PHYSOR 2004 –The Physics of Fuel Cycles and Advanced Nuclear Systems: Global Developments
Chicago, Illinois, April 25-29, on CD-ROM, American Nuclear Society, Lagrange Park, IL. (2004)

New Computational Methodology for Large 3D Neutron Transport Problems

M. Dahmani*, R. Roy and J. Koclas
Nuclear Engineering Institute, École Polytechnique de Montréal,
P.O.Box 6079, Station CV, Montréal, Québec, Canada H3C 3A7

We present a new computational methodology, based on 3D characteristics method, dedicated to solve very large 3D problems without spatial homogenization. In order to eliminate the input/output problems occurring when solving these large problems, we set up a new computing scheme that requires more CPU resources than the usual one, based on sweeps over large tracking files. The huge capacity of storage needed in some problems and the related I/O queries needed by the characteristics solver are replaced by on-the-fly recalculation of tracks at each iteration step. Using this technique, large 3D problems are no longer I/O-bound, and distributed CPU resources can be efficiently used.

KEYWORDS: Neutron transport, characteristics method, iterative linear solver, large-scale problems.

1. Introduction

In the last three decades, it has been practically observed that the CPU power almost doubled every 18 months. This is known as Moore's law. [1] However, the I/O speed increased at slower pace than the CPU speed and the capacity of storage does not follow linearly the CPU speed. According to these computational considerations and taking into account the limitations in term of I/O and memory bounds that are faced when trying to solve large 3D problems; we decided to investigate a computational methodology that requires more CPU power and less storage resources.

This paper aims to explain the new 3D characteristics technique dedicated to solve very large 3D problems (a part or a whole core) without spatial homogenization. In order to eliminate the input/output problems occurring when solving these large problems, we define a new computing scheme that requires more CPU resources than the usual one, based on sweeps over large tracking files. The huge capacity of storage needed in some problems and the related I/O queries needed by the characteristics solver are replaced by on-the-fly recalculation of tracks at each iteration step. Using this technique, large 3D problems are no longer I/O-bound, and distributed CPU resources can be efficiently used.

We begin by describing briefly, in section 2, different computational methodologies used in the DRAGON [2] code to solve a 3D neutron transport problems. In section 3 we present the new computational methodology that we have developed in order to solve the 3D large problems. In the last section, numerical results are shown. The validation of the new solver and the comparison between the two computational methodologies based on characteristics method are presented.

* Corresponding author: Fax: 514-340-4192, Email: mohamed.dahmani@polymtl.ca

2. Transport Calculation Schemes

2.1 Collision Probability Method

The usual deterministic method used in DRAGON for solving the 3D neutron transport equation in supercell geometry is the collision probability (CP) technique. In this context, the discretization of the spatial domain is done by using the ray tracing method [2] to construct the CP matrices (CPM). Two different methodologies can be used. In the following, we will describe these two different ways to perform the transport calculation and discuss their limitations for treating the large 3D problems.

2.1.1 EXCELT-ASM Scheme

The integration lines, required to cover the 3D Cartesian domain, are generated by the *EXCELT* module using the ray tracing method and then stored in a sequential binary file. The tracking file (TF) size increases with increasing number of regions, number of angles and density of tracks. The *ASM* module accesses then to the TF in order to compute the collision, escape and transmission probabilities for each energy group (corresponding to region-to-region, region-to-surface and surface-to-surface respectively). These probability matrices are then algebraically reduced by eliminating the currents from the multigroup system (Fig.1). The module *FLU* is then used to compute the solution to an eigenvalue problem corresponding to a set of group-dependent system matrices. In this representation, we need to store both the tracking file and the collision probability matrices.

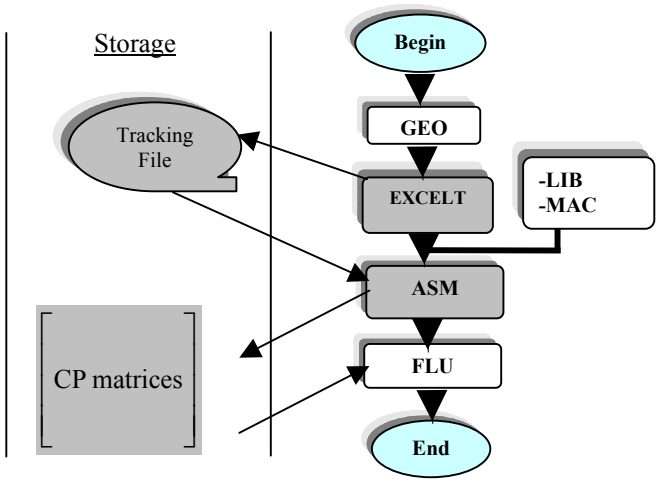

Fig. 1 EXCELT-ASM computational scheme

2.1.2 EXCELL Scheme

The ray tracing procedure and the evaluation of the CP matrices is done in the same module *EXCELL*. The tracking file does not have to be stored. This method is faster than the previous one but it still needs to store the huge CP matrices [3] (Fig. 2).

