LA-UR 94-3259

Conf-950420--8

TITLE: THREEDANT: A CODE TO PERFORM THREE-DIMENSIONAL, NEUTRAL PARTICLE TRANSPORT CALCULATIONS

AUTHOR(S): R. E. Alcouffe

SUBMITTED TO: International Conference on Mathematics and Computations, Reactor Physics, and Environmental Analyses, April 30 - May 4, 1995, Jantzen Beach Red Lion Hotel, Portland, OR

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

THREEDANT: A Code To Perform Three-Dimensional, Neutral Particle Transport Calculations

Raymond E. Alcouffe
Group X-6, MS B226
Los Alamos National Laboratory
Los Alamos, NM 87545

ABSTRACT

The THREEDANT code solves the three-dimensional neutral particle transport equation in its first order, multigroup, discrete ordinate form. The code allows an unlimited number of groups (depending upon the cross section set), angular quadrature up to S-100, and unlimited Pn order again depending upon the cross section set. The code has three options for spatial differencing, diamond with set-to-zero fixup, adaptive weighted diamond, and linear nodal. The geometry options are XYZ and RZΘ with a special XYZ option based upon a volume fraction method. This allows objects or bodies of any shape to be modelled as input which gives the code as much geometric description flexibility as the Monte Carlo code MCNP. The transport equation is solved by source iteration accelerated by the DSA method. Both inner and outer iterations are so accelerated. Some results are presented which demonstrate the effectiveness of these techniques. The code is available on several types of computing platforms.

I. INTRODUCTION

The THREEDANT code is the latest addition to our system of codes which perform neutral particle transport computations on a given system of interest. The system of codes is distinguished by geometrical or symmetry considerations. For example, ONEDANT and TWODANT are designed for one and two dimensional geometries respectively. We have TWOHEX for hexagonal geometries and now THREEDANT for three-dimensional geometries. The design of this system of codes is that they share the same input and edit module and hence the input and output is uniform for all the codes (with the obvious additions need to specify each type of geometry). The codes in this system are also designed to be general purpose solving both eigenvalue and source driven problems.

The main issues that need to be addressed in designing a three-dimensional transport solver are those of the computational time needed to solve a problem and the amount of storage need to accomplish that solution.Of course both these issues are directly related to the number of spatial mesh cells needed to obtain a solution to a specified accuracy, but is also related to the spatial discretization method chosen and the requirements of the iteration acceleration scheme employed as will be noted below. Another related consideration is the robustness of the resulting algorithms as implemented; because insistence on complete robustness has a significant impact upon the computation time. We address each of these issues in the following through which we give reasons for the choices we have made in our approach to this code. And this is useful in outlining how future versions of the code will evolve to better address the shortcomings that presently exist in the code.

II. A GENERAL STATEMENT OF THE SOLUTION APPROACH IN THREEDANT.

THREEDANT employs the multigroup, discrete ordinates method the discretize the momentum variables in the transport equation. This leads to a coupled set of linear partial differential equations in the space variables where the coupling is through the collision source term and the fission term. To illustrate the form of the discretized equation and the source iteration process, we write the system of equations as:

$$\underline{\Omega}_m \bullet \nabla\psi_{g,m}^{k+1/2} + \sigma_{t,g}(\underline{r})\,\psi_{g,m}^{k+1/2}(\underline{r}) = \sum_{l=1}^{L}\sum_{q=-l}^{l}\sigma_{s,l,g\to g}\phi_{g,l,q}^{k+1/2}Y_{l,q}(\underline{\Omega}_m)$$
$$+ \sum_{g'=g+1}^{G}\sum_{l=1}^{L}\sum_{q=-l}^{l}\sigma_{s,l,g'\to g}\phi_{g',l,q}^{k+1/2}Y_{l,q}(\underline{\Omega}_m) + \frac{\chi_g}{k_{eff}}\sum_{g'=1}^{G}(\nu\sigma_f)_{g'}\phi_{g'}^{k}$$
$$+ \sum_{g'=1}^{g-1}\sum_{l=1}^{L}\sum_{q=-l}^{l}\sigma_{s,l,g'\to g}\phi_{g',l,q}^{k}Y_{l,q}(\underline{\Omega}_m) + Q_{g,m}(\underline{r}) \qquad \text{(EQ 1)}$$

$$g=1,\ldots,G \qquad m=1,\ldots,M$$

where:

$$\phi_g(\underline{r}) = \sum_{m=1}^{M} w_m\psi_{g,m}(\underline{r})$$
$$\phi_{g,l,q}(\underline{r}) = \sum_{m=1}^{M} w_m\psi_{g,m}(\underline{r})\,Y_{l,q}(\underline{\Omega}_m)$$

