conf-790102--10

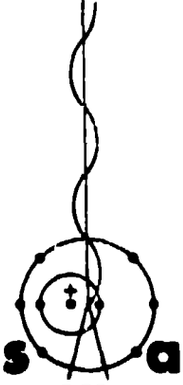
LA-UR-79-38

MASTER

TITLE: A Timing Comparison of Two-Dimensional Discrete-Ordinates Codes for Criticality

AUTHOR(S): W. F. Miller, Jr.
R. E. Alcouffe, T-1
G. E. Bosler, T-1
F. W. Brinkley, Jr.
R. D. O'dell,

SUBMITTED TO: ANS Topical Meeting
Computational Methods in Nuclear Engineering
Williamsburg, Virginia
April 23-25, 1979



los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

Form No. 836 R2
St. No. 2629
1/78

DEPARTMENT OF ENERGY
CONTRACT W-7405-ENG. 36

A TIMING COMPARISON OF TWO-DIMENSIONAL DISCRETE-ORDINATES CODES FOR CRITICALITY CALCULATIONS*

W. F. Miller, Jr., R. E. Alcouffe,
G. E. Bosler, F. W. Brinkley, Jr., and
R. D. O'Dell
Theoretical Division
University of California
Los Alamos Scientific Laboratory
Los Alamos, New Mexico, 87545

ABSTRACT

We have undertaken the task of comparing two-dimensional discrete-ordinates neutron transport computer codes to solve reactor criticality problems. Our fundamental interest is in determining which code requires the minimum Central Processing Unit (CPU) time for a given numerical model of a reasonably realistic fast reactor core and peripherals. The computer codes considered are the most advanced available and, in three cases, are not officially released. Our conclusion, based on the study of four fast reactor core models, is that for this class of problems the diffusion synthetic accelerated version of TWOTRAN, labeled TWOTRAN-DA, is superior to the other codes in terms of CPU requirements.

*Work performed under the auspices of the U. S. Department of Energy.

A TIMING COMPARISON OF TWO-DIMENSIONAL DISCRETE-ORDINATES CODES FOR CRITICALITY CALCULATIONS

INTRODUCTION

With the increased interest in heterogeneous cores in the fast reactor community has come increased use of two-dimensional discrete-ordinates computer codes for criticality calculations. Since there are several such codes more or less available to reactor designers, we have undertaken the task of comparing them in terms of computer Central Processing Unit (CPU) time to aid the user in the selection of the most efficient computational tool. We have considered four codes commonly used at various U. S. national laboratories. The first is a version of the diffusion code DIF3D[1] that has a discrete-ordinates module. This code, hereafter labeled DIF3D-T, is an in-house code developed and used at Argonne National Laboratory (ANL). The second candidate code is DOT-IV,[2] a soon-to-be released code developed at Oak Ridge National Laboratory (ORNL). TWOTRAN-II[3] is a well-established production code developed at Los Alamos Scientific Laboratory (LASL) while TWOTRAN-DA[4] is a newly developed version of that code due for selected release in a preliminary version toward the end of 1979.

THE CANDIDATE CODES

ITERATION STRATEGY

All four candidate codes iteratively solve the multigroup discrete-ordinates equations expressed, for the case of cylindrical (r-z) geometry, as

$$\mu_m \frac{\partial \psi^{\ell}_{mg}(r,z)}{\partial r} + \eta_m \frac{\partial \psi^{\ell}_{mg}(r,z)}{\partial z} \quad \frac{\alpha_{m+\frac{1}{2}} \psi^{\ell}_{m+\frac{1}{2}g}(r,z) - \alpha_{m-\frac{1}{2}} \psi^{\ell}_{m-\frac{1}{2}g}(r,z)}{r w_m} + \sigma_g(r,z)\psi^{\ell}_{mg}(r,z)$$

$$= \frac{\chi_g(r,z)}{k_{eff}} \sum_{h=1}^{IGM} [\nu\sigma_f(r,z)]_h \phi^{\ell-1}_h(r,z) + \sum_{h<g} \sigma_{sh\to g}(r,z)\phi^{\ell}_h(r,z)$$

$$+ \sum_{h>g} \sigma_{sh\to g}(r,z)\phi^{\ell-1}_h(r,z) + \sigma_{sg\to g}(r,z)\phi^{\ell}_g(r,z)$$

$$m = 1, 2, \ldots, M$$
$$g = 1, 2, \ldots, IGM \qquad (1)$$

where we have assumed that scattering is isotropic for simplicity. The notation used in Eq. (1) is standard[5] with m denoting the angular directional subscript, g the energy group subscript and ℓ, the outer iteration index. Note that the scalar flux, ϕ_g, is related to the angular flux, ψ_{mg}, by the familiar quadrature formula

$$\phi_g(r,z) = \sum_{m=1}^{M} w_m \psi_{mg}(r,z) \quad . \tag{2}$$

The candidate computer codes all use the power iteration method[6] in performing the indicated outer iteration cycles in Eq. (1) but they have significantly different approaches to accelerate the process. The DIF3D-T code accelerates the outer iteration process using the same Chebyshev method used in DIF3D.[6] The DOT-IV code employs a variation of the spatial rebalance method[7] whereby on a spatial mesh (that may be more coarse than the problem mesh) neutron conservation is periodically imposed as the iterations proceed. TWOTRAN-II uses a more traditional version of spatial rebalance.[3] Finally, TWOTRAN-DA uses the newly developed diffusion-synthetic acceleration (DSA) method.[4] The DSA approach has been shown to be equivalent to an angular and spatial rebalance method.[8] There are no user input parameters to control the outer iterations for DIF3D-T and TWOTRAN-DA. On the other hand, the DOT-IV and TWOTRAN-II codes allow the user to influence the iteration process by specifying the rebalance mesh as well as the maximum number of inner iterations (discussed below) for each outer iteration.