In CP method, the storage requirements increase as the square of the number of regions. There is no more limitation on the size of the TF, and thus the problem is no longer I/O-bound; however, the memory resources could be easily exceeded, so that the problem is now memory-bound.

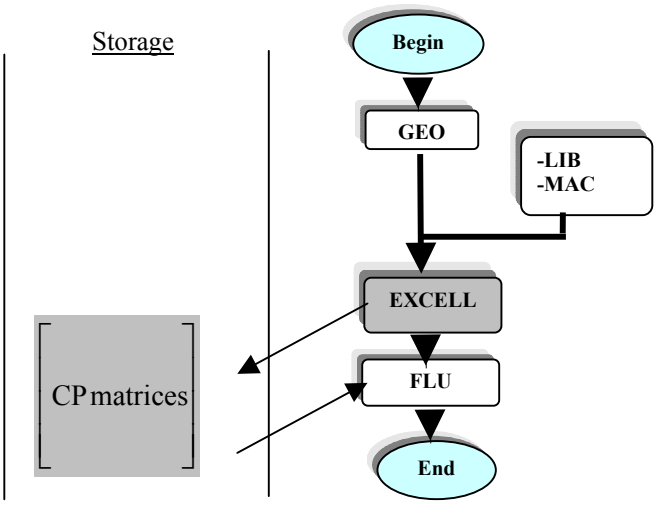

Fig. 2 EXCELL computational scheme

2.2 Characteristics Method

The 3D characteristics method was developed as an alternative for solving large problems. The long-characteristics MOC is based on solving the differential form of the transport equation by following the characteristics of the system (tracking lines) to simulate neutron paths [4]. The scalar flux is constructed by collecting all mean angular fluxes in terms of the entering angular flux and the source of the region. Unlike the CP method, the MOC permits the elimination of the CP matrices, which have the dimensionality of the size of the square of the number of regions. We have shown that such linear MOC solvers can give solutions equivalent with the quadratic CP ones. [5]

2.2.1 EXCELT-MCI Scheme

The same ray tracing method than in CP in EXCELT module can be used to create the tracking file TF. The tracking sweep is done by Looping on number of tracks at each inner iteration. A 3D MOC sequential solver has been implemented as the *MCI* module in DRAGON code. At each loop, the solver needs to access the TF to read tracking data (Fig.3). These I/O operations can be very expensive for large files, and the only possible alternative is to parallelize the I/O operations. [6]

Further, because there are no linear algebra treatments involving the matrices, the memory needed to perform 3D calculation is greatly reduced. However, the major problem that is encountered for large problems, in addition to the fact that the MCI solution can converge slowly, is the lack of space to store the tracking data [7]. In some cases, we have to restrict the number of tracks (less angles or smaller density of tracks) and the transport solution may not have the required accurate.

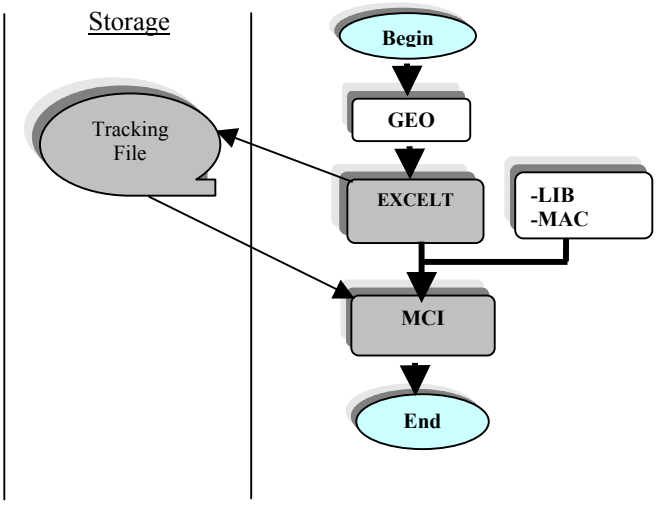

Fig. 3 EXCELT-MCI computational scheme

2.2.1.1 Parallelization

At the solver level, the smallest operation possible is composed by an I/O operation to get one tracking data from the TF and a CPU operation on this single track to compute the angular flux. The global computational operations are the combination of these atomic operations.

