize the computer costs as function of the needed accuracy. The simplification can be pushed until “fundamental mode” calculation for a reactor system where space independence (0-D consideration) is presumed.

The calculations are often carried out under the diffusion approximation for large cores with only few energy groups. But in many cases the diffusion approximation is not accurate enough and transport codes are preferred when the problem specifications are compatible.

Several methods are available for the discretization stage [9]:

- The finite element or nodal method. The reactor is cut into quite big pieces as shown in Fig. 3-1 [11]. The flux is written as development of polynomial functions and a solution can be numerical calculated with orthogonal projection on the boarders of the meshes.

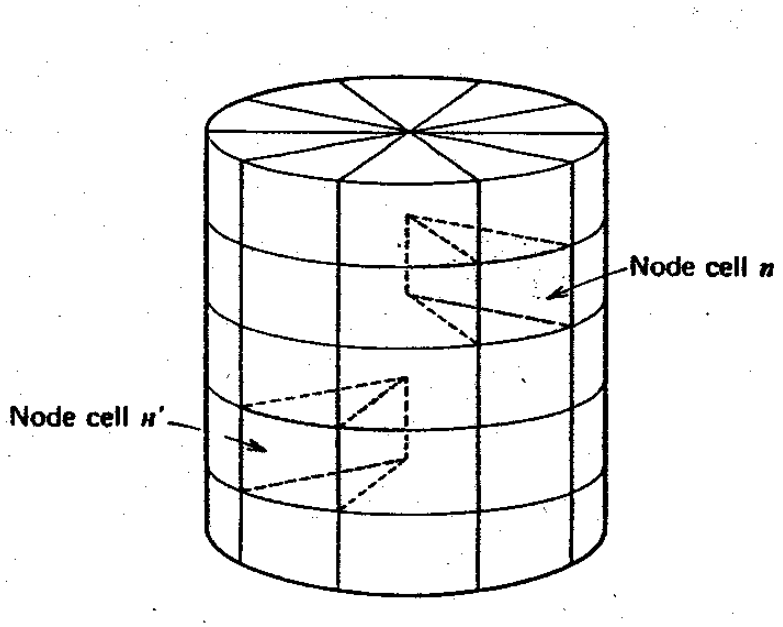


Fig. 3-1 Nodal cell division of a reactor core

- Finite difference methods which evaluate the neutron current from a mesh to its neighbour in a specified direction in accordance with the Fick law. Most diffusion and transport programs apply this method tough the time spent is often quite important.

3.4 Separation of the calculation steps

The A_o and P_o spatial and energetic components must be known to be able to solve the equation (3.2). The determination of these components is very complicated due to the energy depending process of neutron interaction with the material, and due to geometric configurations. To handle with this it is allowed to separate both heterogeneity levels into two treatments: cell codes and flux solver codes also called core codes. These are presented in the present section to explain their differences and complementarities.

3.4.1 Cell codes

These codes represent the smaller level of the discretization scale. They handle with phenomenon appearing between neutrons and materials or neutrons together for very small domains. The characteristic dimension is the mean free path.

- They are handling complex geometries and very high heterogeneities in the material properties. Due to this fact, a high level of discretization is required for space and energy.

- They solve the neutron balance equation in its transport integral form ideally in two or three dimensions, but 0 or 1 dimensional calculations are more often applied for symmetry reasons. At the same time they allow the choice of convenient simplifications for the solver to adapt computing time to the needs.

- They deal with the considering of self shielding resonances and the treatment of multiple resonances.

- They produce multiparameter tables for the correction of cross-sections according to several environments parameters. These are for PWR mainly the coolant density and temperature (since it is the moderator too), the fuel temperature (Doppler Effect) and depletion. This correction can not be bypassed for accurate results since the codes are solving the neutron balance equation in steady state. Between each time step the data stored in tables can be updated by interpolation methods.

These cell codes request microscopic cross sections, isotopic concentrations and material properties information. It is compiled in libraries (ENDF-B, JEF) delivered by international organizations. The data contained in these libraries are however not usable directly by the cell programs and must be pre-processed. The preliminary treatment deals with the implementation of resonance parameters, the gathering in groups and the condensation in energy.

The build libraries have to be then converted to the adapted format according to the cell code requirements. This job can be processed at the FZK by the program couple NJOY-GRUCAL.

The macroscopic cross sections necessary for solving the transport equation are prepared by cell codes trough homogenisation and condensation steps for the considered meshes and energy groups. For cross section processing, an own developed program GRUCAL is used at the FZK to perform macroscopic cross sections calculations. The main characteristic of this procedure is that calculations are very flexible since they are only steered with information given by the user in a control file (Fig. 3-2 adapted from [27]).

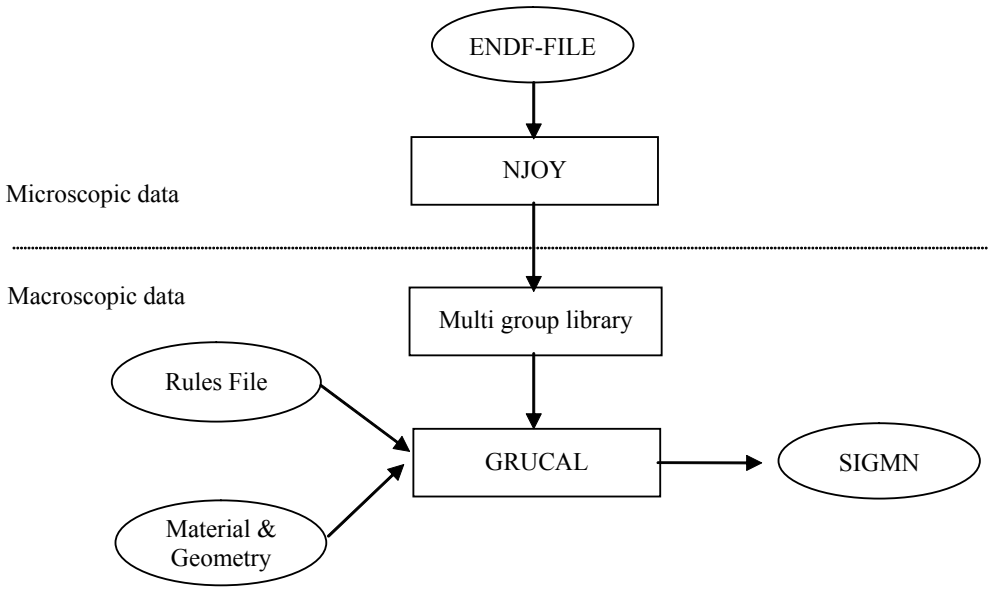


Fig. 3-2 From microscopic cross sections data to macroscopic for a direct use in neutronic codes. Example in KAPROS-E

3.4.2 Flux solver

These kinds of codes calculate the existing coupling between the different core regions. The coupling between the regions is mainly produced by prompt neutrons which have higher travel capabilities due to their low sensibility to local effects (see cross section evolutions in function of the energy). The description of the whole system can be simplified in the geometric representation and in the energy discretization. Calculations are usually performed in 2 or 3 D with up to 69 energy groups depending on the model domain, even 2 or 1 energy groups are sometimes sufficient. These flux solvers usually utilize cross sections prepared by cell code computations.

The both approximations of time independence and of the knowledge of the operators P_o and A_o for each step required other features on the core code.

The core codes can include evolution modules to update the composition of the fuel during the burn up. These codes solve the nuclide evolution equations (specific development can be found in [26]).
They can contain also special modules to furnish information for the definition of interpolation parameter (see section on cell codes). The multi-parameter tables are prepared by the cell

codes but we need information on the power production and on the medium properties to effectuate the interpolation and to apply the cross-sections computed in cell codes. Sometimes, some codes which simulate simplified thermal hydraulic calculation are employed to feed information back to the interpolation algorithms. We see here the pertinence of the coupling between real thermal hydraulic investigation tool and the neutron calculations to improve the accuracy of results. The data computed by the thermal hydraulic code permit the updating at each time step and each calculation node the components of the matrix operator P_o and A_o.

The scheme 3-3 shows the arrangement of data pre-processing, cell code calculation, cell homogenization, flux calculation on core or assembly level and coupling with the thermal hydraulic to feed back the cell code again.

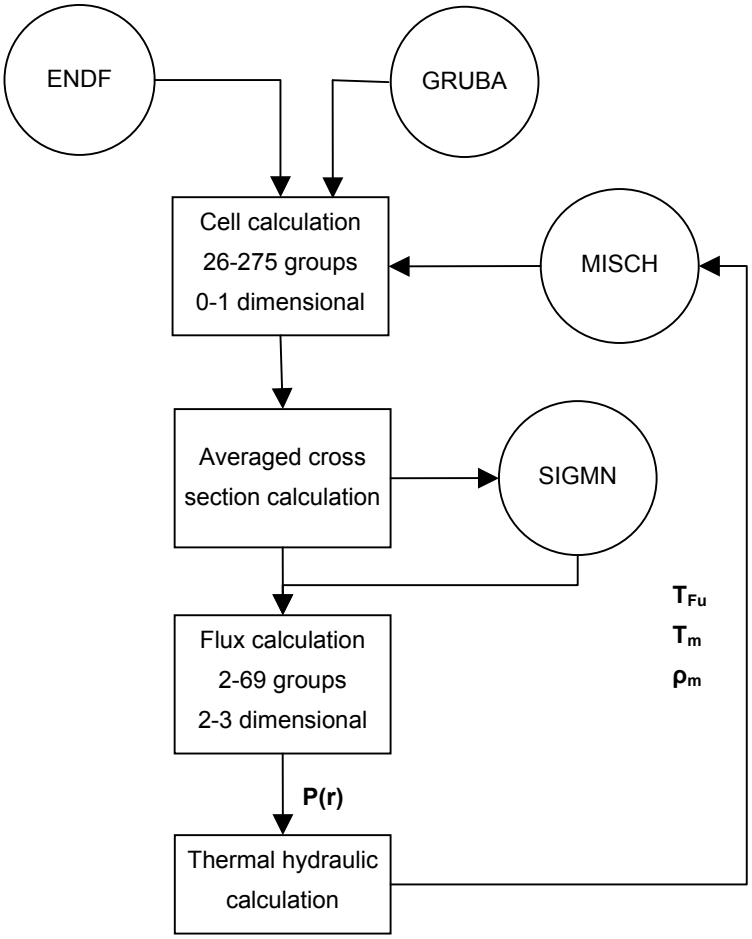


Fig. 3-3 Cross sections updates with thermal hydraulic calculations

3.5 Choice of models and codes for PWR assembly calculations

The present problem concerns the optimized description of material and coolant properties for a square assembly on the pin level. The evolutions of temperatures and density of coolant are computed by COBRA-TF for each pin, along the height. A three dimensional representation and computation of the assembly is then necessary.

Deterministic and Monte Carlo methods are both able to solve the neutron balance equation in 3D. Deterministic calculations based on the S_N method solve the transport equation with good agreement to Monte Carlo results and are not so much time expensive.

The 3 dimensional discrete ordinate transport code DANTSYS (THREEDANT) was selected as flux solver on the assembly level with a S_8 approximation. This is possible due to the limited perimeter of the problem. Since the assembly is square shaped (Fig. 2-5), a resolution using the (x, y, z) Cartesian ordinate system is the most convenient.

On the unit cell level, the method of collision probability (module WEKCPM) is used in one dimension (due to the symmetry resulting of the transformation in Wigner-Seitz cells) to determine the flux and to calculate for each cell the corresponding macroscopic cross section. The method of collision probability needs smaller computation times than S_N methods. The heterogeneities within a fuel unit cell are so homogenized and macroscopic cross sections are determined for the flux solver code.

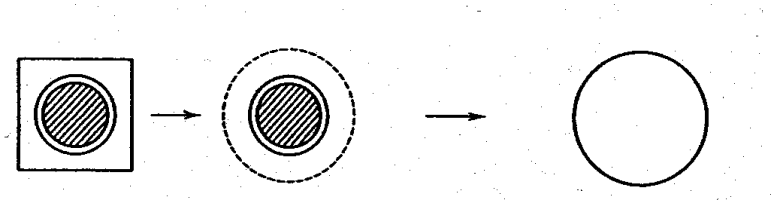


Fig. 3-4 Homogenisation steps for preparing the flux calculation on the assembly

The calculations are performed with a 28 group library, carefully collapsed from a 69 group master library based on ENDF/B-6.5 evaluated data. Exploratory investigations showed that a calculation with 28 energy groups is reasonable and provides accurate results [33].

4 Coupling of thermal hydraulic and neutron physics codes

The work in reference [18] led to the conception and the realisation of two interface programs in the C++ language for the coupling. A first integration of the coupling in the KAPROS-E system via the new procedure COBRAP was achieved, and first results were obtained.

For a more convenient integration of the coupling in KAPROS-E, and since most of the procedures are written in FORTRAN, a new implementation of these procedures in FORTRAN was developed with some major improvements.

The new coupling is able to handle square shaped assemblies of different sizes, but only if the edge pin number is pair. In fact the consideration of assemblies of impair side pin number is more problematic and is not yet treated.
Furthermore flexibility regarding the axial discretization was introduced (it must naturally correspond in KARBUS/DANTSYS and COBRA-TF).

The following section presents first briefly the programs chosen for the coupling, and its realization will be then discussed. The source codes are available in Annex J and Annex K

4.1 Selected codes overview

4.1.1 The program system KAPROS

This part gives a survey of the KAPROS system, a modular system for any kind of reactor physical calculations [4], [9], [29].

4.1.1.1 History

Since the late fifties the Institute for Neutron Physics and Reactor Dynamics (INR) was involved in the conception of calculation codes for reactor simulation. This development has been following the evolution stages of computer.

The improvements of theoretical approaches led to the implementation of corresponding calculation codes for several problems. At first, those were standalone programs but quickly the necessity of interaction was perceived to achieve complete studies. The increasing number of possible models for one real problem made impossible to implement each time a new coupling. Information transfer between each program part had to be also performed manually what was time expensive. The non-existence of standard interfaces made the cooperation between programs increasingly difficult and was an error source.

The KAPROS program is born to solve this problem. It was necessary to create a program system to organize the programming efforts of the institute, and to integrate also renowned international codes. An important purpose was flexibility. The first version of KAPROS appeared in the early seventies as the successor of NUSYS (Nuklearprogrammsystem) of

1965. The basis development took circa 5 years and the first versions of KAPROS were strongly relying on the capabilities of IBM mainframe computers at FZK. In the mid-nineties, the transfer to UNIX workstations became reality and further improvements made it possible to work nowadays on personal computer (under Linux) with the current version (KAPROS 2.02) of KAPROS. The extended version of KAPROS, KAPROS-E, is used in this work [28].

4.1.1.2 Purpose

The implementation of this environment software was not only dedicated to the institute and to the nuclear domain, but the original aim was to produce a strong tool which could be able to handle with every physics and technical domain. In the following the main goals of this enterprise are summarized [4]:

- Flexible and independent use of each incorporated calculation software
- Facilitated information transfer between entities for run or memory purposes
- Appropriate environment for the incorporation of new calculation software
- Maintenance facilities
- Exhaustive documentation

4.1.1.3 The Unix/Linux version of KAPROS

The UNIX version of KAPROS consists of two main parts [29]:

- A KAPROS kernel interacting with
- Procedures, modules and associated libraries

The kernel contains all the general management system routines that make possible the execution of modules. The KSP (KAPROS Steuer Programm) can be used for any task. It governs the information flow by using the so called “Lifeline” to store data.

The “Lifeline” contains:

- The internal lifeline in fast memory working with the transfer of data between modules and lifeline
- A pointer lifeline in fast memory to allow the modules to access to data without data transfer
- An external lifeline on disk storage

All this dataflow is stored in “databloc” which can be used in the future. In the current extended version of KAPROS, KAPROS-E, a lot of couplings with international stand-alone codes have been realized and the subject of this work gives a new example.

4.1.1.4 The procedure KARBUS

Essentially one procedure of KAPROS-E was used for the present work: KARBUS (Karlsruhe Reaktor Burnup System). This sub system was developed for burn up and depletion calculations since the early eighties. As further development of DXBURN, the aim was also to dispose a flexible method to allow calculations with multiple cross-sections programs for reactor investigations and to enable burn-up calculations for reactor systems or unit cells. The basic conception architecture of KARBUS is presented in [9] p.150.

4.1.1.5 Input of KARBUS

A KARBUS calculation job must preliminary be preceded by a preparation task. As well as for other KAPROS-E procedures an input file has to be created (Annex C). This file contains the settings for each calculation step inside the procedure (geometry, material properties, solver options…). The processing of the modules produces a lot of files and the most relevant for this study are the processing information file (saved at the end of the module execution into the name OUTPUT.karbuse), and the “ks_cobra.dat” file reported in Annex E. This last file is edited by KARBUS especially for the coupling with COBRA-TF. It summarizes in a convenient form results from the KARBUS calculation needed by COBRA-TF for the coupling problem.

4.1.1.6 Geometry description

The problem geometry is specified in the input file of KARBUS (see Annex C). Due to symmetries only a quarter of the assembly is modelized. The assembly, as repetition of the same pattern, is cut into computation cells which can be fuel cells (fuel pin, gap, clad and moderator see Fig. 3-4) or water cells, and the axial height is nodalized too. This is set with the input bloc for “MIXCOP” where the value “1” describes a fuel cell and “2” is for a water cell as defined in input bloc “NDCALC”.

The cell numbering system used in KARBUS is presented in Fig. 4-1 for the first section of a quarter of an 18*18 assembly. The first section is at the top of the assembly, the last at the bottom, e.g. we will have 810 computation cells for an 18*18 assembly split up axially into 10 levels, and the 810^{th} is at the bottom of the assembly.

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Fig. 4-1 Numbering system in KARBUS/DANTSYS for an assembly quarter (18*18)

4.1.2 The thermal-hydraulic subchannel investigations tool COBRA-TF

As already explained in section 2, a subchannel code handles the heat transfer inside fuel pins, treat the heat transfer between fuel rods and coolant, and determines the flow regime in the coolant channels.
The code **COBRA-TF** belongs to this category of tools. COBRA-TF stands for **Co**olant **B**oiling in **R**od **A**rrays**- T**wo **F**luids. It was developed at the Department of Energy's Pacific Northwest Laboratory to provide best-estimate thermal-hydraulic analyses of light water reactors. It was particularly designed to treat loss of coolant accident (LOCA) of PWR. The original version of the code was modified during the FLECHT-SEASET experimental program [30] to enhance its predictive capability for reflood transients. Other new features were also added as grid spacer effects and sub-channel thermal radiation treatment.

Technically, this code provides a three-field hydrodynamic representation of two-phase compressible flows where the two fluids are water and steam and the three fields are the liquid film, the liquid droplets and the steam. Each field study can be characterized in three dimensions and is time dependent.
It is possible to use COBRA-TF in two different coordinate systems by input: a rectangular Cartesian one, or a historically relevant subchannel coordinate system. This choice affects in a certain measure the physical equations too (see section 2.3.1.4).
The three-dimensional Cartesian form of the transverse momentum equation was chosen for this work since cross flows play a relevant role for PWR calculations.

Coupled to the hydrodynamic calculations, the heat transfer model determines rates used in the energy equations. It must describe as good as possible what happens during the operation of the plant. The global heat transfer model implemented in COBRA-TF includes 4 sub models:

- Conduction model

- Heat transfer package: selects and evaluates the appropriate heat transfer correlations

- Quench front model: a “fine mesh-rezoning” method which calculates quench front propagation due to both axial conduction and radial heat transfer

- Gap conductance model

Theoretical backgrounds of these models are described in the section 2.3.

4.1.2.1 Interface of COBRA-TF

INPUT:

COBRA-TF needs an input to set the problem geometry and the calculations adjustments. Numerous options are selectable and can be directly enabled by the correct definition of the input file “deck.inp”. It is composed of different groups with separated aims. Basically the main categories are:

- setting up the geometry to model the system

- specifying the fluid conditions and forcing functions to define the state of the system

- setting some other parameters for running the code and interpreting the output

A complete description of this input file can be found in the section 3 of reference [3]. The input file must exist in the directory from which the executable COBRA-TF program is called.

OUTPUT:
On the other side, results of the COBRA-TF processing are written into two output files: deck.run and deck.out. This last file “deck.out” is for our discussion of most interest since it contains the results of the processed investigation. The output files are written in the call directory of the executable program of COBRA-TF.

Approximately, the output file summarizes at first the input file and the deduced conditions, as well as tables for geometry and material properties. It contains then the results of the calculation according to chosen options.

The interesting calculated values are:

- Channel results: pressure, velocity, void fraction, flow rate, enthalpy, and density for every subchannel number.

- Nuclear fuel rod results: fluid temperature, surface heat flux, clad temperatures (inside and outside), and radial fuel temperatures from the centre to the surface. These values appear for each rod number and each rod surface, this for each simulation time.

The decomposition of each rod in surface is relevant to treat the transfer with channel since each surface of a rod is in front of a different channel.

COBRA-TF has the capability to work with two systems of units for its input file: the British unit system, and the metric unit system. The choice between the two systems is made in the first group of the input file (parameter ICOBRA). If the metric system is chosen, the code will convert the data in British units. For convenient way of use, the metric unit system was chosen (ICOBRA = 1).
However the produced output file contains results in British system units.

The underlined data above are required for the coupling problem, even if they are not in a correct form yet.

4.1.2.2 Geometry representation

Contrary to KARBUS/DANTSYS where a quarter of the assembly is modelized, only a eighth of the same assembly is investigated with COBRA-TF. This is possible due to the diagonal symmetry existing within a quarter (see Fig. 4-2).
The spaces between rods, the subchannels, are numbered and are axially nodalized. The number of axial node is set in the input of COBRA-TF with the parameter NONODE. The first node in COBRA-TF is at the bottom of the assembly and the last node model the last upper section of the assembly.
The numbering system encountered in COBRA-TF is presented in Fig. 4-3 and Fig. 4-4 for a eighth of an 18*18 assembly.

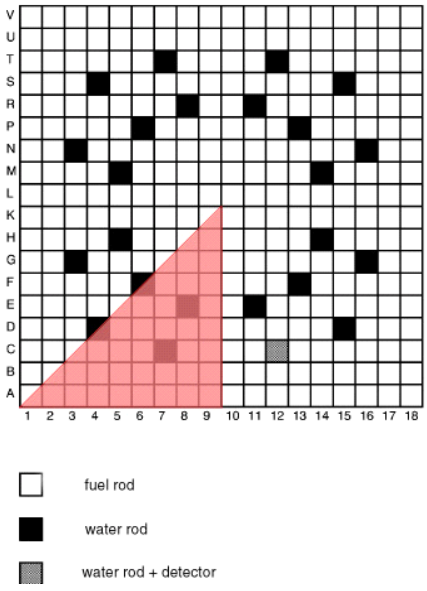


Fig. 4-2 Example of a section of a PWR assembly

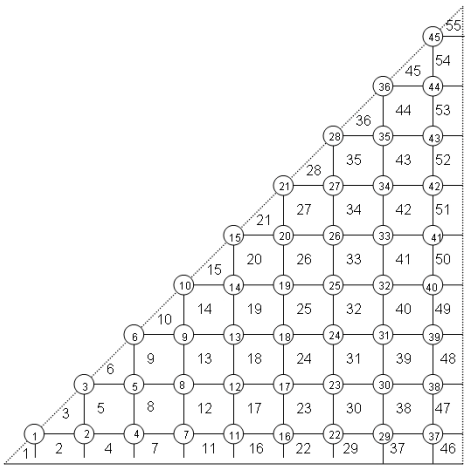


Fig. 4-3 COBRA-TF rod and channel numbering system

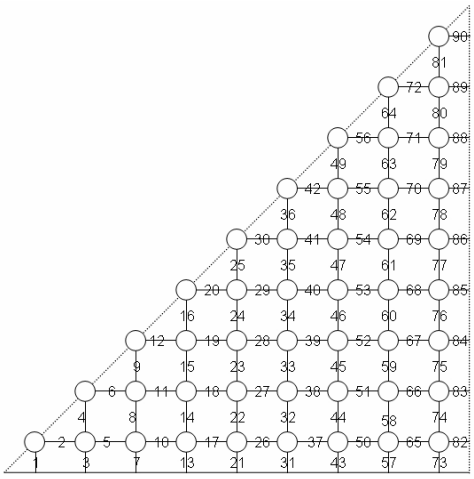


Fig. 4-4 COBRA-TF gap numbering system

4.2 Coupling flow chart

Regarding the intention to couple the neutronic and the thermal hydraulic analysis tools, interface programs have to be written to define and organise properly the information flows from a code to the other. For the present coupling, two programs are required:

- A first one to treat data from the KARBUS/DANTSYS calculation, to analyse them, and to transfer the results to COBRA-TF via the COBRA input file.

- A second one for closing the loop and giving back results from the processing of COBRA-TF. The file print by this program will serve as new starting point for a next KARBUS/DANTSYS calculation.

The Fig. 4-5 gives a general overview of the coupling.

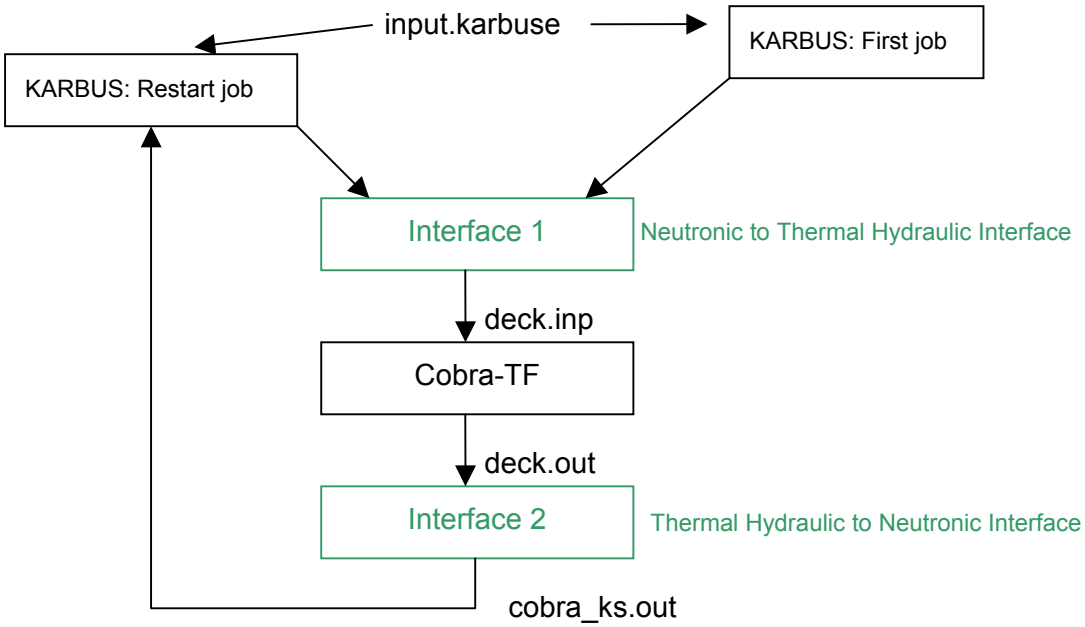


Fig. 4-5 Principal flow chart for the coupling program in KAPROS-E

4.3 The neutronic to thermal-hydraulic program interface: kcntti

The present part provides a description of the program “Interface 1” defined in Fig. 4-5 for linking KARBUS to COBRA-TF.
“kcntti” stands for “**K**APROS-E **C**OBRA **n**eutronic **t**o **t**hermal-hydraulic **i**nterface”. In this section the purposes of the program are described as well as the ways used to fulfil these tasks.

4.3.1 Description of the tasks

A start input for COBRA-TF must exist to set the main and invariable parameters. This invariant template file is named “struc.dat”. It contains already the geometry settings for the considered problem and the solver choices. But at each pass this file must be completed taking into account the numerical results computed by KARBUS. They affect the average linear heat rate per rod (AFLUX) in the Group 1 of the input deck of COBRA-TF and the relative power factor for each rod and node in the group 11 (see the description of the input file of COBRA-TF in section 3 of [3]).
The file struc.dat is kept unchanged but copied into a new file named deck.inp while adding the modifications. The new file created is then read by COBRA to perform the new thermal hydraulic calculation.

Summary of the tasks:

- Open and read definite parts of the file ks_cobra.dat. Store the read values in matrixes for future calculations.

- Proceed further with the calculation of “Afflux” (group 1) and of the axial power tables (group 11)

- Read the structure file, recopy it into “deck.inp” and add results from the precedent step.

4.3.2 Dependencies

kcntti requires three input files and produces an output file named “deck.inp“.

Input:

- input.kcntti: It allows the change of the names of I/O files. These names must as matter of course correspond with the names used by KARBUS and COBRA-TF. The names used by default are specified in the example below.

  Example of an input file for kcntti:

```
Kcntti input file

'ks_cobra.dat     '        „Specify the name of the output file from KARBUS“
'deck.inp         '        „Specify the name of the output file to CobraTF“
'struc.dat        '        “Specify the name of the setting file”

****
```

- struc.dat:

This file is the skeleton of the future deck.inp file. Since this file contains problems description parameters, it is also used by kcntti to configure itself with the problem specifications as assembly size, number of axial level.

- ks_cobra.dat:

Its contains some constitutive data and power results for each calculated cell from the first cell at the left upper corner at the top of the assembly down to the last one at the right down corner at the bottom. The average linear power given in the sixth column is relevant and will be given to COBRA in another form.

Output:

- deck.inp

Input file of COBRA-TF.

4.3.3 Program layout

kcntti consists in a main program file and 4 modules. It has been written in Fortran 90 at first under the Windows OS with the Compaq FORTRAN compiler. The transfer under Linux was realised with the Lahey compiler “lf95”.

4.3.3.1 Architecture description

- “kcntti.f90”

Main program. Call the other modules and subroutines in the correct order.

- module “endoffile”

It secures the portability of the program. In the case of an “End of File” error during the read phase, a definite value is given to a debug parameter which can differ on different systems. The subroutine tests a scratch file to set the value of this parameter and to identify the end of a file without causing troubles.

- module “definition”

This module centralises the declaration and initialisation of the main relevant variables, notably the matrix whose definitions must be defined at each run, according to the size of the model. This module and the corresponding subroutine are called at the beginning of the main program to set the variables.
The file struc.dat is read to configure the variables of kcntti, since it contains the parameters of the problem description (rod and channel number, number of axial levels…)
The subroutine set_connex defines a matrix to establish the correspondence between the geometry model used by KARBUS and COBRA-TF.

- module “read_calc”

It reads the ks_cobra data sheet, and calculates the average linear power and the power ratios. The data flow is computed and stored into matrixes.

- module “write_deck”:

Read the struc.dat file and recopy it into deck.inp while adding the changes in group 1 and 11 (see section 4.3.1).

4.3.3.2 Calculation tasks survey

- Establishing the relationship between the eighth of the assembly simulated by COBRA-TF and the quarter simulated by KARBUS/DANTSYS:

  The Fig. 4-6 shows an assembly quarter. From geometrical arrangement of the fuel rod, the water rods and the mix rods (Fig. 4-2), a diagonal symmetry exists within this quarter as mentioned by the diagonal line.

  This example is based on an 18*18-24 fuel assembly; it means a simulated quarter contains 81 cells (9*9).

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Fig. 4-6 Quarter of an assembly for KARBUS and cell numbering

The eighth of the core simulated with COBRA-TF is represented in grey in the Fig stated below, and contains the rods from number 1 to 45.