To display the inner iteration we drop the outer iteration index and define a group source, Q_g. Equation (1) is then written as

$$\mu_m \frac{\partial \psi_{mg}^n(r,z)}{\partial r} + \eta_m \frac{\partial \psi_{mg}^n(r,z)}{\partial z} + \frac{\alpha_{m+\frac{1}{2}} \psi_{m+\frac{1}{2}g}^n(r,z) - \alpha_{m-\frac{1}{2}} \psi_{m-\frac{1}{2}g}^n(r,z)}{r w_m} + \sigma_g(r,z)\psi_{mg}^n(r,z)$$

$$= Q_g(r,z) + \sigma_{sg\to g}(r,z)\phi_g^{n-1}(r,z) = S_g^{n-1}(r,z) \tag{3}$$

with n the inner iteration index. This process is accelerated in DOT-IV and TWOTRAN-II using spatial rebalance, and in TWOTRAN-DA using DSA. In an analogous approach to that used in DIF3D, the DIF3D-T algorithms calculate and set the number of inner iterations used for each energy group. The number of "inners" is fixed for each outer iteration but may differ with each group. Thus, DIF3D-T has no inner iteration acceleration method in the sense of the approaches used by the other three codes.

DIFFERENCING SCHEME

In the numerical solution of Eq. (3) for a given inner iteration one superimposes a spatial grid on the physical problem such that all material properties are spatial constants within a spatial cell. Letting j denote the axial (z) spatial index and i the radial (r) index, and integrating Eq. (3) over a spatial cell yields the balance equation

$$\mu_m(A_{i+\frac{1}{2}}\psi_{mi+\frac{1}{2}j} - A_{i-\frac{1}{2}}\psi_{mi-\frac{1}{2}j}) + \eta_m\beta_j(\psi_{mij+\frac{1}{2}} - \psi_{mij-\frac{1}{2}}) + \frac{A_{i+\frac{1}{2}} + A_{i-\frac{1}{2}}}{2w_m}(\alpha_{m+\frac{1}{2}}\psi_{m+\frac{1}{2}ij} - \alpha_{m-\frac{1}{2}}\psi_{m-\frac{1}{2}ij}) + \sigma_{ij}V_{ij}\psi_{mij} = V_{ij}S_{ij} \qquad (4)$$

where, again, notation is standard[5] and we have dropped the energy group and inner iteration indices. For a given quadrature direction, (μ_m, η_m) and specified boundary conditions, Eq. (4) is used to sweep through the spatial mesh. For a given phase-space cell, (i,j,m), all geometric and angular coefficients (α, μ, η, A, B, V) are known as well as cross section and source data. Additionally, the previous phase-space cell calculation provides the cell boundary angular fluxes: $\psi_{m-\frac{1}{2}ij}$, $\psi_{mi-\frac{1}{2}j}$, and $\psi_{mij-\frac{1}{2}}$ in the case, for example of $\mu_m > 0$, $\eta_m > 0$. There are four unknown angular fluxes in Eq. (4) and, therefore three more relationships are needed to complete the calculation for the $(ijm)_5$ cell. All four candidate codes use the diamond difference relationships[5]

$$\psi_{mij} = 1/2(\psi_{m+\frac{1}{2}ij} + \psi_{m-\frac{1}{2}ij}) , \qquad (5a)$$

$$\psi_{mij} = 1/2(\psi_{mij+\frac{1}{2}} + \psi_{mij-\frac{1}{2}}) , \text{ and} \qquad (5b)$$

$$\psi_{mij} = 1/2(\psi_{mi+\frac{1}{2}j} + \psi_{mi-\frac{1}{2}j}) . \qquad (5c)$$

There are somewhat subtle differences in the application of Eqs. (5) in each candidate transport code. Equations (4) and (5) do not assure positive cell exiting fluxes $\psi_{m+\frac{1}{2}ij}$, $\psi_{mi+\frac{1}{2}j}$, and $\psi_{mij+\frac{1}{2}}$, in violation of problem physics. For cases where negative exiting fluxes are calculated, algorithms in TWOTRAN-II and TWOTRAN-DA set the offending fluxes to zero. This set-to-zero negative-flux fixup[3] then involves recalculation of the other flux values to insure particle conservation. For cases of negative fluxes, DOT-IV employs a step-difference relationship to replace the appropriate one of Eq. (5). For example, Eq. (5a) is replaced by

$$\psi_{mij} = \psi_{mi+\frac{1}{2}j}$$

insuring a positive flux. It should also be noted that DOT-IV includes a user requested option to use, in lieu of Eqs. (5), more complicated weighted diamond

relationships that guarantee positive exiting fluxes. The DIF3D-T code does not include a fixup but simply uses the calculated negative flux.