In Eqn 1, g is the energy group index, m is the angle index, $Y_{l,q}(\Omega_m)$ are the spherical harmonics, and Ω_m is the discrete direction of particle travel. Thus it is seen that the scattering function is expanded in Legendre polynomials which requires the flux be expanded in spherical harmonics. The terms on the right hand side of Eqn 1 are the within group scattering source, the down scatter, the source from fissions and the upscatter source. Since the order of computation is from higher energy to lower (g=1 corresponds to the highest energy group), the down scatter term is known. The superscript k refers to the iteration index; thus k+1/2 is the current level of iteration and k indicates the previous level. The within group scatter is indicated at level k+1/2 which means that an inner iteration process for each group must be employed to do this. Thus, the source iteration process for solving the transport equation consists in inverting the left hand side of Eqn 1 onto the source (right hand side) which has been evaluated from the previous iteration. We write the source iteration in this fashion because we employ iteration acceleration methods to raise the iteration level to k+1. In the simplest case of no iteration, we simply set $\phi_g^{k+1} = \phi_g^{k+1/2}$. When the DSA acceleration method is used, then ϕ_g^{k+1} is the solution of a specially formulated diffusion equation. The use of this DSA technique greatly speeds the convergence of the source iteration process especially for eigenvalue calculations.

We recapitulate the solution process for the discrete ordinates transport equation as follows:

1. Evaluate the source from a flux guess or the previous iterate.
2. Invert the transport operator (the right hand side of Eqn 1) onto this source.
3. Evaluate the spherical harmonic moments of the flux from the angular flux.
4. Evaluate the scalar flux from the angular flux or for DSA compute the elements of the DSA equation and solve it for the scalar flux in each group.
5. Check for convergence depending upon the convergence criteria specified.
6. If the solution has not met the requirement, return to step 1.

Thus from a computational point of view, the main work is in inverting the transport operator and in solving the diffusion equation if DSA is invoked. The effort in obtaining the spherical harmonic expansion of the flux and computing the various sources is usually less than 10% of the total computation time. In the section where we discuss solution strategy, we give an indication of how we measure the time to invert the transport and diffusion operators and its impact upon the default solution strategy. It turns out that this is an important effect on the performance of the code and on the robustness of the solution method.

III. ISSUES IN THE GEOMETRICAL SETUP

When presented with a capability to do three-dimensional calculations, the user has an automatic desire to be able to model the 'real' geometry that is to be calculated rather than to make an idealized model which adds some unknown degree of uncertainty to the solution obtained. This natural desire has been addressed to some extent in the THREEDANT code in the sense that the user has the option to describe his geometry in terms of bodies much as is done in Monte Carlo codes. A calculational mesh is then overlaid on the geometry and the volume fractions of the intersection of the bodies with each mesh cell is computed. This volume fraction method is formulated so as to preserve the mass in the original body described geometry. The code we are using to set up the body described geometry and to compute the volume fractions on a prescribed XYZ mesh is Frac-in-the box. Future uses of this type of method for geometry description will be through a graphical users interface (GUI), a prototype of which we have running at LANL. This is part of a project to have a common geometrical setup description for both MCNP (our Monte Carlo transport code) and THREEDANT.

We of course have retained the simplified geometric input method in common with the TWODANT and ONEDANT codes. This is based upon having material boundaries fall on the boundaries of a coarse mesh description the faces of which are parallel to the coordinate directions with either XYZ or RZΘ symmetry. In this type of description, there is the possibility of making no error in the location of the interfaces and hence for problems which have this simplicity, this is the preferred method of geometry entry.

