NUMERICAL METHOD FOR THREE DIMENSIONAL STEADY-STATE TWO-PHASE FLOW CALCULATIONS

P. Raymond, I. Toumi (CEA)
CE-Saclay DMT/SERMA
91191 Gif sur Yvette, FRANCE
Tel: (331) 69 08 26 21 / Fax: 69 08 23 81

ABSTRACT

This paper presents the numerical scheme which was developed for the FLICA-4 computer code to calculate three dimensional steady state two phase flows. This computer code is devoted to steady state and transient thermal hydraulics analysis of nuclear reactor cores [1,3]. The first section briefly describes the FLICA-4 flow modelling. Then in order to introduce the numerical method for steady state computations, some details are given about the implicit numerical scheme based upon an approximate Riemann solver which was developed for calculation of flow transients. The third section deals with the numerical method for steady state computations, which is derived from this previous general scheme and its optimization. We give some numerical results for steady state calculations and comparisons on required CPU time and memory for various meshing and linear system solvers.

I. THE FLICA-4 FLOW MODELLING

The two-phase flow model of the FLICA-4 computer code consists in

- a mixture mass conservation equation
- a mixture momentum balance equation
- a mixture energy balance equation
- a phasic mass balance equation which allows the calculation of subcooled boiling flow.

The two phases are assumed to be at the same pressure, and the drift velocity between vapor and liquid phases is given by algebraic relationships.

The modeling of the diffusive terms deals with viscous and turbulent stresses, thermal conduction and turbulent diffusivity.

The required closure relationships for the various equations are :

Phasic Mass Balance:

- Wall vaporization model
- Bulk condensation
- Turbulent mass diffusivity

Momentum:

- Wall friction model
- Singular pressure drop
- Mixing grids effects modelling
- Drift velocity model
- Eddy viscosity model

Energy:

- Wall heat transfer model
- Eddy diffusivity model

II. THE FLICA-4 NUMERICAL SCHEME

The FLICA-4 numerical scheme for transient computations is based on an approximate Riemann solver for the discretization of inviscid flux terms and on a centering scheme for diffusive terms.

Let U be the set of conservative variables, and considering an integral form of the partial differential equations on a computational domain Ω, a finite volume method leads to the following approximation of the set of balance equations :

$$U_i^t - U_i^{t-1} + \Delta t \sum_j (\Phi_I(U_i^t, U_j^t) - \Phi_V(U_i^t, U_j^t)) = S(U_i^t) \qquad \text{(II.1)}$$

where $S(U_i^t)$ is the vector of source terms, and $\Phi_I(U_i^t, U_j^t)$ and $\Phi_V(U_i^t, U_j^t)$ denote the inviscid and viscous contributions respectively to the flux on the cell Ω_i in direction of j, neighbor of i.

For the following set of conservation equations describing an inviscid flow :

$$\partial_t U + \partial_x f(U) = 0 \qquad \text{(II.2)}$$

a usual way to define upwind numerical fluxes is to solve local one dimensional Riemann problems at cell interfaces. The main concept of Roe's approximate Riemann solver is to be the exact solution of the local linearization of this initial nonlinear problem :

$$\partial_t U + A_{ij}\partial_x U = 0 \qquad (II.3)$$

where A_{ij} is an average Jacobian matrix satisfying the property

$$f(U_j) - f(U_i) = A_{ij}(U_j - U_i) \qquad (II.4)$$

The main problem is to find an average Jacobian matrix A_{ij} satsfying Eq. (II.4). Such a matrix is known as a Roe-averaged matrix and was first constructed by Roe for the Euler equations with ideal gases [5]. We follow a weak formulation of Roe's approximate Solver introduced inreference 6 to construct a Roe-averaged matrix for the FLICA-4 two-phase flow model [4].

