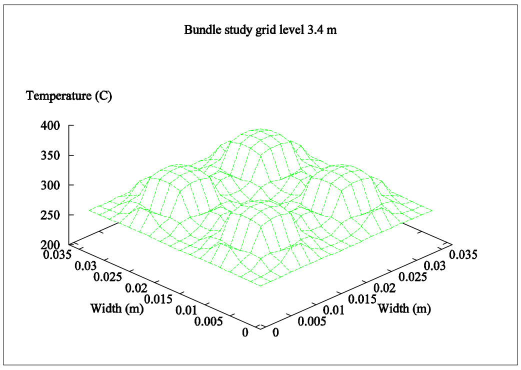
RESEARCH REPORT

NO VTT-R-01678-08 | 14.2.2008



SAFIR2010/TRICOT

Status of PORFLO code development and simulation of the BFBT benchmark problem

Authors: Jaakko Miettinen, Mikko Ilvonen, Ville Hovi

Confidentiality: Public

Report's title Status of PORFLO code development and simulation of the BFBT benchmark problem	
Customer, contact person, address SAFIR2010 Research Programme (VYR,VTT)	Order reference
Project name TRICOT/SAFIR2010	Project number/Short name 13228
Author(s) Jaakko Miettinen, Mikko Ilvonen, Ville Hovi	Pages 22
Keywords Porous media model, two-phase flow, BWR fuel bundle	Report identification code VTT-R-01678-08

Summary

This report describes the work done in the second subtask of the SAFIR2010/TRICOT project. The present work is continuation to the work started in the corresponding previous SAFIR project.

Three-dimensional calculation of a boiling two-phase flow in complex geometry, such as a fuel bundle, is a very challenging task, to which no completely satisfactory solution has been found so far. In PORFLO, the basic features of the computer code include assumption of porous medium, use of five-equation model (with a mixture momentum conservation equation) and description of the problem in Cartesian co-ordinates.

Several major development steps of the PORFLO code were done in 2007. They include bundle-related geometry and meshing pre-processor, improved user interface, modularization and new indexing schemes to facilitate code development, and possibly the most important of all, implementation of the SIMPLE algorithm for coupling of pressure and velocity solutions, including other necessary calculations inside the iteration.

Despite the efforts, results of simulation are not yet satisfactory, the code is not always stable and calculation of solutions may take a very long time. These problems, among others, are studied further in 2008.

Confidentiality	Public

Espoo 31.01.2008

Signatures	Signatures	Signatures
Mikko Ilvonen		Timo Vanttola
Mikko Ilvonen Team Leader	Elina Syrjälahti Project Manager	Timo Vanttola Technology Manager

VTT's contact address

Distribution (customer and VTT)
SAFIR2010 Reference group 3
TK5015

Contents

1 Introduction

PORFLO-related work under the TRICOT project in 2007 was divided into three tasks: general code development, application of the code to the BFBT benchmark problem and application to an open reactor core. Due to difficulties with the BFBT benchmark, the open core simulation was canceled and the efforts were concentrated on general development and on producing an acceptable simulation of the BWR fuel bundle two-phase flow, as measured in the BFBT experiments. Main emphasis of the development work was on the SIMPLE algorithm for pressure-velocity iterative coupling. In a related project, basic documentation and user manual of PORFLO have been produced.

2 General strategy of development

The main strategy in PORFLO development has been to reach a numerically robust and stable version. A reference result produced by direct solution of the matrix equations (with up to 20000 cells) is used to compare with results by iterative solution methods. For realistic results, one million or more cells are desired. The reference solution (with the older basic 'SMABRE' style solution) works when convection terms are left out of the momentum equation. Diffusion terms do not cause a problem. Special consideration has been devoted to keeping the horizontal and vertical flow areas from differing too much, even when the grid has cells that are longer in the vertical direction.

The necessary step towards finer grids is the use of iterative solvers. There are still convergence problems with the CGS algorithm, but it turned out that an older solver (ADI) works even for a problematic case of pressure / flow solution, provided that the convection term of the momentum equation is neglected. The BFBT benchmark has thus been simulated with as many as one million cells. It is attempted next to solve with the convection terms, and to support further development of the new pressure-velocity coupling (SIMPLE algorithm).

The present implementation of the SIMPLE algorithm converges quite slowly, but it has the complete terms of the momentum equation. As soon as convergence can be accelerated, a practically usable 3D two-phase solution is at hand. Work is also continued in the area of CGS and other iterative solvers.

3 New user interface & main branching in the code

The PORFLO code used to be tailored at source code level for each individual application, like particle bed or isolation condenser. Now a branching at the main level is available to handle application specific input and perform various application specific initializations for particle bed, isolation condenser, BWR fuel bundle, open PWR core and steam generator.

The most essential part of the code is the solution of the basic conservation equations for the 3-dimensional structured mesh. The basic solution includes the combined pressure field and volumetric velocity, prediction of the void fraction and consequently the volumetric flow distribution split into phase flow rates, explicit integration of the mass flow distribution and consequently the mass balance error for the next time step, as well as liquid and gas energy equations and temperature distribution in the solid structure. In future development, additional equations may be solved for the turbulent kinetic energy and energy dissipation.

For the pressure-volumetric velocity -solution and void fraction prediction a matrix inversion is needed. The matrix structure is a wide band sparse matrix. With the present CPU capability and old solution algorithms, a case with typically 30000 mesh points (nodes) can be inverted. The full matrix to be inverted would include 900 million elements. With a sparse matrix approach only 58 million matrix elements needs to be processed. For bigger nodalizations, up to one million mesh points, iterative inversion procedures are needed. In these procedures with one million nodes only 9 million elements need to be stored.

Before the actual solution, the boundary conditions need to be defined for diverse calculation cases. This part is partially case dependent. The original facility setup may be so complex that case specific input and output processing is needed for generating the generic nodal information and for extracting the most interesting results. Another alternative would be programming a generic interface similarly with the system codes. The case specific user interface was found more efficient for PORFLO, which is aimed for analyzing very different types of facilities.

In an isolation condenser, heat is generated in the heat transfer tubes condensing vapor. A special module in PORFLO is used for one-dimensional conservation equations inside the condensing tubes, and condensing efficiency is controlled by regulating the water level in the steam and condensate collectors. The condenser includes free water surface, with atmospheric pressure on that level. The final separation of liquid and vapor takes place on this level. Specific input handling is needed for the isolation condenser. The output is tailored according to the needs of the case as well.