In addition to the negative-flux fixup, the differencing schemes in the codes are dissimilar in another way. The normally employed angular quadrature schemes are defined such that several quadrature points share a common value of η. On each η level a problem exists as to how to solve Eq. (4) for the quadrature point corresponding to the smallest value of μ. Taking, for example, the case of $m = 1$, the $\psi_{\frac{1}{2}ij}$ needed in Eq. (4) is not known. For this case TWOTRAN-II, TWOTRAN-DA, and DIF3DT use the step relationship

$$\psi_{\frac{1}{2}ij} = \psi_{1ij} \quad . \tag{6}$$

On the other hand, DOT-IV uses the so-called starting direction, solving the transport equation with $\mu = -1$ and η equal to each of its distinct values. For these cases, the α's of Eq. (4) are zero. For $m = 1$ for example, one solves the equation

$$(-1)(A_{i+\frac{1}{2}}\psi_{\frac{1}{2}i+\frac{1}{2}j} - A_{i-\frac{1}{2}}\psi_{\frac{1}{2}i-\frac{1}{2}j}) + \eta_1 B_j(\psi_{\frac{1}{2}ij+\frac{1}{2}} - \psi_{\frac{1}{2}ij-\frac{1}{2}}) + \sigma_{ij}V_{ij}\psi_{\frac{1}{2}ij} = V_{ij}S_{ij} \tag{7}$$

in conjunction with Eqs. (5b) and (5c). The resulting $\psi_{\frac{1}{2}ij}$ are then used in Eq. (4) and Eqs. (5). As discussed in the numerical results section, the different treatments of the angular variables can result in slight variations in the eigenvalues.

A third variation to the solution processes used in the three codes is unique to DIF3DT. In lieu of the conventional sweeping of all cells for each direction described above, DIF3D-T solves Eqs. (4) and (5) using the unique up-down strategy of TPT.[9]

Table 1 provides a recapitulation of the above discussed features of the candidate codes.

CANDIDATE CODE COMPARISON STRATEGY

GENERAL

It is certainly not a straightforward task to perform computer running time comparisons of large computer codes. The answer to the question "which code is fastest?" unfortunately may depend upon the strategy used for comparison. Our overall strategy has four key features: 1) use one computer and one operating system, 2) measure only CPU time ignoring input/output (I/O) efficiency, 3) pick specific physical problems with specific mesh arrangements that are typical of those selected by fast reactor designers, 4) use the same differencing methods and convergence criteria, as much as possible, for each code. Thus, what is being compared is the effectiveness of the iteration acceleration methods as implemented in each code.

COMPUTER AND OPERATING SYSTEM

Our code testing was performed using the CDC-7600 computers at LASL operating under the LTSS-FTN (OPT=2) operating system. Although LTSS is time sharing and, therefore, susceptible to considerable movement of jobs in and out of large core memory (LCM) and small core memory (SCM) our reported running times minimize this effect as much as possible. Such minimization was assured through submission of all jobs at nights or on weekends when the computers at LASL operate in more of a batch-like mode with minimal job movement.

TIME MEASUREMENTS

The CDC-7600 has three levels of bulk storage: SCM, central or small core memory; LCM, fast access large core memory; and disk, slow access peripheral memory. The test problems solved in the following section were not large enough to require significant use of disk memory and thus I/O was not an important factor. To time the solution of very large problems, one must consider both CPU and I/O times and thus the overall times will be a function of the efficiency of the code's I/O routines. Since three of the candidate codes have not been released and at least two do not have completed, optimized I/O packages, we have elected to consider only CPU times. Thus the study of the efficiency of I/O strategies is beyond the scope of this work but we do not wish to minimize its importance for large problems.

PROBLEMS AND METHODS

We have selected four fast reactor test problems that include most of the physical arrangements likely to be encountered by fast reactor designers. We have selected a reasonable quadrature order (S_4 in all cases), group structure and spatial mesh for each model problem. To insure insofar as is possible that the comparisons are, for the same computational effort, we have used the same spatial differencing scheme (diamond differencing with negative flux fixup) for each code except for DIF3DT which, as was previously mentioned, does not allow negative flux fixup.

It should be pointed out that several features of DOT-IV were not fully exploited by this exercise. That code allows the user to specify a variable mesh where, essentially, the number of radial invervals can vary with the axial interval. Also, the weighted-diamond difference scheme can be selected which could in principle, allow acceptable accuracy for a coarser grid with some penalty of increased CPU time per cell calculation. DOT-IV allows the angular quadrature order to vary with spatial position and the user to specify diffusion theory for some groups and transport theory for others. With all these features, then, the very sophisticated DOT-IV user could in principle obtain an accurate solution to the test problem with fewer phase-space cells and, perhaps, less CPU time. Due to the incredible code comparison complications associated with pursuing these DOT-IV approaches, we have kept all mesh and methods for all codes as similar as possible.

We have also used, whenever possible, the same iteration convergence criteria for each candidate code. A separate convergence criterion is input for the eigenvalue, the fission source for each spatial cell, and the scalar flux for each spatial cell and for each energy group. Thus, we assume an interest

not only in k_{eff}, but also in the flux distribution. This required that slight changes be made to TWOTRAN-II; namely, a fission-source convergence check was added and the check on rebalance factor convergence was disabled. Also the check on cell scalar flux convergence was made to be identical to that used in DOT-IV.