Once the matrix A_{ij} is constructed, the numerical inviscid flux between meshes i and j will then be given by the following expression

$$\Phi_I^{ij} = \frac{1}{2}\left(f(U_i^t) + f(U_j^t) - \left|A_{ij}^t\right|(U_j^t - U_i^t)\right) \qquad (II.5)$$

The matrix A_{ij}^t can be split into :

$$A_{ij}^t = (A_{ij}^t)^+ + (A_{ij}^t)^- \qquad (II.6)$$

with

$$(A_{ij}^t)^+ = R_{ij}^{-1}\left(\frac{|\Lambda_{ij}| + \Lambda_{ij}}{2}\right)R_i$$
$$(A_{ij}^t)^- = R_{ij}^{-1}\left(\frac{|\Lambda_{ij}| - \Lambda_{ij}}{2}\right)R_i \qquad (II.7)$$

Λ_{ij} is a diagonal matrix containing the eigenvalues of A_{ij} and R_{ij}^{-1} and R_{ij} are matrices which contain the right and left eigenvectors of A_{ij}.

It is to be noted that both $(A_{ij}^t)^+$ and $(A_{ij}^t)^-$ have non-negative eigenvalues.

The flux Φ_I^{ij} can be written in either of the two equivalent form :

$$\Phi_I^{ij} = f(U_i^t) - (A_{ij}^t)^-(U_j^t - U_i^t)$$
$$\Phi_I^{ij} = f(U_j^t) + (A_{ij}^t)^+(U_j^t - U_i^t) \qquad (II.8)$$

Using the first expression for the flux through the interface $\partial\Omega_{ij}$ and the second for the flux through the opposite interface, leads to a flux balance equation of the form :

$$\frac{U_i^t - U_i^{t-1}}{\Delta t} + \left(\sum_k C_k^t\right)U_i^t - \sum_k (C_k^t U_k^t) = 0 \qquad (II.9)$$

where the sum is over all neighbors k of Ω_i. The set of resulting equations is both conservative and positive because all the matrix coefficients involved have non-negative eigenvalues.

To solve for U^t in (II.9), one formally needs to solve a set of nonlinear algebraic equations iteratively. The iterations needed to solve this system of nonlinear equations, at each time step, may obviate the savings gained with the ability to take larger time steps. One way to avoid this is to linearize the implicit operator by estimating the matrices C_k^t at the previous time, and solve the linearized form by other means.

This linearization in time, destroys the conservation property of the scheme. Thus, for transient calculations, a two step method is used. The first step (predictor) computes an approximation of the solution by estimating the linearized matrices at the previous time. The second step uses this estimation to compute fluxes and to solve the balance equations.

III. STEADY-STATE NUMERICAL METHOD

For industrial computations, it is necessary to determine accurate initial conditions which are compatible with the transient numerical treatment. One classical way to compute a steady state is to perform a transient computation with constant boundary conditions. Generally, this technique leads, after a finite time, to a steady state if the numerical scheme possesses good conservative properties. The major disadvantage of this method results from the fact that generally it is expensive in CPU time. The steady state numerical method which is derived from the previously described numerical scheme permits this obstacle to be overcome.

A. Delta Formulation

We introduce a delta formulation

$$\delta U_i = U_i^t - U_i^{t-1} \qquad (III.1)$$

Then, substituting (III.1) into (II.9) and using a first order approximation for the source terms

$$S(U_i^t) = S(U_i^{t-1}) + \frac{\partial S}{\partial U}(U_i^{t-1})\,\delta U_i + O(\Delta t^2) \qquad (III.2)$$

we get the following linear system

$$\left(\frac{I}{\Delta t} + \sum_k C_k^{t-1} - \frac{\partial S}{\partial U}\right)\delta U_i - \sum_k C_k^{t-1}\delta U_k =$$
$$S(U_i^{t-1}) - \sum_k \Phi_I(U_k^{t-1}) - \sum_k \Phi_V(U_k^{t-1}) \qquad (III.3)$$

A first remark can be made on the construction of the implicit operator : the right hand side of (III.3) contains a conservative discretization of the balance equations of the two-phase flow model. It follows that the steady-state solution obtained when the time variation of conservative variables leads to zero, is

- consistent with the conservation form
- independent of the time step used in the iterations
- a spatially first-order accurate approximation to the steady-state of the partial differential equation. However, higher-order approximations can be achived with the same implicit operator, by using higher-order extension of Roe's numerical flux in the right hand side of (III.3).