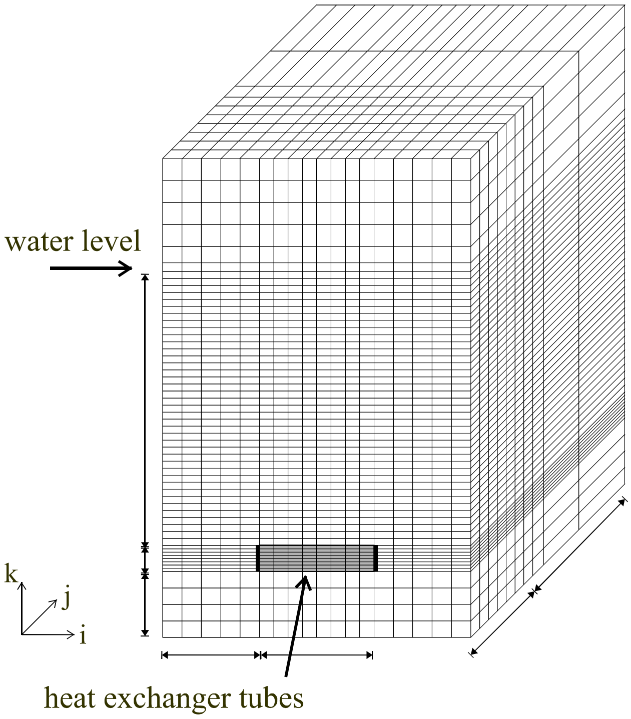


Figure 1. Model of IC pool in the simulations with PORFLO.

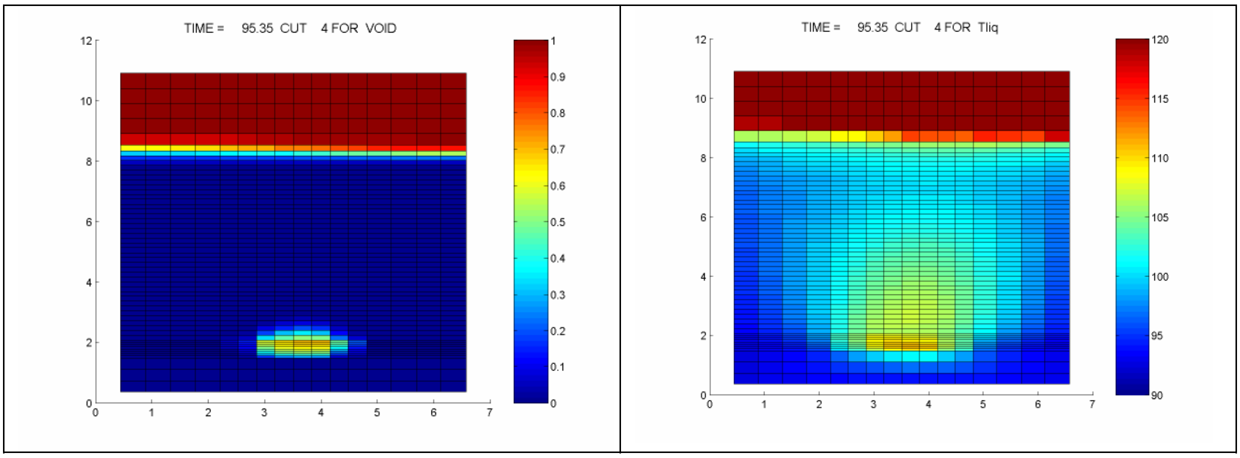


Figure 2. Pool void fraction and temperature distribution in PORFLO simulation.

In the particle bed dryout experiment, heat is generated by electric current conducted through a heating coil. The test rig is located inside a constant pressure vessel and the water level above the heated bed is controlled with a combination of level measurement and a feedwater pump. The final separation of bubbles takes place on the water level. The calculation geometry has cylindrical symmetry. Vapor condensation on the pressure vessel wall and heat conduction through the steel walls have to be considered in the model as well. The bed material itself is 0.4 mm to 3 mm diameter oxide particles, which have a specific heat capacity and conductivity. The case specific input definition contains all these features.

In a steam generator, the heat input into the fluid via heat transfer tubes has to be described. One nodalization is used for the tubing and another for the condensing pool. The input specification has to define the linking between individual primary and secondary nodes. On the secondary side, components exist for feedwater control and level measurement. The pressure boundary condition for the vapor space is essential.

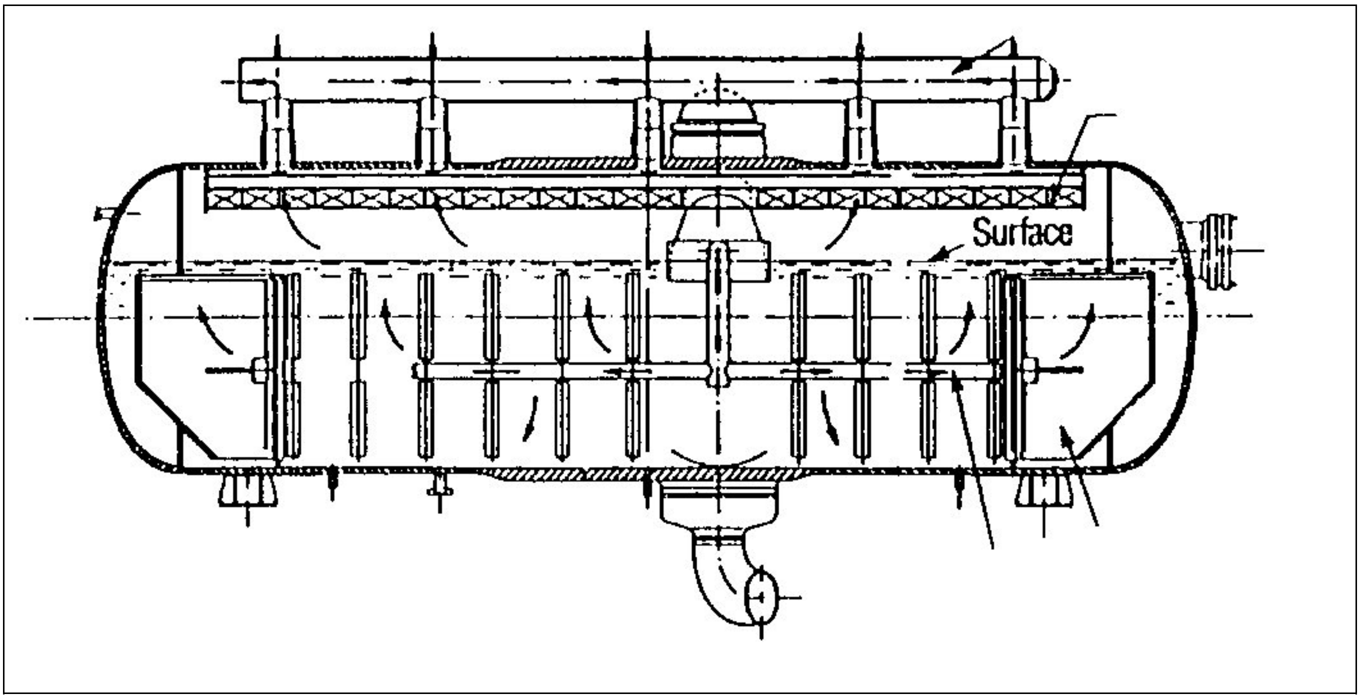


Figure 3. Horizontal steam generator applied in the Loviisa VVER-440 plant..

