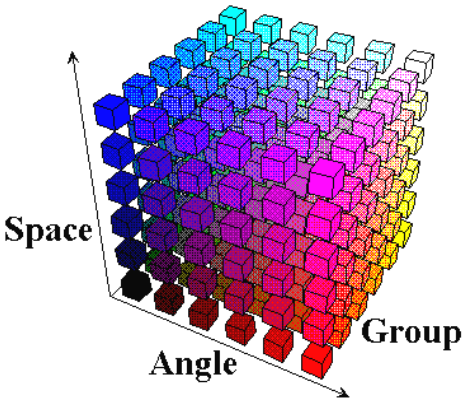
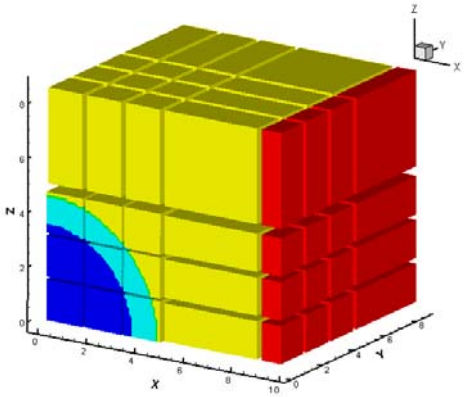
PENTRAN™ Code System

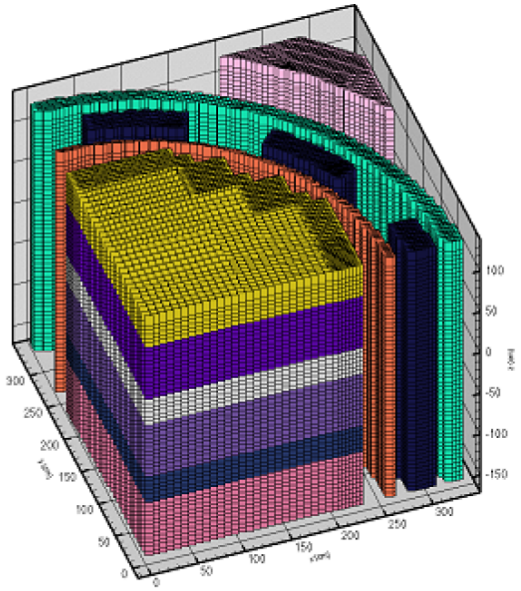
Parallel Environment Neutral-Particle TRANsport

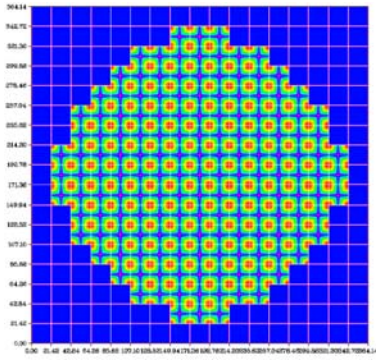
Parallel Distributed Decomposition of
Discrete Ordinates (Sn) in 3-D Cartesian Geometry

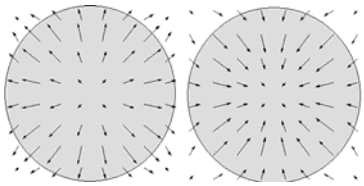
Users Guide to Version 9.4X.5 Series











By Glenn Sjoden
and Alireza Haghighat

November 2008

HSW TECHNOLOGIES LLC
www.hswtech.com

www.hswtech.com

VERSION 9.4X.5 RELEASE NOTES, LEGAL NOTICE, & LICENSING AGREEMENT

This version of *PENTRAN* includes many improvements to the original code architecture, and culminates thirteen years of code development, testing, and evaluation since the first version was introduced in 1996. The *PENTRAN* code runs in ANSI FORTRAN-90, and there are no patches or code settings to be made; the *same* source code compiles and runs on *any* parallel machine, and has been tested on *IBM*, *SUN*, and CRAY/*SGI* parallel computers, and numerous "*Beowulf*"*-class* PC-Clusters. Parallel operation of *PENTRAN* on PC-clusters under the *Linux* Operating System have been extremely successful using the Portland Group, ABSOFT, Intel, and VAST-Linux FORTRAN compilers with any Message Passing Interface (MPI) library set, including either LAM/MPI, Open MPI , or MPICH. In addition, the *PENTRAN* code system has undergone significant testing; the latest applications include a dry storage fuel shipping container and a series of analytical "TIEL" benchmarks.

LICENSING: Parties interested in obtaining further information on *PENTRAN* should contact Dr Glenn Sjoden (glenn@hswtech.com) or Dr Ali Haghighat (ali@hswtech.com) at HSW Technologies, LLC, http://www.hswtech.com, telephone: (352) 871-1099. No-cost licenses are granted to universities through an agreement with RSICC at Oak Ridge National Laboratory, provided assurances for export control are properly completed by individual users as required under applicable US laws and codes; in all cases, no warranties are expressed or implied.

About the Cover: A visualization of a virtual array of processors with decomposition of the transport equation used in *PENTRAN* in the angular, energy, and spatial (*x y z* Cartesian) domains; a pulsed neutron assembly; a 3-D model of the Boiling Water Reactor rendered using *PENMSHXP*, and a fuel pin depicting the "net current" vector displaying the net radiation flow in each fine mesh, and a whole core PWR reactor with 289 sub-assemblies modeled in *PENTRAN*.

PENTRAN SYSTEM USERS GUIDE CONTENTS

FOREWORD

The overall vision of solving very large 3-D radiation transport problems better and faster than ever before was the fundamental reason for the development of the *PENTRAN* code. The idea of the need for such a code came originally from Ali Haghighat, who suggested the distributed parallel Sn code development idea as a PhD dissertation project for Glenn Sjoden. Working together from 1995-1996, Glenn and Ali initally developed the first version of *PENTRAN* on the IBM-SP2 supercomputer using a "new" parallel standard, MPI. *Compaq Digital Visual FORTRAN* and its predecessor releases provided superb programming and debugging vehicles for code development with strict enforcement of ANSI-FORTRAN. Since then, it has been continuously and extensively improved, and has been successfully used on numerous parallel machines. Continuing development and application of *PENTRAN* is leading to new research, providing great insight in discrete ordinates transport theory and parallel processing, although there always remains more to learn. This User's Guide is intended to describe the features, input parameters, and general operating characteristics of *PENTRAN*. Undoubtedly, additions and amendments will be made to this document, as the code is under continuous development and testing.

ACKNOWLEDGMENTS & CONTRIBUTING RESEARCHERS

The authors wish to thank the following for their research and contributions:

Dr Vefa Kucukboyaci contributed significantly to the group window option, and on his own developed and integrated the Restart Capability for large multi-group problems. He also applied *PENTRAN* to real problems that identified key refinements needed to make the initial code release completely tractable for large problems, and greatly assisted in code benchmarking.

Dr Ce Yi has been very instrumental in expanding the mesh generation and code deck development tool PENMSH-eXPress (PENMSH-XP), enabling highly accurate discretizations and visualizations in 3-D. Dr Yi also developed a merged SN – Method of Characteristics (MoC) method with Haghighat and Sjoden in a test version to be released in late 2008.

Dr Gianluca Longoni contributed significantly by developing the P_n-T_n quadrature routines with ordinate splitting, performed numerous tests of the new criticality sequence, and developed a preconditioning strategy to accelerate the criticality eigenvalue calculation.

Dr Melissa Hunter provided valuable experience and many crucial and timely *TWOTRAN-II* and *THREEDANT* solutions for benchmarking.

Dr Benoit Dionne established arbitrary Pn scattering order and efficiency improvements for the code, and tested it for 3-D electron transport applications; he showed that coupled photon-electron transport computations with *PENTRAN* and CEPXS-GS were feasible.

Dr Bojan Petrovic provided *DORT*/*TORT* benchmarks and valuable technical advice on cross section formats, and assisted in applying the code to test problems. He is also the inventor of the directional weighting in the DTW scheme.

Other researchers have used *PENTRAN* to solve large, complex models, demonstrating that the code can handle challenging distributed transport problems. These problems have included the BWR Core Shroud (**Vefa Kucukboyaci**) various applications of Venus and development of *PENXMSH* (**Apisit Patchimpattapong**) the OECD MOX Fuel benchmarks, (**Ce Yi**) the spherical shell cross section simulation and criticality testing (**Mike Wenner**) the Dry Storage Fuel Cask (**Daniel Shedlock**). Large core models and adjoint validations (**Travis Mock** , **Kevin Manalo, and Ryanne Kennedy**) High resolution phantom dosimetry (**Ahmad Al-Basheer** and **Monica Ghita**), TIEL Benchmarks (**Monica Ghita** and **Gabriel Ghita**), 3-D Burnup studies with PENBURN (**Kevin Manalo, Travis Mock, Thomas Plower,** and **Mireille Rowe**), and SNM detector validation with forward and adjoint applications (**Gabriel Ghita**).

PRINCIPAL AUTHOR'S STATEMENTS

Dr Glenn Sjoden *wishes to acknowledge the following:* I would like to thank Ali Haghighat for his sage advice and encouragement to "push the envelope" to reach new levels of achievement, and the US Air Force, for a dynamic, rewarding career in the Armed Forces of the United States. I am also indebted to my wife Patricia, who always demonstrated incredible patience in supporting my work, as did my 3 children. Finally, and not the least, I thank God for wisdom to develop this code.

-Glenn E. Sjoden, PhD, PE

Dr Ali Haghighat *wishes to ackowledge the following:* I thank my wife Mastaneh and my son Aarash for their faithful patience and support throughout my work.

-Alireza Haghighat, PhD

1. TECHNICAL OVERVIEW

The *PENTRAN (Parallel Environment Neutral-particle TRANsport)* code was initially developed with the following research goals and code development objectives in mind:

RESEARCH GOALS

(1) To demonstrate the parallel scalability of a 3-D transport code designed from scratch for scalable parallel implementation with full variable decomposition on distributed memory and computing architectures, (2) To implement and test parallel algorithm phase space decomposition strategies based on problem physics and load balancing/parallel efficiency issues, (3) To provide for and investigate the benefits of adaptive discrete ordinates spatial differencing schemes as they relate to problem physics, decomposition, and load balancing, (4) To test the benefits of differencing and acceleration methods in conjunction with items (2) and (3) above, and (5) to provide higher order strategies for variable mesh coupling for increased accuracy and scalability in large transport problems.

SPECIFIC OBJECTIVES USED FOR CODE DEVELOPMENT

The code should iteratively solve 3-D Cartesian, multigroup problems with anisotropic scattering via P_L Legendre moments, with level symmetric S_N angular quadratures. Industry standard *FIDO* input with vacuum and specular reflective boundaries should be allowed for. Sources should be definable as volumetric or plane surface incident fluxes, which may vary with space, angle, and energy. Further, a parallel memory structure should be used, where memory intensive arrays should be defined for local (as opposed to global) maximum dimensions to reduce storage extent and overhead. In addition, the code should permit varying mesh cells with a coarse grid topology. Also, the code should employ coarse mesh rebalance acceleration, as a minimum. Various differencing schemes should be readily selectable (e.g. linear diamond, directional theta-weighted, etc) via adaptive algorithm logic, and based on problem physics; "smart" ordering of scattering sweeps, such as Alternating Direction Sweeps, or *ADS* (Haghighat, 1992), and other tools, many readily extracted from the literature, should be implemented where practical, again according to problem physics. The code should be written in ANSI FORTRAN, and parallelized for message passing using the standardized MPI Message-Passing Interface library for portability to most any distributed memory parallel machine architecture (Gropp, et al, 1994).

CODE DEVELOPMENT STATUS

Coding of *PENTRAN* began in June 1995. It is now at the point of being fully mature. In addition, code refinement and development continues today. A fully parallel 3-D, two-grid (using "medium" and "fine" mesh transport grids) version of the code is in place in ANSI FORTRAN-90 and is currently ~40,000 lines.

Scalable parallel process testing with complete angular, energy, and spatial domain decomposition, block adaptive (discontinuous) meshing, first order Taylor Projection Mesh Coupling, fully definable sources, vacuum and reflective boundary conditions, multigroup, arbitrary order anisotropic scattering, multigrid coarse mesh zoned rebalancing, and selectable differencing for both forward and adjoint transport, with self-tuning memory has been exercised over and over on numerous problems. Test problems have demonstrated exact agreement (within the convergence criteria) with TWOTRAN-II, THREEDANT, DORT, and TORT production codes for all fixed source problems tested.

In addition, the code has been experimentally benchmarked in 3-D using models of the Venus-3 Reactor owned by SCK in Belgium and extensively tested (with excellent results) using the Kobayashi 3-D benchmark problems. Criticality eigenvalue problem results compared between TWOTRAN-II and *PENTRAN* also yielded excellent agreement, as did recent tests directly compared to MCNP computation in a variety of applications, including the 2-D MOX Benchmark. Most recently, *PENTRAN* was applied to Ganapol's "TIEL" quasi-analytic benchmark suite; results were excellent, and the code demonstrated very consistent numerics at high quadraures and PL orders.

PENTRAN has been applied to all aspects of transport theory research. *PENTRAN* is available for no-cost use at universities that meet certain acceptance criteria under a limited license agreement. A minimum requirement includes the need for an 8-processor Linux based cluster or larger parallel computing platform. Commercial licenses are available, and discount programs are available for transport theory related research contracts awarded at the University of Florida.

Current University partners include:

The Ohio State University, Nuclear Engineering Program

The University of Florida , Department of Nuclear and Radiological Engineering

The University of Florida , Florida Institute of Nuclear Detection and Security

UNIQUE *PENTRAN* CODE FEATURES

ANSI Code, Parallel I/O. The *PENTRAN* code has been designed from scratch in ANSI FORTRAN *and adapted to* ANSI FORTRAN-90 (to take advantage of dynamic array allocation) to be parallelized on a distributed memory, multiple instruction, multiple data (MIMD) machine architecture using the standard Message Passing Interface (MPI) message passing language. Any distributed memory MIMD parallel system running FORTRAN with MPI could be used to execute *PENTRAN* without modification. All Input/Output (I/O) is performed by each processor in parallel (as required) to the fullest extent possible. This includes input processing, initializations, and file output. Output files are written only for the local energy groups and spatial cells processed on a given processor (with local angles written in the case of angular fluxes), in accordance with local memory partitioning of the problem to *n* processors.

Parallel Memory. Parallel memory utilization was a paramount design goal. All dimensions for memory intensive arrays (e.g. angular fluxes) are partitioned **locally**. That is, these arrays only need be as large as required for the largest locally stored spatial grid, number of local energy groups, and local angular sweep octants based on the problem being solved. Tuning of memory parameters also occurs automatically in a two-level memory model to minimize memory useage. This design is possible due to the independent memory of each processor on a distributed memory MIMD machine; in theory, if the problem becomes larger, one simply can add more processes (with further decomposition) to obtain a solution.