37	38	39	40	41	42	43	44	45
29	30	31	32	33	34	35	36	44
22	23	24	25	26	27	28	35	43
16	17	18	19	20	21	27	34	42
11	12	13	14	15	20	26	33	41
7	8	9	10	14	19	25	32	40
4	5	6	9	13	18	24	31	39
2	3	5	8	12	17	23	30	38
1	2	4	7	11	16	22	29	37

Fig. 4-7 COBRA-TF simulated part of the assembly and completion to obtain a quarter

The starting point is the eighth of the assembly simulated by COBRA-TF.

The subroutine set_connex is deployed to build the other eighth of the core per symmetry, since KARBUS works with a whole quarter. The completion must be flexible and work with any size of square assembly if the boarder size contains a pair number of pin. For impair numbers the splitting of the assembly in quarter and eighth is more problematic and this aspect was not yet considered. The matrix “Connex(:;:)” serves the right addressing between the two models, e.g. the results from the calculation cells 31 and 51 of KARBUS must coincide with the results of the pin 19 in the cobra model, at the correct axial level too.

The construction principle is based on the fact that if x_{ij} are the elements of the matrix, and n is the number of edge elements for a eighth (or quarter), the following equation (4.1) is verified:

$$X_{i,j} = X_{(n-j+1),(n-i+1)} \tag{4.1}$$

With this formula the whole quarter can be emulated and is represented and the matrix “connex” is given in Fig. 4-7 for the chosen example.

- Calculation of the average pin linear power and of the power ratios:

  KARBUS provides the linear power with ks_cobra.dat for every computation cell. These results, thanks to the matrix Connex (:), are converted in results in the COBRA-TF numbering format. A matrix rod_pw (:,:) is created, where the first parameter is the COBRA pin number and the second one the axial level. An average linear power is from here easily calculated and each value of the matrix rod_pow is then divided by this mean value to give the power ratio used for the completion of the 11th group of the “deck.inp”.

4.4 The thermal-hydraulic to neutronic program interface: kcttni

“kcttni” stands for “**K**APROS-E **C**OBRA **t**hermal hydraulic **t**o **n**eutronic **i**nterface”.

4.4.1 Description of the tasks

kcttni correspond to the “interface 2” in Fig. 4-5 and is called after the processing of COBRA-TF for closing the loop.

The interface reads information from the output file from COBRA-TF “deck.out”. The data are then processed to convert them to the international unit system and to calculate average values for each cell. KARBUS will use these values later to start a new iteration with updated material properties.

Finally kcttni must deliver a data sheet with the following entries:

- H20 average temperature of the liquid phase [K]
- Clad average temperature [K]
- Fuel average temperature [K]

- H20 density [g/cm^3]

Because the file is dedicated to KARBUS, information must be given in the KARBUS numbering system. One line describes so a computation cell, beginning at the top of the assembly and finishing on the last cell at the bottom. To establish the correspondence between both numbering systems, the same subroutines are used as for kcntti.

The interface program kcttni has some other tasks but they are not direct in relation with the interface problem discussed here. Further explanations of the advanced features managed by kcttni will be discussed later in section 4.6.3.

4.4.2 Dependencies

kcttni requires three input files and produces an output file named “cobra_ks.out“.

Input:

- “input.kcttni”:
  This file specifies the kcttni working parameters. It allows changing the names of the I/O files. These names must as matter of course correspond with the names used by KARBUS and COBRA-TF.
  In comparison with the input file for kcntti, this file contains some relevant other entries for its configuration.

  Example for an input.kcttni file:

kcttni input file

'deck.out ' “Here must be set the name of the Output file from Cobra”

'cobra_ks.dat ' “Name of the file given to Karbus for a new job”

'struc.dat ' “Name of the configuration skeleton file for Cobra”

'5.0 ' “Simulation time chosen for the processing of kcttni”

****
****

' 58' "Weighting parameter for the part of the calculated distribution in the new distribution. Values can be entire values from 0 to 100 "

'0' Convergence criteria fulfilled?(0 or 1). Written here by kcttni. 0 must be the begin value

Maximum variation rate for each category of result at this Iteration: 0.0002 0.0004 0.0019 0.0002

The entries represented in bold characters can be adjusted by the user if necessary. The other part of the input file is written by the interface program to give information back to the user.
The simulation time entry determines at which time the program reads the information concerning the rods results in “deck.out”. The time for the channels results differs usually from this one and the program selects the nearer results time. This second time is edited in the summary of the execution of kcttni.

- “struc.dat”:

  Contrary to what was explained for kcntti, this file is only used by kcntti to configure itself, particularly to create dynamically the size of the matrixes.

- “deck.out”:

  Output file from the processing of COBRA-TF. It contains sorted by simulation time step, the results for fuel rods and channels (The results printed in this file are flexible and can be chosen by rightly filling the group 14 of the input deck. For our investigations we need results for all channels and fuel rods).

Output:

- “cobra_ks.dat”:

Actually written by kcttni to give a new starting point to KARBUS.

4.4.3 Program description

kcttni consists of 12 modules and a main program. As well as kcntti, kcttni was written in Fortran 90 at first under the Windows OS with the Compaq FORTRAN compiler. The transfer under Linux was realised with the Lahey compiler “lf95”. Similarities exist between the two programs. The realization of the geometry correspondence between COBRA and KARBUS appears in the same manner in both interface codes.

4.4.3.1 Architecture layout

The components of kcttni are described in this section.

- kcttni.f90

Main program: it manages the call order of modules and subroutines.

- module “endoffile” :

This module is called at the beginning of the main program to set the value of an error parameter (IOSTAT) which is used by the other parts of the program in the “read” instructions. It permits to recognise the end of the file during the read without prompting an error message and breaking the program. This module is also used in kcntti.

- module “definition”:

This module centralises the declaration and initialisation of the main relevant variables. First, it reads the input file and the “struc.dat” to affect values to its functioning parameter, notably the matrix whose definition must be defined at each run. Actually the size of the matrix depends of the size of the modelled assembly and of the number of axial nodes. This module and the corresponding subroutines are called at the beginning of the main program. The subroutine set_connex build the geometry correspondence matrix.

- module “conversion”:

It contains the necessary conversion functions from British system units to the international system units.

- module “read_store”:

A find-function is implemented in the main program. As soon as a particular string is found in the file according to the chosen simulation time, the module with its subroutines are called. The aim is to read the results concerning the fuel rods from the deck.out file and to store the data in matrixes, while processing before the unit conversion.

- module “readstore_dens”:

Identically, this module reads and stores the values for the water density. This data are not at the same place as the others for fuel pins in the source file. Because of this a module was dedicated.
A find-function was also implemented to find out the position of results, which depends strongly of the output options set for COBRA-TF and of the axial nodalization of the model.

- module “average”:

It manages the calculation of average values required by KARBUS. The results are stored in other new matrixes. The computation tasks handled in this module are described in the next part.

- module “Edit_results”:

The resulting data sheet “cobra_ks.dat” is edited by this module and most precisely his subroutine to feed KARBUS with updated starting values for water, fuel and clad properties.

4.4.3.2 Calculation tasks survey

Calculations made by kcttni are presented here. Actually the results given by the processing of COBRA-TF in its output deck are not directly exploitable with KARBUS. The following section explains the transition to required data.

See Annex F and Annex G for more details.

- H20 average temperature:

The temperatures of water are read for every rod (variable rd_nb) at each rod surface (variable rd_surf) and every axial node (variable no_nb) for the mentioned simulation time. These values are stored in the 3 dimensional matrix:

H20_tp (rd_nb, rd_surf, no_nb+2).

On the other side KARBUS requires only a mean value for each rod at every axial level. Averaged values on the different wetted surfaces are calculated. The 2 dimensional matrix *H20_tpres (rd_nb, no_nb+2)* is filled with the results.

- Clad average temperature:

The inside and outside clad temperatures are read for every rod at each rod surface and every axial node for the mentioned simulation time. These values are stored in the 4 dimensional matrix: Cl_tp (rd_nb, rd_surf, 2, no_nb+2).

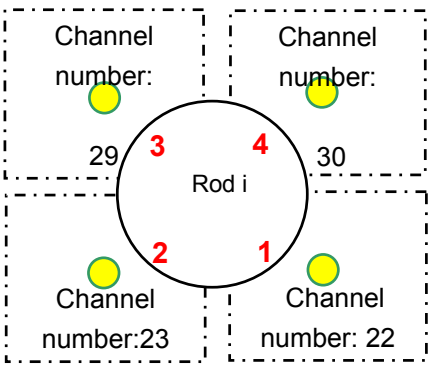
An averaging calculation is performed to fulfil KARBUS input requirements.

Results are stored in the following 3 dimensional matrix (since the parameter of the second dimension is fixed and equal to 3): cl_tpres (rd_nb, 3, no_nb+2).

- H20 average density around a rod:

The density values are not read at the same place as the other values of fuel rods. They are read for every channel at each axial node at the due time. The axial nodalization for the channel results and the rod results differ even if the physical boundaries are the same. To provide the density of water around a rod, one average value must be calculated for every wetted surface of the rod, with the values of density in the channel overlooking this surface.

The dependences between each rod surface and its outfacing channel are set in the matrix *Heat_cond (rd, sf).* Here *Heat_cond (i, 3)* =29 for example (Fig. 4-6).



Rod surfaces: **1,2,3,4**

As read in the deck.out file:
Nuclear fuel Rod no. i

- surface no 1 of 4
  Conducts heat to channels 22
- surface no 2 of 4
  Conducts heat to channels 23
- surface no 3 of 4
  Conducts heat to channels 29
- surface no 4 of 4
  Conducts heat to channels 30

Density at this node j resulting of the average value of the density at the channel node above and under it.

Fig. 4-8 Rod surfaces and outfacing channels

The Fig. 4-9 shows the relevant dependencies between the scalar calculation cells for the rods and the channels (see the User's manual part of [3] for details). They are put together on the same scale for the explanation but are not corresponding in the reality at the same geographical region.

For this example 10 axial levels are represented (NONODE=10). NONODE is defined in the input file of COBRA-TF in the Group 4. DXS represents the vertical node length and is set at the same place. The product of the both values specifies the height of the channel.

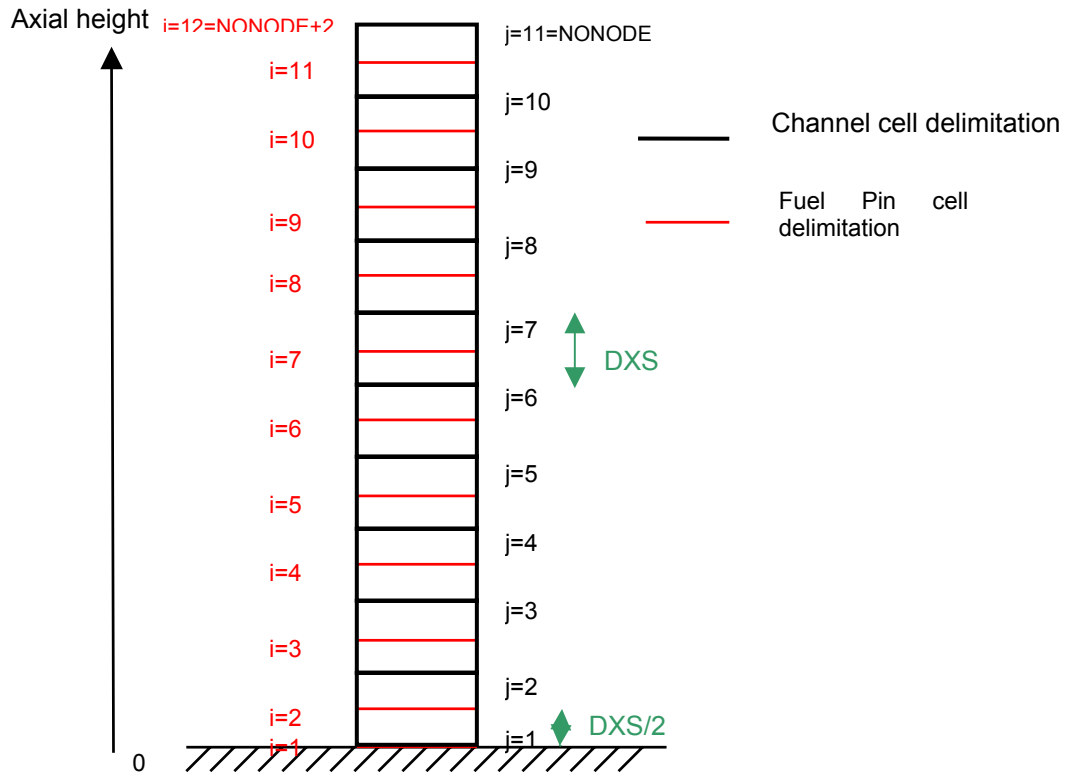


Fig. 4-9 Length discretization in COBRA-TF for channels and rods.

Derived of the COBRA-TF computational cell structure [3], the state variables such as pressure, enthalpy, and density are calculated at the cell centers and assigned to nodes according to the channel specific node numbering. To obtain the density for the i=n fuel node, an average value of the density for the channel nodes j=n and j=n-1 has to be supplementary performed. The Fig. 4-9 presents the different axial nodalization systems for 10 axial levels.

The final results are written in the matrix *H20_densres (rd_nb, no_nb+1)*

- Fuel average temperature:

The temperatures of the fuel are read for every rod at each rod surface and every axial node for the mentioned simulation time. These values are stored in a 4 dimensional matrix:

Fu_tp (rd_nb, rd_surf, 7, no_nb+2).

In the radial direction the fuel temperatures are calculated in several points from the centre to the fuel surface according to the options set in COBRA-TF. This radial discretization is not currently flexible in kcttni at contrary to the axial nodalization. kcttni can only

run with a radial discretization containing 5 nodes within the fuel pin (exception made of the centre and of the surface). In contrast the positions of these nodes are not fixed in the code and this is made during the read phase of the deck.out file (stored in the matrix “rad(:)”). That is to say that any fuel pin could be investigated, if only 5 inner fuel nodes are needed.

At first and as well as for the other case a mean value of the surfaces at each radial node must be calculated. Then KARBUS requires only one value per rod and node. A section equivalent temperature has to be defined as following:

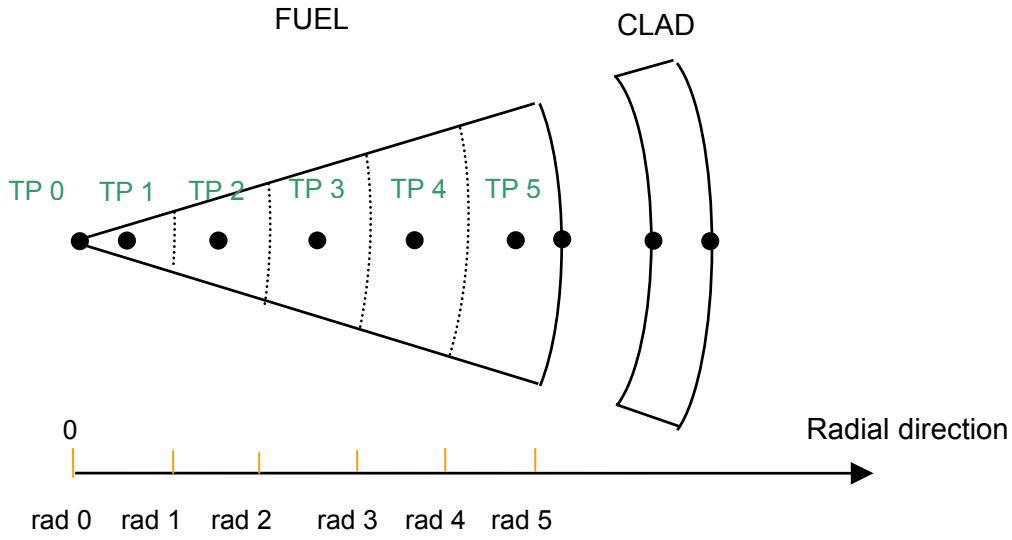


Fig. 4-10 Conduction node positioning for the nuclear fuel rode geometry

Radial conduction nodes are positioned within the conductor (Fig. 4-10). Each material region is divided into sub regions of equal radial thickness. A node for the COBRA-TF conduction model is located at the centre of mass of each sub region (exceptions are made for the inside and outside surface). The fuel centreline temperature is calculated by Hermite interpolation. The fuel average temperature given to KARBUS is calculated with the formula (4.2).

$$\hat{T} = \frac{1}{\pi \times rad5^2} \times \sum_{i=1}^{5} \pi \times TPi \times (rad(i)^2 - rad(i-1)^2) \qquad (4.2)$$

For simplicity, the rad(i) correspond to the radial distance represented in the Fig. 4-10 rad i and not to the ith element of the matrix rad(:) defined in the file kcttni.f90. In fact the indices differ of one numerating unity between both.

The final results are store in the two-dimensional matrix Fu_tpres (rd_nb, 8, no_nb+2) at the fixed value 8 for the second dimension.

4.5 Overall flow chart with interface programs

With the past discussions as background, it is now possible to give a more detailed overview of the coupling structure in Fig. 4-5. The components are represented with their usual names and the dependencies are also mentioned. The procedure COBRAP steers the progression steps and count the iterations. As soon the required iteration number is reached COBRAP break the loop and the program stop. Further information on the procedure COBRAP is reported in Annex I.

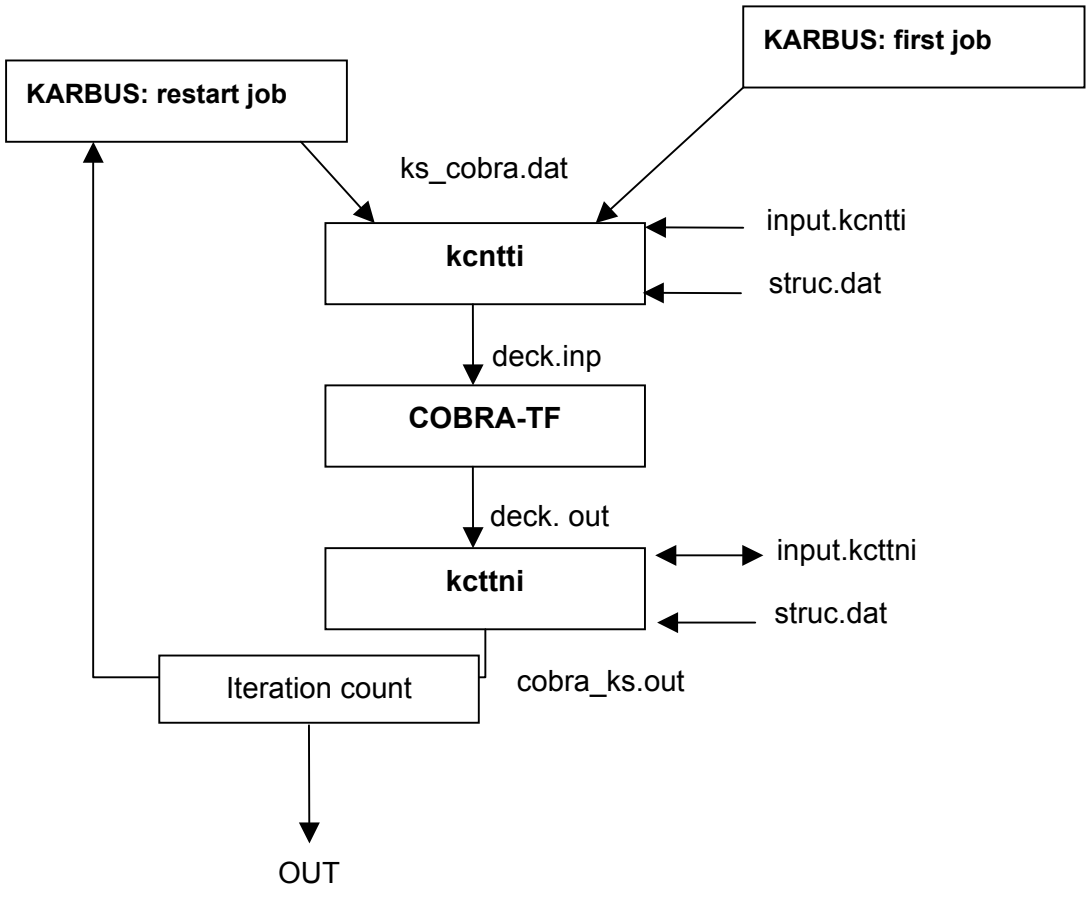


Fig. 4-11 Coupling structure overview in KAPROS-E

4.6 Investigation of the weighting of feedback data

4.6.1 Introduction

A first coupling between KAPROS-E/KARBUS/DANTSYS and COBRA-TF was possible as the achievement of the work presented in reference [18].

The new coupled calculation code KAPROS-E/KARBUS/DANTSYS/COBRA-TF was used to study the evolution behaviour of the linear pin power and other results. The coupled calculations were performed in a loop until a definite iteration count was reached. The first results are presented below in Fig. 4-12 [31].

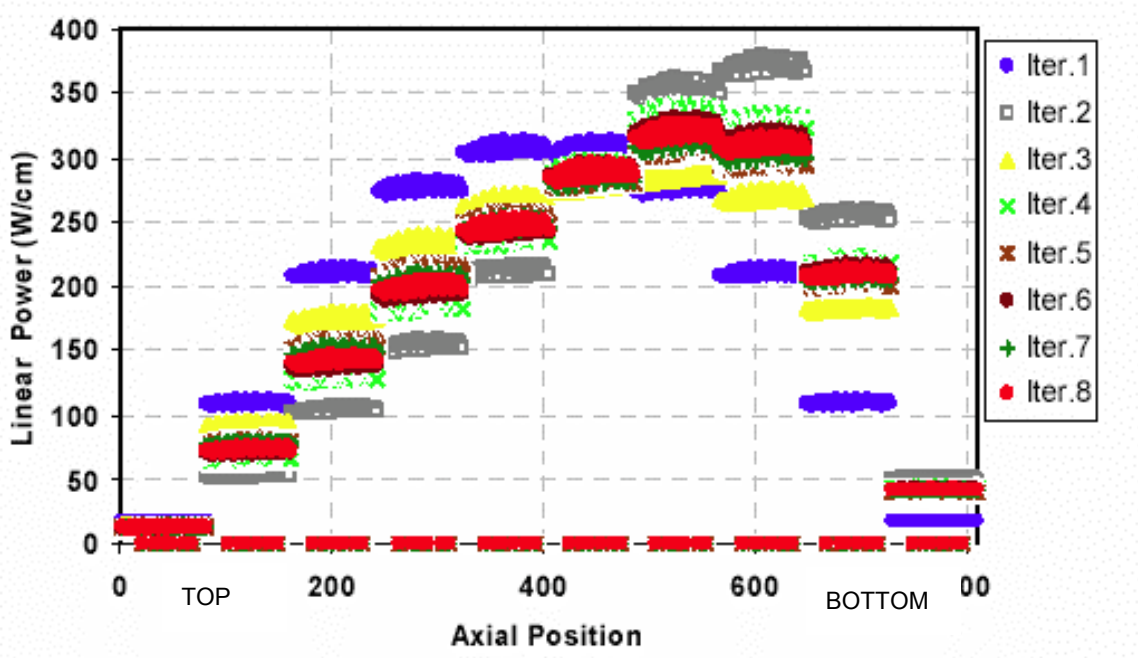


Fig. 4-12 Development of the linear power of fuel rods along the axial nodes [31]

The calculation begins with a first KARBUS job, where coolant properties are set as constant along the axial direction. This results in a cosine shape of the first iteration (blue curve). Then feedback results modify the shape of the linear power distribution for each iteration. As we can see, the last iteration differs strongly from the first one, and the maximum linear power moves towards lower axial levels, e.g. towards the bottom of the assembly.

The figure shows an oscillating behaviour for the linear power and the same effect is confirmed by the behaviour of other investigated distributions (coolant temperature and density, fuel and clad temperature). The oscillations have smaller amplitudes with an increasing number of iteration counts, and convergence is reached after about 8 iterations [31].

These results are satisfying and improve the prediction capabilities of safety parameter at the fuel pin scale.

However the computer time costs for this study are quite important. The calculation required more than 2 days on an IRS Linux-Cluster node to run the coupling program on an 18*18-24 assembly [6] for 10 axial layers and 28 energy groups during 8 iterations. If possible, this computation time should be reduced without changing the calculation parameters as the number of axial levels or the number of energy groups.

One consideration way to save computing time is to study the convergence behaviour and to try to speed it up. Among other possibilities, the relaxation method was already used at the FZK for another problem concerning the coupling of KAPROS and RELAP5 [32] and had shown good capabilities to obtain faster results under several conditions.
At each iteration step, new distributions of coolant temperatures, densities etc. are calculated and the concept consists in mixing the new calculated distributions with the precedents from the past iteration, and to give it then again to the next program.

4.6.2 Weighting function

A similar method has been implemented for the coupling KAPROS-E/KARBUS/DANTSYS with COBRA-TF. It is aiming at a faster convergence, but keeping in mind that the convergence must not be disturbed. Obviously the influence of the weighting parameter is relevant for such methods, and a compromise has to be found between acceleration and convergence conservation or optimization. This dilemma must be kept in mind.

A simple linear law for the weighting was chosen and is schematically expressed in the formula (4.3). It is based on the mix of two distributions with an adjustable but constant weighting parameter.

$$\text{New distribution}_{\text{relaxation}} = \frac{(100-W)*(\text{Old distribution}) + W*(\text{New calculated distribution})}{100} \quad (4.3)$$

W is the weighting parameter satisfying $0 \le W \le 100$. This parameter can be set in the input file of kcttni.

For W=0 the convergence behaviour is annihilated and we will have no change from the second iteration since the same data will be always given to KARBUS at each run.
For W=100 the mix with the previous distribution is disabled and the results are those obtained without relaxation.

4.6.3 Weighting function implementation

The weighting function affects the results calculated by COBRA-TF: coolant temperature and density as well as fuel and cladding temperatures. These results are calculated for all iterations by COBRA-TF thanks data from KARBUS and are written normally in deck.out. The weighting is then achieved externally in the interface program kcttni, which however already manipulates the COBRA data for other purposes.

In section 4.3 and 4.4 the two interface programs are described. For kcttni, in 4.4, not all features were discussed because they concern the implementation of the relaxation function just mentioned.

The following section concerns the implementation of a weighting function to steer the convergence behaviour, and the addition of a convergence test to stop the program when a chosen convergence criteria is fulfilled.

This necessitates the accomplishment of the following jobs:

- Set the name of the file where the previous COBRA-TF results for mixing with the new ones were stored. This is the aim of the module “setlastcobra”. When no previous results are available, typically for the first iteration, the value 1 is given to the integer parameter “jump_modlastcobra” and instructions to weight the distributions are ignored.

- Read and store into matrix the data from the previous “cobra_ks.dat file” for future manipulations with the weighting function. The module “lastcobra” achieves this work.

- Define the weighting function. This definition is made in the module “weighting” with the creation of the function “mix (new, old)”. This allows the future change or improvement of the weighting function, without having to make changes everywhere.
  Obviously, for advanced functions, changes should be made since our present function requires only a parameter.

- Calculation of the variation rate between the previous distribution and the new one for comparison with a convergence criterion. This is done in the module “testconvergence”.
  The convergence criterion “Conv” is defined here directly in the source code since a reasonable value has not to be changed at each job. The subroutine “test_conv” works out for each category of results the maximum variation rate between the new and the previous distribution. These maximum variation rates are written back in the input file of kcttni to inform the user during the calculation on the convergence approach (see the input file presentation in section 4.4.2). That offers the possibility to abort the calculation if the precision seems to be sufficient, e.g. the variation rates between the present and the previous iteration are small enough.

  The order of the written information in the input file is: H_20 temperature maximum variation rate; clad temperature maximum variation rate; fuel temperature maximum variation rate and finally the maximum variation rate of water densities.

  The module compares the results obtained with the convergence criteria at each run. If they satisfy the criteria, it writes “1” instead of “0” in the input file of kcttni in the convergence rubric. That means that the run of the coupled program can stop at the end of the iteration

4.7 Final coupling flow chart

The Fig. 4-13 shows a final overview of the coupling structure. The interface program kcttni which handles important tasks is shown in more details. This architecture requires an uplevelled program to steer the processing development. This procedure COBRAP exists al-

ready as module of KAPROS-E and was adapted to the new interface programs and to the convergence survey. KCNTTI and KCTTNI are standard KAPROS-E module skeletons, calling the new developed FORTRAN-90 packages kcntti and kcttni respectively (executable programs). The blue lines represent the interventions of COBRAP to manage the global sequence.

It is not possible to interrupt the execution of the procedure COBRAP directly in kcttni. That is why an information transfer is operated between kcttni and the module COBRAP via the input file. The procedure was adapted to read at every step the input.kcttni file to determine if a new iteration must be performed or not.

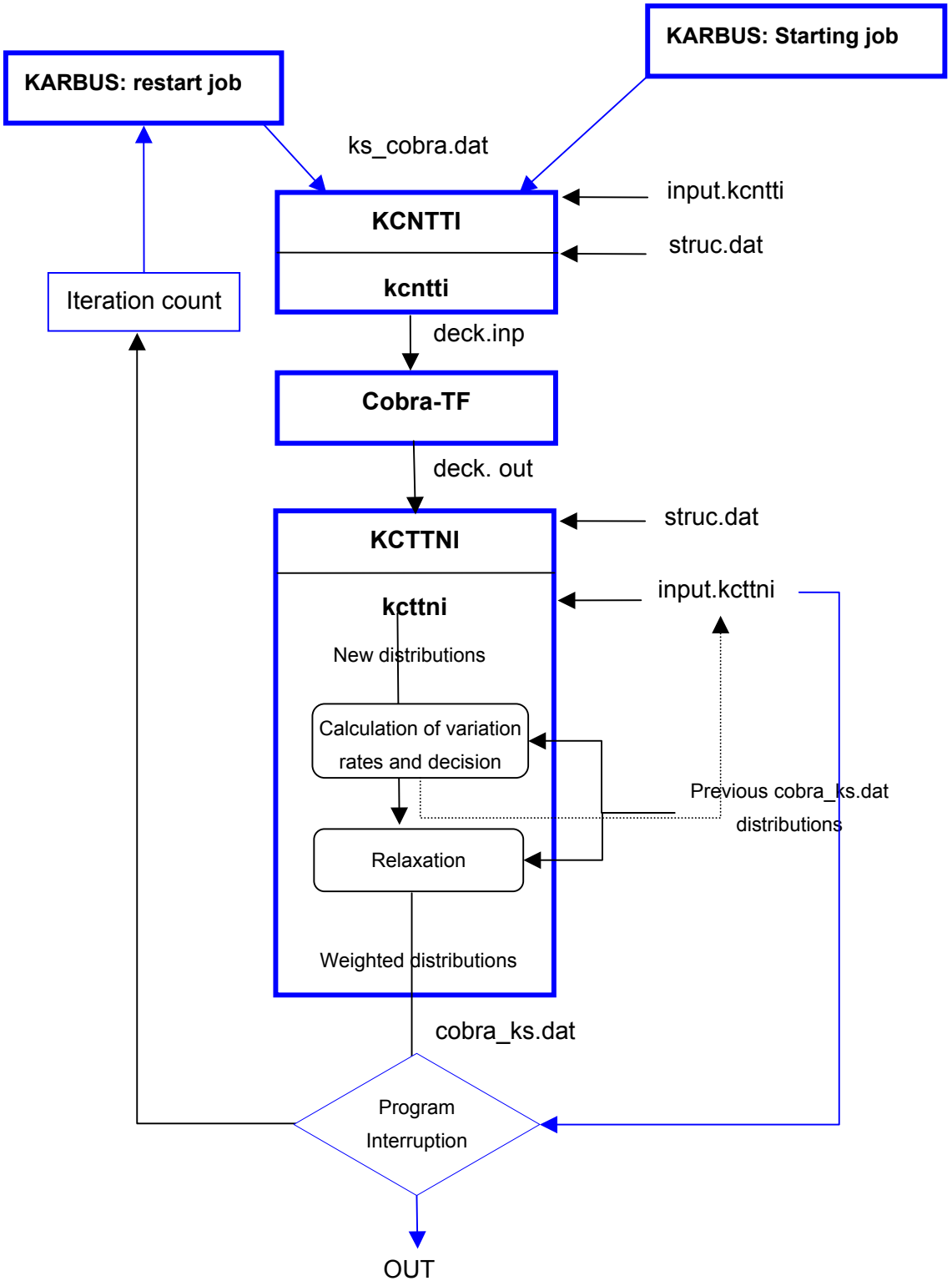


Fig. 4-13 Final coupling flow chart in KAPROS-E

5 Application of the new code system to a PWR subassembly benchmark model

The past part provides requirements and explanations on the coupling problem of KARBUS/DANTSYS and COBRA-TF. The two interface codes and their capabilities were presented as well as elements of their realisation.