The generation of the three-dimensional reactor core nodalization is tested first with the TRAB-SMABRE code system. After satisfactory results, the input generation is adjusted for the PORFLO as well.

For the BFBT calculation, the specific mesh generation algorithm requests the main parameters for the bundle structure, like the rod diameter, pitch, rod type, channel width and rounding. The initialization part defines the desired computational mesh, matching for all rods a similar meshing. Inside the rods three different fields exist: uranium oxide, gas gap and cladding. For the surface nodes, between solid and fluid nodes, the heat transfer characteristics need to be defined and calculated dynamically every time step. Critical power prediction needs specific handling. The inlet condition for the bundle flow is defined by the inlet velocity, and the output boundary condition by constant pressure. The output processing needs case specific handling as well.

A minor code modification was implemented to incorporate a more realistic momentum boundary condition for the incoming flow. Otherwise, mass errors grow prohibitively large. Phase separation is calculated from the drift flux model and seems to be realistic.

Also, a new indexing scheme of grid cells and faces was implemented throughout the PORFLO code, making future developments easier.

After experience has been gained from tens of applications, one could consider the possibility for more generic user input. For example, the present user input of the Fluent code is not user-friendly enough for versatile application.

4 Geometry interface & meshing for fuel bundles

An easy-to-use preprocessor program for fuel bundle geometry data import, meshing and geometry-related initialization of the porosity model was completed. Various fuel designs with different channel boxes, arrangements of fuel rods and water rods, and power distributions are now easily imported. Meshing is not exactly body-fitted, but nevertheless always fitted to the rods and sub-channels in the best possible way. Mesh size can be varied easily. Various data on heated / unheated areas and volumes of the structures are delivered for each mesh cell to be used in the PORFLO simulation.

5 Basics of PORFLO application to the BFBT benchmark

Currently PORFLO has been optimized to simulate a BWR fuel bundle. Single-phase fluid enters the fuel bundle from the bottom and a two-phase mixture flows out of the bundle at the top. The calculations are performed on a non-uniform orthogonal grid and velocities are solved on a backward staggered grid to prevent the *checkerboard pressure field* –effect. Since PORFLO is a porous media model, the grid need not follow the structural interfaces; instead, porosities are defined as a fraction occupied by the fluid from the total volume.

6 Pressure-Velocity Coupling

The former solution procedure combined the equation for mixture mass with the three mixture momentum equations to yield a single system of equations for pressure. The resulting pressure field is then used to obtain the velocity field directly. One of the perks of this method is that the amount of calculations needed for solving a problem with one system of equations is much less than a problem with several systems of equations. Added to this, the former solution procedure does not require iteration within the time step.