Space, Angle, & Energy Decomposition. Full phase space decomposition is available (space, angle, and energy), with fully automatic scaling of the problem to *n* processes (based on a user specified *decomposition weighting vector*). Also, a specific number of processes can be locked for each decomposition variable if desired; or, decomposition scaling in any one variable can be blocked (restricted to one processor). The automatic scaler/mapper will attempt to best adapt decomposition to the user's weighting vector and the number of processes assigned at execution. Following decomposition, *PENTRAN* includes automatic load balancing and red-black options to maximize efficiency. To assist the user, *PENTRAN* can be tasked to automatically set F90 memory parameters to optimal settings.

Communicators. To further maximize parallel execution efficiency, communication among processors is, where practical, carried out only between processors specifically involved in a given task, e.g. for those processors all computing transport sweeps through particular coarse cells in the same energy group in a multigroup transport problem. This is accomplished by MPI process "communicators" constructed during problem initialization to exchange data between specific groups of processors. The number of communicators constructed is minimized to the number uniquely required to reduce network buffering

overhead. The communicator structure is fully automatic and completely transparent to the user.

Adaptive Differencing strategy. Different differencing schemes can be assigned to different coarse meshes; therefore, differencing can be *adaptive* based on each coarse mesh. Linear Diamond Differencing without fixup (DD) and with set-to-zero fixup (DZ), Directional Theta-Weighted (DTW), Eponential-Directional Iterative (EDI), and Exponential Directional Weighted (EDW) differencing schemes are independently selectable for each coarse mesh. The DD and EDW schemes are included only for test purposes and not intended for routine use (it is strongly recommended that DD *not* be used since it is without fixup). An **adaptive** (DZ,DTW,EDI) scheme to automatically shift DZ to DTW (in the event a DZ negative flux fixup is detected), and shift from DTW to EDI (if a DTW angular flux weight factor that is too high is encountered) is built in (default) and user-adjustable. Performance metrics are available for each differencing scheme, enabling the user to assess differencing performance in each coarse mesh for either the medium and/or fine mesh grids.

Coarse, Medium, Fine Grids, and TPMC. Variable 3-D meshing (along each of x, y, and z) is available between different coarse meshes, with "medium" and "fine" multigrid meshing within each coarse cell. (Coarse cells are set to contain heterogeneous zones, where differencing is performed on medium and fine grids within each coarse mesh cell; this is often referred to as "Block AMR", or "Block Adaptive Mesh Refinement"). As a consequence of variable 3-D meshing, Different spatial aspect ratios may be used along any set of axes in 3-D (e.g. "boxoids" or "voxels") within each coarse mesh, and mesh grid distances need *not* be the same along adjoining surface boundaries between different coarse mesh cells. Mesh interpolations for angular fluxes between adjacent coarse cell surfaces (where coarse mesh cells are not necessarily on the same processor) are accomplished using Taylor Projection Mesh Coupling (TPMC). This is performed in all transport sweeps to increase accuracy and minimize the information loss when moderate and dense mesh grids are interfaced on a common boundary between two adjacent coarse cells. Note that particle conservation is strictly applied in this process. A simplified multigrid method that can provide speedup and conserves memory uses a single set of arrays for (local) angular fluxes on both the medium and fine mesh grids. The medium grid angular fluxes are relaxed to convergence using a tolerance less than that for the fine grid, whereupon they are projected onto the fine grid (overwriting the last medium grid values) using a new Taylor Projection Mesh Coupling (TPMC) scheme. The preconditioned fine grid values are then iterated to final convergence. This is a "simplified" multigrid method, since no residual corrections are projected with cycling between the medium and fine grids. Transport solution projections are only performed from medium to fine grids, and accelerate the solution by preconditioning the fine grid in a nested iteration. Varying speedups are possible with this simplified multigrid scheme, depending on the problem, with the advantage of conserving memory by not storing the medium grid explicitly.

Rebalance & ADS. Standard coarse mesh rebalancing (CMR), implemented with restrictions and damping as Partial Current Rebalancing (PCR), and/or system rebalancing (SR) is available for acceleration, and can be used simultaneously with multigrid. There is no restriction on the differencing scheme that can be used with either CMR (PCR) or multigrid. Each processor independently performs rebalance using a direct solution (Cholesky factorization in the case of CMR (PCR)) over a zoned subset of coarse meshes (selected by the user) to obtain group rebalancing factors following transport sweeps. Further, rebalancing factors are used to scale only the *scalar* flux rather than the *angular* flux, as this prevents the need for message passing to complete additional angular quadratures prior to the next source iteration (if angular decomposition is invoked). Angular sweeps are ordered/decomposed in a sequence for Alternating Direction Sweeping (ADS). This can drastically increase convergence in some problems when used with rebalance (Haghighat, 1992). A new preconditioned Simplified Sn scheme is under development, and is supported in part by this version.

Flux Moment Preconditioning. *PENTRAN* now incorporates options that allows the user to precondition the 0^{th} and 1^{st} flux moments in the initial S_N source iteration sweep using an effective initial guess to the polar angle flux moments to provide acceleration to the S_N source iteration. The *REPRO tool* preconditions the problem flux moments using the previous transport run result, and therefore helps to accelerate the solution (Plower, 2007). This stemmed from an initial effort by Longoni, Haghighat, and Sjoden to implement the Simplified Sn Equations (SS_N) to serve as a low order synthetic acceleration scheme to accelerate the standard discrete ordinates source iteration, which suffers from a spectral radius close to one (slow convergence) with significant scattering in the problem. Due to numerical stability issues, a coupled residual iterative algorithm for an intrinsic SS_N-S_N acceleration was found to be impractical. An alternative approach using the fully converged SS_N flux moment solution as an independent flux preconditioner (rendered via a parallel code developed by Longoni and Haghighat called PENSS_N) for the S_N transport source iteration sweep can result in significant overall acceleration (on the order of 300% for SS_N+S_N compared to S_N alone) for criticality eigenvalue problems (Longoni, 2004).

Subdomains. The spatial coarse mesh structure in *PENTRAN* fundamentally defines rebalance subdomains, parallel spatial decomposition subdomains, and adaptive Sn differencing subdomains. Although spatial decomposition is not required, more than a single coarse mesh must be defined to permit spatial decomposition on more than a single processor. Since a processor synchronization is performed following completion of a coarse mesh, a sufficient number of fine meshes should be contained within each coarse mesh to maintain computational load granularity and rebalance integrity. One restriction is that there be an equal number of defined coarse cells, energy groups, and directions partitioned to each processor; this is needed for parallel synchronization to prevent "deadlocks." *PENTRAN* performs this automatically to the extent permissible, again based on the decomposition weight vector (**decmpv**) specified by the user.

Fixed Sources. Fixed sources can be defined as volumetric or as planar boundary fluxes; sources can have completely arbitrary spatial, angular, and/or energy distributions. Spatial and angular source distributions can be defined independently by energy group. Group dependent source scale factors can be specified. Criticality eigenvalue solutions are also implemented and have been benchmarked using test problems. Criticality problems and problems that include upscatter can not be used with the group window option or the restart option; these features are currently reserved for downscatter-only fixed source problems.

Quadratures and Legendre Scattering Expansion. Full support for level-symmetric angular quadratures through S_{20}, and arbitrary S_N order (to the limit of memory allocation) for P_n-T_n angular quadratures fully conserving even and odd moment conditions. Also, Legendre scattering order (P_N) is now completely arbitrary (to the limit of memory allocation). In addition, *PENTRAN* now supports the ability to implement user biased angular quadrature sets (over-riding built in quadratures if the file "quadset.pen" containing the user defined quadratures is placed in the local problem execution directory).

***FIDO* & Execution.** Industry standard, free field format "*FIDO"* input is used, with standardized order for cross sections. All cross sections are assumed to be blended and assembled outside of *PENTRAN*. Provisions are made for row, column, binary, and *ORNL-GIP*-binary formats, with and without Legendre coefficients multiplied in advance. Wall-clock time, maximum iterations, and convergence tolerance are all independent means of execution control, particularly useful for batch data processing.

PENDATA. A data management utility, *PENDATA,* seamlessly gathers data automatically following a parallel *PENTRAN* run, and provides several options for the user in stripping results from parallel output files, including data extractions from binary file storage. This is an essential utility for a code with parallel I/O.

***PENMSH-XP* Mesh or Results Generation.** Planar images (*z*-levels) or 3-D TECPLOT renderings are available using the *PENMSH-XP* utility. This utility generates a 3-D Cartesian mesh and can be used to automatically generate a *PENTRAN* input deck. Graphical schematics of 3-D problem geometry are automatically generated for use in the *TECPLOT*™ plotting/graphics package. 2-D slices are also rendered by *PENMSH-XP* as .png images; note the 3-D rendering of images (and flux results) requires minimal effort in *TECPLOT*. Use of *PENMSH-XP* reduces problem input preparation to a trivial process, even for large, intricate geometries

Platform Independent Code. The code is written in ANSI FORTRAN-90; it has been implemented on a single-processor *Pentium*-PC, an *IBM RISC*-6000 Workstation, *SUN* Workstation, and in parallel on the *IBM Scalable PowerParallel System-2* (*IBM-SP2*) and *SGI Origin-2000* supercomputers, and on a variety of *Linux* based PC Clusters, and most recently on an Apple G5 cluster. Benchmark testing has demonstrated that *PENTRAN* is greater than 97% parallelizeable, resulting in excellent parallel speedups. Actual performance depends

on the problem being solved, differencing and acceleration methods used, problem load balancing, red-black coloring, and applied decomposition strategy. Also, due to the inherent scalable parallel memory structure used, increasingly large 3-D problems that could not be solved on single processor platforms and/or within a reasonable time period can be solved in parallel using *PENTRAN* by adding processors.

2. THEORY AND APPLICATION

MULTIPROCESSING TERMINOLOGY

Parallel Processing. The idea of multiprocessing and the evolution of parallel computing began in the late 1950s and continues today. Due to the ten-fold increase in computational performance during each five year period since then, multiprocessing has made great progress. The reasons for attempting to solve any numerically intensive problem with multiprocessing are simple: to reduce execution time, obtain higher accuracy, and/or solve problems that are larger than can be solved using a traditional, single processor von Neumann architecture (Freeman and Phillips, 1992). It is useful to define some common parallel processing terms; they are only briefly mentioned here and will appear throughout this manual.

Machine Classes. There are four classes of machines, as introduced by Flynn (1972):

SISD - Single Instruction/Single Data Stream (traditional von Neumann machine)
SIMD - Single Instruction/Multiple Data Stream (lock-step arrays, vector machines)
MISD - Multiple Instruction/Single Data Stream (all tasks contribute to one data set)
MIMD - Multiple Instruction/Multiple Data Stream (multiple independent tasking)

Shared and Distributed Memory. *Shared memory* MIMD systems are constructed so that each processor has global memory access and are typically limited to tens of processors due to the large number of physical connections to the memory map. *Distributed memory* MIMD parallel computers maintain completely independent memories, where processors exchange information by message passing over a high speed network; each processor independently executes code and can perform independent input/output (I/O) if allowed for in the parallel algorithm. Distributed memory processors, viewed as "nodes," are typically connected together using a variety of topologies, and can range in number into the thousands.

Speedup and Efficiency. Parallel performance models are necessary for analyzing and quantifying parallel speedup and efficiency. Parallel *speedup* (S_P) measures the overall reduction in computing time to solve a problem. It is defined as the wall-clock time on a serial (single) processor divided by the wall-clock time on P processors:

$$S_P = T_S / T_P \tag{2.1}$$

Parallel *efficiency* (E_P) measures the economic advantage of the parallelization by comparing the speedup factor to the allocated number of processors (Freeman and Phillips, 1992):

$$E_P = S_P / P \tag{2.2}$$

It is assumed here that the parallel algorithm *overhead,* the extra executable code and storage required to expedite parallel execution, is negligible during single processor execution. This is a typical convention often adopted in measuring parallel speedup (Werner, 1981).

Amdahl's Law. Also, an upper bound on anticipated parallel speedup can be determined by applying the *Amdahl's Law*, which states that given the fraction of a code that is parallelizeable: $0<f_P<1$, the maximum observed speedup for P processors with parallel communication time (T_C) is equal to:

$$S_P = \frac{1}{(1-f_P) + f_P/P + T_C/T_S} \tag{2.3}$$

where in the limit of an infinite number of processes (assuming zero communication time):

$$\lim_{p\to\infty} S_P \to \frac{1}{(1-f_P)} \tag{2.4}$$

Therefore, from equation (2.4), if the parallelizeable portion of a code is $f_P = 0.80$, the maximum *theoretically* observed speedup is 5.0 regardless of the number of additional processors added to the problem. In reality, due to increasing parallel instruction and communication overhead with the addition of more and more processors, there will be a point (depending on f_P , system architecture, and problem size) where adding more processors leads to extremely low efficiencies. This may be irrelevant if the code is scalable in memory (as in the case of *PENTRAN*), where, regardless of speed, the problem requires some number of processors to be solved *at all*. A plot of Amdahl's law (assuming *Tc*=0) depicting maximum theoretical speedup based on parallel fraction and associated efficiency as a function of processors is provided for illustration below.

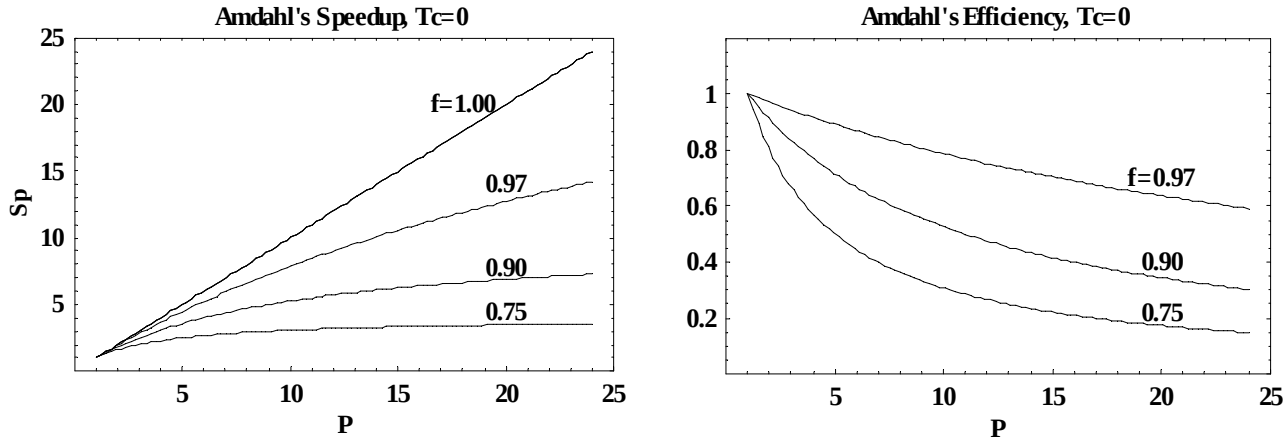


Fig. 2.1. Amdahl's Law, yielding maximum theoretical speedup (Left) and parallel efficiency (Right), depicted for various parallel code fractions (f) and number of processors (P) (Sjoden, 1997).