The coupled program must now be tested and validated. For this aim, a well defined problem is chosen as investigation basis, and this PWR benchmark plant is presented in the following section. The first investigations on this problem are then reported and discussed. An important part of this section examines the convergence behaviour of the coupled distributions and demonstrates the capabilities of the new coupling.

5.1 Description of the benchmark model

A GRS Benchmark proposal was chosen. It was already the investigation basis of the work in reference [18] and is well known at the institute. These reasons make easier the verifications and the results analysis.

5.1.1 Assembly description

Assembly specifications from FRAMATOME-ANP and GRS are used for a fuel assembly of type FA 18*18-24 [6]. This notation signifies that the assembly contains 18*18=324 elements, divided into 300 fuel pins and 24 control rods.

The Fig. 5-1 shows a schematic section of the assembly.

The constitutive symmetries are revealed by the lines drawn on the figure and satisfy the conditions already mentioned in 4.3.3.2 for the restriction of the domain of study. The investigations on the problem could be reduced to the consideration of only one eighth of the whole assembly, since all the other parts can be deducted from it. However KARBUS and DANTSYS model one quarter of the core. The simulation domains for COBRA-TF and for KARBUS/DANTSYS are mentioned in the figure.

Practically, the cut of the assembly in cells (fuel cell and control rod cell) is allowed because the constitutive materials within an assembly are laid out in a relatively regular way (exception made of control rods or guide tubes) and this repetition of the same material pattern constitutes a grid as presented in Fig. 5-1. The consideration of the composition of each cell leads to the description of another sub scale of modelization of the problem.

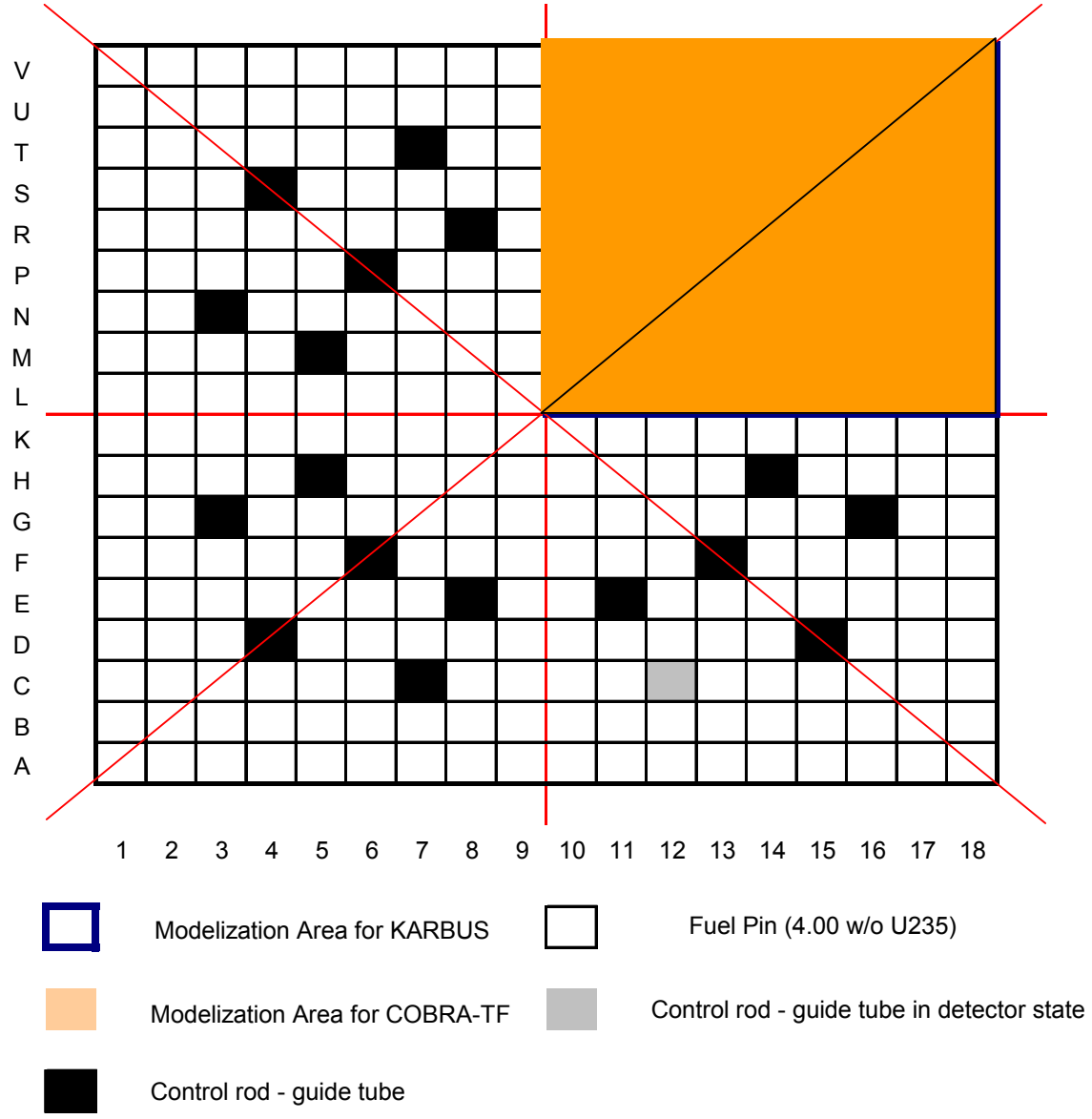


Fig. 5-1 18x18-24 Fuel assembly

5.1.2 Cell considerations

In the case of a PWR, each fuel cell in the assembly consists of a cylindrical fuel pin part surrounded of water, playing at the same time the part of moderator and coolant. Such a cell is presented in Fig. 5-2 where the architecture of the fuel pin is shown in details.

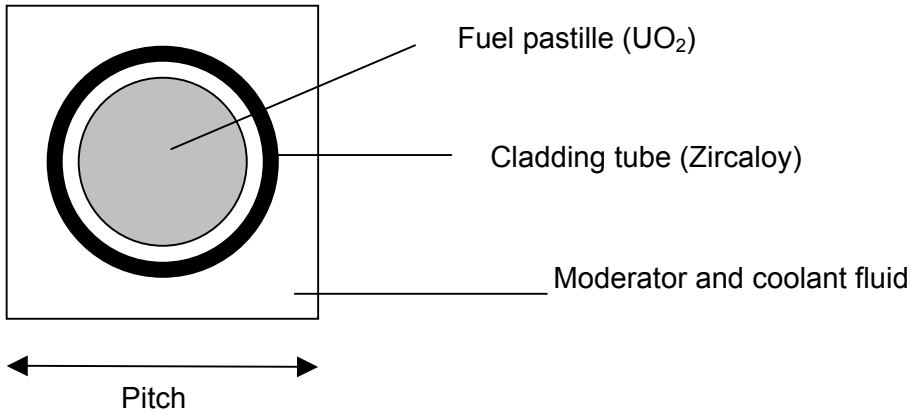


Fig. 5-2 Unit cell

A square water zone is associated at each pin, with an edge equal to the pitch (space between fuel pins).

The benchmark specifications are the following:

Assembly identification:	18x18-24
Pin number pro assembly	300
Pitch (cm)	1.270

Active fuel rod length (cm)	390.0
Fuel (UO_2 235) (wt%)	4
Fuel pastille diameter (cm)	0.805

Clad inner diameter (cm)	0.822
Clad external diameter (cm)	0.950
Clad thickness (cm)	0.064
Clad Material	ZRY-4

Table 5-1 Benchmark specifications

Further specifications of the power plant can be found in Annex H.

5.1.3 Neutron physics and thermal-hydraulic models

A quarter of the assembly is modelized in KARBUS and also in DANTSYS, e.g. 81 cells as shown in Fig 5-2, where the grid represents already the cell delimitations of KARBUS for a section. As already said the intention is to perform a 3 dimensional study. The assembly length must be adequately investigated to take into account the axial variation of coolant or material properties. The active fuel rod amounts 390 cm and a discretization into 10 levels of 39 cm seems to be reasonable and sufficient. KARBUS and DANTSYS work finally with 810 computation cells.
The unit cell calculation with the module WEKCPM uses a concentric discretization in 23 material zones: 16 for the fuel and the gap, 3 for the cladding and 4 for the moderator. This discretization does not concern directly the unit cells as they are presented in fig 5-3 but is applied to equivalent Wigner-Seitz computation cells. Regarding the dimension of the pitch these discretizations attempt to provide accurate results. The control guide cells are represented empty for this problem and filled with water. These choices are defined in the input for KARBUS in Annex C, and also in the DANTSYS input in Annex D.

To allow a direct communication between the neutronic and the thermal hydraulic codes, the model simulated by COBRA-TF must be equivalent. A eighth of assembly contains 45 rods and 55 outfacing channels. The controls rods are modelized in COBRA with rods without power. This is naturally not correct but simplifies the problem since the control rod influences are not the topic. All the assembly is modelized with only one section and nodalized in ten axial levels.
The coolant flows in the assembly from the bottom towards the top through the modelized subchannels. Each sub-channel is characterized according to its properties since all channels do not play the same roles. They are classified in seven categories which are illustrated in the Fig. 5-3 and the corresponding properties specified to COBRA-TF are given in Table 5-2 [18].

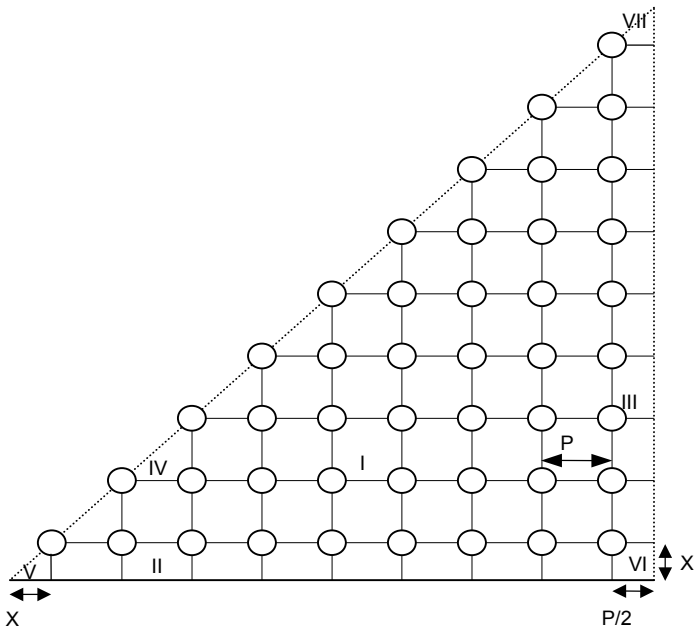


Fig. 5-3 Subchannel differentiation

Channel type	flow area [m²]	wetted perimeter [m]	flow [kg/s]
I	9,041E-5	2,985E-2	2,924E-01
II	5,409E-5	1,492E-2	1,749E-01
III	4,520E-5	1,492E-2	1,462E-01
IV	4,520E-5	1,492E-2	1,462E-01
V	1,599E-5	0,373E-2	5,171E-02
VI	2,705E-5	0,746E-2	8,746E-02
VII	1,130E-5	0,373E-2	3,654E-02

Table 5-2 Channel properties

The domains of KARBUS and COBRA-TF are numbered with their own systems (see Fig. 4-3 and Fig. 4-4)

5.2 First results and comparison

This part gives a survey of the first results obtained. The relaxation functions were not implemented yet, and results can be directly compared with the previous coupling in reference [18].

5.2.1 Model investigations without relaxation: first results

The first calculation was made with 4 iterations. The results are presented along the following figures.

5.2.1.1 Linear power evolution

The linear power distributions worked out by KARBUS in “ks_cobra.dat” are presented in the Fig. 5-4.
The 810 computation cells are developed in abscises and the 10 axial levels can be easily identified. Within every level, the linear power is drawn for the 81 rods. The evolution within an axial level is disadvantageous visible in this form of representation but will be later commented.
The coupling begins with a first job of KARBUS with assumed and constant properties for the coolant temperature and density. A cosine shape is logically observed for the first iteration as it was the case in section 4.6.1.
As already mentioned the first 81 cells are corresponding to the top of the core model (first section) and each group of 81 cells is a section lower towards the bottom of the core, where the coolant is introduced. The middle of the core is placed between the levels 5 and 6 where the maximum of the first cosine shape should be observed. The points appearing on the abscise axe represent the water rods contained in a quarter of the core. At each level we have obviously 6 of these points for the 6 water rods (Fig. 4-2).

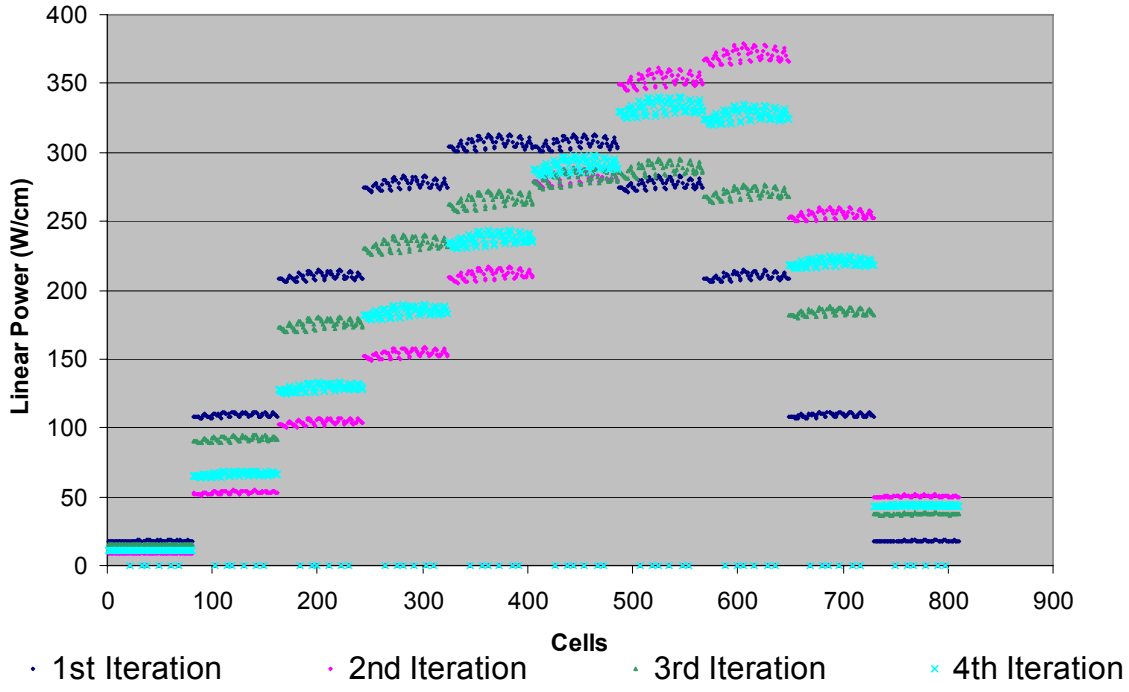


Fig. 5-4 Development of the linear power of rods along the axial levels

An oscillating behaviour is discernable, i.e. the first and the second iteration are enclosing the next two. It is maybe early to speak of convergence, since only four iterations were calculated but the results are encouraging. This point will be closer discussed in a dedicated section.

The shape of the power distribution moves in a general way towards the bottom of the core. Firstly, the maximum axial linear power is at the middle of the assembly height. The second

iteration shows a strong correction and assumes a maximum at the level 8. That is corrected one more time at the iteration 3, where the maximum linear power average is reached at the level 7.

5.2.1.2 Coolant temperature and density distributions

The curves presented in this part and those of the next section are drawn from data computed by COBRA-TF for all iterations.

The both figures have to be considered in the same time since the temperature of water and its density are physically strongly correlated.
The coolant penetrates the core (e.g. the assembly) from the bottom and goes out at the top. The temperature curve has the typical shape of the evolution of coolant temperature along the axial direction in a PWR. The water stores heat from the burn up of the fissile material within the assembly and arises with a higher temperature. This is correlated with a gradual decrease of the water density during its route towards the exhaust.

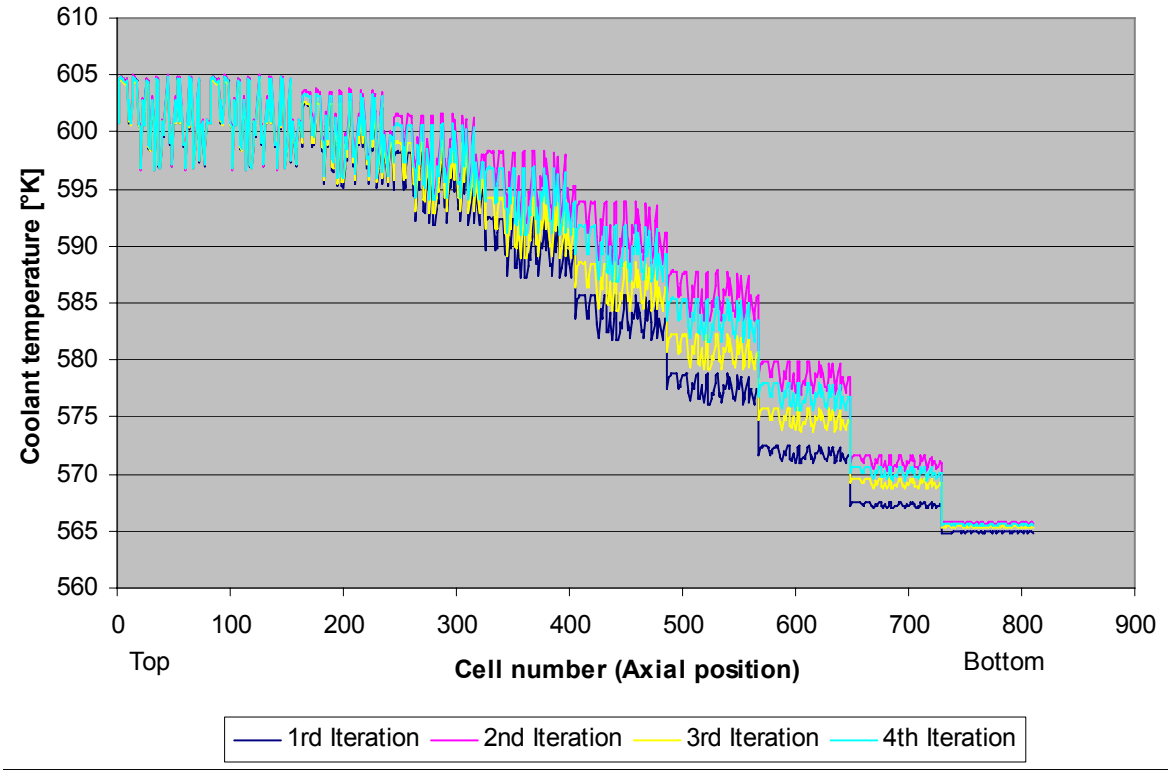


Fig. 5-5 H_20 Temperature evolution

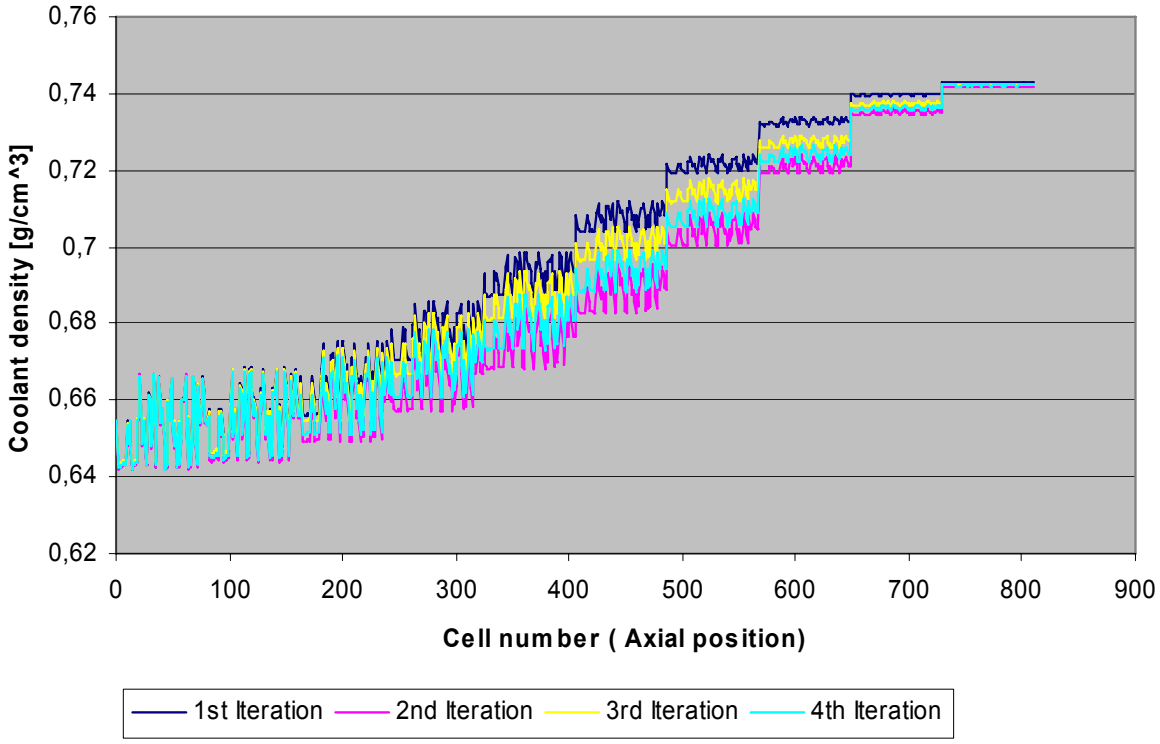


Fig. 5-6 H_20 Density evolution

The water has a double role since it is used at the same time as coolant but also as moderator. The neutron thermalization is then less effective when the density scales down, since neutrons need more time to go trough the resonant zone and reach the thermal zone. This affects the power production and cause the displacement of the curve seen in Fig. 5-4.

The difference of temperatures and densities within a level are quite insignificant at the inlet but are becoming more and more important while going higher in the assembly. Effectively, the water temperatures and densities are quite homogeneous at the inlet section but the different powers transferred to channels produce these disparities which increase with the progression of the coolant.

5.2.1.3 Fuel and Clad temperature

Other relevant values for the coupling are the material temperatures. These have a direct influence on the neutronic part to perform realistic power distribution calculations. The fuel temperature is presented in Fig. 5-7 and Fig. 5-8 gives the clad temperature evolution.

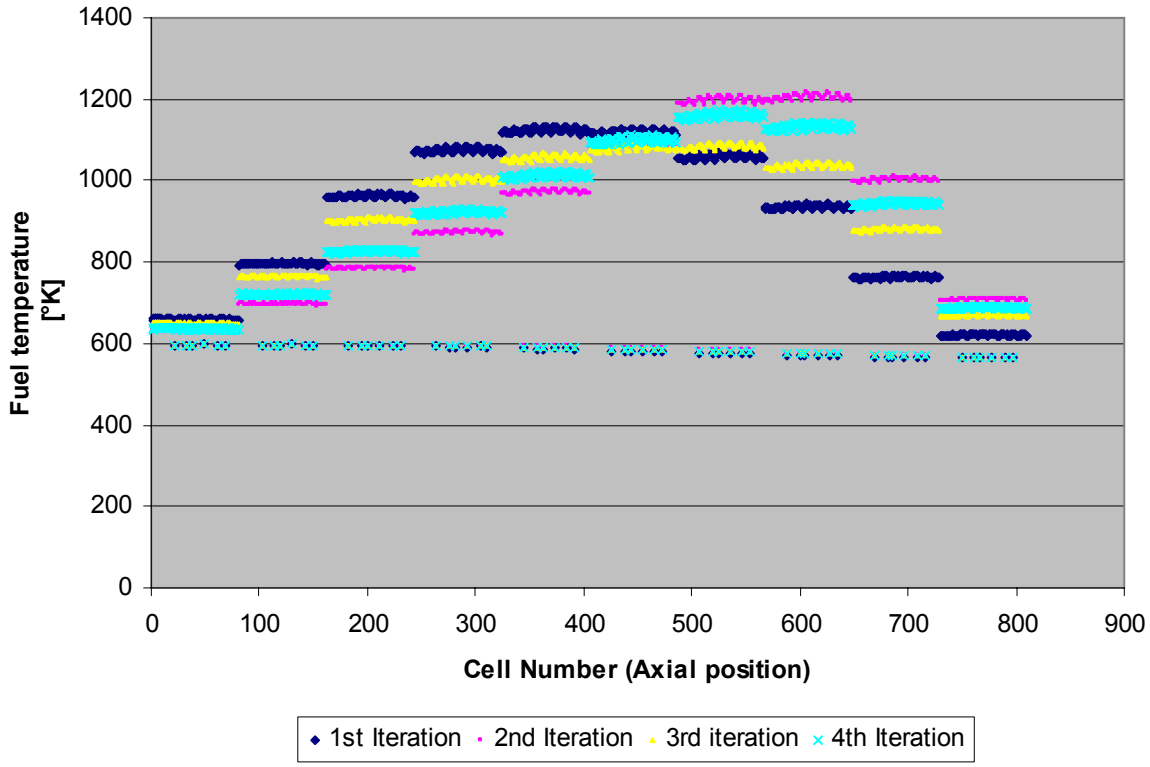


Fig. 5-7 Development of the fuel average temperature along the axial nodes

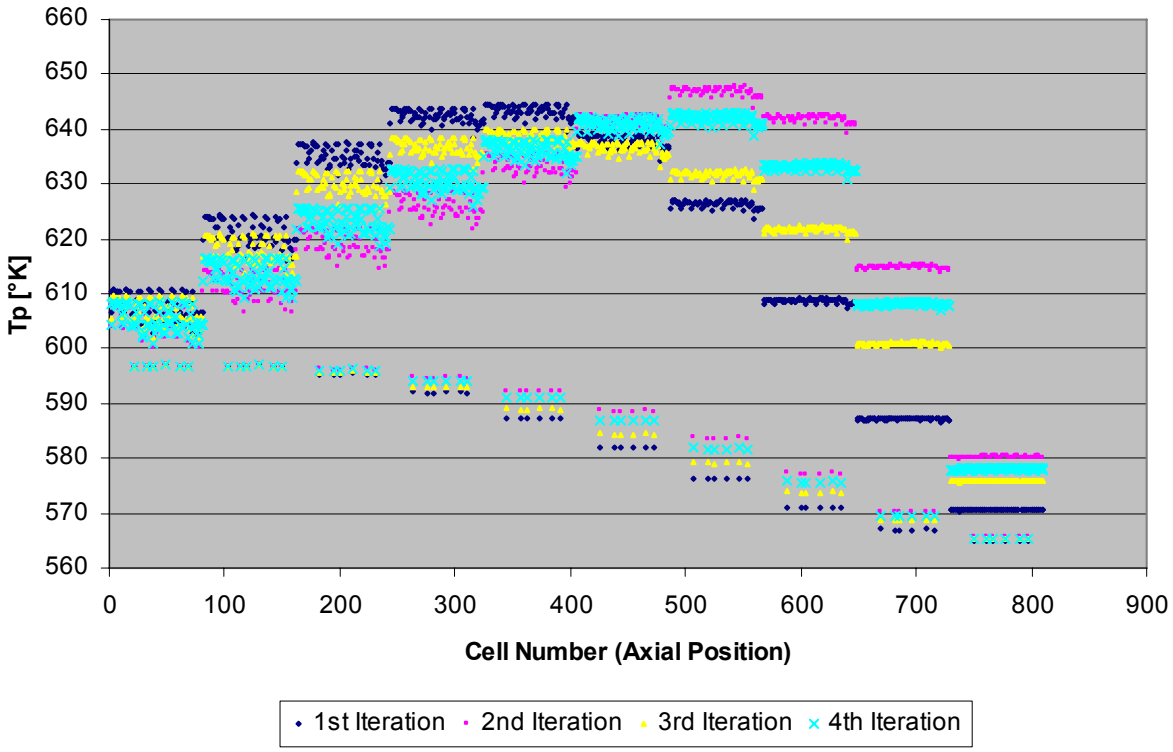


Fig. 5-8 Development of clad temperatures along the axial nodes

The displacement of the curve shapes towards the bottom of the core is still pronounced. The maximum clad temperatures seem to converge to lower values than it was presumed by the first iteration.

The temperature evolution of the clad of the water rods follows the evolution of the water temperature.

5.2.1.4 Fission event distribution within a section

The last representations provide a good overview of results. At contrary, it is difficult to observe easily the disparities of values within an axial section.

The Fig. 5-9 shows the fission repartition within a quarter of section. This repartition is directly proportional to the power production. The grid on the figure does not correspond to the lattice modelized in KARBUS but the geometry can be compared to the Fig. 5-1.

The water rods are dyed in deep blue and are the place of the lower numbers of fission events. At contrary the zones of high fission activity are near the moderator rods, and more particularly between two or three of them. We can also note by the observation of the figure that low fission activities are located at the external periphery and at the centre of the section, i.e. where the moderator influence is less pronounced.

The repartition of fission events shows the good conservation of the diagonal symmetry by the coupling program and the correct implementation of the geometry correspondence between KARBUS and COBRA-TF. The small deviations in symmetry are mainly caused by the interpolation schemes of the plotting software.