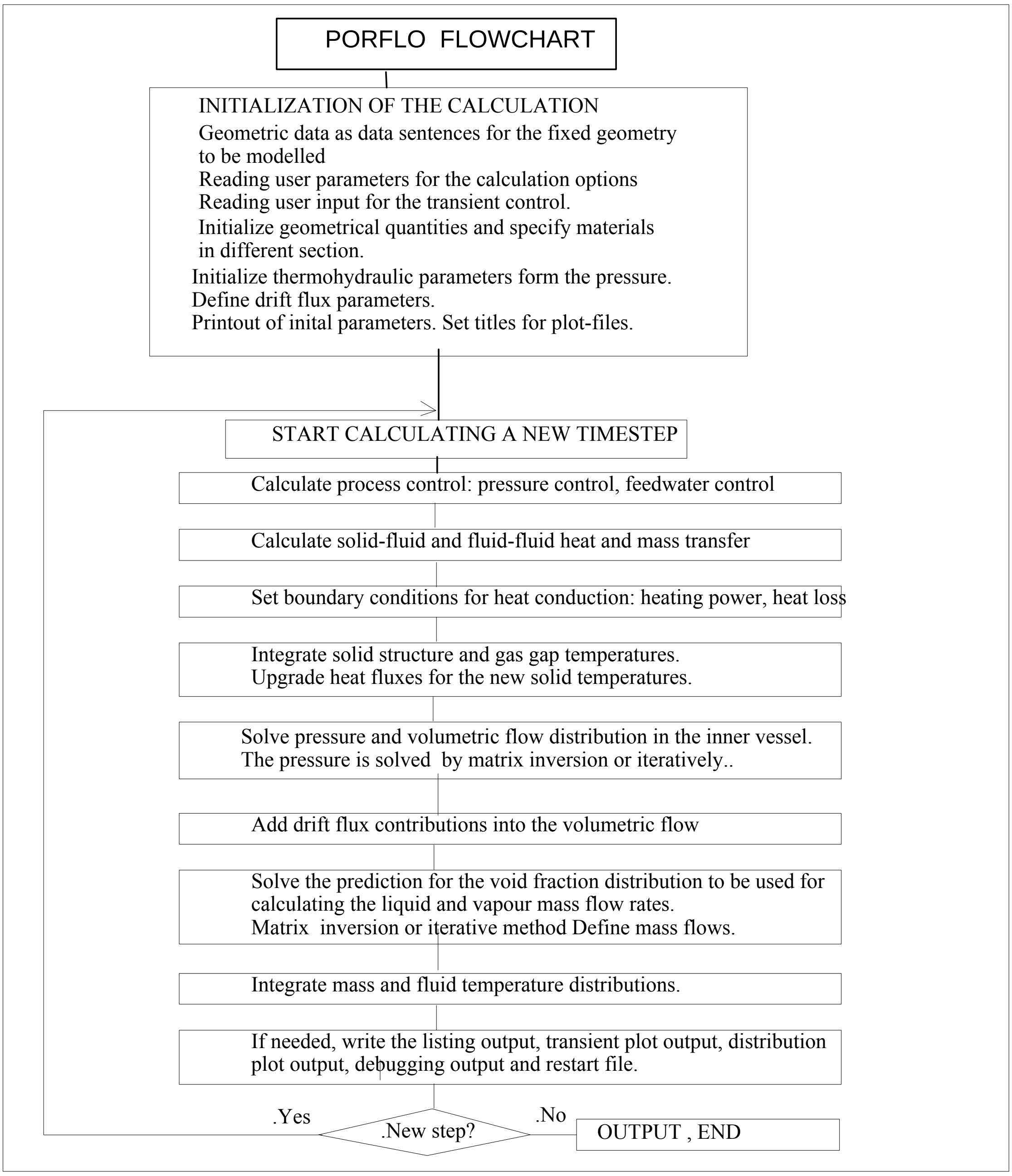

PORFLO FLOWCHART
INITIALIZATION OF THE CALCULATION
Geometric data as data sentences for the fixed geometry to be modelled
Reading user parameters for the calculation options
Reading user input for the transient control.
Initialize geometrical quantities and specify materials in different section.
Initialize thermohydraulic parameters form the pressure.
Define drift flux parameters.
Printout of inital parameters. Set titles for plot-files.
START CALCULATING A NEW TIMESTEP
Calculate process control: pressure control, feedwater control
Calculate solid-fluid and fluid-fluid heat and mass transfer
Set boundary conditions for heat conduction: heating power, heat loss
Integrate solid structure and gas gap temperatures.
Upgrade heat fluxes for the new solid temperatures.
Solve pressure and volumetric flow distribution in the inner vessel.
The pressure is solved by matrix inversion or iteratively..
Add drift flux contributions into the volumetric flow
Solve the prediction for the void fraction distribution to be used for calculating the liquid and vapour mass flow rates.
Matrix inversion or iterative method Define mass flows.
Integrate mass and fluid temperature distributions.
If needed, write the listing output, transient plot output, distribution plot output, debugging output and restart file.
.Yes
.New step?
.No
OUTPUT , END

Figure 4. Original SMABRE type of algorithm.

Another approach is to couple the pressures and velocities indirectly, which leads to an iterative procedure where the approximations for pressures and velocities are improved with every cycle.

The SIMPLE algorithm has been programmed in PORFLO to be optionally used in place of the older ‘SMABRE style’ method. An earlier implementation of SIMPLE gave good results in multidimensional calculation of one-phase flow. The advantage of SIMPLE is that the matrix diagonals in solution of the pressure and velocity corrections are stronger than in the SMABRE method. Test runs have been conducted in a Master’s thesis work.

7 SIMPLE algorithm

SIMPLE algorithm, Semi-Implicit Method for Pressure-Linked Equations, is included in the current version of PORFLO as its own subroutine. At the current configuration the subroutine serves as an independent part that solves the pressure and velocity distributions from a given set of data; in other words: there is no feedback from any other part of the program during the calculation of a time step.

Another approach would be to solve all other conservation equations inside the SIMPLE-iteration, which could be done with relatively little coding effort, if it seems necessary in the future. Solving all other conservation equations inside the iteration loop would enable the use of longer time steps, especially useful in transient calculations, since the solution procedure would become more implicit, as the pressure-velocity coupling would have a feedback from changes in pressure and heat exchange through local mixture densities.

Iterative procedures, like SIMPLE, require more calculations compared to direct pressure-velocity coupling, since firstly multiple systems of equations have to be solved during one iteration cycle, the momentum equations and the pressure correction, and secondly multiple iterations are needed to reach a converged solution. In this perspective it would seem unreasonable to use SIMPLE. However, fully implicit discretization can be applied to formulate the momentum equations and all the terms in the momentum equations, convection, diffusion and even turbulence, can be introduced without significant hardship. This cannot be said about the former solution procedure.

It is important to recognize that though SIMPLE makes no assumptions about the solver which the systems of equations are to be solved with, being an iterative procedure itself, the intermediate solutions of the iteration cycles do not have to be solved precisely; only the final solution is of importance. Therefore careful consideration of the convergence criteria can significantly reduce the amount of calculations needed to perform one time step, and hence the overall computational time is reduced as well.

7.1 Iteration procedure

SIMPLE and its several variants start with guessed pressure and velocity fields which are first input to the momentum equations to obtain improved values for the velocities. The improved velocities are used in the pressure correction equation, which is obtained by combining the mass and momentum conservation equations. The pressure corrections are used to yield corrected pressures and velocities, which are again used in the momentum equation at the start of the next cycle. The sequence of operations in SIMPLE algorithm is represented in figure 1. Though the current version of the code is used to simulate a steady state situation, the type of SIMPLE algorithm used is actually transient, since firstly the whole simulation marches forward in time and secondly transient terms are included in the governing equations; for example, inertia of the flow transferred from the previous time step is included in the momentum equations.