Load Balancing and Granularity. Other important terms include *load balancing* and *granularity*. Load balancing involves distributing work to processors evenly to maximize parallel efficiency. Algorithm granularity is a qualitative term that refers to the number of process operations that can be executed by each processor before a synchronization (or communication) of the processors must be implemented. These synchronizations can be viewed as serial barriers that limit parallel performance. Conventional definitions of grain size are:

- fine grain - unit numbers of operations before synchronization
- medium grain - tens of operations before synchronization
- coarse grain - hundreds (or more) of operations before synchronization

Computation/Commnication Ratio. The computation/communication ratio varies from machine to machine; this is the ratio of the CPU instruction speed in flops (floating point operations per second speed, often stated in Mflops, or millions of flops) attainable on the system to the relative speed of data transfer between processors. The speed of data transfer is related to communication latency, or the time required to send a zero byte-length message, and the communication bandwidth, in megabytes per second. The computation/communication ratio is often more of an issue on distributed memory, message passing machines, as network communication data rates are typically orders of magnitude slower than typical Mflop rates (Gropp, et al, 1994).

Dedicated vs. Non-dedicated. Parallel machine availability is a practical performance issue. If processors are available to users in a dedicated mode, then during parallel execution, a single user has complete control of the processors, and parallel performance can be accurately determined. Alternatively, in a non-dedicated (interactive) mode, processors are simultaneously available to many users; absolute parallel performance may be difficult to verify in this case.

Parallel Scalability. With regard to a working definition of parallel scalability, scalable algorithms maximize computation to communication ratio, minimize serial operations, maximize algorithm and data parallelism, and maximize efficiency for the architecture (shared versus distributed memory) (Gerner, 1995). While there is a great deal more that can be mentioned about multiprocessing fundamentals and terminology, more complete discussions can be found elsewhere (see Freeman and Phillips, 1992, and Gropp, et al, 1994, and Chandy and Misra, 1988).

Multigroup Transport Equation. Deterministic discrete ordinates approximations of the transport equation invoke a discretization of the energy, angle, and space variables. Discretization of energy is accomplished by spectrally averaging over energy groups (g=1,G), from high to low energies, resulting in the multigroup transport formulation. In steady state, the multigroup transport equation is (Lewis and Miller, 1993); the left side includes loss by leakage and collision, and scatter, fission, and independent sources are on the right:

$$\hat{\Omega}\cdot\nabla\psi_g(\vec{r},\hat{\Omega})+\sigma_g(\vec{r})\psi_g(\vec{r},\hat{\Omega})= \tag{2.5}$$

$$\sum_{g'=1}^{g}\int_{4\pi}d\Omega'\sigma_{s\,g'\to g}(\vec{r},\hat{\Omega}'\cdot\hat{\Omega})\psi_g(\vec{r},\hat{\Omega}')+\frac{\chi_g}{k_o}\sum_{g'=1}^{G}\int_{4\pi}d\Omega'\nu\sigma_{f\,g'}(\vec{r})\psi_{g'}(\vec{r},\hat{\Omega})+q_{ind\,g}(\vec{r},\hat{\Omega})$$

Angular Variable. Note that the angular variable is normalized on the unit sphere in the above formulation, so that integration over Ω is expressed in terms of the polar angle cosine μ and azimuthal angle φ as:

$$\int_{4\pi}d\Omega=\int_{-1}^{1}\frac{d\mu}{2}\int_{0}^{2\pi}\frac{d\varphi}{2\pi}=1 \tag{2.6}$$

Hereafter, this is implicitly assumed. The scattering term is then expanded using a truncated set of spherical (surface) harmonics, with
$\hat{\Omega}\to<\theta,\varphi>$, $(\hat{\Omega}'\cdot\hat{\Omega})\to(\mu_o)$, $\mu=(\cos\theta)$, $\mu'=(\cos\theta')$:

$$\sigma_{s\,g'\to g}(\vec{r},\mu_o)=\sum_{l=0}^{L}(2l+1)\sigma_{s\,g'\to g,l}(\vec{r})P_l(\mu_o) \tag{2.7}$$

$$\sigma_{s\,g'\to g,l}(\vec{r})=\int_{-1}^{1}\frac{d\mu_o}{2}\sigma_{s\,g'\to g}(\vec{r},\mu_o)P_l(\mu_o) \tag{2.8}$$

$$\mu_o=\mu\mu'+(1-\mu^2)^{1/2}(1-\mu'^2)^{1/2}\cos(\varphi-\varphi') \tag{2.9}$$

The Legendre polynomial $P_l(\mu_o)$, using the Legendre Addition Theorem, is:

$$P_l(\mu_o)=\frac{1}{(2l+1)}\sum_{k=-l}^{l}Y_{l,k}^{*}(\theta',\varphi')Y_{l,k}(\theta,\varphi) \tag{2.10}$$

The spherical (surface) harmonics $Y_{l,k}$ and $Y_{l,k}^{*}$ are defined in terms of the Associated Legendre polynomials and an exponential term:

$$Y_{l,k}(\theta,\varphi)=\sqrt{(2l+1)\frac{(l-k)!}{(l+k)!}}P_l^k(\mu)\exp(ik\varphi) \tag{2.11}$$

$$Y_{l,-k}(\theta,\varphi)=(-1)^k Y_{l,k}^{*}(\theta,\varphi) \tag{2.12}$$

Using equations (2.10), (2.11), and (2.12), $P_l(\mu_o)$ can be written:

$$P_l(\mu_o) = P_l(\mu)P_l(\mu') + 2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu)P_l^k(\mu')\cos(k(\varphi-\varphi')) \tag{2.13}$$

By trigonometric identity:

$$\cos(k(\varphi-\varphi')) = \cos(k\varphi)\cos(k\varphi') + \sin(k\varphi)\sin(k\varphi') \tag{2.14}$$

The vectors $\hat{\Omega} \rightarrow <\theta, \varphi>$ on the unit sphere can be expressed as a set of direction cosines projected parallel to the *x, y,* and *z* axes, respectively, as $<\mu, \eta, \xi>$ Note that η and ξ can be expressed in terms of polar angle cosine μ and the azimuthal angle φ:

$$\eta = \sqrt{1-\mu^2}\cos(\varphi) \tag{2.15}$$

$$\xi = \sqrt{1-\mu^2}\sin(\varphi) \tag{2.16}$$

A 3-D Cartesian geometry (using a right handed coordinate system) is shown in Figure below.

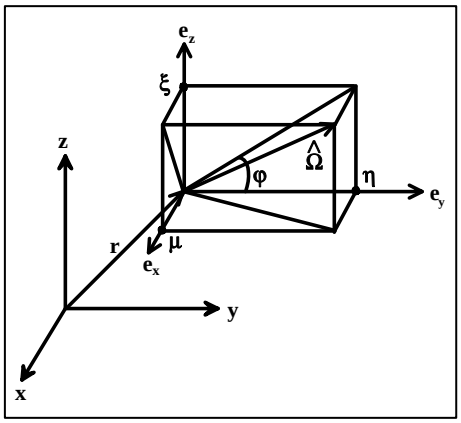


Fig 2.2: 3-D Cartesian Geometry

If the streaming operator $\hat{\Omega}\cdot\nabla$ in equation (2.5) is expanded in 3-D Cartesian coordinates, it becomes:

$$\hat{\Omega}\cdot\nabla = \mu\frac{\partial}{\partial x} + \eta\frac{\partial}{\partial y} + \xi\frac{\partial}{\partial z} \tag{2.17}$$

3-D Cartesian Boltzmann Transport Equation. Substituting equations (2.7), (2.13), (2.14), and (2.17) into equation (2.5), we obtain the Legendre expanded multigroup form of the transport equation in 3-D Cartesian geometry (Lewis and Miller, 1993, and Bell and Glasstone, 1985) considering only fission sources:

$$(\mu\frac{\partial}{\partial x}+\eta\frac{\partial}{\partial y}+\xi\frac{\partial}{\partial z})\psi_g(x,y,z,\mu,\varphi)+\sigma_g(x,y,z)\psi_g(x,y,z,\mu,\varphi)= \tag{2.18}$$

$$\sum_{g'=1}^{G}\sum_{l=0}^{L}(2l+1)\sigma_{s,g'\to g,l}(x,y,z)\{P_l(\mu)\phi_{g',l}(x,y,z)+2\sum_{k=1}^{l}\frac{(l-k)!}{(l+k)!}P_l^k(\mu)\cdot$$

$$[\phi^k_{C\,g',l}(x,y,z)\cos(k\varphi)+\phi^k_{S\,g',l}(x,y,z)\sin(k\varphi)]\}+\frac{\chi_g}{k_o}\sum_{g'=1}^{G}\nu\sigma_{f\,g'}(x,y,z)\phi_{g',0}(x,y,z)$$

where $\mu = x$ direction cosine for angular ordinate
$\eta = y$ direction cosine for angular ordinate
$\xi = z$ direction cosine for angular ordinate
ψ_g = group g angular particle flux (for groups g=1,G)
φ = azimuthal angle constructed from $\arctan(\xi/\eta)$, with proper phase shift
σ_g = total group macroscopic cross section
l = Legendre expansion index ($l = 0, L$), L=o or odd truncation
$\sigma_{sg'\to g,l}$ = l th Legendre moment of the macroscopic differential scattering cross section from group $g'\to g$ (Equation (2.7))
$P_l(\mu)$ = l th Legendre polynomial
$\phi_{g',l}$ = l th Legendre scalar flux moment for group g
$P_l^k(\mu)$ = l th , k th Associated Legendre polynomial
$\phi^k_{C\,g',l}$ = l th , k th Cosine Associated Legendre scalar flux moment for group g
$\phi^k_{S\,g',l}$ = l th , k th Sine Associated Legendre scalar flux moment for group g
χ_g = group fission distribution constant (neutrons)
k_o = criticality eigenvalue (neutrons)
$\nu\sigma_{f\,g}$ = group fission production (neutrons)

The flux moments, $\phi_{g',l}$, $\phi^k_{C\,g',l}$ and $\phi^k_{S\,g',l}$ are defined in terms of μ' and φ' as:

$$\phi_{g',l}(x,y,z)=\int_{-1}^{1}\frac{d\mu'}{2}P_l(\mu')\int_0^{2\pi}\frac{d\varphi'}{2\pi}\psi_{g'}(x,y,z,\mu',\varphi') \tag{2.19}$$

$$\phi^k_{C\,g',l}(x,y,z)=\int_{-1}^{1}\frac{d\mu'}{2}P_l^k(\mu')\int_0^{2\pi}\frac{d\varphi'}{2\pi}\cos(k\varphi')\psi_{g'}(x,y,z,\mu',\varphi') \tag{2.20}$$

$$\phi^k_{S\,g',l}(x,y,z)=\int_{-1}^{1}\frac{d\mu'}{2}P_l^k(\mu')\int_0^{2\pi}\frac{d\varphi'}{2\pi}\sin(k\varphi')\psi_{g'}(x,y,z,\mu',\varphi') \tag{2.21}$$

The adjoint transport operator H^+ can be derived using the adjoint identity for real valued functions and the forward multi-group transport operator, where $\langle\ \rangle$ represents integration over all independent variables:

(2.22) $$\left\langle \psi_g^+ \, H \, \psi_g \right\rangle = \left\langle \psi_g \, H^+ \psi_g^+ \right\rangle$$

(2.23) $$H = \hat{\Omega} \cdot \nabla + \sigma_g(\vec{r}) - \sum_{g'=1}^{G} \int_{4\pi} d\Omega' \, \sigma_{s\, g' \to g}(\vec{r}, \hat{\Omega}' \cdot \hat{\Omega})$$

The angular adjoint (importance) function is ψ_g^+, and H is the forward transport operator. Applying the adjoint boundary condition that particles leaving a bounded system have an importance of zero in all groups (converse of the vacuum boundary condition in a forward calculation) with the above equations, and requiring a continuous importance function mathematically leads to the multi-group adjoint transport operator:

(2.24) $$H^+ = -\hat{\Omega} \cdot \nabla + \sigma_g(\vec{r}) - \sum_{g'=1}^{G} \int_{4\pi} d\Omega' \, \sigma_{s\, g \to g'}(\vec{r}, \hat{\Omega} \cdot \hat{\Omega}')$$

This operator is not Hermitian, and has units of inverse length. Also, the minus sign on the streaming term is an indicator that adjoint particles travel along a reversed direction, where scattering progresses from group *g back* to other groups *g'* (those groups formerly contributing to group *g* in the forward transport operator).

Note that the adjoint transport operator, as in Equation (2.24), can also be derived directly from physical principles based on the conservation of neutron importance. This type of derivation does not require a strict mathematical application of vector identities and a zero importance condition for particles leaving the system, yet this approach does result in the same operator as given in Equation (2.24). No matter how it is derived, the adjoint function is associated with the importance of neutral particles with respect to some objective (Bell and Glasstone, 1985). For example, the particles can be neutrons and the objective could be is an absorption in He-3 to yield an (n,p) reaction.

To solve for the adjoint function, a forward transport solver can be directly used if angular, cross section, and energy group indices are properly treated. That is, a forward transport algorithm can be used to solve an adjoint transport problem if the group cross sections and sources are transposed, including the cross section scattering matrix, re-ordered to commence from group G to 1. In this case, all angles are considered to be defined implicitly in opposite (negative) directions, with group G adjoint sources input and reported into the

forward code as group 1, and group G-1 adjoint sources input and reported as group 2, etc. This is precisely how the adjoint is solved for using the *PENTRAN* system, although the transpose of the forward cross sections is performed internally by the code. Once solved for, the dimensionless adjoint function provides the neutron or photon importance throughout the problem phase space relative to a particular response, defined by the adjoint source. The adjoint can then be used in several ways.

Ultimately, we wish to predict the overall response *R* of a detector in counts per second using the adjoint function. For problems of this type, the count rate *R* is based on the number of absorption reactions per unit time in the He-3 gas (leading to an (n,p) reaction) caused by neutrons. If such a fixed neutron source/detector problem is proposed, the neutron flux must satisfy the transport equation:

$$H\psi_g = q_g \tag{2.25}$$

and the inhomogeneous adjoint equation should be satisfied with an adjoint source aliased to the group detector response cross section $\sigma_{d\,\mathrm{g}}$, according to:

$$H^{+}\psi_g^{+} = \sigma_{d\,\mathrm{g}} \tag{2.26}$$

Applying Equations (2.22). (2.25), and (2.26), and integrating over all variables results in a very useful expression for detector response *R*:

$$R = \left\langle \psi_g \sigma_{d\,\mathrm{g}} \right\rangle = \left\langle \psi_g^{+} q_g \right\rangle \tag{2.27}$$

This indicates that the detector response can be obtained by complete integration of the source distribution with the adjoint function obtained using Equation (2.27)—for any arbitrary source distribution. Therefore, note that *R* can be computed directly from the results of either of (i) *several* forward transport computations for each neutron source, or (ii) a single adjoint transport computation. For a more detailed discussion of the adjoint, we refer to several available references (Lewis and Miller, 1993; Bell and Glasstone, 1985).