For the body described geometry, there is a source of error in the sense that exact interface information has been lost due to the volume fraction approach. This has implications as to meshing of the system as well as the accuracy to be expected in the final calculation. That is, the boundary can be considered resolved as far as the particle transport is concerned if the mesh into which it is to be homogenized is less that a mean-free-path (mfp). This size of mesh is also that recommended for the simple spatial differencing schemes; however for more elaborate and hence more accurate spatial differencing methods, the mesh may be considerably larger than a mfp. We do not have too much experience with the higher order differencing methods in these cases so this is an area for further exploration.

As an exercise to demonstrate the ability of the code to treat general geometry and to somewhat assess the errors that can be made, we have run a bare cylinder of oralloy in a void as an RZ, TWODANT problem to determine k_{eff}. The cylinder has a radius of 5.97 cm and a height of 11.94 cm. We then set up the same problem in THREEDANT as an XYZ problem with the axis of the cylinder parallel to the Z axis, and then another problem with the axis tilted at 45 degrees in the XZ plane. The results are shown in Table I. The TWODANT was run with the 16 group Hanson&Roach cross sections in S20 square quadrature with a spatial mesh of 20x20. The THREEDANT runs were made with the spatial mesh shown in the table with an S6 square quadrature.

TABLE 1. K_{eff} for a Bare Cylinder in TWODANT and THREEDANT.

Spatial Mesh	TWODANT	THREEDANT (Parallel to Z)	THREEDANT (45 deg in XZ)
20x20	0.8506		
40x40x20		0.8495	0.8490
60x60x30		0.8497	0.8497

These results show that this cylinder has been successfully modelled and that as is physically reasonable, the two orientations in 3D has only a small effect on the results. The disagreement between the RZ model and the XYZ model can be made smaller by refining the mesh and angle in both cases. Similar problems can be run by the user to assess the impact of the volume fraction approach on the accuracy of the results for a particular problem.

IV. ISSUES IN COMPUTATIONAL EFFICIENCY

As pointed out in Sect II above, computational efficiency is affected by many ingredients. The main ingredients we address here and which are addressed in THREEDANT involve: (1) the CPU time/phase space cell to invert the transport operator, (2) the amount of time needed to solve the DSA equations, (3) the spatial differencing scheme, and (4) vectorization, parallelization and super scalar considerations. Efficiency also includes the memory required to implement the methods used and is affected by the type of calculation to be done, eigenvalue or source driven. Note that item (4) above is an ingredient to all issues of CPU time to do a computation and the amount of memory required because of the way the algorithms are coded to take advantage of vectorization, super scalar and cash management and especially parallelization. We consider each of the items in turn and indicate how they are handled in the code

4.1 CPU Time to Invert Transport Operator.

This time is governed by the computing platform type, of course. But on each platform certain optimizations can be done to reduce the amount of time spent in inverting the transport operator. Another consideration is the type of spatial differencing scheme used to solve the transport equation. The more elaborate the differencing method the more operations that need be performed and hence the more costly it is to use. We assess the time to invert the transport operator by quoting the average CPU time per phase space cell; i.e., the time per energy group per angle per spatial mesh cell. As will be mentioned in Sect V, the spatial differencing schemes that are used in THREEDANT are the diamond differencing method with set-to-zero fixup, the adaptive weighted diamond method, and the linear nodal method. Vectorization and parallelization methods (for massively parallel, SIMD architectures) use the same method for all differencing schemes. Thus, on the Cray YMP we invert the transport operator a Z-plane at a time using diagonal line sweeps in the XY plane, following the direction of travel of the particles. Thus the vector length is variable and is equal to the length of the diagonal in the XY plane. In the current XYZ solver, we make this somewhat more efficient by including all the angles in an octant in the sweep so that a gather-scatter operation is done. This is still vectorizable and is quite efficient on the Cray. On the Thinking Machines CM200, we employ a diagonal plane sweep incorporating all the angles in an octant which tends to maximize the number of processors that can be kept busy doing the inversion, but here we are talking about problems of a million spatial mesh cells or more for maximum efficiency. In Table 2, we present an average of the cell time for a variety of computing platforms and the spatial differencing schemes in THREEDANT when the sweeper is most efficient.