Consequently, the second step (conservation step) of the numerical scheme is not necessary to compute a steady state flow.

A second remark on the implicit scheme comes from an examination of the CPU time distribution

between the various phases of the resolution of the predictor step. One can consider three main phases :

- The matrix coefficients computation
- The matrix preconditioning
- The linear system resolution

B. Direct Methods

If the linear system is solved by using a direct method (LDU decomposition for a band matrix), the second phase which approximately varies with the square of the number of unknowns is generally the more expensive phase. Due to the positivity of the Roe linearization, it can be shown that it is not necessary to perform the matrix coefficient computation and the preconditioning at each time step,. Then a great amount of CPU time can be gained especially for a mesh with a large number of nodes.

C. Iterative Methods

For a very large number of nodes, the linear system can be solved directly but in many cases iterative methods are more efficient then direct ones, especially when dealing with large sparse systems. In the FLICA-4 computer code, iterative methods are available to solve large linear systems : a conjugate gradient squared (CGS) method with partial preconditionning and the Generalized Minimal Residual(GMRES) algorithm [7].

The Jacobian matrix derived from system (III.3) is a block tridiagonal matrix due to the axial meshing. The implementation of CGS and GMRES in FLICA-4 takes full advantage of the highly sparse regular structure of the matrix resulting from the implicit operator.

In order to obtain a good convergence rate, a preconditioning have been implemented. The diagonal preconditionning which consist in a complete LU factorization of the main diagonal blocks and an Incomplete Block LU factorization (IBLU) for which the matrix preconditionning is :

$$K = (L+D)\,D^{-1}\,(D+U)$$

Where D is the diagonal matrix, L is the block lower matrix and U the upper block matrix.

Due to the partial preconditionning of these two methods the gain in CPU time is less important than for direct methods if the matrix coefficients and the preconditioning are not performed at each time step.

D. Steady State Algorithm

As said in section II-A, the Roe linearization of fluxes is consistent with the conservation form of the flow equations. In particular, for the steady state regime, the Roe linearization leads to :

$$\nabla f(U) = [A]\,\partial_x U = S(U)$$

The first remark resulting from the principles of the Roe linearization is that it is not necessary to compute the conservation step in order to reach a steady state.

A second remark is due to the fact that the Roe linearization matrix is constructed in order to be positive definite. Due to the fact that in the physical model, dissipative terms will lead to a steady state flow regime if there are not nonlinear instabilities, and if the linearization of these terms preserves the dissipative behaviour of the numerical scheme, the previous linear system will converge to a steady state regime if the matrix [A] is supposed to be a constant matrix.

Since variations in the definition of matrix coefficients between two time steps lead to second order variation for the solution of the linear system (III.3), one can consider that it is not necessary to compute the matrix coefficients at each time step. Consequently it is possible to gain CPU time by saving the preconditioned form of the matrix and using it for more than one time step.

IV. NUMERICAL RESULTS

Various numerical results for steady state calculations will be presented and discussed in this section. Evaluation of the steady state numerical method of FLICA-4 are performed on various computational geometries : experimental rod bundle and a reactor core with assemblies [2]. Iterative and direct linear solvers efficiency will be analyzed.

Rod-bundle experiment steady state calculations

Classical steady state computations are performed in order to determine local physical parameters for interpretating Critical Heat Flux experiments performed with rod bundles test facilities.

For the OMEGA Facility (6x6 electrically heated rods, PWR geometry), the calculation is performed on 1/8 of the test section, because the radial power distribution is symmetric.

