

**COMPUTATIONAL FLUID DYNAMICS APPLICATIONS IN
NUCLEAR REACTOR THERMAL HYDRAULICS**

**NÜKLEER REAKTÖR ISIL HİDROLİK ANALİZLERİNDE
HESAPLAMALI AKIŞKANLAR DİNAMİĞİ UYGULAMALARI**

Bora Kalpaklı

A THESIS OF MASTER OF SCIENCE

Prepared in the NUCLEAR ENGINEERING SECTION

According To Regulation of

The Institute for Graduate Studies in Pure

And Applied Sciences of Hacettepe University.

2006

To the institute for Graduate Studies in Pure and Applied Sciences,

This work has been approved by the Graduate Committee as partial fulfilment of the requirements for the degree of **MASTER OF SCIENCE** in the field of **NUCLEAR ENGINEERING**.

Graduate Committee

Head _____

Assoc. Prof. Dr. Cemal Niyazi Sökmen

Member _____

Dr. Ayhan Yilmazer

Member _____

Assistant Prof. S. Sinan Keskin

APPROVED

This thesis has been approved on / / 2006 by the Graduate Committee
established by the board of the institute

/ / 2006

Prof. Dr. Ahmet R.ÖZDURAL
HEAD OF THE INSTITUTE FOR GRADUATE
STUDIES IN PURE AND APPLIED SCIENCES

COMPUTATIONAL FLUID DYNAMICS APPLICATIONS IN NUCLEAR THERMAL HYDRAULICS

Bora Kalpaklı

ABSTRACT

New generation nuclear reactors are different in many ways from traditional designs. Thermal-hydraulics analysis of these reactors requires more powerful, dependable, and detailed multi dimensional computational tools. Computational Fluid Dynamics (CFD) has great potential to become an essential tool in flow and thermal analyses of nuclear reactors.

In this thesis, two possible applications of CFD in nuclear reactor thermal hydraulics are considered. In the first part of the thesis, thermal hydraulics of pebble-bed high temperature gas cooled reactors is considered and a porous medium approach is developed to be used with the FLUENT CFD code. The model is assessed for PBMR-400 reactor concept. In the second part, Multiphase Computational Fluid Dynamics (MCFD) is considered and for this purpose a new three dimensional two-fluid model with Interfacial Area Concentration Transport Equations is developed for the FLUENT CFD code.

Keywords: Pebble-bed reactor, CFD, MCFD, IATE, multiphase flow, porous medium.

Advisor: Assoc. Prof. C. Niyazi Sökmen, Hacettepe University, Nuclear Engineering Department, Nuclear Engineering Section

NÜKLEER REAKTOR ISIL HİDROLİĞİNDE HESAPLAMALI AKIŞKANLAR DİNAMİĞİ UYGULAMALARI

Bora Kalpaklı

ÖZ

Yeni nesil nükleer reaktörler geleneksel reaktörlere göre birçok farklılığa sahiptirler. Bu tip reaktörlerin ısı- hidrolik analizinde daha güçlü, güvenilir ve detaylı çok boyutlu hesaplamalı yöntemlere ihtiyaç duyulmaktadır. Hesaplamalı Akışkanlar Mekaniği (HAD) nükleer reaktörlerin ısı- hidrolik analizinde temel araçlardan biri olma konusunda yüksek potansiyele sahiptir.

Bu tez çalışmasında, nükleer reaktör ısı- hidroliğinde iki muhtemel HAD uygulaması üzerinde duruldu. Çakıl yataklı, yüksek sıcaklık, gaz soğutmalı reaktörlerin ısı- hidroliği tezin ilk bölümünde çalışıldı ve bu amaçla FLUENT yazılımı içinde yeni bir gözenekli ortam modellemesi geliştirildi. Bu modelin uygulaması PBMR-400 reaktörü üzerinde yapıldı. Tezin ikinci bölümünde, çok fazlı hesaplamalı akışkanlar dinamiği üzerinde duruldu ve bu amaçla yine FLUENT yazılımı içinde yüzey alanı taşınım denklemlerini içeren üç boyutlu iki-sıvı modeli geliştirildi.

Anahtar Sözcükler: Çakıl yataklı reaktörler, CFD, MCFD, IATE, çok fazlı akışlar, gözenekli ortam.

Danışman: Doç. Dr. C. Niyazi Sökmen, Hacettepe Üniversitesi, Nükleer Enerji Mühendisliği Bölümü, Nükleer Enerji Mühendisliği Anabilim Dalı

ACKNOWLEDGEMENTS

I would like to thank my advisor Assoc. Prof. C. Niyazi Sökmen for his helpful cooperation and guidance. I would also thank all the members of Hacettepe University Nuclear Engineering Department.

Special thanks are due to my family for their support and encouragement.

Bora Kalpakli

Hacettepe University

August, 2006

TABLE OF CONTENTS

ABSTRACT.....	iii
ÖZ	iv
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES	ix
LIST OF TABLES.....	x
CHAPTER 1: INTRODUCTION.....	1
1.1 COMPUTATIONAL FLUID DYNAMICS (CFD)	1
1.2 FINITE VOLUME METHOD.....	2
1.3 FLUENT CFD SOFTWARE	4
CHAPTER 2: CFD APPLICATION FOR PEBBLE-BED NUCLEAR REACTOR	
THERMAL HYDRAULICS	6
2.1 INTRODUCTION.....	6
2.1.1 Pebble-bed modular reactor (Pbmr-400)[4].....	7
2.2 MODEL DESCRIPTION.....	10
2.2.1 Heat Conduction Equation for Pebble-bed.....	10
2.2.2 Energy Equation for Coolant in Pebble-bed.....	11
2.2.3 The Mass Conservation Equation	11
2.2.4 Momentum Conservation Equation.....	12
2.2.5 Turbulence Modeling In Pebble-Bed.....	12
2.2.6 Numerics of Heat Transfer at Core-Reflector Interfaces.....	12
2.3 ASSESMENT OF THE MODEL IN FLUENT	14
2.3.1 Heat Conduction Equation for Pebble-bed.....	15
2.3.2 Modeling Source Terms	16
2.3.3 Effects of Void Fraction ε	18
2.4 MODEL APPLICATION	19
2.4.1 Simplifications Introduced.....	19
2.4.2 Material Properties.....	23
2.4.3 Power Profile	26

2.4.4 Reactor Main Coolant Specification	26
2.4.5 Turbulence Modeling	26
2.4.5 Radiation Heat Transfer Modeling [18].....	27
2.5 DISCUSSION AND THERMAL-HYDRAULICS RESULTS	28
2.5.1 Grid Structure	28
2.5.2 Grid Sensitivity	29
2.5.3 Convergence and Analysis of Global Balances	30
2.5.3 Thermal-hydraulics Results	31
Chapter 3: MODELING INTERFACIAL AREA TRANSPORT EQATION WITH 3D TWO-FLUID COMPUTATIONAL MULTI-PHASE FLUID DYNAMICS MODEL	42
3 . 1 I N T R O D U C T I O N	42
3.2 POPULATION BALANCE MODELS.....	42
3.3 ITERFACIAL AREA CONCENTRATION TRANSPORT EQAUTION	45
3.4 MAJOR BUBBLE INTERACTION MECHANISMS.....	48
3.4.1 Coalescence due to Turbulence Induced Random Collision	50
3.4.2 Coalescence Through Entrainment in the Wake Region of a Preceding Bubble.....	50
3.4.3 Bubble Break-up due to Turbulent Impact.....	51
3.5 CONSERVATION EQAUTATIONS FOR A TWO-FLUID (EULER-EULER) MODEL WITH INTERFACIAL AREA TRANSPORT MODELING.....	51
3.5.1 Volume Fractions.....	52
3.5.2 Conservation of Mass	52
3.5.3 Conservation of Momentum	52
3.5.4 Conservation of Energy	55
3.5.5 Interfacial Area Concentration Transport Equation	55
3.6 IMPLEMENTAION OF THE MODEL IN FLUENT CODE.....	55
3.6.1 Case Study	56
3.6.2 Simulation Results	56
CHAPTER 4: CONCLUSIONS	62
Bibliograpy	64
APPENDIX 1	1

APPENDIX 2.....	16
INDEX	28
Resume.....	29

LIST OF FIGURES

Figure 1: Organizational structure of the components available in FLUENT [18].	5
Figure 2 : PBMR fuel design.	8
Figure 3: PBMR-400 Core configuration.	10
Figure 4 : Vector and Gradient Definitions	13
Figure 5: Effective thermal conductivity in a Pebble Bed	15
Figure 6: PBMR 2-D Geometry Layout [5].	20
Figure 7: Thermal conductivity of A3-3 matrix graphite as a function of temperature and neutron fluence irradiated at 950 °C. [5].	24
Figure 8: ATR-2E graphite thermal conductivity as a function of temperature and dose. [5].	24
Figure 9: Model grid structure	29
Figure 10 Contours of Temperatures in K.	32
Figure 11 Contours of fuel surface temperatures.	33
Figure 12: Temperature distribution of solid materials.	34
Figure 13: Coolant temperature in core zone.	35
Figure 14 Helium coolant temperatures at different core heights.	36
Figure 15 Fuel surface temperatures.	37
Figure 16 Temperature difference between fuel surface and helium coolant.	38
Figure 17: Effective pebble-bed conductivity (Zehner-Schlunder).	39
Figure 18: Effective pebble-bed conductivity (W/cm/K)	40
Figure 19: Helium velocities at various core heights.	41
Figure 20: Mechanisms of bubble coalescence and break-up [12].	49
Figure 21: Radial volume fraction profiles of air bubbles at different heights.	57
Figure 22: Interfacial area concentration profiles of air bubbles at different heights.	58
Figure 23: Diameter profiles of air bubbles at different heights.	59
Figure 24: Volume Fractions of bubbles at 2.5 m height.	60
Figure 25: Volume Fractions of bubbles at 3.5 m height.	60
Figure 26: Interfacial area concentrations at 2.5m height.	61
Figure 27: Interfacial area concentrations at 3.5m height.	61

LIST OF TABLES

Table 1 : Plant Target Specifications	7
Table 2: Core geometrical specifications.....	20
Table 3 In-Core flow channel geometrical specifications	22
Table 4 Approximate Effective Porosities in Reflector Regions due to gaps and slits..	23
Table 5 Emissivity of materials	25
Table 6 Thermal conductivities of materials	25
Table 7 Main helium flow parameters.....	26
Table 8: Global parameters	32

CHAPTER 1: INTRODUCTION

1.1 COMPUTATIONAL FLUID DYNAMICS (CFD)

CFD is commonly accepted as the numerical solution of the equations governing the fluid flow, such as the set of Navier-Stokes equations, continuity and any additional conservation equations, by computational methods. Computational Fluid Dynamics has received extensive attention since the advent of the digital computer.

CFD is useful in two ways. Firstly, it can be used to model physical fluid phenomena that can not be easily simulated or measured with a physical experiment. Secondly, it can be used to investigate physical fluid systems more cost effectively and more rapidly than with experimental procedures.

The development of modern computational fluid dynamics began with the advent of the digital computer in the early 1950s. Finite difference methods (FDM) and finite element methods (FEM) are the basic tools used in CFD and have different origins. In 1910, at the Royal Society of London, Richardson presented a paper on the first FDM solution for stress analysis of a masonry dam. In contrast, the first FEM work was published in the Aeronautical science Journal by Turner, Clough, Martin and Topp for applications to aircraft stress analysis in 1955. Since then, both methods have been developed extensively in fluid dynamics, heat transfer, and related areas [6].

Finite volume methods, having formulations related to both FDM and FEM, have become increasingly popular in recent years because of their simple data structure. Finite volume methods are a class of discretization schemes that have proven highly successful in approximating the solution of a wide range of conservation equations. They are extensively used in fluid mechanics, meteorology, electromagnetics, semi-conductor simulation, models of biological processes and many other engineering areas governed by conservative systems that can be written in integral control volume form [6].

1.2 FINITE VOLUME METHOD

Finite volume methods (FVM), often called control volume methods, are discretization methods which are well suited for the numerical simulation of various types (elliptic, parabolic or hyperbolic) of conservation laws; it has been extensively used in several engineering fields, such as fluid mechanics, heat and mass transfer. FVM are formulated by spatial integration of the governing equations. The integrated terms are approximated by either finite differences or finite elements, discretely summed over the entire domain. One of the most important of FVM is their flexibility for unstructured grids.

For illustration, the following steady state conservation equation is considered

$$\nabla \cdot (\rho \phi \vec{u}) = \nabla \cdot (\Gamma_\phi \nabla \phi) + S_\phi \quad (1)$$

The finite volume equations are obtained as integrating the equation over a control volume V

$$\int_V \nabla \cdot (\rho \phi \vec{u}) dV = \int_V \nabla \cdot (\Gamma_\phi \nabla \phi) dV + \int_V S_\phi dV \quad (2)$$

Using the divergence theorem Equation (2) becomes

$$\oint (\rho \phi \vec{u}) \cdot d\vec{A} = \oint (\Gamma_\phi \nabla \phi) \cdot d\vec{A} + \int_V S_\phi dV \quad (3)$$

In Equation (3), $\vec{A}$ is surface area vector and S_ϕ is source of ϕ per unit volume. Equation (3) is applied to each control volume, or cell, in the computational domain. Discretization of Equation (3) on a given cell yields

$$\sum_f^{N_{faces}} \rho_f \vec{u}_f \phi_f \cdot \vec{A}_f = \sum_f^{N_{faces}} \Gamma_\phi \nabla \phi \cdot \vec{A}_f + S_\phi V \quad (4)$$

where

N_{faces} = number of faces enclosing the cell

ϕ_f = value of ϕ at the face f

$\vec{A}_f$ = face normal vector where $|\vec{A}_f|$ is equal to area of face f

V = cell volume.

The face values ϕ_f are required for calculating the convection terms in Equation (4) and must be interpolated from the cell center values. This is accomplished using an upwind scheme. Two of the upwind schemes used in this thesis are first-order upwind scheme and the second order upwind scheme.

With the first order upwind scheme only first order accuracy is obtained, quantities at cell faces are determined by assuming that the cell center values of any field variable represents a cell average value and hold through the entire cell, then it is assumed that the face values are identical to cell values. When first order upwind scheme is used, the face value ϕ_f is set equal to the cell center value of ϕ in the upstream cell.

In the second order upwind scheme, higher order accuracy is achieved at cell faces through a Taylor series expansion of the cell-centered solution about the cell centroid. The face value ϕ_f is computed using the Equation(5).

$$\phi_f = \phi + \nabla \phi \cdot \Delta \vec{s} \quad (5)$$

In Equation(5), ϕ and $\nabla \phi$ are the cell-centered value and its gradient in the upstream cell, and $\Delta \vec{s}$ is the displacement vector from the upstream cell centroid to the face centroid. This formulation requires the determination of the gradient $\nabla \phi$ in each cell. This gradient is computed using the divergence theorem, which in discrete form is written as in Equation(6).

$$\nabla \phi = \frac{1}{V} \sum_f^{N_{faces}} \tilde{\phi}_f \vec{A} \quad (6)$$

In Equation(6), the face values $\tilde{\phi}_f$ are computed by averaging ϕ from the two cells adjacent to the face.

The first-order discretization generally yields better convergence than the second-order scheme. However it generally will yield less accurate results, especially on triangular and tetrahedral grids. Especially for complex flows, second-order scheme provides more accurate and better results regardless of the grid structure.

The basic idea of FVM is to obtain system algebraic equations for the discretized control volume and control surface written such as in Equation(4). In this process, the conservation of all variables enforced across the control surfaces. As a result there is no artificial creation or destruction of conserved variable [6].

The finite volume methods are cost effective, because the calculation of flows at the surface of the adjoining control volumes need be performed only once since the expression is the same for both control volumes, differing only in sign. This provides both cost reduction and algorithmic simplicity.

1.3 FLUENT CFD SOFTWARE

FLUENT is a state-of-the-art computer program for modeling fluid flow and heat transfer in complex geometries. FLUENT provides complete mesh flexibility, including the ability to solve your flow problems using unstructured meshes that can be generated about complex geometries with relative ease. Supported mesh types include 2D triangular/quadrilateral, 3D tetrahedral/hexahedral/pyramid/wedge, and mixed (hybrid) meshes. FLUENT also allows you to refine or coarsen your grid based on the flow solution. FLUENT is a finite volume CFD code based on the methods explained in section 1.1.1 [18].

The standard FLUENT package includes the following components:

- FLUENT, the solver.
- GAMBIT, the preprocessor for geometry modeling and mesh generation.

- TGrid, an additional preprocessor that can generate volume meshes from existing boundary meshes.
- Filters (translators) for import of surface and volume meshes from CAD/CAE packages such as ANSYS, CGNS, I-deas, NASTRAN, PATRAN, and others.

The organizational structure of these components is shown in Figure 1.

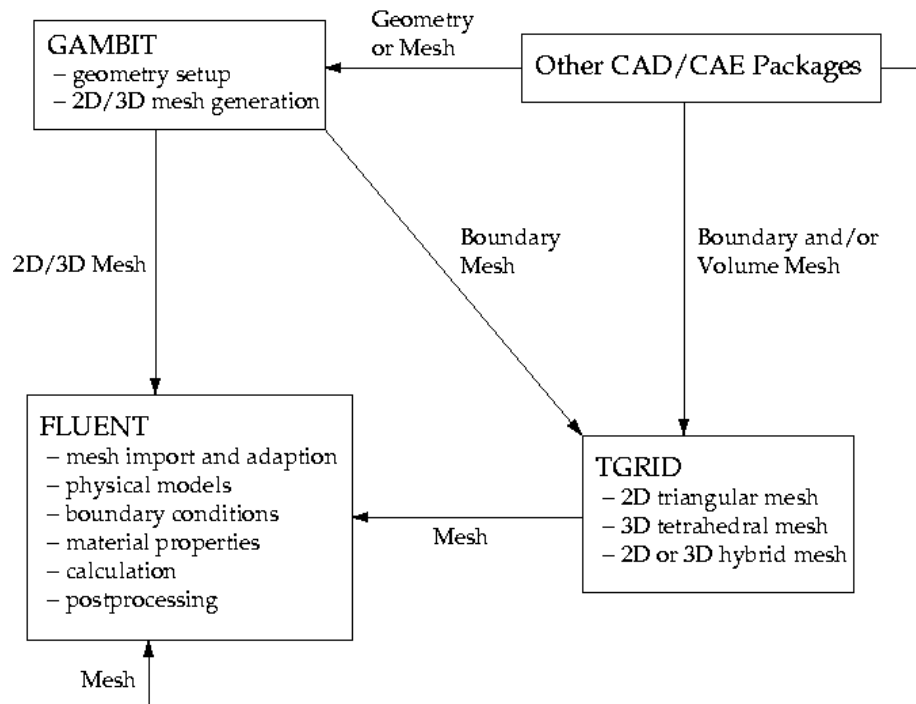


Figure 1: Organizational structure of the components available in FLUENT [18].

CHAPTER 2: CFD APPLICATION FOR PEBBLE-BED NUCLEAR REACTOR THERMAL HYDRAULICS

2.1 INTRODUCTION

In a pebble-bed reactor, the reactor core is consisted of randomly placed fuel balls, helium coolant and graphite reflectors. Due to this geometrical complexity and complicated mesh structure, modeling the exact core geometry is not a feasible approach. Such a model requires too much computational power.

Another complexity is the number of heat transfer mechanisms in the core. There are three heat transfer modes in a pebble-bed core: conduction, convection and radiation. Heat conduction occurs where the fuel balls touch other neighboring balls and reflector surface. At the operating temperatures, heat transfer by conduction is much lower than radiation heat transfer between the fuel surfaces. Radiation heat transfer also occurs between fuel ball surfaces and reflector walls. Helium coolant cools the core and reflector blocks by convection.

Although FLUENT has very flexible geometry capabilities and physical models, it is not possible to overcome complexities and capture the physical properties of the pebble bed without a new modeling approach. This new model includes a new porous zone definition different from the one in FLUENT. The difference between the porous medium model in FLUENT and the new model is the capability of handling all the heat transfer modes explained above. FLUENT porous model includes only extra momentum sources in flow equations, however one needs more new boundary conditions, energy source definitions and one more scalar equation to take account the heat transfer between fuel balls, helium coolant and reflector blocks. A very important drawback of the FLUENT porous media model is the assumption of thermal equilibrium between the porous medium and the fluid.

Significant modeling improvement is expected in the definition of porous model for pebble-bed thermal-hydraulics by this new model.

2.1.1 Pebble-bed modular reactor (Pbmr-400)[4]

PBMR is helium cooled, graphite moderated, and modular high temperature reactor. PBMR design features a direct coupling of the high temperature reactor system to a gas turbine. Very high level of passive safety without active engineered systems is achieved. The direct cycle gas turbine eliminates the complex and costly steam cycle for power conversion. This design results in a significantly higher thermal efficiency. Passive safety is accomplished by the use of high integrity particle fuel that has a strong negative temperature coefficient and large thermal capacity. The fuel retains its integrity under high temperature accident conditions and has good resistance to chemical attack, e.g. from water or air ingress. The thin annular core design allows the fuel decay heat to be conducted through reactor structures and graphite reflectors to vessel cavity and then to atmosphere. This provides the reactor with a high degree of inherent safety that is essentially meltdown proof. In addition to high level of safety and competitive economic objectives, PBMR-400 project has established a number of target specifications;

Table 1 : Plant Target Specifications

Rated output per module	165 MWe
Continuous power range	40-100%
Construction schedule (nominal)	24 months
Planned outage duration and frequency	30 days every 6 years
Plant availability	95%
Seismic (peak ground acceleration)	0.4 g
Emergency planning zone	<400 meters

The main safety feature of the PBMR-400 is due to its high quality fuel. Previous tests, as part of the German HTR program, have shown that as long as the maximum fuel temperature remains below about 1600°C, fission products will be contained within the fuel and its TRISO coatings. TRISO fuel consists of a small UO₂ kernel with a porous carbon buffer layer, a dense pyrolytic carbon layer, a very hard silicon carbide layer, and another dense pyrolytic carbon layer. The resulting coated particles are mixed in a

graphite matrix, then pressed and sintered into a sphere of six centimeters diameter. This fuel design is shown in Figure 2.

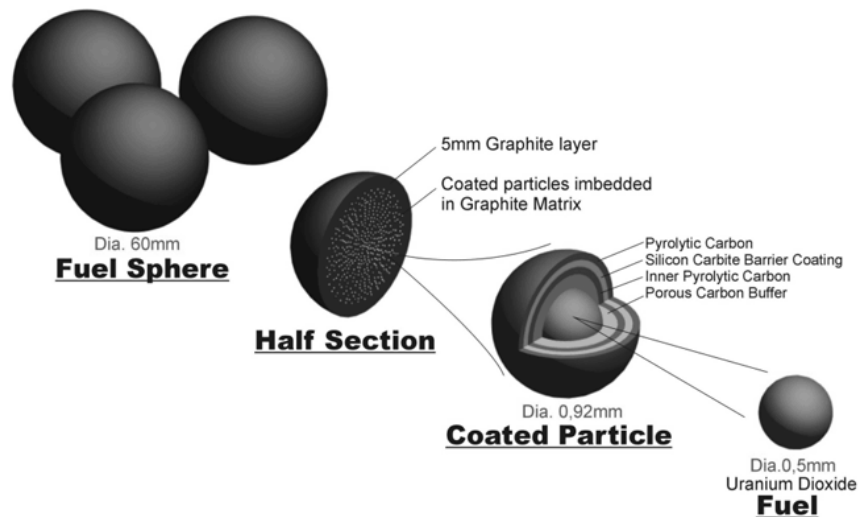


Figure 2 : PBMR fuel design.

The PBMR is based on a relatively simple design with passive safety features in contrast to existing commercial reactors that use active engineered safety systems. The passive safety feature of PBMR is the result of its special design, the materials used and the fuel used. The key safety features of PBMR reactor design includes; the radionuclide retention capability of the fuel particle coatings, small operational excess reactivity, a large negative temperature coefficient, and a passive heat removal capability of the reactor design. In the PBMR, the removal of decay heat is achieved by conduction, convection, and radiation heat transfer from the fuel particles through the reactor to the environment. The high surface area to volume ratio of the core promotes this heat transfer process and ensures that the core never reaches a temperature at which significant degradation of the fuel can occur. A steel pressure vessel is utilized instead of the concrete pressure vessel of prior HTRs. Decay heat from the core during accident conditions can now be passively rejected to the atmosphere via conduction through the vessel walls to the Reactor Cavity Cooling System.

The thermodynamic cycle of the PBMR is a closed Brayton cycle with recuperation. Helium exits the reactor at 900°C and passes successively through the high and low pressure turbines that drive the compressors and then to the power generation turbine which is coupled to the main electrical generator. The exhaust helium out of the power generation turbine passes through the recuperator transferring its heat to the helium returning to the reactor. The helium coolant then passes through two coolers and the compressors, increasing its pressure to 9 MPa before re-entering the recuperator for the second time prior to returning to the reactor at 482°C. The overall efficiency of this cycle is calculated to be approximately 42%.

The PBMR core has a annular geometry with an inner radius of 1.0 m and an outer radius of 1.85 m. The core consists of approximately 452000 fuel spheres with an active height of 11.0 m. This geometry results in a relatively low power density of 4.8 MW/m³.