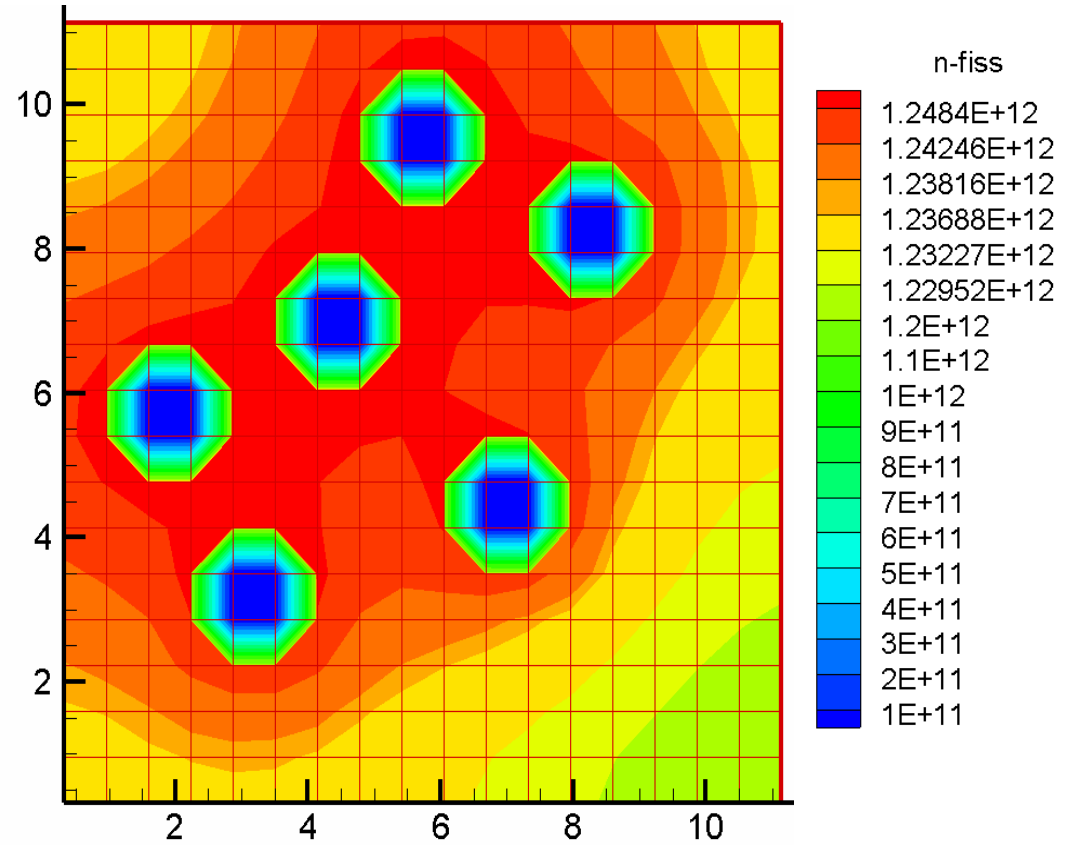


Fig. 5-9 Repartition of fission events within a section quarter

5.2.2 Results validation

All the considerations made during the presentation of the first results are logical and confirm the correct treatment of the problem by the coupling code.

Upon the work referenced in [18], other calculations were performed on the benchmark problem. These calculations were achieved with the two C++ interface programs and without relaxation. Then it is possible to compare the results of both coupling codes.

The Table 5-3 shows the divergences of results between the former program and the new coupling. The maximum variation appears for the comparison of the iteration 3 and is smaller than 1 percent for linear power data.

These differences are however negligible because their orders of magnitude are much lower than reasonable values for the convergence criteria.

Variation Rate	ks_cobra.dat_001	ks_cobra.dat_002	ks_cobra.dat_003	ks_cobra.dat_004
Max variation	0,0001	0,0011	0,0072	0,0028
Mean variation	0,0000	0,0001	0,0020	0,0007

Table 5-3 Deviations between the new FORTRAN-90 version of the coupling and the C++ version

5.3 Simplified calculation model for an efficient testing

An important disadvantage encountered during the development of the coupling and the test phase was the time-consuming run. One calculation with the model specified in 5.1 and 10 iterations required approximately 60 hours on an IRS Cluster node.

A testing model on the same benchmark specifications was created to remediate to this problem, using a simpler problem description.

Only 3 axial levels are simulated with this model to save computing time in the two executions of KAPROS-E and COBRA-TF. Furthermore the neutronic calculations are performed with 4 energy groups instead of the 28 energy groups for the normal model presented in 5.1.3.
The neutronic calculations need nearly 80 percent of the total computing time. These modifications permit to perform a calculation with 10 iterations approximately in 15 hours on the same computer. On the other part the results do not have to be considered in details because of the rough description of the problem. The aim was only to have a faster test platform.

It might be of some importance for further developments and some explanations, as well as commentaries on results, will be discussed in this section.

5.3.1 Implementation of model modifications

The starting points of the modifications are the configurations files used for the benchmark model. However some modifications concerning the treatment of the new study specifications are required in the input files for KAPROS-E, DANTSYS and COBRA-TF.

5.3.1.1 KARBUS

The COBRAP procedure steers the running of the coupling program, as shown in Fig. 4-13. Its input file “input.cobrap” contains setting information for KARBUS in its first part (the same as presented in Annex C). Then the configuration blocs for KCNTTI, KCTTNI, and COBRAP are following but are not concerned by the modifications.

The modifications related to KARBUS are in the following described:

- Bloc MIXCOP:

  3 levels are modelized. Only 243 calculation cells are needed: ‘MISP’ 243

  3 matrix representations give information on the constitution of the three sections.

- Bloc DXBURN:

28 (energy groups) must be replaced with 4 in the two fields ‘GRC1’ and ‘GRC2’.

- Bloc GRUCAL:

The 4 groups library from Collib must be set: '/opt/KAPROS/data/G04COLLIB’

- Bloc TRANSX:

IGM is set to 04 and NUMMAT is set to 243 as well as the matrix NUMMAT must now contain effectively cells up to 243.

Bloc RTWODF:

‘COLL’ 243 and the matrix must also contain 243 cells.

5.3.1.2 DANTSYS

DANTSYS, as stand alone code, possesses its own input file which does not appear in the input for COBRAP. Some changes must be implemented since the calculation of flux on assembly level is also made with 4 energy groups, and on three axial levels as well as for cell calculation in KARBUS.

Below is an extract of the modified DANTSYS input file.

```
   2     0     0
 dantsys input for GRS DWR benchmark
 28    energy groups,   1/4 assembly test with 1 xs set
/
/ ** block (i) **
/
 igeom=  14
 ngroup=  4
 isn=  8
 niso=   243
 mt=   243
 nzone=   243
 im=  9
 it= 18
 jm=  9
 jt= 18
 km=   3
 km=  39
……..
CUT
……..

 zmesh= 0. 130. 260. 390.
 zints=  13  13  13

 zones=
   1    2    3    4    5    6    7    8    9;
  10   11   12   13   14   15   16   17   18;
```

```
……
CUT
……
226  227  228  229  230  231  232  233  234;
235  236  237  238  239  240  241  242  243;
```

The modifications are written in bold characters.

5.3.1.3 COBRA-TF

The modifications must be introduced in the skeleton configuration file “struc.dat”. The energy group modifications do not concern the COBRA-TF calculation configuration, which is rather affected by the level number modification. (See the input description in [3])

- GROUP 2 - Channel Description

The INOD values must be set to 1 or 4 depending on the considered section boundary.

- GROUP 4 - Vertical Channel Connection Data

NONODE is set to 3 and DXS must be redefined to 1, 3 (m). Effectively the length of the fuel rods does not change (390 cm) but only three nodes are required.

- GROUP 11 - Axial Power Tables and Forcing Functions

Here below is the commencement of the modified Group 11 for power tables and forcing functions. The changes are in bold characters.

```
*NGRP NAXP   NQ NGPF
  11   1   4   0
*  I NAXN
   1   5
*      Y    AXIAL       Y    AXIAL       Y    AXIAL       Y    AXIAL
     0.0     0.0    0.65    0.79    1.95    1.55    3.25    0.79
     3.9       0
```

5.3.2 Results with the simplified model without relaxation

In this section, as mirror of results presented in the paragraph 5.1.3, the first investigations done with the simplified model are discussed. The weighting function was disabled by setting the weighting parameter to 100.

The Fig. 5-10 gives the linear power evolution of 6 Iterations for the simplified model. The first iteration shows an axially symmetrical distribution where the maximum is the middle of

the assembly as for the detailed model. The evolution shows one more time a displacement of the power shape towards lower axial levels due to a higher moderator density and also efficiency in this region.

The proper convergence behaviour is remarkable and will be discussed later in more detail. After 6 iterations a satisfying convergence solution is not reached and other iterations are necessary. The variation rate between the iteration 5 and 6 amounts still quite 11 percents.

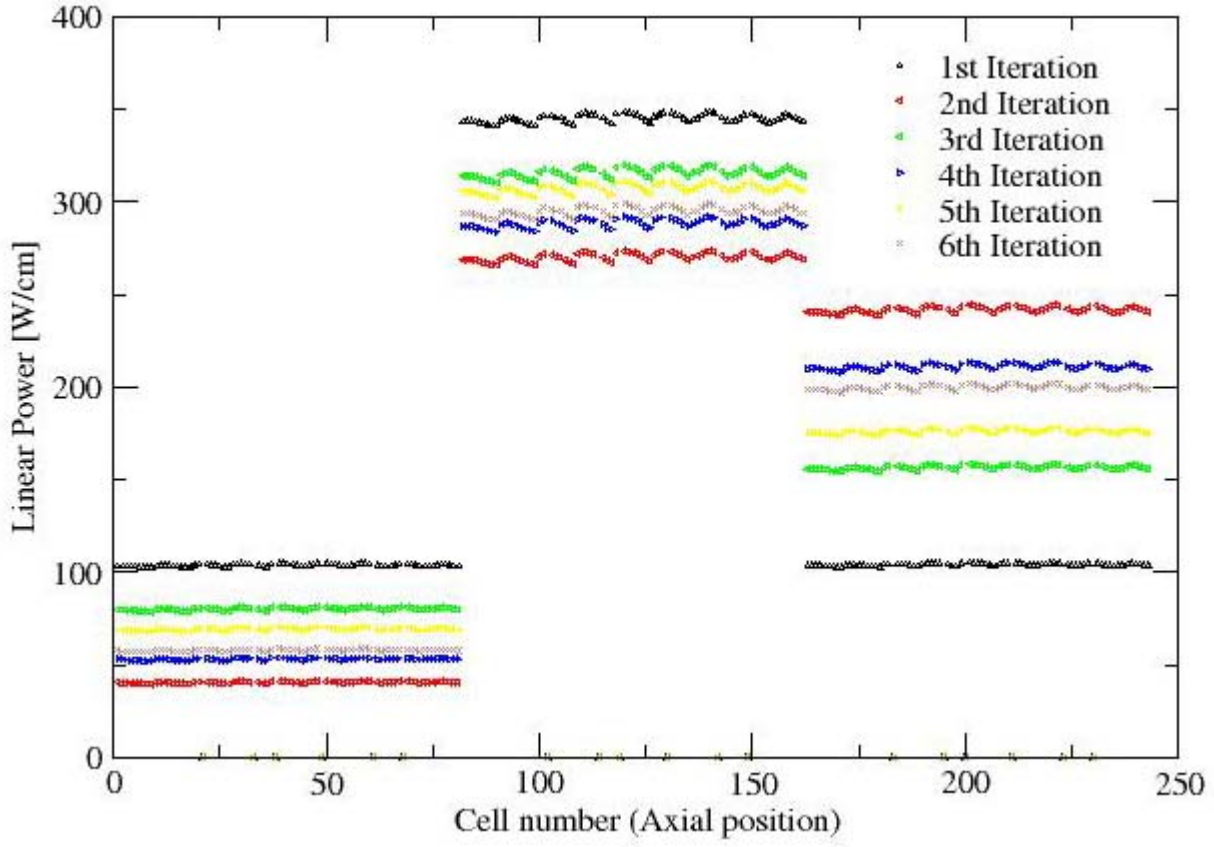


Fig. 5-10 Development of the linear power along the axial positions– Without relaxation

5.4 Analysis of convergence behaviour

The aim is here to study the influence of the weighting parameter on the convergence behaviour and to estimate the gain provided by the relaxation implementation. The study of the convergence behaviour was mainly done on the basis of the simplified model. This was the most convenient way to test several weighting parameter and their influences without spending too much time in calculation. The lessons learned of this study will be later presented for the benchmark model as final achievement for the coupling program capabilities.

5.4.1 Investigations on the simplified model

The investigations treat the linear power rating given by KARBUS. The results obtained without relaxation, and under relaxation with selected weighting parameters (W=50, W=65,

W=75), are compared to a reference solution. This reference solution has been calculated for ten iterations, with relaxation (w=75), and was chosen due to the particularly proper convergence behaviour and only little changes in results since the iteration 5. The variations from the iteration 5 are namely less than 0.5 percent. This reference is used permanently in the following for investigations on the simplified model.

5.4.1.1 Without relaxation

The previous presented curves (Fig. 5-10 for example) give a good overview of the convergence behaviour but a more convenient representation as shown in Fig. 5-11 is chosen for the next developments.

An average linear power value is calculated for every section and iteration. This middle values are presented along the iteration count.

The level 2 and 3 have the same power at the first iteration due to the cosine shape and their axially symmetric position. We can easily see here their different evolutions, and the displacement of power shape towards the bottom of the core.

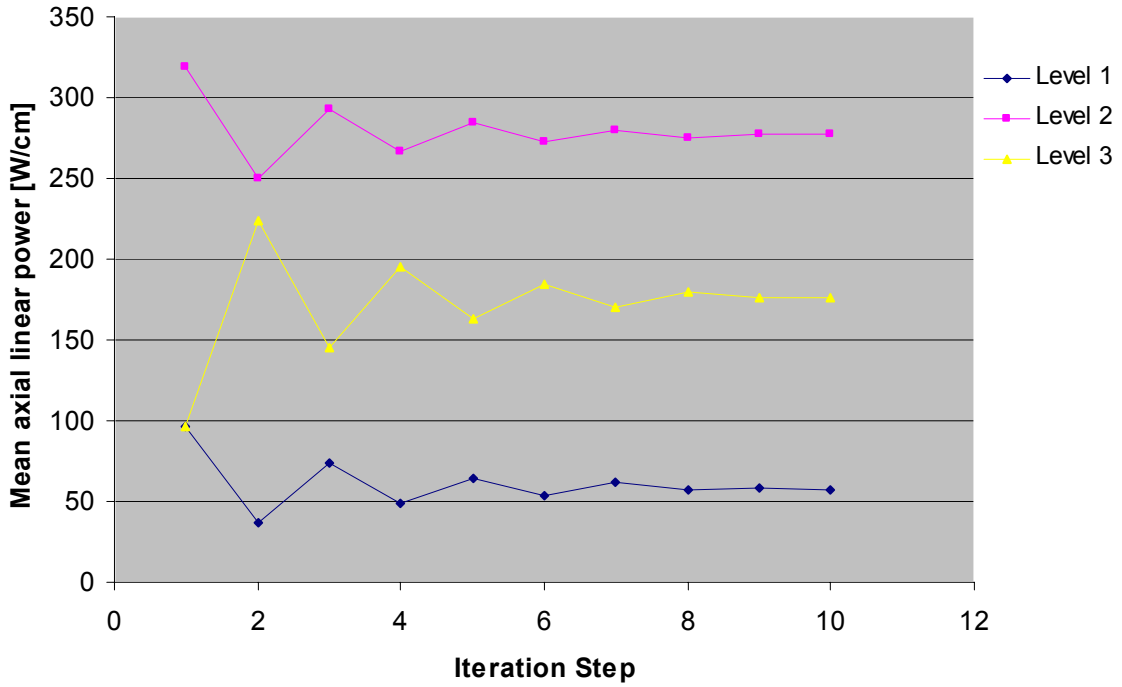


Fig. 5-11 Analysis of the convergence behaviour of the linear power

The Table 5-4 mentions error percentages occurring in comparison to the reference solution. The error percentage scales down with an increasing iteration count. The convergence is reached at iteration 8 for a convergence criterion of 2 percents.

	Iterations									
Level	1	2	3	4	5	6	7	8	9	10
1	0,653	0,354	0,280	0,153	0,105	0,076	0,067	0,019	0,002	0,005
2	0,150	0,099	0,053	0,038	0,024	0,017	0,006	0,009	0,000	0,001
3	0,453	0,273	0,175	0,111	0,073	0,051	0,031	0,020	0,001	0,003

Table 5-4 Error to the reference solution - Without relaxation

These results were obtained with the final COBRAP program by setting the weighting parameter to 100 to disable the relaxation influence.

5.4.1.2 Investigations on the simplified model under relaxation

The last paragraph showed the convergence behaviour without any relaxation method. The method introduces in COBRAP must accelerate but conserve the convergence. The frontier between these two conditions has to be explored by using different weighting parameters to find a compromise.

- **W=50**

  A first trial for the new added relaxation function was made with a weighting parameter of 50. The distributions given to KARBUS are then an average of the old previous distributions and the new coming out of COBRA-TF.

  The results for the linear power distributions are shown in Fig. 5-12. In comparison with the evolution visible without relaxation in Fig. 5-10 we can remark that the oscillating convergence behaviour is lost. This remark is also clarified by Fig. 5-13 and Fig. 5-14.
  The evolution presented for the first section in Fig. 5-14 is the same as for the two other sections. The converging solution goes through a maximum before decreasing and trying to stabilize itself to the reference solution. The errors made in comparison to the reference solution are visible for each section and iteration in
  Table 5-5.

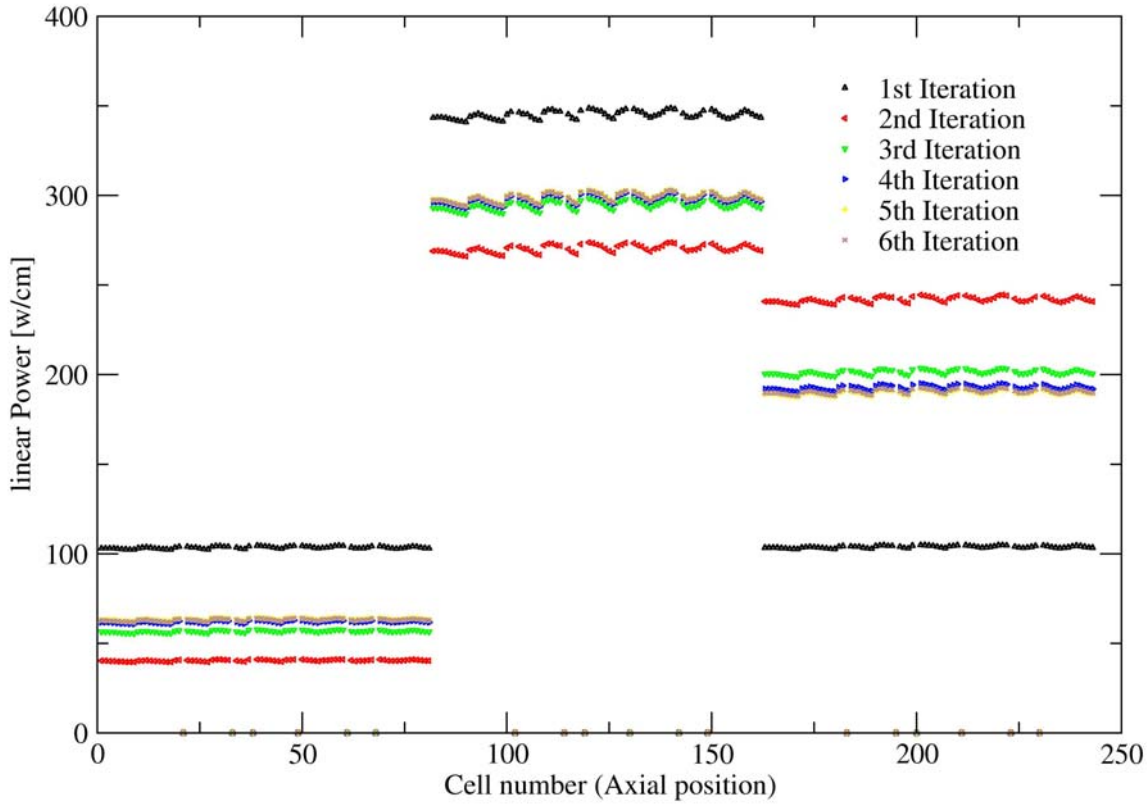


Fig. 5-12 Development of the linear power along the axial positions - W= 50

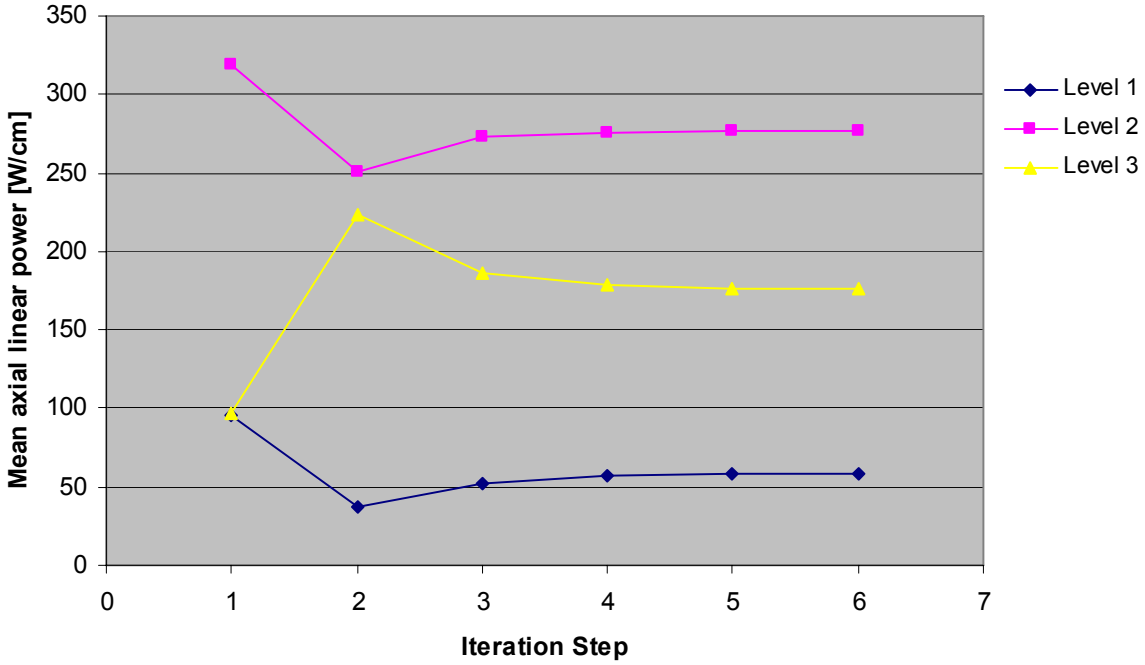


Fig. 5-13 Analysis of the convergence behaviour of the linear Power - W=50

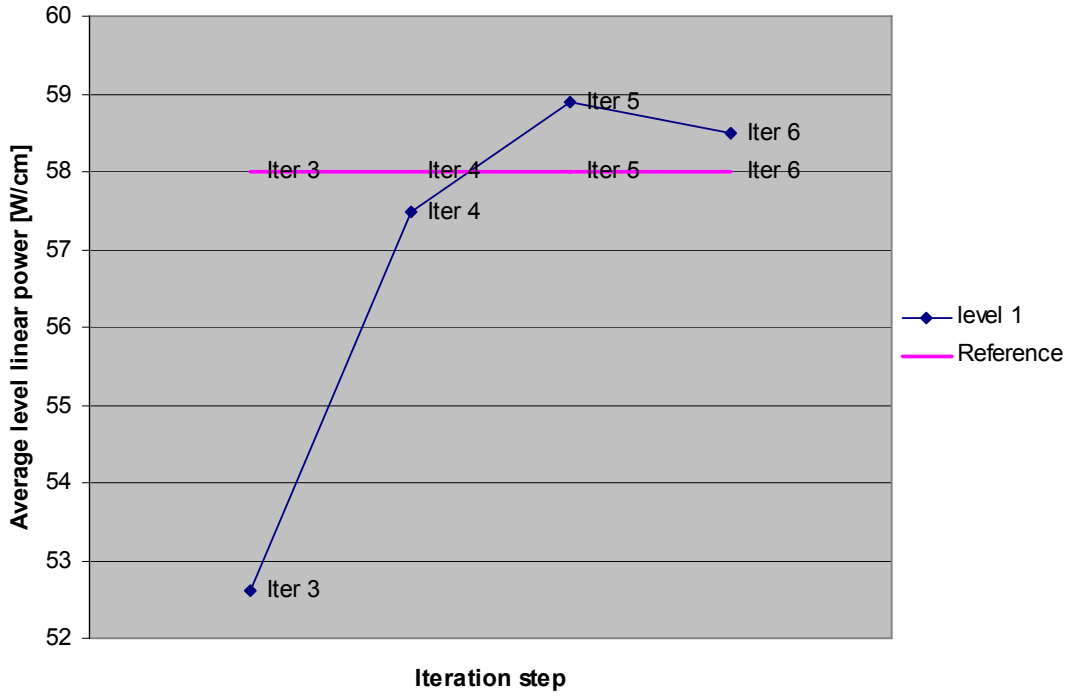


Fig. 5-14 Zoom on the convergence behaviour from the iteration 3 (level 1) - W=50

	Iterations					
Level	1	2	3	4	5	6
1	0,6535	0,3537	0,0927	0,0089	0,0156	0,0086
2	0,1500	0,0991	0,0185	0,0074	0,0028	0,0036
3	0,4526	0,2733	0,0599	0,0146	0,0007	0,0028

Table 5-5 Error to the reference solution - W=50

The error percentages can be compared with those from Table 5-4 for the case without relaxation and show a faster convergence.
The comparison to the reference solution shows convenient results from the 4th iteration differing of less than two percents. Without relaxation, we have to wait the iteration 8 to obtain a similar accuracy.

The approach to a stabilized solution can be however improved as it is shown in the Fig. 5-14 for the top section. This comportment is due to a small weighting parameter.

- **W=65**

According to an assumed final value of the reference linear power distribution, a value of W=65 was chosen. This value was presumed to deliver a fast approach to the “solution”, when the convergence is not disturbed.

The Fig. 5-15 represents the evolution of the linear power given by COBRAP for 6 iterations with the mentioned weighting parameter. The iteration 5 is not drawn for clarity reasons. The 3, 4 and 6 are already superposed and difficult to discern.

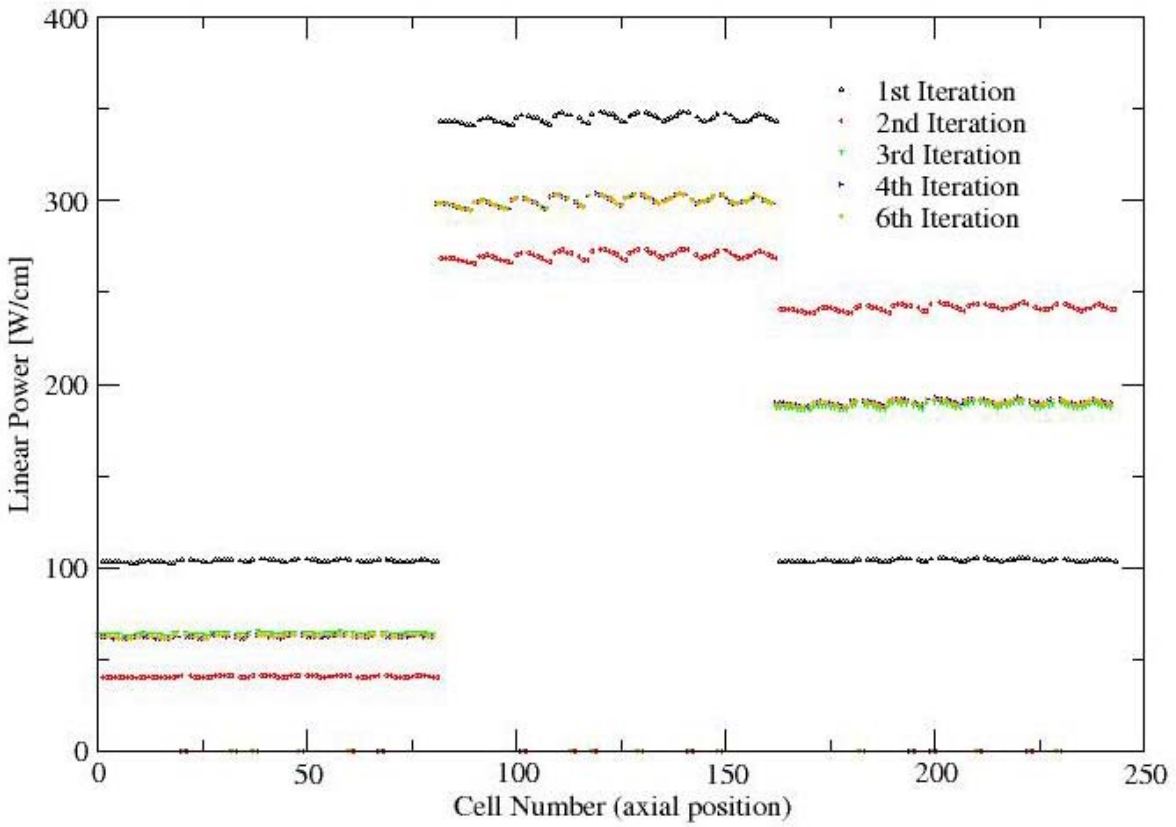


Fig. 5-15 Development of the linear power along the axial positions - W= 65

The following figures present a closer look into the convergence analysis.