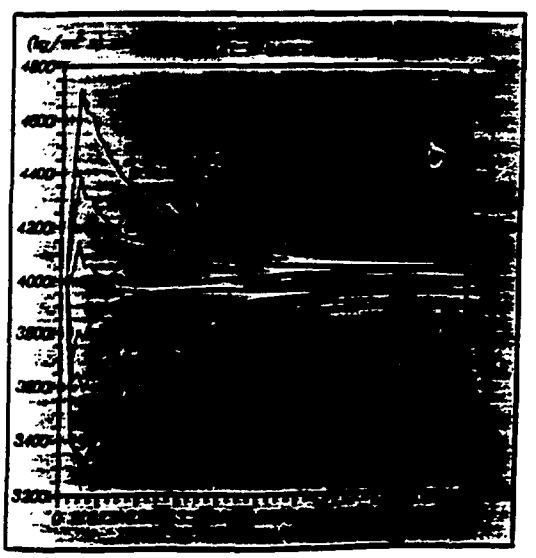
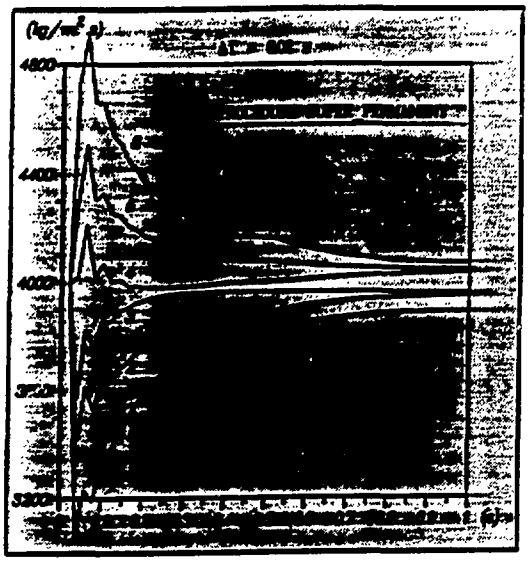


Fig 1. Mass Flow rate evolution for the six subchannels obtained by a transient calculation.

The computation field represents 6 sub-channels of the section meshed into 6×32 cells. Figures 1 and 2 give the convergence toward steady state conditions obtained by the FLICA4 classical transient calculation and by the Steady State algorithm. One can observe that the two methods converge in the same way to the same solution. Only the transient phenomena at the beginning of the computation, due to the propagation of the initial conditions, are slighty different.



CAMPAGNE OMEGA 87YZI4SU

Fig 2. Mass Flow rate evolution for the six subchannels obtained by the steady state algorithm.

1/8 PWR Core steady state calculations

To compare the performance of the steady state algorithm with various linear solvers, computations are performed on 1/8 of PWR core (26 assemblies). Axially, 3 zones are taken into account :

- Inlet zone without any vertical structures.
- Core zone with an axial cosine profile for power and a radial distribution.
- Outlet zone

The mesh is composed of 806 elementary cells. The steady state is computed with the algorithm previously described. The following table gives the CPU time distribution for one time step, obtained by using various linear solver.

Phase	CPU
Construction of the linear system	2,90 s
Matrix preconditioning	
direct method	8,90 s
CGS (diagonal)	1,75 s
GMRES (diagonal)	1,75 s
CGS (IBLU)	0,85 s
Resolution of the linear system	
direct method	0,05 s
CGS (diagonal)	19,74 s
GMRES (diagonal)	30,79 s
CGS (IBLU)	3,22 s

Table 1. (PWR 1/8 core): CPU time distribution for one time step.

The steady state computation is obtained after 51 time steps equal to 10^{-2}s. The total CPU time on a CRAY-2 computer which is necessary with the steady state algorithm is 27 s. By comparaison, a classical transient calculation needs 610 s, to calculate the same solution. The CPU time for solving the linear system at each time step is more important for iterative methods with diagonal preconditionning than for the direct method, but may be improved by using a better preconditioning like the Icomplete Block LU factorization (IBLU). However, iterative methods require less memory space than the direct one.

PWR Core steady state calculations

For this larger test case, the PWR core is fully meshed with 157 assemblies (Fig. 5) and 31 levels which gives 4867 elementary cells.