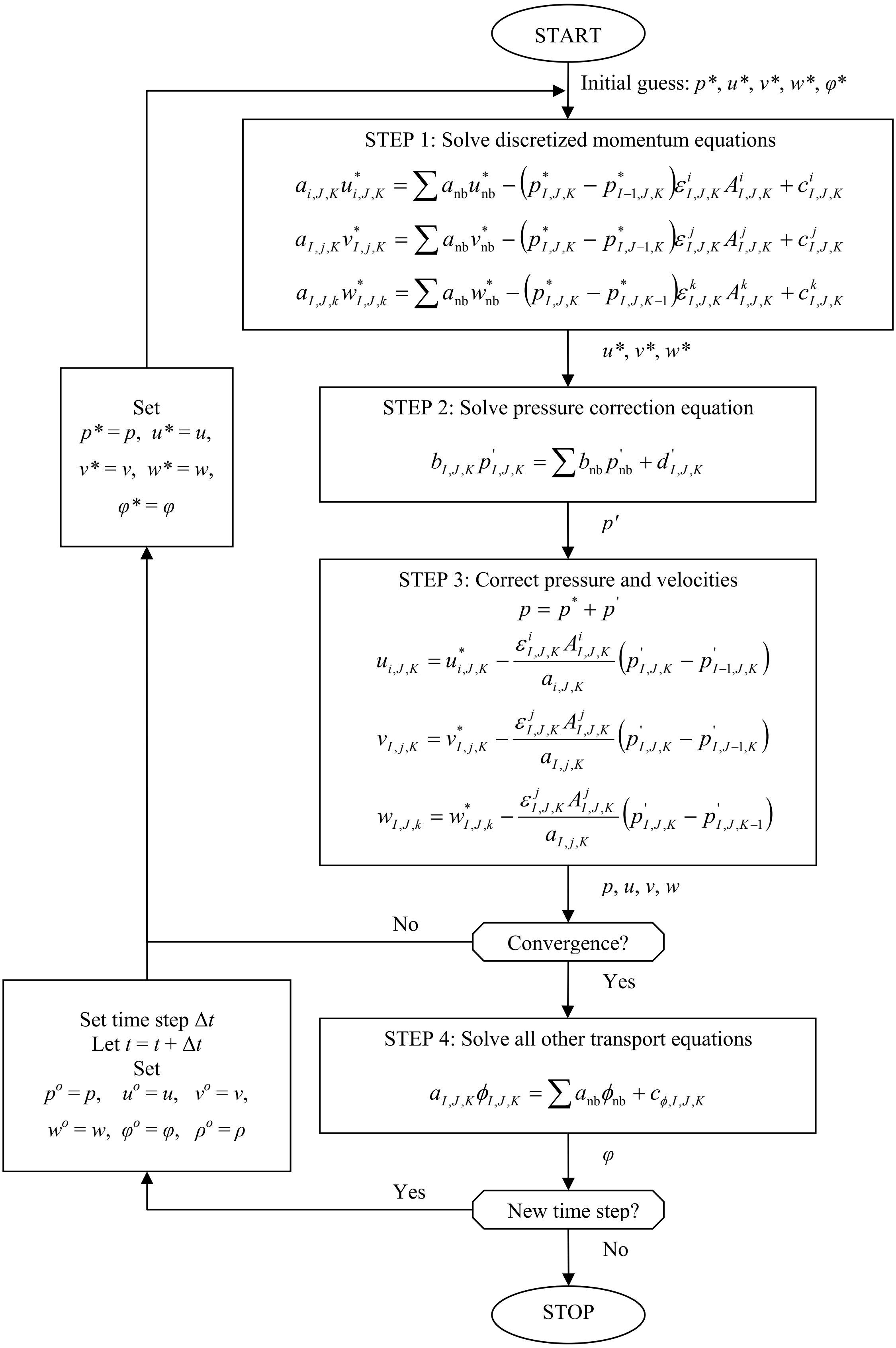

START
Initial guess: p^*, u^*, v^*, w^*, φ^*
STEP 1: Solve discretized momentum equations
$a_{i,J,K}u^*_{i,J,K} = \sum a_{\text{nb}}u^*_{\text{nb}} - \left(p^*_{I,J,K} - p^*_{I-1,J,K}\right)\varepsilon^i_{I,J,K}A^i_{I,J,K} + c^i_{I,J,K}$
$a_{I,j,K}v^*_{I,j,K} = \sum a_{\text{nb}}v^*_{\text{nb}} - \left(p^*_{I,J,K} - p^*_{I,J-1,K}\right)\varepsilon^j_{I,J,K}A^j_{I,J,K} + c^j_{I,J,K}$
$a_{I,J,k}w^*_{I,J,k} = \sum a_{\text{nb}}w^*_{\text{nb}} - \left(p^*_{I,J,K} - p^*_{I,J,K-1}\right)\varepsilon^k_{I,J,K}A^k_{I,J,K} + c^k_{I,J,K}$
u^*, v^*, w^*
STEP 2: Solve pressure correction equation
$b_{I,J,K}p'_{I,J,K} = \sum b_{\text{nb}}p'_{\text{nb}} + d'_{I,J,K}$
p'
STEP 3: Correct pressure and velocities
$p = p^* + p'$
$u_{i,J,K} = u^*_{i,J,K} - \frac{\varepsilon^i_{I,J,K}A^i_{I,J,K}}{a_{i,J,K}}\left(p'_{I,J,K} - p'_{I-1,J,K}\right)$
$v_{I,j,K} = v^*_{I,j,K} - \frac{\varepsilon^j_{I,J,K}A^j_{I,J,K}}{a_{I,j,K}}\left(p'_{I,J,K} - p'_{I,J-1,K}\right)$
$w_{I,J,k} = w^*_{I,J,k} - \frac{\varepsilon^j_{I,J,K}A^j_{I,J,K}}{a_{I,j,K}}\left(p'_{I,J,K} - p'_{I,J,K-1}\right)$
p, u, v, w
Convergence?
No
Set
$p^* = p,\ \ u^* = u,$
$v^* = v,\ \ w^* = w,$
$\varphi^* = \varphi$
Yes
STEP 4: Solve all other transport equations
$a_{I,J,K}\phi_{I,J,K} = \sum a_{\text{nb}}\phi_{\text{nb}} + c_{\phi,I,J,K}$
φ
New time step?
Yes
Set time step Δt
Let $t = t + \Delta t$
Set
$p^o = p,\ \ u^o = u,\ \ v^o = v,$
$w^o = w,\ \ \varphi^o = \varphi,\ \ \rho^o = \rho$
No
STOP

Figure 5. Transient SIMPLE algorithm.

8 SIMPLE algorithm Development

Thus far the most profound drawback of the solution algorithm has been the lack of diagonal dominance of the pressure correction equation when it is formulated assuming incompressible flow, a presumption which under BWR steady state conditions seems quite valid. The lack of diagonal dominance renders most iterative solvers in their basic form useless when solving the pressure correction equations. Direct solvers, on the other hand, become inefficient compared to iterative solvers when the number of calculation nodes is increased.

If transient calculations, where sudden drops in pressure occur, were to be performed on this code, it would be best to reconsider the sequence of solving the equations. Moving all other conservation equations inside the SIMPLE-iteration could be practical in this perspective, as was mentioned before.

8.1 The effect of manipulating the pressure correction matrix

A quick test was made where the diagonal dominance of the pressure correction matrix was artificially increased. Such modifications can be done only if the final converged solution remains unchanged. In this case the pressure correction can be seen as a tool to nudge the pressure field towards the final solution. When velocity and pressure fields satisfy each other, the pressure corrections, given by the pressure correction equations, approach zero, hence the small increase in the diagonal term has no effect; only the path to reach the converged solution has been changed.

The pressure correction equation was solved using a direct solver, Gaussian elimination, and the number of SIMPLE-iterations was observed with each time step. Though the aim of this test ultimately is to be able to use an iterative method to solve the pressure correction equations, a direct method was chosen in order to eliminate the effect of convergence criterion of the iterative solver in the number of SIMPLE-iterations.

The test showed that the artificial increase in the diagonal dominance increased the amount of SIMPLE-iterations with a factor of 10-1000. In a situation where most of the computational time is spent solving the pressure correction equations, this would mean that an iterative solver should be able to solve the pressure correction equations in 1/10 to 1/1000 of the time compared to Gaussian elimination.

8.2 The effect of under-relaxation

SIMPLE algorithm needs under-relaxation, due to both the iterative nature of the procedure and the nonlinearity of the equations, in order to obtain a converged solution. Both the pressure corrections and the velocities are under-relaxed. One of the first things to do was to determine a robust set of under-relaxation parameters and the range where the solution seems to converge. A more detailed study is unnecessary, at the moment, since the rate of convergence varies considerably with different geometry and flow conditions. The full effect of the under-relaxation parameters on computational time is best seen when the number of SIMPLE-iterations are compared throughout the simulation.

At the moment the most robust under-relaxation parameters seem to be, $\alpha_p = 0.5$ and $\alpha_u = 0.5$, where the under-relaxation factors are for pressure and velocity, respectively. A slight increase in any of them seems to decrease the number of SIMPLE-iterations, but at the same time the fluctuation of pressure corrections and velocities is increased within the iteration procedure.

8.3 SIMPLE variants

When the basic SIMPLE algorithm has been coded, the introduction of other SIMPLE variants, such as SIMPLEC, SIMPLE-Consistent, and SIMPLER, SIMPLE-Revised, is relatively straightforward; only minor modifications or additions are needed. The current version of the code includes both SIMPLEC and SIMPLER algorithms. At the current configuration, though relatively little testing has been done, SIMPLEC seems quite promising, since it requires virtually no more computation effort than SIMPLE and most papers that have compared the performance of the two algorithms report a significant reduction in the number of iterations per time step.