In reference to a He-3 detector, in the traditional forward case, the standard Boltzmann transport equation must be solved to yield a scalar flux for a specific neutron source *q* placed inside the Sample Chamber. For a number of different types of sources, *separate computations* must be performed for *each source*. Once the neutron flux is determined for each scenario, the detector response in the He-3 could be numerically computed in the conventional manner using group cell fluxes and detector cross sections, as given in Equation (2.28).

$$R = \int_{V_d, \forall E} \phi(x,y,z,E)\, \sigma_d(x,y,z,E)\, dx\, dy\, dz\, dE \approx \sum_{\substack{\Delta Vi \in V_d \\ g=1,G}} \phi_{g,i}\, \sigma_{d\,g,i}\, \Delta V_i \tag{2.28}$$

where: R = Detector Response, counts/s

V_d = Detector Volume, cm^3

(x, y, z) = Spatial Location(s) of He - 3 Tubes

$\phi(x, y, z, E)$ = Spatial, Energy Dependent Scalar Flux, $\text{n/cm}^2\text{/s}$, from quadrature of ψ

$\sigma_d(x, y, z, E)$ = Spatial, Energy Dependent Detector Cross Section, cm^2/cm^3

$\phi_{g,i}$ = *i*th Cell Scalar Flux for Group g, $\text{n/cm}^2\text{/s}$, from quadrature of $\psi_{g,i}$

$\sigma_{d\,g,i}$ = *i*th Cell Detector Cross Section for Group g, cm^2/cm^3

ΔV_i = *i*th Cell Volume, cm^3

In the adjoint case, the adjoint transport equation must be solved using an *adjoint source* that is equal in magnitude to the detector response cross section placed in each location, for this discussion, occupied by He-3 tubes. This yields the adjoint function throughout the problem phase space, and represents the importance of neutrons in each spatial location and energy group relative to a response in the collective of He-3 tubes. Then, the He-3 tube count rate R due to *any* neutron source q placed in a specified location is computed as given in Equation (2.29).

$$R = \int_{V_q, \forall E} \phi^{+}{}_{d}(x', y', z', E)\; q(x', y', z', E)\, dx'\, dy'\, dz'\, dE \approx \sum_{\substack{\Delta V_i \in V_d \\ g=1,G}} \phi^{+}{}_{d\,g,i}\; q_{g,i}\; \Delta V_i \tag{2.29}$$

where: R = Detector Response, counts/s

V_q = Source Volume, cm^3

(x', y', z') = Spatial Location of Nonzero Source Cells

$\phi^{+}{}_{d}(x', y', z', E)$ = Spatial, Energy Dependent Scalar Adjoint Function for Detector d, from quadrature

$q(x', y', z', E)$ = Spatial, Energy Dependent Source, $\text{n/cm}^3\text{/s}$

$\phi^{+}{}_{d\,g,i}$ = ith Cell Scalar Adjoint Function for Group g, Detector d

$q_{g,i}$ = ith Cell Source Density for Group g, $\text{n/cm}^3\text{/s}$

ΔV_i = ith Cell Volume, cm^3

Therefore, the appropriate adjoint source (used for this problem) is a unit source weighted by the group absorption cross sections for He-3, placed in each He-3 location. The phase space integral of any arbitrary cell source distribution weighted by the computed adjoint function yields the total detector response R. Hence, a single adjoint calculation may be used to predict R from any conceivable source distribution. One should make note of the duality of the response R predicted by theory in Equation (2.27) and stated in Equations (2.28) and (2.29). However, responses computed using forward and adjoint solutions can only be directly comparable if the numerical truncation error from either computation is negligible. In general, responses are typically comparable within some prescribed margin of error as a result of numerical effects.

Numerical Issues. To solve for the adjoint function using the adjoint transport equation, the forward transport equation (2.18) can be used if all angles $\hat{\Omega}$ are taken to be $-\hat{\Omega}$, with angular and energy group indexes transposed (since the transport operator is not self adjoint). Any forward transport algorithm can be used to solve adjoint transport problems if the cross sections and sources are transposed and group re-ordered from group G to 1, with angles implicitly defined in opposite directions (Bell and Glasstone, 1985).

Adjoint and the Importance Function. Therefore, once solved for, the adjoint function provides the neutron or photon importance throughout the problem phase space relative to a particular response, defined by the adjoint source. The adjoint function can then be used in several ways. One use of the adjoint function is to determine the regions/energies that most affect the response to help determine limiting mesh intervals in the geometry. Further, a deterministic adjoint solution may be used to assign importances for variance reduction in non-analog Monte-Carlo applications, which can add extreme efficiency to such calculations (Wagner and Haghighat, 1996). *PENTRAN* can be used to solve for the adjoint function; transposition of all cross sections, etc is performed internally by the code, and the user is reponsible only for properly defining the transposed adjoint source, and noting that groups and directions are reported implicitly reversed.

Angular Quadrature. In the discrete ordinates approximation, it is assumed that the transport equation only holds for a set of *M* distinct angular directions $\hat{\Omega} \rightarrow < \mu, \eta, \xi >$ on the unit sphere. In standard S_N calculations, a numerical quadrature is used to integrate the discrete ordinate angular fluxes to obtain flux moments. In practice, quadrature sets must have the following properties:

$$\sum_{m=1}^{M} w_m = 1.0 \tag{2.30}$$

$$\sum_{m=1}^{M} w_m \mu_m^n = \sum_{m=1}^{M} w_m \eta_m^n = \sum_{m=1}^{M} w_m \xi_m^n = 0 \quad \text{for } n \text{ odd, since} \quad J_{net} = \sum_{m=1}^{M} w_m \Omega_m \psi_m \tag{2.31}$$

$$\sum_{m=1}^{M} w_m \mu_m^n = \sum_{m=1}^{M} w_m \eta_m^n = \sum_{m=1}^{M} w_m \xi_m^n = \frac{1}{n+1} \quad \text{for } n \text{ even} \tag{2.32}$$

Because net current J_{net} in an isotropic flux is *zero*, this requires that equations (2.30) and (2.31) hold true. Equations (2.31) are known as the *odd moment conditions*, and because they must be satisfied, the quadrature set must be symmetric on the unit sphere for each set of directions, invariant with respect to 90-degree axis rotations. Equations (2.32) are known as the *even moment conditions*, required to insure proper integration of the Legendre functions. The Legendre polynomials must be represented due to the expansion in the scattering term, so that:

$$\frac{\delta_{ll'}}{2l+1} = \int_{-1}^{1} \frac{d\mu}{2} P_l(\mu') P_{l'}(\mu') \tag{2.33}$$

and for Associated Legendre Polynomials:

$$\frac{\delta_{ll'}\delta_{kk'}}{2l+1} \frac{(l+k)!}{(l-k)!} = \int_{-1}^{1} \frac{d\mu}{2} P_l^k(\mu') P_{l'}^{k'}(\mu') \tag{2.34}$$

For example, from the first even moment condition, with *n*=2, the quadrature set should satisfy equations (2.33) and (2.34) for the P_1 (first order only) Legendre Polynomials. Since, for any unit angular direction vector $\hat{\Omega}$, the ordinate must lie on the unit sphere:

$$\mu_i^{\,2} + \eta_j^{\,2} + \xi_k^{\,2} = 1 \qquad \text{where} \quad i \in [1, M],\ j \in [1, M],\ k \in [1, M] \tag{2.35}$$

Level Symmetric Quadrature. Due to a required rotational symmetry in three dimensions, the following recursion relationship must hold for level symmetic quadrature in any given octant:

$$\mu_i^{\,2} = \mu_1^2 + C(i-1) \qquad \text{where} \quad C = \frac{2}{N-2}(1 - 3\mu_1^2) \quad \text{and} \quad 2 \le i \le \frac{N}{2} \tag{2.36}$$

The N in any 3-D S_N quadrature corresponds to the number of *levels* from *each* direction cosine on the unit sphere, and there are $M=N(N+2)$ ordinates on the unit sphere, with $M_{oct}=N(N+2)/8$ in each octant, with $N/2$ distinct direction cosine values (Stamm'ler and Abbate, 1983). To derive an S_6 level symmetric quadrature set as an example, using the first octant, six equations with six unknowns must be solved for to provide unique, symmetric direction cosines and corresponding level weights. The following equations must be solved simultaneously using equations (2.30), (2.32), and (2.36) (equations (2.31) are then satisfied implicitly):

Symmetry conditions:

(2.37) $w_1 + w_2 + w_3 = 1$ (weights will be multiplied by 1/8 for all octants, M ordinates)

$\mu_2^{\,2} = \mu_1^2 + C(2-1)$ (with $C = \frac{2}{6-2}(1-3\mu_1^2)$)

$\mu_3^{\,2} = \mu_1^2 + C(3-1)$ (again with $C = \frac{2}{6-2}(1-3\mu_1^2)$)

Even Moment Conditions:

(2.38) $w_1\mu_1^2 + w_2\mu_2^2 + w_3\mu_3^2 = \frac{1}{2+1}$

$w_1\mu_1^4 + w_2\mu_2^4 + w_3\mu_3^4 = \frac{1}{4+1}$

$w_1\mu_1^6 + w_2\mu_2^6 + w_3\mu_3^6 = \frac{1}{6+1}$

Level Symmetric Point Weights. Since the weights w_i in equations (2.37) and (2.38) are *level weights* (note there are 6/2=3 *levels* in an octant for S_6), another set of equations is required to obtain *point weights* $w_{m\,i}$; these can be derived from the ordinate pattern in the first octant (see the Figure at right depicting the point weight pattern within an octant):

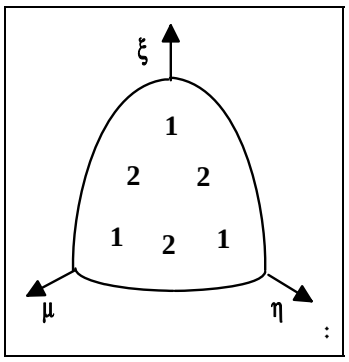


Fig 2.3: Point Weight Pattern for S_6

(2.39) $2w_{m1} + w_{m2} = w_1;\ \ 2w_{m2} = w_2;\ \ \ 1w_{m1} = w_3$

Using Equations (2.37) to (2.39), the quadrature set for S6 can be solved for. Note that from the even moment conditions in equation (2.38), this S6 quadrature set will properly integrate Legendre moments through P3 . All level symmetric quadratures from S2 through S20 were derived for use in *PENTRAN* using equations (2.30), (2.32), and (2.34), along with equations similar to (2.31). No level symmetric quadratures satisfying Equations (2.33) and (2.34) are available beyond S20 due to the appearance of unphysical negative weights. Therefore, some other angular quadrature set must be used to satisfy the moment conditions beyond S20 (in *PENTRAN*, this need is met with the use of the Pn-Tn

Quadratures). To provide for the possibility of increased quadrature order while properly representing the integration of spherical harmonics represented in the scattering source, a provision for arbitrary order Legendre-Chebychev (Pn-Tn) quadrature sets are available in *PENTRAN*.

Legendre-Chebyshev (P_n-T_n) Quadratures. In the P_n-T_n methodology, we set the ξ levels on the z-axis equal to the roots of Legendre polynomials, but for the azimuthal angles on each level we use the roots of the Chebyshev T_N polynomials of first kind (Longoni and Haghighat, 2001) . The Chebyshev polynomials of first kind have the following formulation:

$$T_l[\cos(\omega)] \equiv \cos(l\omega) \tag{2.40}$$

The Chebyshev polynomials are orthogonal and satisfy the following condition:

$$\int_{-1}^{1} dy T_l(y) T_k(y)(1-y^2)^{-1/2} = \begin{cases} 0, l \neq k \\ \pi, l = k = 0 \\ \pi/2, l = k \neq 0 \end{cases}$$

$$y = \cos(\omega) \tag{2.41}$$

Again, using the ordering of the level symmetric quadrature set, we set the azimuthal angles on each level using the following formulation (Carlson and Lathrop, 1964):

$$\omega_{l,i} = \left(\frac{2l-2i+1}{2l}\right)\frac{\pi}{2} \qquad \omega_{l,i} \in \left(0, \frac{\pi}{2}\right), \; i = 1 \ldots l \tag{2.42}$$

In Eq. (2.42), "*l*" is the level number. Unlike the level symmetric quadratures, the (P_n-T_n) sets do not present negative weights for S_N orders higher than 20.

Demonstration of Even Moment Condition Preservation. Again, the main advantage of the level symmetric sets is that they preserve the even and odd moments of the direction cosines, preserving orthogonality and leading to an accurate solution. In this section, we compare the new (P_n-T_n) quadrature sets based on the magnitude of the even moments of the direction cosines.

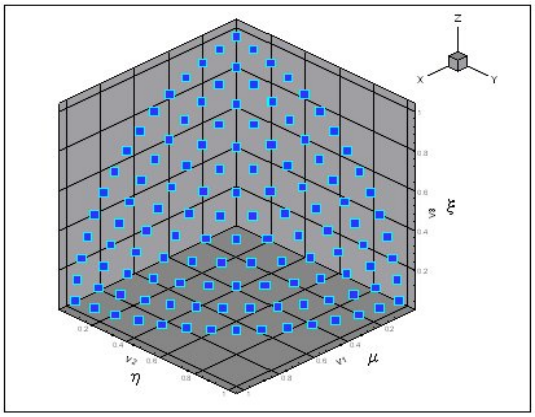


Fig. 2.4 Discrete directions selected on one octant with S_{30} P_n-T_n quadrature set

Table 2.1 Even-Moments for P_n-T_n S_{30} set

Moment Order (n even)	$\sum_{i=1}^{M} w_i \mu_i^n$	$\sum_{i=1}^{M} w_i \eta_i^n$	$\sum_{i=1}^{M} w_i \xi_i^n$	$\frac{1}{1+n}$
2	0.333333333	0.333333333	0.333333333	0.333333333
4	0.199999962	0.199999962	0.2	0.2
6	0.142857143	0.142857143	0.142857143	0.142857143
8	0.111111111	0.111111111	0.111111111	0.111111111
10	0.090909091	0.090909091	0.090909091	0.090909091
12	0.076923077	0.076923077	0.076923077	0.076923077
14	0.066666667	0.066666667	0.066666667	0.066666667
16	0.058823529	0.058823529	0.058823529	0.058823529
18	0.052631579	0.052631579	0.052631579	0.052631579
20	0.047619048	0.047619048	0.047619048	0.047619048
22	0.043478261	0.043478261	0.043478261	0.043478261
24	0.043478261	0.043478261	0.043478261	0.043478261
26	0.037037037	0.037037037	0.037037037	0.037037037
28	0.034482759	0.034482759	0.034482759	0.034482759
30	0.032258065	0.032258065	0.032258065	0.032258065

Ordinate Splitting Method. The Ordinate Splitting (OS) Technique is developed for solving problems with highly peaked angular flux and/or source. The idea is to introduce more directions at local regions; for this purpose we split a direction into a number of directions with equal weights. These directions are positioned symmetrically around the direction of interest and their weights are calculated by equally dividing the original weight among the new split directions and the direction of interest, as depicted in Fig. 2.5a. This technique because of its local refinement can be considered as a biasing approach. In *PENTRAN*, the solid angle associated with a specific ordinate is sub-divided into equal size sectors. The added directions are placed on the corners and flanking the original direction is at the center as shown in Fig. 2.5b. A proper selection of the sector surrounding the original direction is performed to avoid overlapping with other directions. The number of additional directions is chosen with a parameter called segmentation, where #directions=$(2*nseg -1)^2$

Fig. 2.5b shows a S_{16} P_n-T_n quadrature set with three split angles; for this case we have set nseg = 2 (Longoni and Haghight, 2001).