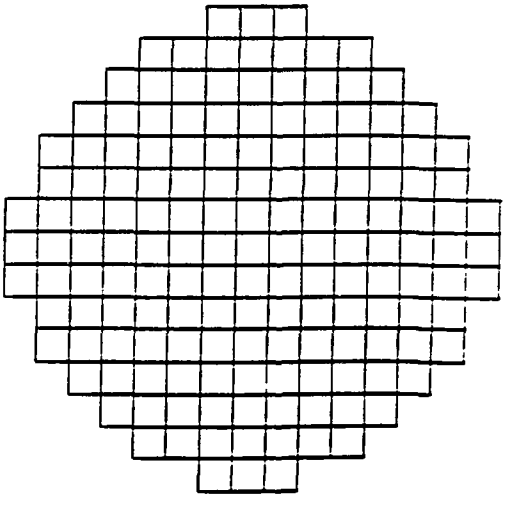


Fig.3 : PWR core : radial mesh

Table 3. gives the CPU time distribution for one time step, obtained by using various linear solver. The CGS

algorithm has been used with the diagonal and the IBLU matrix preconditioning

Phase	CPU
Construction of the linear system	7,25 s
Matrix preconditioning	
direct method	1320,00 s
CGS (Diagonal)	22,30 s
CGS (IBLU)	24,20 s
Resolution of the linear system	
direct method	0,40 s
CGS (Diagonal)	322,00 s
CGS (IBLU)	70,74 s

Table 2: PWR core : CPU time distribution for one time step.

The results in Table 2. show that:

1. The CGS method is superior to the direct one for this problem if the matrix preconditioning is performed at each time step, as for instance for transient calculations, but for steady state calculation the direct method could be faster than iterative one's if .the matrix preconditionning is not frequently computed.

2. The convergence of the CGS algorithm is improved by the IBLU preconditioning. The extra work required for the IBLU preconditioning is clearly worthwile since the number of iterations needed to reach the same convergence precision is much lower.

V. CONCLUSION

The algorithm developed for steady state 3-D two phase flow computations in reactor cores or rod bundles permits calculation with a gain of about twenty on CPU time as compared with a classical steady state computation calculated by using a transient algorithm with constant boundary conditions. For steady state computations, direct methods for solving the linear system give better results then iterative methods. Nevertheless, iterative methods will be more suitable for transient computations where matrix construction and preconditioning are required at each time step.

The comparison between the Conjugate Gradient Square method (CGS) and the Generalized Minimal Residual algorithm (GMRES) gives equivalent CPU times for preconditioning and solving linear systems. The GMRES method will be preferable if one considers that the convergence to the solution is always monotonic. However the major CPU time gain is obtained by a better preconditionnning like an incomplete Block LU factorization, which reduces the total number of requiered iterations for a given precision.

Many improvements may be performed in order to accelerate the convergence of the proposed steady state algorithm :

- optmisation of the interval between two calculations and preconditining of the matrix;

- acceleration of the convergence rate towards the steady state regime by using a gradient method at each time step;

- parallelisation of linear solvers on massively parallel computers is also a way to obtain fast computations of 3-D two phase flow steady states.

VI. REFERENCES

1. D. Caruge, P. Raymond : "FLICA-4 a Computer Code for 3-D Two Phase Flows Computation of PWR Cores" Proceedings of the ANS International Topical Meeting on Advances in Mathematics, Computations, and Reactor Physics, Pittsburgh, Pennsylvania (1991).

2. P. Raymond, D. Caruge, T. de Gramont : "Two-Phase Flow Computation of Rod Bundle experiments with the FLICA 4 Computer Code", 26 th. European Two-Phase Flow Group Meeting, Rome (1991)

3. D. Caruge, P. Raymond : "FLICA 4 : A Finite Volume Implicit Computer Code for 3-D Two-Phase Flows Computation of P.W.R. Cores", European Two-Phase Flow Group Meeting, Paris (1989).

4. I. Toumi : "An Effective Non-Iterative Method for Three Dimensional Two-Phase Flow Computations "Proceedings of the International Conference On Nuclear Engineering, Tokyo (1991).

5. P. L. Roe : "Approximate Riemann solvers, parameter vectors and difference schemes", J. Comput. Phys. 43 357-372 (1981).

6. I. Toumi : "A weak formulation of Roe's approximate Rieman solver", Acc. for pub. in J. Comput. Phys.