The pebbles gradually move downward in the core bed from top loading locations and exit via one of three de-fueling chutes at the bottom of the bed. After exiting the core, the pebbles are checked for general spherical integrity and burnup. If insufficient burnup has been achieved, a pebble is returned to the top of the core by the pneumatic Fuel Handling and Storage System. On average, pebbles make about six passes through the core with a burnup of approximately 95000 MWd/t before being placed in spent fuel storage tanks. In the equilibrium cycle the fuel has enrichment of 9.6 wt% U235.

The inner and outer reflectors of the annular core are composed of graphite blocks that provide moderation and reflection of neutrons and provide locations for control systems. There are 24 control rods in outer reflector and eight Reserve Shutdown System locations in the inner reflector where borated graphite spheres are inserted. This configuration is shown in Figure 3.

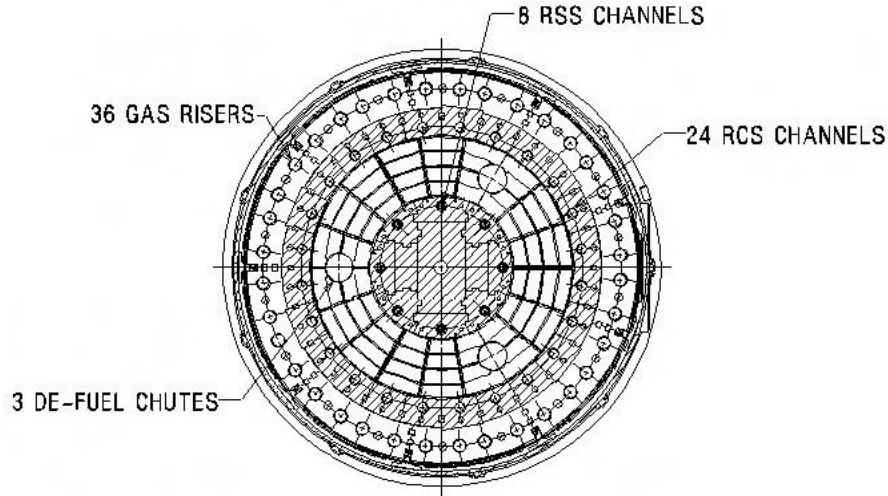


Figure 3: PBMR-400 Core configuration.

2.2 MODEL DESCRIPTION

2.2.1 Heat Conduction Equation for Pebble-bed

Due to temperature gradients in radial and axial directions, heat fluxes occur in a pebble-bed. These fluxes occur by means of conduction and radiation and can be calculated by an effective thermal conductivity k_p of pebble-bed. This effective thermal conductivity takes into consideration conduction and thermal radiation and can be employed to write an effective conduction equation in terms of fuel surface temperatures T_{fs} . Equation (7) is used for the simulation of the heat transfer through a pebble-bed.

$$\rho_p c_{pp} \frac{\partial T_{fs}}{\partial t} - \nabla \cdot k_p \nabla T_{fs} = \dot{q}_p \quad (7)$$

In Equation(7), ρ_p and c_{pp} are effective pebble-bed density and volumetric heat capacity. The volumetric energy source $\dot{q}_p$ of pebble-bed is defined as

$$\dot{q}_p = \dot{Q} - \frac{6\alpha}{d}(1-\varepsilon)(T_{fs} - T_c) \quad (8)$$

The quantities $\dot{Q}$, α , d , ε and T_c are volumetric heat source from fission, convection heat transfer coefficient of fuel ball surface, diameter of fuel ball, void fraction of pebble-bed and coolant temperature, respectively.

2.2.2 Energy Equation for Coolant in Pebble-bed

The energy equation is defined as

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\vec{u}(\rho E + p)) = \nabla \cdot (k_{eff} \nabla T_c + \vec{\tau}_{eff} \cdot \vec{u}) + \dot{q}_c \quad (9)$$

where $\vec{u}$ and ρ are coolant velocity and density, respectively. In Equation(9), k_{eff} is effective conductivity defined as $k_c + k_t$, where k_c is heat conductivity of the coolant and k_t is the turbulent thermal conductivity, defined according to turbulent model being used. $\dot{q}_c$ is the average volumetric heat source from the fuel ball surfaces and defined by Equation(10).

$$\dot{q}_c = \frac{6\alpha}{d}(1-\varepsilon)(T_{fs} - T_c) \quad (10)$$

In Equation(9), E is defined as

$$E = h - \frac{p}{\rho} + \frac{u^2}{2} \quad (11)$$

where h is the sensible enthalpy.

2.2.3 The Mass Conservation Equation

The equation for conservation of mass or continuity equation is written as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \quad (12)$$

2.2.4 Momentum Conservation Equation

Conservation of momentum is described in the following form

$$\frac{\partial}{\partial t}(\rho \vec{u}) + \nabla \cdot (\rho \vec{u} \vec{u}) = -\nabla p_s + \nabla \cdot \vec{\tau} + \rho \vec{g} + \vec{F} \quad (13)$$

where $\vec{g}$ is the gravitational force. In Equation(13), the momentum source $\vec{F}$ includes a viscous loss term and an inertial loss term due to resistance of pebble-bed.

2.2.5 Turbulence Modeling In Pebble-Bed

The effect of turbulence in pebble-bed is suppressed by setting the turbulent contribution to coolant viscosity equal to zero. The inlet turbulence quantities are transported through the pebble-bed, but their effect on the fluid mixing and momentum is ignored. In addition, the generation of turbulence is set to zero in the pebble-bed.

2.2.6 Numerics of Heat Transfer at Core-Reflector Interfaces

At the reflector walls which encloses the pebble-bed core, heat transfer occurs by two mechanisms, first one occurs between fuel balls and reflector wall surface and the second one occurs between the coolant and the reflector wall surface. In core region, two energy equations (for heat conduction in pebble-bed and for coolant) are solved while only one conduction equation is solved in reflector region. These three equations may be connected by the method described below.

Assuming that the gradient of a scalar is available and unstructured mesh, the face value of the scalar ϕ can be approximated by,

$$\phi_f = \phi_0 + \nabla \phi \cdot d\vec{r} \quad (14)$$

where $d\vec{r}$ is the vector that connects the cell centroid with the face centroid. The gradient in this case is evaluated at the cell centroid where ϕ_0 is also stored. Then, the diffusive flux q_f'' , across a face f , of a scalar ϕ is given by,

$$q_f'' = -\Gamma_f \nabla \phi \cdot \vec{A} \quad (15)$$

where Γ_f is the diffusion coefficient at the face and $\vec{A}$ is the face area normal vector.

The diffusive flux q_f'' defined in Equation(15), may be approximated as,

$$q_f'' = -\Gamma_f \left[\frac{(\phi_0 - \phi_f)}{ds} \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} + \nabla \phi \cdot \vec{e}_s \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} - \nabla \phi \cdot \vec{A} \frac{1}{|\vec{A}|} \right] \quad (16)$$

In Equation(16), ds is the distance between cell centroid and the face centroid, and $\vec{e}_s$ is the unit normal vector directed from centroid of cell to face centroid which are shown in Figure 4.

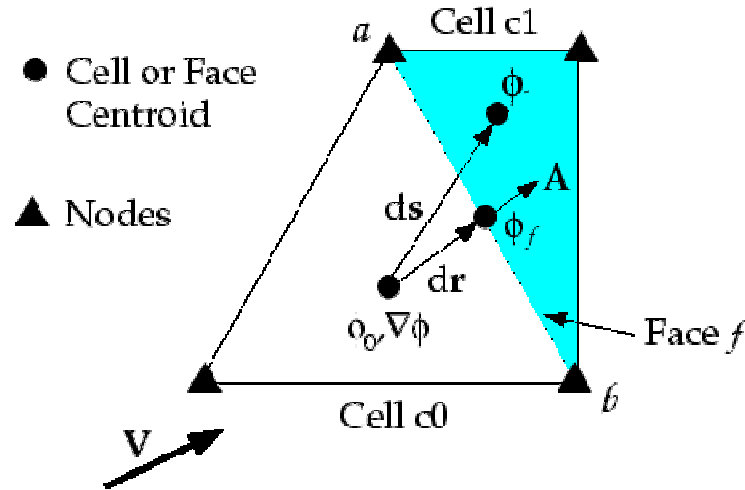


Figure 4 : Vector and Gradient Definitions

With the knowledge of definitions above, thermal boundary condition at the core-reflector interface for the heat equation of pebble-bed can be defined as,

$$q''_{w,p} = -k_p \left[\frac{(T_{fs,c0} - T_w)}{ds} \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} + \nabla T_{fs} \cdot \vec{e}_s \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} - \nabla T_{fs} \cdot \vec{A} \frac{1}{|\vec{A}|} \right] \quad (17)$$

Also, thermal boundary condition at the reflector wall for the coolant energy equation can be written as,

$$q''_{w,c} = -k_{eff} \left[\frac{(T_{c,c0} - T_w)}{ds} \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} + \nabla T_c \cdot \vec{e}_s \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} - \nabla T_c \cdot \vec{A} \frac{1}{|\vec{A}|} \right] \quad (18)$$

And finally, the thermal boundary condition on the core-reflector interface for the heat conduction equation of reflector region can be defined as,

$$q''_{w,R} = k_p \left[\frac{(T_{fs,c0} - T_w)}{ds} \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} + \nabla T_{fs} \cdot \vec{e}_s \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} - \nabla T_{fs} \cdot \vec{A} \frac{1}{|\vec{A}|} \right] + k_{eff} \left[\frac{(T_{c,c0} - T_w)}{ds} \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} + \nabla T_c \cdot \vec{e}_s \frac{|\vec{A}|}{\vec{A} \cdot \vec{e}_s} - \nabla T_c \cdot \vec{A} \frac{1}{|\vec{A}|} \right] \quad (19)$$

2.3 ASSESMENT OF THE MODEL IN FLUENT

FLUENT CFD software is highly customizable by means of user defined scalar (UDS) transport equations and user defined functions (UDF) which are coded in C programming language. UDFs are compiled by an ANSI C compiler and dynamically loaded with FLUENT to enhance the standard features of the code. They can be coded using some C macros named 'define macros' supplied by FLUENT. FLUENT code comes with required C header files which enable the user to access data from FLUENT solver using predefined macros and functions.

Extra transport equations can be solved for any arbitrary user defined scalar by means of UDS adding capability of the software. For an arbitrary scalar, FLUENT solves the equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i} \left(\rho u_i \phi_k - \Gamma_k \frac{\partial \phi_k}{\partial x_i} \right) = S_{\phi_k} \quad k = 1, \dots, N \quad (20)$$

where Γ_k and S_{ϕ_k} are the diffusion coefficient and source term supplied by the user as constants or functions defined by means of UDFs. For the steady-state case FLUENT solves the following equation, if the convective flux is not to be computed.

$$-\frac{\partial}{\partial x_i} \left(\Gamma_k \frac{\partial \phi_k}{\partial x_i} \right) = S_{\phi_k} \quad k = 1, \dots, N \quad (21)$$

2.3.1 Heat Conduction Equation for Pebble-bed

The Equation(7) is modeled using UDS feature of FLUENT. One UDS transport equation is used without convective flux term. Effective conductivity k_p and heat source $\dot{q}_p$ terms are defined using UDFs. These source codes are listed in Appendix 1.

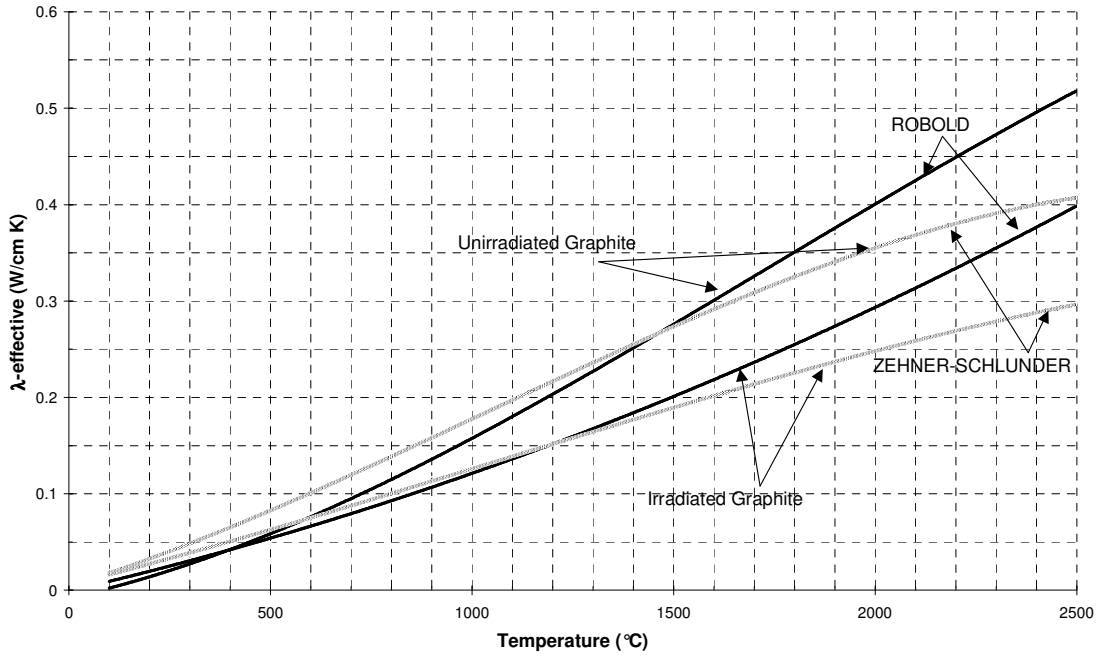


Figure 5: Effective thermal conductivity in a Pebble Bed

The effective heat conductivity k_p of pebble-bed core, which synthesizes the effects of conduction and thermal radiation among the fuel element surfaces, is described according to model by Zehner-Schlunder [1]. This model is depicted in Figure 5. This model is employed to describe the heat transfer between pebbles at temperatures from 100 °C to 2500 °C.

The average convective heat transfer coefficient, α , of fuel-ball surfaces, in Equation(8) and Equation(10) is defined as,

$$\alpha = \frac{\text{Nu} k_c}{d} \quad (22)$$

Where k_c and d are heat conductivity of the coolant and diameter of fuel element respectively. Nusselt number, Nu , is evaluated as,

$$\text{Nu} = 1.27 \frac{\text{Pr}^{1/3}}{\varepsilon^{1.18}} \text{Re}^{0.36} + 0.033 \frac{\text{Pr}^{1/2}}{\varepsilon^{1.07}} \text{Re}^{0.86} \quad (23)$$

In Equation(23) Prandtl number, Pr , is defined as,

$$\text{Pr} = \frac{c_p \mu}{k_c} \quad (24)$$

where c_p and μ are the heat capacity and dynamic viscosity of the coolant, respectively. Reynolds number, Re , is defined as,

$$\text{Re} = \frac{|\vec{u}| \rho d}{\mu} \quad (25)$$

2.3.2 Modeling Source Terms

The source terms $\dot{q}_p$, $\dot{q}_c$ and $\vec{F}$ in equations (7), (9) and (13) are modeled by means of special macros defined as Define_Source macros. The UDF source codes for these

macros are listed in Appendix 1. The models of the terms $\dot{q}_p$ and $\dot{q}_c$ are defined in Equations (8) and (10), respectively.

The momentum source term $\vec{F}$ in Equation(13) is defined as,

$$\vec{F} = \psi \frac{1-\varepsilon}{\varepsilon^3} \frac{1}{d} \frac{\rho}{2} |\vec{u}| \vec{u} \quad (26)$$

where ψ is the pressure drop coefficient and it is a function of Reynolds number.

The dependence of the pressure drop coefficient on Reynolds number, $\psi = \psi(\text{Re})$, has been investigated by numerous experimenters for unique-size spherical packing, multi-size spherical packing and multi-size pacing. For the application to pebble-bed type high temperature reactors the results of the randomly packed bed with spherical particles are considered only. The reference [2] published a review which treats about thirty papers. In it the dependence of ψ on Re is given by

$$\psi = \frac{320}{\left(\frac{\text{Re}}{1-\varepsilon}\right)} + \frac{6}{\left(\frac{\text{Re}}{1-\varepsilon}\right)^{0.1}} \quad (27)$$

Re is defined as in Equation(25).

Equation(27) is confirmed by experiments up to $\text{Re}/(1-\varepsilon) = 5 \times 10^4$. The first term of Equation(27) represents the asymptotic solution for laminar flow, and the second for turbulent flow, i.e.

$$\psi_\ell = 320 \left(\frac{\text{Re}}{1-\varepsilon}\right)^{-1.0} \quad (28)$$

$$\psi_t = 6 \left(\frac{\text{Re}}{1-\varepsilon}\right)^{-0.1} \quad (29)$$

2.3.3 Effects of Void Fraction ε

The experimental data of the pressure drop coefficient, measured by different authors, indicate a large scatter. This evidence is to a considerable extent due to a deficit of knowledge about the actual void fraction, ε . The distribution of the spheres in a pebble bed is no longer random near the wall because of the orientation forced by the presence of the wall. In a distance of half sphere diameter from the wall the void fraction decreases from unity to a value depending on D/d , where D is the core diameter. The high values of void fraction near the wall cause non uniform velocity distribution across the core.

Benenati and Brosilow measured the distribution of the void fraction as a function of the core radius for several ratios of D/d . At a distance of $d/2$, void fraction has a minimum and then levels out with decreasing amplitudes to a value of 0.375 at large distances from the wall.

In estimating the magnitude of the wall effect, it is assumed that the core consists of two parts of different void fractions ε_w and ε_c . The quantities ε_w and ε_c are a function of D/d . Approximating the experimental results of Benenati and Brosilow [3], wall void fraction ε_w is found by

$$\varepsilon_w = \frac{63.6}{\left(\frac{D}{d} + 15\right)^2} + 0.43 \quad (30)$$

and central void fraction can be calculated by means of the equation of conservation of mass:

$$\varepsilon_c = \frac{\varepsilon_t - \varepsilon_w}{\left(1 - \frac{d}{D}\right)^2} + \varepsilon_w \quad (31)$$

2.4 MODEL APPLICATION

The model described in section “2.2 Model description” is applied to study thermal-hydraulics of Pebble-Bed Modular Reactor (PBMR-400). All the calculations are done according to definitions and parameters taken from Coordinated Research Program No.5 (CRP-5) benchmark definition [5].

2.4.1 Simplifications Introduced

A basic set of input data and boundary conditions that describe the general reactor layout, material definition, thermal-hydraulics benchmark specification [5] was compiled as a part of the ongoing verification and validation effort within PBMR for the activities of the Coordinated Research Program No. 5 (CRP-5), under the auspices of the International Atomic Energy Agency (IAEA). These data set and boundary conditions are used in the calculations. The simplifications used were also introduced in the data sets.

The reference design for the PBMR was derived from the 400 MW PBMR design. Simplifications could make the core design essentially two dimensional (r,z) as shown in Figure 6. It includes flattening of the upper surface of the pebble-bed and the removal of the bottom cone and de-fuel chutes, resulting in a flat bottom reflector. Reactor and core structure dimensions are given in Table 2 and Table 3.

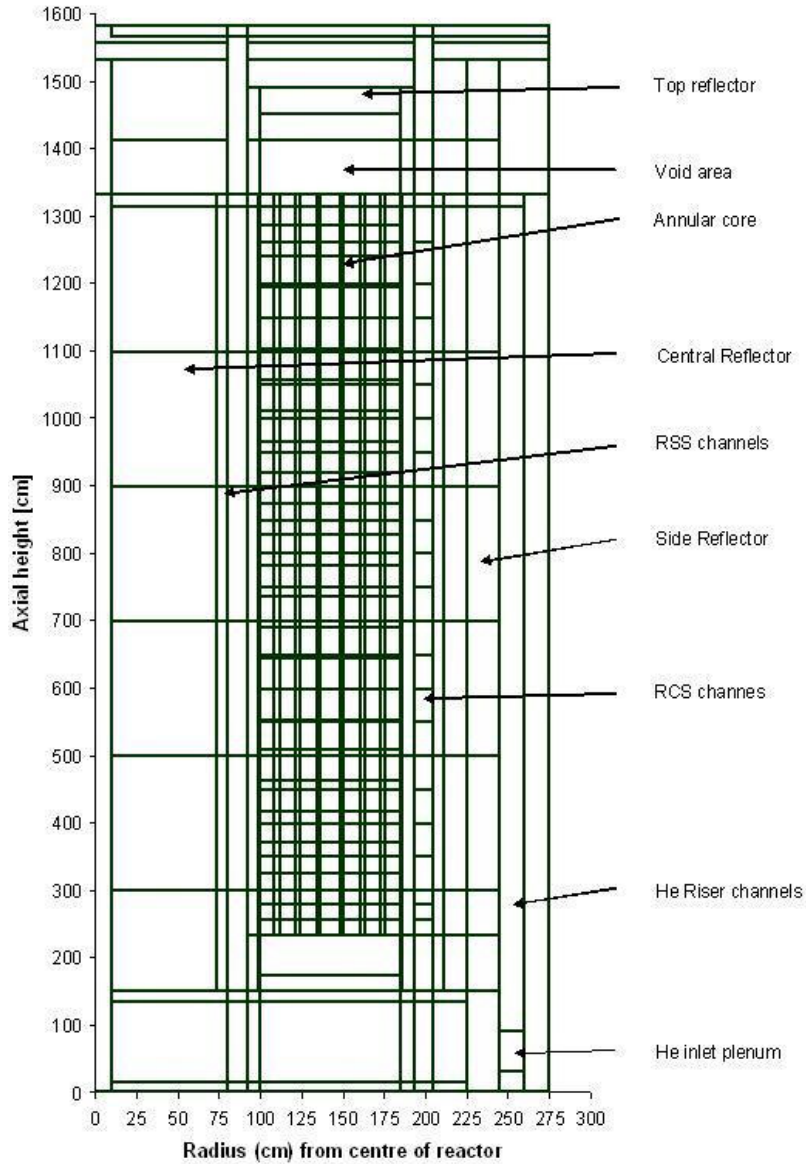


Figure 6: PBMR 2-D Geometry Layout [5]

Table 2: Core geometrical specifications

#	Description	Unit	Value
1	Equivalent core outer diameter	m	3.7
2	Outer diameter of central reflector.	M	2.0
3	Effective cylindrical height of the core (Flattened core surface at the top to the volume averaged flattened bottom cone).	M	11.0
4	Total fuel volume	M ³	83.7156

#	Description	Unit	Value
5	Effective height of the upper void cavity (levelled core surface to bottom of top reflector).	M	0.785
6	Outer diameter of boring / hole in centre of central reflector	M	0.2
7	Effective annular thickness of the inner reflector block.	M	0.4
8	Equivalent annular thickness of the outer reflector block.	M	0.5
9	Inner diameter of the core barrel	M	5.750
10	The wall thickness of the core barrel.	M	0.05
11	Inner diameter of the RPV	M	6.2
12	The wall thickness of the RPV.	M	0.180
13	Total height of top reflector	M	1.7
14	Total height of bottom reflector	M	5.52
15	Top steel plate thickness	M	0.3
16	Bottom steel plate thickness	M	0.1
17	Effective volume of fuel in bottom cones (as from CAD – flattened cone surfaces).	M ³	6.7183
18	Effective volume of fuel in top fuel piles.	M ³	3.8872
19	Fuel effective cylindrical height (flattened top to the top / start of cone)	M	10.117
20	Height of the upper void cavity (bottom of top reflector to top of fuel piles)	M	0.328
21	Bottom reflector cone maximum axial height from top of de-fuelling chute	M	1.483
22	Number of de-fuelling chutes		3
23	De-fuelling chutes inner diameter	M	0.4
24	De-fuelling chutes are equally spaced (120°) on a pitch centre diameter (PCD) of:	M	2.88
25	Equi-spaced central reflector inspection holes	#	16
26	Inspection holes diameter	M	0.05
27	Inspection holes are on a pitch circle diameter (PCD) of:	M	1.830

Table 3 In-Core flow channel geometrical specifications

#	Description	Unit	Value
1	Axial distance between bottom of the core and the inlet plenum cross-section	m	1.42
2	Inlet plenum inner diameter	m	4.822
3	Inlet plenum outer diameter	m	5.262
4	Inlet plenum height	m	0.6
5	Diameter of cold gas inlet pipes (2 of)	m	0.6
6	Inlet pipes centre points	° (degrees)	40 and 320
7	Number of gas risers in outer reflector	#	36
8	PCD of risers.	m	5.042
9	Diameter of risers (typical)	m	0.170
10	Distance from bottom of top reflector to top of top inlet plenum	m	0.705
11	Top inlet plenum inner diameter	m	4.5
12	Top inlet plenum outer diameter	m	4.75
13	Top inlet plenum height	m	0.7
14	Centre line axial distance between bottom inlet plenum and outlet plenum	m	2.0
15	Outlet plenum inner diameter	m	1.6
16	Outlet plenum outer diameter	m	2.1
17	Outlet plenum height	m	1.2
18	Flow diameter of outlet duct to hot outlet pipe	m	1.034

Thermal-hydraulics simplifications include the speciation of stagnant helium between the barrel and Reactor Pressure Vessel (RPV) and stagnant air between the RPV and heat sink (outer boundary). The coolant flow is restricted to upwards flow from the inlet below the core within a porous ring in the reflector and downwards flow through the pebble bed to the outlet plenum. Approximate effective porosities in reflector regions due to gaps and slits are listed in Table 4. No cooling and or leakage paths were defined. Other simplifications include the assumption that all heat sources (from fission) will be deposited locally, i.e. in the fuel, and that no other heat sources exist outside the core (for example neutron absorption in boron shielding regions).