To reduce the computation time, the parallelization of the solver was performed by distributing the tracks on several processors. A broadcast operation is activated to communicate the local solution between processes [5, 8]. The parallelization of the I/O operations was also implemented. [6]

3. A New Methodology

The idea is to take advantage of the best features of the above methods in order to be able to solve large 3D problems. Intuitively, the ideal choice would be a linear characteristics method without I/O bounds. The new MCG technique is an alternative to the MCI technique. It is based on the elimination of the use of the tracking file to store the data. The MCG inner solver is based on this process:

1- Generate a track;
2- Calculate the angular flux on this track;
3- Add this contribution to compute the scalar flux;
4- Destroy the track and go to 1.

This new solver was fully implemented and results were compared with the usual MCI calculations (see Fig. 4).

The solver generates tracks, as needed, and then the angular flux is computed along the track to contribute in the scalar flux calculation. In this way, the storage of tracks is not necessary. Like in MCI, in order to reach a convergence of the solution, more iterations than in CP method are required. This is due to the current convergence and to the flux initialization (flat flux).

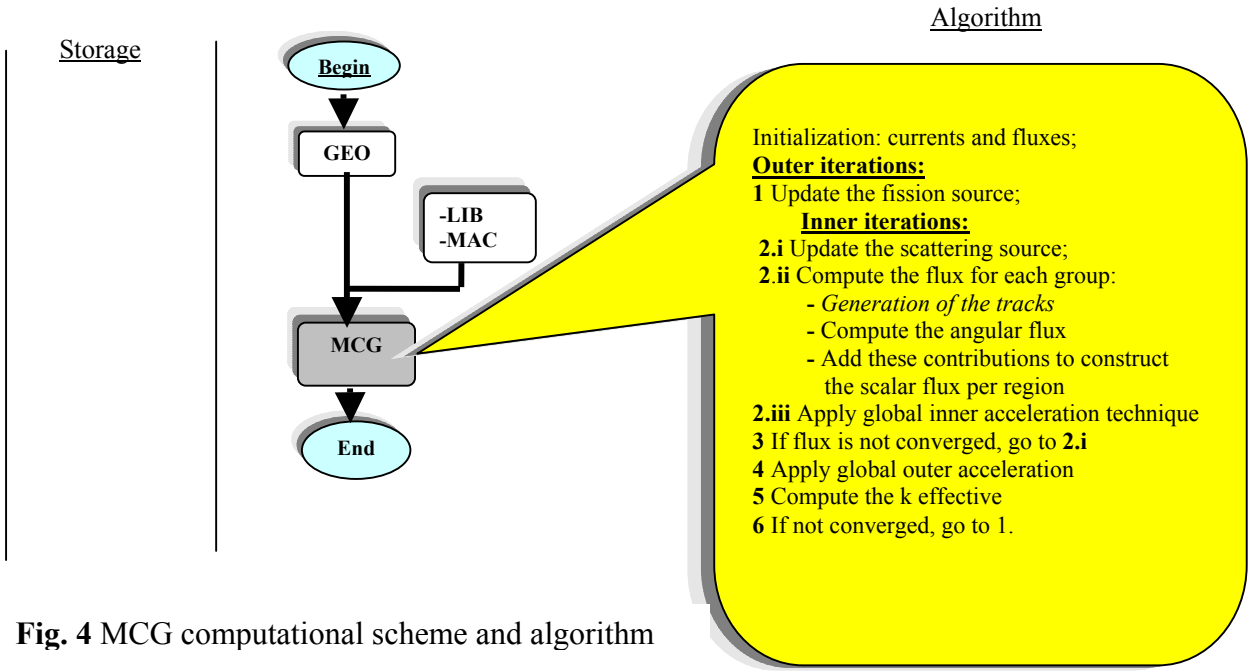

Fig. 4 MCG computational scheme and algorithm

3.2 Summary

In Table 1 we summarize the characteristics of the four computational methodologies described above. For the large-scale problems, each of these methodologies has computational limitations. For the new methodology that we have developed, the only hypothetical limitation would be the CPU bounds.