TABLE 2. Cell Time in Microseconds for a Variety of Platforms and Differencing Methods.

Platform	Diamond	Diamond (STZ)	AWDD	CL Nodal	LL Nodal
Cray/YMP	0.25	0.5	0.5	1.25	2.2
TM/CM200	0.1	0.2	-	-	-
IBM/RS6000/560	5.0	10.0	10.0	-	-
Sun/Sparc10	12.5	25.0	25.0	-	-

In Table 2, the column labelled diamond is the diamond method with no fixup, STZ refers to set-to-zero fixup, CL Nodal is the constant linear nodal method, and LL Nodal is the linear-linear nodal method. This table shows that the transport sweeper performs quite well on these platforms and that, as mentioned above, the spatial differencing scheme does influence greatly the amount of time needed to invert the transport operator.

4.2 Time Needed to Solve the DSA Equations.

If the source iteration process were rapidly convergent, then the bulk of the computation time would just that required to invert the transport operator. However, the source iteration process is frequently very slowly convergent so some means of iteration convergence acceleration must be used to obtain optimal efficiency in the transport calculations. This paper assumes that the reader is familiar with the concept of diffusion synthetic acceleration of the source iterations. Suffice it to say here that in THREEDANT, the DSA method involves the solution of a diffusion equation for each of the groups in the inner iterations and the solution of the multigroup diffusion equation for the outer iterations. To define inner and outer iterations, we refer back to Eqn 1. The first line of Eqn. 1 is the inner iteration which is the source iteration process that accounts for the coupling of the angular flux through the within-group scattering source. In this iteration process, the other scattering terms and the fissions are assumed known from the previous iteration or the previous groups and are thus held fixed. The outer iteration process accounts for the coupling of the group fluxes through the fissions and upscattering processes. The DSA procedure can be used (and is used in THREEDANT) to accelerate the convergence of each of these processes. In the DSA process the main assumption is that we can replace transport iterations by diffusion (or some other similar low order operator) iterations and achieve a more computational efficient solution. This being said, we must somehow measure the efficiency of the acceleration process against the CPU time required to use it to ascertain what the overall benefit will be.

Now the three-dimensional diffusion operator is by no means a trivial operator to invert; the inversion process is itself an iterative procedure. In THREEDANT we have three methods to invert the diffusion operator: (1) multigrid with line relaxation, (2) conjugant gradient with line sweep preconditioner, and (3) successive line relaxation. In general multigrid is the most efficient and thus is the default. In general the time to invert the diffusion operator is about the same time as that required to invert the transport operator using S4 quadrature. Thus in order for the DSA procedure to be efficient, it must be effective enough to reduce the number of iterations by at least a factor of three. In order to demonstrate some of the effectiveness of the DSA process and to assess the impact of the inversion of the diffusion operator we present three problems which have been described in detail elsewhere. Two of the problems are eigenvalue problems and the third is a shielding problem. The first eigenvalue problem is a two group LWR model and the second is a four group fast reactor assembly. The shielding problem is the iron-water shield of ref -. In Table 3 we present some timing and iteration results on the Cray/YMP.

TABLE 3. Timing and Iteration Data for Three Problems from THREEDANT/YMP.

Problem	Accelerator	Keff	Total CPU time(s)	# inners	# outers	trans time(s)	Diffusion time (s)	Work units[a]
LWR	none	0.96239	228.3	580	82	228.3	-	-
	DSA/MG	0.96236	12.7	15	7	5.7	6.6	441
	DSA/CG	0.96236	16.0	17	8	6.2	7.2	774
	DSA/SR	0.96236	29.2	15	7	5.7	23.0	3390
FBRL	none	0.96973	905.7	1960	88	905.7	-	-
	DSA/MG	0.96999	53.6	53	10	24.1	27.7	1982
	DSA/CG	0.96999	54.1	54	10	24.4	27.8	2955
	DSA/SR	0.96999	69.3	53	10	24.6	42.9	6483
3DFEH2O	none	-	716.5	233	1	716.5	-	-
	DSA/MG	-	186.4	51	1	154.4	29.1	480
	DSA/CG	-	206.9	54	1	167.1	36.8	1044
	DSA/SR	-	301.8	52	1	160.2	138.7	6396

a. A work unit is a line relaxation through the entire 3D mesh.