Table 4 Approximate Effective Porosities in Reflector Regions due to gaps and slits

#	Description	Unit	Value
1	Effective void fraction of inner 60 cm block of central reflector		0.0
2	Effective void fraction of bulk of outer block of central reflector (containing the SAS channel and inspection holes not included in this value) (60 cm – 94 cm)		0.0
3	Effective void fraction of outer region of outer block of central reflector (coolant slots only) (94 cm – 100 cm)		0.035
4	Effective void fraction of inner block of the side reflector (185 cm – 225 cm) (approximated over the mesh)		0.007
5	Effective void fraction of first part of the outer block of the side reflector (225 cm – 259.5 cm)		0.0
6	Effective void fraction of first part of the outer block of the side reflector (259.5 cm – 275 cm)		0.058
7	Effective void fraction of bottom reflector (helium outlet slots) to be used below the core.		0.193

Note: In the above values none of the holes and penetrations was included. For example the hole in the central reflector is not included in the central reflector inner block effective porosity.

2.4.2 Material Properties

All material properties are defined according to definitions in CRP-5 benchmark definition of PBMR-400 [5].

The thermal conductivity of graphite as a function of fast neutron dose and temperature is of importance for calculating the temperature distribution. This is also true for the fuel element graphite (NUKEM A3-3) as well as for the structural graphite. The A3-3 graphite data is taken from the work by Binkele [24], up to a fast dose of approximately $6.09 \times 10^{21} \text{ cm}^{-2}$ EDN (approximately $1.59 \times 10^{21} \text{ cm}^{-2}$ EDN is expected for the PBMR fuel) and a temperature up to 1 000 °C which is the range covered by the experimental work. Above 1 000 °C, the data is conservatively extrapolated as a constant value as shown in Figure 7.

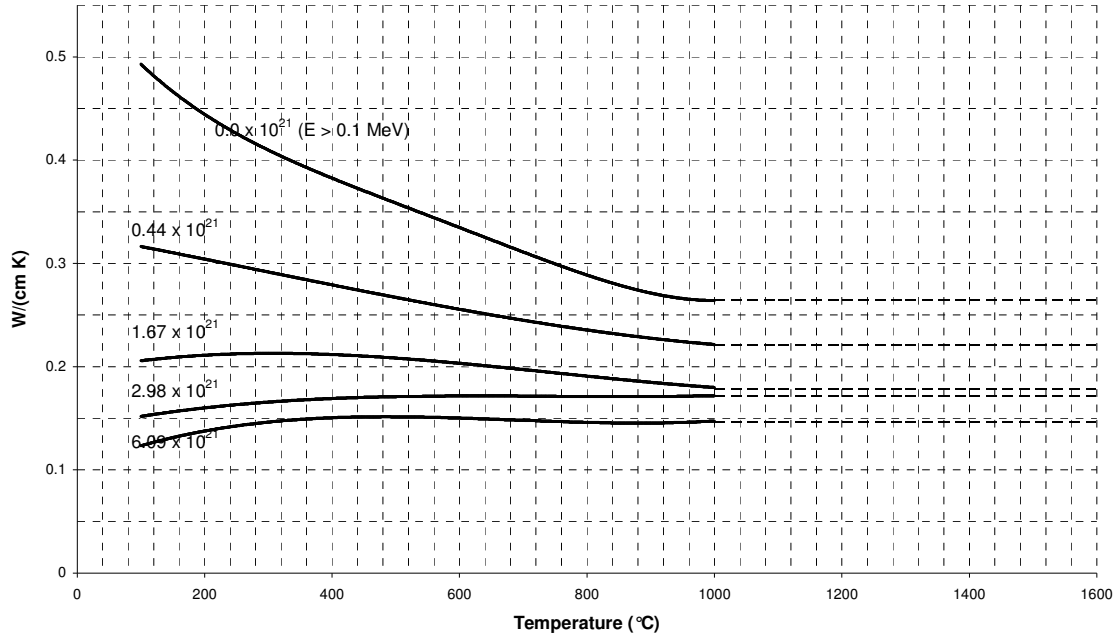


Figure 7: Thermal conductivity of A3-3 matrix graphite as a function of temperature and neutron fluence irradiated at 950 $^{\circ}C$. [5]

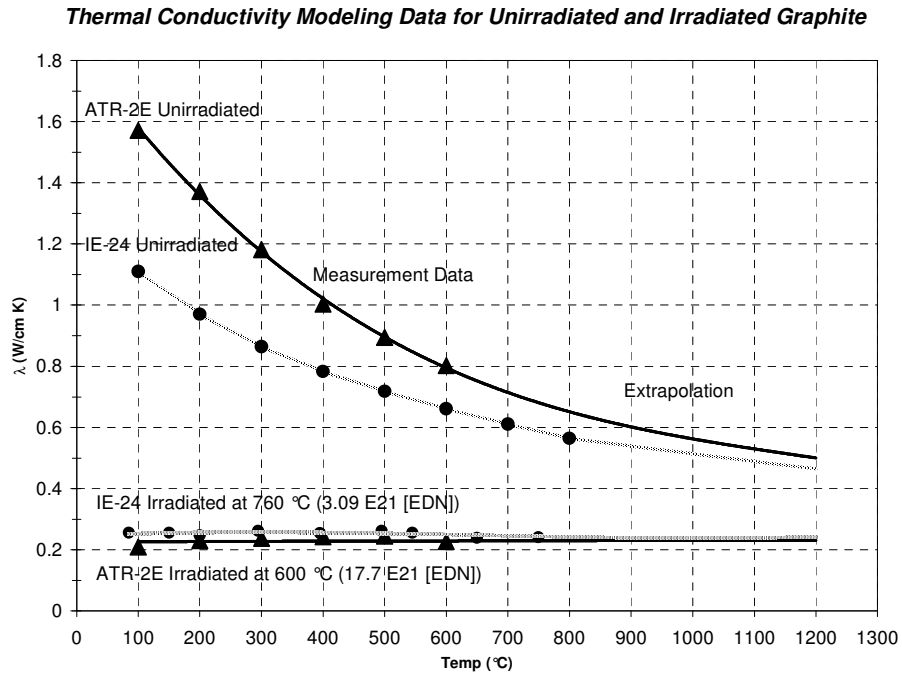


Figure 8: ATR-2E graphite thermal conductivity as a function of temperature and dose. [5]

The thermal conductivities of the reflectors blocks are based on the measured properties of ATR-2E graphite. In Figure 8, a comparison is provided for the irradiated and un-irradiated datasets of the measured thermal conductivities for ATR-2E graphite.

The emissivity values used in radiation heat transfer calculations and radiation boundary conditions are listed in

Table 5 Emissivity of materials

#	Description	Unit	Value
1	The emissivity of the graphite structures.		0.8
1	The emissivity of the core barrel: Type 316 Stainless Steel		0.85
3	The emissivity of the RPV (The convective heat transport in the reactor cavity may be accounted for by an increase in the effective thermal conductivity.)		0.91
4	The emissivity of concrete.		0.9

Thermal conductivities used in this study are summarized below in Table 6.

Table 6 Thermal conductivities of materials

#	Description	Unit	Value
1	Pebble bed: Zehner-Schlünder and Robold (see Figure 5)	$\text{W m}^{-1} \text{K}^{-1}$	Correlation
2	Matrix graphite: Nukem A3-3 data irradiated at 950 °C (see Figure 7)	$\text{W m}^{-1} \text{K}^{-1}$	Measured data
3	Reflector graphite: Temperature-fluence dependent thermal conductivity of the inner reflector blocks and central column as provided in Figure 8	$\text{W m}^{-1} \text{K}^{-1}$	Measured data
4	Core Barrel (Type 316 SS) – Assumed constant	$\text{W m}^{-1} \text{K}^{-1}$	17.0
5	Reactor Pressure Vessel (SA 508) – Assumed constant	$\text{W m}^{-1} \text{K}^{-1}$	38.0
6	Helium stagnant $k = 2.682 \times 10^{-3} T^{0.71(1-2 \times 10^{-9} p)} (1 + 1.123 \times 10^{-8} p)$; where k (W/m/K), T (K), and p (Pascal).	$\text{W m}^{-1} \text{K}^{-1}$	Correlation used

#	Description	Unit	Value
7	Air stagnant $k = a \cdot b^T \cdot T^c$; where $a = 9.1446\text{E-}05$; $b = 9.9958\text{E-}01$; $c = 1.0148\text{E+}00$	$\text{W m}^{-1} \text{K}^{-1}$	Correlation used

2.4.3 Power Profile

A simplified axially averaged power profile was constructed from the relative power distribution in five equidistant core axial channels (17 cm each). The relative power densities for the five channels are given below [5]:

1 (Fuel)	2 (Fuel)	3 (Fuel)	4 (Fuel)	5 (Fuel)
1.119	1.023	0.953	0.957	0.945

The interpretation is that the power density in the 1st axial flow channel is calculated as $1.119 * (400 \text{ (MW)} / 83.7156 \text{ m}^3)$, $1.023 * (400 \text{ (MW)} / 83.7156 \text{ m}^3)$ in the 2nd, etc.

2.4.4 Reactor Main Coolant Specification

Table 7 Main helium flow parameters

#	Description	Unit	Value
1	He inlet temperature	$^{\circ}\text{C}$	488.1
2	Total inlet mass flow rate	kg/s	185.31
3	Inlet Pressure	kPa	8915

2.4.5 Turbulence Modeling

The standard $k-\varepsilon$ model was used in the model. This model is based on transport equations for the turbulent kinetic energy ' k ' and its dissipation rate ' ε '. These equation are given as,

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] + G_k - \rho \varepsilon \quad (32)$$

and

$$\frac{\partial}{\partial t}(\rho\varepsilon) + \frac{\partial}{\partial x_i}(\rho\varepsilon u_i) = \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] + C_{1\varepsilon} \frac{\varepsilon}{k} (G_k + C_{3\varepsilon} G_b) - C_{2\varepsilon} \rho \frac{\varepsilon^3}{k} \quad (33)$$

In these equations, G_k represents the generation of turbulent kinetic energy due to mean velocity gradients, calculated as

$$G_k = \mu_t S^2 \quad (34)$$

where S is the modulus of the mean rate of stress tensor, defined as

$$S \equiv \sqrt{2S_{ij}S_{ij}} \quad (35)$$

$C_{1\varepsilon}$, $C_{2\varepsilon}$, and $C_{3\varepsilon}$ are constants, σ_k and σ_ε are turbulent Prandtl numbers for k and ε . These constants have the following values:

$$C_{1\varepsilon} = 1.44, \quad C_{2\varepsilon} = 1.92, \quad C_\mu = 0.09, \quad \sigma_k = 1.0, \quad \sigma_\varepsilon = 1.3 \quad (36)$$

The effect of turbulence was suppressed by setting the turbulent contribution to coolant viscosity equal to zero within porous defined regions listed in Table 4. The inlet turbulence quantities are transported through these regions, but their effect on the fluid mixing and momentum is ignored. In addition, the generation of turbulence is set to zero in these regions.

2.4.5 Radiation Heat Transfer Modeling [18]

For the modeling of radiative heat transfer in enclosures discrete ordinates (DO) model was used. The discrete ordinates (DO) radiation model solves the radiative transfer equation (RTE) for a finite number of discrete solid angles, each associated with a vector direction $\vec{s}$. The DO model solves for as many transport equations as there are directions $\vec{s}$.

The DO model considers the radiative transfer equation (RTE) in the direction $\vec{s}$ as a field equation,

$$\nabla \cdot (I(\vec{r}, \vec{s}) \vec{s}) + (a + \sigma_s) I(\vec{r}, \vec{s}) = an^2 \frac{\sigma T^4}{\pi} + \frac{\sigma_s}{4\pi} \int_0^{4\pi} I(\vec{r}, \vec{s}') \Phi(\vec{s} \cdot \vec{s}') d\Omega' \quad (37)$$

where $\vec{r}$ = position vector

$\vec{s}$ = direction vector

$\vec{s}'$ = scattering direction vector

a = absorption coefficient

n = refractive index

σ_s = scattering coefficient

σ = Stefan-Boltzmann constant

I = radiation intensity

T = local temperature

Φ = phase function

Ω' = solid angle

For the helium cavities in the model, Equation (37) becomes

$$\nabla \cdot (I(\vec{r}, \vec{s}) \vec{s}) = 0 \quad (38)$$

2.5 DISCUSSION AND THERMAL-HYDRAULICS RESULTS

2.5.1 Grid Structure

2D Axi-symmetric grid structure of the reactor model is shown in Figure 9. The yellow line in Figure 9 shows the symmetry axis. In the core region and in the most of the reflector regions quadrilateral meshes were used. Total number of meshes is 37075.

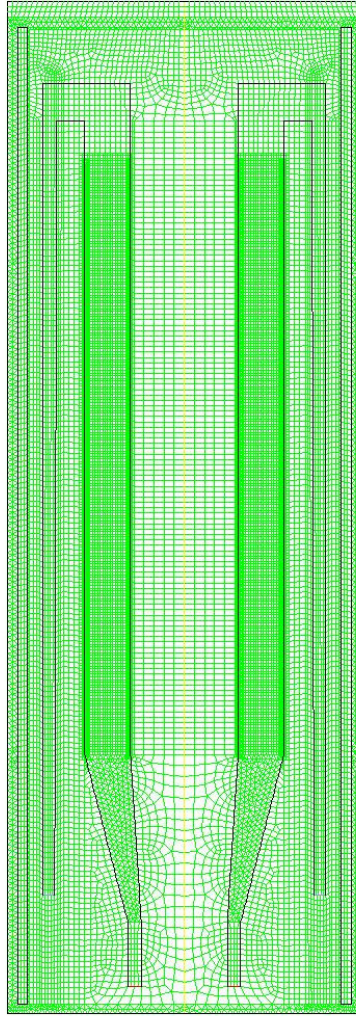


Figure 9: Model grid structure

For analyzing the grid sensitivity, calculations were repeated for two other grids with 73174 cells and 225409 cells.

2.5.2 Grid Sensitivity

Calculations were repeated for three different grids with different grid sizes of 37075, 73174, and 225409 meshes. The same results for global parameters were obtained for these different grids. However the convergence behavior of the solutions is worse for finer meshes. This may be due to increasing numerical errors resulting from very small mesh sizes.

2.5.3 Convergence and Analysis of Global Balances

Due to the method by which power distribution was modeled, there were small differences in total power input to the core for the three different grid structures used for the calculations. The power distribution was hold in a User Defined Memory (UDM) and total power was calculated by integrating this UDM field over the core volume.

In judging convergence, energy and mass balances are analyzed in addition to residuals. For the global energy balance, the following equation can be written as,

$$E_{c,i} - E_{c,e} - E_{sysb} = Q_{core} \quad (39)$$

Where $E_{c,i}$ and $E_{c,e}$ are the total energy of the coolant entering the system and the total energy of the coolant leaving the system. E_{sysb} is the total heat transfer at the system boundaries and Q_{core} is the total core power obtained by integrating the core power distribution over the core volume.

For the case of the grid with 37075 cells the convergence was obtained in approximately 3000 iterations. The total core power Q_{core} is 400.662 MW . The total energy removed from the system by the helium coolant $E_{c,i} - E_{c,e}$ is 399.061 MW . The total heat transfer at the system boundaries (in this case this is core vessel) is -1.601 MW . Using this values, $E_{c,i} - E_{c,e} - E_{sysb} = 400.662$ MW , which is the calculated total core power.

For the side reflector walls adjacent to the core, following balance equation can be written.

$$Q_{c,w} + Q_{b,w} = -Q_{rs,w} \quad (40)$$

Where $Q_{c,w}$ and $Q_{b,w}$ are the total heat transfer from helium coolant and the total heat transfer from the fuel balls to reflector at the core side of the reflector-core wall

interface. $Q_{rs,w}$ is the total heat transfer at reflector side of the reflector-core wall interface. These values are calculated as,

$$Q_{c,w} = 22621.95 \text{ W} \quad Q_{b,w} = -583230.11 \text{ W} \quad Q_{rs,w} = 560608.20 \text{ W}$$

An interesting observation about the values above is about the value $Q_{c,w} = 22621.95 \text{ W}$ which means helium is heated by the reflector wall. This is normal due to higher temperature values on the reflector wall which is heated by the fuel balls adjacent to the wall.

For the case of the grid with 73174 cells the convergence was obtained in approximately 5000 iterations. The global energy balance parameters for this case can also be written as,

$$E_{c,i} - E_{c,e} = 397.434 \text{ MW} \quad E_{sysb} = -1.601 \text{ MW} \quad Q_{core} = 399.035 \text{ MW} \quad (41)$$

$$E_{c,i} - E_{c,e} - E_{sysb} = 399.035 \text{ MW}$$

For the case of the grid with 225409 cells the convergence was obtained in approximately 6000 iterations. The global energy balance parameters for this case can be written as,

$$E_{c,i} - E_{c,e} = 397.940 \text{ MW} \quad E_{sysb} = -1.601 \text{ MW} \quad Q_{core} = 399.5439 \text{ MW} \quad (42)$$

$$E_{c,i} - E_{c,e} - E_{sysb} = 399.541 \text{ MW}$$

2.5.3 Thermal-hydraulics Results

Another possible way of modeling a pebble-bed reactor was by using the Eulerian multi phase flow model available in FLUENT. In this method, the secondary solid phase may be modeled as packed bed which can capture the pebble-bed characteristics. However this method has also its own difficulties. In addition, the radiation heat transfer models were not implemented with the Eulerian multi phase models in FLUENT yet.

In Table 8, key global results are listed.

Table 8: Global parameters

Core Pressure Drop	237 kPa
Helium Outlet Temperature	1093 K
Average Fuel Surface Temperature	1014 K
Average Helium Temperature in Core	954 K
Average Temperature of Central Reflector	1023 K
Average Temperature of Outer Reflector	754 K

The temperature distribution is shown in Figure 10. The contours in the core and cavity regions enclosed in black lines show helium temperatures and the others show solid material temperatures. The fuel surface temperatures which are calculated by solving user defined scalar transport equations are shown in Figure 11. The temperature distribution for all solid material is shown in Figure 12. The counter lines in core region in Figure 12 shows fuel surface temperatures.

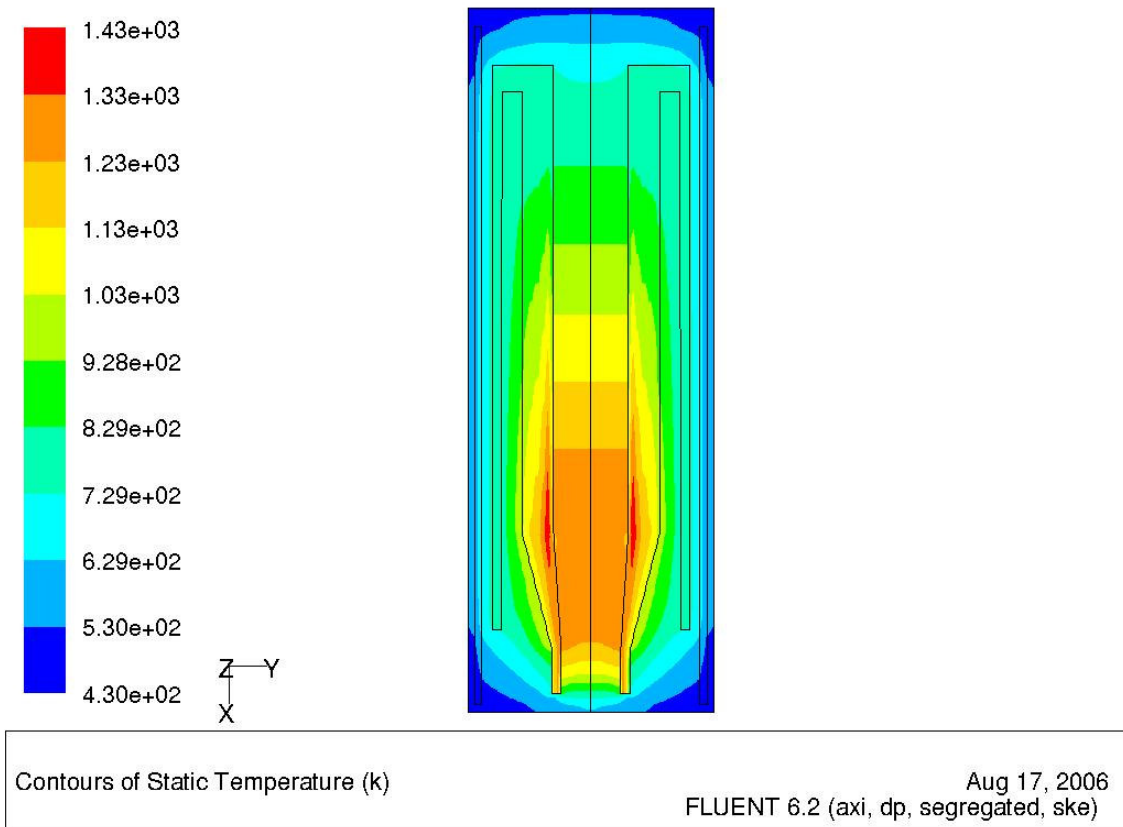
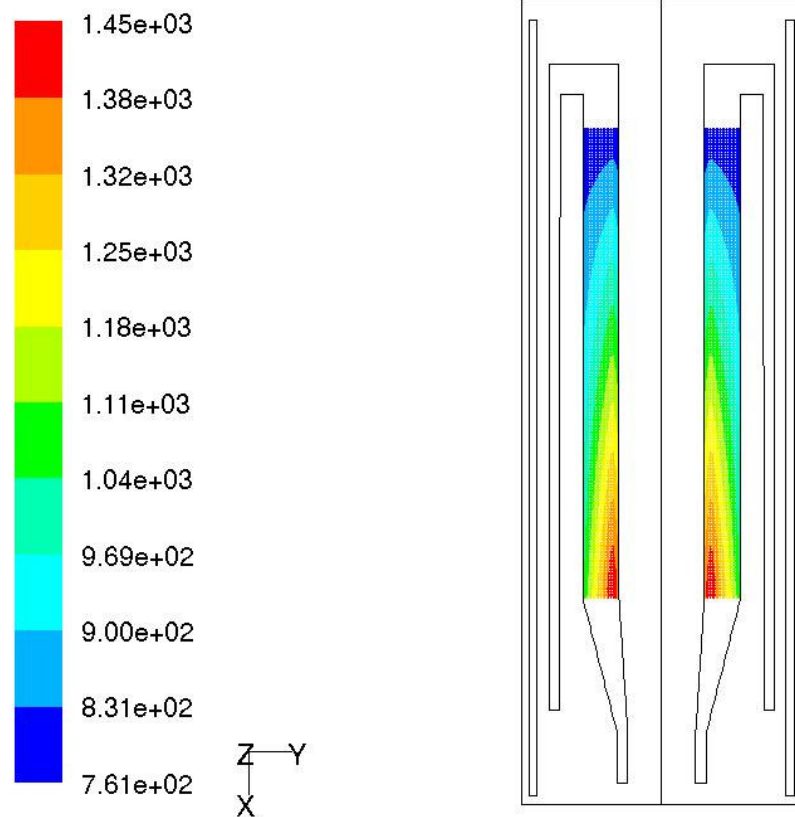


Figure 10 Contours of Temperatures in K.



Contours of User Scalar 0

Aug 17, 2006
FLUENT 6.2 (axi, dp, segregated, ske)

Figure 11 Contours of fuel surface temperatures.

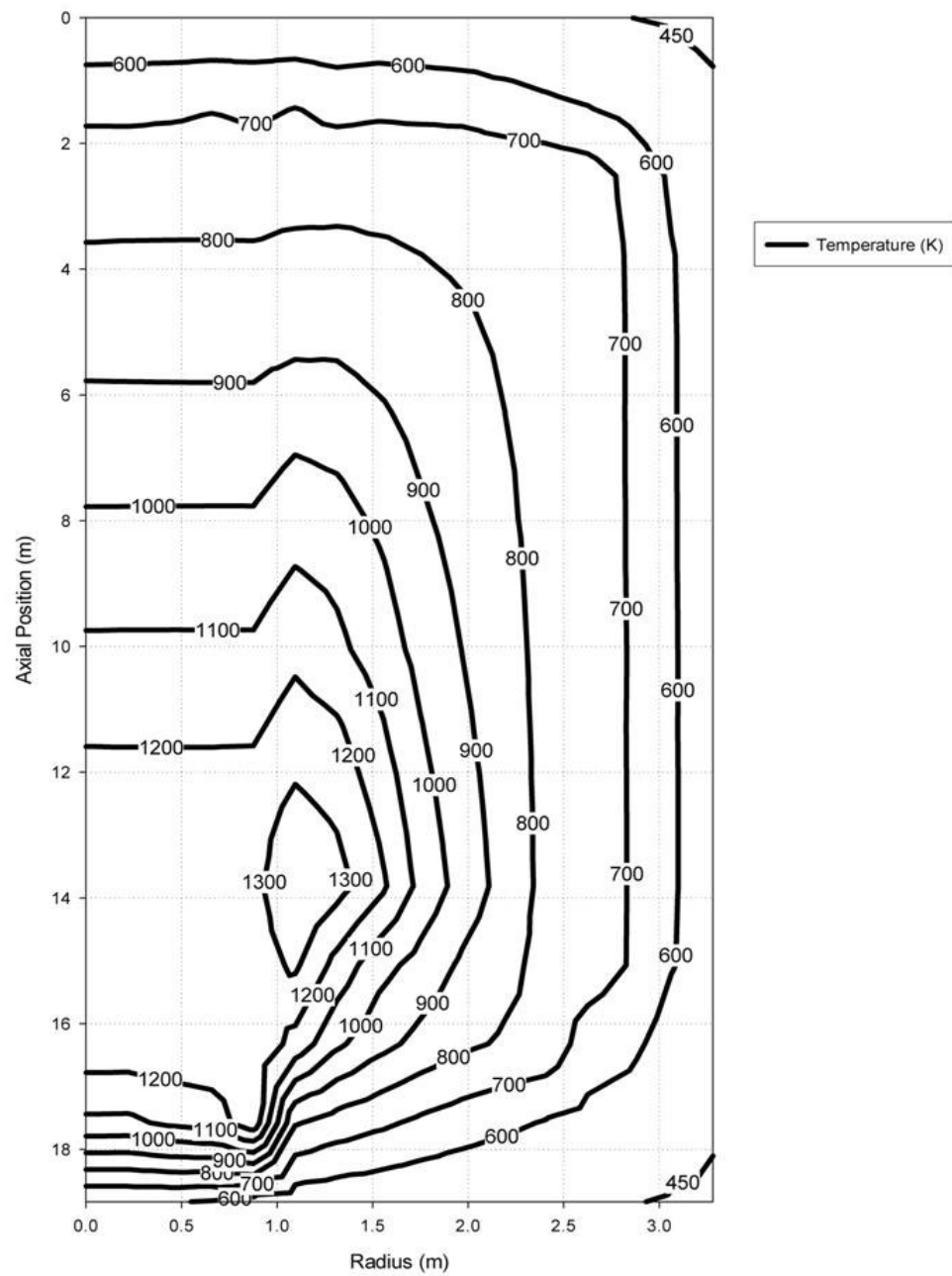


Figure 12: Temperature distribution of solid materials.