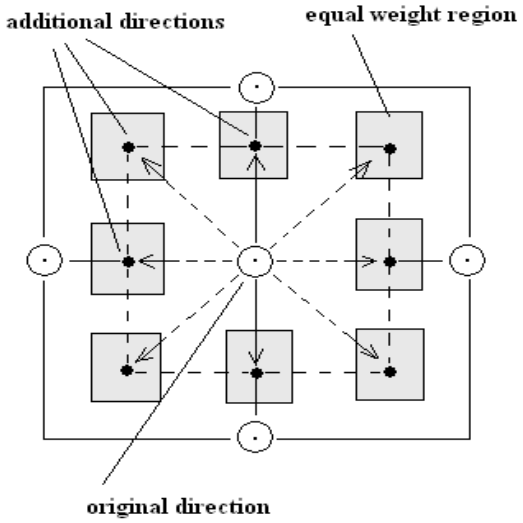


Fig. 2.5a Ordinate Splitting Technique

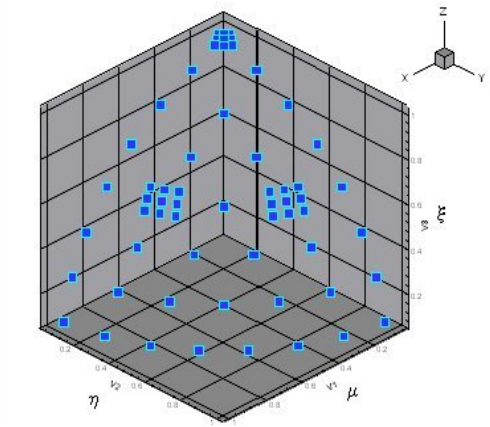


Fig. 2.5b S_{16} P_n-T_n quadrature set modified with the Ordinate Splitting Technique

At this point, a spatial approximation to equation (2.18) is required. As mentioned earlier, several approaches can be made to formulate a discrete ordinates spatial differencing scheme. Zeroth *spatial* Legendre Functions are:

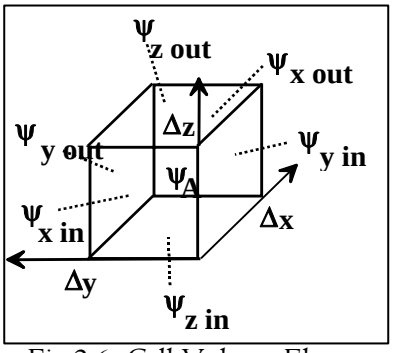


Fig 2.6 Cell Volume Element

$$P_0(x) = 1 \qquad P_0(y) = 1 \qquad P_0(z) = 1 \tag{2.43}$$

To derive the zeroth spatial moment balance equation, equation (2.18), multiplied by equations (2.43), is integrated over a local cell volume and divided by the integral of the product of equations (2.43), also integrated over the cell volume. For our purposes, a cell volume has parallelepiped dimensions $(\Delta x, \Delta y, \Delta z)$. Assuming that particles are traveling in a positive direction, edge and center flux integrals are represented by integral averages. If we consider that directions traveled could be negative, then considering $|\mu|,|\eta|,|\xi|$ for a positive sense in the equations, which always occurs if the equations are derived *in the direction of particle motion*. The *zeroth spatial moment balance equation* (omitting the group *g* subscript for brevity) is:

$$\frac{|\mu_m|}{\Delta x}(\psi_{\text{out}\,x} - \psi_{\text{in}\,x}) + \frac{|\eta_m|}{\Delta y}(\psi_{\text{out}\,y} - \psi_{\text{in}\,y}) + \frac{|\xi_m|}{\Delta z}(\psi_{\text{out}\,z} - \psi_{\text{in}\,z}) + \sigma\psi_A = q_A \tag{2.44}$$

For a positive angular vector, where $(\mu,\eta,\xi) > 0$ with the entering and exiting surface averaged angular fluxes normal to the *x*-axis are given by:

$$\psi_{\text{in}\,x} = \frac{1}{\Delta y \Delta z}\int_0^{\Delta y}\int_0^{\Delta z} \psi_m(0,y,z)\, P_0(y)\, P_0(z)\, dy\, dz$$

$$\psi_{\text{out}\,x} = \frac{1}{\Delta y \Delta z}\int_0^{\Delta y}\int_0^{\Delta z} \psi_m(\Delta x,y,z)\, P_0(y)\, P_0(z)\, dy\, dz \tag{2.45}$$

Surface terms normal to the *y*- and *z*- dimensions are defined in a similar manner. The cell volume-averaged angular flux is given by:

$$\psi_A = \frac{1}{\Delta x \Delta y \Delta z}\int_0^{\Delta x}\int_0^{\Delta y}\int_0^{\Delta z} \psi_m(x,y,z)\, P_0(x)\, P_0(y)\, P_0(z)\, dx\, dy\, dz \tag{2.46}$$

The volume-averaged source term q_A is defined in a similar manner. Note that we refer to the surface averaged terms that enter and leave the cell as the "in" and "out" subscripts, respectively, while the "*A*" subscripts denote cell average quantities.

Seven Unknowns. Equation (2.44) is exact, but contains seven unknowns. We can consider the three entrant values ("in") are known from boundary values, and that the collective cell averaged volumetric source q_A is assumed to be known from a previous source iteration (in the standard S_N source iteration scheme). Therefore, only the cell average angular flux ψ_A and the exiting ("out") surface values are unknowns, where these latter values are obtained using a set of *auxiliary equations*. Auxiliary equations amount to "fitting functions" that resemble the behavior of the angular flux across the spatial cell, and they establish the accuracy of the differencing method (Lewis and Miller, 1993).

Weighted Schemes. For weighted spatial differencing schemes, the following auxiliary equations are assumed to hold between cell average and boundary angular fluxes:

$$\psi_{\text{out } x} = 1/a(\psi_A + \psi_{\text{in } x}(a-1))$$

$$\psi_{\text{out } y} = 1/b(\psi_A + \psi_{\text{in } y}(b-1))$$

$$\psi_{\text{out } z} = 1/c(\psi_A + \psi_{\text{in } z}(c-1))$$

(2.47)

DD and TW Schemes. Note that the standard Diamond Differencing (DD) scheme results when a=½, b=½, and c=½ in equations (2.47); the DD scheme is second order accurate, but may lead to negative solutions. (Lewis and Miller, 1993). In such situations, a "negative flux set to zero fixup" of the Diamond scheme is commonly used. In this paper, we denote the Diamond scheme with the zero fixup as Diamond-Zero (DZ). Furthermore, it is worth noting that the negative flux fixup has also demonstrated to be the source of load imbalance in parallel processing solutions (Haghighat, Hunter, and Mattis, 1995). To overcome the inherent difficulties of the Diamond scheme, Rhoades and Engle (1977) developed the Theta-Weighted (TW) scheme that is always positive.

Oscillatory Behavior. Recently, Petrovic and Haghighat (1996) have shown that in multidimensional geometries, non-physical oscillations occur because of the "mismatch" between the direction of particles (along a characteristic) and the spatial axis where the differencing is carried out, even with very high mesh refinement. The oscillations are attributed to a "forced" relationship of the average angular flux to the boundary fluxes (depending on the auxiliary equation), where no directionally dependent boundary contribution relative to each axis is taken into account (Petrovic and Haghighat, 1996). The non-physical oscillations inherent in solutions rendered by the Diamond scheme only add to the previously mentioned difficulties with positivity, and can be quite detrimental to convergence, especially in parallel execution.

DTW Scheme. To remedy this, Petrovic and Haghighat developed the Directional Theta-Weighted scheme (Petrovic and Haghighat, 1996) that is an extension of TW. To derive a Cartesian form of the TW scheme for weight factor *a* (for $\psi_{\text{out}\,x}$), equations (2.47) are placed into equation (2.44). Solving for ψ_A, and again using equations (2.47), an expression for $\psi_{\text{out}\,x}$ is obtained. The Diamond relations are then assumed to hold for the *y*- and *z*-directions (with *b*=½ and *c*=½).

To force positivity, arbitrary fixed theta-weighting parameters $(\theta(\mu_m))$ are introduced into the formulation; $|\mu_m|/(a\Delta x)$ is dropped from the denominator, and *a* is assimilated into the parameters. Assuming the lower bound of $\psi_{\text{out}\,x}$ is strictly zero, we obtain an equation for the "*a*" weight:

$$a = 1 - \frac{q_A + \frac{|\mu_m|}{\Delta x}\psi_{\text{in}\,x} + \theta\left(\frac{|\eta_m|}{\Delta y}\psi_{\text{in}\,y} + \frac{|\xi_m|}{\Delta z}\psi_{\text{in}\,z}\right)}{\left(2\frac{|\eta_m|}{\Delta y} + 2\frac{|\xi_m|}{\Delta z} + \sigma\right)\psi_{\text{in}\,x}} \tag{2.48}$$

Using a similar procedure along the *y*- and *z*-axes to yield weights for "*b*" and "*c*," respectively:

$$b = 1 - \frac{q_A + \frac{|\eta_m|}{\Delta y}\psi_{\text{in}\,y} + \theta\left(\frac{|\mu_m|}{\Delta x}\psi_{\text{in}\,x} + \frac{|\xi_m|}{\Delta z}\psi_{\text{in}\,z}\right)}{\left(2\frac{|\mu_m|}{\Delta x} + 2\frac{|\xi_m|}{\Delta z} + \sigma\right)\psi_{\text{in}\,y}} \tag{2.49}$$

$$c = 1 - \frac{q_A + \frac{|\xi_m|}{\Delta z}\psi_{\text{in}\,z} + \theta\left(\frac{|\mu_m|}{\Delta x}\psi_{\text{in}\,x} + \frac{|\eta_m|}{\Delta y}\psi_{\text{in}\,y}\right)}{\left(2\frac{|\mu_m|}{\Delta x}\psi_{\text{in}\,x} + 2\frac{|\eta_m|}{\Delta y} + \sigma\right)\psi_{\text{in}\,z}} \tag{2.50}$$

Petrovic and Haghighat specified that for maximum smoothing in directions perpendicular to respective characteristics (to minimize oscillations), the fixed theta parameters in the TW scheme could be modified to allow variable theta parameters that are dependent on the characteristic of the incident radiation. As a result, the Directional Theta Weighted (DTW) scheme employs the 3-D theta parameters given in equation (2.43) substituted for each θ parameter in equations (2.48), (2.49), and (2.50).

$$\theta(\mu_m) = \mu_m^2 \qquad \theta(\eta_m) = \eta_m^2 \qquad \theta(\xi_m) = \xi_m^2 \tag{2.51}$$

where μ, η, ξ are the direction cosines along the *x*-, *y*-, and *z*- axes, respectively.

The angular flux weighting factors are subsequently used to obtain the DTW average cell angular flux, given by:

$$\psi_A = \frac{q_A + \frac{|\mu_m|}{a\Delta x}\psi_{\text{in}\,x} + \frac{|\eta_m|}{b\Delta y}\psi_{\text{in}\,y} + \frac{|\xi_m|}{c\Delta z}\psi_{\text{in}\,z}}{\left(\frac{|\mu_m|}{a\Delta x} + \frac{|\eta_m|}{b\Delta y} + \frac{|\xi_m|}{c\Delta z} + \sigma\right)} \tag{2.52}$$

Therefore, using equations (2.51) in equations (2.48), (2.49), and (2.50), respectively, the DTW scheme uses direction-based parameters to obtain angular flux weighting factors, from which average and inherently positive exiting angular fluxes are derived using equation (2.47). The DTW scheme is clearly non-linear in the way the angular flux weights (*a,b,c*) are derived from directionally dependent parameters, incident fluxes, and volumetric sources. To be consistent, these weights are restricted to the range between ½ and 1, with accuracy approaching second order truncation when all weights are ½ (equivalent to the Diamond scheme). This truncation error is evident if equations (2.47) are substituted into (2.44). If that result is subtracted from equation (2.44), and Taylor's series expansions are applied about ψ_A, the trucation error of the DTW formulation is:

$$\begin{aligned} \in_{DTW} &= (1 - \frac{1}{2a})\frac{\Delta x}{\rho_x}\frac{\partial \psi}{\partial x}|_A + \frac{\Delta x^2}{8a\rho_x}\frac{\partial^2 \psi}{\partial x^2}|_A + \\ &(1 - \frac{1}{2b})\frac{\Delta y}{\rho_y}\frac{\partial \psi}{\partial y}|_A + \frac{\Delta y^2}{8b\rho_y}\frac{\partial^2 \psi}{\partial y^2}|_A + (1 - \frac{1}{2c})\frac{\Delta z}{\rho_z}\frac{\partial \psi}{\partial z}|_A + \frac{\Delta z^2}{8c\rho_z}\frac{\partial^2 \psi}{\partial z^2}|_A + O(\Delta^3) \end{aligned} \tag{2.53}$$

$$\text{where} \quad \rho_u = \frac{\sigma \Delta u}{|\tau_m|} \quad \text{for} \quad u \in \{x, y, z\} \text{ and } \tau_m \in \{\mu_m, \eta_m, \xi_m\}$$

A *positive* truncation error $\in_{DTW}$ indicates that DTW will *underestimate* the solution, while a *negative* truncation error indicates DTW will *overestimate* the solution. When DTW weights are all ½, the truncation error is identical to that of the Diamond scheme, influenced only by second partial derivatives. Note that when DTW weights are greater than ½, the truncation error is influenced by *both* first *and* second partial derivatives of the angular flux (Sjoden, 1997). Because of the directional weighting of DTW, a set of angular fluxes along different paths will contain both over- *and* underestimated angular fluxes. In addition to being positive and free of oscillations from the directional weighting, the DTW scheme can be significantly more accurate than the Diamond scheme, mainly because we typically are interested only in the *scalar flux* (integrated over all directions). The *combined* effects of over- and underestimates of the angular flux among different directions with DTW often cancel during integration (quadrature), resulting in more accurate *scalar fluxes*. In the special case where the flux is relatively flat (such as in the middle of a reactor core), DTW weights will be near unity, and the truncation error will be very small due to small flux gradients.