In the above MG refers to multigrid, CG refers to conjugate gradient and SR refers to successive relaxation.It is seen from these examples that the acceleration is certainly an efficient way to go to reduce the number of iterations by nearly a factor of 60 for the eigenvalue problems and a factor of 4 in the source driven shielding problem. This translates into a savings of about a factor of 20 in CPU time in the eigenvalues and a factor of nearly 4 for the shield. We also see that from the work unit point of view, the MG solver is the most efficient giving a slight edge in computational time for the diffusion compared to CG and substantial for SR.

4.3 Spatial Differencing Scheme

We elaborate a little here on the spatial differencing schemes and their use in THREEDANT. The diamond with set-to-zero fixup has been the workhorse for all the DANT codes in the past. This is because they are on the whole adequate for eigenvalue calculations which has been the main use of the codes. However, as we move into deep penetration problems such as encountered in shielding and reactor vessel activation problems, diamond is not the most efficient. It suffers mainly from having to restrict the mesh size to around a mean free path. Meshes large than that require numerous fixups and hence leads to spatial oscillations in the solution which for many shielding applications are unacceptable. The adaptive weighted diamond method described for THREEDANT use in ref- alleviates some of these problems with little or no extra cost. This is a predictor corrector method which imposes a monotonicity condition on the solution and hence smooths out the spatial oscillations in diamond with little loss in overall accuracy. We illustrate the effectiveness of the method to reduce oscillations by presenting in Fig. 1 a plot of traverses in the iron-water shield problem using both methods. It is seen that the oscillations are indeed substantially reduced. However, there are many cases where the mesh is so large that accuracy is compromised even with the AWDD method. In these cases we have implemented the linear nodal method as either constant- linear or linear-linear using the default DSA accelerator. The use of this accelerator is not as effective with the nodal as with diamond but in many cases it performs well enough. We note that the nodal method is more costly by a factor of from 2.5 to 4 that AWDD, but the savings in the number of meshes required for a specified accuracy makes this ultimately the most efficient for deep penetration problems.

V. SOLUTION STRATEGY AND QUALITY OF SOLUTION AIDS

Mostly in the name of computational efficiency, it is expedient to devise a solution strategy to solve three-dimensional transport problems. The strategy is different for source driven calculations from that used in eigenvalue calculations. Both involve the assumption that the iteration accelerator is effective in reducing the error in the solution over that of the unaccelerated case. We outline each of these strategies in the following:

5.1 Source Driven Problems

In source driven problems without fissions, the strategy is to converge the inner iterations for each group according to the input convergence criterion. This criterion is such that the change in the scalar flux in each spatial mesh cell must be less than the input error, ex. less than 0.1%. If there is no upscatter, then the problem is solved. If there is upscatter, then outer iterations must be done and so the inner iteration procedure is repeated with the updated scattering source. When this source has been converged to the convergence criterion, the problem is then complete. If there are fissions in the problem, then the strategy switches to that used for eigenvalue problems.

5.2 Eigenvalue Problems

In this case it is of little use to converge the inner iterations until the source has been converged. Thus when the DSA is effective, the strategy is to do only one inner iteration per outer for each group. The source is then iterated using the multigroup DSA equation with the transport information from the one inner. With the new source, then another inner is done for each group and the source is then recomputed using the multi-group DSA. This process continues until the source is converged with the supplied transport information. The inners are then converged for each group. It is very rare that another source iteration will need be done as the new source using the converged inner information is usually within the convergence criterion of the old source. This strategy greatly reduces the number of times that the transport operator need be inverted and hence reduces the CPU time for solution. It is crucial though that the acceleration process be effective, or else this procedure will not converge. The code will detect these situations for the most part and turn off the DSA for the affected groups. In these cases the multigroup DSA will be used with some of the groups not participating and the overall strategy is still effective. If not, we wind up in the worst case with solving the problem with no acceleration; but for a properly posed problem, this is very rare.