Unlike the other candidate codes DIF3D-T does not use a check on the cell scalar flux to end inner iterations since the number of such iterations is fixed by the code. We did, however, verify that the DIF3D-T scalar fluxes did not differ substantially from the converged TWOTRAN-II values through the use of an auxiliary pointwise-flux comparison code.

NUMERICAL RESULTS

ZPPR-7A PROBLEM

Our first test problem was a three-group rectangular geomety model of the ZPPR-7A critical assembly mockup of a parfait core depicted in Fig. 1. For our first series of calculations we used the spatial mesh and rebalance mesh indicated in this figure. Table 2 depicts the CPU times for the ZPPR-7A calculations. For ease of discussion we have depicted the tabulated results as sets of calculations. The first set of four results indicates that TWOTRAN-DA is significantly superior to the other codes. The values labeled "Maximum Inners/Outer" are input by the user of TWOTRAN-II and DOT-IV but are automatically supplied for DIF3D-T and TWOTRAN-DA. The running times obtained using the former two codes are sensitive to this number and the times reported are the best obtained after some experimentation. We will return to this point while discussing the other test problems.

The next set of two results in Table 2 depict the effect of changing the convergence criteria. It should be noted that although the TWOTRAN-II eigenvalue is essentially unchanged, the DOT result is substantially different and, in fact, the DOT k_{eff}'s disagree in the third digit. These results indicate that, at least with DOT-IV, the user must be cautious in choosing the popular procedure of specifying a tighter convergence criterion for the eigenvalue than for the spatial cell quantities and assuming that the former is truly known to a greater precision than the latter.

The next set of results depict the significant CPU time savings with TWOTRAN-II and DIF3D-T when a diffusion theory flux is used as a guess to initiate the iteration procedure. Such a flux guess will not substantially affect TWOTRAN-DA run times since the iteration process begins with a diffusion calculation. DOT-IV calculations performed at ORNL[10] indicate that similar improvements in run times with a diffusion guess are obtained with that code. The last set of results in Table 2 indicate the effect of using tighter convergence criteria on CPU time.

Table 3 depicts results for the same ZPPR-7A problem but with twice the number of equal radial (x) and axial (y) intervals such that the problem mesh was 58 x 56. The original 29 x 28 mesh was retained as the rebalance mesh. In this table, as well as those to follow, one convergence criterion value was used for the eigenvalue, fission source and scalar flux. The first set

of four results again depicts the superiority of TWOTRAN-DA. In the TWOTRAN-II run, the last few outer iterations required very few inner iterations per group. Apparently, the DIF3D-T calculated requirement that exactly 30 inner iterations be performed for each outer iteration is inefficient for this problem. Keep in mind, however, that with a bad choice of "Maximum Inners/Outer" the DOT-IV and TWOTRAN-II CPU times can increase substantially.

The DOT-IV code was significantly slower than TWOTRAN-II for this mesh. The convergence patterns of these codes were vastly different. Of the 157 total TWOTRAN-II inner iterations, 105 of them occurred in the first two outer iterations due to the 60 allowed inner iterations per outer iteration. When this latter number was input to DOT-IV, the CPU time increased to 7.2 minutes. Clearly, the rebalance strategies of DOT-IV and TWOTRAN-II are quite different and can give quite different performances. It should further be noted that the rebalance mesh must be as fine as possible (within storage limitations) to achieve reasonable CPU times. For a very coarse rebalance mesh, TWOTRAN-II takes considerably longer than DIF3D-T to run.

In the single TWOTRAN-II run comprising the second set of Table 3 results we again see that a diffusion flux guess can significantly reduce transport CPU times. The last set of results in Table 3 are for tighter convergence criteria.

LCCBR MODEL

The second problem was a Large Core Commercial Breeder Reactor (LCCBR) model also considered by the Large Core Code Evaluation Working Group of the U. S. Department of Energy, Division of Reactor Research and Technology.[11] The physical problem, depicted in Fig. 2, was solved with six energy groups, with S_4 quadrature, and in cylindrical geometry. The rebalance mesh, not depicted in Fig. 2, was 34 x 18 as compared to the 67 x 34 fine mesh. Thus there were approximately four fine mesh cells in each coarse mesh cell.

Table 4 depicts the results for the LCCBR problem where all convergence criteria were set to 10^{-4}. The first set of four calculations, again demonstrate the excellent run times obtained with TWOTRAN-DA. It should be noted in all fairness to DOT-IV that use of the starting directions in lieu of step starting (see Table 1) results in an increase in calculational effort of 33% with, in principle, some degree of increase in accuracy of the solution.

The next set of two results demonstrate the effect of changing the value of the maximum allowed inner iterations per outer iteration. For this problem, TWOTRAN-II is somewhat insensitive to this parameter while DOT-IV run times depend strongly on it. The last set of calculations demonstrate the improvement obtained with a diffusion flux guess.

FTR MODEL

The third test problem was a cylindrical geometry model of the Fast Test Reactor (FTR). A four group approximation and 10^{-4} convergence criteria were used. The rebalance mesh was 16 x 34 as compared to the 31 x 68 fine mesh. Table 5 provides the numerical results for this problem where the pattern is similar to that of the previous problem. Again keep in mind the additional

computational effort of DOT-IV due to the use of starting directions. Note that the DOT-IV k_{eff} is significantly less than those calculated by the other codes. This behavior was pursued in conjunction with the last problem discussed below.