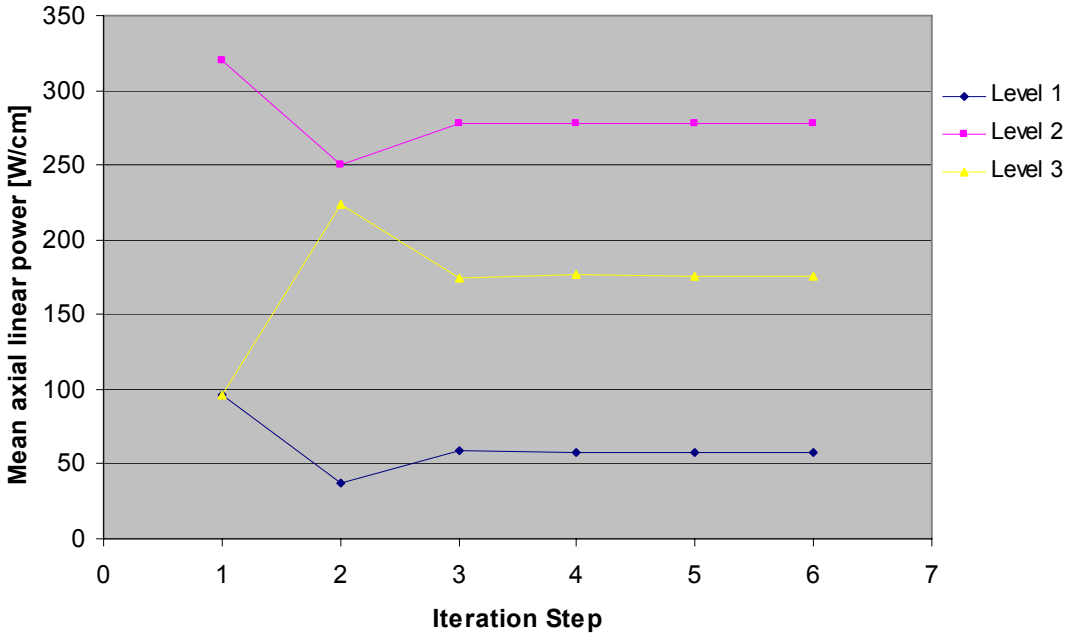


Fig. 5-16 Analysis of the convergence behaviour of the linear Power - W=65

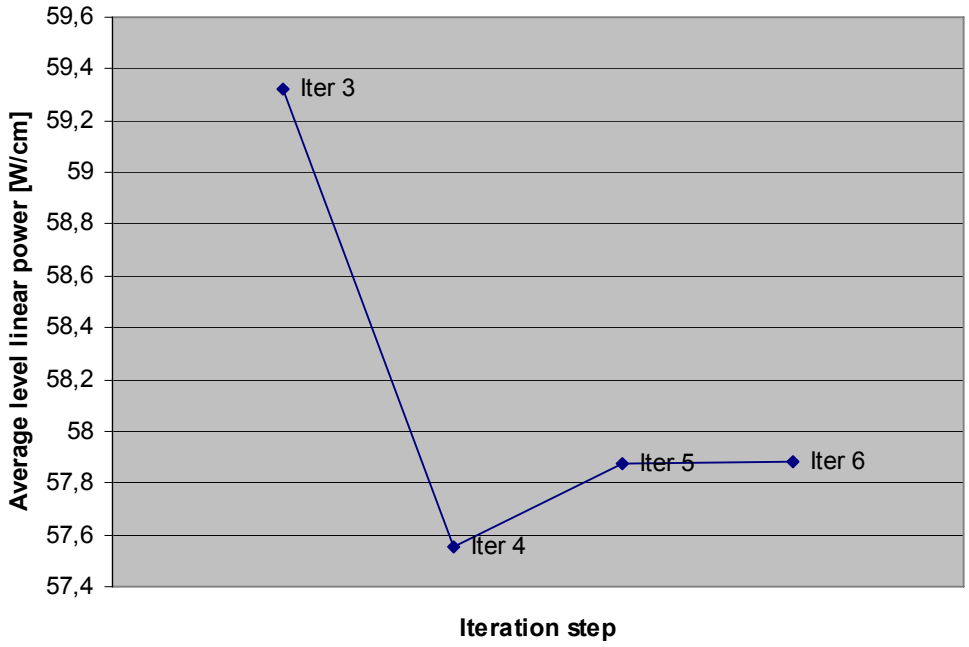


Fig. 5-17 Zoom on the convergence behaviour from the iteration 3 (level 1)- W=65

	Iterations					
Level	1	2	3	4	5	6
1	0,6535	0,3537	0,0229	0,0075	0,0019	0,0018
2	0,1500	0,0991	0,0006	0,0002	0,0002	0,0003
3	0,4526	0,2733	0,0084	0,0027	0,0009	0,0010

Table 5-6 Error to the reference solution - W=65

The Fig. 5-16 proves an accelerated behaviour in comparison with Fig. 5-11. A stabilized distribution is reached from the 4th iteration according to values given in Table 5-6 with an error less than 1 percent.

On the other hand, the Fig. 5-17 zooms on the convergence evolution for the iterations 3 to 6 of the level 1. We can see that the oscillating behaviour is lost from the iteration 4 and becomes asymptotic. This convergence mode is better than that obtained with W=50 since the approach seems to be then pure asymptotic. A similar behaviour is observed for the second level. The results in Table 5-6 confirm this assumption and proved a faster convergence.

A zoom on the behaviour of the level 3 in Fig. 5-18 brings new interesting observations. For this level the oscillating behaviour is present even if it disappears for the two other levels. This remark leads to think that we have approximately reached a transition value for the weighting parameter W.
We can expect that for smaller values of W, the evolution will tend to an asymptotic or assimilate one, as it was the case for W=50. At contrary for higher values the behaviour will tend to be oscillating again. This is also logical because we are approaching the convergence mode without relaxation with high weighting parameters, and the behaviour was an oscillating one.

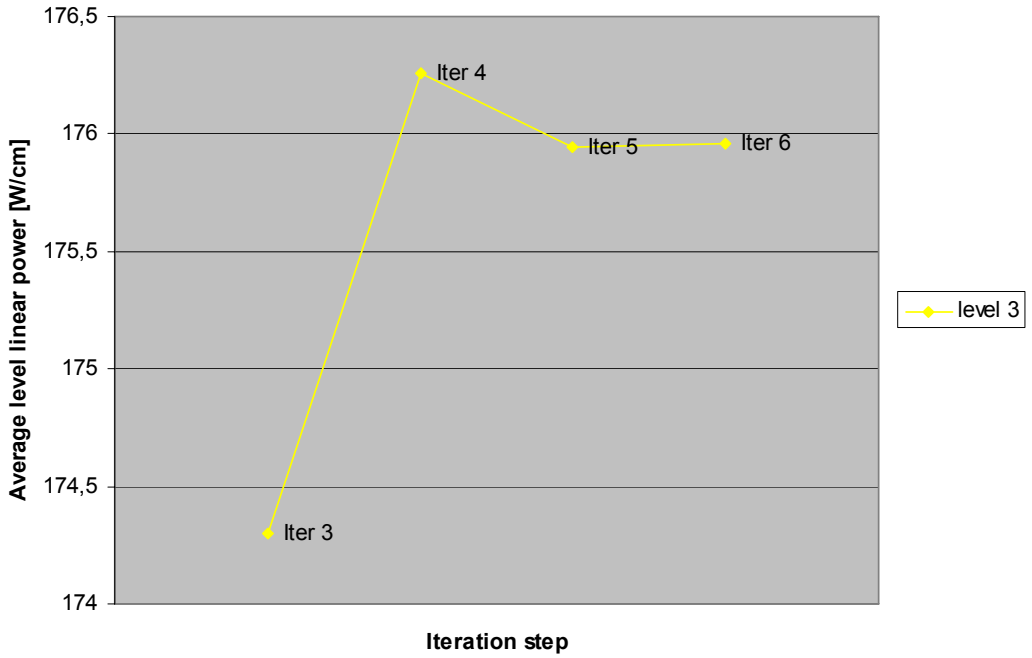


Fig. 5-18 Zoom on the convergence behaviour of level 3 - W=65

- **W=75**

From this statement, a new weighting parameter was chosen to confirm these suppositions. The following figures represent like before the linear power evolution and some details on the convergence behaviour.

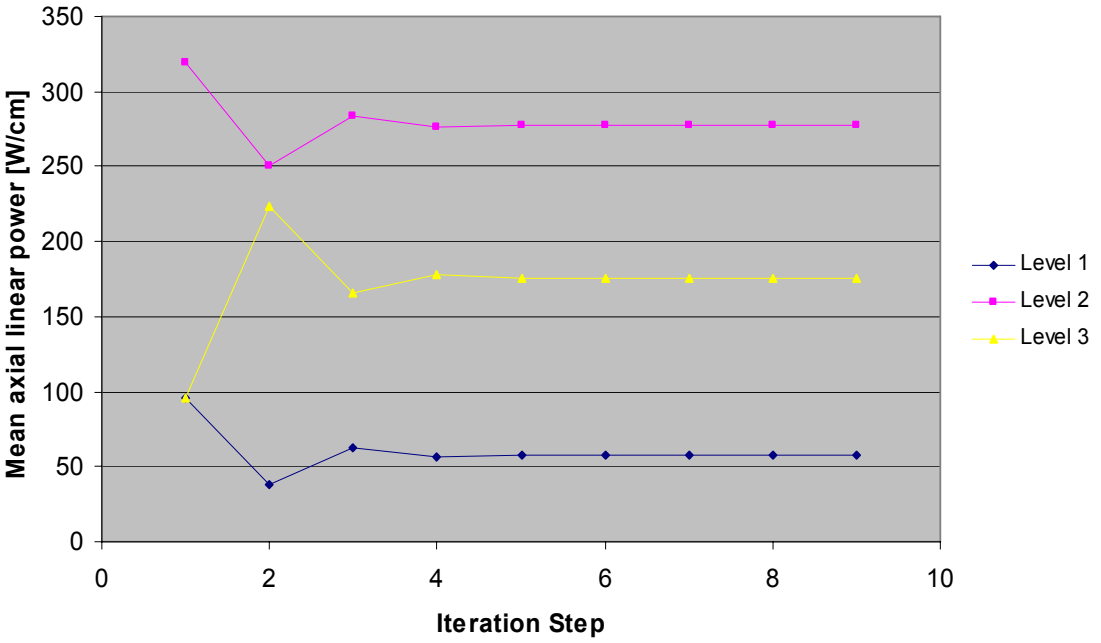


Fig. 5-19 Analysis of the convergence behaviour of the linear Power - W=75

The oscillating convergence mode shows in Fig. 5-20 is present for each level at this weighting parameter. These observations confirm what we presumed.

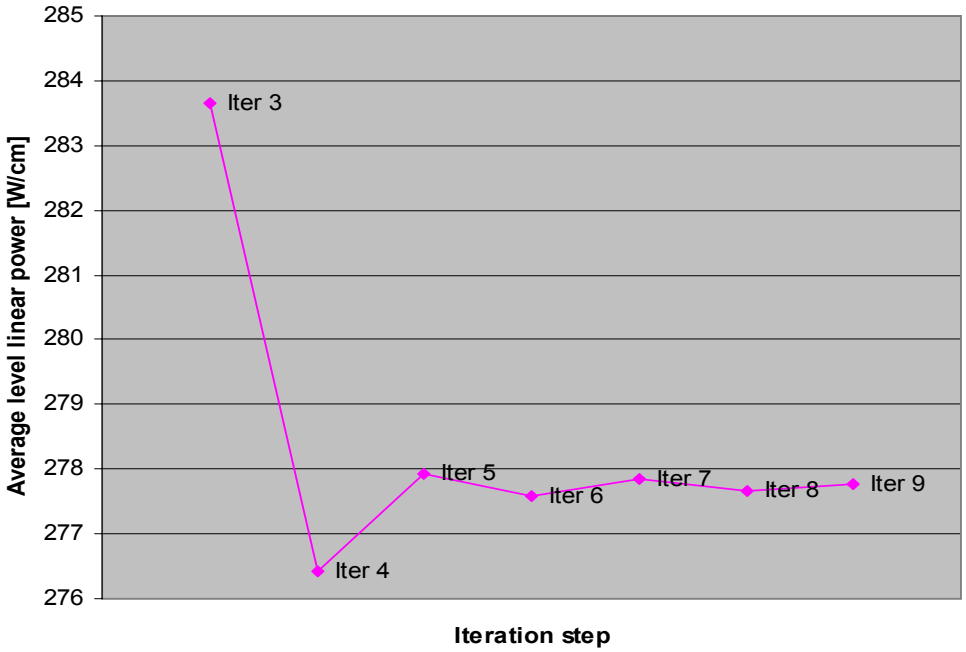


Fig. 5-20 Zoom on the convergence behaviour of level 2 - W=75

	Iterations								
Level	1	2	3	4	5	6	7	8	9
1	0,653	0,354	0,080	0,027	0,003	0,004	0,003	0,002	0,002
2	0,150	0,099	0,021	0,005	0,001	0,000	0,000	0,000	0,000
3	0,453	0,273	0,060	0,016	0,002	0,002	0,002	0,001	0,001

Table 5-7 Error to the reference solution - W=75

The Table 5-7 proves the proper convergence until the iteration 9. The error made to the reference decreases with an increasing number of iteration. The comparison with the convergence tables relating to the other weighting parameters shows that the obtainment of a defined accuracy for results needs a little bit more time than for the case W=65. On the other hand the behaviour is better than for W=50 which show an instable behaviour in the approach of the solution for small error divergences.

- **Summary for the simplified model**

It is not necessary for the validation of the calculated solution to have oscillating convergence behaviour. This type of convergence is however suitable and convenient because the attempted solution is framed by the results of each iteration. It is also easy during the run of a calculation to estimate the error made to a presumed stable solution, and to break up the run if a sufficient accuracy is reached.

The speed of reaching the results with an oscillating approach can be probably improved by choosing a weighting parameter a little bit higher than 65.

The evolution of the error to the presumed solution is given in Fig. 5-21 for different weighting parameters to summarize the discussion of the section. The relaxation offers an important acceleration to the convergence phenomena.

It is also important to optimize the choice of the weighting parameter to obtain a fast and convenient convergence. We see that for the iteration 3, the error to the reference solution is in one case 9% for W=50, whereas it amounts only quite 2% for W=65.

The results for the two first iterations are however the same for all weighting parameters. This is due to the relaxation function implementation. The relaxation affects the distributions of COBRA-TF and the discussion treats here the data coming from KARBUS. For the first iteration the relaxation is not effective since no previous results are available for the weighting. During the second iteration previous results of the COBRA-TF distributions exist but KARBUS deliver the linear power before that kcttni process the weighting of the previous distributions. The relaxation effects are then visible after the third iteration for linear power considerations.

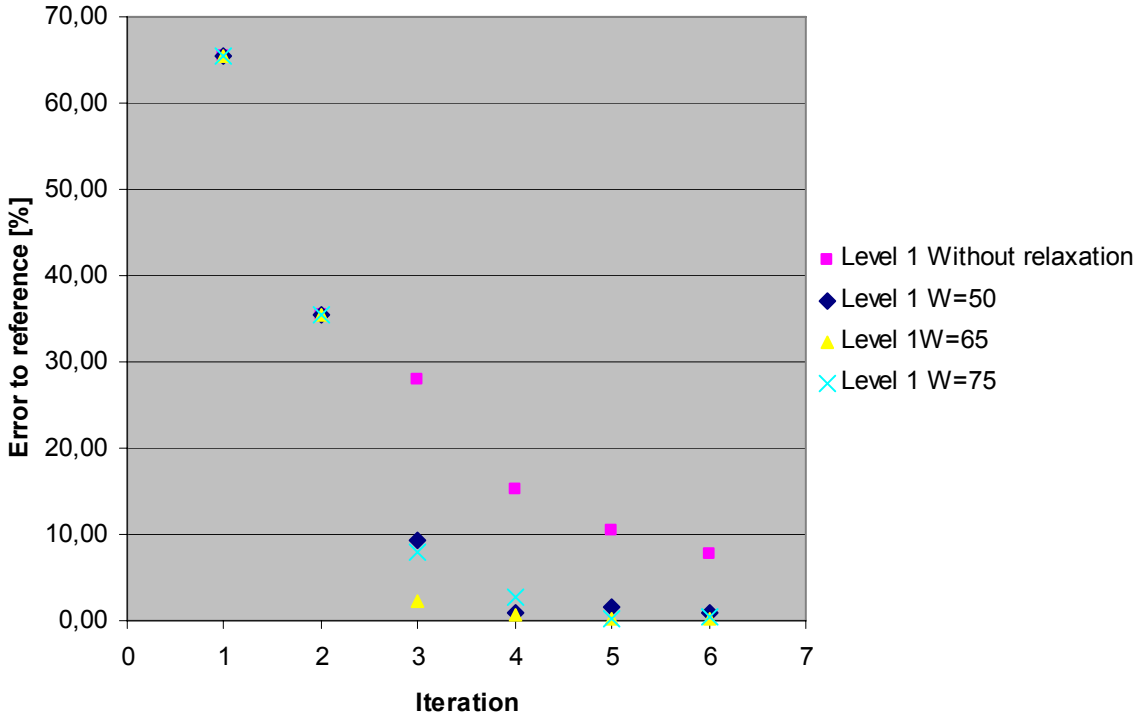


Fig. 5-21 Comparison of the approach to the reference solution for different weighting parameters for the level 1

5.4.2 Investigations on the detailed model

The precedent model was developed to provide of a better platform for testing the implementations on the coupling program. The faster obtainment of results led to study the conver-

gence at first with this model. In this way the pertinence of the calculations run with the detailed model was better. On the other hand we could only assume an equivalent behaviour between the simple and the detailed model. The present section discusses the results related to the nuclear power rating distributions.

For simplicity and due to the numerous axial layers, only three layers are presented in the next part: sections 5, 6 and 7 from the top of the bundle. They are the most relevant and representative, since they are the scene of the higher energy production. Therefore the comparison with safety criteria must be carefully made in this area.
The choice 5, 6, 7 instead of 6, 7, and 8 is motivated by the fact that the convergence behaviour of the layers 7 and 8 are similar. It was then more interesting to study the level 5 instead of the 8.

For the comparison, a reference solution was chosen. As we will see, the convergence behaviour is more problematic for the detailed model than for the previous case. The iteration 10 of a calculation without relaxation was selected as reference.

5.4.2.1 Results without relaxation

The first results without relaxation were presented in 5.1.3. The calculation was done with 4 iterations and it is not sufficient for the present topic. A new calculation with 10 iterations is presented here and the last iteration serves as reference solution. The results obtained in the next sections with different convergence criteria are compared with this reference.

The Fig. 5-22 presents the linear power distribution for 10 iterations without relaxation. The first four are naturally similar to those shown in Fig. 5-4. The oscillating behaviour to reach a stabilized solution is visible but must be investigated in details as for the simplified model, since it is difficult to see what is happening for example in the 6^{th} section.

An average value is calculated for the linear power within each level and the resulting values are drawn as function of the iteration number in the Fig. 5-23 to observe the convergence.

We can immediately observe a difference with the convergence behaviour of the simplified model. Whereas the behaviour obtained with the simplified model was a pure oscillating one, this detailed model shows irregularities and notably for the 6^{th} level between the iterations 2 and 3.
This phenomenon does not cause any problems since the variations are negligible (see Table 5-8) and permits the stabilization of the solution.

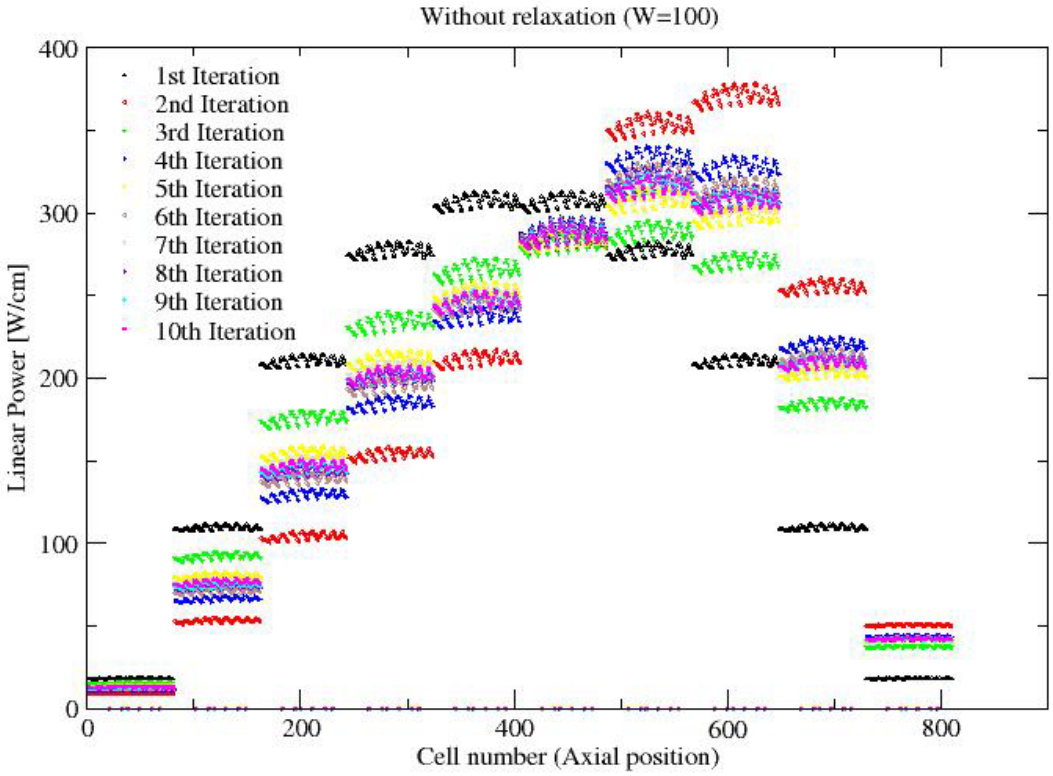


Fig. 5-22 Development of the linear power along the axial position - Without relaxation

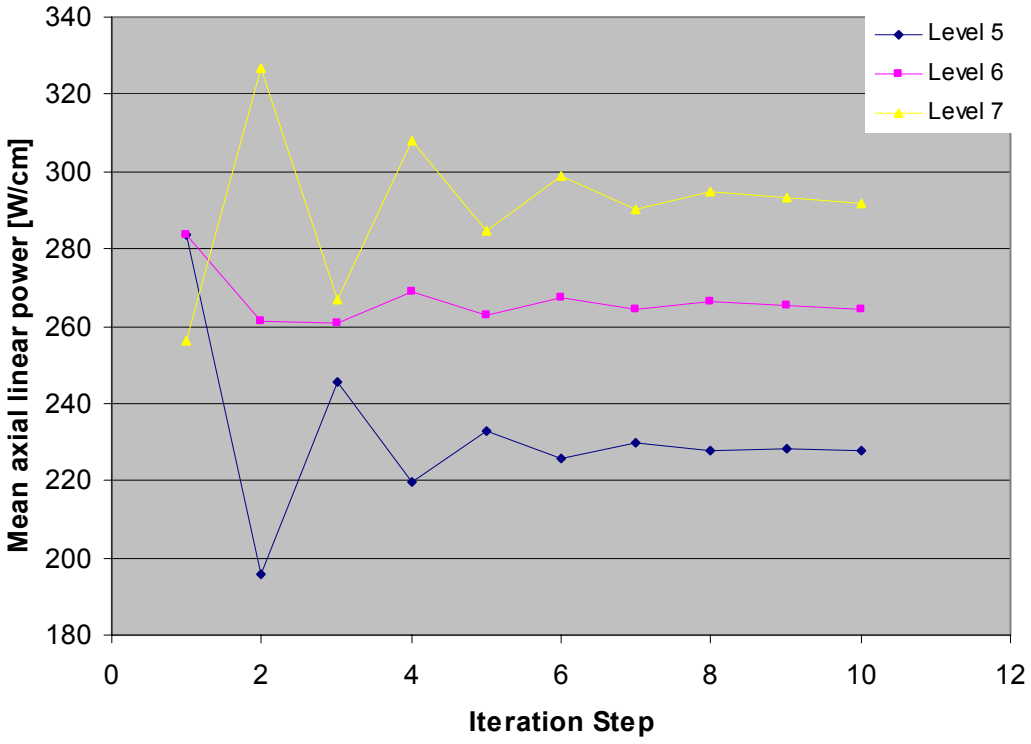


Fig. 5-23 Analysis of the convergence behaviour of the linear power - Without relaxation

	Iterations								
Level	1	2	3	4	5	6	7	8	9
5	0,245	0,142	0,077	0,038	0,021	0,010	0,008	0,001	0,001
6	0,073	0,013	0,015	0,017	0,006	0,012	0,000	0,008	0,003
7	0,121	0,120	0,086	0,056	0,025	0,025	0,005	0,012	0,005

Table 5-8 Error to the reference solution - Without relaxation

The errors to the reference are less than 2 percents at the iteration 7 for the levels considered. At contrary we can guess by showing the Fig. 5-22 that there are other levels (3 or 4 for example) which will need more time to converge. However they are of less importance for material limitation studies and we can accept finally a higher incertitude on results of these levels.

To summarize, the convergence phenomenon observed for the detailed model is more difficult to describe than for the simplified model. The singularities of the approach to a considered stable solution are difficult to explain but a stable solution is however conveniently reached.

5.4.2.2 Investigations on the detailed model under relaxation

- **W=75**

The first try was run with a weighting parameter equal to 75. It was assumed as a prudent value to accelerate the convergence without causing too many perturbations. The curves representing the evolution of the linear power in function of the iteration count and for each level are presented in Fig. 5-24. The influence of the relaxation method is clearly visible after the iteration 3.
The Fig. 5-25 specifies the convergence approach for the level 5, 6 and 7. It shows that the level 7 is the slowest to converge, as it was the case without relaxation. The maximum of the power curve moves since the iteration 4 inside this level. It is then of high importance. However the errors in comparison to the reference solution are negligible since the iteration 4 with values lower than 1.5 percents (see Table 5-9).

The Table 5-9 shows also perturbations while approaching a stabilized solution and this is confirmed by the Fig. 5-26. The figure zooms on the convergence approach for the level 7 since the iteration 3. Same disturbances are also present for the levels 5 and 6 since the iteration 7.

The amplitude of the perturbations is low but this observation led to choose convenient results without relaxation as reference solution, rather than results from relaxed convergence. It is namely difficult to say by showing the last iterations where the final stable solution should be.

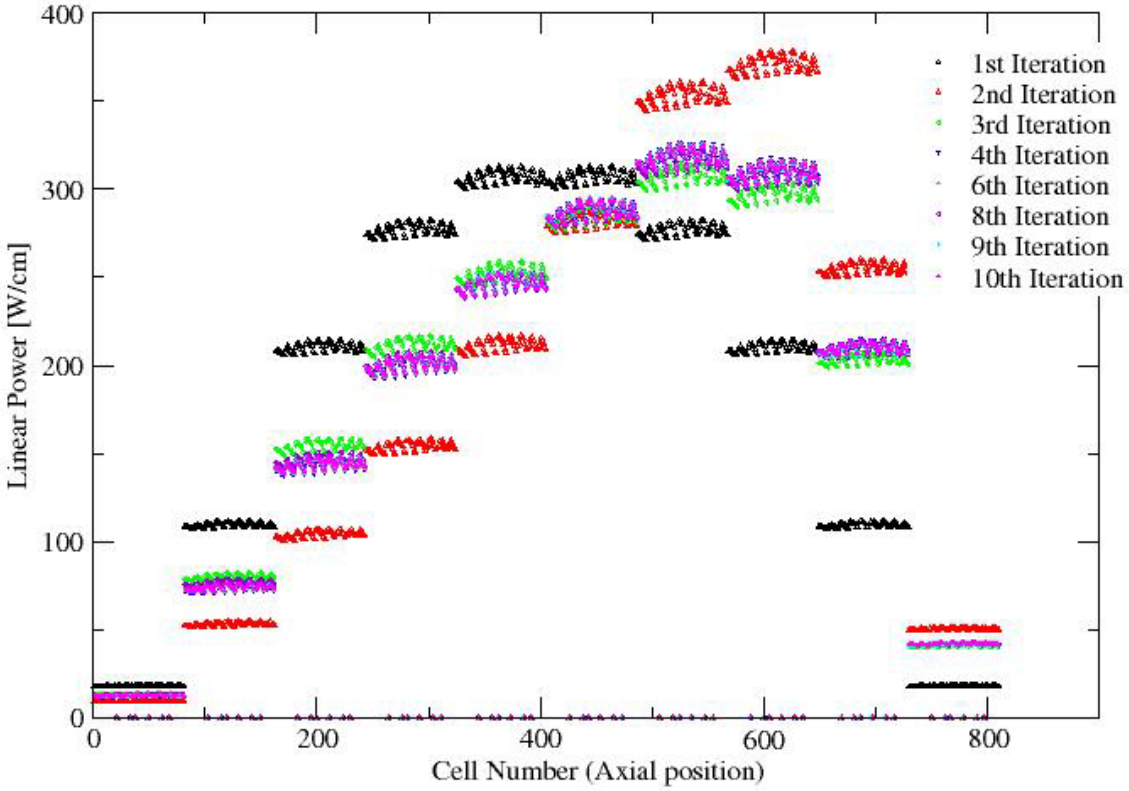


Fig. 5-24 Development of the linear power along the axial position - W=75

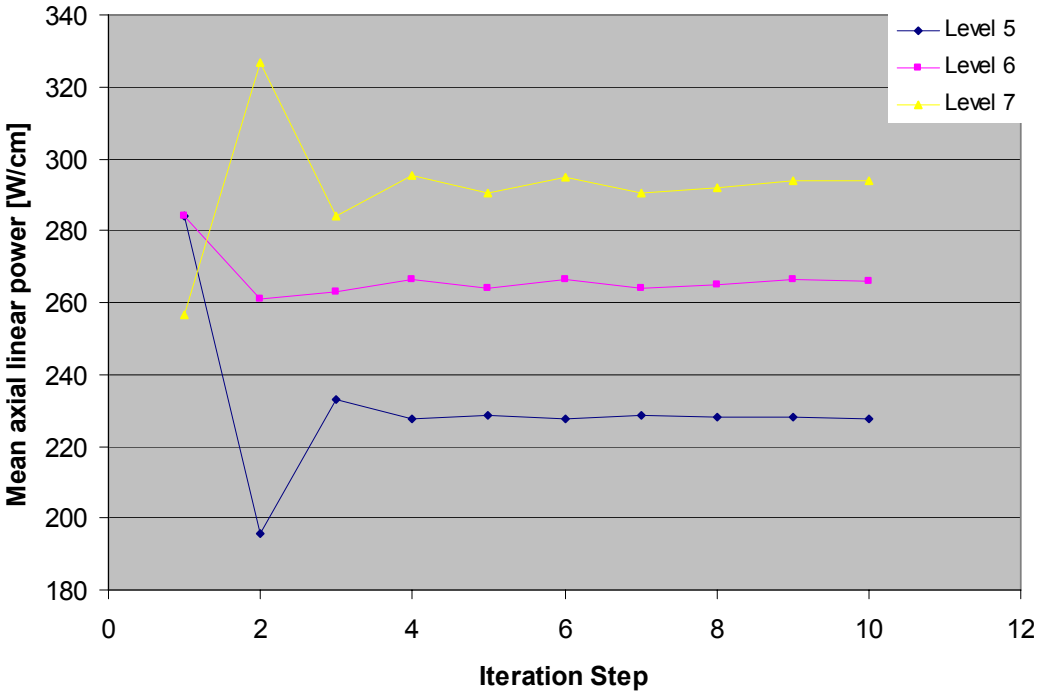


Fig. 5-25 Analysis of the convergence behaviour of the linear power - W=75

	Iterations									
Level	1	2	3	4	5	6	7	8	9	10
5	0,245	0,142	0,023	0,003	0,003	0,001	0,003	0,002	0,002	0,001
6	0,073	0,013	0,005	0,008	0,001	0,007	0,001	0,001	0,006	0,005
7	0,121	0,120	0,026	0,013	0,004	0,011	0,004	0,000	0,008	0,008

Table 5-9 Error to the reference solution - W=75

Fig. 5-26 Zoom on the convergence behaviour since the iteration 3 (level 7) - W=75

- **W=65**

The last weighting parameter showed faster convergence capabilities than without relaxation. But the relaxation method introduces disturbances in the convergences. W=75 was a prudent value but it was sure by observing the results without relaxation that the convergence could be more accelerated with a lower parameter. A lower weighting parameter is chosen in this section to observe the effects on the results. A figure showing the development of the linear power along the axial position is not given since the shape is nearly the same as Fig. 5-24. The focus ports on the convergence observation presented in Fig. 5-27 and in Table 5-10.