5.3 Quality of Solution Aids

With the complications of the iteration solution process outlined above and the use of spatial differencing schemes that require some sort of fixup, it is prudent to provide an indication of the quality of the solution obtained and to indicated iteration convergence problems. One of the most important aids is a balance table. This is data derived from the solution which gives the quantities in the angle-integrated balance equation derived from the conservative form of the discretized transport equation. Thus terms appear such as the absorption rate, the self-scattering rate, the in-scattering rate, the out-scattering rate, the source rate, the fission rate, and the net leakage rate for each group. Also computed is the error in the balance between the sources and losses in each group and the sum of the groups. Since this is a spatially integrated balance it should be less than the convergence criterion specified. This table is valuable because it expresses the physics of the problem being solved; and if there is something amiss, this table will show it.

Another aid is the iteration monitor which records the history of the error in the solution process and the number of iterations required for solution. The code also monitors the convergence rate and compares it with the expected rate of convergence. If the computed rate is far below the expected rate, the code prints out this information. The slower than expected rate of convergence is usually due to nonlinearities in the spatial discretization which come from the fixups. This is turn is due to the mesh being too large. Hence improvements in the iteration convergence performance can be seen if the mesh is made finer in most cases. Of course the user may know from this information which groups are giving problems, but there is nothing to say which spatial region. A way to provide this information is to provide the number of fixups used in each coarse mesh spatial region of the problem as is done in TWODANT. This will be provided in some form in the THREEDANT code in the future.

VI. OTHER CAPABILITIES IN THREEDANT

We have included other useful capabilities in the THREEDANT code to solve particular problems. Below is not an exhaustive list but indicates some of the directions we have taken.

6.1 Boundary Conditions

Besides vacuum boundary conditions, we have also included reflective boundary conditions on all faces as needed, albedo boundaries, periodic boundaries and white boundaries. This allows the modelling of many types of cell problems and other related phenomena.

6.2 Group Dependent Sn

The Sn order can be input as group dependent. Thus some groups which require it can use a high order angular quadrature while others use a lower order as needed. This is for the sake of efficiency in that the computing time is roughly proportional to the number of angles used (depending on the amount of vectorization). This option can even be used to specify some groups be solved by diffusion theory only.

6.3 First Collision Source

In some problems of interest, the source is in a spatially very localized region. If the scattering is also relatively weak, serious ray effects will develop in the solution. Ray effects are the unphysical spatial oscillations in the solution due to the finite number of angles being used. One way to alleviate this, is to use a first collision source method. In THREEDANT we use a ray tracing option that does an analytic ray trace from the source region through all other regions of the problem. From this is constructed a first collision source which is spread throughout the problem region. Another transport problem is then solved using discrete ordinates and this first collision source. The solution to the original problem is then the sum of the ray tracing solution and the Sn solution. This technique greatly improves the accuracy of the solution to such problems.