However, if the solution of other conservation equations is performed within the SIMPLE-iteration, SIMPLER might prove to be more useful. At the present the computational effort to calculate a certain amount of iterations is doubled when using SIMPLER, without similar reduction in number of iterations needed to reach a converged solution.

8.4 Assertions

Some assertions have been included in the code to monitor the iteration procedure. One of them simply monitors the maximum values of velocities and compares them with Courant's criteria. Though fully implicit discretization has been used to formulate the momentum equations, old values of local mixture densities still have to be used, which makes the overall procedure stringent to Courant's criteria. Another assertion aims to detect inconsistent mass flow rates in momentum equations over consecutive horizontal planes. Due to an averaging process of the velocity node mass flow rates the latter is not trivial, but a necessity, in order to reach a valid solution.

8.5 Linear solver & preconditioning

At the moment the effect of preconditioning on the usefulness of an iterative solver, namely CGS, which is a Krylov subspace method, is being studied; the reasons for this being the possible savings in computational time and the ability to solve bigger systems.

In 3D problems, the size of the linear set of difference equations easily grows prohibitively large. For example, a 100 x 100 x 100 Cartesian computational grid has one million grid cells, and thus the coefficient matrix A of the linear set is a million-by-million one. Solving by basic matrix manipulations is not possible because of the memory needed for matrix elements and the CPU time for float operations. On the other hand, in iterative methods the most CPU-intensive basic operation is usually the product of A and some vector. Then, it is sufficient to store only the non-zero elements of A and the number of float operations basically grows linearly with the number of grid cells; of course, additional work may also result in the form of more iteration rounds needed for convergence in a larger problem. Furthermore, it is straightforward to parallelize the computation.

PORFLO is a code for solving 3D fluid dynamics problems for water and steam. A fast and memory-efficient iterative solver was needed for larger problem sizes. The coefficient matrix may be non-symmetric. A preliminary comparison study of some appropriate well-known iterative solver algorithms (GMRES, BCG, QMR, CGS, BICGSTAB, CGNR) was performed with typical equation sets of PORFLO. The CGS (Conjugate Gradient Squared) algorithm turned out to be a fast and robust one, and has since been extensively used in PORFLO.

CGS belongs to the class of so-called Krylov subspace methods [Saad 2003, p. 151]. An m-dimensional Krylov subspace has the form κ_m (A, v) = span {v, Av, A^2v, …, A^{m-1}v} and contains all the vectors x of R^n that can be written as x = p(A)v, where p is a polynomial of degree m-1 or less. Essentially, the inverse A^{-1} is approximated by p(A), where p is a suitable polynomial. This is accomplished by projecting the problem onto a Krylov subspace and the approximation to the solution vector is extracted from the subspace. The dimension m of the subspace increases during the iteration. The residual is minimized in each successive subspace. Theoretically, complete convergence is reached when m equals problem size. The m basis vectors tend to become linearly dependent, which can be corrected by some orthogonalization scheme or, like CGS, the intrinsically nonorthogonal Lanczos biorthogonalization algorithm for non-symmetric matrices. As a historical note, the Krylov methods were discovered in the 1950s but abandoned for decades because of the inherent loss of linear independence. Basic algorithms based on Lanczos biorthogonalization require a matrix-by-vector product by both A and A^T, but the latter only contributes to certain scalars needed by the algorithm. The CGS algorithm avoiding the operations with A^T was developed by Sonneveld in 1984.

The CGS algorithm can be derived from BCG (biconjugate gradient method) by certain algebraic manipulations. BCG in turn can be derived from the Lanczos biorthogonalization procedure. For PORFLO, the CGS algorithm is provided by the subroutine cgs_solver_sparse. For the multiplications by matrix A, the memory- and CPU-efficient subroutine matmul_sparse is used. These multiplications consume most of the CPU time of the algorithm. Potential failures to continue iterating include scalar scaling factors α and β becoming zero or infinite. Other known problems are build-up of rounding errors or even floating point overflow due to the squaring of the residual (which is inherent to CGS), especially when convergence is irregular. There are variants of CGS with smoothing of the convergence behavior.

The BFBT benchmark application revealed new problems in the PORFLO iterative linear solver, which was previously believed to be universal, stable and robust. In the basic (older) solution method employed in PORFLO, the pressure solution must be very accurate in order to calculate a realistic flow field. Currently, preconditioning techniques are being studied to overcome the difficulties. Possibilities include suitable multiplications of the equations or other kind of explicit preconditioning matrix used inside the solver algorithm. Other techniques include artificially increasing the diagonal dominance, tuning the CGS shadow residual vector, or explicit residual refinement. Unfortunately, no universally reliable method for solving an arbitrary large linear system exists today, and so development must partially proceed through trial and error for each specific problem.

There are frequent problems with large matrices where the solver either stagnates at a nearby point of the correct solution vector, or starts to diverge after visiting a nearby point in the solution space. Especially the pressure matrix is very weakly diagonally dominant, with only 0.01 % dominance. The void prediction is easier to iterate, because the diagonal dominance is in the range of 2 - 20 %. The iterative matrix inversion is the only possibility for inverting large matrices. The simplest methods, Jacobi, one-directional Gauss-Seidel and two-directional Gauss-Seidel, work already for the void fraction prediction with a reasonable convergence in less that 50 iteration steps. The convergence of the pressure matrix inversion is more complex. 5000 iterations is not sufficient with Jacobi and Gauss-Seidel iterations. Convergence can be achieved with disturbed equations (converges easily, if only small changes) with the ADI method (altering direction implicit) for the pressure matrix in less than 2000 iterations. But the ADI solution includes axis equations solved with a 3-band matrix inversion implicitly.

Better convergence is striven for by iterative matrix inversions. At present, good convergence characteristics have been obtained for the 100000 mesh points nodalization. The problematic part in these solutions is the so-called CGS shadow residual, which is commonly under scientific discussions in international meetings. Probably the best way forwards is understanding iterative solutions detailed enough. In parallel with this work the set up of equations may need to be improved. The problematic term is the mass error in the pressure equation. The question is, how efficiently the mass error term is put into the equations for the next time step. When the convergence problems for the iterative solution have been clarified, the whole work can be concentrated on making then physical model more exact.

8.6 Summary of some major developments of the SIMPLE approach

During each SIMPLE iteration round, momentum has to be calculated / solved. It used to be calculated pointwise, using old (explicit, known) values of velocity in the neighboring cells.

Now momentum is actually solved from a system of equations, one of the three components at a time.