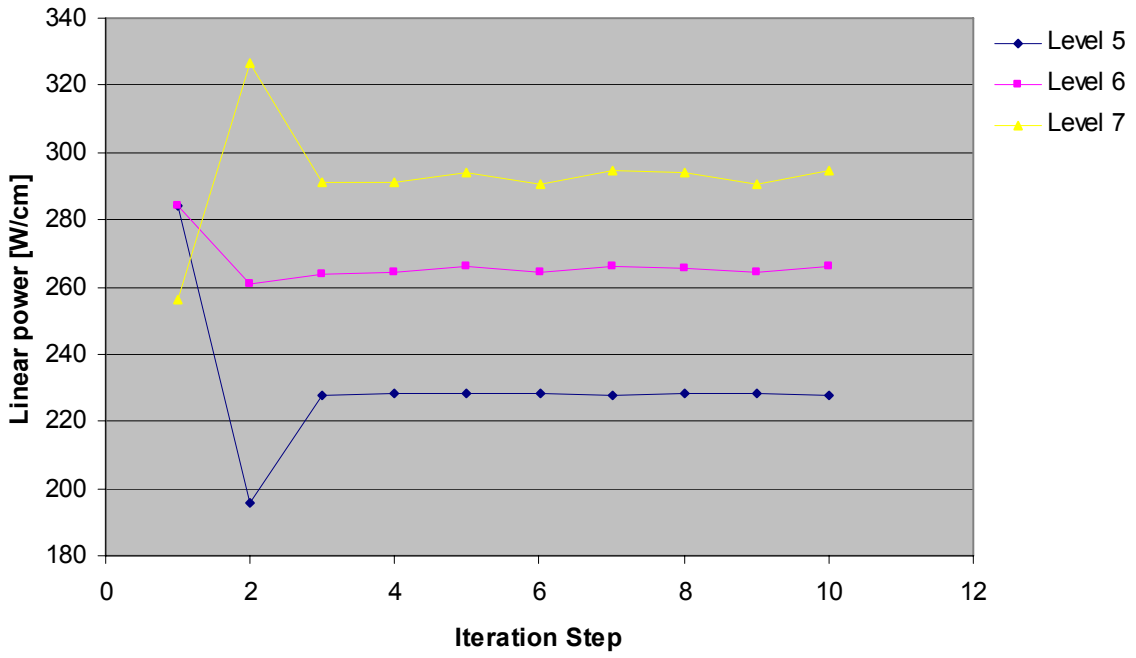


Fig. 5-27 Analysis of the convergence behaviour of the linear power - W=65

	Iterations									
Level	1	2	3	4	5	6	7	8	9	10
5	0,245	0,142	0,001	0,001	0,000	0,003	0,001	0,000	0,003	0,001
6	0,073	0,013	0,003	0,001	0,006	0,001	0,007	0,005	0,001	0,007
7	0,121	0,120	0,001	0,002	0,009	0,003	0,011	0,007	0,003	0,010

Table 5-10 Error to the reference solution - W=65

Like before for W=75 or without relaxation, the level 5 converges fastest and the level 7 is always the most problematic.

After the first two iterations where the relaxation is not effective, a really fast approach to the reference solution is obtained as shown for the iteration 3 and 4 in Table 5-10. The convergence is faster than for W=75 in the previous section. Results have an error lower than 1 percent at the iteration 3. Even if the error are then always under 2 percents for the ten iterations, the table shows an irregular behaviour which must be closer analysed.

The Fig. 5-28 present the typical convergence shape obtained for the levels 6 and 7 from the iteration 3. In spite of the fast approach the convergence is lost. The convergence of the level 5 is also problematic (Fig. 5-29). The convergence was a little bit slower with W=65 but the error after the 6^{th} iterations were quite stable and under the percent. That is not the case for the present value of the weighting parameter.

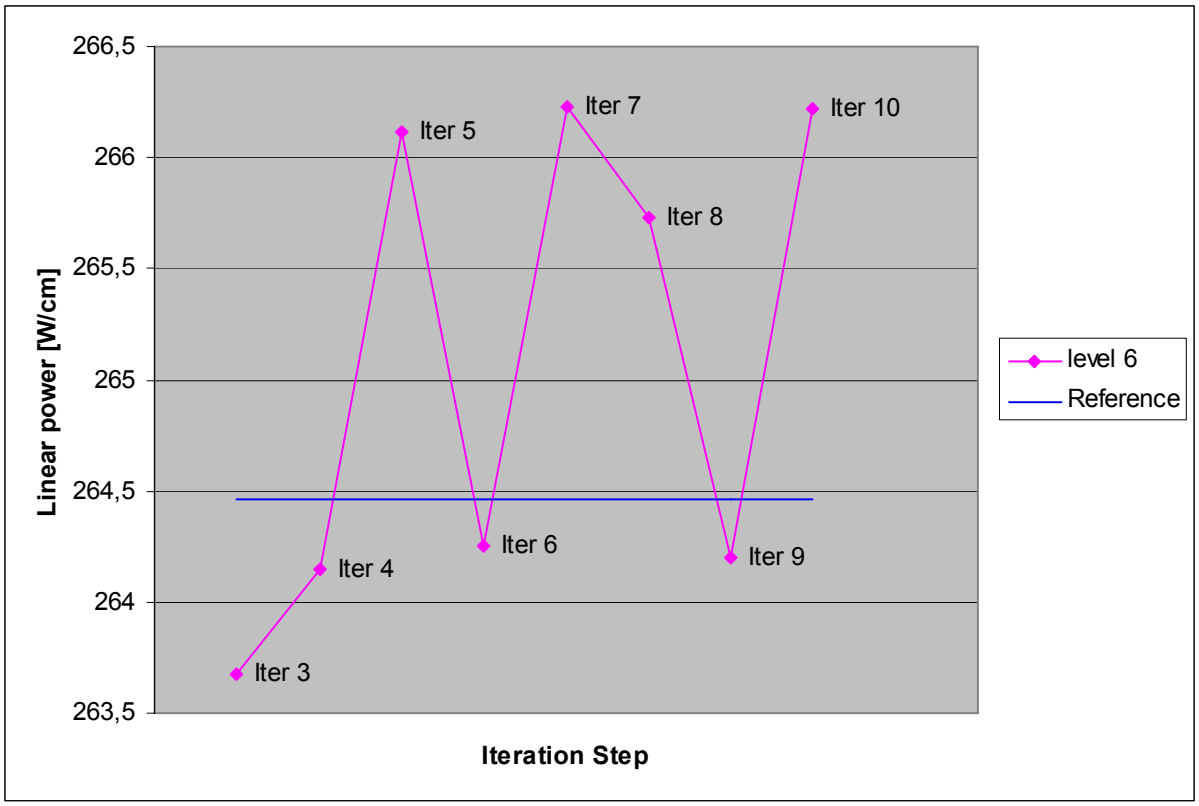


Fig. 5-28 Zoom on the convergence behaviour from the iteration 3 (level 6) - W=65

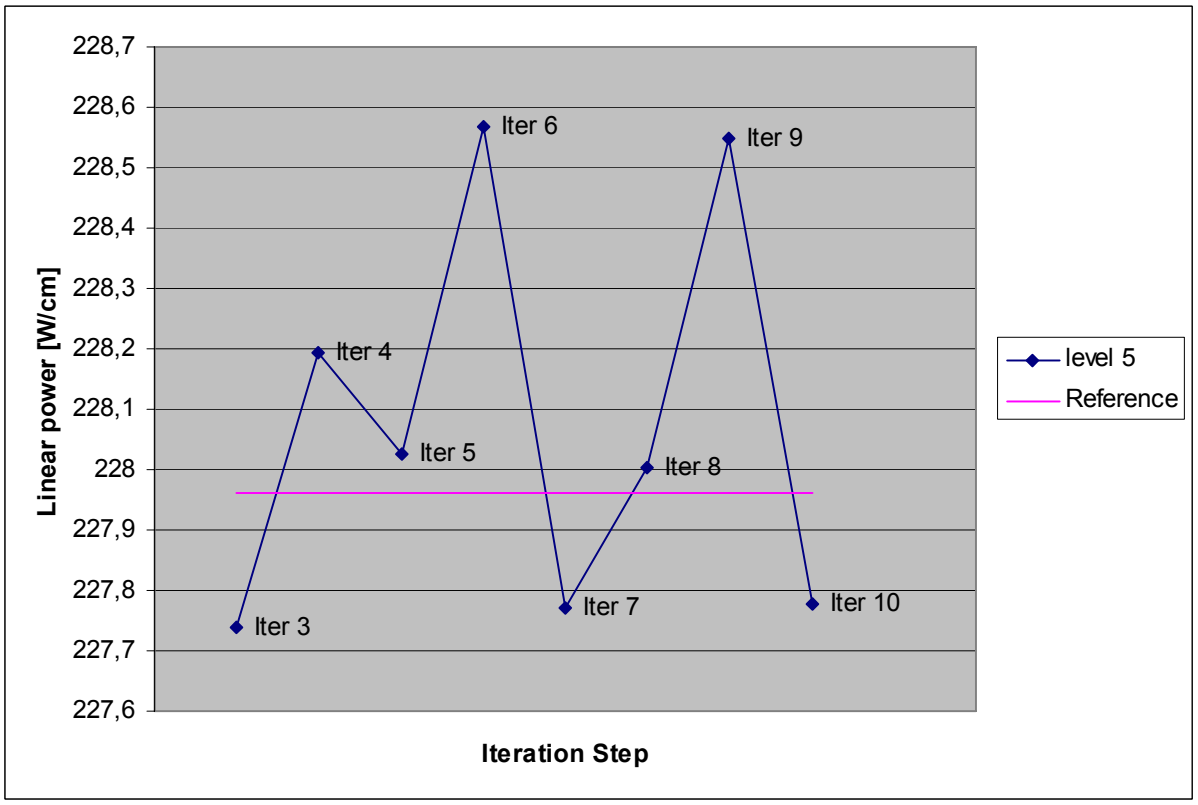


Fig. 5-29 Zoom on the convergence behaviour from the iteration 3 (level 5) - W=65

- **W=58**

A calculation with W=58 was also achieved and the presentation in details of results is not interesting since they provide no other remarks than the case W=65. The evolution of results is very chaotic (see Fig. 5-30) and the error table is given below in Table 5-11.

	Iterations									
Level	1	2	3	4	5	6	7	8	9	10
5	0,245	0,142	0,016	0,001	0,000	0,000	0,003	0,001	0,000	0,003
6	0,073	0,013	0,003	0,003	0,004	0,005	0,001	0,006	0,005	0,004
7	0,121	0,120	0,013	0,006	0,006	0,007	0,003	0,010	0,007	0,005

Table 5-11 Error to the reference solution - W=58

Fig. 5-30 Zoom on the convergence behaviour from the iteration 3 (level 7) - W=65

- **Summary for the detailed model:**

The convenient description of the problem by the detailed model may provide a good accordance with the real distributions. The iterative process leads to a stabilized solution after 10 iterations without relaxation, which was chosen as reference. The first iteration for the linear power calculated by KARBUS/DANTSYS, and the reference solution computed with the coupling program are compared in Fig. 5-31. The power distribution for the first iteration has a typical cosine shape due to the assumption of constant material properties for the neutronic calculation. Then the feedback with the thermal hydraulic is taken into account and the initial shape is corrected with an increase of the maximum linear power and a displacement of this maximum to the bottom of the assembly. The divergence between these two curves shows the pertinence of the approach developed in this work.

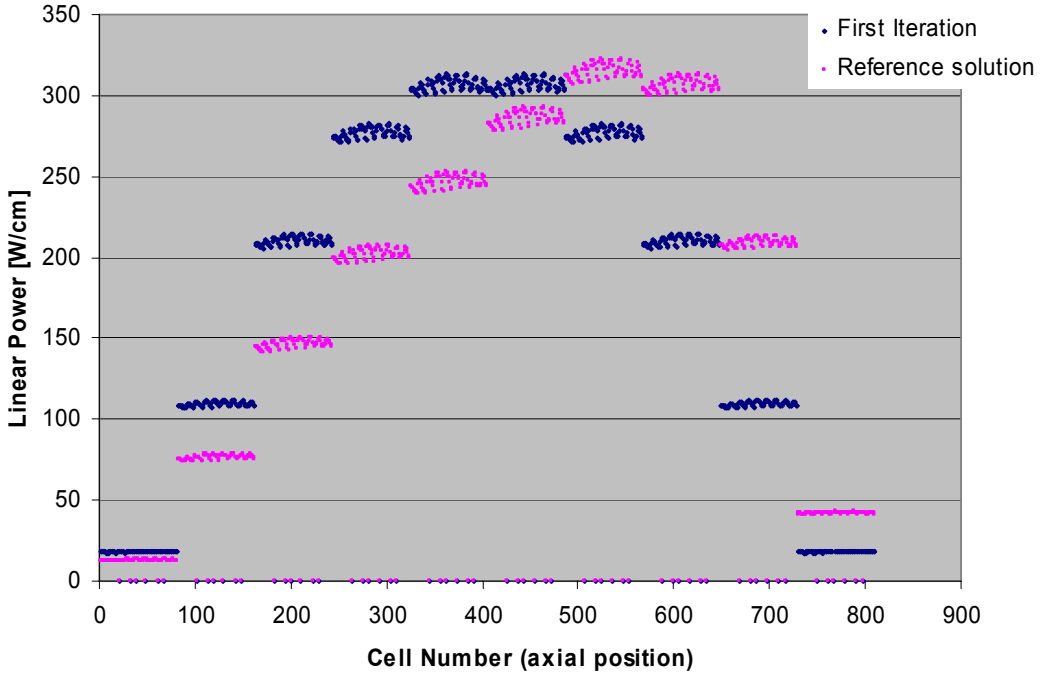


Fig. 5-31 Comparison of the first distribution of the linear power and the reference solution with the coupling procedure COBRAP

Concerning the convergence analysis, the investigations on the detailed model are more difficult to discuss than those for the simplified model, even without relaxation. The relaxation method provides however good abilities to accelerate the calculation, but the choice of the weighting parameter must be carefully considered. The evolutions within the studied levels are very different and the following figures present for each level a summary of the convergence analysis discussion.

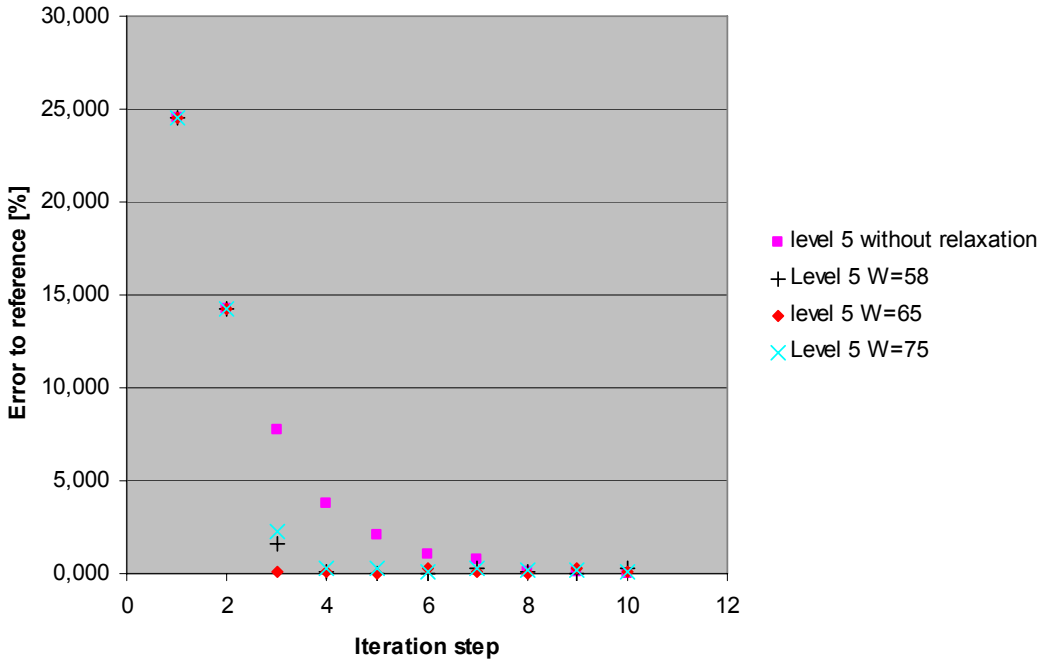


Fig. 5-32 Comparison of the approach to the reference solution for different weighting parameters for the level 5

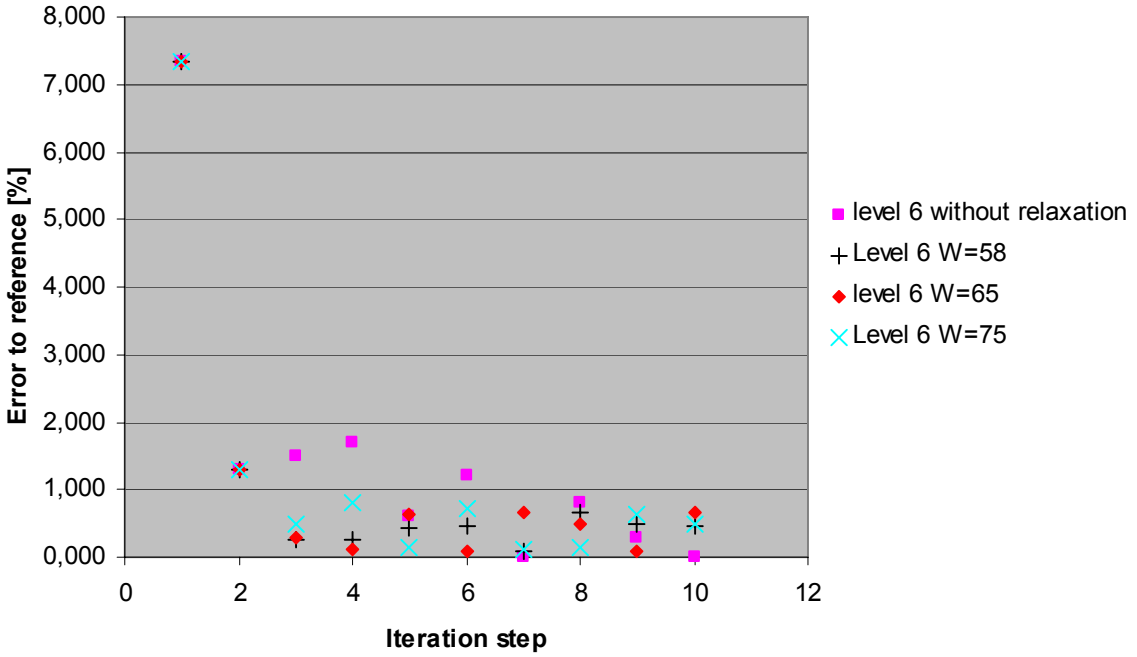


Fig. 5-33 Comparison of the approach to the reference solution for different weighting parameters for the level 6

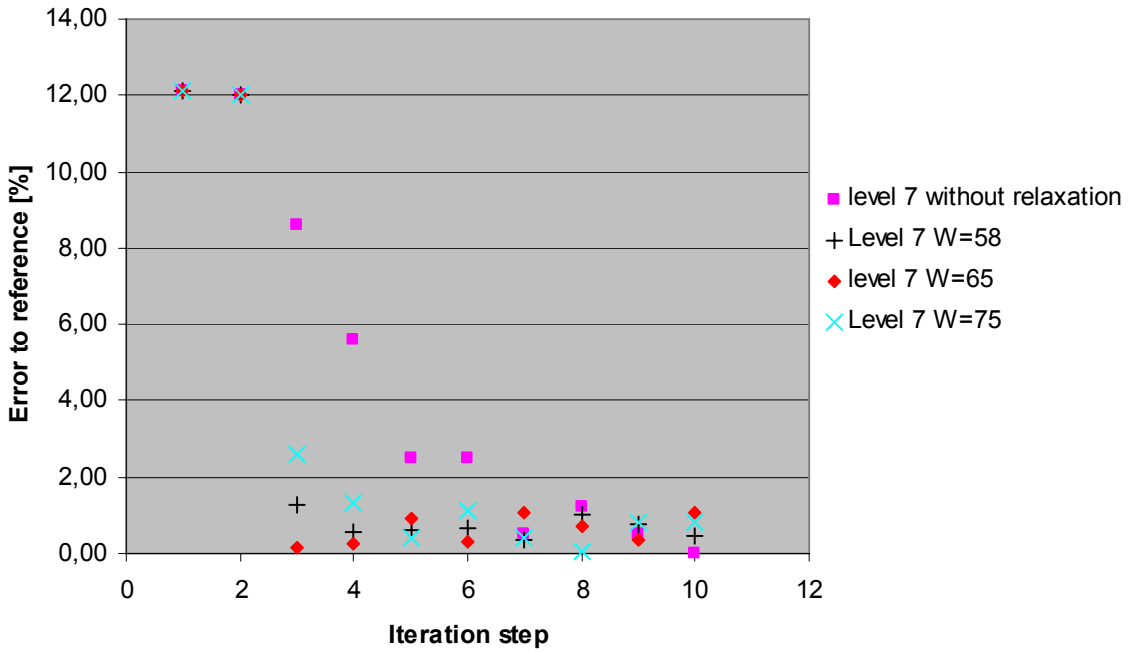


Fig. 5-34 Comparison of the approach to the reference solution for different weighting parameters for the level 7

The weighting parameter W=65 provides the faster approach with a very low error after the iteration 3. The following evolution is chaotic but this is also the case for other parameters and without relaxation. A weighting parameter equal to 65 was already a judicious choice for the simplified model.

The main problem is the way of testing the convergence in kcttni. The reference solution is not known and the program tests the variation rate between the past distributions and the new ones. Chaotic approaches can cause the stop of the calculation because the convergence criterion is fulfilled while the error to the real solution is higher than the convergence criteria. This can not be the case if the oscillating behaviour is conserved, since the final solution is contained between the two last iterations and the error to the real solution is obligatory smaller than the variation rate.

For investigations on a new problem, higher values of the weighting parameter have to be preferred. Another possibility could be to specify a very small convergence criterion in kcttni and to survey the evolution of the variation rates. These are written for each iteration in the input file of kcttni and can be checked. The run can be then manually interrupted.

6 Summary and outlook

The intention to couple neutronic and thermal hydraulic calculations for steady-state investigations on the fuel pin level in nuclear reactor cores required firstly the correct choice of codes. COBRA-TF and the neutronic combination KAPROS-E/DANTSYS were chosen and the modalities of their applications on this type of problem were presented. Interface programs were required to organize the data flow between the two investigation domains and their conceptions necessitated a deeper insight in the neutronic and thermal hydraulic tools to produce an effective solution for the coupling.
Two interfaces, KCNTTI (**K**APROS-E **C**OBRA **N**eutronic **T**o **T**hermal hydraulic **I**nterface) and KCTTNI (**K**APROS-E **C**OBRA **T**hermal hydraulic **T**o **N**eutronic **I**nterface), were written in FORTRAN-90 and validated in comparison with the work in reference [18]. The coupling code was successfully implemented in the neutron system KAPROS-E via modifications of the steering procedure COBRAP.
First investigations with COBRAP on a GRS PWR assembly benchmark problem proposal showed encouraging results. On the other hand the time consumption to obtain stabilized results is important, due to the iterative approach of the problem. In order to accelerate the convergence of results, a relaxation method has been implemented. This method consists in mixing the new distributions computed by the thermal hydraulic code with distributions from the previous iteration for the next feedback to the neutronic calculation. It results in a satisfying decrease of the computing time by a factor of about two. In contrast the relaxation can introduce disadvantageous effects on the convergence approach. An oscillating convergence is convenient since the error to a stabilized solution is framed by the results from the two past iterations. Then the error made in comparison to the final solution is easily evaluated. For other cases this error is hard to predict and this can be problematic with a right use of the convergence test implemented in KCTTNI.

With this approach the prediction capability is extended to a fuel pin description in PWR. It permits an accurate evaluation of properties margins with respect to safety criteria and such approaches represent nowadays an interesting improvement direction. This is an important intermediate step to be able then to improve and use the pin-power reconstructions methods in advanced transient calculations for PWR.

Literature

[1] D. Barber, T. Downar, and W. Wang, „Final Completion report for the coupled RELAP5/PARCS Code, Report PU/NE-98-31, Purdue University/Scientech, 1998

[2] F. Odar, C. Murray, R. Shumway, M. Bolander, D. Barber, J. Mahaffy, „TRAC/RELAP Advanced Computational Engine (TRACE) v4.0 User's Manual", US NRC, 2003

[3] COBRA/TRAC - A Thermal-Hydraulics Code for Transient Analysis of Nuclear Reactor Vessels and Primary Coolant Systems, Pacifist Northwest Laboratory

[4] G. Buckel, W. Höbel; "Das Karlsruher Programmsystem KAPROS" Teil 1, KFK 2253, 1976

[5] R.E. Alcouffe, R.S. Baker, F.W. Brinkley, D.R. Marr, R.D. O'Dell, and W.F. Walters, „DANTSYS: A Diffusion Accelerated Neutral Particle Transport Code System", LA-12969-M, 1995

[6] D. PORSCH, at al.; "Spezifikation eines DWR-Brennelementes UO2 (4 w/o U-235) 18x18-24, für Vergleichsrechnungen", Framatome ANP GmbH, Erlangen, 20. July 2004

[7] „Weiterentwicklung und Verifikation/Validierung von Rechenprogrammen für die Reaktorsicherheit", (GRS)mbh, Köln, November 2005

[8] I. Dor, „The CATHARE V2.5 3D Module; Description and Validation", CATHARE-NEPTUNE Seminar, Grenoble, France, 2004

[9] C.H.M Broeders, "Entwicklungsarbeiten für die neutronenphysikalische Auslegung von Fortschrittlichen Druckwasserreaktoren (FDWR) mit kompakten Dreiecksgittern in hexagonalen Brennelementen", KFK 5072, 1992

[10] D. Basile "COBRA-EN, an Upgrade Version of COBRA-3C/MIT Code for Thermal-Hydraulic Transient Analysis of Light Water Reactor Fuel Assemblies and Cores", ENEL, December 1987

[11] J.J. Duderstadt, L.J Hamilton, "Nuclear Reactor Analysis", University of Michigan, 1976

[12] A. Bergeron, "Assessment of the FLICA-IV code on rod bundle experiments", NURETH-9, April 2001

[13] J. Bussac, P. Reuss, „Traité de Neutronique", Hermann ed. ,1985

[14] N.E. Todreas, M.S. Kazimi, „Nuclear Systems I: Thermal Hydraulic Fundamentals“, Massachusetts Institute of Technology, 1993

[15] M.J Thurgood, K.R. Crowell, J.M Kelly, ”COBRA-TF Equations and constitutive models”, Battelle, Pacific Northwest Laboratory, Richland, Washington, June1981

[16] C. Herer, D. Gallori,”Thermohydraulique des réacteurs à eau sous pression”, Techniques de l’Ingénieur, BN 3050, 2000

[17] N.E. Todreas, M.S. Kazimi, „Nuclear Systems II: Elements of Thermal Hydraulic Design“, Massachusetts Institute of Technology, 2001

[18] B. Becker, “Investigations of COBRA-EN and COBRA-TF for core and assembly simulations and coupling of COBRA-TF and KARBUS”, FZKA Technical report, 2005

[19] D.G. Cacuci, Vorlesung “Energiesysteme” Teil 2, Institut für Kerntechnik und Reaktorsicherheit an der Universität Karlsruhe (TH), 2003

[20] R.T Lahey Jr, F.J Moody,”The thermal-hydraulics of a boiling water nuclear reactor”, American Nuclear Society, 1977

[21] G.B. Bruna, B. Guesdon,”Méthodes de calcul neutronique de coeur”, Techniques de l’Ingénieur, B 3070, 1996

[22] J.F Briesmeister, Editor, “MCNP- A General Monte Carlo Code for Neutron and Photon Transport, Version 3 A”, LA-7396-M Rev.2, UC-32, 1986

[23] A. Baur, L. Bourdet, G. Dejonghe, J. Gonnord, A. Monnier, J.C. Minnal, T. Vergnaud, „TRIPOLI 2: Three Dimensional Polyenergetic Monte Carlo Radiation Transport Program, Volume I“, Engineering Physics Information Centers OLS 80-110, 1980

[24] L.B. Levitt, R.C. Lewis, „VIM-1, A Non-Multigroup Monte Carlo Code for Analysis of Fast Critical Assemblies“, AI-AEC-12951, Atomic International (1970)

[25] P. Reuss, „-Bases de neutronique - Physique et calcul des réacteurs”, Technique de l’Ingénieur, BN 3015, 2006.

[26] R. Barjon, „Physique des réacteurs nucléaires”, Institut des Sciences Nucléaires, Grenoble, 1993.

[27] P. Oberle,” Erstellung einer 78-Gruppenkonstanten Bibliothek mit Energien bis 150 MeV für KAPROS”, Diplom Thesis, Institut für Reaktorsicherheit, Forschungszentrum Karlsruhe, 2004

[28] C.H.M Broeders, R. Dagan, V. Sanchez, A. Travleev, „KAPROS-E: Modular Program System for Nuclear reactor Analysis, Status and Results of selected Applications“, Jahrestagung Kerntechnik Düsseldorf, 2004

[29] D. Woll; “Introduction to the use of the Unix-Version of the Karlsruhe PROgram System KAPROS”, Forschungszentrum Karlsruhe, FZKA 6280, 1999

[30] C. Frepoli, L.E. Hochreiter; “Description of the Modifications included in the RBHT Version of COBRA-TF”, Penn State University, Mechanical and Nuclear Engineering department, 2000

[31] V. Sanchez, B. Becker, C.H.M Broeders, P. Vinson, „A Procedure for coupled Neutron Physics Calculation of the pinwise Power Distribution within a Fuel Assembly“, Jahrestagung Kerntechnik Aachen, 2006

[32] C.H.M Broeders, V. Sanchez, E. Stein, A. Travleev,”Validation of coupled neutron physics and thermal-hydraulics analysis for HPLWR”, ICAPP’03, Cordoba, 2003

[33] C.H.M Broeders, private communication, 2006

[34] V. Sanchez, „Steamline break transients“, Topic 5.1, „The 2003 Frédéric Joliot/Otto Hahn Summer School“, Forschungszentrum Karlsruhe, 2003

Annex A Assumptions and notations for thermal hydraulic conservation equations in subchannel tools

See reference [3] for more details.

Assumptions:

- Gravity is the only body force
- There is no volumetric heat generation or radiation
- The pressure is the same in all phases
- The dissipation and the material derivative of the pressure can be neglected in the enthalpy formulation of the energy equation

These assumptions are usually justified for situations encountered in reactor safety analysis.

Notations used in the formulas (2.1), (2.2) and (2.3).