VII. CONCLUSIONS

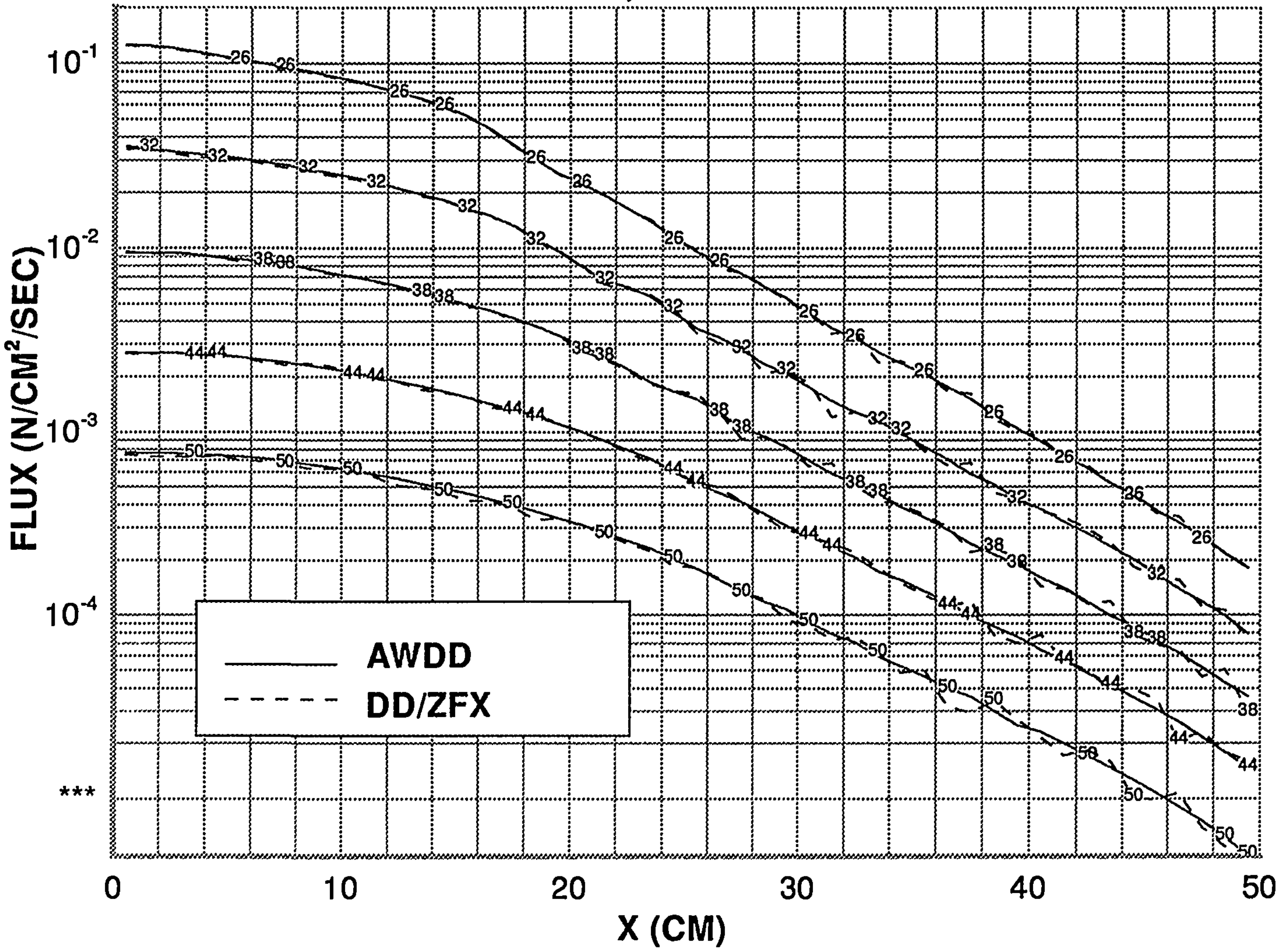
THREEDANT is a code still actively under development but is also a mature code that has capabilities to solve many three-dimensional transport problems in computationally efficient manner. The flexible geometry options are particularly attractive as well as the attention paid to solving eigenvalue problems. It is up and running on a variety of computing platforms and takes advantage of the vector, super scalar or parallel opportunities that are possible on each one. Not too much has been mentioned on memory management in this paper because not a whole lot of effort has gone into a modern memory management capability for the code. This is the weakest part of the code and we will be addressing it seriously in the near future. This is because one of the main impediments for using the code is that it requires a great deal of memory. Other future developments will include linking it to a GUI so that the same geometrical setup is available to THREEDANT and MCNP. Also we intend to implement other more advanced spatial differencing schemes which, from research being done here, look promising for deep penetration problems.

VIII. REFERENCES

(Will be provided in the final draft)

FIG. 1

AWDD VS DD W/SET-TO-ZERO ON 3D FE-H2O SHIELD PROBLEM.
COMPARISON OF FLUX TRAVERSES FOR SELECTED J LINES.
GROUP 1, K PLANE 50.


FLUX (N/CM²/SEC)
10⁻¹
10⁻²
10⁻³
10⁻⁴
***
X (CM)
AWDD
DD/ZFX