RECRITICALITY MODEL

The last test problem is depicted in Fig. 4. The problem models a reactor core configuration during the latter stages of a severe hypothetical core meltdown. Four energy groups, 10^{-4} convergence criteria, and cylindrical geometry were used. The rebalance mesh was 16 x 22.

Table 6 provides the CPU time comparisons for this problem where again, TWOTRAN-DA performs well. The DOT-IV k_{eff} result is not consistent with the other transport calculations when one considers the 10^{-4} convergence criteria. We undertook a cursory evaluation of this discrepancy in the following way. The effect of the starting direction in DOT as compared to step starting in the other codes was removed by simply assuming rectangular in lieu of cylindrical geometry. The angular terms of Eq. (4) (terms in α) do not appear in the rectangular transport equation. Now, of course, the problem has changed and, thus, the eigenvalues are expected to change. The DOT-IV and TWOTRAN-II k_{eff}'s became 1.1893 and 1.1897, respectively. We next note that the two codes use different negative-flux fixup schemes. Accordingly, negative-flux fixup in both codes was disabled yielding eigenvalues of 1.1894 and 1.1896, respectively. Finally, rebalance acceleration was made inoperative in both codes resulting in unaccelerated calculations. After ten minutes of CPU time neither code had converged the problem in the spatial cell scalar flux values. However, the iteration pattern indicates that both codes are converging to a k_{eff} of 1.1897. Accordingly, we surmise that the eigenvalue discrepancies of Tables 5 and 6 are due to the slightly different solution and acceleration algorithms used in the candidate codes.

CONCLUSIONS

We have studied four fast reactor model problems to determine the CPU performance of four two-dimensional discrete-ordinates neutron transport codes. Several important trends were evident from this exercise. First, we found that TWOTRAN-II and DOT-IV CPU times are sensitive to the rebalance mesh and the maximum allowed inner iterations per outer iteration. Our experience indicates that, within storage limitations, the rebalance mesh should be made reasonably fine. The TWOTRAN-DA and DIF3D-T codes, on the other hand, do not include input parameters that appreciably affect the CPU performance. Thus, the user does not have to be an expert to effectively use the codes. The TWOTRAN-DA code is at least twice as fast as any of the candidate codes, even when a diffusion flux guess is used to initiate the iteration process. With an unfortunate selection of DOT-IV and TWOTRAN-II input parameters, this factor can be much greater than two.

ACKNOWLEDGMENTS

The authors wish to acknowledge many helpful conversations with C. H. Adams of ANL and W. A. Rhoades of ORNL. We also thank both individuals for computer results for our test problems run on their respective computer systems.

REFERENCES

1. Information and code obtained via personal communication with C. H. Adams and E. E. Lewis, Argonne National Laboratory (1978).

2. W. A. Rhoades, "The DOT-IV Variable Mesh Discrete Ordinates Transport Code," Proc. Fifth Int'l. Conf. on Reactor Shielding, Knoxville, Tennessee, April 18-23, 1978. Information and code obtained via personal communication with W. A. Rhoades, Oak Ridge National Laboratory (1978).

3. K. D. Lathrop and F. W. Brinkley, Jr., "TWOTRAN-II: An Interfaced, Exportable Version of the TWOTRAN Code for Two-Dimensional Transport," Los Alamos Scientific Laboratory report LA-4848-MS (July 1973).

4. R. E. Alcouffe, "Diffusion Synthetic Acceleration Methods for Diamond-Differenced Discrete-Ordinates Equations," Nucl. Sci. Eng. 64, 344-355 (1977).

5. B. G. Carlson and K. D. Lathrop, "Transport Theory - The Method of Discrete Ordinates," Computing Methods in Reactor Physics, H. Greenspan, C. N. Kelber, and D. Okrent, Eds. (Gordon and Breach, New York, 1968).

6. D. R. Ferguson and K. L. Derstine, "Optimized Iteration Strategies and Data Management Considerations for Fast Reactor Finite Difference Diffusion Theory Codes," Nucl. Sci. Eng. 64, 593-604 (1977).

7. W. A. Rhoades, R. L. Childs, and W. W. Engle, Jr., "Comparison of Rebalance Stabilization Methods for Two-Dimensional Transport Calculations," Trans. Am. Nucl. Soc. 30, 583 (1978).

8. W. F. Miller, Jr., "Generalized Rebalance: A Common Framework for Transport Acceleration Methods," Nucl. Sci. Eng. 65, 226-236 (1978).

9. L. A. Hageman, "Numerical Methods and Techniques used in the Two-Dimensional Neutron Transport Program TPT," WARD-TM-1125(L), Bettis Atomic Power Laboratory (November 1973).

10. W. A. Rhoades, Oak Ridge National Laboratory, personal communication (1978).

11. E. Kujawski and H. S. Barley, "Benchmark Analysis of Liquid-Metal Fast Breeder Reactor Nuclear Design Methods," Nucl. Sci. Eng. 64, 90 (1977).