The helium coolant temperature distribution in the core region is shown in Figure 13 and heights. And the fuel ball surface temperatures are shown in Figure 15. The temperature difference between fuel surface and helium coolant is shown in Figure 16.

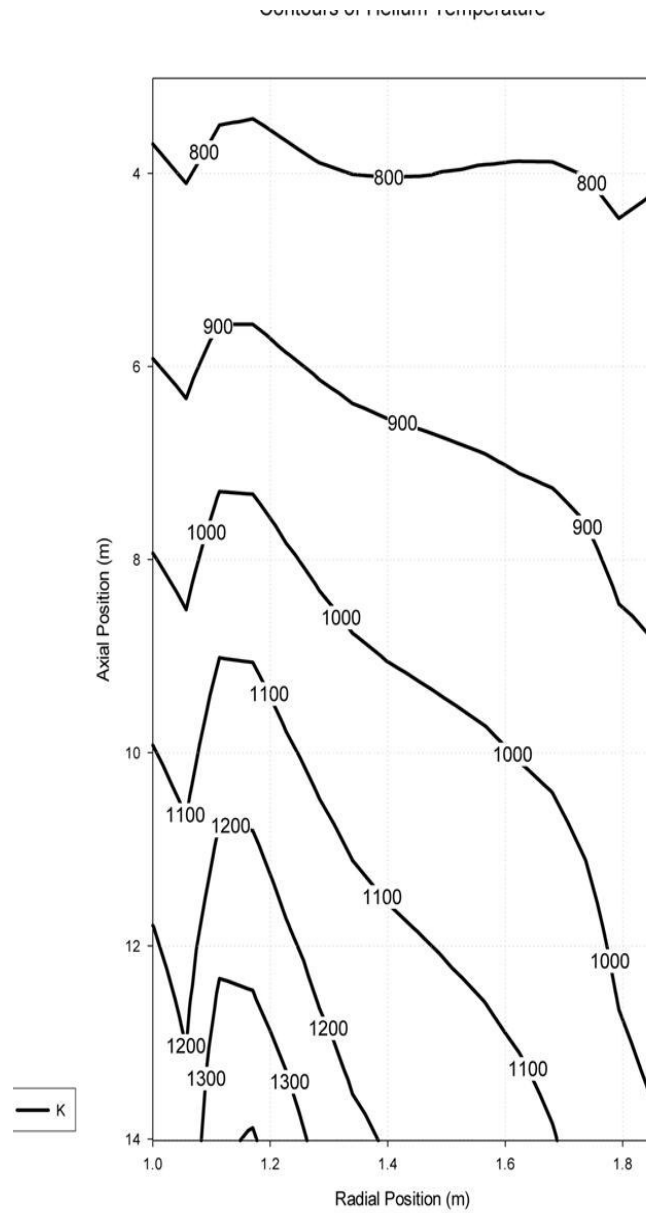
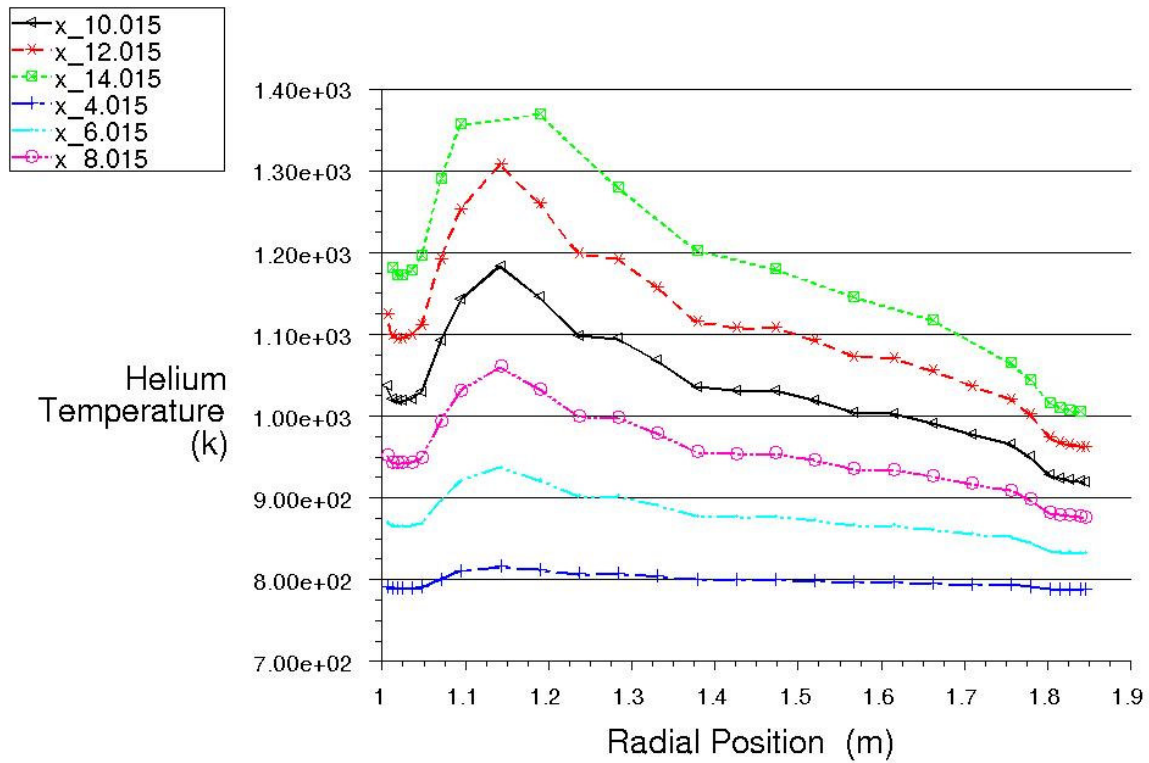


Figure 13: Coolant temperature in core zone.



Static Temperature

Aug 17, 2006
FLUENT 6.2 (axi, dp, segregated, ske)

Figure 14 Helium coolant temperatures at different core heights.

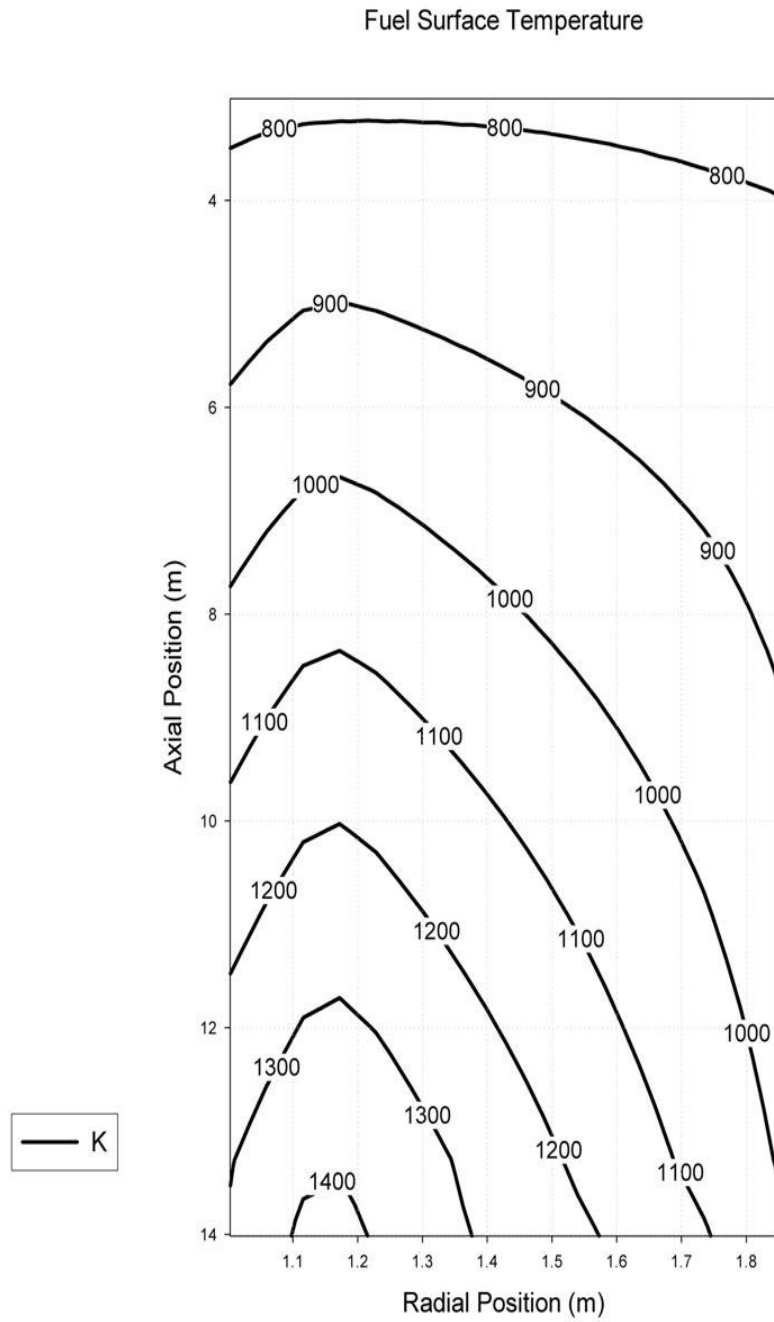


Figure 15 Fuel surface temperatures.

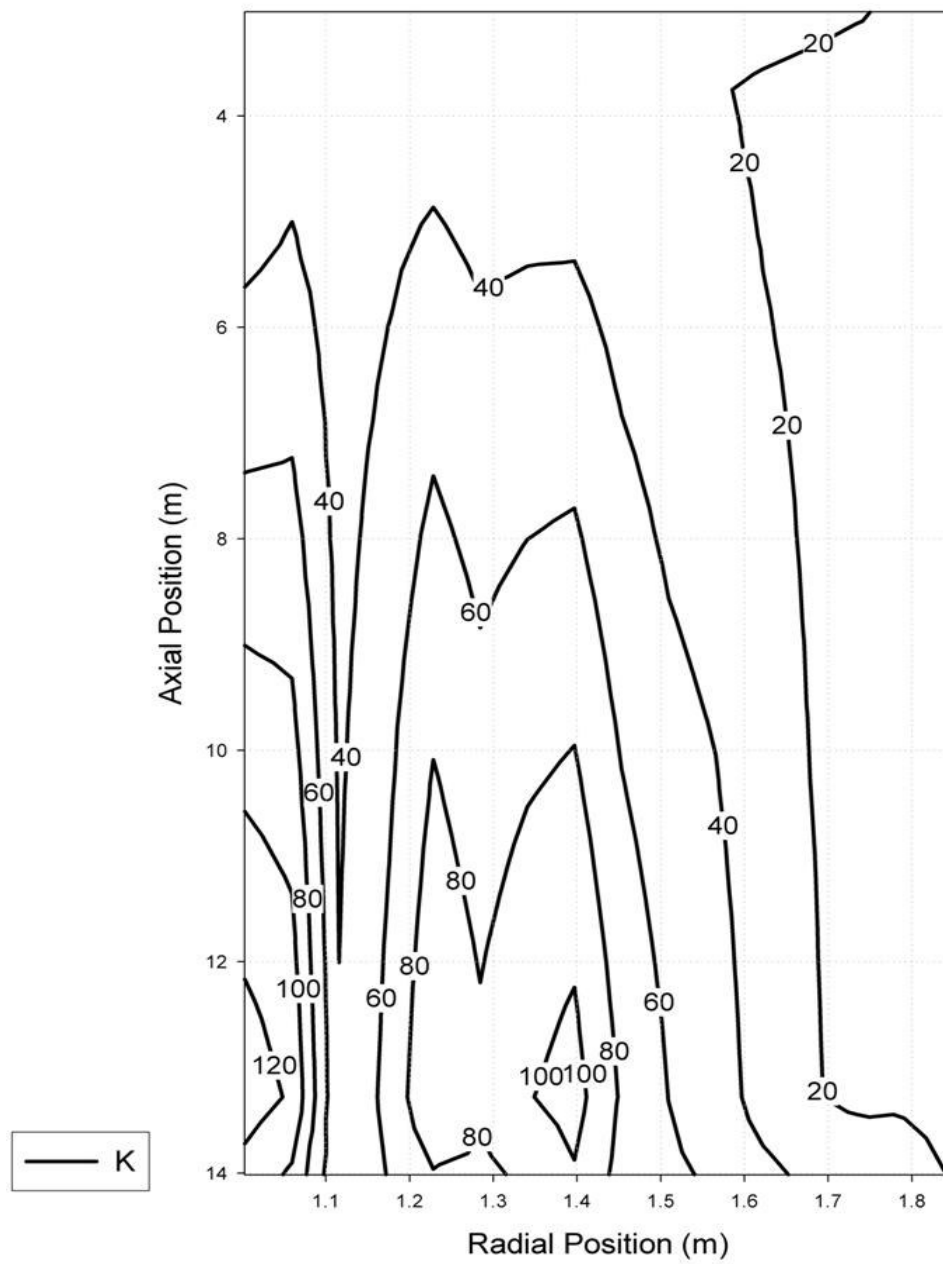


Figure 16 Temperature difference between fuel surface and helium coolant.

Contours of pebble-bed effective conductivity calculated according to Zehner-Schlunder are shown in Figure 17 and Figure 18.

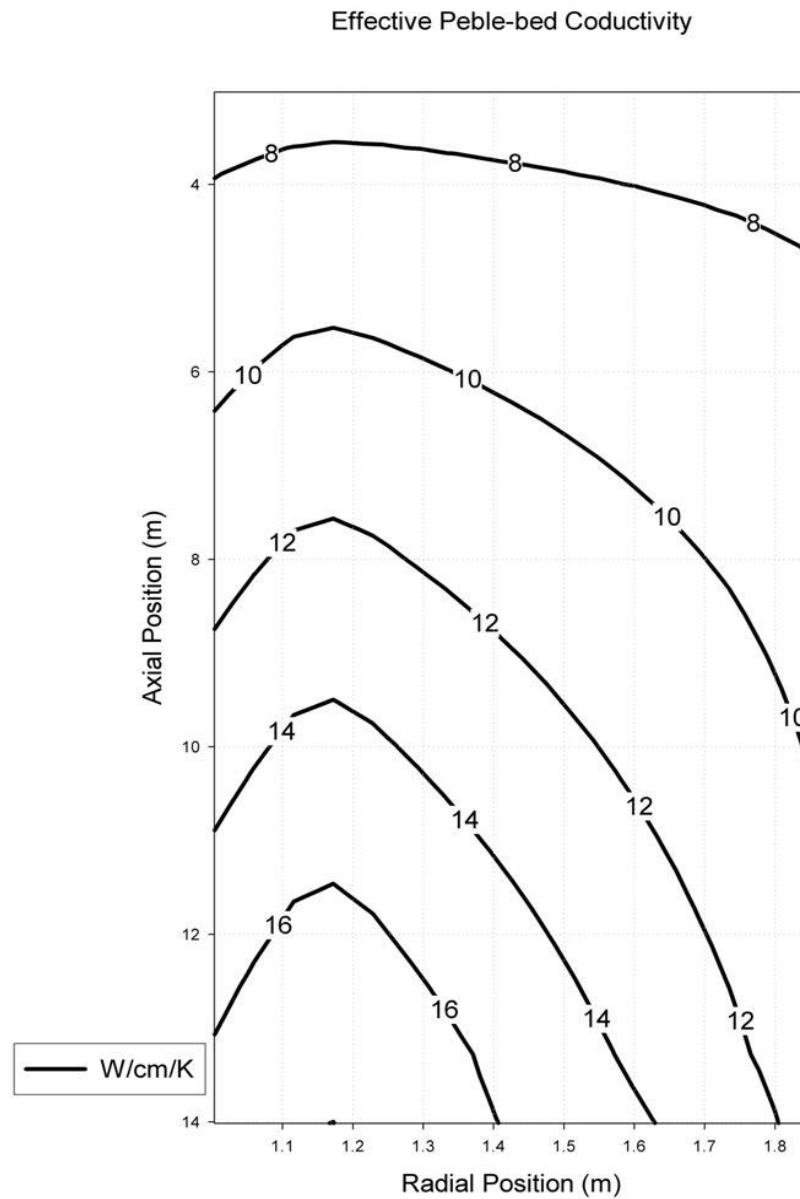
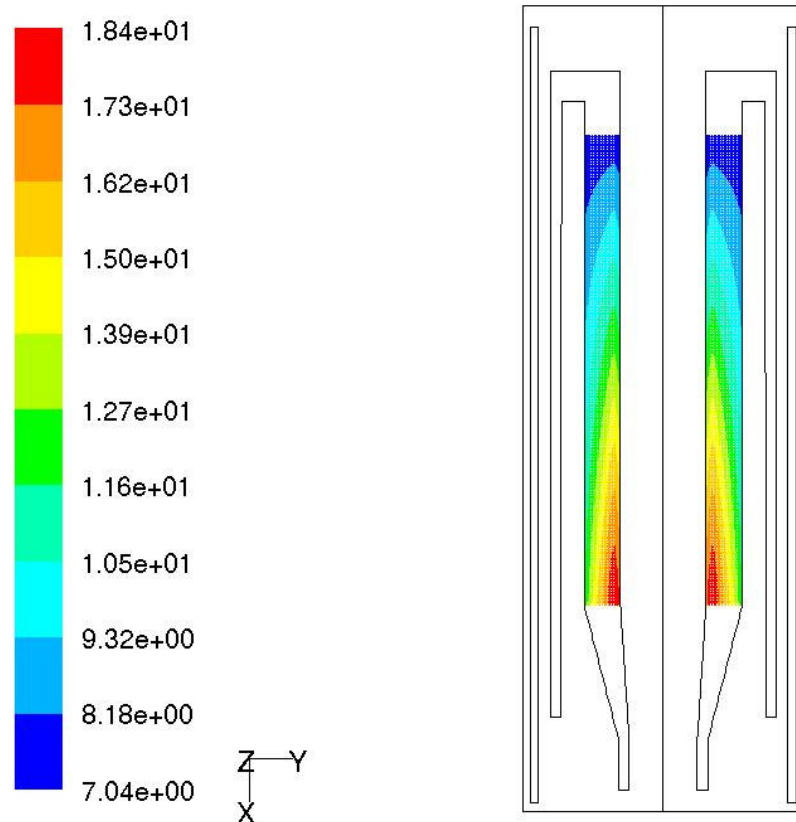


Figure 17: Effective pebble-bed conductivity (Zehner-Schlunder).

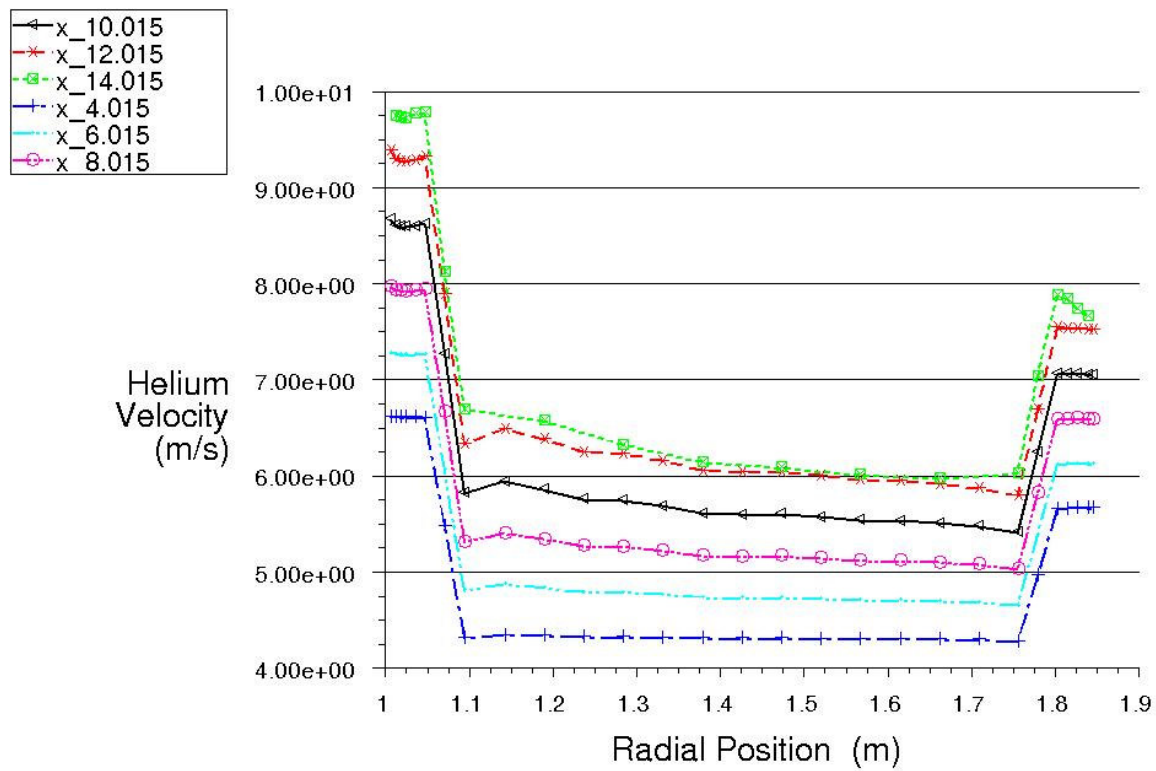


Contours of Diffusion Coef. of User Scalar 0

Aug 17, 2006
FLUENT 6.2 (axi, dp, segregated, ske)

Figure 18: Effective pebble-bed conductivity (W/cm/K).

The helium axial velocity profiles at various core heights are shown in Figure 19. Velocities are higher near reflector walls due to lower void fractions in these regions.



Axial Velocity

Aug 17, 2006
FLUENT 6.2 (axi, dp, segregated, ske)

Figure 19: Helium axial velocities at various core heights.

CHAPTER 3: MODELING INTERFACIAL AREA TRANSPORT EQATION WITH 3D TWO-FLUID COMPUTATIONAL MULTI-PHASE FLUID DYNAMICS MODEL

3.1 INTRODUCTION

Nearly all presently existing codes for modeling transient inhomogeneous two-phase flow are based on the two-fluid model using separate balance equations for mass, momentum and energy for two phases. These equations are derived from a combined volume and time averaging process starting with the local instantaneous conservation equations for the individual phases. However, much of the information on local flow conditions is in this averaging process and additional constitutive relations have to be provided as source terms describing the interfacial transport of mass, momentum and energy [8]. In two-fluid model, the interfacial area concentration a_i and the void fraction, α , are the key geometric parameters that characterize inter-phase transport processes and needs to be specified in the constitutive relations. Accurate estimation of these parameters is essential in accurate assessment of the given system. Most of the current thermal-hydraulics system analysis codes, such as RELAP5, TRAC, etc, implements flow-regime dependent correlations. This approach does not accurately represent the dynamical changes in the interfacial structure of the system. Therefore, improvements in the evaluation of the interfacial area will enhance the capability of the thermal-hydraulics analysis codes and assess better accuracy [9].

The interfacial area transport equation is based on the Boltzmann transport equation, where the equation was expressed as an integro-differential equation of the particle distribution function. This is considered in the next section.

3.2 POPULATION BALANCE MODELS

The particle number density function can be written in the form

$$f(V, \vec{x}, \vec{u}, t) dV d\vec{x} d\vec{u} dt \quad (43)$$

where f is the probable number of fluid particles in the range of volume size dV around V , in the spatial range $d\vec{x}$ about a position $\vec{x}$, having velocity range $d\vec{u}$ around $\vec{u}$, at a given time t .