Under-relaxation of velocities has been implemented and is now performed at the end of each SIMPLE iteration for the corrected velocities, one at a time. There was an older approach of performing the under-relaxation in connection with solving the momentum equations. Separated under-relaxation makes it easier and clearer to set the momentum equations and monitor the SIMPLE iterative procedure. The relaxation coefficients of velocities are not (or, are only optionally) fed into the pressure correction equation, as decreasing them could increase the pressure correction inappropriately.

The criterion of accepted convergence in SIMPLE is the maximum proposed pressure correction (typically 0.2 Pa). Other possibilities include velocity or pressure changes from one iteration round to another.

8.7 Debugging

As might be expected, most of the coding effort has been spent on debugging the program. Some of the most severe errors found include saving the improved velocities after momentum equations. A simple mistake in an if-sentence prevented the velocities nearest to the fuel rods from being updated during the iteration, which lead to decrease in velocity near the fuel rods. In a situation where the friction has been set uniform across the whole domain, this was clearly illogical. However, the source of this was quite hard to backtrack, since due to the strong coupling between pressure correction and momentum equations the symptoms were seen everywhere.

9 Specification of the BFBT benchmark problem

The BFBT problem tackled in TRICOT so far is that of Phase 1 (void distribution benchmark), Exercise 1 (steady-state sub-channel grade benchmark). In this problem, the boiling flow does not change in time, except of course for the two-phase turbulent fluctuations. The simulated void fraction field is, in the end, expressed as average values in a mesh whose elements correspond to the sub-channels between fuel rods. Simulation is compared with measurements of an X-ray computer tomography scanner located at 50 mm above the heated length. The spatial resolution of the measurements is as fine as 0.3 mm x 0.3 mm, but the time taken by one scan is 15 s, so that only a steady-state time average is acquired. The simulations by PORFLO start with a cold bundle immersed in water-only flow. After turning on the heating power, the heater rods reach their operating temperature in less than 10 s, after which one may observe an increasing simulated void fraction. A detailed comparison with measurements was not yet performed, as several refinements of the simulation are still in progress.

10 BFBT workshop in may 2007

Jaakko Miettinen participated in the BFBT workshop in Paris to discuss the problems of the simulation with other experts. Whereas isolation condensers and steam generators are macro scale facilities for the PORFLO application, and the particle bed dryout experiment is a medium scale facility, the BFBT experiment can be considered as a micro scale application for the PORFLO code. The aimed mesh size in the cross section is 1.5 mm. The resolution of the most accurate measurements for the void fraction is 0.3 mm. But this mesh size would be too small, because the 3.6 m long rod bundle needs to be modeled in the vertical direction as well with 50 - 200 mesh points.

The benchmark includes several tasks and it is based on the accurate measurements for the BWR test bundles carried out at JNES (Japan Nuclear Safety organization). The phase I of the benchmark includes the prediction for the void fraction distribution as four exercises:

Exercise 1 - Steady state sub-channel average void fraction
Exercise 2 - Steady state sub-channel microscopic void fraction
Exercise 3 - Transient macroscopic void fraction
Exercise 4 - Uncertainty analysis for the void fraction

Phase II of the benchmark includes prediction of critical heat flux as four exercises:

Exercise 0 - Steady state pressure drop in the bundle
Exercise 1 - Steady state critical power in the bundle
Exercise 2 - Transient critical power in the bundle

Exercise 3 - Uncertainty analysis for the critical power

The benchmark project as a whole has been started already two years earlier and VTT had no possibilitiy to participate in the work from the beginning. But after participating in the latest project meeting of May 8-9, the benchmark coordination promised that VTT and other late-started organizations can still participate in all parts of the exercise. In TRAB-SMABRE development the bundle phenomena need to be described in such detail that the BFBT benchmark with modeling of subcooled boiling, void fraction distribution and dryout mechanism greatly supports the system code development as well. Thus participation in the complete work can be greatly emphasized.

11 Results of BFBT simulations

At the moment the simulations reach several seconds in real time to a point where significant boiling occurs. The effect of viscosity was studied in the latest runs. One of them did not include viscosity and in the other viscosity varied in horizontal planes along the length of the fuel bundle. The one without viscosity reported higher local values of both void fractions and horizontal flow rates, whereas the other with vertically varying viscosity had more even distribution of the variables in question. Maximum local void fractions have reached values from 0.6-1.0 depending on whether viscosity is included in the simulation or not.