Table 1 The summary of the four computational schemes

Computational scheme	Method	TF	Storage	Parallelization	Limitations
EXCELT-ASM	CP	yes	TF CPM	no	Memory and I/O bounds
EXCELL	CP	no	CPM	no	Memory bounds
EXCELT-MCI	MOC	yes	TF	Message Passing : PVM, MPI, MPI-IO	Memory and I/O bounds
MCG	MOC	no	no	no	CPU bounds!

4. Numerical Results

We will present some results used to validate the MCG module. The tests were done for CANDU-6 supercells with different configurations as described in [9]. Transport equation is then

solved by critical buckling search with order leakage treatment B_1. For standard tests, we use an EQ_4 angular quadrature, an isotropic reflection option, and a 89 energy groups.

The results given by the module *EXCELL* with HELI normalization option (which gives almost the same results than the reference used in [8]) are considered as references, we use then the relative error for the calculated value of the incremental cross-sections. In Table 2-3, we can see that the results given by the *MCG* module are very close to those given by *EXCELT-ASM* (based on CP method). The *MCG* gives better results than *EXCELT-MCI* because it is based on the same tracking procedure than in *EXCELL* and use the same variable precision. In Table 4, we report the total CPU time used for these two simulations. The MCG takes more CPU time than all other methodologies. The ratio between the total CPU time spent by *EXCELT-ASM* and *EXCELL* is 3.6, it is only 1.9 between *MCG* and *EXCELT-MCI* for these cases.

Table 2 Results for BCAINT Case

Incremental cross-sections	MCG Methodology	MCG Error (%)	EXCELT-MCI Error (%)	EXCELT-ASM Error (%)
$\Delta\Sigma_t^1$	7.1641925E-04	4.33E-03	4.16E-03	0
$\Delta\Sigma_a^1$	1.3236834E-05	5.94E-02	5.10E-02	9.6842E-03
$\Delta\Sigma_s^{1\leftarrow 1}$	6.3281296E-04	2.45E-03	4.95E-03	-1.0572E-03
$\Delta\Sigma_s^{1\leftarrow 2}$	1.5222793E-07	2.20E-02	4.26E-02	0.019696
$\Delta\Sigma_t^2$	3.3613721E-04	-6.30E-02	-7.98E-02	0.017723
$\Delta\Sigma_a^2$	2.6223146E-04	-1.11E-02	1.07E-02	0.022467
$\Delta\Sigma_s^{2\leftarrow 1}$	7.0351729E-05	-9.72E-03	2.20E-02	0.024411
$\Delta\Sigma_s^{2\leftarrow 2}$	7.3685341E-05	-3.26E-01	-4.38E-01	-0.033187

Table 3 Results for BCAOUT Case

Incremental cross-sections	MCG Methodology	MCG Error (%)	EXCELT-MCI Error (%)	EXCELT-ASM Error (%)
$\Delta\Sigma_t^1$	5.8347536E-04	9.40E-06	-5.10E-05	0.
$\Delta\Sigma_a^1$	1.0809423E-05	-0.00011	0.00033	9.69E-05
$\Delta\Sigma_s^{1\leftarrow 1}$	5.1470812E-04	1.28E-05	-3.11E-05	-6.2E-06
$\Delta\Sigma_t^{1\leftarrow 2}$	1.2468421E-07	-6.18E-05	0.00016	0.00019
$\Delta\Sigma_t^2$	2.7318534E-04	0.00039	-0.00076	0.
$\Delta\Sigma_a^2$	2.1568943E-04	3.56E-05	-8.95E-05	0.00021
$\Delta\Sigma_s^{2\leftarrow 1}$	5.7983421E-05	0.00024	-0.0003	0.00025
$\Delta\Sigma_s^{2\leftarrow 2}$	5.7289320E-05	0.00032	-0.0029	-0.0010

Table 4 The total CPU time Spent for the four methodologies

Methodology	EXCELL	EXCELT-ASM	EXCELT-MCI	MCG
Total CPU (s)	48	176	858	1682

In Fig. 5, we plot the size of the tracking file in MB versus the number of regions; we can see that the size of TF, and therefore the capacity of storage, varies linearly with the number of regions. By varying the number of regions from 46 to 2480, the disk space needed increase by a factor 34. For this test case we expect, for 4500 regions, that we will need approximately 1.3 GB of disk space. In Fig. 6, the CPU time spent by the two solvers is shown when varying the number of regions. The CPU time is almost linear for the both solvers. The MCG solver takes, obviously, more CPU time than MCI solver. Both solvers reach the convergence almost at the same number of iterations.