Table 1. Comparison of Key Features of Candidate Codes

Code	Negative Flux Fixup	Angular Starting Procedure	Iteration Acceleration Scheme	User Control of Iterations	Space-Angle Mesh Sweep
DIF3D-T	None	Step Start	Chebyshev (Fixed Inners/ outer/group)	None	TPT
DOT-IV	Step-Fixup (Optionally use Weighted Scheme)	Starting Direction	Rebalance	Significant	Standard
TWOTRAN-II	Set-to-Zero Fixup	Step Start	Rebalance	Significant	Standard
TWOTRAN-DA	Set-to-Zero Fixup	Step Start	Diffusion Synthetic Acceleration	None	Standard

Table 2. Coarse Mesh ZPPR-7A Problem Results[a] Using Various Transport Codes

Computer Code	Eigenvalue Convergence Criterion	Fission Source and Flux Convergence Criterion	Flux Guess	k_{eff}	Total Inner Iterations	Maximum Inners/Outer	Total CPU Time (m)
DIF3D-T	10^{-4}	10^{-4}	Flat	0.97923	390	30	1.3
DOT-IV	10^{-4}	10^{-4}	Flat	0.97911	212	15	1.3
TWOTRAN-II	10^{-4}	10^{-4}	Flat	0.97922	319	60	1.5
TWOTRAN-DA	10^{-4}	10^{-4}	Flat	0.97923	60	b	0.3
TWOTRAN-II	10^{-5}	10^{-3}	Flat	0.97921	160	60	1.1
DOT-IV	10^{-5}	10^{-3}	Flat	0.97660	133	30	0.8
DIF3D-T	10^{-4}	10^{-4}	Diffusion	0.97923	180	30	0.7[c]
TWOTRAN-II	10^{-4}	10^{-4}	Diffusion	0.97924	206	60	0.9[c]
TWOTRAN-II	10^{-5}	10^{-5}	Flat	0.97923	437	60	2.2
TWOTRAN-DA	10^{-5}	10^{-5}	Flat	0.97923	68	b	0.4

[a]The diffusion code DIF3D gives an eigenvalue of 0.97826 in approximately 0.1 min.

[b]TWOTRAN-DA specifies one inner iteration/group/outer iteration until fission source and eigenvalue convergence. Then as many inner iterations as are needed are allowed to obtain spatial cell scalar flux convergence.

[c]CPU times include DIF3D run time to obtain diffusion flux.

Table 3. Fine Mesh ZPPR-7A Problem Results[a] Using Various Transport Codes

Computer Code	Convergence Criterion	Flux Guess	k_{eff}	Total Inner Iterations	Total Outer Iterations	Maximum Inners/ Outer	Total CPU Time (min)
DIF3D-T	10^{-4}	Flat	0.98004	390	13	30	4.5
DOT-IV	10^{-4}	Flat	0.97990	204	16	15	4.9
TWOTRAN-II	10^{-4}	Flat	0.98001	157	14	60	2.2
TWOTRAN-DA	10^{-4}	Flat	0.98004	41	7	b	0.9
TWOTRAN-II	10^{-4}	Diffusion	0.98006	93	7	60	1.6[c]
TWOTRAN-II	10^{-5}	Flat	0.98004	243	22	60	3.4
TWOTRAN-DA	10^{-5}	Flat	0.98004	47	9	b	1.3

[a] The diffusion code DIF3D gives an eigenvalue of 0.97621 in approximately 0.3 min.

[b] TWOTRAN-DA specifies one inner iteration/group/outer iteration until fission source and eigenvalue convergence. Then as many inner iterations as are needed are allowed to obtain spatial cell scalar flux convergence.

[c] Same as Table 2.

Table 4. LCCBR Problem Results Using Various Transport Codes[a]

Computer Code	Flux Guess	k_{eff}	Total Inner Iterations	Outer Iterations	Maximum Inners/ Outer	Total CPU Time (min)
DIF3D-T	Flat	1.00821	765	17	45	8.8
DOT-IV	Flat	1.00806	473	19	30	7.5
TWOTRAN-II	Flat	1.00821	499	22	120	4.6
TWOTRAN-DA	Flat	1.00820	52	7	b	1.0
TWOTRAN-II	Flat	1.00821	425	23	60	4.1
DOT-IV	Flat	1.00808	709	19	60	11.2
DIF3D-T	Diffu-sion	1.00819	540	12	45	7.0[c]
TWOTRAN-II	Diffu-sion	1.00821	257	22	60	2.8[c]

[a]The diffsuion code DIF3D gives an eigenvalue of 1.00607 in approximately 0.2 min.

[b](Same as Table 3.)

[c](Same as Table 3.)

Table 5. FTR Problem Results Using Various Transport Codes[a]

Computer Code	Flux Guess	k_{eff}	Total Inner Iterations	Outer Iterations	Maximum Inners/ Outers	Total CPU Time (min)
DIF3D-T	Flat	0.99462	495	11	45	5.6
DOT-IV	Flat	0.99404	222	13	20	3.5
TWOTRAN-II	Flat	0.99455	223	10	40	2.0
TWOTRAN-DA	Flat	0.99461	35	5	b	0.6
DIF3D-T	Diffusion	0.99458	495	11	45	5.6[c]
TWOTRAN-II	Diffusion	0.99460	96	4	40	1.0[c]

[a]The diffusion code DIF3D gives a k_{eff} of 0.99087 in approximately 0.1 min CPU time.

[b](Same as Table 4.)

[c](Same as Table 4.)