EDW Scheme. While DTW may not be a highly accurate scheme in all situations, it behaves reliably in general situations (positivity, stability, with derivatives having proper signs, etc). Therefore, a *predictor- corrector* exponential scheme that uses DTW to *predict* a solution that is then *corrected* by an exponential fit should be stable and more accurate than DTW alone. Using this approach, the following inherently positive (provided the coefficient a_o is positive) exponential auxiliary equation is proposed:

$$\psi_m(x,y,z) = a_o \, exp(\lambda_i \, P_1(x)/|\mu_m|) \, exp(\lambda_j \, P_1(y)/|\eta_m|) \, exp(\lambda_k \, P_1(z)/|\xi_m|) \tag{2.54}$$

First order spatial Legendre functions, orthogonal to equations (2.35) over the widths of a single cell, are:

$$P_1(u) = \frac{2u}{\Delta u} - 1 \qquad \text{where } 0 \le u \le \Delta u \text{ and } u \in \{x,y,z\} \tag{2.55}$$

The exponential coefficients (λ) define the overall profile of equation (2.54); these coefficients are normally obtained by root solving 1st moment transcendental conservation equations (Mathews, Sjoden, and Minor, 1994), (Walters, Wareing, and D. Marr, 1995), and (Wareing and Alcouffe, 1995). To avoid the computational overhead of explicitly conserving the 1st moment balance equations, Sjoden and Haghighat proposed using a DTW solution to provide good estimates of these coefficients in a *predictor* step in the following manner. Consider a single cell of dimensions $(\Delta x, \Delta y, \Delta z)$. By taking first partial derivatives of equation (2.54) with respect to *x*-, *y*-, and *z*- axes, and assuming that in the limit as cell dimensions approach zero, $\psi(x,y,z) \to \psi_A$, then λ_i, λ_j, and λ_k can be separated as follows:

$$\frac{1}{\psi_A}\frac{\partial \psi}{\partial x}|_A = \frac{2\lambda_i}{\Delta x|\mu_m|} \qquad \frac{1}{\psi_A}\frac{\partial \psi}{\partial y}|_A = \frac{2\lambda_j}{\Delta y|\eta_m|} \qquad \frac{1}{\psi_A}\frac{\partial \psi}{\partial z}|_A = \frac{2\lambda_k}{\Delta z|\xi_m|} \tag{2.56}$$

Then, the first partial derivatives in equation (2.56) can be approximated, again using "out" and "in" cell surface references, using a standard finite difference formulation, where $u \in \{x,y,z\}$:

$$\frac{\partial \psi}{\partial u}|_A = \frac{(\psi_{\text{out}\,u} - \psi_{\text{in}\,u})}{\Delta u} - \frac{\Delta u^2}{24}\frac{\partial^3 \psi}{\partial u^3}|_A + O(\Delta u^3) \tag{2.57}$$

Then, using the *predicted* angular fluxes initially calculated using DTW (where DTW predicted angular fluxes are denoted by $\tilde{\psi}$) we can obtain explicit estimates (after algebraic simplification) for λ_i, λ_j, and λ_k. For the *x*-, *y*-, and *z*- dimensions, respectively:

$$\lambda_i \approx \frac{(\tilde{\psi}_{\text{out}\,x} - \tilde{\psi}_{\text{in}\,x})|\mu_m|}{2\tilde{\psi}_A} \qquad \lambda_j \approx \frac{(\tilde{\psi}_{\text{out}\,y} - \tilde{\psi}_{\text{in}\,y})|\eta_m|}{2\tilde{\psi}_A} \qquad \lambda_k \approx \frac{(\tilde{\psi}_{\text{out}\,z} - \tilde{\psi}_{\text{in}\,z})|\xi_m|}{2\tilde{\psi}_A} \tag{2.58}$$

Note that use of equation (2.57) inherently assumes that the average of the first partial derivative of the angular flux is assumed to be at the cell center. However, note further that this equation only needs to estimate the derivative of the angular flux as opposed to the angular flux itself. Furthermore, the formulations cast in equation (2.58) demonstrate that $\lambda_i, \lambda_j,$ and λ_k are based on a dimensionless ratio of the DTW predicted angular fluxes, thus reducing the sensitivity of computing a precise derivative. Since the DTW scheme provides estimates for the exponential constants, we must solve for the coefficient a_o in equation (2.54) using the zeroth moment balance defined in equation (2.44). To do this, we first perform the integrations for the *outbound* surface and cell volume angular fluxes as in equations (2.45) and (2.46) defined using the exponential auxiliary equation (2.54). (Note that *inbound* surface averaged fluxes and the cell volumetric source terms are assumed to be known). Placing the resulting formulations into the balance equation (2.44), we can obtain a solution (albeit rather cumbersome) for a_o; substituting the resulting expression for a_o back into the formulations for the outbound surface and volume averaged angular fluxes, we can obtain ψ_A (after more algebraic simplification):

$$\begin{aligned}\psi_A = &\left(\exp(\frac{2\lambda_i}{|\mu_m|})-1\right)\left(\exp(\frac{2\lambda_j}{|\eta_m|})-1\right)\left(\exp(\frac{2\lambda_k}{|\xi_m|})-1\right)\\ &\cdot\frac{1}{\beta}\left(q_A+\frac{|\mu_m|}{\Delta x}\psi_{\text{in}\,x}+\frac{|\eta_m|}{\Delta y}\psi_{\text{in}\,y}+\frac{|\xi_m|}{\Delta z}\psi_{\text{in}\,z}\right)\end{aligned} \tag{2.59}$$

where β in equation (2.59) is defined by:

$$\begin{aligned}\beta = &\frac{2\lambda_i}{\Delta x}\left(\exp(\frac{2\lambda_i}{|\mu_m|})\right)\left(\exp(\frac{2\lambda_j}{|\eta_m|})-1\right)\left(\exp(\frac{2\lambda_k}{|\xi_m|})-1\right)+\\ &\frac{2\lambda_j}{\Delta y}\left(\exp(\frac{2\lambda_j}{|\eta_m|})\right)\left(\exp(\frac{2\lambda_i}{|\mu_m|})-1\right)\left(\exp(\frac{2\lambda_k}{|\xi_m|})-1\right)+\\ &\frac{2\lambda_k}{\Delta z}\left(\exp(\frac{2\lambda_k}{|\xi_m|})\right)\left(\exp(\frac{2\lambda_i}{|\mu_m|})-1\right)\left(\exp(\frac{2\lambda_j}{|\eta_m|})-1\right)+\\ &\sigma\left(\exp(\frac{2\lambda_i}{|\mu_m|})-1\right)\left(\exp(\frac{2\lambda_j}{|\eta_m|})-1\right)\left(\exp(\frac{2\lambda_k}{|\xi_m|})-1\right)\end{aligned} \tag{2.60}$$

The outbound cell fluxes can be defined in terms of the cell average angular flux:

$$\begin{aligned}\psi_{\text{out}\,x} &= \psi_A\frac{2\lambda_i}{|\mu_m|}\left(1-\exp(\frac{-2\lambda_i}{|\mu_m|})\right)^{-1}\\ \psi_{\text{out}\,y} &= \psi_A\frac{2\lambda_j}{|\eta_m|}\left(1-\exp(\frac{-2\lambda_j}{|\eta_m|})\right)^{-1}\\ \psi_{\text{out}\,z} &= \psi_A\frac{2\lambda_k}{|\xi_m|}\left(1-\exp(\frac{-2\lambda_k}{|\xi_m|})\right)^{-1}\end{aligned} \tag{2.61}$$

Equations (2.58) through (2.61) therefore provide a correction to the initial DTW predicted angular fluxes using exponential functions based on the auxiliary equation (2.54). This methodology allows the exponential coefficients (λ) to be predicted at very low cost using the DTW scheme (Sjoden and Haghighat, 1997). This is the Exponential Directional Weighted (EDW) method; it is absolutely positive, stable, directionally weighted, and is significantly more accurate than the DTW scheme in streaming problems with relaxed cell intervals. Demonstrations as to the effectiveness of EDW in streaming problems can be found in the literature.

To obtain an expression for truncation error of the EDW scheme, we again make the assumption that in the limit of small cells, the DTW and EDW solutions are identical. Expanding the exponential arguments using a Taylor's Series truncated to third order, substituting into equation (2.44), and then subtracting that result from equation (2.44) yields an expression for the truncation error of the EDW scheme (again after additional algebraic simplification):

$$\in_{EDW} = \frac{\Delta x}{2\rho_x}\frac{\partial \psi}{\partial x}|_A + \frac{\Delta x^2}{8\rho_x}\frac{\partial^2 \psi}{\partial x^2}|_A + \tag{2.62}$$

$$\frac{\psi_A}{\rho_x}\left(1 - \frac{\psi_A}{\psi_A + (-\Delta x/2)\ \partial\psi/\partial x|_A + (\Delta x^2/(6\psi_A))\ (\partial\psi/\partial x|_A)^2}\right) + \frac{\Delta y}{2\rho_y}\frac{\partial \psi}{\partial y}|_A + \frac{\Delta y^2}{8\rho_y}\frac{\partial^2 \psi}{\partial y^2}|_A +$$

$$\frac{\psi_A}{\rho_y}\left(1 - \frac{\psi_A}{\psi_A + (-\Delta y/2)\ \partial\psi/\partial y|_A + (\Delta y^2/(6\psi_A))\ (\partial\psi/\partial y|_A)^2}\right) + \frac{\Delta z}{2\rho_z}\frac{\partial \psi}{\partial z}|_A + \frac{\Delta z^2}{8\rho_z}\frac{\partial^2 \psi}{\partial z^2}|_A +$$

$$\frac{\psi_A}{\rho_z}\left(1 - \frac{\psi_A}{\psi_A + (-\Delta z/2)\ \partial\psi/\partial z|_A + (\Delta z^2/(6\psi_A))\ (\partial\psi/\partial z|_A)^2}\right) + O(\Delta^3)$$

From equation (2.62), note that the truncation error (and therefore the accuracy) of the EDW scheme is dependent on both the slope and concavity (for each partial derivative) of the angular flux, with a strong influence from first partial derivatives. Therefore, the EDW equations result in a *correction* to the DTW *predicted* angular fluxes using exponential functions based on the auxiliary Equation (2.54). In this way, the exponential constants are predicted at low cost using the DTW scheme relative to first moment methods [Sjoden and Haghighat, 1997].

EDI Scheme. The EDI scheme is also a predictor-corrector scheme with the exponential basis used in EDW, however, EDI includes *iterative refinement* of the exponential constants (that are held fixed in EDW) to yield increased accuracy. EDI is derived via a method analogous to the EDW scheme up to Equation (2.56). Then, recognizing that Equation (2.56) is an exact expression, we integrate both sides over the cell volume, avoiding the application of any limiting assumptions (as in Equation (2.57)), where both sides of

Equations (2.56) are integrated over the cell dimensions $(\Delta x, \Delta y, \Delta z)$ divided by cross sectional area:

(2.62a)

$$\iiint \left(\frac{\partial \psi}{\partial x} = \frac{2\lambda_i}{\Delta x \,|\, \mu_m \,|}\ \psi(\mathrm{x,y,z})\right) dx\, dy\, dz \qquad \rightarrow \qquad (\psi_{out\,\mathrm{x}} - \psi_{in\,\mathrm{x}}) = \frac{2\lambda_i}{|\,\mu_m\,|}\ \psi_A$$

(2.62b)

$$\iiint \left(\frac{\partial \psi}{\partial y} = \frac{2\lambda_j}{\Delta y \,|\, \eta_m \,|}\ \psi(\mathrm{x,y,z})\right) dx\, dy\, dz \qquad \rightarrow \qquad (\psi_{out\,\mathrm{y}} - \psi_{in\,\mathrm{y}}) = \frac{2\lambda_j}{|\,\eta_m\,|}\ \psi_A$$

(2.62c)

$$\iiint \left(\frac{\partial \psi}{\partial z} = \frac{2\lambda_k}{\Delta z \,|\, \xi_m \,|}\ \psi(\mathrm{x,y,z})\right) dx\, dy\, dz \qquad \rightarrow \qquad (\psi_{out\,\mathrm{z}} - \psi_{in\,\mathrm{z}}) = \frac{2\lambda_k}{|\,\xi_m\,|}\ \psi_A$$

Therefore, preserving surface and cell averaged variables (as in Equation (2.45)) yields a formulation for each exponential constant that is albeit identical to that presented in Equations (2.58). This establishes that Equations (2.58) indeed hold for each exponential constant, but are derived as *exact analytic expressions* without *a priori assumptions* regarding the *location* of the average angular flux over the cell or truncation error, etc. Most importantly, this directly establishes that a fixed point iteration can be performed to refine each exponential constant $\lambda_i, \lambda_j, \lambda_k$ by *successive iteration (I-1, I, I+1…)* of Equations (2.63) given here:

(2.63a)
$$\lambda_{i,I} = f(\lambda_{i,I-1}) = \frac{(\psi_{out\,\mathrm{x}}(\lambda_{i,I-1}) - \psi_{in\,\mathrm{x}}(\lambda_{i,I-1}))\,|\,\mu_m\,|}{2\psi_A(\lambda_{i,I-1})}$$

(2.63b)
$$\lambda_{j,I} = g(\lambda_{j,I-1}) = \frac{(\psi_{out\,\mathrm{y}}(\lambda_{j,I-1}) - \psi_{in\,\mathrm{y}}(\lambda_{j,I-1}))\,|\,\eta_m\,|}{2\psi_A(\lambda_{j,I-1})}$$

(2.63c)
$$\lambda_{k,I} = h(\lambda_{k,I-1}) = \frac{(\psi_{out\,\mathrm{z}}(\lambda_{k,I-1}) - \psi_{in\,\mathrm{z}}(\lambda_{k,I-1}))\,|\,\xi_m\,|}{2\psi_A(\lambda_{k,I-1})}$$

For any unique fixed point iteration, and especially so in multivariate implementations in a general algorithm, it can be shown that the fixed point iteration will remain convergent on a finite, nonzero interval [*p,q*] by adhering to the first derivative criterion in Equations (2.63) [Sjoden, 2007]:

(2.64)
$$\left|\frac{\partial f\,(\lambda_{i,I-1})}{\partial \lambda_{i,I-1}}\right| < 1 \qquad \left|\frac{\partial g\,(\lambda_{j,I-1})}{\partial \lambda_{j,I-1}}\right| < 1 \qquad \left|\frac{\partial h\,(\lambda_{k,I-1})}{\partial \lambda_{k,I-1}}\right| < 1$$

In summary, an approximate Sn solution is assumed along each direction, *m*, within a 3-D rectangular cell, of the form of Equation (2.54), where the coefficient and exponential constants are to be determined by both boundary conditions and the requirement of particle conservation over the cell. As a result, note that there are four degrees of freedom, requiring

four equations. Three equations are obtained directly from boundary conditions. Specifically, there are three inflow cell faces, and the average flux on each such face is defined by boundary conditions. Thus, because the EDI scheme is continuous rather than discontinuous, the approximate solution must satisfy the correct average flux value on each incoming face, so that the fourth equation relates to particle conservation. Specifically, if we substitute the approximate solution into the transport equation and integrate over the cell, the resulting balance equation must be satisfied by the solution. We also note that the EDI and EDW schemes are similar in their basis; they differ in that the EDW scheme uses the values from Equations (2.58) as one-time values, whereas the EDI scheme uses them as starting values to begin a fixed point iteration of the exponential constants for continued iteration to convergence, as given in Equations (2.63). Tests for divergence are performed, and if any fixed point iteration should violate Equations (2.64), iterations cease with a default to EDW values. Therefore, the EDI scheme is engaged initially using Equations (2.58) to yield an initial starting guess (from DTW), with successive applications of Equations (2.63) applied in accordance with Equations (2.64) to yield a stable fixed-point iteration to solve for more accurate values of each exponential constant $\lambda_i, \lambda_j, \lambda_k$, yielding accurate constants. An additional improvement recently discussed involves an Aitken extrapolation of the third and subsequent iteraties of the exponential constants, which improves convergence and accuracy (Yi and Sjoden, 2008).