α_k = average k-phase void fraction
ρ_k = average k-phase density
$\underline{U}_k$ = average k-phase velocity
Γ_k = average rate of mass transfer to phase k from the other phases
$\underline{g}$ = acceleration of gravity
p = average pressure
$\underline{\underline{\tau}}_k$ = average k-phase viscous stress tensor (stress deviator)
$\underline{M}_k^{\Gamma}$ = average supply of momentum to phase k due to mass transfer to phase k
$\underline{M}_k^{d}$ = average drag force on phase k by the other phases
$\underline{\underline{T}}_k^{T}$ = k-phase turbulent (Reynolds) stress tensor
h_k = average k-phase enthalpy
$\underline{Q}_k$ = average k-phase conduction vector
$\underline{q}_k^{T}$ = k-phase turbulent heat flux
h_k^{i} = surface average enthalpy of phase k

Annex B Common forced convection correlations and physical properties of some typical coolants

Common Forced Convection Correlations

$Pr < 0.1$ (liquid metals)

$Nu = 6.3 + 0.03\ (Re\ Pr)^{0.8}$ (constant heat flux)

$Nu = 4.8 + 0.03\ (Re\ Pr)^{0.8}$ (constant temperature)

$0.5 < Pr < 1.0$

$Nu = 0.022\ Pr^{0.6}\ Re^{0.8}$ (constant heat flux)

$Nu = 0.021\ Pr^{0.6}\ Re^{0.8}$ (constant temperature)

$1.0 < Pr < 20$ (water and light liquids)

$Nu = 0.0155\ Pr^{0.6}\ Re^{0.83}$

$Pr > 20$ (oils and other viscous liquids)

$Nu = 0.0118\ Pr^{0.3}\ Re^{0.9}$

Physical Properties of Some Typical Coolants

Coolant	T_{inlet} [°C]	ΔT_{core} [°C]	Pressure [bar]	Δp_{core} [bar]	$\overline{u_z}$ [cm/sec]	Density [g/cm^3]	h_s [W/cm^2°K]
H_2O	300	20	100	2	400	.70	2.8–4.5
He	500	400	40	1	16,000	.0024	.005–.05
Na	400	150	10	10	400	85	2.2–5.5

Annex C Input file for KARBUS

Input for the GRS benchmark proposal. Detailed model.

```
*$ KSPARM VERON
*KSIOX DBN=INPUT ARCHIV,TYP=CARD,PMN=PRDUM
32  'SEQ ' 'GEN '
1024 'KSSKUX ARCHIVE LINUX           '
8    ' GRS DWR BENCHMARK             '
*$*$
*KSIOX DBN=INPUT CHICOR,TYP=CARD,PMN=PRDUM
  'DAFI'
*$ 'CNTR'
  'DUMM'
  'NOPR'
*$*$
*KSIOX DBN=INPUT WETHES,TYP=CARD,PMN=PRDUM
'BEGI'
*$*$
*KSIOX DBN=INPUT MIXCOP,TYP=CARD,PMN=PRDUM
'MISP' 810
1       1       1       1       1       1       1       1       1
1       1       1       1       1       1       1       1       1
1       1       2       1       1       1       1       1       1
1       1       1       1       1       2       1       1       1
1       2       1       1       1       1       1       1       1
1       1       1       2       1       1       1       1       1
1       1       1       1       1       1       2       1       1
1       1       1       1       2       1       1       1       1
1       1       1       1       1       1       1       1       1

.....
CUT
.....
1       1       1       1       1       1       1       1       1
1       1       1       1       1       1       1       1       1
1       1       2       1       1       1       1       1       1
1       1       1       1       1       2       1       1       1
1       2       1       1       1       1       1       1       1
1       1       1       2       1       1       1       1       1
1       1       1       1       1       1       2       1       1
1       1       1       1       2       1       1       1       1
1       1       1       1       1       1       1       1       1


*$*$
```

```
*KSIOX DBN=INPUT WEPERS,TYP=CARD,PMN=PRDUM
'BEGI'
*$ 'BUCI' 5  1.E-3
*$*$
*KSIOX DBN=INPUT WEFILE,TYP=CARD,PMN=PRDUM
'DUMM'
'NOPR'
*$*$
*KSIOX DBN=INPUT WEKCPM,TYP=CARD,PMN=PRDUM
*$ 'CNTR'
*$ 'PFLX'
  'NOPR'
*$*$
*KSIOX DBN=INPUT REMCOR,TYP=CARD,PMN=PRDUM
*$ 'CNTR'
*$ 'NCOR' 1 28
  'DUMM'
  'NOPR'
*$*$
*KSIOX DBN=INPUT GRDASQ,TYP=CARD,PMN=PRDUM
*$ 'CNTR'
  'DUMM'
  'TYPG'
*$*$
*KSIOX DBN=INPUT BURNUP,TYP=CARD,PMN=PRDUM
'NAMS' 'H      ' 'FPP 9   '
'ORDT' 36
'CUTO' 1.E-3
'BUTB'
69
                'KR 83   ' 'ZR 91   ' 'ZR 93   ' 'ZR 96   '
'NB 95   ' 'MO 95   ' 'MO 97   ' 'MO 98   ' 'MO100   ' 'TC 99   '
'RU101   ' 'RU102   ' 'RU103   ' 'RU104   ' 'RU106   ' 'RB101   '
'RB103   ' 'RH103   ' 'RH105   ' 'PD105   ' 'PD106   ' 'PD108   '
'AG109   ' 'CD111   ' 'CD113   ' 'IN115   ' 'I 127   ' 'I 129   '
'XE131   ' 'XE132   ' 'XE133   ' 'XE134   ' 'XE135   ' 'CS133   '
'CS134   ' 'CS135   ' 'LA139   ' 'CE141   ' 'PR141   ' 'PR143   '
'ND143   ' 'ND144   ' 'ND145   ' 'ND146   ' 'ND147   ' 'ND148   '
'ND150   ' 'PM147   ' 'PM148   ' 'PM48M   ' 'SM147   ' 'SM149   '
'SM150   ' 'SM151   ' 'SM152   ' 'EU153   ' 'EU154   ' 'EU155   '
'GD154   ' 'GD155   ' 'GD156   ' 'GD157   ' 'GD158   ' 'GD160   '
'TB159   ' 'DY164   ' 'LU176   ' 'SM148   ' 'PD107   '
*$*$
*KSIOX DBN=CELL SPEC,TYP=CARD,PMN=PRDUM,IND=1
*$ MODULE NAME FOR GROUPCONSTANT CALCULATIONS
  'GRUCEL  '
*$ SN IACC ITMAX  EPS  NBUC BGHT BWDT
```

```
  04   1    06  1.E-4   0    0.   0.
*$ N(I),I=1,NM
  16 3 4
*$ NTEXT(I),I=1,MIN(0,15)
  ' STANDARD CELSPECIFICATION DATE FOR LWR CELLS              '
*$*$
*KSIOX DBN=CELL SPEC,TYP=CARD,PMN=PRDUM,IND=2
*$ MODULE NAME FOR GROUPCONSTANT CALCULATIONS
  'GRUCAL  '
*$ SN IACC ITMAX  EPS  NBUC BGHT BWDT
  04   1    06  1.E-4   0    0.   0.
*$ N(I),I=1,NM
  16 3 4
*$ NTEXT(I),I=1,MIN(0,15)
  ' STANDARD CELSPECIFICATION DATE FOR LWR CELLS              '
*$*$
*KSIOX DBN=INPUT KARBUS,TYP=CARD,PMN=PRDUM
'EFTB'
10 5.5
'U 234   '  193.   2.45
'U 235   '  193.5  2.418
'U 236   '  193.   2.45
'U 238   '  197.2  2.81
'NP237   '  200.   2.5
'PU238   '  200.   2.55
'PU239   '  201.8  2.871
'PU240   '  195.   2.9
'PU241   '  202.1  2.972
'PU242   '  200.   3.00

'BUCO' 'O101'
*$ 'NGLC'
'ARCA'
'ARBU'
*$ 'CHIT' 2
'NOPR'
'WN2N'
*$ 'POWI'
*$ 'BU1D'    09
*$ 'BU1D'    01
 1
 10.00
-170.50


 1
 115.00
```

```
-170.50

 ……..
CUT
……
  1
 125.00
-170.50

'TPOW' 10*5.38610
*$ 'CNTR'
*$*$
*KSIOX DBN=INPUT DXBURN,TYP=CARD,PMN=PRDUM,IND=1
'GRC1' 4 1  2  3  28
'GRC2' 4 1  2  3  28
'BUCO' 'D101'
'EPFI' 208.
'GRST' 4
'GRNR' 4
*$ 'POWR' 10*5.87E-4
'GHET'
*$*$
*KSIOX DBN=INPUT NDCALC,TYP=CARD,PMN=PRDUM

*$ 'PUFR'

   'GEO '  1

*$ 'NOPR'

*$ 'CNTR'

'PVEC'

5
'PU238   '  19.4  238.  0
'PU239   '  19.4  239.  1
'PU240   '  19.4  240.  0
'PU241   '  19.4  241.  1
'PU242   '  19.4  242.  0

'UVEC'
4
'U 234   '  19.0   234.  0
'U 235   '  19.0   235.  1
'U 236   '  19.0   236.  0
'U 238   '  19.0   238.  0
```

```
'SVEC'
1
*$ 'ZR      '   6.5485  91.22
  'ZR      '   7.7309  91.22  *$ multiplied by 1.18056

'VOFR'

'MINP'
2
'       ' 'U 235   ' 'U 235   ' *$ DWR BENCHMARK CELL / UO2 FUEL
773. 605.8 583. 1.28264 0.040  0.000 0.000 0.411 0.064 0.89683 0.74730

  0.0000 1.0000 0. 0. 0.
0.0 0.040000 0.0 0.960000
1.
1.

'ADDM'  2
  'B  10   '  572.  4.1419E-6    *$ 500 ppm  rho_h2o=0.74730
  'B  11   '  572. 16.6719E-6    *$ 500 ppm

'FINV' 2.10603E5 1  *$ 1000 KG HEAVY METAL

'       ' 'U 235   ' 'U 235   ' *$ DWR BENCHMARK GUIDE TUBE
773. 605.8 583. 1.E+09  0.000  0.84 0.16 0.411 0.064 0.89683 0.74730

  0.0000 1.0000 0. 0. 0.
0.0 0.040000 0.0 0.960000
1.
1.

'ADDM'  2
  'B  10   '  572.  4.1419E-6    *$ 500 ppm  rho_h2o=0.74730
  'B  11   '  572. 16.6719E-6    *$ 500 ppm

'FINV' 2.10603E5 1  *$ 1000 KG HEAVY METAL

*$*$
*KSIOX DBN=INPUT GRUMIX,TYP=CARD,PMN=PRDUM,IND=1
'NDCA'
'NOPR'
'GHET'
*$*$
*KSIOX DBN=GRUCAL,TYP=CARD,PMN=PRDUM,IND=1
'GRUBA   '
*$ '/opt/KAPROS/data/G69P5E65B '
```

```
   '/opt/KAPROS/data/G28COLLIB '
*$ '/opt/KAPROS/data/G18COLLIB '
'STEUER  '
   '/opt/KAPROS/data/F69UD04 '
*$ '/opt/KAPROS/data/F69UD06 '
'GRUCAL  '
   'COLLIB  '  '      '  '      '
*$ 'WIMSLIB '  '      '  '      '
'MISCH   '
*$ 'NOPRINT '
'NOTGRMAT'
'DATBLOCK'
*$ DEFINE  EVALUATION TYPES , SN2N NECESSARY FOR COLLUP
'TYP     '
'ZUSATZ  '  5
 'SI     '
 'SN2N   '
 'STRTR  '
 'SMT01  '
 'SMT02  '
'AUSWERT '
'ZUSATZ  '  1
 *$ TYP, LABEL, 0=MAC 1=MIC, SUMMATION UEBER N, NAMEN
'XSONE   '  'O      '  1  1 'O      '

'GRUCEND '
*$*$
*KSIOX DBN=EINGABE TRANSX,IND=1,TYP=CARD,PMN=KETT
'TWODANT'
*$  IGM  ISCT  IHT   IUPS   MDCHI  NUMMAT  NTRANS  ICHI
    28     2     8    51      0     810       0      1
*$ IFILXS IFICHI  IPRINT
     22     20       0
*$ MATTAB(I),I=I,NUMMAT
    -1   -2   -3   -4   -5   -6   -7   -8   -9
   -10  -11  -12  -13  -14  -15  -16  -17  -18
   -19  -20  -21  -22  -23  -24  -25  -26  -27
   -28  -29  -30  -31  -32  -33  -34  -35  -36
   -37  -38  -39  -40  -41  -42  -43  -44  -45
   -46  -47  -48  -49  -50  -51  -52  -53  -54
……..
CUT
……
  -784  -785  -786  -787  -788  -789  -790  -791  -792
  -793  -794  -795  -796  -797  -798  -799  -800  -801
  -802  -803  -804  -805  -806  -807  -808  -809  -810
*$ IUNIT1  IUNIT2
```

```
    33     23
*$*$
*KSIOX DBN=FILLC XS TYPES E,IND=1,TYP=CARD,PMN=PRDUM
'XSONE   O    MIC'
*$*$
*KSIOX DBN=INPUT RTWODF,TYP=CARD,PMN=KETT
'DUMM'
  'COLL'
810

   1   2   3   4   5   6   7   8   9
  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  24  25  26  27
  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45
  46  47  48  49  50  51  52  53  54
  ..................................................................
CUT
.....................................................................
  784  785  786  787  788  789  790  791  792
  793  794  795  796  797  798  799  800  801
  802  803  804  805  806  807  808  809  810

'NCB0'
   'CNTR'
'EPFI' 210.
*$*$
*GO SM=ARCHIV
*GO SM=KARBUSE
```

Annex D Input file for DANTSYS

Input for the GRS benchmark proposal. Detailed model.

```
  2    0    0
 dantsys input for GRS DWR benchmark
 28   energy groups,   1/4 assembly test with 1 xs set
/
/ ** block (i) **
/
 igeom=  14
 ngroup= 28
 isn=  8
 niso=   810
 mt=   810
 nzone=   810
 im=  9
 it= 18
 jm=  9
 jt= 18
 km=  10
 kt=  40
 maxscm=  20000000
 maxlcm= 200000000
 t
/
/ ** block ii (geometry) **
/
 xmesh=
 0.000  1.272  2.544  3.816  5.088  6.360  7.632  8.904 10.176 11.448
 xints=
    2    2    2    2    2    2    2    2    2
 ymesh=
 0.000  1.272  2.544  3.816  5.088  6.360  7.632  8.904 10.176 11.448
 yints=
    2    2    2    2    2    2    2    2    2
 zmesh= 0. 39. 78. 117. 156. 195. 234. 273. 312. 351. 390.
 zints=  4  4  4   4   4   4   4   4   4   4

 zones=
   1   2   3   4   5   6   7   8   9;
  10  11  12  13  14  15  16  17  18;
  19  20  21  22  23  24  25  26  27;
  28  29  30  31  32  33  34  35  36;
  ……..
CUT
```

```
  ..............................
  775   776   777   778   779   780   781   782   783;
  784   785   786   787   788   789   790   791   792;
  793   794   795   796   797   798   799   800   801;
  802   803   804   805   806   807   808   809   810;

 t
/
/ ** block iii (cross sections) **
/
 lib=xslib
 iht=  8
 ihs= 51
 ihm=119
 maxord=  2
 ifido=  0
 ititl=  0
 i2lp1=  0
 balxs=  0
 edname= phi n-fiss scapt sn2n  nusf
 t
/
/ ** block iv (mixing) **
/
 matls=isos
 assign=matls
 t
/
/ ** block v (solver) **
/
 ievt=  1
 isct=  2
 ith=  0
 ibl=  1
 ibr=  1
 ibt=  1
 ibb=  1
 ibfrnt= 0
 ibback= 0
/ epsi=  5.000E-05
 epsi=  5.000E-04
 iitl=  1
 iitm=100
 oitm= 40
 fluxp=  0
 xsectp=  0
 fissrp=  0
```

```
 sourcp=  0
 norm=1.000
 balp=  0
 insors=  0
 chi=
 t
/
/ ** block vi (edits) **
/
/ pted=  1
prplted= 3
edoutf=  3
/ kplane= 22
igrped=  0
resdnt=  1
zned=  1
edxs=  5  6  7  8 2
power= 1.34652
 t
```

Annex E ks_cobra.dat file

1CORE DATA FOR BURNUP-STEP 1 FOR 810 ZONES.

ZONE	FUEL-RADIUS CM	CELL-RADIUS CM	ZONE-VOLUME CM**3	ZONE-POWER MEGAWATT	POWER-RATING **W/CM**	W/CM**3R	W/CM**3F	FLUX (ABS)
1	0.4110	0.7176	6.3101E+01	6.9331E-04	17.78	10.99	33.50	1.3944E+15
2	0.4110	0.7176	6.3101E+01	6.9413E-04	17.80	11.00	33.54	1.3942E+15
3	0.4110	0.7176	6.3101E+01	6.9433E-04	17.80	11.00	33.55	1.3936E+15
4	0.4110	0.7176	6.3101E+01	6.9400E-04	17.80	11.00	33.53	1.3945E+15
5	0.4110	0.7176	6.3101E+01	6.9156E-04	17.73	10.96	33.41	1.3932E+15
6	0.4110	0.7176	6.3101E+01	6.9005E-04	17.69	10.94	33.34	1.3932E+15
7	0.4110	0.7176	6.3101E+01	6.8828E-04	17.65	10.91	33.26	1.3925E+15
8	0.4110	0.7176	6.3101E+01	6.8748E-04	17.63	10.89	33.22	1.3930E+15
9	0.4110	0.7176	6.3101E+01	6.8598E-04	17.59	10.87	33.14	1.3917E+15
10	0.4110	0.7176	6.3101E+01	6.9527E-04	17.83	11.02	33.59	1.3942E+15
11	0.4110	0.7176	6.3101E+01	6.9875E-04	17.92	11.07	33.76	1.3943E+15
12	0.4110	0.7176	6.3101E+01	7.0338E-04	18.04	11.15	33.99	1.3954E+15
……..								
……..								
796	0.4110	0.7176	6.3101E+01	7.0925E-04	18.19	11.24	34.27	1.3973E+15
797	0.4110	15020.8203	6.3101E+01	1.9181E-19	0.00	0.00	0.00	1.5691E+15
798	0.4110	0.7176	6.3101E+01	7.1233E-04	18.27	11.29	34.42	1.3975E+15
799	0.4110	0.7176	6.3101E+01	7.0802E-04	18.15	11.22	34.21	1.3980E+15
800	0.4110	0.7176	6.3101E+01	6.9935E-04	17.93	11.08	33.79	1.3952E+15
801	0.4110	0.7176	6.3101E+01	6.9459E-04	17.81	11.01	33.56	1.3949E+15
802	0.4110	0.7176	6.3101E+01	6.9071E-04	17.71	10.95	33.37	1.3925E+15
803	0.4110	0.7176	6.3101E+01	6.9362E-04	17.79	10.99	33.51	1.3949E+15
804	0.4110	0.7176	6.3101E+01	6.9648E-04	17.86	11.04	33.65	1.3944E+15
805	0.4110	0.7176	6.3101E+01	7.0349E-04	18.04	11.15	33.99	1.3961E+15
806	0.4110	0.7176	6.3101E+01	7.0965E-04	18.20	11.25	34.29	1.3970E+15
807	0.4110	0.7176	6.3101E+01	7.0574E-04	18.10	11.18	34.10	1.3972E+15
808	0.4110	0.7176	6.3101E+01	7.0004E-04	17.95	11.09	33.82	1.3960E+15
809	0.4110	0.7176	6.3101E+01	6.9576E-04	17.84	11.03	33.62	1.3949E+15
810	0.4110	0.7176	6.3101E+01	6.9356E-04	17.78	10.99	33.51	1.3951E+15

TOTAL VOLUME 4.732575E+04 TOTAL POWER 5.386100E+00 MEAN POWER 1.138091E+02 WATT/CM**3 REACTOR.

==========

Annex F Relevant parts of the COBRA-TF Output deck

0 channel results date 07:11:2005 time 10:03:51 ***1." Sub-Channel Model of PWR18x18-24 Bundle ******

simulation time = 4.99688 seconds fluid properties for **channel 53**

node no.	**dist.** (ft.)	pressure (psi)	velocity (ft/sec)			void fraction			flow rate (lbm/s)		flow reg.	heat added (btu/s)		gama (lbm/s)	
			liquid	vapor	entr.	liquid	vapor	entr.	liquid	vapor	entr.			liquid	vapor
11	12.80	2287.434	19.30	19.29	0.09	1.0000	0.0000	0.0000	0.38050	0.00000	0.00000	5	0.761E+00	0.000E+00	0.00
10	11.52	2292.533	14.27	5.58	0.07	1.0000	0.0000	0.0000	0.28145	0.00000	0.00000	1	0.221E+01	0.000E+00	0.00
9	10.24	2293.477	15.85	15.86	0.08	1.0000	0.0000	0.0000	0.31628	0.00000	0.00000	1	0.344E+01	0.000E+00	0.00
8	8.96	2294.550	15.61	15.61	0.08	1.0000	0.0000	0.0000	0.31672	0.00000	0.00000	1	0.431E+01	0.000E+00	0.00
7	7.68	2295.619	15.31	15.31	0.08	1.0000	0.0000	0.0000	0.31698	0.00000	0.00000	1	0.479E+01	0.000E+00	0.00
6	6.40	2296.683	15.00	15.00	0.07	1.0000	0.0000	0.0000	0.31728	0.00000	0.00000	6	0.478E+01	0.000E+00	0.00
5	5.12	2297.739	14.73	14.73	0.07	1.0000	0.0000	0.0000	0.31802	0.00000	0.00000	1	0.429E+01	0.000E+00	0.00
4	3.84	2298.787	14.51	14.51	0.07	1.0000	0.0000	0.0000	0.31868	0.00000	0.00000	1	0.341E+01	0.000E+00	0.00
3	2.56	2299.827	14.37	14.37	0.07	1.0000	0.0000	0.0000	0.31961	0.00000	0.00000	1	0.218E+01	0.000E+00	0.00
2	1.28	2300.855	14.30	14.30	0.07	1.0000	0.0000	0.0000	0.32072	0.00000	0.00000	1	0.732E+00	0.000E+00	0.00
1	0.00	2301.785	14.35	14.33	0.00	1.0000	0.0000	0.0000	0.32227	0.00000	0.00000	1	0.000E+00	0.000E+00	0.00

node no.	dist. (ft.)	enthalpy (btu/lbm)						density (lbm/ft3)			net entrain
		vapor	hg	vapor-hg	liquid	hf	liq. - hf	mixture	liquid	vapor	mixture
11	12.80	1114.46	1114.46	0.00	651.60	702.05	-50.45	651.60	**40.70358**	6.52357	40.7034	0.000
10	11.52	1113.97	1113.97	0.00	651.45	702.67	-51.23	651.45	40.71860	6.54915	40.7185	0.000
9	10.24	1113.89	1113.89	0.00	644.45	702.79	-58.34	644.45	41.18481	6.55371	41.1848	0.000
8	8.96	1113.79	1113.79	0.00	633.59	702.91	-69.32	633.59	41.89328	6.55889	41.8932	0.000
7	7.68	1113.69	1113.69	0.00	620.00	703.04	-83.04	620.00	42.75492	6.56405	42.7549	0.000
6	6.40	1113.59	1113.59	0.00	604.91	703.16	-98.25	604.91	43.67918	6.56920	43.6791	0.000
5	5.12	1113.49	1113.49	0.00	589.88	703.29	-113.40	589.88	44.56728	6.57431	44.5672	0.000
4	3.84	1113.40	1113.40	0.00	576.42	703.41	-126.99	576.42	45.33590	6.57938	45.3359	0.000
3	2.56	1113.30	1113.30	0.00	565.75	703.53	-137.79	565.75	45.92787	6.58440	45.9278	0.000
2	1.28	1113.21	1113.21	0.00	558.96	703.65	-144.69	558.96	46.29631	6.58925	46.2963	0.000
1	0.00	1113.14	1113.14	0.00	556.69	703.74	-147.05	556.69	**46.41888**	6.59287	46.4188	0.000

……..

CUT

……..

nuclear fuel rod no. 2 **simulation time = 5.00 seconds**

surface no. 3 of 4

---------- conducts heat **to channels 4** 0 0 0 0 0 geometry type = 1

and azimuthally to surfaces 2 and 4 no. of radial nodes = 7

****

rod node no.	**axial** location (in.)	**fluid temperatures** liquid (deg-f)	vapor (deg-f)	surface heat flux (b/h-ft2)	transfer mode	**-clad temperatures-** outside (deg-f)	inside (deg-f)	gap conductance (b/h-ft2-f)	**-fuel temperatures-** surface (deg-f)	center (deg-f)
12	153.54	608.7	653.7	0.1095E+05	spl	610.45	613.25	998.5	626.20	667.17
11 *	145.87	608.7	653.7	0.5404E+05	spl	617.27	631.13	998.5	695.00	897.22
10 *	130.51	608.7	653.7	0.1257E+06	spl	630.31	662.55	998.5	811.09	1281.37
9 *	115.16	605.2	653.8	0.1955E+06	spl	637.37	687.51	998.5	918.54	1649.98
8 *	99.80	599.6	653.9	0.2450E+06	spl	640.17	703.01	998.5	992.56	1909.28
7 *	84.45	592.5	653.9	0.2722E+06	spl	637.88	707.70	998.5	1029.45	2048.11
6 *	69.09	584.3	654.0	0.2719E+06	spl	630.08	699.81	998.5	1021.13	2038.41
5 *	53.74	575.9	654.1	0.2440E+06	spl	617.48	680.07	998.5	968.43	1881.40
4 *	38.39	568.2	654.1	0.1939E+06	spl	601.68	651.43	998.5	880.63	1606.27
3 *	23.03	561.9	654.2	0.1237E+06	spl	583.60	615.33	998.5	761.53	1224.39
2 *	7.68	557.7	654.3	0.5207E+05	spl	567.07	580.43	998.5	641.97	836.79
1	0.00	557.7	654.3	0.1040E+05	spl	559.58	562.25	998.5	574.54	613.46

rod **node** no.	axial location (in.)	stored energy (btu/lbm)	--fuel-clad interface-- gap width (in.)	pressure (psig)	**----------fuel temperatures (deg-f)----------** radii in inches 0.0249	0.0557	0.0898	0.1245	0.1585
11 *	145.87	0.449E+02	0.00000	0.00	893.0	876.3	836.5	775.3	695.0
10 *	130.51	0.656E+02	0.00000	0.00	1271.6	1232.7	1140.2	997.8	811.1
9 *	115.16	0.831E+02	0.00000	0.00	1634.8	1574.3	1430.5	1209.0	918.5
8 *	99.80	0.955E+02	0.00000	0.00	1890.3	1814.4	1634.1	1356.6	992.6
7 *	84.45	0.102E+03	0.00000	0.00	2027.0	1942.7	1742.4	1434.0	1029.5
6 *	69.09	0.101E+03	0.00000	0.00	2017.4	1933.1	1733.1	1425.1	1021.1
5 *	53.74	0.936E+02	0.00000	0.00	1862.5	1786.9	1607.4	1331.0	968.4
4 *	38.39	0.800E+02	0.00000	0.00	1591.3	1531.2	1388.5	1168.8	880.6
3 *	23.03	0.617E+02	0.00000	0.00	1214.8	1176.5	1085.5	945.3	761.5
2 *	7.68	0.409E+02	0.00000	0.00	832.8	816.6	778.3	719.3	642.0

Annex G cobra_ks.dat file

CobraTF data - !!only for coupling problems with Karbuse (IRS 05) !

NB	H20temp [K]	Clad Av [K]	Fuel Av [K]	H20 dens [g/cm^3]
1	600.678	606.425	656.226	0.654885
2	604.553	610.416	660.214	0.643232
3	604.817	610.701	660.536	0.642401
4	604.803	610.666	660.458	0.642462
5	604.706	610.551	660.254	0.642751
6	604.622	610.450	660.065	0.643034
7	604.553	610.342	659.711	0.643275
8	604.469	610.251	659.583	0.643534
9	604.687	610.455	659.741	0.642865
10	600.817	606.604	656.481	0.654418
11	602.608	608.517	658.729	0.649002
12	600.692	606.609	657.200	0.654703
13	602.872	608.767	658.984	0.648211
14	604.914	610.800	660.639	0.642107
15	604.747	610.616	660.363	0.642644
16	604.650	610.491	660.144	0.642967
17	604.576	610.378	659.734	0.643174
18	604.469	610.251	659.583	0.643534

….

CUT

…..

796	564.900	570.644	621.456	0.742752
797	564.872	564.850	564.872	0.742800
798	564.886	570.653	621.799	0.742781
799	564.900	570.635	621.408	0.742756
800	564.914	570.594	620.812	0.742733
801	564.872	570.510	620.347	0.742867
802	564.891	570.575	620.232	0.742770
803	564.928	570.590	620.392	0.742730
804	564.914	570.625	620.711	0.742730
805	564.886	570.653	621.236	0.742760
806	564.872	570.679	621.727	0.742787
807	564.886	570.658	621.336	0.742765
808	564.900	570.636	620.897	0.742739
809	564.900	570.589	620.472	0.742743
810	564.844	570.508	620.298	0.742906

Annex H Benchmark specifications

The following tables are taken from [18]

Tab. 6-1 Reactor data:

Reactor data		
Reactor thermal output	MW	3850
Electric power	MW	ca. 1350
Number of fuel assemblies in core		193
Power per dm3 reactor core	kW/dm3	ca. 95,3
Average fuel power per kg U ca (18x18-24)	kW/kg	. 37,4
Mass flow rate in core	kg/s	18682
Inlet temperature (full power)	°C	292
Outlet temperature (full power)	°C	325,5

Tab. 6-2 General Data for one rod assembly:

Fuel assembly data (cold geometry, 20 °C)		
assembly-identification		18x18-24 Uran
Assembly data		
Assembly edge length incl. water gap	cm	23.000
Pitch (fuel rod distance)	cm	1.270
Fuel rod data		
Pellet diameter	cm	0.805
Number of fuel rod in an assembly		300
Active rod length	cm	390.0
Cladding-tube inner diameter	cm	0.822
Cladding-tube outer diameter	cm	0.950
Cladding-tube material		ZRY-4
Guide-tube data		
Number of guide tubes per assembly		24
Guide-tube inner diameter	cm	1.110
Guide-tube outer diameter	cm	1.232
Guide-tube material		
Fuel assembly data (hot geometry, 310 °C)		
Active rod length	cm	391.56
Assembly edge length incl. water gap	cm	23.116
Pitch (fuel rod distance)	cm	1.272
Cladding-tube inner diameter	cm	0.822
Cladding-tube outer diameter	cm	0.950
Guide-tube inner diameter	cm	1.1127
Guide-tube outer diameter	cm	1.235

Tab. 6-3 Reference values for max. power reactor condition:

	Reference values
q' (W/cm)	170.5
cB (ppm)	500
Tm (°C)	310
Tc (°C)	332.8
Tf (°C)	500
p (bar)	158

q'	specific rod power	T_c	clad-tube temperature
c_B	Bore concentration in ppm B_{nat}	T_f	fuel temperature (average)
T_m	moderator temperature	p	pressure of cooling liquid

Annex I The procedure COBRAP

The procedure COBRAP was especially created for the coupling of KAPROS-E with COBRA-TF. It is written in FORTRAN-77 like most other KAPROS procedures. The module COBRAP steers the coupling as shown in Fig. 4-13.

COBRAP manages KARBUS and the two interface modules KCNTTI and KCTTNI. COBRAP runs two different runs of KARBUS. For the first iteration a start job of KARBUS is performed with constant values for the thermal hydraulic parameters specified by input. After this first run, KARBUS is called for the other iterations in a different form taking into account the results from the thermal hydraulic calculation via the file cobra_ks.dat.

KCNTTI and KCTTNI are standard KAPROS-E module skeletons, calling the new developed FORTRAN-90 packages kcntti and kcttni respectively. The input blocs for these modules are in the “input.cobrap” file. However, the executables programs kcntti and kcttni need their own external setting files “input.kcntti” and “input kcttni” respectively (see section 4).

At contrary DANTSYS is externally coupled and the input of COBRAP contains only information for the data transfer from KARBUS to DANTSYS. COBRAP calls the program DANTSYS with the shell “DANTSH” after the KARBUS run. DANTSYS possesses its own input file “input.twodant”, which must be present in the same directory as the other files to run COBRAP.

COBRA-TF is also externally coupled and is called by COBRAP via the execution of the COBRA shell “COBRSH”. This shell calls the execution of COBRA-TF and handles the numbering of the input and output files (deck.run, deck.inp, deck.out, ks_cobra.dat, cobra_ks.dat) according to the iteration count. The input files specific to COBRA-TF mentioned in the chapter 4 must be present in the call directory to run COBRAP.