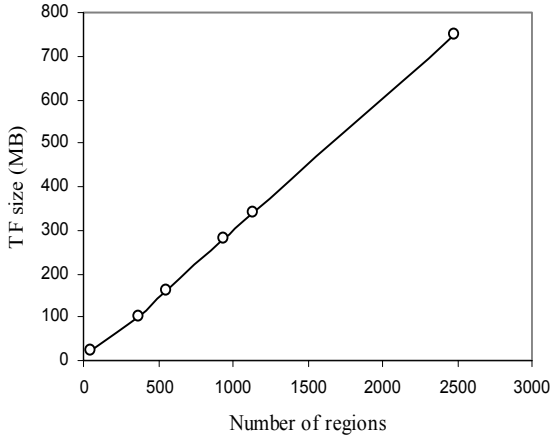

Fig. 5 Tracking file size with varying number of regions.

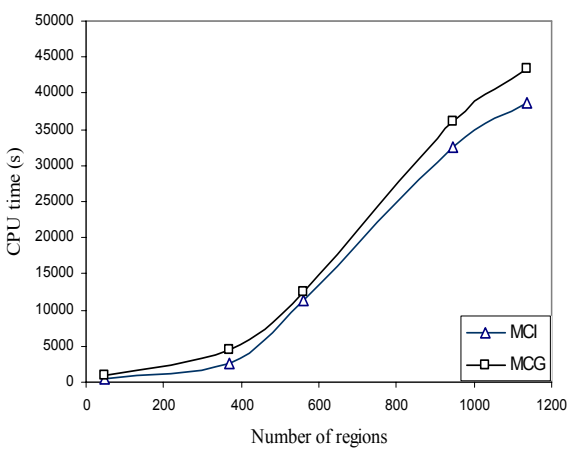

Fig. 6 CPU time for MCI and MCG solvers for different number of regions

5. Conclusion

We have presented a new technique that we have developed in order to solve the large 3D problems. This computational methodology is no longer I/O-bound, comparing to other methodologies that are both CPU and memory consuming. It was a compromise between the need of the huge storage resources and utilization of more CPU power. The computing time is greater than with MCI methodology but eliminating the tracking file reduces drastically the general storage resources needed. However, with the parallelization of the MCG solver, the use of an adequate flux/current initialization and the application of an appropriate acceleration technique, the MCG methodology is the most adapted one for solving large 3D transport problems with a reasonable computing time.

Acknowledgements

This work has been carried out partly with the help of grants from the Natural Science and Engineering Research Council of Canada.

References

1) http://www.intel.com/silicon/mooreslaw.htm
2) G. Marleau, A Hebert and R. Roy, "A User's Guide for DRAGON", Report IGE-174 Rev.3., École Polytechnique de Montréal, December (1997). (see also http://www2.polymtl.ca/nucl)
3) R. Roy, A Hebert and G. Marleau, "A Transport method for Treating Three-Dimensional Lattice of Heterogeneous Cells," *Nucl. Sci. Eng.*, **101**, pp. 227-225 (1989).
4) J. R. Askew, "A Characteristics Formulation of the Neutron Transport Equation in Complicated Geometries," Report AEEW-M 1108, Atomic Energy Establishment, Winfrith (1972).
5) G. J. Wu and R. Roy, "A New Characteristics Algorithm for 3D Transport Calculations," *Ann. Nucl. Energy,* **30**, pp. 1-16 (2003).
6) M. Dahmani, J. Chavez-Guzman, R. Roy and J. Koclas, "Data Management in Parallel Transport Calculation," Proc. Super Computing in Nuclear Applications, Paris, September 23-26, 2003.
7) M. Dahmani, G. J. Wu, R. Roy and J. Koclas, "Development and Parallelization of the Three-Dimensional Characteristics Solver MCI of DRAGON," Proc. PHYSOR-2002, Seoul, Korea, October 7-10, 2002.
8) M. Dahmani, R. Roy and J. Koclas, "Parallel Distribution of Tracking for 3D Neutron Transport Calculation. ANS Conf. M&C 2003, Gatlinburg, Tennessee, April 6-11, 2003.
9) R. Roy, G. Marleau, J. Tajmouati and D. Rozon, "Modeling of CANDU Reactivity Control Devices with the Lattice Code DRAGON," *Ann. Nucl. Energy,* **21**, pp. 115-132 (1994).