The distribution function is often approximated as function of $f(V, x, t)$ for practical applications and a Boltzmann type equation for the fluid particle number density distribution function can be written as,

$$\frac{\partial f}{\partial t} + \nabla(f\vec{u}) + \frac{\partial}{\partial V} \left(f \frac{dV}{dt} \right) = \sum_j S_j \quad (44)$$

where $\vec{u}$ is the local particle velocity and S_j are the particle number source terms including particle coalescence, particle breakup and creation of particles due to new nucleation. These source terms are defined across a range of particle volume as follows [8]:

Particle break-up:

$$S_1(V) = \int_V^{V_{\max}} \beta(V', V) n(V') g(V') f(V') dV' \quad \text{gain} \quad (45)$$

$$S_2(V) = -g(V) f(V) \quad \text{loss} \quad (46)$$

Particle coalescence:

$$S_3(V) = \int_{V_{\min}}^{V/2} \lambda(V - V', V') h(V - V', V') f(V - V') f(V') dV' \quad \text{gain} \quad (47)$$

$$S_4(V) = \int_{V_{\min}}^{V_{\max} - V} \lambda(V, V') h(V, V') f(V') f(V) dV' \quad \text{loss} \quad (48)$$

where $\beta(V, V')$ is the distribution function of daughter particles produced upon break-up of parent particles having volume V' , $n(V')$ is the number of daughter particles

produced by break-up of a particle having a volume of V' , $g(V')$ is break-up frequency of a particle having the volume V' , $\lambda(V, V')$ is the coalescence efficiency once a collision has occurred between two particles with volumes V and V' , and $h(V, V')$ is the collision frequency of particles of volumes V and V' .

Some phenomenological models for the interaction terms S_j were summarized by references [10] and [11]. Such models require detailed information on the actual physical break-up and coalescence mechanisms. The detailed volume dependent particle number density transport equations would be too tedious and complicated for use in the field equations of most two-phase flow studies, where the primary focus is on the average fluid particle behavior [12]. On the other hand, the particle distribution function approach allows the definition of important flow parameters which can be related to empirically to measured data [8]. For example the number of particles is

$$n = \int_{V_{\min}}^{V_{\max}} f(V) dV \quad (49)$$

The volume fraction of particles in a given volume is

$$\alpha = \int_{V_{\min}}^{V_{\max}} Vf(V) dV \quad (50)$$

and the interfacial area concentration is

$$a_i = \int_{V_{\min}}^{V_{\max}} Af(V) dV \quad (51)$$

where A is the surface area of a particle having volume V .

The average particle velocity is written as

$$\vec{u}_n = \frac{\int_{V_{\min}}^{V_{\max}} \vec{u} f(V) dV}{\int_{V_{\min}}^{V_{\max}} f(V) dV} \quad (52)$$

a balance equation for the particle number density n can be defined as:

$$\frac{\partial n}{\partial t} + \nabla \cdot (n \vec{u}_n) = \sigma^N \quad (53)$$

where the source term σ^N for the particle number density is:

$$\sigma^N = \int_{V_{\min}}^{V_{\max}} \left(\sum_j S_j \right) dV \quad (54)$$

An approach for solving such two-phase flows is the grouping of particle sizes and solving the corresponding balance and conservation equations describing the bubble number density, mass, momentum and energy for each group. These methods also require detailed models for physical break-up and coalescence mechanisms for large number of particle-size groups and are computationally inefficient. Nevertheless, some commercial CFD codes, CFX and FLUENT 6.3, implements such multi-group bubble-size methods. Hence, in this thesis a better approach, namely, interfacial area transport equation, which was suggested by Ishii [13], is coded to be used in FLUENT 6.2.16.

3.3 INTERFACIAL AREA CONCENTRATION TRANSPORT EQUATION

The preliminary formulation of interfacial area transport equation (IATE) for bubbly flow was given by reference [12]. The formulation presented in this study was given in reference [9].

To formulate the IATE, first Equation (44) is multiplied by the average interfacial area A_i of particles with volume V then integrated over the volume of all particle sizes obtaining

$$\frac{\partial a_i}{\partial t} + \nabla(a_i \bar{u}_i) - \left(\frac{\dot{V}}{V} \right) \int_{V_{\min}}^{V_{\max}} f V dA_i = \int_{V_{\min}}^{V_{\max}} \left[\sum_j S_j + S_{ph} \right] A_i dV \quad (55)$$

where the interfacial velocity u_i is given by

$$\bar{u}_i(x, t) = \frac{\int_{V_{\min}}^{V_{\max}} f(V, x, t) A_i(V) \bar{u}(V, x, t) dV}{\int_{V_{\min}}^{V_{\max}} f(V, x, t) A_i(V) dV} \quad (56)$$

In deriving Equation(55), it is assumed that the time rate of change in the relative particle volume is independent of its volume. The volume source can be expressed as

$$\frac{1}{V} \frac{dV}{dt} = \frac{1}{\alpha} \left[\frac{\partial \alpha}{\partial t} + \nabla \alpha \bar{u}_g - \eta_{ph} \right] \quad (57)$$

The volume-equivalent and surface-equivalent diameters of fluid particles with volume V and surface A_i are defined by

$$V = \frac{\pi}{6} D_e^3 \quad (58)$$

and

$$A_i = \pi D_s^2 \quad (59)$$

Then, the interfacial transport equation is given by

$$\frac{\partial a_i}{\partial t} + \nabla(a_i \bar{u}_i) - \frac{2}{3} \left(\frac{a_i}{\alpha} \right) \left(\frac{\partial \alpha}{\partial t} + \nabla \alpha \bar{u}_g - \eta_{ph} \right) = \int_{V_{\min}}^{V_{\max}} \left[\sum_j S_j + S_{ph} \right] A_i dV \quad (60)$$

In Equation(60), the source and sink terms at the right hand side of the equation should be specified by the constitutive relations and they can be written as

$$\int_{V_{\min}}^{V_{\max}} \sum_j S_j A_i dV = \sum_j \phi_j \quad (61)$$

where ϕ_i are the interfacial area concentration source and sink terms. Furthermore, since the particle number density can be defined in terms of a_i and α , the source and sink terms for a_i can be specified through the mechanistic models developed for particle number density. Firstly, particle number density source terms R_i are defined as

$$\int_{V_{\min}}^{V_{\max}} \sum_j S_j dV = \sum_j R_j \quad (62)$$

The geometric relation between bubble number density n and a_i and α is can be defined as

$$n = \psi \left(\frac{a_i^3}{\alpha^2} \right) \quad (63)$$

where ψ is a factor depending on the shape of the bubbles and it is equal to $1/(36\pi)$ for spherical bubbles. It is defined as

$$\psi = \frac{1}{36} \left(\frac{D_{sm}}{D_e} \right)^3 \quad (64)$$

where D_{sm} is the Sauter mean diameter defined by

$$D_{sm} = \frac{6\alpha}{a_i} \quad (65)$$

Thus, ϕ_i can be rewritten as

$$\phi_i = \frac{1}{3\psi} \left(\frac{\alpha}{a_i} \right)^2 R_j \quad (66)$$

Similarly, the nucleation source ϕ_{ph} for interfacial area concentration is rewritten as

$$\phi_{ph} = \pi D_{bc}^2 R_{ph} \quad (67)$$

where D_{bc} is the critical bubble size, which should be determined depending on the given nucleation process, namely, critical cavity size for the bulk boiling or condensation process, and bubble departure size for the wall nucleation. The R_{ph} should be modeled independently based on the given phase changing mechanism [9].

Thus, using the constitutive relations given above, the interfacial area transport equation can be rewritten as

$$\frac{\partial a_i}{\partial t} + \nabla (a_i \vec{u}_i) = \frac{2}{3} \left(\frac{a_i}{\alpha} \right) \left(\frac{\partial \alpha}{\partial t} + \nabla \alpha \vec{u}_g - \eta_{ph} \right) + \frac{1}{3\psi} \left(\frac{\alpha}{a_i} \right)^2 \sum_j R_j + \pi D_{bc}^2 R_{ph} \quad (68)$$

or in terms of ϕ_i

$$\frac{\partial a_i}{\partial t} + \nabla (a_i \vec{u}_i) = \frac{2}{3} \left(\frac{a_i}{\alpha} \right) \left(\frac{\partial \alpha}{\partial t} + \nabla \alpha \vec{u}_g - \eta_{ph} \right) + \sum_j \phi_j + \phi_{ph} \quad (69)$$

3.4 MAJOR BUBBLE INTERACTION MECHANISMS

The major particle interaction mechanisms, which may attribute to the particle coalescence or disintegration, can be summarized as [9]

- Random collision (RC): coalescence driven by turbulent eddies.
- Wake entrainment (WE): bubble coalescence due to acceleration of the following bubbles in the wake of a preceding bubble.
- Turbulent impact (TI): bubble disintegration upon impact of turbulent eddies.

- Shearing-off (SO): small bubbles shearing off around the base rim of a large bubble.
- Surface instability (SI): breakup of a large bubble due to surface instability at the interface.
- Rise velocity (RV): collision due to the difference in the bubble rise velocity.
- Laminar shear (LS): break-up due to the laminar shear in viscous fluid.
- Velocity gradient (VG): collision due to the velocity gradient

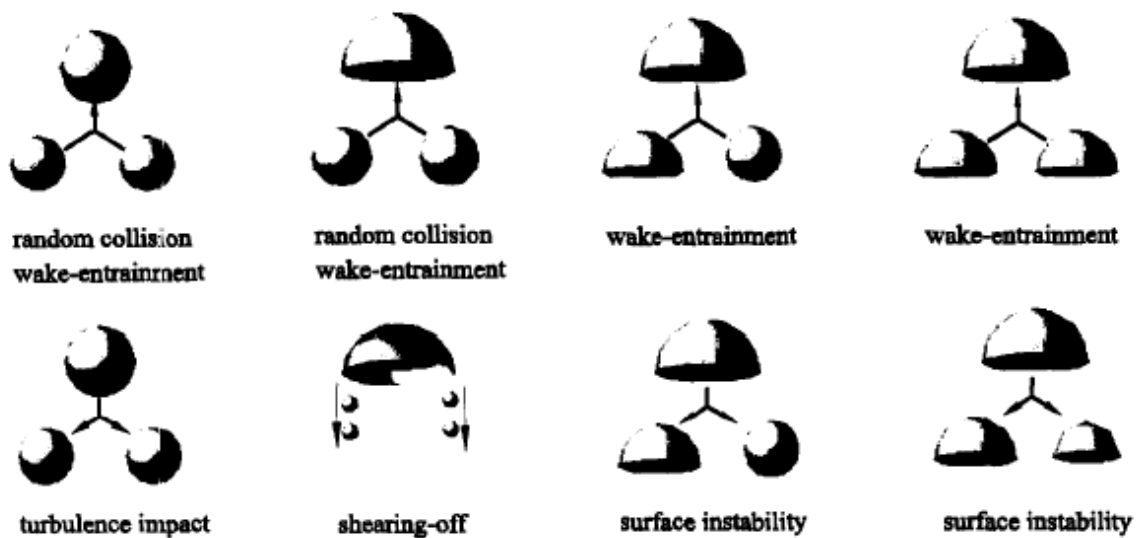


Figure 20: Mechanisms of bubble coalescence and break-up [12].

The major contribution to in the a_i transport can be attributed to three mechanisms. They are the turbulent impact (TI), the random collision (RC), and the wake entrainment (WE). The LS and VG may be neglected in the source and sink terms. Furthermore, the RV can be considered as a part of WE and can be dropped for simplicity of the model [9].

Detailed mechanistic models for the important interactions were presented in reference [14].

3.4.1 Coalescence due to Turbulence Induced Random Collision

Random collision between bubbles driven by turbulent eddies is one of the major coalescence mechanisms in gas–liquid two-phase mixtures. The collision among multiple bubbles (more than two) is very complex and considered as a low-probability event compared to two-bubble collision. Therefore, for simplicity, a binary collision is assumed. The net sink term for a_i due to this interaction term is defined as [14]

$$\phi_{RC} = -0.17 C_{RC} \frac{\varepsilon^{1/3} \alpha^{1/3} a_i^{5/3}}{\alpha_{\max}^{1/3} (\alpha_{\max}^{1/3} - \alpha^{1/3})} \times \left[1 - \exp \left(-C_{RC1} \frac{\alpha_{\max}^{1/3} \alpha^{1/3}}{\alpha_{\max}^{1/3} - \alpha^{1/3}} \right) \right] \quad (70)$$

where ε is the turbulent kinetic energy dissipation rate per unit mixture mass, C_{RC1} accounts for the proportionality between the average size of eddies and the mean distance between bubbles and can be approximated for bubbly flow as $C_{RC1} = 3$ [16]. C_{RC} is an adjustable coefficient and should be determined based on experimental data. $\alpha_{\max}$ is the dense packing limit of void fraction when the coalescence approaches infinity. A rational choice for $\alpha_{\max}$ is approximately 0.8 occurring at the transition point from slug to annular flow [12].

3.4.2 Coalescence Through Entrainment in the Wake Region of a Preceding Bubble

Another important bubble coalescence mechanism is the entrainment of the bubbles following in the wake of a preceding bubble. When bubbles enter the critical wake region of a preceding bubble, they accelerate and coalesce with the preceding bubble [14]. Similar to the calculation in RC, the net sink term of a_i due to the WE, can be obtained as [15]

$$\phi_{WE} = -0.27 C_{WE} u_r C_D^{1/3} a_i^2 \quad (71)$$

where C_{WE} is an adjustable experimental constant. u_r is the terminal velocity of the bubble relative to the liquid motion. C_d is the drag coefficient for bubbles.

3.4.3 Bubble Break-up due to Turbulent Impact

Turbulence is one of the major mechanisms causing bubbles to break-up in gas liquid flow. The net source rate for a_i can be obtained as [15]

$$\phi_{TI} = 0.12 C_{TI} \varepsilon^{1/3} (1 - \alpha) \frac{a_i^{5/3}}{\alpha^{2/3}} \times \exp\left(-\frac{We_{c,TI}}{We}\right) \sqrt{1 - \frac{We_{c,TI}}{We}} \quad (72)$$

Again, C_{TI} is an adjustable parameter and should be evaluated experimentally. $We_{c,TI}$ is the critical Weber number and also should be evaluated with experimental data. The reported value of this number varies in a wide range [17]. For an air-water system, suggested value for $We_{c,TI}$ is 2.3 [10]. We is the Weber number and defined as

$$We = \frac{\rho_f \bar{u}_t^2 D_b}{\sigma} \quad (73)$$

where ρ_f is the density of continuous liquid, $\sqrt{\bar{u}_t^2}$ is the eddy velocity, and σ is the surface tension of continuous liquid phase.

3.5 CONSERVATION EQUATIONS FOR A TWO-FLUID (EULER-EULER) MODEL WITH INTERFACIAL AREA TRANSPORT MODELING

All the conservation equations for a full computational multi-phase fluid dynamic model of a two-fluid model with IATE are defined in the next sections. In this part a gas-liquid system is considered and all parameters are defined according to this assumption. The main continuous phase is assumed to be the liquid phase denoted by subscript l and the second phase is assumed as the gas (bubbles) phase denoted by subscript b .

3.5.1 Volume Fractions

Phasic volume fractions α represent the space occupied by each phase, and the laws of conservation of mass and momentum are satisfied by each phase individually [18].

By the definition volume fractions of volume fractions the volume phases are defined as

$$V_l = \int_V \alpha_l dV \quad V_b = \int_V \alpha_b dV \quad (74)$$

where

$$\alpha_l + \alpha_b = 1 \quad (75)$$

The effective densities of the phases are defined as

$$\hat{\rho}_l = \alpha_l \rho_l \quad \text{and} \quad \rho_b = \alpha_b \rho_b \quad (76)$$

where ρ is the physical density of corresponding phase.

3.5.2 Conservation of Mass

The continuity equation for separate phases ($i = l, b$) is defined by

$$\frac{\partial}{\partial t}(\alpha_i \rho_i) + \nabla \cdot (\alpha_i \rho_i \vec{u}_i) = \dot{m}_i \quad (77)$$

where $\dot{m}_i$ is the mass source for phase i from other phases and $\sum_{i=l,b} \dot{m}_i = 0$.

3.5.3 Conservation of Momentum

The momentum conservation equation for phases $i = l, b$ is defined as

$$\begin{aligned} \frac{\partial}{\partial t}(\alpha_i \rho_i \vec{u}_i) + \nabla \cdot (\alpha_i \rho_i \vec{u}_i \vec{u}_i) + \nabla \cdot (\alpha_i [\bar{\vec{\tau}}_i + \bar{\vec{\tau}}_i^T]) &= \alpha_i \nabla P + \alpha_i \rho_i \vec{g} \\ &+ \vec{F}_{\text{int},i} + \vec{F}_{\text{lift},i} + \vec{F}_{\text{vm},i} + \vec{F}_{\text{w},i} \end{aligned} \quad (78)$$

P is the pressure shared by all phases, $\vec{F}_{\text{int}}$ is interaction force density between all phases, $\vec{F}_{\text{lift}}$ is lift force density, $\vec{F}_{\text{vm}}$ is virtual mass force density and $\vec{F}_{\text{w}}$ is the wall force density.

The inter-phase force density $\vec{F}_{\text{int}}$ can be defined in terms of interfacial area concentration as

$$\vec{F}_{\text{int},b} = -\vec{F}_{\text{int},l} = -\frac{1}{8}\rho_l C_D |\vec{u}_r| \vec{u}_r a_i \quad (79)$$

where $\vec{u}_r$ is the relative velocity defined as

$$\vec{u}_r = \vec{u}_l - \vec{u}_b \quad (80)$$

For the drag coefficient C_D the model of Schiller and Naumann may be used which is the default model in FLUENT CFD software [18,20];

$$C_D = \begin{cases} 24(1 + 0.15 \text{Re}^{0.687}) / \text{Re} & \text{Re} \leq 1000 \\ 0.44 & \text{Re} > 1000 \end{cases} \quad (81)$$

where the Reynolds number defined as

$$\text{Re} = \frac{\rho_l |\vec{u}_l - \vec{u}_b| D_b}{\mu_l} \quad (82)$$

Small particles experience lift force $\vec{F}_{\text{lift}}$ perpendicular to direction of flow due to velocity gradients in the continuous liquid phase. The lift force acting on a bubble can be computed as [18, 21]

$$\vec{F}_{\text{lift}} = -0.5\rho_l\alpha_b\vec{u}_r \times (\nabla \times \vec{u}_l) \quad (83)$$

where $\vec{F}_{\text{lift},b} = -\vec{F}_{\text{lift},l}$.

The virtual mass force $\vec{F}_{vm}$ is a result from the acceleration of the liquid caused by the motion of the bubble. Inertia of the liquid mass exerts a virtual mass force on the accelerating bubbles which can be defined by [18, 21]

$$\vec{F}_{vm} = 0.5\alpha_b\rho_l\left(\frac{d_l\vec{u}_l}{dt}-\frac{d_b\vec{u}_b}{dt}\right) \quad (84)$$

The term $\frac{d_i}{dt}$ denotes the phase time derivative in the form

$$\frac{d_i}{dt} = \frac{\partial\phi}{\partial t} + (\vec{u}_i \cdot \nabla)\phi \quad (85)$$

where $\vec{F}_{vm,b} = -\vec{F}_{vm,l}$.

The wall force $\vec{F}_w$ moves bubbles away from the wall and it was proposed by Antal et al [22]. The wall force density on the bubbles phases may be given by [22]

$$F_{w,b} = -\frac{2\alpha_b\rho_l|\vec{u}_{ax}|^2}{D_b}\left(C_{w1} + C_{w2}\frac{D_b}{2y}\right)\delta_w\vec{n}_w \quad (86)$$

where y is the distance between the bubble centre and the wall and calculated as

$$y = (\vec{x} - \vec{x}_w) \cdot \vec{n}_w \quad (87)$$

$\vec{n}_w$ is a unit vector which is normal to the wall of the conduit, and,

$$\vec{u}_{ax} = \vec{u}_r - [\vec{n}_w \cdot \vec{u}_r]\vec{n}_w \quad (88)$$

$$\begin{aligned} C_{w1} &= -0.104 - 0.06|\vec{u}_r| & C_{w2} &= 0.147 \\ C_{wb} &= 1.5 \end{aligned} \quad (89)$$

$$\delta_w = \begin{cases} 1.0 & y > D_b \\ 0.0 & \text{otherwise} \end{cases} \quad (90)$$

3.5.4 Conservation of Energy

In this study, only the isothermal cases are considered, however the energy conservation equation for a two fluid model can be written in internal energies h_i for each phase $i = b, l$ as [18]

$$\frac{\partial}{\partial t}(\alpha_i \rho_i h_i) \nabla \cdot (\alpha_i \rho_i \vec{u}_i h_i) = -\alpha_i \frac{\partial p_i}{\partial t} + \vec{\tau}_i : \nabla \vec{u}_i - \nabla \cdot \vec{q}_i + \sum_j (Q_{ji} + \dot{m}_{ji} h_{ji} - \dot{m}_{ij} h_{ij}) \quad (91)$$

where h is the specific enthalpy, $\vec{q}$ is the heat flux, Q is the intensity of heat exchange between the phases, and h_{ij} is the inter-phase enthalpy.

3.5.5 Interfacial Area Concentration Transport Equation

IATE was considered in section 2.3 and was defined in Equation(69). The coupling of the model equations and IATE can be achieved by writing all the constitutive relations in terms of interfacial area concentration and volume fractions. For instance, assuming spherical bubbles, Equation (65) can be written as

$$D_b = \frac{6\alpha_b}{a_i} \quad (92)$$

which may be used in all equations defined before.

3.6 IMPLEMENTAION OF THE MODEL IN FLUENT CODE

The IATE model implementation is succeeded by improving the Euler-Euler multiphase model in FLUENT by means of C macros and the header files provided with FLUENT code (see section 1.3.2). The model equations were given in section 2.5. Except the wall force density, other force densities for the momentum equations are already included in FLUENT. This extra term is included by means of the UDF utilities of FLUENT (see Appendix 2).

Interfacial area concentration transport equation is implemented by solving an extra scalar equation (see section 1.3). The bubble interaction mechanism is also modeled in the source code (see Appendix 2).

The adjustable model constants should be determined by finding the values that yield the best agreement with the available experimental data sets. In this study, the model constants from the paper by Ishii *et al* [9] were considered. The model coefficients used in the present study are listed below.

$$\begin{aligned} C_{RC1} &= 3.0, & C_{RC} &= 0.003, \\ \alpha_{\max} &= 0.8, & C_{WE} &= 0.042, \\ C_{TI} &= 0.026, & We_{c,TI} &= 8.0 \end{aligned} \quad (93)$$

3.6.1 Case Study

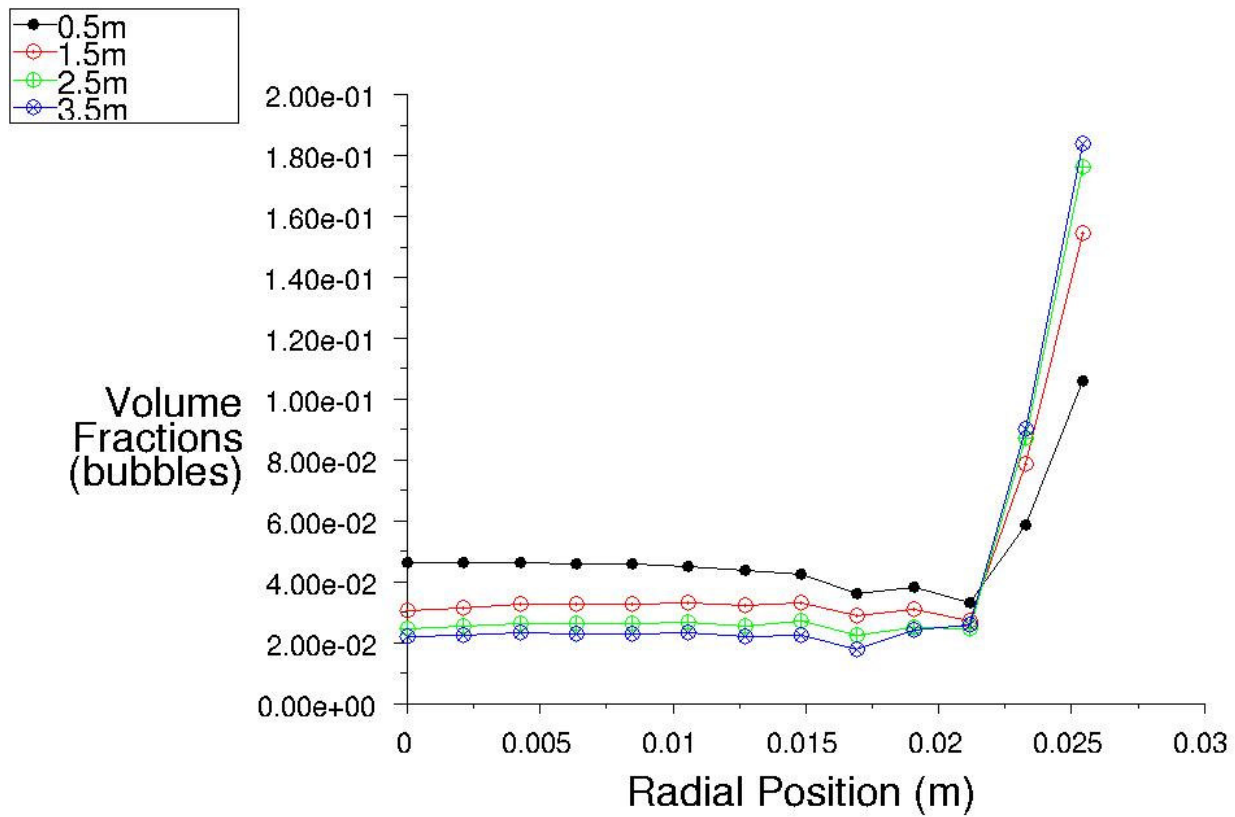
For a case study of the model, one of the geometry and the upward flow conditions described in reference [25] was considered. The geometry of the test model includes a vertical cylinder with an inner diameter of 50.8 mm and height of 3.8 m. Superficial velocities of water and air bubbles at the inlet (from bottom) are 0.986 m/s and 0.0473 m/s respectively. The bubble volume fraction and mean diameter at the inlet are 5.12% and 1 mm.