Annex J kcntti

!!!Main program

```
        use definition
        use read_calc
        use write_deck
        call set_eof      !! Same module as used for kcttni to specify the value of the Iostat argument
        call read_inputbloc
        call selfdefinition
        call set_connex
        open(15,File=cobfil,status="old",position="rewind",action="readwrite", err=140)
        call readpow
        call calc_ave
        print*
        print*,"Power calculation complete"
        write(*,"(A15,F6.2,A6)") " Average Power: ",pow_ave, "kW/m "
        print*,"writing ",cobinf
        call write_deck_inp
        close(15)
        stop
140     Print*,"Error occurs in kcntti :"
        Print*, "ks_cobra.dat could not be opened"
        end program

----------------------------------------Module endoffile-----------------------------------------------------
!       DETERMINES THE IOSTAT NUMBER FOR AN END OF FILE
!       cf the Fortran manual for the definition of the iostat parameter

        module endoffile
        integer:: eof=0 !code returned by fortran for an end of file
        integer:: lu_temp=99 !scratch file for the test
                contains
        subroutine set_eof
        open (lu_temp,status="scratch",err=99)
        read(lu_temp,"(2x)",iostat=eof)
        close(lu_temp)
99      end subroutine
        end module
----------------------------------------Module definition----------------------------------------------------
        module definition
        !describes and centralizes the main variable definitions
        use endoffile
        implicit none
        character(16)::cobfil,cobtmp,cobinf
        integer n,ax_nb,rd_nb,ch_nb,surfconnex,surf_nb,gr_nb,koderr,loop
        integer::rd_surf=4
```

```
      real pow_ave,dxs
      integer i,j,k,l
      private i,j,k,l
      character(5)::buffer,puffer
      private buffer
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      !At the beginning of the execution the matrixes could not be defined due to the fact their
      !constitutive parameters are not defined. They must be declared as "allocatable". !The instruc-
      tion "allocate" can
      !then be used do define them all in due time.
      Integer,allocatable:: connex(:,:)
      real,allocatable::rod_pw(:,:)
      real,allocatable::axial_nd(:)
      !!!!!!!!!!
      contains !
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      subroutine read_inputbloc
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      !Aim is to set the names of the files used by reading the input file
      open(7,FILE="input.kcntti",status="old",Action="read",err=140)
      read(7,*)
      read(7,105,err=145) cobfil,cobinf,cobtmp
105   Format(3(t2,A16,/))
      return
140 print*,"Error while opening the input.kcntti file "
      stop
145   print*,"Error while reading the input.kcntti file. Check its validity»
      stop
      end subroutine
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      subroutine selfdefinition
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      open(8,File=cobtmp,status="old",position="rewind",action="read")
      do while (koderr/=eof)
      read(8,113,iostat=koderr) puffer
113   format(A5)
            if (puffer(2:5)=="NGRP") then
            read(8,"(I5)",advance="no",err=125) gr_nb
            select case(gr_nb)
            case(2)
            read(8,"(I5)") ch_nb
            case(4)
            read(8,"(2(/),t11,I5,5x,F5.3)") ax_nb,dxs
            case(8)
            read(8,"(I5)") rd_nb
            end select
125         end if
```

```
        end do
        n=(sqrt(real(1+8*rd_nb))-1) /2
        close(8)
        end subroutine
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        subroutine set_connex
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        allocate (axial_nd(ax_nb+2))

        do l=2,ax_nb+1
        axial_nd(1)=0.
        axial_nd(2)=dxs/2
        axial_nd(l+1)=axial_nd(l)+dxs
        axial_nd(ax_nb+2)= dxs*ax_nb
        end do

        allocate (Connex(n,n))
        i=n; k=1

        Do while (i>=1)
                j=1
                Do while (j<=(n-i+1))
                connex(i,j)=k
                connex((n-j+1),(n-i+1))=connex(i,j)
                j=j+1;    k=k+1
                End do
        i=i-1
        End do

        allocate (rod_pw (rd_nb,ax_nb+2))
        rod_pw=0
        end subroutine
        end module
------------------------------------Module read_calc ------------------------------------------------
module read_calc
        use definition
        implicit none
        real linpow,sum
        integer::i,j,l,rd
        private l,i,j,rd
contains
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Read cordat.dat and store the values in the matrix rod_pw
        subroutine readpow
        read(15,110)
        110 format(5(/))
        l=ax_nb+1
        do while(l>=2)
```

```
                i=1
                do while(i<=n)
                        j=1
                        do while(j<=n)
                        read(15,111) linpow
111                     format(t60,F6.2)
                        rod_pw(connex(i,j),l)=linpow
                        j=j+1
                        end do
                i=i+1
                end do
        l=l-1
        end do
        end subroutine
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Calculation of the ratio power for each rod and level regarding the global average value
        subroutine calc_ave
        sum=0
        pow_ave=0

        !!!!
        rd=1
        do while (rd<=rd_nb)

                l=2
                do while (l<=ax_nb+1)
                sum=sum+rod_pw(rd,l)
                l=l+1
                end do
        rd=rd+1
        end do

        pow_ave=sum/(rd_nb*ax_nb)
!       pow_ave2=sum/810 !!!!!!!!!!!! (B. Becker)
        !!!!
        rd=1
        do while (rd<=rd_nb)

                l=2
                do while (l<=ax_nb+1)
                rod_pw(rd,l)=rod_pw(rd,l)/pow_ave
                l=l+1
                end do
        rd=rd+1
        end do

        pow_ave=pow_ave/10 !!!! unity conversion
        end subroutine
```

```
end module
-------------------------------Module write_deck-----------------------------------------------------
Module write_deck
        use endoffile
        use definition
        implicit none
        character(85)::buffer
        character(5)::gr_nb2
        integer rd,c,m,el
        private m,el
contains
        subroutine write_deck_inp  !!!!This subroutine copies the contains of struc.dat into deck.inp but
introduces changes!for the groups 1 and 11.
        call set_eof
        open(8,File=cobtmp,status="old",position="rewind",action="read")
        open(9,File=cobinf,status="new",position="rewind",form="formatted",action="write",err=200)
        koderr=0
        do while (koderr/=eof)
        read(8,113,iostat=koderr) buffer
113     format(A85)
                write(9,"(A85)") buffer

                if (buffer(2:5)=="NGRP") then
                read(8,113) buffer
                gr_nb2=adjustl(buffer(1:5))
                        select case (gr_nb2)  !!!!!!
                        case("1")           !!!!!!
                        write(9,114) 1,1
114                     format(t5,I1,t10,I1)
                        write(9,"(A)") "*      PREF        HIN        GIN      AFLUX       GHIN     VFRAC1
VFRAC2    RSBF "
                        write(9,115) "    15.8E6   2.591E6        0 ",pow_ave,"1.56E6    1.0000     .9999
1.0"
115                     format(A31,t33,F8.3,t45,A34)
                        read(8,*);read(8,*)
                        case("11")          !!!!!!
                        write(9,116) 11,rd_nb,4,0
116                     format(t4,I2,t9,I2,t15,I1,t20,I1)
                        read(8,113) buffer
                        rd=1
                        do while(rd<=rd_nb)
                        write(9,"(A85)") buffer
                        write(9,"(t4,I2,t9,I2)") rd,ax_nb+2
                        write(9,"(A85)") "*        Y     AXIAL         Y     AXIAL         Y     AXIAL         Y
AXIAL     "
```

```
                    !!!!!!! Write the power ratio for each node in the axial power tables !!!!(Group
                    11)
                    m=1
                    do while(m<=(ax_nb+2))
                            el=1

                            do while(el<=4.and.m<=(ax_nb+2))
                            write(9,117,advance="no") axial_nd(m),rod_pw(rd,m)
117                         format(5x,F5.3,5x,F5.3)
                            el=el+1
                            m=m+1
                            end do

                            write(9,*)
                    end do

                    rd=rd+1
                    end do

                    do while (c<=4)
                    read(8,"(A85)") buffer
                    c=c+1
                    end do

                    case default
                    write(9,"(A85)") buffer
                    end select
160             end if
        end do
        close(8);close(9);close(15)
        print*,"Processing kcntti complete"
        stop
200     print*
        print*,"The Output could not be created."
        print*,"Make sure it does not already exist!"
        end subroutine
        end module
```

Annex K kcttni

!Main Program

```
      !Settings from input.kcttni. Must be in the same directory
            use definition
            use read_store
            use readstore_dens
            use endoffile
            use average
            use edit_results
            use setlastcobra
            use lastcobra
            use testconvergence_new
      implicit none
      integer koderr,rd,surf
      integer ::k=0,i,j,f
      real rad1,rad2,rad3,rad4,rad5
      character (20)::motlu1
      character (17)::motlu2
      character (37)::motch1,concaten

      call set_eof            !called to set the Iostat returned value at each run and on every !oper-
      ating system, the corresponding
      ! value could differ
      call read_inputbloc
      call selfdefinition
      call set_connex

      motch1="nuclear fuel rod no.simulation time ="
      open (12,FILE=cobout,status="old",action="read",&
      position="rewind")

      do while (koderr/=eof)
            read(12,100,err=102,iostat=koderr) motlu1,rd,motlu2,t
100         format(10x,A20,I3,25x,A17,1x,F7.5)
102   concaten=(motlu1//motlu2)
                  if(motch1==concaten)  then
                        if (t==time.and.rd<=rd_nb)then
                        k=k+1
                        read(12,110,iostat=koderr) surf,surf_nb
110                     format(24x,I3,3x,I3)
                        read(12,120) surfconnex
120                     format(t68,I2,8(/))
                        Heat_cond(rd,surf)=surfconnex
                              i=no_nb+2
                              do while (i>=1)
```

```
                                    call read1(rd,surf,i)
                                    i=i-1
                                    end do
                                read(12,121,err=130) rad1,rad2,rad3,rad4,rad5
121                             format(4(/),63x,F6.4,4(4x,F6.4))
rad(1)=0;rad(2)=conv_IM(rad1);rad(3)=conv_IM(rad2);rad(4)=conv_IM(rad3)
rad(5)=conv_IM(rad4);rad(6)=conv_IM(rad5)

                                    Read(12,122)
122                                 Format(/)
                                    j=no_nb+1 ! Reinitialise the j value for the other cycles of the
loop
                                    do while(j>=2)
                                    call read2(rd,surf,j)
                                    j=j-1
                                    end do
                                end if
                        endif
        end do
130             close(12)
                open (12,FILE=cobout,status="old",action="read",&
        position="rewind")
                call read_dens

        if (k==0) then
        print*
        print*,"There is no result for the specified simulation time"
        stop
        end if
        call calc_H20_Tp_aver
        call calc_cl_tp_aver
        call calc_H20_dens_aver
        call calc_fuel_tp_aver
        call set_lastcobra
        !Test if the weighting procedure has to be achieved. This is not the case for the first !Iteration
        because
        !no old distribution exists
        if (jump_modlastcobra==0) then
        call process_lastcobrafile
        else
        H20_tpold=0
        cl_tpold=0
        H20_densold=0
        Fu_tpold=0
        weight=100
        end if
        call write_results
```

```
        call test_conv
        call summary(tdens)
        end program

---------------------------------------Module endoffile---------------------------------------------------
!       DETERMINES THE IOSTAT NUMBER FOR AN END OF FILE
!       cf the Fortran manual for the definition of the iostat parameter
module endoffile
        integer:: eof=0 !code returned by fortran for an end of file
        integer:: lu_temp=99 !scratch file for the test
        contains
        subroutine set_eof
        open (lu_temp,status="scratch",err=99)
        read(lu_temp,"(2x)",iostat=eof)
        close(lu_temp)
        99      end subroutine
end module

------------------------------------Module definition------------- ---------------------------------------
module definition
        !describes and centralizes the main variable definitions
        use endoffile
        implicit none
        character(16):: cobfil,cobout,cobtmp
        character(16):: lastcobra_ks
        real t,time
        integer n,no_nb,rd_nb,ch_nb,surfconnex,surf_nb,weight,jump_modlastcobra
        integer::rd_surf=4
        integer i,j,k,gr_nb,koderr
        private i,j,k,gr_nb,koderr
        character(5):: puffer
        Integer,allocatable:: connex(:,:)
!Definition of the matrix in mod_average
        real,allocatable ::H20_tpres(:,:)
        real,allocatable ::cl_tpres(:,:,:)
        real,allocatable ::H20_densres(:,:)
        real,allocatable ::Fu_tpres (:,:,:)
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!       Matrix from readdensity
        real,allocatable:: H20_dens(:,:)
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!       from  read_store
        real,allocatable ::Ax_loc (:)
        real,allocatable ::H20_tp (:,:,:)
        real,allocatable ::Cl_tp (:,:,:,:)
        real,allocatable ::Fu_tp (:,:,:,:)
        real,dimension (6):: rad
```

```
      integer,allocatable ::Heat_cond(:,:)
!     Matrix in mod_lastcobra
      real,allocatable ::H20_tpold(:,:)
      real,allocatable ::cl_tpold(:,:)
      real,allocatable ::H20_densold(:,:)
      real,allocatable ::Fu_tpold (:,:)
      !!!!!!!!!!
      contains !
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      subroutine read_inputbloc
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      !Aim is to set the names of the handled files by reading the input file
      open(7,FILE="input.kcttni",status="old",Action="read",err=140)
      read(7,*)
      read(7,105,err=145) cobout,cobfil,cobtmp,time,weight
105   Format(3(t2,A16,/),t2,F7.5,3/,t2,I3)
      !print*,weight;read*
      close(7)
      return
140 print*,"Error while opening the input.kcttni file "
      close(7)
      stop
145   print*,"Error while reading the input.kcttni file. Check its validity "
      close(7)
      stop
      end subroutine
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      subroutine selfdefinition
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      open(8,File=cobtmp,status="old",position="rewind",action="read")
      do while (koderr/=eof)
      read(8,113,iostat=koderr) puffer
113   format(A5)
            if (puffer(2:5)=="NGRP") then
            read(8,"(I5)",advance="no",err=125) gr_nb

            select case(gr_nb)
            case(2)
            read(8,"(I5)") ch_nb
            case(4)
            read(8,"(2(/),t11,I5)") no_nb
            case(8)
            read(8,"(I5)") rd_nb
            end select
125         end if

      end do
```

```
    n=(sqrt(real(1+8*rd_nb))-1) /2
    close(8)
    end subroutine
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    subroutine set_connex
    allocate (Connex(n,n))
    i=n; k=1
    Do while (i>=1)
        j=1
        Do while (j<=(n-i+1))
        connex(i,j)=k
        connex((n-j+1),(n-i+1))=connex(i,j)
        j=j+1;    k=k+1
        End do
    i=i-1
    End do

    allocate (Ax_loc (no_nb+2))
    allocate (H20_tp (rd_nb,rd_surf,no_nb+2))
    H20_tp=0
    allocate (Cl_tp (rd_nb,rd_surf,2,no_nb+2))
    Cl_tp=0
    allocate (Fu_tp (rd_nb,rd_surf,7,no_nb+2))
    Fu_tp=0
    allocate (Heat_cond(rd_nb,rd_surf))
    Heat_cond=0
    allocate (H20_dens(ch_nb,no_nb+1))
    H20_dens=0
    allocate (H20_tpres(rd_nb,no_nb+2))
    allocate (cl_tpres(rd_nb,2+1,no_nb+2))
    allocate (H20_densres(rd_nb,no_nb+1))
    allocate (Fu_tpres (rd_nb,7+1,no_nb+2))

    !Set the size of Matrix of mod_lastcobra
    allocate (H20_tpold(rd_nb,no_nb+2))
    allocate (cl_tpold(rd_nb,no_nb+2))
    allocate (H20_densold(rd_nb,no_nb+1))
    allocate (Fu_tpold (rd_nb,no_nb+2))
    end subroutine
end module

-----------------------------------Module conversion-----------------------------------------------------
!!      conversions  Inch/Meter and Fahrenheit/Kelvin
!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
module conversion
```

```
        contains

        real function conv_IM(inc) !inch-meter
        real,intent(inout)::inc
        conv_im=inc*0.0254
        return
        end function

        real function conv_FK(F) !fahrenheit-kelvin
        real,intent(inout)::F
        conv_fk=((F+459.67)/1.8)
        return
        end function

        real function conv_dens(lbm) !lbm/ft^3 to g/cm^3
        real,intent(inout)::lbm
        conv_dens=(lbm*0.01602)
        return
        end function
end module

---------------------------------Module read_store-----------------------------------------------------
Module read_store
        use conversion
        use definition
        implicit none

        real loc,h20tp,cladtp1,cladtp2,fueltpc,fueltps
        real tprad1,tprad2,tprad3,tprad4,tprad5
Contains
!Lecture of the first bloc of values
        Subroutine read1 (rod,surf,node)
        integer rod,surf,node
        intent(inout) rod,surf,node
        read(12,110,err=114) loc,h20tp,cladtp1,cladtp2,fueltps,fueltpc
110 format(12x,F6.2,5x,F6.1,40x,F7.2,5x,F7.2,17x,F7.2,5x,F7.2)
        ax_loc (node)=conv_IM(loc)
        H20_tp (rod,surf,node)=conv_FK(h20tp)
        Cl_tp (rod,surf,1,node)=conv_FK(cladtp1)
        Cl_tp (rod,surf,2,node)=conv_FK(cladtp2)
        Fu_tp (rod,surf,1,node)=conv_FK(fueltpc)
        Fu_tp (rod,surf,7,node)=conv_FK(fueltps)
        return
114     print*,"Error while reading the values for rod ",rod," and node ",node
        stop
        end subroutine

!Lecture of the second one for the fuel temperatures
        Subroutine read2 (rod,surf,node)
```

```
        integer rod,surf,node
        intent(inout) rod,surf,node
        read(12,123,err=124) tprad1,tprad2,tprad3,tprad4,tprad5
123 format(63x,F6.1,4(4x,F6.1))
        Fu_tp (rod,surf,2,node)=conv_FK(tprad1)
        Fu_tp (rod,surf,3,node)=conv_FK(tprad2)
        Fu_tp (rod,surf,4,node)=conv_FK(tprad3)
        Fu_tp (rod,surf,5,node)=conv_FK(tprad4)
        Fu_tp (rod,surf,6,node)=conv_FK(tprad5)
        return
124     Print*,"Error while reading fuel temperatures (second block of deck.out)"
        Print*,"rod nb. ",rod," node nb. ",node
        stop
        end subroutine
end module
-----------------------------------Module readstore_dens-----------------------------------------

! Read the density properties of Coolant
module readstore_dens
        use endoffile
        use definition
        use conversion

        implicit none
        private::motlu2,motlu1,fichier,concaten,koderr,motch1,l
        character (31)::motlu2
        character (17)::motlu1
        character (48)::fichier,concaten
        character (48)::motch1
        real:: liquidens,tdens
        integer ::l,koderr,ch
        integer ::jump,ju
contains
        subroutine read_dens
        call set_eof
        !Target word in the file:
        motch1="simulation time =fluid properties for channel»
        do while (koderr/=eof)
                read(12,100,err=98,iostat=koderr) motlu1,t,motlu2,ch
100             format(t11,A17,t34,F7.5,t61,A31,T92,I2)
98              concaten=(motlu1//motlu2)
                jump=9+(no_nb+1)
                if (concaten==motch1.and.(ABS(time-t)<=0.1).and.ch<=ch_nb) then !verify §that the
                target word and
                !the word which is read are corresponding, and the simulation time in a !certain meas-
                ure too
                tdens=t
```

```
                    ju=1
                    do while(ju<=jump)
                    read(12,*)
                    ju=ju+1
                    end do
                              l=no_nb+1
                              Do while (l>=1)
                              read(12,112,err=113) liquidens
112                           format(T97,F8.5)
                              H20_dens(ch,l)=conv_dens(liquidens)
                              l=l-1
113                           end do
                    endif
          end do
99        close (12)
          end subroutine
end module
---------------------------Module average -----------------------------------------------------

Module average! Calculation of average values
use definition
          implicit none
          private:: rd,sf,nd,k,i,j,p,sum
          real,parameter:: pi=3.1415927
          integer ::rd,sf,nd,k,i,j,p
          real ::sum
contains
                    subroutine calc_H20_Tp_aver
                    rd=1
                              do while (rd<=rd_nb)

                                        nd=1
                                        do while(nd<=no_nb+2)
                                                  sf=1;sum=0;k=0
                                                  do while(sf<=4.and.H20_tp(rd,sf,nd)/=0)
                                                  sum=sum+H20_tp(rd,sf,nd)
                                                  k=k+1
                                                  sf=sf+1
                                                  end do
                                                  H20_tpres(rd,nd)=sum/k
                                                  nd=nd+1
                                        end do
                              rd=rd+1
                              end do
                              Print*,"H20 temperature calculation complete"
                    end subroutine
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

!Average of Clad temperature over the number of surfaces and the inside and outside !surfaces

```
subroutine calc_Cl_tp_aver
rd=1
do while (rd<=rd_nb)
        nd=1
        do while(nd<=no_nb+2)
                i=1
                do while(i<=2)
                        sf=1;sum=0;k=0
                        do while(sf<=4.and.Cl_tp(rd,sf,i,nd)/=0)
                        sum=sum+Cl_tp(rd,sf,i,nd)
                        k=k+1
                        sf=sf+1
                        end do
                        Cl_tpres(rd,i,nd)=sum/k
                i=i+1
                end do
        Cl_tpres(rd,3,nd)=(Cl_tpres(rd,1,nd)+Cl_tpres(rd,2,nd))/2
        nd=nd+1
        end do
rd=rd+1
end do
Print*,"Clad temperature calculation complete"
end subroutine
```

!!!

!Calculation of the average density of H20 aroud a rod

```
subroutine calc_H20_dens_aver
rd=1
do while (rd<=rd_nb)
        nd=1
        do while(nd<=no_nb+1)
                        sf=1;sum=0;k=0
                        do while(sf<=4.and.Heat_cond(rd,sf)/=0)
                        sum=sum+H20_dens(Heat_cond(rd,sf),nd)
                        k=k+1
                        sf=sf+1
                        end do
        H20_densres(rd,nd)=sum/k
        nd=nd+1
        end do
j=no_nb
do while (j>1)
H20_densres(rd,j)=(H20_densres(rd,j)+H20_densres(rd,j-1))/2
j=j-1
enddo
```

```
                    H20_densres(rd,1)=0
                    rd=rd+1
                    end do
                    Print*,"H20 density calculation complete"
                    end subroutine
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Calculation of average fuel temp
                    subroutine calc_fuel_tp_aver
                    rd=1
                    do while (rd<=rd_nb)
                              nd=1
                              do while(nd<=no_nb+2)
                                   i=1
                                   do while(i<=7)
                                        sf=1;sum=0;k=0
                                        do while(sf<=4.and.Fu_tp (rd,sf,i,nd)/=0)
                                        sum=sum+Fu_tp(rd,sf,i,nd)
                                        k=k+1
                                        sf=sf+1
                                        end do
                                        Fu_tpres(rd,i,nd)=sum/k
                                   i=i+1
                                   end do
                              p=1;sum=0
                                   do While(p<=5)
                                   sum=sum+(Fu_tpres(rd,p+1,nd))*Pi*(rad(p+1)**2-rad(p)**2)
                                   p=p+1
                                   end do

                              Fu_tpres(rd,8,nd)=sum/(Pi*rad(6)**2)
                              nd=nd+1
                              end do
                    rd=rd+1
                    end do
                    Write(*,"(A,3(/))") " Fuel temperature calculation complete"
                    end subroutine
end module

------------------------------------Module Edit_results ---------------------------------------------
! Write the calculated values in the output File
          module Edit_results
          use definition
          use read_store
          use weighting
          integer ::l,i,k,NB=1
          private i,k,l,NB
```

```
      contains
      subroutine write_results
      write (*,"(/,A)") "Creating the file and writing the results. Please wait..."
      open (13,FILE=cobfil,status="new",action="write",position="rewind",err=135)
      Write(13,"(A)") "CobraTF data - !!only for coupling problems with KARBUS (IRS 05) !"
      Write(13,131) "NB","H20temp","Clad Av","Fuel Av","H20 dens"
131 Format(4(A8,8x),A8)
      Write(13,131) "","[K]","[K]","[K]","[g/cm^3]"
      Write(13,"(A)") "-------------------------------------------------------------------------"
      l=no_nb+1
      Do while (l>=2)
            i=1
            Do while(i<=n)
                  j=1
                  Do while(j<=n)
                  !print*,H20_tpres(connex(i,j),l),H20_tpold(connex(i,j),l);read*
                  Write(13,132) NB,mix(H20_tpres(connex(i,j),l),H20_tpold(connex(i,j),l)), &
                  mix(Cl_tpres(connex(i,j),3,l),Cl_tpold(connex(i,j),l)), &
                  mix(Fu_tpres(connex(i,j),8,l),Fu_tpold(connex(i,j),l)), &
                  mix(H20_densres(connex(i,j),l),H20_densold(connex(i,j),l))
132               format(I8,3(8x,F8.3),8x,F8.6)
                  NB=NB+1
                  j=j+1
                  end do
            i=i+1
            end do
      l=l-1
      end do

      write (*,"(/,A,/)") "Processing kcttni complete"
      close(13)
      return
135 write (*,"(/,A,/,/)") "ERROR OCCURS while creating the output file. Make sure it does not already
exist"
      close (13)
      stop
140   end subroutine
      subroutine summary(tdens)
      intent(in) tdens
      write(*,"(/,A,/)") "Summary ----------------------------------------------------------------"
      write(*,"(A11,A16,2x,A10,A16,/)",advance="no") "Read from: ",cobout,"Write in: ",cobfil
      write(*,"(A19,A16,/)",advance="no") "Geometry set from: ",cobtmp
      write(*,"(/,A20,I2,A3,I2)") "Assembly dimension: ",2*n," x ",2*n
      write(*,"(A21,I2)") "Axial levels number: ",no_nb
      write(*,"(A16,I2)") "Channel number: ",ch_nb
      write(*,"(A12,I2)") "Rod number: ",rd_nb
```

```
        write(*,"(A53,F7.5,A8)") "The density values are read for a simulation time of ",tdens," seconds"
        write(*,"(/,A,A)") "Relaxation calculated from new datas and ",lastcobra_ks
        write(*,"(/,A,/)") "--------------------------------------------------------------------------"
        end subroutine
end module

------------------------------Module setlastcobra ------------------------------------------------------
module setlastcobra

        use definition
        integer  step,koderr2
contains
        subroutine set_lastcobra
        step=1

        do while (koderr2==0)
        !print*,iostat,koderr2
        lastcobra_ks(1:14)="cobra_ks.dat_0";write(lastcobra_ks(15:16),"(I2.2)") step
        !print*,lastcobra_ks;read*
        open(14,FILE=lastcobra_ks,status="old",action="read",position="rewind",iostat=koderr2,err=45)
        close(14)
        step=step+1
45      end do

        step=step-1
        jump_modlastcobra=0
        ! jump_modlastcobra will be set to 1 if there is no precedent !results for cobfil. E.g the module last_cobra
        !has to be jumped
        If (step>=1) then
        write(lastcobra_ks(15:16),"(I2.2)") step
        else
        jump_modlastcobra=1
        end if
        end subroutine
end module
-------------------------Module lastcobra ------------------------------------------------------------

module lastcobra
        use definition
        private i,j,l
contains
        subroutine process_lastcobrafile
        write (*,"(/,A)") "Processing the last Output file from CobraTF."
        open (14,FILE=lastcobra_ks,status="old",action="read",position="rewind
        read (14,"(3/)")
```

```
        l=no_nb+1
        Do while (l>=2)
                i=1
                Do while(i<=n)
                        j=1
                        Do while(j<=n)
                        Read(14,132,err=133) H20_tpold(connex(i,j),l),Cl_tpold(connex(i,j),l),&
                        Fu_tpold(connex(i,j),l),H20_densold(connex(i,j),l)
132                     format(t9,3(8x,F8.3),8x,F8.6)
                        j=j+1
                        end do
                i=i+1
                end do
        l=l-1
        end do
133     close (14)
        return

135 write (*,"(/,A,/)") "Error while processing the precedent Cobra output file."
        close (14)
        stop
        end subroutine
end module

----------Module weighting----------
!!      Define the function "mix" for weighting the part of the old distribution in the new one
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
module weighting
        use definition
contains
        real function mix(new,old)
        real,intent(inout)::new,old
        mix=(weight*new+old*(100-weight))/100
        return
        end function
        end module

----------Module test_convergence----------
module testconvergence_new
        use definition
        real::conv=0.006 !"convergence" criterion for the variation rates
contains
        subroutine test_conv
        character(80)::buffer
        integer u
        real h20tp_var,Cltp_var,Futp_var,h20dens_var
        real max1,max2,max3,max4
```

```
!Summation
l=no_nb+1
max1=0;max2=0;max3=0;max4=0
Do while (l>=2)
        i=1
        Do while(i<=n)
                j=1
                Do while(j<=n)
                !Calculation of the variation rate for each pair of value (new/old)
                h20tp_var=abs(H20_tpres(connex(i,j),l)-
H20_tpold(connex(i,j),l))/H20_tpres(connex(i,j),l)
                Cltp_var=abs(cl_tpres(connex(i,j),3,l)-
cl_tpold(connex(i,j),l))/cl_tpres(connex(i,j),3,l)
                Futp_var=abs(Fu_tpres(connex(i,j),8,l)-
Fu_tpold(connex(i,j),l))/Fu_tpres(connex(i,j),8,l)
                h20dens_var=abs(H20_densres(connex(i,j),l)-
H20_densold(connex(i,j),l))/H20_densres(connex(i,j),l)

                !Find for each category of result the maximum variation rate. These !will be
                used for comparison with the Convergence Criterion
                If (h20tp_var>max1) then
                max1=h20tp_var
                end if

                If (Cltp_var>max2) then
                max2=Cltp_var
                end if

                If (Futp_var>max3) then
                max3=Futp_var
                end if

                If (h20dens_var>max4) then
                max4=h20dens_var
                end if

                j=j+1
                end do
        i=i+1
        end do
l=l-1
end do
If (max1<=conv.and.max2<=conv.and.max3<=conv.and.max4<=conv) then
Write(*,"(/,A,//)")"Convergence criterion fulfilled"
Indic_conv=1
else

Write(*,"(/,A,//)") "A new Iteration must be performed to fulfil the !convergence criterion"
Indic_conv=0
```

```
        end if
        !Update the input file of kcntti for the convergence criterion at each iteration. !This !value will
        be
        !read by cobrap to determine the end of the processing.
        open(19,FILE="input.kcttni",status="old",Action="readwrite",position="Rewind")
        u=1

        do while (u<=8)
        Read(19,*) !;Read(19,*) buffer;Read(19,*) buffer;Read(19,*) buffer;Read(19,*) buffer
        u=u+1
        end do

        write(19,200) """,Indic_conv,"'      Convergence criterion fulfilled?(0 or 1). Written here by
        kcttni.0 must be the begin value"
200     Format(A1,I1,A)
!       Write(19,201) "****","****"
201     Format(A4,/,A4)
        write(19,"(A70,4(1x,F6.4))") "Maximum variation rate for each category of result at this Itera-
tion: ",max1,max2,max3,max4
        write(*,*) "Maximum variation rate for each category of result at this Iteration:
",max1,max2,max3,max4
        close(19)
        end subroutine
end module
```