Overall, the EDI scheme is implemented in a method nearly identical to that of EDW, using the same equations as the EDW scheme, save for the fixed point iterations on the exponential constants. To summarize this, cell sources and cell incident fluxes are assumed to be known. The DTW scheme is used to provide the starting value for the exponential constants $\lambda_i, \lambda_j, \lambda_k$; note that if the incident fluxes are zero, as at a vacuum boundary, no further action is taken beyond the DTW step. For the vacuum boundary cell case, the solution is based only on the DTW scheme, since the continuous exponential EDI scheme is not applicable in that situation. Note the EDW and EDI formulations are strictly positive, and that some exponential arguments can be re-used among formulations to minimize floating point operations. Exponential constants $\lambda_i, \lambda_j, \lambda_k$ are then recomputed and evaluated for stability according to Equations (2.63) and (2.64), with a use of EDW for unstable cases. If stable, Equations (2.59) to (2.61) are re-applied, etc, and if average angular fluxes are converged, it is assumed the EDI iteration is complete. The exact performance and efficiency relative to EDI converegnce is quite problem dependent, although there are numerical checks for numerical consistency in the scheme. If the upper limit of iterations per ordinate in EDI is achieved, application of the EDW scheme prevents oscillation in the solution and prevents instability, although this can restrict the numerical accuracy benefit derived from the iterative refinement of the exponential constants. As a practical issue, the average number of fixed point iterations over all sweeps and iterations is reported as a performance metric among each coarse mesh zone for the EDI scheme in *PENTRAN*.

Schemes in *PENTRAN*. The DD, DZ, DTW, EDI, EDW, and EDH differencing schemes are fully implemented into *PENTRAN*. These schemes are selected for each coarse mesh, and can be controlled by the code in an automated "Adpative Differencing Strategy." Metrics reported for each coarse mesh and scheme are noted (/sweep = "per angular flux sweep"):

Table 2.2 Differencing scheme Options with Adaptive Strategy and Upgrade Criteria

Diff No	Method Acronym-Description	Avg Metric Description	Upgrade Criteria	Method Lock-in
0	DD = Linear Diamond/No Fixup	Not Used	None	0
1	DZ = Linear Diamond/Zero Fixup	Fixups/Sweep	Fixup	-1
2	DTW = Directional Theta Weighted*	MaxWgt/Sweep	W=0.9500	-2
3	EDI = Exponential-Directional Iterative	Iterations/Sweep	$\sigma\Delta h_{\max}$ =0.02	-3
4	EDW = Expon-Direct Weighted	DTWuse/Sweep	None	-4
5	EDH = Exp-Direct. Omega Hybrid	DTWuse/Sweep	None	-5

*Upgrade possible when Source density/Collision density default ratio *qfratio* < 1.00

Adaptive Differencing Strategy. The adaptive differencing strategy (the mechanics of which are described in detail below) enables the code to automatically select the best possible scheme for the energy group being solved, and the "upgrade criteria" to determine which scheme is selected, has been refined over recent years to yield optimum solution accuracy. The differencing metrics provide the user with useful information about the relative accuracy of the differencing in fine meshes contained by each coarse mesh. *PENTRAN* allows for the user to take advantage of an adaptive differencing capability where the code selects from among the DZ, DTW, or EDI schemes to remove most of the difficulty in determining the appropriate scheme used on fine meshes within a particular coarse mesh and energy group. At the start of each new in-group sweep, *PENTRAN* resets the scheme used for every fine mesh in each coarse mesh to the originally prescribed scheme, and the schemes adapt to the most appropriate scheme, on a coarse mesh basis.

Standard Adaptive Differencing Strategy Mechanics. The *adaptive differencing* strategy in *PENTRAN* works in the following manner: assume (for illustration) that the DZ scheme is initially assigned (but not locked by using a '-1') in each coarse mesh. An automatic differencing scheme transfer from DZ to DTW takes place if a negative flux fixup is encountered anywhere in a coarse mesh. This is followed later by another transfer from DTW to EDI if any maximum linear weight factor (as given in Eqs (2.48), (2.49), or (2.50)) beyond a user specified maximum weight factor (*recommended as a default of 0.95, but user adjustable in the first entry of the "dtwmxw" vector parameter*) is detected for DTW within a coarse mesh. This is performed since a high weight factor indicates DTW is being pressed to maintain positivity in a severe streaming environment, so that the shift to EDI enables an exponential treartment for cells that are optically thick, since these scenarios are best handled using the exponential basis of the EDI scheme.

Automatic Exceptions to the Standard Adaptive Strategy. The following are exceptions to the standard adaptive strategy just presented:

- $\sigma\Delta h_{\max}$.
  - Considering that $\Delta h_{\max} = \max(\Delta x, \Delta y, \Delta z)$, this value is computed to evaluate the optical cell thickness, if $\sigma\Delta h_{\max}$ is less than a user specified value (0.02 is the default), then for these vanishingly thin cells, the DTW scheme is not upgraded to the EDI scheme, since a very small optical thickness is such that DTW is perfectly adequate to resolve the angular flux accurately. This value is set in the second entry of the "dtwmxw" vector parameter in the input deck.
- *qfratio*.
  - Since the logic to upgrade a differencing scheme from DTW to EDI is based on any DTW weight factor exceeding a value near unity (recommended default of 0.95, based on significant testing), where as mentioned, in a streaming situation with no or low-level sources, this is required to maintain positivity at the expense of accuracy. However, using this metric to determine the scheme upgrade criteria, a shift from the DTW to EDI schemes also occurs in any mesh cell that contains a strong source simply because the angular flux in these situations is often relatively *flat* (resulting in a low angular flux gradient). Therefore, with a strong source present, this leads to DTW weight factors close to unity, and causes a conflict with the upgrade criteria just presented, which is undesirable, since the DTW scheme performs very well in regions where there is a strong source—DTW weight factors are close to unity because the flux is inherently flat.
  - Therefore, note that if the angular flux is inherently flat due to the presence of a strong source, a step scheme would be very effective—the "step" scheme results algebraically if the weights are set to unity along each direction for DTW. This scenario has been mitigated in *PENTRAN* through the use of the *qfratio*.
  - Considering the group dependent transport equation divided through by the collision density term, with group scatter, fission, and independent sources:

$$\frac{\hat{\Omega}\cdot\nabla\psi_g}{\sigma_g\psi_g}+1=\frac{(q_{sg}+q_{fg}+q_{ind\,g})}{\sigma_g\psi_g}=qfratio$$

    Where a qfratio > 1.00 indicates a "source dominated" cell, and a ratio < 1.00 indicates a "streaming dominated" cell, where "source" includes the combined scatter, fission, and independent angular source terms. Note this simple relationship is readily available when solving for angular fluxes within each cell.
  - The *qfratio* is the computed ratio of the cell angular source density to the cell angular collision density; if this ratio is greater than a user prescribed value (based on testing, the default for *qfratio* = 1.00), then the DTW scheme is automatically selected without attention to the DTW weight factors, since in a source dominated cell, the DTW scheme performs optimally. Therefore, with the *qfratio* parameter, upgrades will *only* occur when the fine mesh cell is one that is "streaming dominated" away from source regions. The *qfratio* value is set in the third entry of the "dtwmxw" vector parameter (see the description in Block IV input). The recommended default value of *qfratio* is 1.00.

Differencing Scheme Lock Feature. *PENTRAN* also allows the user to restrict ("lock-in") any differencing scheme in a particular coarse mesh (by setting the differencing scheme number in that respective coarse mesh to a negative number, as noted in Table 2.2, far right). For vanishingly thin cells, experience has shown that the user may wish to "lock" the DTW scheme in those cells. If the use of a specific algorithm is not a strict requirement, *adaptive differencingis recommended.*

Parallel Differencing Issues. If parallel decomposition is used with adaptive differencing, a synchronization is made among the processors working on a particular coarse mesh to upgrade to the *same* adaptive differencing scheme, even if an upgrade is not required by *all* processors. This lends the adaptive procedure to a degree of numerical consistency, so that the user is certain of the differencing algorithm rendering a solution in a particular coarse mesh. Note that in the case of a fuel pin bundle immersed in water, flux gradients could be steep, and the EDI method may end up being selected, since it has demonstrated flexibility and accuracy in handling source region attenuation problems. Clearly, zones containing strong sources may not likely require many (if any) DZ fixups, although this depends on the cross sections, mesh size, and parallel decomposition strategy used; Hunter, et al demonstrated that when DZ is used, parallel decomposition can directly increase the number of fixups applied.

3-D Flux Moments. Regardless of the differencing method used, following an update of the group source and angular flux transport sweep through each cell, cell group scalar, cosine, and sine flux moments are updated using the latest cell average iterates. Therefore, equations (2.65) are based on the cell averaged angular flux as presented in equation (2.46), and are updated using the following quadrature expressions (with an implicit group *g* subscript):

$$\phi_l = \frac{1}{8} \sum_{m=1}^{M} w_m \psi_{A\,m} P_l(\mu_m) \tag{2.65a}$$

$$\phi^k_{C\,l} = \frac{1}{8} \sum_{m=1}^{M} w_m \psi_{A\,m} P^k_l(\mu_m) \cos(k\varphi_m) \tag{2.65b}$$

$$\phi^k_{S\,l} = \frac{1}{8} \sum_{m=1}^{M} w_m \psi_{A\,m} P^k_l(\mu_m) \sin(k\varphi_m) \tag{2.65c}$$

Note that these expressions include the 1/8 term from the criteria that quadrature weights were derived summing to one in each octant. Azimuthal angles are determined from:

$$\varphi_{m0} = \arctan\left(\frac{\xi_m}{\eta_m}\right) \tag{2.66}$$

where care must be taken to obtain the correct phase of the angle on the unit sphere. If $\xi_m < 0,\ \eta_m < 0,$ then $\varphi_m = (\varphi_{m0} - \pi)$. If $\xi_m > 0,\ \eta_m < 0,$ then $\varphi_m = (\varphi_{m0} + \pi),$ otherwise, $\varphi_m = \varphi_{m0}$.

Boundary Conditions. Standard boundary conditions include specular reflective, albedo, and vacuum (zero return current) boundaries. For albedo boundaries, entering angular fluxes are reflected from surfaces with normal vectors parallel to the *x* axis, are, with $\widehat{\Omega}_m = \langle \mu_m, \eta_m, \xi_m \rangle$:

$$\psi_{xBdy}(\widehat{\Omega}_m) = a\ \psi_{xBdy}(\widehat{\Omega}'_m) \quad \text{where} \quad \widehat{\Omega}_m \cdot \hat{i} = -\widehat{\Omega}'_m \cdot \hat{i} \quad \text{and} \quad \widehat{\Omega}'_m = \langle -\mu_m, \eta_m, \xi_m \rangle \tag{2.67}$$

Similar formulations hold for angular fluxes reflected from a surface normal to the *y*-axis with $\widehat{\Omega}'_m = \langle \mu_m, -\eta_m, \xi_m \rangle$ and normal to the *z*-axis with $\widehat{\Omega}'_m = \langle \mu_m, \eta_m, -\xi_m \rangle$. In each case, the albedo factor *a* can be energy group dependent, and is equal to unity if the boundary is fully reflective, or equal to zero if the boundary is a vacuum. *PENTRAN* allows for group dependent albedo boundaries on each surface.

Rebalance Methods. *PENTRAN* contains options for standard rebalance methods, as well as for a simplified multigrid acceleration scheme. Each iterative acceleration schemes is compatible with any of the available differencing schemes with no special treatment, and they can be used simultaneously to accelerate the iteration process. Coarse mesh rebalancing involves requiring the integral balance equation to hold for each energy group for each coarse mesh (Reed, 1971). System rebalance is follows the same overall process and procedure as coarse mesh rebalancing, although rebalance is only performed on each energy group over the entire problem. Due to the parallel memory structure in *PENTRAN*, both of these methods require integration and message passing to complete the rebalance.

The integral balance equation for each group, after applying Gauss' theorem to the streaming divergence, is

$$\int_A J \cdot dA + \int_V (\sigma - \sigma_s)\phi_o dV = \int_V (Q_{s(D,U)} + \frac{Q_{fiss}}{k_o})\, dV \tag{2.68}$$

where

J = leakage current across surfaces A bounding volume V

$\sigma - \sigma_s$ = within group removal cross section

ϕ_o = scalar flux

$Q_{S\,(D,U)}$ = down- and up-scattering source

$\frac{Q_{fiss}}{k_o}$ = volumetric fission source term

Q_{Ext} = volumetric external source term

Introducing rebalance factors f_{ijk} for each coarse mesh of volume V_{ijk} in an *x*-*y*-*z* gridspace for each energy group, and using partial currents normal to each respective surface, equation (2.69) is a general expression for rebalance for each coarse mesh; surfaces are labeled with respect to each local coarse mesh cell:

$$f_{ijk}\,((Jx^-_{ijk}|A_{x\ In}) + (Jx^+_{ijk}|A_{x\ Out}) + (Jy^-_{ijk}|A_{y\ Right}) + (Jy^+_{ijk}|A_{y\ Left}) + \tag{2.69}$$

$$(Jz^-_{ijk}|A_{z\ Bottom}) + (Jz^+_{ijk}|A_{z\ Top}) + (\sigma - \sigma_s)\phi_{o\ ijk}V_{ijk}) - f_{i-1,jk}(Jx^+_{i-1,jk}|A_{x\,Out}) - f_{i+1,jk}(Jx^-_{i+1,jk}|A_{x\,In})$$

$$-f_{i,\,j-1,k}(Jy^+_{i,\,j-1,k}|A_{y\,Left}) - f_{i,\,j+1,k}(Jy^-_{i,\,j+1,k}|A_{y\,Right}) - f_{ij,k-1}(Jz^+_{ij,k-1}|A_{z\,Top}) - f_{ij,k+1}(Jz^-_{ij,k+1}|A_{z\,Bottom})$$

$$= (Qs_{(D,U)\ ijk} + \frac{Q_{fiss}}{k_o}_{\ ijk} + Q_{Ext\ ijk})V_{ijk}$$

LU Factorization. Simultaneous solution of the system of equations that result from Equation (2.62) for each coarse mesh in each energy group is required; this is performed in *PENTRAN* using a direct Cholesky-LU factorization algorithm.

Partial Current Rebalance. All rebalance factors are damped using restrictions equivalent to Partial Current Rebalance (Rhoades, 1981). Therefore, this means that maximum rebalance factors are set by the worst imbalance, with a damping factor applied in further rebalance operations to prevent divergence. To retain scalability and bound the memory required for this direct solution method, *zoned rebalancing* over subsets of coarse meshes is required if the number of coarse meshes exceeds the maximum rebalance matrix size, *or if the user specifies a subset of contiguous zones* over which to constrain particle balance following each source iteration. If all of the coarse cells in a zone are not local to the processor, rebalance for that zone is by-passed. This direct, zoned solution scheme for rebalance is necessary (as opposed to an iterative technique) to achieve adequate processor synchronization and minimize rebalance overhead.