The mixture $k - \varepsilon$ turbulence model [18] is used during the solution process due to bad convergence behavior of other models available in FLUENT for this case. To improve the convergence behavior, solution was started with the mixture model available in FLUENT to obtain an initial solution for the Eulerian model. The second order upwind methods were chosen for discretization of all solution variables.

3.6.2 Simulation Results

The radial profiles bubble volume fractions at heights of 0.5, 1.5, 2.5 and 3.5 meters are given below in Figure 21. As seen, volume fraction of bubbles increases and makes a peak near the pipe walls as the flow develops. This is expected due to lift forces. When

the lift force was ignored in solution, the peaks formed in the middle of the pipe instead of near pipe walls.

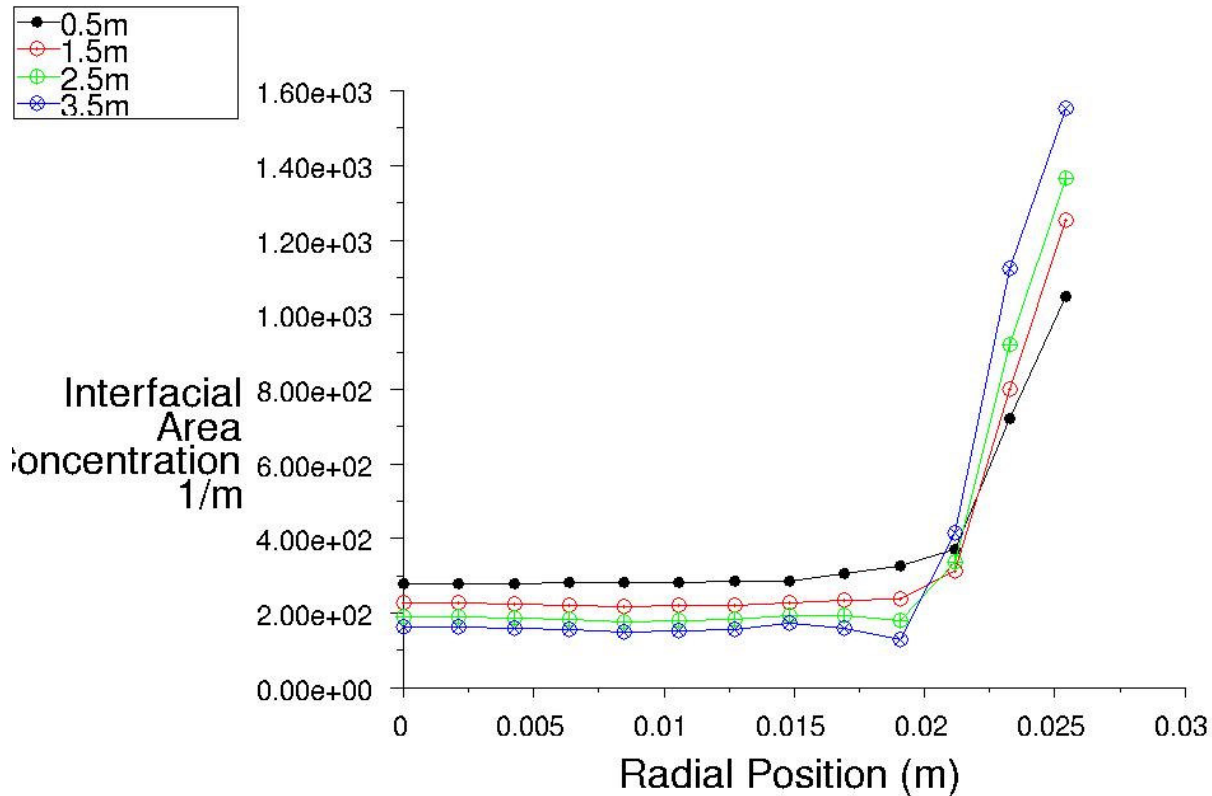


Volume fraction (phase-2)

Aug 30, 2006
FLUENT 6.2 (axi, dp, segregated, eulerian, ske)

Figure 21: Radial volume fraction profiles of air bubbles at different heights.

Interfacial area concentration profiles at the heights of 0.5 m, 1.5 m, 2.5 m and 3.5 m are shown in Figure 22. The interfacial area profiles are similar to the volume fraction profiles. This can be expected as the interfacial area concentration is proportional to the volume fraction of bubbles and inversely proportional to the bubble Sauter mean diameter, [see Equation (65)].

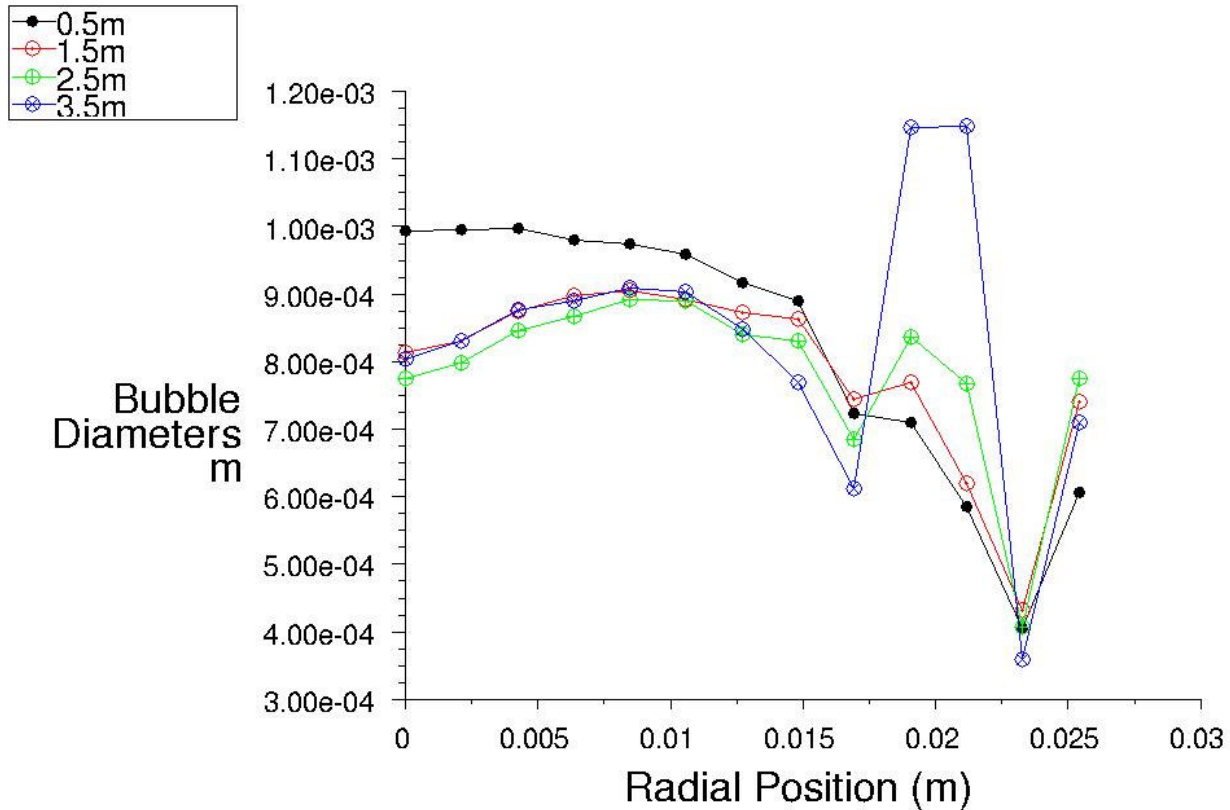


User Scalar 0 (mixture)

Aug 30, 2006
FLUENT 6.2 (axi, dp, segregated, eulerian, ske)

Figure 22: Interfacial area concentration profiles of air bubbles at different heights.

Diameter profiles of air bubbles are shown in Figure 23. As the flow develops, higher diameter values are seen and bigger bubbles are moved laterally in the pipe. This movement is mainly due to lift forces and the wall forces may prevent the bubbles from further moving to the side walls. This may be seen well in the profile at 3.5 m height.



User Memory 3 (mixture)

Aug 30, 2006
FLUENT 6.2 (axi, dp, segregated, eulerian, ske)

Figure 23: Diameter profiles of air bubbles at different heights.

The calculations were repeated without coupling Euler model with IATE for the purpose of comparison with the standard two-fluid model (Euler), in which the bubble interactions are ignored and the bubble size is constant. In Figure 24, Figure 25, Figure 26 and Figure 27, volume fraction and interfacial area concentration profiles are graphed at 2.5 m and 3.5 m positions using the results from both models.

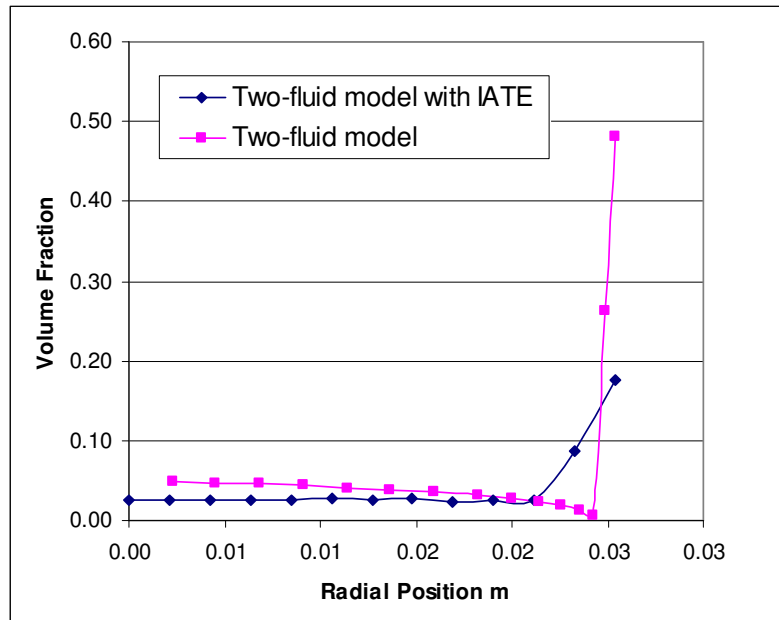


Figure 24: Volume Fractions of bubbles at 2.5 m height.

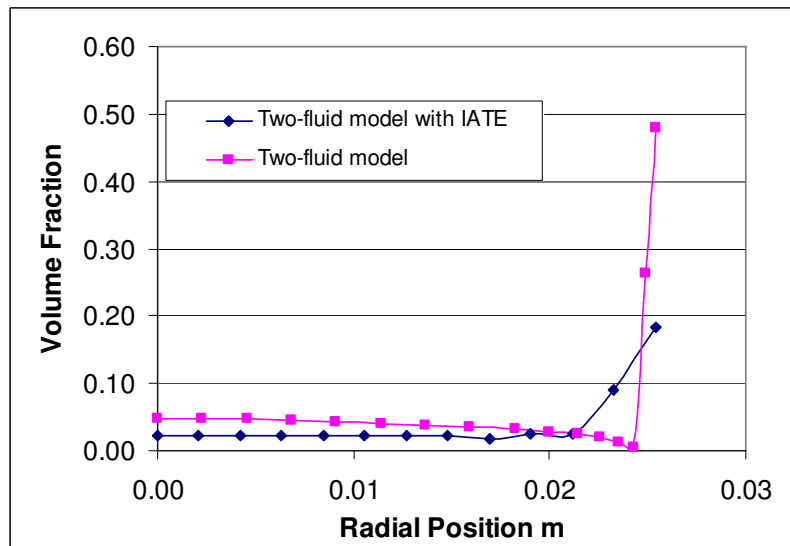


Figure 25: Volume Fractions of bubbles at 3.5 m height.

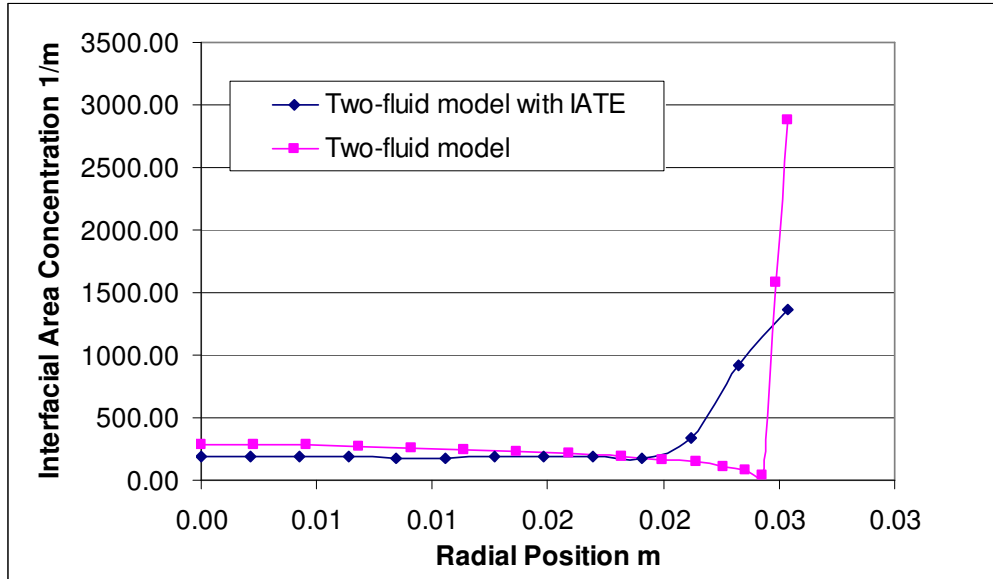


Figure 26: Interfacial area concentrations at 2.5m height.

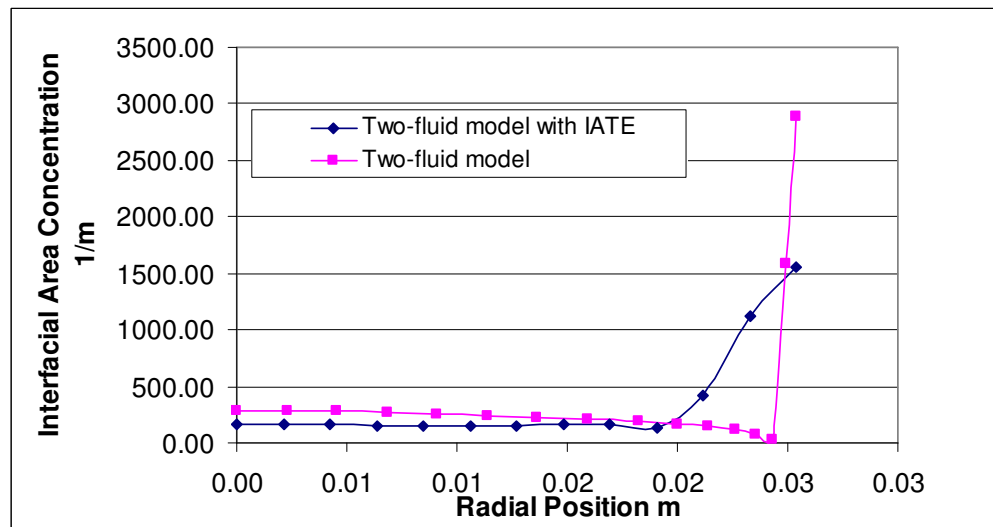


Figure 27: Interfacial area concentrations at 3.5m height.

CHAPTER 4: CONCLUSIONS

This thesis consists of two studies. In the first part of the thesis, thermal hydraulics of pebble-bed high temperature gas cooled reactors was considered and a porous medium approach was modeled in FLUENT code. The model was assessed for PBMR-400 reactor concept. In the second part, Multiphase Computational Fluid Dynamics (MCFD) is considered and for this purpose Interfacial Area Transport Equations (IATE) are modeled and implemented in FLUENT CFD code.

In Chapter 2, a detailed new porous model was developed and successfully assessed in FLUENT CFD software. Significant modelling improvement is achieved for pebble-bed reactor thermal-hydraulics by this new model.

FLUENT CFD software is highly customizable by means of user defined scalar (UDS) transport equations and user defined functions (UDF) which are coded in C programming language. This feature of the code allows experienced CFD users to add new physical and mathematical models to the software.

A special turbulence formulation for porous mediums is a controversial issue and it is suggested as a future study. In this study, the semi empirical formulations for momentum sources, Equation(27), includes turbulence source terms and the core zone was modelled as a laminar zone. This approach is also controversial and should be validated against experimental results.

In Chapter 3, a detailed two-fluid model coupled with Interfacial Area Transport Equations was introduced and successfully implemented in FLUENT CFD code. The simulation results seem physical, however this method should be validated by experimental studies and fine tuning of the adjustable experimental parameters is needed.

An alternative method is the population balance method as implemented in CFX and FLUENT 6.3 codes. However, these methods consist of grouping of bubble sizes and

corresponding equations as a result they are computationally very expensive and require more detailed models for inter-group and intra-group interaction mechanisms.

BIBLIOGRAPHY

- [1] Zehner, P. and Schluender EU., 1970, Thermal conductivity of granular materials at moderate temperatures, Chemie-Ingenieur-Technik, 42 (14), pp. 933-941.
- [2] Sicherheitstechnische Regel des KTA, Auslegung der Reaktorkerne von Gasgekühlten, Hochtemperaturreaktoren, Teil 3: Druckverlust in Kugelhaufen, KTA-Dok.-Nr. 3102/78/1
- [3] R.F. Benenati and C.B. Brosilow, 1962, Void Fraction Distribution in Beds of Spheres, A.I.Ch.E.J. 8 no. 6, 233-236 (1962).
- [4] Sue Ion, David Nicholls, Regis Matzie, Dieter Matzner, 2003, Pebble Bed Modular Reactor The First Generation IV Reactor To Be Constructed, World Nuclear Association Annual Symposium 3-5 September 2003-London.
- [5] CRP-5 Benchmark Definition: PBMR-400 Neutronics and Thermal-hydraulics Description, Project: PBMR Demonstration Module, Document Number: 022017, Document Source: MR300, Document Type: 3330, Revision: 1, 2003/12/04.
- [6] Chung T.J., Computational Fluid Dynamics, 2002, Cambridge University Press, 218-239.
- [7] Barthand Timothy, 2004, Ohlberger Mario, Finite volume methods: Foundation and analysis, John Wiley & Sons.
- [8] Worth, B., Städtke, H., Franchello, G., 2003, Modeling of Interfacial Area Transport In The JRC "ATFM" Code, ASTAR International Workshop on "Advanced Numerical Methods for Multidimensional Simulation of Two-phase Flow", September 15-16, 2003, Garching, Germany.
- [9] Kim, S., Sun X., Ishii M., Beus S. G. and Lincoln F., 2002, Interfacial area transport and evaluation of source and sink terms for confined air-water bubbly flow., Nuclear Engineering and Design 219 (2002) 61-75.
- [10] Prince, M. J. And Blanch, H. W., 1990, Bubble coalescence and break-up in air-sparged bubble columns. AIChE Journal, 36, 1485.
- [11] Lafi, A. Y. And Reyes, Jr, J. N., 1991, Phenomenological models for fluid particle coalescence and breakage. Technical report, OSU-NE-9120, Department of Nuclear Engineering, Oregon State University, Corvallis, Oregon.

- [12] Wu, Q., Kim, S., Ishii, M. and Beus S. G., 1998, One-group interfacial area transport in vertical bubbly flow, *Int. J. Heat Mass Transfer*. Vol. 41, Nos 8-9, pp. 1103-1112.
- [13] Ishii, M., 1975. *Thermo-Fluid Dynamic Theory of Two-Phase Flow*. Eyrolles, Paris.
- [14] Sun, X., Kim, S., Ishii, M. and Beus, S., 2004, Modeling of bubble coalescence and disintegration in confined upward two-phase flow, *Nuclear Engineering and Design* 230 (2004) 3-26.
- [15] Sun, X., 2001. Two-group interfacial area transport equation for a confined test section. Ph. D. thesis, Purdue University, West Lafayette, IN.
- [16] Kim, S., 1999. Interfacial area transport equation and measurement of local interfacial characteristics. Ph. D. thesis, Purdue University, West Lafayette, IN.
- [17] Sevik, M. and Park, S. H., 1973, The splitting of drops and bubbles by turbulent fluid flow. *Journal of Fluid Engineering*, 95, 53.
- [18] FLUENT 6.2.16 CFD software user guide.
- [19] Richard, T., Lahey, Jr., 2005, The simulation of multidimensional multiphase flows, *Nuclear Engineering and Design* 235 (2005) 1043-1060.
- [20] Schiller, L., and Naumann, Z., *Z. Ver. Deutsch. Ing.*, 77:318, 1935.
- [21] Drew, D. A., and Lahey, R. T., 1993, In *Particulate Two-Phase Flow*, pages 509-566. Butterworth-Heinemann, Boston.
- [22] Antal, S. P., Lahey, R. T. Jr., and Flaherty, J. E., 1991, Analysis of phase distribution in fully developed laminar bubbly two-phase flow,' *Int. J. Multiphase Flow*, 17, 635–652.
- [23] El-Wakil, M. M., *Powerplant Technology*, 1984, McGraw-Hill, Inc. 362-390.
- [24] Binkele, L., 1974, Ein Verfahren zur Bestimmung der Wärmeleitfähigkeit von neutronbestrahlten Graphiten bei Temperaturen zwischen 50 und 1 000 °C; JÜL-1096-RW, August.
- [25] Hibiki, T., Goda, H., Kim, S., Ishii, M. and Uhle, J., 2004, Structure of vertical downward bubbly flow, *International Journal of Heat and Mass Transfer* 47 1847-1862.