Table 6. Recriticality Problem Results Using Various Transport Codes[a]

Computer Code	Flux Guess	k_{eff}	Total Inner Iterations	Outer Iterations	Maximum Inners/ Outer	Total CPU Time (min)
DIF3D-T	Flat	1.06714	360	9	40	3.1
DOT-IV	Flat	1.06644	310	18	20	3.1
TWOTRAN-II	Flat	1.06700	279	14	40	1.7
TWOTRAN-DA	Flat	1.06708	38	9	b	0.5
DIF3D-T	Diffusion	1.06714	360	9	40	3.2[c]
TWOTRAN-II	Diffusion	1.06701	150	9	40	1.1[c]

[a]The diffusion code DIF3D gives a k_{eff} of 1.04442 in approximately 0.1 min CPU time.

[b](Same as Table 5.)

[c](Same as Table 5.)

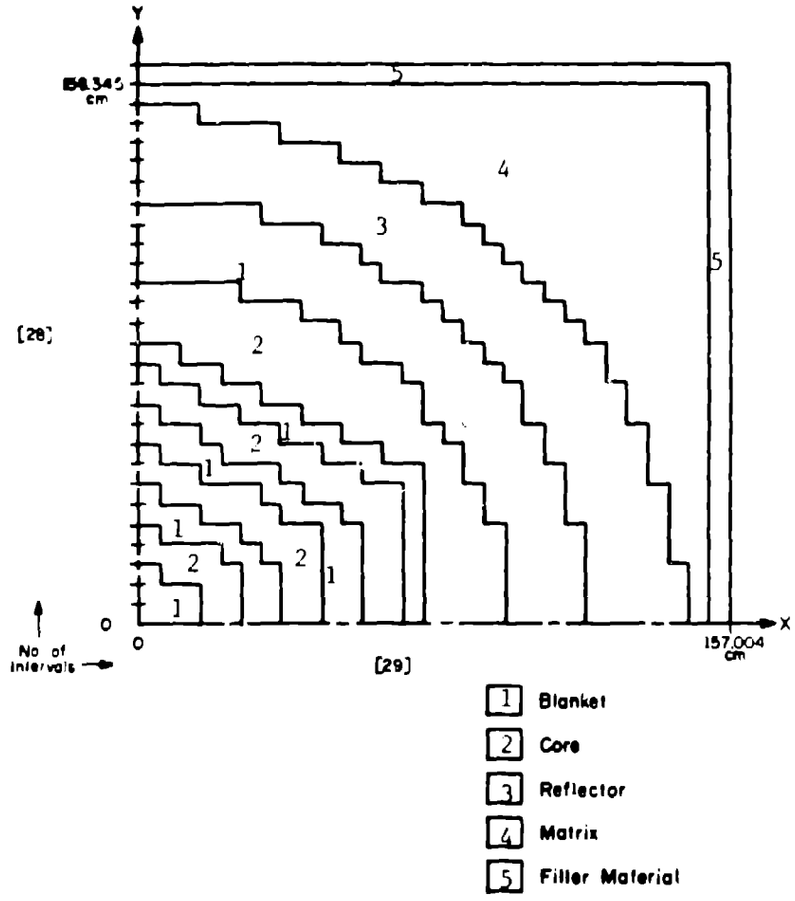


Fig. 1. ZPPR-7A problem.

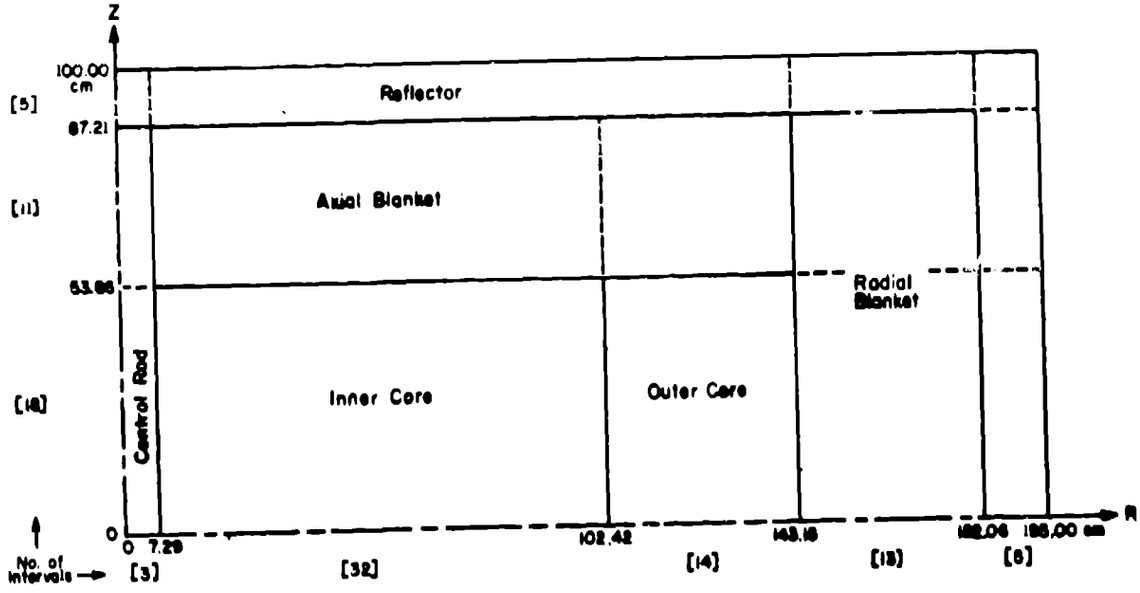


Fig. 2. LCCBR model problem.