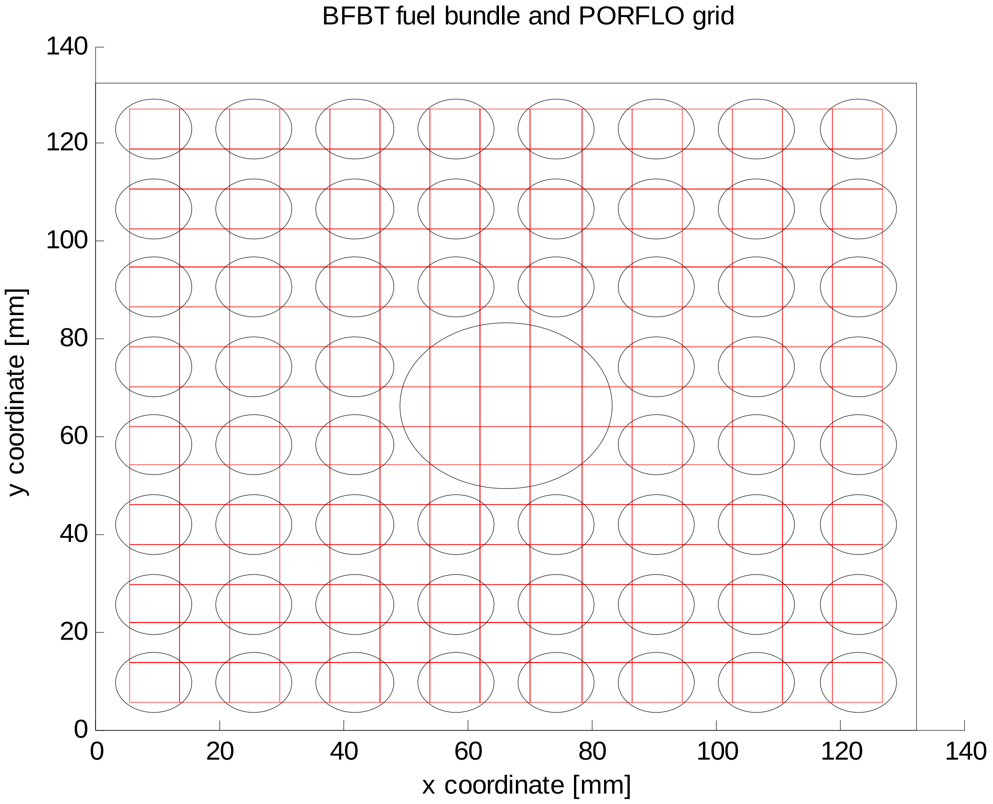


Figure 6. The most coarse mesh studied for the BFBT calculation

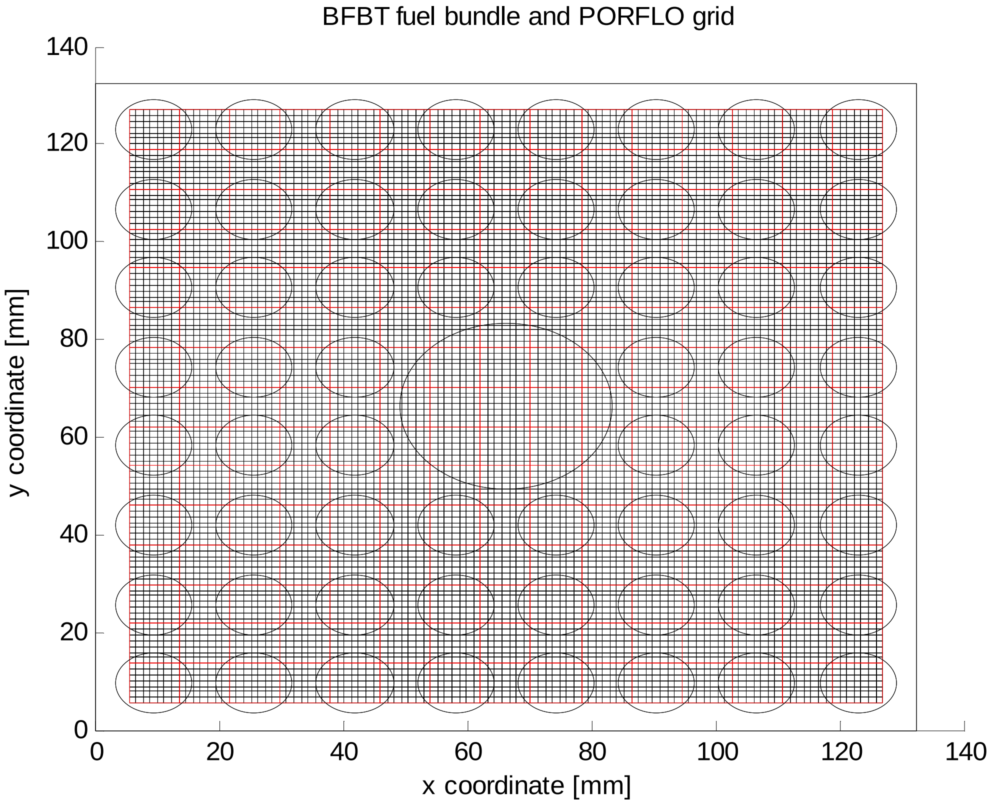


Figure 7. The fine mesh used for the subchannel studies for the BFBT calculation. The present results are for the 2 x 2 test bundle used for the numerical study with different solutions.

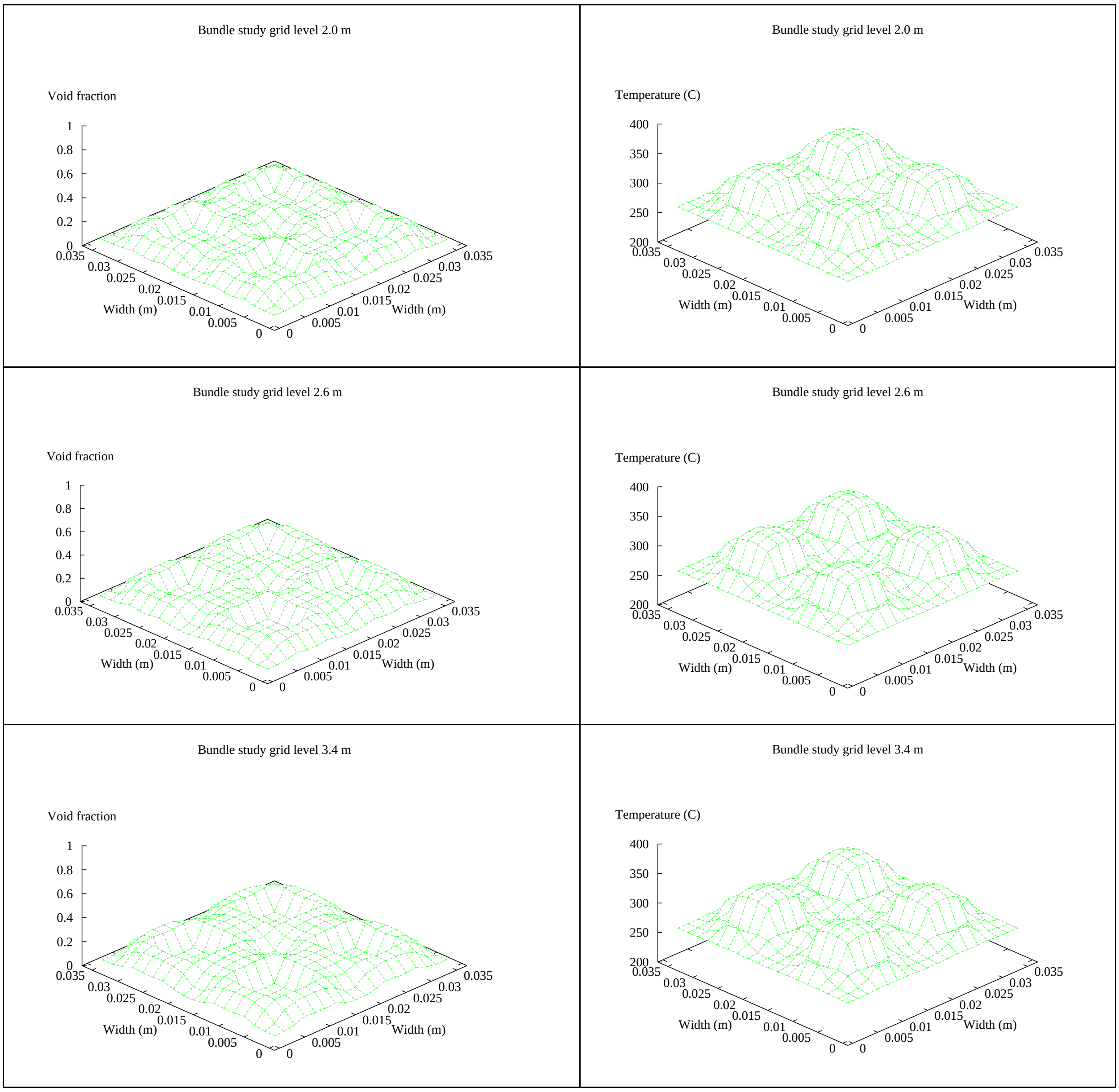


Figure 8. Void fraction and temperature distributions on different axial levels for the numerical study 2x2 bundle.