APPENDIX 1

Source code for Chapter 2

```
/******
UDF UDS : Pebble Bed
******/

#include "udf.h"
#include "sg.h"

#define PI 3.14159265

/* Global variables *****/

enum
{
    UDS_Pebble_Surface,
    TOT_UDS
};

static int iLimited = 0, nCells = 0, IRun0 = 0, IPowerRead = 0;

/* Global functions *****/

/* This function should be changed for considered core design */
double dGetVacancyFraction(real* Pos)
{
    double dE = 0.38811; /* Vacancy Fraction of Pebble Bed */

    if(Pos[1] > 1.79) dE = 0.44672;
    if(Pos[1] < 1.06) dE = 0.48094;
    return dE;
}

/* This function should be changed for considered core design */
double dGetPowerDensity(real* x, int eq, Thread* t, cell_t c)
{
    if(IPowerRead) return C_UDMI(c,t,4);

    double dQDOT;

    /* ***** */
    /* x[1] is r When Axisymmetric, x[1] is y When 2D or 3D */
    /* ***** */
    if (x[1] > 1.0 && x[1] < 1.17) dQDOT = 7.0221e+06;
    else if(x[1] >= 1.17 && x[1] < 1.34) dQDOT = 5.5501e+06;
    else if(x[1] >= 1.34 && x[1] < 1.51) dQDOT = 4.5535e+06;
    else if(x[1] >= 1.51 && x[1] < 1.68) dQDOT = 4.0853e+06;
    else if(x[1] >= 1.68) dQDOT = 3.6455e+06;
    else
    {
        printf("Error in dGetPowerDensity(real* x), for the call from equation %d ! \n", eq);
        dQDOT = 0;
    }

    return dQDOT;
}
```



```

/* Flux boundary profile for helium side of core-reflector interface which is two sided
uncoupled wall.
!!!!!!!!!!!!!! The wall should be uncoupled !!!!!!!!!!!!!!! */
DEFINE_PROFILE(CoreWall_Helium2Ref, ptCoreSide, i)
{
    Thread *ptRefSide, *ptHelium;
    cell_t cHelium;
    ptRefSide = ptCoreSide->shadow;
    ptHelium = ptCoreSide->t0;

    if (!FLUID_THREAD_P(ptHelium))
        Error("In UDSBoundary1; ERROR ! \n No fluid domain next to wall ! \n");

    real A[ND_ND], /* The face area normal vector */
        dr0[ND_ND], /* The vector that connects the centroid of cell c0 to the face centroid */
        es[ND_ND]; /* The unit normal vector in the direction from centroid */

    double ds, /* Distance between cell and face centroids */
        A_by_es; /* A.A/A.es */

    double TRef, /* Reflector side face temperature */
        THelium, /* Core side cell temperature */
        HeCon; /* Effective thermal conductivity of helium */

    register int f;
    for( f = 0 ; f < ptCoreSide->nelements ; ++f )
    {
        cHelium = F_C0(f,ptCoreSide);
        TRef = F_T(f,ptRefSide);
        THelium = C_T(cHelium,ptHelium );
        HeCon = C_K_L(cHelium,ptHelium);

        /* Get face geometry for core side */
        BOUNDARY_FACE_GEOMETRY(f,ptCoreSide,A,ds,es,A_by_es,dr0);

        if( !ptHelium->storage[SV_T_G] )
        {
            ((real*)(ptCoreSide->storage[SV_PROFILE_0+i]))[f] = HeCon * (TRef - THelium) / ds;
        }
        else
        {
            ((real*)(ptCoreSide->storage[SV_PROFILE_0+i]))[f] = -HeCon * ((THelium - TRef) * NV_MAG(A) / (ds * NV_DOT(A,es))
                + NV_DOT(C_T_G(cHelium,ptHelium),es) * NV_MAG(A) / (ds * NV_DOT(A,es))
                - NV_DOT(C_T_G(cHelium,ptHelium),A) / NV_MAG(A));
        }
    }
}

/* Flux boundary profile for reflector side of core-reflector interface which is two sided
uncoupled wall.
!!!!!!!!!!!!!! The wall should be uncoupled !!!!!!!!!!!!!!! */
DEFINE_PROFILE(ReflectorSideFluxBoundary, ptRefSideWall, i)
{
    Thread *ptCoreSideWall,
        *ptCore; /* Interior core domain */
    int iCoreCell;
    ptCoreSideWall = ptRefSideWall->shadow;
    ptCore = ptCoreSideWall->t0;

    if (!FLUID_THREAD_P(ptCore))
        Error("In UDSBoundary1; ERROR ! \n No fluid domain next to wall ! \n");
}

```

```

real A[ND_ND], /* The face area normal vector */
dr0[ND_ND], /* The vector that connects the centroid of cell c0 to the face centroid */
es[ND_ND]; /* The unit normal vector in the direction from centroid */

double ds, /* Distance between cell and face centroids */
A_by_es; /* A.A/A.es */

double TRef, /* Reflector side face temperature */
THelium, /* Core side cell temperature */
HeCon, /* Effective thermal conductivity of helium */
TBall, /* Pebble surface temperature (UDS) */
UDSDiff; /* Effective pebblebed conductivity (UDS diffusivity) */

register int f;
for( f = 0 ; f < ptRefSideWall->nelements ; ++f )
{
    iCoreCell = F_C0(f,ptCoreSideWall);
    TRef = F_T(f,ptRefSideWall);
    THelium = C_T(iCoreCell,ptCore);
    HeCon = C_K_L(iCoreCell,ptCore);
    TBall = C_UDSI(iCoreCell,ptCore,UDS_Pebble_Surface);
    UDSDiff = C_UDSI_DIFF(iCoreCell,ptCore,UDS_Pebble_Surface);

    BOUNDARY_FACE_GEOMETRY(f,ptCoreSideWall,A,ds,es,A_by_es,dr0);

    if( !ptCore->storage[SV_T_G] || !ptCore->storage[SV_UDSI_G(UDS_Pebble_Surface)] )
        ( (real*)(ptRefSideWall->storage[SV_PROFILE_0+i]) )[f] = HeCon * (THelium - TRef) / ds
            + UDSDiff * (TBall - TRef) / ds;

    else
        ( (real*)(ptRefSideWall->storage[SV_PROFILE_0+i]) )[f] = HeCon * ((THelium - TRef) * NV_MAG(A) / (ds *
NV_DOT(A,es))
            + NV_DOT(C_T_G(iCoreCell,ptCore),es) * NV_MAG(A) / (ds * NV_DOT(A,es))
            - NV_DOT(C_T_G(iCoreCell,ptCore),A) / NV_MAG(A))
            + UDSDiff * ((TBall - TRef) * NV_MAG(A) / (ds * NV_DOT(A,es))
            + NV_DOT(C_UDSI_G(iCoreCell,ptCore,UDS_Pebble_Surface),es)
            * NV_MAG(A) / NV_DOT(A,es)
            - NV_DOT(C_UDSI_G(iCoreCell,ptCore,UDS_Pebble_Surface),A) / NV_MAG(A));
}
}

/* This code calculates Fuel Ball Centre Temperatures, Fuel Ball (moderator) Average Temperatures,
Average Fuel Kernel Centre Temperature and Average Fuel Kernel temperature
then stores them in UDM-0, UDM-1, UDM-2 and UDM-3 respectively */
DEFINE_ON_DEMAND(Calculate_Fuel_Temperatures)
{
    Domain* pFluid;
    Thread* t; /* Core zone thread pointer */
    double dQ, dQt; /* Power density, total heat generated in fuel */
    double dRf = 0.025, dR = 0.03; /* Fuel matrix radius, Fuel radius */
    double dK; /* Fuel matrix graphite conductivity */
    double dTs; /* Fuel surface temperature */
    real x[ND_ND]; /* Cell Position */
    int nk = 13222; /* Number of fuel kernels in a ball */
    double dKK; /* Fuel kernel conductivity */
    double dRk = 250E-6; /* Fuel kernel radius */
    double dVrk; /* Volume of a fuel kernel */

```

```

double dQk, dQkt; /* Power density in a fuel kernel, total power in a fuel kernel */

pFluid = Get_Domain(1);
t = Lookup_Thread(pFluid,17);
dVk = 4/3*PI*pow(dRk,3);

register int c;
for(c = 0; c < t->nelements; ++c)
{
    dTs = C_UDSI(c,t,UDS_Pebble_Surface);

    /* Constant Graphite A3-3 conductivity assumption in fuel ball (~1.67*10E21 EDN) */
    dK = 7.3E-9 * pow(dTs,3) - 2.485E-5 * SQR(dTs) + 0.022 * dTs + 15.155;

    C_CENTROID(x,c,t);

    dQ = dGetPowerDensity(x,0,t,c);
    dQ = dQ/0.61;
    dQt = dQ * 4/3 * PI * pow(dR,3);

    /* Store fuel centre temperature in UDM-0 */
    C_UDMI(c,t,0) = dQ * SQR(dRf) / (6 * dK) + dQt/(4*PI*dK) * (1/dRf - 1/dR) + dTs;

    C_UDMI(c,t,1) = ( dQt/dK*( 0.5*( SQR(dR) - SQR(dRf) ) + 1/(3*dR)*( pow(dRf,3) - pow(dR,3) ))
        + 4/3*PI*dTs*( pow(dR,3) - pow(dRf,3)) + 8*PI*dQ*pow(dRf,5)/(90*dK)
        + 4/3*PI*pow(dRf,3)*(dQt/(4*PI*dK)*(1/dRf - 1/dR) + dTs)) / (4/3*PI*pow(dR,3));

    /* Assuming fuel kernel surface temperature is UDM-1 */
    /* Constant UO2 conductivity assumption (Westinghouse UO2 conductivity) */
    dKK = 100/(11.8 + 0.0238*(C_UDMI(c,t,1) - 273.15)) + 8.775E-11 * pow((C_UDMI(c,t,1) - 273.15),3);
    dQkt = dQt / nk;
    dQk = dQkt / dVk;
    C_UDMI(c,t,2) = dQk * SQR(dRk) / (6*dKK) + C_UDMI(c,t,1);
    C_UDMI(c,t,3) = dQk * SQR(dRk) / (15*dKK) + C_UDMI(c,t,1);
}
IRun0 = 1;
}

DEFINE_ON_DEMAND(Read_Power_Density)
{
    int nx = 10, ny = 4; /* Number of spectrum zones in x, in y (in r) */
    double x0 = 3.015, x1 = 14.015, y0 = 1.0, y1 = 1.85; /* Core zone position specification */
    double dXgs, dYgs; /* Grid size in x and y (for spectrum zones) */
    Domain* pFluid;
    Thread* t; /* Core zone thread pointer */
    double x[ND_ND]; /* Cell Position */
    register int c; /* Cell index */
    double* pdPower; /* Spectrum zone power densities */

    pFluid = Get_Domain(1);
    t = Lookup_Thread(pFluid,17); /* Get core zone */

    if((pdPower = malloc(nx*ny*sizeof(double))) == NULL)
        Error(" Memory allocation failed in Read_Power_Density !");

    for(c = 0; c < nx*ny; ++c)    *(pdPower + c) = 0;

    dXgs = (x1-x0)/((double)nx);    dYgs = (y1-y0)/((double)ny);

    double dPowTab[10][4] =
    {
        4.7968e+06, 3.8243e+06, 3.6044e+06, 3.9880e+06,
        6.7950e+06, 5.2540e+06, 5.0267e+06, 5.7931e+06,
        8.3027e+06, 6.3998e+06, 6.0695e+06, 7.0307e+06,
        8.8669e+06, 6.9167e+06, 6.5963e+06, 7.6508e+06,
        8.7838e+06, 6.7635e+06, 6.4578e+06, 7.4534e+06,
        7.8841e+06, 6.1580e+06, 5.7956e+06, 6.6957e+06,
    }
}

```

```

6.2338e+06, 4.7895e+06, 4.5915e+06, 5.3643e+06,
3.9443e+06, 3.0762e+06, 2.9040e+06, 3.3713e+06,
1.4321e+06, 1.1890e+06, 1.1030e+06, 1.1572e+06,
3.4663e+05, 3.0147e+05, 2.7394e+05, 2.3881e+05 };

pdPower = dPowTab;

/* register int ix;
for(ix = 0; ix < nx; ++ix)
{
    register int iy;
    for(iy = 0; iy < ny; ++iy)
    {
        *(pdPower + ix*ny + iy) = ;
        Message("%f ",*(pdPower + ix*ny + iy));
    }
    Message("\n");
}*/

for(c = 0; c < t->nelements; ++c)
{
    C_CENTROID(x,c,t);
    register int ix;
    for(ix = 0; ix < nx; ++ix)
    {
        if(!((x[0] > (x0 + (double)ix * dXgs)) && (x[0] < (x0 + (double)(ix+1) * dXgs)))) continue;
        register int iy;
        for(iy = 0; iy < ny; ++iy)
        {
            if(!((x[1] > (y0 + (double)iy * dYgs)) && (x[1] < (y0 + (double)(iy+1) * dYgs)))) continue;
            C_UDMI(c,t,4) = *(pdPower + ix*ny + iy);
        }
    }
}
IPowerRead = 1;
}

/* This function writes zone averaged moderator and kernel temperatures to some files.
Zone grid structure is defined by the variables nx,ny,x0,x1,y0 and y1 (defined in function) */
DEFINE_ON_DEMAND(Write_Fuel_Temperatures)
{
    if(!Run0 != 1) Error("Run Calculate_Fuel_Temperatures first !");

    int nx = 10, ny = 4; /* Number of spectrum zones in x, in y (in r) */
    double x0 = 3.015, x1 = 14.015, y0 = 1.0, y1 = 1.85; /* Core zone position specification */
    double dXgs, dYgs; /* Grid size in x and y (for spectrum zones) */
    double* pdTball; /* Average ball (moderator) temperatures */
    double* pdTkernel; /* Average kernel temperatures */
    Domain* pFluid;
    Thread* t; /* Core zone thread pointer */
    Thread* tc; /* Central Reflector zone thread pointer */
    Thread* ts; /* Side Reflector zone thread pointer */
    double x[ND_ND]; /* Cell Position */
    double* pdVolume; /* Spectrum zone volumes */
    register int c; /* Cell index */

    pFluid = Get_Domain(1);
    t = Lookup_Thread(pFluid,17); /* Get core zone */

    if((pdTball = malloc(nx*ny*sizeof(double))) == NULL)
        Error(" Memory allocation failed in Write_Fuel_Temperatures!");
    if((pdTkernel = malloc(nx*ny*sizeof(double))) == NULL)
        Error(" Memory allocation failed in Write_Fuel_Temperatures!");
    if((pdVolume = malloc(nx*ny*sizeof(double))) == NULL)
        Error(" Memory allocation failed in Write_Fuel_Temperatures!");

    for(c = 0; c < nx*ny; ++c)

```

```

{
    *(pdTball + c) = 0; *(pdTkernel + c) = 0; *(pdVolume + c) = 0;
}

dXgs = (x1-x0)/((double)nx);    dYgs = (y1-y0)/((double)ny);

for(c = 0; c < t->nelements; ++c)
{
    C_CENTROID(x,c,t);
    register int ix;
    for(ix = 0; ix < nx; ++ix)
    {
        if(!((x[0] > (x0 + (double)ix * dXgs)) && (x[0] < (x0 + (double)(ix+1) * dXgs)))) continue;
        register int iy;
        for(iy = 0; iy < ny; ++iy)
        {
            if(!((y[1] > (y0 + (double)iy * dYgs)) && (y[1] < (y0 + (double)(iy+1) * dYgs)))) continue;
            *(pdTball + ix*ny + iy) += C_UDMI(c,t,1) * C_VOLUME(c,t);
            *(pdTkernel + ix*ny + iy) += C_UDMI(c,t,3) * C_VOLUME(c,t);
            *(pdVolume + ix*ny + iy) += C_VOLUME(c,t);
        }
    }
}

register int i;
for(i = 0; i < nx*ny; ++i)
{
    *(pdTball + i) = *(pdTball + i) / *(pdVolume + i);
    *(pdTkernel + i) = *(pdTkernel + i) / *(pdVolume + i);
}

FILE *pfModerator, *pfKernel;
if((pfModerator = fopen("Moderator.dat","w")) == NULL) Error("File error in Write_Fuel_Temperatures !");
if((pfKernel = fopen("Kernel.dat","w")) == NULL) Error("File error in Write_Fuel_Temperatures !");

register int ix;
for(ix = 0; ix < nx; ++ix)
{
    register int iy;
    for(iy = 0; iy < ny; ++iy)
    {
        fprintf(pfModerator, "%f ", *(pdTball + ix*ny + iy));
        fprintf(pfKernel, "%f ", *(pdTkernel + ix*ny + iy));
    }
    fprintf(pfModerator, "\n"); fprintf(pfKernel, "\n");
}
fclose(pfModerator); fclose(pfKernel);
}

/* Do not use this macro !!!!!!!!!!!!!!!!!!!!!!! */
#define _HEAT_FLUX(Pebble2CoreWall, f, t, c0, t0, cid, cir)
{
    if(t0 == NULL) return;
    if (!Data_Valid_P() || !FLUID_THREAD_P(t0)) return;

    double ds, /* Distance between cell and face centroids */
           A_by_es; /* A.A/A.es */

    double Tface, /* Face temperature */
           UDS, /* UDS(Ball surface temperature of the cell adjacent to face f */
           UDSDiff; /* User-defined scalar diffusivity (Pebble bed conductivity) */

    Tface = F_T(f,t);
    UDS = C_UDSI(c0,t0,UDS_Pebble_Surface);
    USDiff = C_UDSI_DIFF(c0,t0,UDS_Pebble_Surface);

    real NV_VEC(A), /* The face area normal vector */

```

```

        NV_VEC(dr0), /* The vector that connects the centroid of cell c0 to the face centroid */
        NV_VEC(es); /* The unit normal vector in the direction from centroid */

BOUNDARY_FACE_GEOMETRY(f,t,A,ds,es,A_by_es,dr0);

if( !t0->storage[SV_UDSI_G(UDS_Pebble_Surface)] )
{
    *cid = (real) UDSDiff * (Tface - UDS) * A_by_es / ds;
}
else
{
    *cid = (real) (UDSDiff * (Tface - UDS) * A_by_es / ds

                + UDSDiff * ( NV_DOT(C_UDSI_G(c0,t0,UDS_Pebble_Surface),A)

                - NV_DOT(C_UDSI_G(c0,t0,UDS_Pebble_Surface),es) * A_by_es));
}
}

DEFINE_SOURCE(UDSPowerProfile, c, t, dS, eqn)
{
    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

    double dTC; /* Helium Temperature */
    dTC = ((real*)(t->storage[SV_T]))[c];
    /* dTC = C_T(c,t); */

    double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

    double dAK ; /* Surface Area of a Single Fuel Element m^2 */
    dAK = PI * pow(dOD , 2.0);

    double dVK; /* Volume of a Fuel Sphere m^3 */
    dVK = PI * pow(dOD,3) / 6.0;

    double dE; /* Vacancy Fraction of Pebble Bed */
    dE = dGetVacancyFraction( x );

    double dVIS; /* Dynamic Viscosity of Helium Pa*s */
    dVIS = 3.674E-7 * pow(dTC , 0.7);

    double dP; /* Helium Pressure Pa */
    dP = C_P(c,t) + op_pres;

    double dLAM; /* Heat Conductivity of Helium w/m/K */
    dLAM = C_K_L(c,t);

    double dTK; /* Fuel Element Surface Temperature K.( This is the solution of UDS transport equation.) */
    dTK = C_UDSI(c,t,UDS_Pebble_Surface);

    /* Calculate Heat Transfer Coefficient on Fuel Element Surface */
    double dPr , dRe , dNu , dALFA;

    dPr = C_CP(c,t) * dVIS / (dLAM);
    dRe = C_R(c,t) * (fabs(C_U(c,t)) + fabs(C_V(c,t))) * dOD / dVIS;
    dNu = 1.27 * pow(dPr,(1.0/3.0)) * pow(dRe,0.36) / pow(dE,1.18)
        + 0.033 * pow(dPr,0.5) * pow(dRe,0.86) / pow(dE,1.07);
    dALFA = dNu * dLAM / dOD;

    /* Calculate Heat Transfer to Helium from a Ball */
    double dQCVB;
    dQCVB = dALFA * dAK * (dTK - dTC);

    double dQDOT; /* Power Density */

```

```

dQDOT = dGetPowerDensity( x, eqn, t, c );

/* Calculate Heat Generation in a Fuel Sphere */
double dQK;
dQK = dQDOT * dVK / (1-dE);

if(dQCVB > dQK) return 0;

dS[eqn] = - dQDOT * dALFA * dAK / dQK; /* Implicit part of the source term */
return dQDOT * (1 - dQCVB/dQK); /* Return explicit part */
}

DEFINE_SOURCE(HeliumSourcePowerProfile, c, t, dS, eqn)
{
    nCells = t->nelements;

    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

    double dTC; /* Helium Temperature */
    dTC = C_T(c,t);

    double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

    double dAK ; /* Surface Area of a Single Fuel Element m^2 */
    dAK = PI * pow(dOD , 2.0);

    double dVK; /* Volume of a Fuel Sphere m^3 */
    dVK = PI * pow(dOD,3) / 6.0;

    double dE; /* Vacancy Fraction of Pebble Bed */
    dE = dGetVacancyFraction( x );

    double dVIS; /* Dynamic Viscosity of Helium Pa*s */
    dVIS = 3.674E-7 * pow(dTC , 0.7);

    double dP; /* Helium Pressure Pa */
    dP = C_P(c,t) + op_pres;

    double dLAM; /* Heat Conductivity of Helium w/m/K */
    dLAM = C_K_L(c,t);

    double dTK; /* Fuel Element Surface Temperature K.( This is the solution of UDS transport equation.) */
    dTK = C_UDSI(c,t,UDS_Pebble_Surface);

    /* Calculate Heat Transfer Coefficient on Fuel Element Surface */
    double dPr , dRe , dNu , dALFA;

    dPr = C_CP(c,t) * dVIS / (dLAM);
    dRe = C_R(c,t) * (fabs(C_U(c,t)) + fabs(C_V(c,t))) * dOD / dVIS;
    dNu = 1.27 * pow(dPr,(1.0/3.0)) * pow(dRe,0.36) / pow(dE,1.18)
        + 0.033 * pow(dPr,0.5) * pow(dRe,0.86) / pow(dE,1.07);
    dALFA = dNu * dLAM / dOD;

    /* Calculate Heat Transfer to Helium from a Ball */
    double dQCVB;
    dQCVB = dALFA * dAK * (dTK - dTC);

    double dQDOT; /* Power Density */
    dQDOT = dGetPowerDensity( x, eqn, t, c );

    /* Calculate Heat Generation in a Fuel Sphere*/
    double dQK;
    dQK = dQDOT * dVK / (1-dE);

    if(dQCVB > dQK)
    {

```

```

    iLimited++;
    return dQDOT;
}
/*dS[eqn] = - dQDOT * dALFA * dAK; /* Implicit part of the source term. ! Assumes d fALFA / d dTC = 0 ! */
return dQDOT * dQCVB / dQK; /* Return explicit part */

}

DEFINE_ADJUST(adjust, d)
{
    if(iLimited > 0) Message("Helium source power limited in %d cells of total %d! \n", iLimited, nCells);
    iLimited = 0;
}

DEFINE_SOURCE(X_Momentum_Source, c, t, dS, eqn)
{
    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

    double dE; /* Vacancy Fraction of Pebble Bed */
    dE = dGetVacancyFraction( x );

    double dVIS; /* Dynamic Viscosity of Helium Pa*s */
    dVIS = 3.674E-7 * pow(C_T(c,t), 0.7);

    double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

    double dRe; /* Reynolds number */
    dRe = C_R(c,t) * C_U(c,t) * dOD / dVIS;
    if(dRe == 0) return 0;

    /*printf("%f %f \n", C_U(c,t), C_V(c,t));*/
    double dCoef; /* Fractional coefficient */
    dCoef = 320 / ( dRe / (1 - dE) ) + 6 / pow(( dRe / (1 - dE) ), 0.1);

    if(dCoef != dCoef) return 0;

    dS[eqn] = -(1 - dE) / (pow(dE,3) * dOD * 2) * C_R(c,t) * ( 320 * dVIS * (1-dE) / (C_R(c,t) * dOD) +
        9.6 / pow((C_R(c,t) * dOD / (dVIS * (1 - dE))),0.1)
        * pow(C_U(c,t),0.9));

    if(dS[eqn] != dS[eqn]) return 0;

    double dRvalue;
    dRvalue = -dCoef * (1 - dE) * C_R(c,t) * SQR(C_U(c,t)) / (pow(dE,3) * dOD * 2);

    if(dRvalue != dRvalue) return 0;
    return dRvalue;
}

DEFINE_SOURCE(X_Momentum_Source1, c, t, dS, eqn)
{
    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

    double dE; /* Vacancy Fraction of Pebble Bed */
    dE = dGetVacancyFraction( x );

    double dVIS; /* Dynamic Viscosity of Helium Pa*s */
    dVIS = 3.674E-7 * pow(C_T(c,t), 0.7);

    double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

    double dRe; /* Reynolds number */

```

```

double dV; /* Velocity magnitude */
dV = sqrt(SQR(C_U(c,t))+SQR(C_V(c,t)));
dRe = fabs( C_R(c,t) * dV * dOD / dVIS);
if(dRe == 0) return 0;
if(dRe != dRe)
{
    printf("In \"X_Momentum_Source1\" Reynolds is NAN, returned zero ! \n");
    return 0;
}
/*printf("%f %f \n",C_U(c,t),C_V(c,t));*/
double dCoef; /* Fractional coefficient */
dCoef = 320 / ( dRe / (1 - dE) ) + 6 / pow(( dRe / (1 - dE) ) , 0.1);

if(dCoef != dCoef)
{
    printf("In \"X_Momentum_Source1\" dCoef is NAN, returned zero ! \n");
    return 0;
}

dS[eqn] = -160 * dVIS * SQR((1 - dE)) * C_U(c,t) / (pow(dE,3) * SQR(dOD) * dV)
- 3 * pow(dVIS,0.1) * pow((1 - dE),1.1) * pow(C_R(c,t),0.9) / (pow(dE,3) * pow(dOD,1.1))
* C_U(c,t) * (1.9 * SQR(C_U(c,t)) + 2 * SQR(C_V(c,t))) / pow(SQR(dV),1.05);

if(dS[eqn] != dS[eqn]) dS[eqn] = 0;

double dRvalue;

/*if(C_U(c,t) > 0) dRvalue = -dCoef * (1 - dE) * C_R(c,t) * SQR(C_U(c,t)) / (pow(dE,3) * dOD * 2);
else dRvalue = dCoef * (1 - dE) * C_R(c,t) * SQR(C_U(c,t)) / (pow(dE,3) * dOD * 2);*/

dRvalue = -dCoef * (1 - dE) * C_R(c,t) * dV * C_U(c,t) / (pow(dE,3) * dOD * 2);

if(dRvalue != dRvalue)
{
    printf("In \"X_Momentum_Source1\" x momentum source is NAN, returned zero ! \n");
    return 0;
}
return dRvalue;
}

DEFINE_SOURCE(Y_Momentum_Source, c, t, dS, eqn)
{
    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

    double dE; /* Vacancy Fraction of Pebble Bed */
    dE = dGetVacancyFraction( x );

    double dVIS; /* Dynamic Viscosity of Helium Pa*s */
    dVIS = 3.674E-7 * pow(C_T(c,t) , 0.7);

    double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

    double dRe; /* Reynolds number */
    dRe = C_R(c,t) * C_V(c,t) * dOD / dVIS;

    /*printf("%f %f \n",C_U(c,t),C_V(c,t));*/
    double dCoef; /* Fractional coefficient */
    if(dRe == 0) return 0;

    dCoef = 320 / ( dRe / (1 - dE) ) + 6 / pow(( dRe / (1 - dE) ) , 0.1);
    if(dCoef != dCoef) return 0;

    dS[eqn] = -(1 - dE) / (pow(dE,3) * dOD * 2) * C_R(c,t) * ( 320 * dVIS * (1-dE) / (C_R(c,t) * dOD) +
    9.6 / pow((C_R(c,t) * dOD / (dVIS * (1 - dE))),0.1)
    * pow(C_V(c,t),0.9));
    if(dS[eqn] != dS[eqn]) return 0;

    double dRvalue;