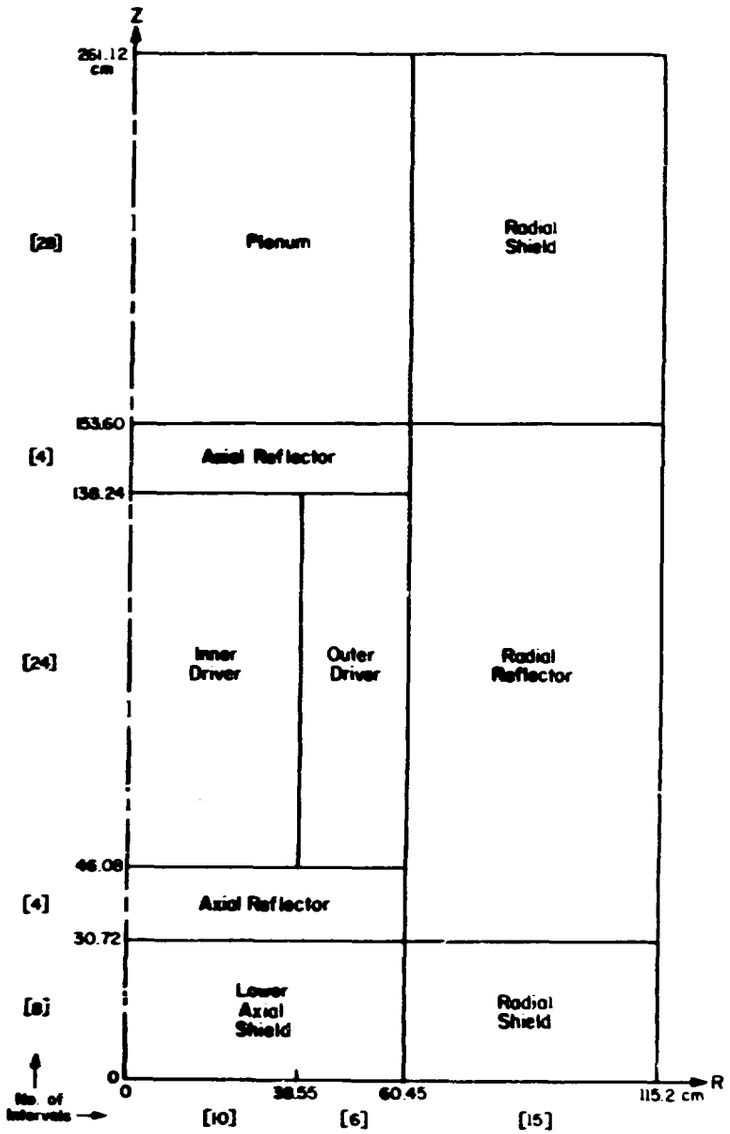


Fig. 3. FTR model.

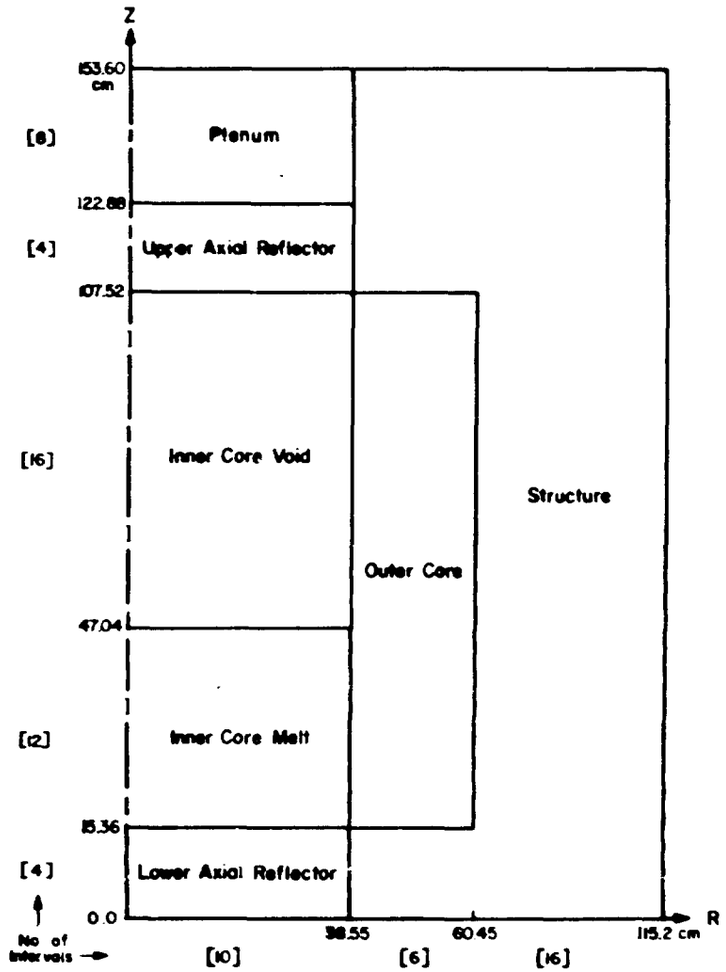


Fig. 4. Recriticality model.