```

```

dRvalue = -dCoef * (1 - dE) * C_R(c,t) * SQR(C_V(c,t)) / (pow(dE,3) * dOD * 2);

if(dRvalue != dRvalue) return 0;
return dRvalue;
}

DEFINE_SOURCE(Y_Momentum_Source1, c, t, dS, eqn)
{
    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

    double dE; /* Vacancy Fraction of Pebble Bed */
    dE = dGetVacancyFraction( x );

    double dVIS; /* Dynamic Viscosity of Helium Pa*s */
    dVIS = 3.674E-7 * pow(C_T(c,t), 0.7);

    double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

    double dV; /* Velocity magnitude */
    dV = sqrt(SQR(C_U(c,t))+SQR(C_V(c,t)));

    double dRe; /* Reynolds number */
    dRe = C_R(c,t) * dV * dOD / dVIS;

    /*printf("%f %f \n",C_U(c,t),C_V(c,t));*/
    double dCoef; /* Fractional coefficient */
    if(dRe == 0) return 0;
    if(dRe != dRe)
    {
        printf("In \"Y_Momentum_Source1\" Reynolds is NAN, returned zero ! \n");
        return 0;
    }

    dCoef = 320 / ( dRe / (1 - dE) ) + 6 / pow(( dRe / (1 - dE) ), 0.1);
    if(dCoef != dCoef)
    {
        printf("In \"Y_Momentum_Source1\" dCoef is NAN, returned zero ! \n");
        return 0;
    }

    /* dS[eqn] = -160 * dVIS * SQR((1 - dE)) * C_V(c,t) / (pow(dE,3) * SQR(dOD) * dV)
    - 3 * pow(dVIS,0.1) * pow((1 - dE),1.1) * pow(C_R(c,t),0.9) / (pow(dE,3) * pow(dOD,1.1))
    * C_V(c,t) * (1.9 * SQR(C_V(c,t)) + 2 * SQR(C_U(c,t))) / pow(SQR(dV),1.05);*/

    if(dS[eqn] != dS[eqn]) dS[eqn] = 0;

    double dRvalue;
    /*if(C_V(c,t) > 0) dRvalue = -dCoef * (1 - dE) * C_R(c,t) * SQR(C_V(c,t)) / (pow(dE,3) * dOD * 2);
    else dRvalue = dCoef * (1 - dE) * C_R(c,t) * SQR(C_V(c,t)) / (pow(dE,3) * dOD * 2);*/

    dRvalue = -dCoef * (1 - dE) * C_R(c,t) * dV * C_V(c,t) / (pow(dE,3) * dOD * 2);

    if(dRvalue != dRvalue)
    {
        printf("In \"Y_Momentum_Source1\" y momentum source is NAN, returned zero ! \n");
        return 0;
    }

    return dRvalue;
}

/* X momentum source, correlation from FLUENT manual */
DEFINE_SOURCE(Ergun_X, c, t, dS, eqn)
{
    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

```

```

double dE; /* Vacancy Fraction of Pebble Bed */
dE = dGetVacancyFraction( x );

double dVIS; /* Dynamic Viscosity of Helium Pa*s */
dVIS = 3.674E-7 * pow(C_T(c,t) , 0.7);

double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

dS[eqn] = -150 * dVIS * SQR((1 - dE)) / ( SQR(dOD) * pow(dE , 3) )
- 3.5 * C_R(c,t) * (1 - dE) * C_U(c,t) / ( dOD * pow(dE , 3));
if(dS[eqn] != dS[eqn]) return 0;

double dRvalue;
dRvalue = -150 * dVIS * SQR((1 - dE)) * C_U(c,t) / ( SQR(dOD) * pow(dE , 3) )
- 1.75 * C_R(c,t) * (1 - dE) * SQR(C_U(c,t)) / ( dOD * pow(dE , 3));

if(dRvalue != dRvalue) return 0;

return dRvalue;
}

/* Y momentum source, correlation from FLUENT manuel */
DEFINE_SOURCE(Ergun_Y, c, t, dS, eqn)
{
    real x[ND_ND]; /* Cell Position */
    C_CENTROID(x,c,t);

    double dE; /* Vacancy Fraction of Pebble Bed */
    dE = dGetVacancyFraction( x );

    double dVIS; /* Dynamic Viscosity of Helium Pa*s */
    dVIS = 3.674E-7 * pow(C_T(c,t) , 0.7);

    double dOD = 0.06; /* Outer Diameter of a Fuel Element m */

    dS[eqn] = -150 * dVIS * SQR((1 - dE)) / ( SQR(dOD) * pow(dE , 3) )
- 3.5 * C_R(c,t) * (1 - dE) * C_V(c,t) / ( dOD * pow(dE , 3));
if(dS[eqn] != dS[eqn]) return 0;

    double dRvalue;
    dRvalue = -150 * dVIS * SQR((1 - dE)) * C_V(c,t) / ( SQR(dOD) * pow(dE , 3) )
- 1.75 * C_R(c,t) * (1 - dE) * SQR(C_V(c,t)) / ( dOD * pow(dE , 3));

    if(dRvalue != dRvalue) return 0;

    return dRvalue;
}

DEFINE_PROPERTY(HeliumThermalConductivity, c, t)
{
    double dTC; /* Helium Temperature K */
    dTC = C_T(c,t);

    double dP; /* Helium Pressure Pa */
    dP = C_P(c,t) + op_pres;

    return 2.682E-3 * (1.0 + 1.123E-8 * dP) * pow(dTC , (0.71*(1.0 - 2.0E-9 * dP)));
}

DEFINE_DIFFUSIVITY(EffectivePebbleBedHeatConductivity, c, t, i)
{
    double T;
    T = C_UDSI(c,t,i) - 173.15; /* Pebble surface temperature is assumed to be in K */
    if(T < 250) T = 250;

    /* Calculate heat conductivity of uniformed pebble bed using German Breitbach correlation. W/m/K */
    return 1.1538E-4 * pow(T , 1.6622);
}

```

```

}

/* Effective thermal conductivity of pebble bed */
DEFINE_DIFFUSIVITY(Zehner_Schlunder, c, t, eqn)
{
    /* Corralation table   T   No dose   6.09E21 */
    double dTab[15][3] = {
        373.15 , 2.66E-2 , 1.73E-2 ,
        473.15 , 3.45E-2 , 2.42E-2 ,
        573.15 , 4.55E-2 , 3.39E-2 ,
        673.15 , 5.99E-2 , 4.51E-2 ,
        773.15 , 7.75E-2 , 5.77E-2 ,
        873.15 , 9.68E-2 , 6.99E-2 ,
        973.15 , 1.16E-1 , 8.17E-2 ,
        1073.15 , 1.35E-1 , 8.99E-2 ,
        1173.15 , 1.55E-1 , 1.01E-1 ,
        1273.15 , 1.75E-1 , 1.12E-1 ,
        1573.15 , 2.21E-1 , 1.46E-1 ,
        1873.15 , 2.54E-1 , 1.81E-1 ,
        2173.15 , 2.75E-1 , 2.18E-1 ,
        2473.15 , 2.92E-1 , 2.53E-1 ,
        2773.15 , 3.04E-1 , 2.81E-1
    };

    double dDose = 1.59E21; /* Fast neutron dose of pebbles */
    double dK11, dK22;

    if(C_UDSI(c,t,eqn) < dTab[0][0])
    {
        dK11 = (C_UDSI(c,t,eqn) - dTab[0][0]) * (dTab[0][1] - dTab[1][1])
            /(dTab[0][0] - dTab[1][0]) + dTab[0][1];

        dK22 = (C_UDSI(c,t,eqn) - dTab[0][0]) * (dTab[0][2] - dTab[1][2])
            /(dTab[0][0] - dTab[1][0]) + dTab[0][2];

        return 100 * (dDose / 6.09E21 * (dK22 - dK11) + dK11);
    }

    if(C_UDSI(c,t,eqn) > dTab[14][0])
    {
        dK11 = (C_UDSI(c,t,eqn) - dTab[14][0]) * (dTab[14][1] - dTab[13][1])
            /(dTab[14][0] - dTab[13][0]) + dTab[14][1];

        dK22 = (C_UDSI(c,t,eqn) - dTab[14][0]) * (dTab[14][2] - dTab[13][2])
            /(dTab[14][0] - dTab[13][0]) + dTab[14][2];

        return 100 * (dDose / 6.09E21 * (dK22 - dK11) + dK11);
    }

    register int i;
    for(i = 0 ; i < 14 ; i++)
    {
        if(C_UDSI(c,t,eqn) >= dTab[i][0] && C_UDSI(c,t,eqn) <= dTab[i+1][0])
        {
            dK11 = (C_UDSI(c,t,eqn) - dTab[i][0]) * (dTab[i+1][1] - dTab[i][1])
                /(dTab[i+1][0] - dTab[i][0]) + dTab[i][1];

            dK22 = (C_UDSI(c,t,eqn) - dTab[i][0]) * (dTab[i+1][2] - dTab[i][2])
                /(dTab[i+1][0] - dTab[i][0]) + dTab[i][2];

            break;
        }
    }

    return 100 * (dDose / 6.09E21 * (dK22 - dK11) + dK11);
}

```

```

DEFINE_PROPERTY(He_Density, c, t)
{
    double dTC; /* Helium Temperature K*/
    dTC = C_T(c,t);

    double dP; /* Helium Pressure Pa */
    dP = C_P(c,t) + op_pres;
    /*printf("OK %f %f \n",fTC,fP);*/
    return 48.14E-5 * dP / dTC * 1/( 1 + 0.4446E-5 * dP / pow(dTC , 1.2));
}

DEFINE_PROPERTY(He_Viscosity, c, t)
{
    return 3.674E-7 * pow(C_T(c,t) , 0.7);
}

```

Source code for Chapter 3

16


```

/* Source and sink terms for Dispersed Bubbles */

/*****/
DEFINE_SOURCE( Dis_Bubble_Source, c, t, dS, eqn)
{
    cell_t c0, c1 = -1;
    Thread *t0, *t1 = NULL;
    int n;
    face_t f;
    Thread *tf;

    real V[ND_ND], /* Volume fraction gradient vector */
        A[ND_ND], /* Face normal vector */
        rVolFraction,
        rSource,
        rS;

    real rCRC = 3.0, /* Proportionality between average size of edies and the mean distance between bubbles.
                     This value is approximated as 3.0 by Kim,1999 */
        rCRC1 = 0.003, /* Adjustable coefficeint based on experimental data (random collision) */
        rVfm = 0.62, /* Maximumum volume fraction */
        rExp,
        rVf;

    real rCD, /* Drag coefficient */
        rVr[ND_ND], /* Relative velocity */
        rVrMag, /* Relative velocity magnitude */
        rCWE = 0.042; /*Adjustable coefficeint based on experimental data.(wake entrainment) */

    real rCTI = 0.026, /* Adjustable coefficeint based on experimental data.(turbulent impact) */
        rWeCTI=8, /* Critical Weber number */
        rD, /* Diameter of bubbles */
        rWe, /* Weber number */
        rST = 0.07275; /* Surface tension of air water at 20cel.*/

    real rVOF_dt; /* Time derivative of volume fraction */

    register int j;
    Thread *tDis_Bubble = THREAD_SUB_THREAD(t,enDispersed_Bubble);
    Thread *tCon_Liquid = THREAD_SUB_THREAD(t,enContinuous_Liquid);

    rVf = C_VOF(c,tDis_Bubble);
    NV_D(V, =, 0.0,0.0,0.0);

    /*if(C_VOF(c,tDis_Bubble) == 0.) return 0;*/

    /*****/
    /* Calculate gradient of volume fraction */
    /*c_face_loop(c, tDis_Bubble, n)
    {
        f = C_FACE(c,tDis_Bubble,n);
        tf = C_FACE_THREAD(c,tDis_Bubble,n);

        if (BOUNDARY_FACE_THREAD_P( tf ))
            rVolFraction = C_VOF(c,tDis_Bubble);

        else
        {
            c1 = F_C1(f,tf);
            t1 = F_C1_THREAD(f,tf);
            c0 = F_C0(f,tf);
            t0 = F_C0_THREAD(f,tf);

            rVolFraction = (C_VOF(c0,t0) + C_VOF(c1,t1)) / 2.0;

```

```

    }

    F_AREA(A, f, tf);
    register int j;
    for(j=0 ; j < ND_ND ; j++) V[j] += A[j] * rVolFraction / C_VOLUME(c,t);

}

for(j=0 ; j < ND_ND ; j++) V[j] = V[j] / C_VOLUME(c,t);*/
for(j=0 ; j < ND_ND ; j++) V[j] = C_UDMI(c,t,j);

/*****/

/* Source term due to bubble expansion or generation */

/*****/
if(ND_ND == 2)
{
    rSource = 2.0/3.0 * C_UDSI(c,t,enDis_Buble_eqn) / C_VOF(c,tDis_Bubble)
        * ( C_VOF(c,tDis_Bubble) * ( C_DUDX(c,tDis_Bubble) + C_DVDY(c,tDis_Bubble)
        + C_V(c,tDis_Bubble) * fabs(V[1]) + C_U(c,tDis_Bubble) * fabs(V[0]) );

    dS[eqn] = 2.0/ 3.0 / C_VOF(c,tDis_Bubble)
        * ( C_VOF(c,tDis_Bubble) * ( C_DUDX(c,tDis_Bubble) + C_DVDY(c,tDis_Bubble) )
        + C_V(c,tDis_Bubble) * fabs(V[1]) + C_U(c,tDis_Bubble) * fabs(V[0]) );
}
else
{
    rSource = 2.0/3.0 * C_UDSI(c,t,enDis_Buble_eqn) / C_VOF(c,tDis_Bubble)
        * ( C_VOF(c,tDis_Bubble) * ( C_DUDX(c,tDis_Bubble) + C_DVDY(c,tDis_Bubble) + C_DWDZ(c,tDis_Bubble) )
        + C_V(c,tDis_Bubble) * V[1] + C_U(c,tDis_Bubble) * V[0] + C_W(c,tDis_Bubble) * V[2] );

    dS[eqn] = 2.0/3.0 / C_VOF(c,tDis_Bubble)
        * ( C_VOF(c,tDis_Bubble) * ( C_DUDX(c,tDis_Bubble) + C_DVDY(c,tDis_Bubble) + C_DWDZ(c,tDis_Bubble) )
        + C_V(c,tDis_Bubble) * V[1] + C_U(c,tDis_Bubble) * V[0] + C_W(c,tDis_Bubble) * V[2] );
}

/*****/

/* Sink term from Coalescence due to turbulence induced random collision */

/*****/
if(pow(rVfm,1/3) - pow(rVf,1/3) == 0.0) rExp = 0;
else rExp = exp(-rCRC * pow(rVfm * rVf,1/3) / (pow(rVfm,1/3) - pow(rVf,1/3)));

rS = -0.17 * rCRC1 * pow(C_D(c,t),1/3) * rVf * pow(C_UDSI(c,t,enDis_Buble_eqn),5/3) / pow(rVfm,2/3) * (1 - rExp);
rSource += rS;

C_UDMI(c,t,enRCR) = pow(-rS,3) / (SQR(rVf) * 36 * M_PI);

dS[eqn] += 5/3 * -0.17 * rCRC1 * pow(C_D(c,t),1/3) * rVf * pow(C_UDSI(c,t,enDis_Buble_eqn),2/3) / pow(rVfm,2/3) * (1 -
rExp);

/*****/

/* Sink term due to wake entrainment */

/*****/

rVr[0] = C_U(c,tDis_Bubble) - C_U(c,tCon_Liquid);
rVr[1] = C_V(c,tDis_Bubble) - C_V(c,tCon_Liquid);
if(ND_ND == 3) rVr[2] = C_W(c,tDis_Bubble) - C_W(c,tCon_Liquid);

```

```

rVrMag = NV_MAG(rVr);

rCD = Calculate_Drag_Coefficient(tDis_Bubble, tCon_Liquid, c);

rS = -0.27 * rCWE * rVrMag * pow(rCD,1/3) * pow(C_UDSI(c,t,enDis_Buble_eqn),2);
rSource += rS;

C_UDMI(c,t,enWER) = pow(-rS,3) / (SQR(rVf) * 36 * M_PI);

dS[eqn] += -0.27 * 2.0 * rCWE * rVrMag * pow(rCD,1/3) * C_UDSI(c,t,enDis_Buble_eqn);

/*****/

/* Source term due to bubble disintegration upon turbulent impact */

/*****/

rD = 3 * rVf / C_UDSI(c,t,enDis_Buble_eqn);

rWe = C_R(c,tCon_Liquid) * 2 * C_K(c,t) * rD / rST;

/*Message("%f \n ",rVf);*/

rS = 0.12 * rCTI * pow(C_D(c,t),1/3) * (1 - rVf) * pow(C_UDSI(c,t,enDis_Buble_eqn),5/3) / pow(rVf,2/3)
* exp(- rWEcTI / rWe) * sqrt(fabs(1 - rWEcTI / rWe));
rSource += rS;

C_UDMI(c,t,enTIR) = pow(rS,3) / (SQR(rVf) * 36 * M_PI);

dS[eqn] += 0.2 * rCTI * pow(C_D(c,tCon_Liquid),1/3) * (1 - rVf) * pow(C_UDSI(c,t,enDis_Buble_eqn),2/3) / pow(rVf,2/3)
* exp(- rWEcTI / rWe) * sqrt(fabs(1 - rWEcTI / rWe));

/*****/

/* Transient source term */

/*****/

/*Alloc_Storage_Vars(tDis_Bubble->domain, SV_VOF_M1, SV_VOF_M2, SV_NULL);

rVOF_dt = fabs(C_VOF_M1(c,tDis_Bubble) - C_VOF_M1(c,tDis_Bubble))
/ (RP_Get_Real("physical-time-step") / dt_factor);

rSource += 2.0/3.0 * C_UDSI(c,t,enDis_Buble_eqn) / C_VOF(c,tDis_Bubble)* rVOF_dt;

dS[eqn] += 2.0/3.0 * C_VOF(c,tDis_Bubble)* rVOF_dt;

Free_Storage_Vars(tDis_Bubble->domain, SV_VOF_M1, SV_VOF_M2, SV_NULL);*/

/*if(dS[eqn] != dS[eqn]) dS[eqn] = 0; */
if (rSource != rSource) rSource = 0;
/*dS[eqn] = 0;*/
return rSource;
}

/*oo000000oo000000oo000000oo000000oo000000oo000000oo000000oo000000oo000000oo00000oo*/
/*oo000000oo000000oo000000oo000000oo000000oo000000oo000000oo000000oo000000oo00000oo*/
/*****/

/* Momentum sources for dispersed bubble */

/*****/

DEFINE_SOURCE( D_B_X_Momentum_Source, c, t, dS, eqn)

```

```

{

    /* Wall force */

    /* Wall force */

    real rVr[ND_ND], /* Relative velocity */
        rVra[ND_ND], /* Relative axial velocity */
        rNw[ND_ND], /* Unit normal vector to the wall */
        rMw, /* Wall force */
        rCw1, /* Correlation coefficient 1 */
        rCw2; /* Correlation coefficient 2 */

    Thread *pContinuous_Liquid = THREAD_SUB_THREAD(t->super_thread,enContinuous_Liquid);

    rVr[0] = C_U(c,t) - C_U(c,pContinuous_Liquid);
    rVr[1] = C_V(c,t) - C_V(c,pContinuous_Liquid);
    if(ND_ND == 3) rVr[2] = C_W(c,t) - C_W(c,pContinuous_Liquid);

    rNw[0] = C_UDMI(c,t->super_thread,enCell_to_N_Face_x);
    rNw[1] = C_UDMI(c,t->super_thread,enCell_to_N_Face_y);
    if(ND_ND == 3) rNw[2] = C_UDMI(c,t->super_thread,enCell_to_N_Face_z);

    NV_V_VS(rVra, =, rVr, -, rNw, *, NV_DOT(rNw, rVr));

    rCw1 = -0.104 * -0.06 * NV_MAG(rVr);
    rCw2 = -0.147;

    if(C_UDMI(c,t->super_thread,enCell_Wall_Dist) < C_UDMI(c,t->super_thread,enD))
        rMw = -2.0 * C_VOF(c,t) * C_R(c,t->super_thread) * NV_MAG2(rVra)
            * (rCw1 + rCw2 * C_UDMI(c,t->super_thread,enD)
            /(2.0 * C_UDMI(c,t->super_thread,enCell_Wall_Dist)))
            * rNw[0] / C_UDMI(c,t->super_thread,enD);
    else
        rMw = 0.0;

    return rMw;
}

DEFINE_SOURCE( D_B_Y_Momentum_Source, c, t, dS, eqn)
{
    /* Wall force */

    /* Wall force */

    real rVr[ND_ND], /* Relative velocity */
        rVra[ND_ND], /* Relative axial velocity */
        rNw[ND_ND], /* Unit normal vector to the wall */
        rMw, /* Wall force */
        rCw1, /* Correlation coefficient 1 */
        rCw2; /* Correlation coefficient 2 */

    Thread *pContinuous_Liquid = THREAD_SUB_THREAD(t->super_thread,enContinuous_Liquid);

    rVr[0] = C_U(c,t) - C_U(c,pContinuous_Liquid);
    rVr[1] = C_V(c,t) - C_V(c,pContinuous_Liquid);
    if(ND_ND == 3) rVr[2] = C_W(c,t) - C_W(c,pContinuous_Liquid);

    rNw[0] = C_UDMI(c,t->super_thread,enCell_to_N_Face_x);
    rNw[1] = C_UDMI(c,t->super_thread,enCell_to_N_Face_y);
    if(ND_ND == 3) rNw[2] = C_UDMI(c,t->super_thread,enCell_to_N_Face_z);

    NV_V_VS(rVra, =, rVr, -, rNw, *, NV_DOT(rNw, rVr));

    rCw1 = -0.104 * -0.06 * NV_MAG(rVr);

```



```

/*****/

/* Wall force */

/*****/

real rVr[ND_ND], /* Relative velocity */
rVra[ND_ND], /* Relative axial velocity */
rNw[ND_ND], /* Unit normal vector to the wall */
rMw, /* Wall force */
rCw1, /* Correlation coefficient 1 */
rCw2; /* Correlation coefficient 2 */

Thread *pDispersed_Bubble = THREAD_SUB_THREAD(t->super_thread,enDispersed_Bubble);

rVr[0] = C_U(c,pDispersed_Bubble) - C_U(c,t);
rVr[1] = C_V(c,pDispersed_Bubble) - C_V(c,t);
if(ND_ND == 3) rVr[2] = C_W(c,pDispersed_Bubble) - C_W(c,t);

rNw[0] = C_UDMI(c,t->super_thread,enCell_to_N_Face_x);
rNw[1] = C_UDMI(c,t->super_thread,enCell_to_N_Face_y);
if(ND_ND == 3) rNw[2] = C_UDMI(c,t->super_thread,enCell_to_N_Face_z);

NV_V_VS(rVra, =, rVr, -, rNw, *, NV_DOT(rNw, rVr));

rCw1 = -0.104 * -0.06 * NV_MAG(rVr);
rCw2 = -0.147;

if(C_UDMI(c,t->super_thread,enCell_Wall_Dist) < C_UDMI(c,t->super_thread,enD))
rMw = 2.0 * C_VOF(c,pDispersed_Bubble) * C_R(c,t) * NV_MAG2(rVra)
* (rCw1 + rCw2 * C_UDMI(c,t->super_thread,enD)
/(2.0 * C_UDMI(c,t->super_thread,enCell_Wall_Dist))) * rNw[0] / C_UDMI(c,t->super_thread,enD);
else
rMw = 0.0;

return rMw;
}

DEFINE_SOURCE( Y_Momentum_Source_C_L, c, t, dS, eqn)
{
/*****/

/* Wall force */

/*****/

real rVr[ND_ND], /* Relative velocity */
rVra[ND_ND], /* Relative axial velocity */
rNw[ND_ND], /* Unit normal vector to the wall */
rMw, /* Wall force */
rCw1, /* Correlation coefficient 1 */
rCw2; /* Correlation coefficient 2 */

Thread *pDispersed_Bubble = THREAD_SUB_THREAD(t->super_thread,enDispersed_Bubble);

rVr[0] = C_U(c,pDispersed_Bubble) - C_U(c,t);
rVr[1] = C_V(c,pDispersed_Bubble) - C_V(c,t);
if(ND_ND == 3) rVr[2] = C_W(c,pDispersed_Bubble) - C_W(c,t);

rNw[0] = C_UDMI(c,t->super_thread,enCell_to_N_Face_x);
rNw[1] = C_UDMI(c,t->super_thread,enCell_to_N_Face_y);
if(ND_ND == 3) rNw[2] = C_UDMI(c,t->super_thread,enCell_to_N_Face_z);

NV_V_VS(rVra, =, rVr, -, rNw, *, NV_DOT(rNw, rVr));

rCw1 = -0.104 * -0.06 * NV_MAG(rVr);
rCw2 = -0.147;

```



```

{
  real physical_dt, phi_old;
  physical_dt = RP_Get_Real("physical-time-step");

  *apu = -C_VOLUME(c,t)/ physical_dt; /*implicit part*/

  phi_old = C_STORAGE_R(c,t,SV_UDSI_M1(i));
  *su = C_VOLUME(c,t)* phi_old / physical_dt; /*explicit part*/
}

```

INDEX

CFD, 1
CFX, 45
COALESCENCE, 50
CONVERGENCE, 30
CRP-5, 19
drag coefficient, 53
euler-euler, 55
FDM, 1
FEM, 1
FLUENT, 4
FVM, 2
GAMBIT, 4
GRID, 28
IATE, 46
lift force, 53
Nusselt number, 16
PBMR, 7
Random collision, 48
RELAP5, 42
Reynolds number, 16, 53
Sauter mean diameter, 47
Shearing-off, 49
Surface instability, 49
TURBULENCE, 26, 27
Turbulent impact, 48
two-fluid model, 51
UDF, 14
UDS, 14
virtual mass force, 54
void fraction, 18
volume fractions, 52
Wake entrainment, 48
wall force, 54

RESUME

Name Surname: Bora Kalpaklı

Birth Place: Ankara, Turkey

Birth Date: March,11 1978

Marital Status: Single

Education and Academic Positions:

- High School: Yenimahalle Mustafa Kemal Lisesi, 1991-1993
- Undergraduate, Hacettepe University, Department of Nuclear Engineering
1995-2003.

Language: English