***PENTRAN* Rebalanceing Options.** A user can specify System Rebalance, Partial Current Coarse Mesh Rebalance, or an alternating combination (System-PCR) rebalance scheme. The number of zones can be defined, with a maximum damping factor for PCR, and skip options (see the section on *PENTRAN* Input)

Although it is rare, it should be noted that numerical instability, even with PCR damping, can occur with certain processor decomposition strategies (as noted by Reed for serial rebalance). To mitigate this, there are parallel synchronization options for each method (see the section on *PENTRAN* Input). Forced synchronization can also be useful, since latency effects on some older parallel machines may require more parallel synchronization; this arises since rebalancing deals with select "planes" of processors, depending upon how the user has decomposed the problem. The user should always default to a minimal synchronization rebalance option (unless there is a reason not to, such as a processor deadlock), because added processor synchronization adds to parallel overhead.

Multigrid Methods. Multigrid spatial acceleration methods essentially use coarse grid iterates projected to correct fine grid iterates, and are best described by transforming the spatial domain into the frequency domain (with units of inverse length) using the Fourier transform. High frequency errors present in successive fine grid iterates are readily reduced by each fine grid transport source iteration. However, the low frequency persistent errors present in the fine grid iterates become increasingly difficult to reduce with mesh grid refinement. In comparing a fine grid to a coarse grid formulation of a finite difference problem, the number of possible Fourier frequencies correspond to the number of equations in the linear system. Since this depends on the stepsize used in the problem, a larger mesh step size will limit the number of possible Fourier modes (frequencies). The highest frequency error components at a point on a coarse mesh will typically correspond to low or mid-range frequency error components for that same point on a fine mesh. Therefore, if a value from one iteration on the coarse grid were projected to correct the fine grid solution, the correction should be greater (and most often is significantly greater) than a single fine grid iteration by itself. This is because the low frequency errors in a fine grid problem are most effectively reduced by using a correction from a coarse grid solution. Therefore, multigrid methods effectively increase the rate of convergence in the iterative scheme. Because the asymptotic rate of convergence is the negative logarithm of the spectral radius of the linear system, use of a multigrid procedure effectively reduces the spectral radius in comparison to the spectral radius of the single fine grid system. This is the basis of the multigrid (also called multi-level) approach.

Workload Issues. For transport applications, Nowak, Larsen, and Martin (1987) demonstrated via Fourier analysis that if a mesh size is doubled (for computing on coarser mesh with a high scattering ratio), the number of iterations required to reduce the *high frequency error* by a factor of 10 is ***doubled*** (demonstrated using a weighted diamond scheme). At the same time, in three dimensions, the ***number of cells is reduced by 1/8*** (a factor of two along each axis). ***Therefore, the overall work required on the coarser mesh is 25% of that on the fine mesh.*** Nowak, et. al. also observed that if the scattering ratio is small, multigrid methods will not be as beneficial, since the transport operator effectively reduces the iterative error. This is expected, since the spectral radius of the difference equation is directly proportional to the scattering ratio.

From this, it is readily inferred that problems that are "less elliptic" (e.g. with lower scattering ratios) are dominated by errors that require a high frequency ***local*** iterative correction, and are therefore solved readily by the transport operator on the fine grid. Alternatively, problems with high scattering ratios are elliptic in nature and are dominated by a ***global*** low frequency error; these problems benefit the most from a low frequency iterative error correction provided by a multigrid solution. Overall, the key to obtaining accelerated convergence in a multigrid scheme is to use a powerful (yet computationally inexpensive) equation for a coarse grid iteration step. Yet due to the immense memory requirements

demanded by multigroup parallel transport methods, there is a need to conserve the memory demanded per processor.

Simplified "Slash(\)" Multigrid Scheme in *PENTRAN*. To capture some of the beneficial acceleration effects of a multigrid scheme and still conserve memory, the *PENTRAN* code has both a medium and fine mesh grid structure contained inside each coarse mesh. The medium grid can be automapped to set materials to the "nearest neighbor" fine mesh material, so one need only describe the fine mesh material distribution in the input.

For a medium grid, *PENTRAN* iterates to a prescribed tolerance on the medium grid, and then overwrites medium grid angular flux values as they are projected onto the fine grid. In this way, the same arrays are used to store both grids. Recall that angular fluxes are partitioned in memory local to each processor in *PENTRAN*, so that only the portion of the phase space assigned to each processor is stored. In spite of this, use of ***two*** transport grids would add to memory overhead, and increase the minimum processor pool required to solve a large problem. Therefore, the medium grid angular fluxes are lost after they are converged to a suitable tolerance and then projected onto the fine grid using 3-D Taylor Projection Mesh Coupling, or TPMC (presented in the next section). Using this procedure, the low frequency error on the fine grid can be largely eliminated, causing the solution to be dramatically accelerated. This scheme of projecting converged coarse grid values to the fine grid with TPMC is defined here as a "Simplified" Multigrid approach, and differs from more conventional multigrid methods ("V" cycle) that use residual error corrections between grids. Using this multigrid scheme in *PENTRAN*, test problems with high scattering ratios have demonstrated convergence with as much as an order of magnitude fewer iterations compared to the same problem solved using a single grid iteration sequence (Sjoden and Haghighat, 1996).

ACCELERATION SCHEMES: PRECONDITIONING OPTIONS

PENTRAN has a capability to use preconditioning to initialize zeroth and first Legendre based polar angle flux moments at the start of a transport computation. Due to storage limitations, even with binary data format options, only zeroth and first Legendre polar angle moments are stored (supporting the first two moments in Equations 2.65a). This means that higher moment terms (beyond 1st order) and associated Legendre moments (e.g. sine and cosine moments from the 3-D S_N equations, etc, as in Equations (2.65b) and (2.65c)) are *not* preconditioned. Because the dominant terms are the polar angle terms, preconditioning can provide speedups in the iterative solution, reducing the time and number of iterations by a factor of from two to five, depending on the problem, decomposition, iterative algorithm, etc.

***REPRO* Tool for Preconditioning.** Often, one must rerun a transport problem in several parametric studies, such as in depletion and burnup, in which case it is useful to be able to take a previously completed *PENTRAN* transport solution and use it to precondition the flux moments so as to accelerate problem convergence. The *REPRO tool* preconditions the problem flux moments using a previous transport run result, and therefore helps to accelerate the solution. This is an effective tool, since in spite of the fact that *PENTRAN* has a parallel memory structure, the *REPRO* tool has been implemented in a manner that is totally independent of the parallel processing execution/processor decomposition, leaving the user total flexibility in being able to precondition a problem with any previously converged run (for that same problem). Typically, speedups range from a factor of two to five over a flat-weighted starting flux.

Use of *REPRO* is accomplished by first executing *PENTRAN* on a problem of interest. Then, flux moments are extracted (as they normally are) from the binary (*probname.f#* files) files using PENDATA (with the user selecting the "REPRO" flux moment data format). Then, *REPRO* is executed, and builds a set of preconditioned flux files that, if present in the same subdirectory where the problem deck resides, will be automatically read at the start of the iteration sequence. A message indicating the use of preconditioning file read operations is indicated in processor output files.

SS_N-S_N Acceleration. Spcial access is provided in *PENTRAN* to link to any Simplified Discrete Ordinates (based on Simplified S_N equations (SS_N), which are an approximation to the transport equation using "even parity" that only have polar angle dependence). These SS_N equations result in a diffusion-like operator; there are limitations on the efficacy of the SS_N method, and the user should be aware of these. Access to SS_N setup is available by placing keywords into the "Title" line of the *PENTRAN* problem deck; this approach provides a methodology for preconditioned acceleration to reduce the iterative spectral radius. Longoni successfully applied his *FAST- SS_N* (Flux Accelerated Simplified Transport) technique with *PENTRAN* to achieve speedup in several problems (Longoni, 2004). As with *REPRO*, A message indicating the use of preconditioning file read operations is indicated in processor output files.

TAYLOR PROJECTION MESH COUPLING

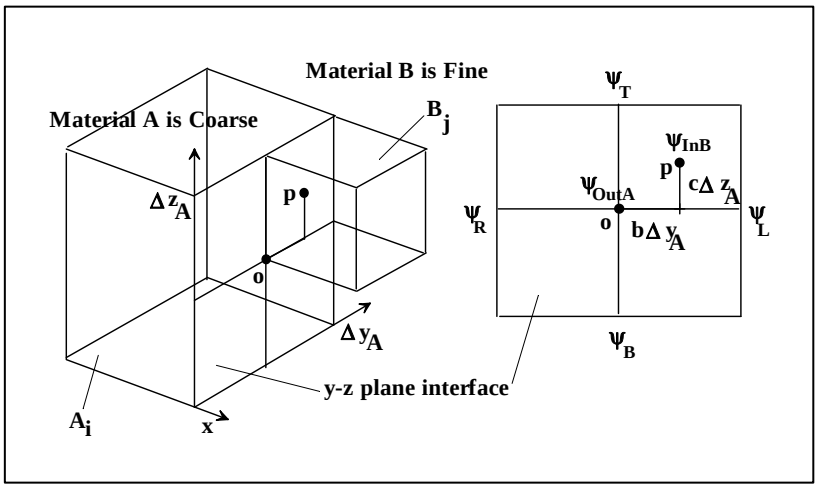


Fig 2.7: Transfer from "A" Coarse to "B" Fine where I=1 and J=4

Discontinuous Grids. In 3-D discrete ordinates (S_N) transport methods, it is desirable (and at times necessary) to use discontinuous spatial grids in coarse cells or at material interfaces. On a single processor machine, variable meshing permits high definition in regions of interest, with coarse grids used in less important regions. In parallel processing, variable meshing can be used to reduce load imbalances in problems spatially decomposed over large processor arrays.

In a typical S_N iterative solution scheme, a source iteration is performed, followed by sweeps in angular flux for each discrete ordinate using a spatial differencing scheme. As the sweeps progress through the various spatial grids in a problem, particle conservation must be maintained at any discontinuous grid interfaces. Typically, this is accomplished using a simple balance of particles streaming across a boundary surface, resulting in a zeroth order approximation. This procedure works well in the case of sweeps progressing from fine grids to coarse grids; however, a loss of information occurs as angular sweeps progress from coarse grids to fine grids.

TPMC. *PENTRAN* uses the *Taylor Projection Mesh Coupling* (*TPMC*) scheme for an *x*-*y*-*z* S_N method that attempts to mitigate the loss of angular flux information as sweeps are made from coarse to fine grid interfaces. Note that the relationship between surface and cell averaged angular fluxes is determined using a spatial differencing scheme, as already discussed. In the figure above, consider a transfer of angular fluxes from the "A" boundary to the "B" boundary, crossing the (*y*-*z*) plane in the +*x* sweep direction. The mismatch between the cells results in a discontinuity that must be resolved while sweeping along angles.

Taylor Projection Algorithm. To reduce the loss of information that occurs when a transfer of boundary fluxes is made from a coarser -to- finer grid pitch, a Taylor Projected Mesh Coupling of surface angular fluxes is used. This *TPMC* scheme amounts to using the coarser A surface angular fluxes to approximate partial derivatives in a truncated Taylor series expansion; the expansion is used to project surface angular fluxes exiting A into B. Taylor interpolation methods have been employed in computational fluids and heat applications,

but often with difficulty in maintaining strict conservation (Kallinderis, 1992, and Phillips and Schmidt, 1984).

The first step in the TPMC scheme involves the direct transfer of the angular fluxes exiting cell A at point ***o***, ψ_{OutA}, to the angular flux entering a B cell at point ***p***, which is ψ_{InB}. Using a truncated Taylor expansion from surface centers connected between A and B:

(2.69) $$\psi_{\text{inB}} = \psi_{\text{outA}} + b\,\Delta y_A \frac{\partial \psi}{\partial y}|_A + c\Delta z_A \frac{\partial \psi}{\partial z}|_A + O(\Delta^2)$$

With partial derivatives approximated by central differences computed from A boundary values:

(2.70) $$\frac{\partial \psi}{\partial y}_A = \frac{(\psi_L - \psi_R)}{\Delta y_A} + O(\Delta^2) \quad \text{and} \quad \frac{\partial \psi}{\partial z}_A = \frac{(\psi_T - \psi_B)}{\Delta z_A} + O(\Delta^2)$$

Note that the scale factors *b* and *c* are the relative fractions of cell A width connecting surface centers along the *y* and *z* axes, respectively. Combining equations (2.63) and (2.641) results in

(2.71) $$\psi_{InB} = \psi_{OutA} + b(\psi_L - \psi_R) + c(\psi_T - \psi_B)$$

Particle Conservation. Equation (2.71) permits a simple first order projection of angular boundary fluxes exiting from the coarser A cell entering into the finer B cells. During a transport sweep for a given angular ordinate, this process is repeated for all B cell boundaries. If a flux projection yields negative values, the *absolute value* of the *most negative* angular flux projected among the B surfaces is then added into each B surface flux, causing the minimum projected flux to be zero. Because particle conservation during the transfers from *I* cells of A to *J* cells of B must be conserved, normalized *projection fractions* f_j for each B surface angular flux are then computed by normalizing all values projected (in Figure 2.7, *I*=1 and *J*=4):

(2.72) $$f_{\text{j}} = \left(\frac{\psi_{inB_j}}{\sum_{j=1}^{J} \psi_{inB_j}} \right)$$

Particle balance is achieved by first computing the particle outflow from the surface of A:

(2.73) $$flow_{\text{A}} = \sum_{i=1}^{I} (\psi_{outA_i}) \mu_m (\Delta y_{Ai} \Delta z_{Ai})$$

Then, using equations (2.72) and (2.73), the *final TPMC* projected angular fluxes are obtained from

$$\psi_{InB_j} = \frac{f_j \cdot flow_A}{\mu_m \, \Delta y_{B_j} \, \Delta z_{B_j}} \tag{2.74}$$

Note that a traditional, *zeroth order* projection is obtained by setting the scaling factors b and c to *zero*. In the event that grids do not directly "match," nearest neighboring cell centers are used to project angular fluxes.

Based on test problems, the *TPMC* scheme provides fine grid fluxes that are 3 times or more accurate than traditional zeroth-order methods in cases where a finer meshed region of interest (ROI) is surrounded by coarser meshes. Therefore, *TPMC* as a minimum mitigates some effects of using coarser grids in cells surrounding a ROI. In more realistic problems with steep gradients, the differences between *TPMC* and zeroth-order schemes are more pronounced. Accuracy of fine mesh fluxes have shown a factor of three improvement in the Kobayashi problems using TPMC as opposed to zeroth order coupling in problems using discontinuous grids (Sjoden and Haghighat, 2000).

Criticality. An algorithm to determine the criticality eigenvalue of a system, consistent with complete parallel phase space decomposition, is in place in *PENTRAN*. The existence of the criticality eigenvector is mathematically derived from Perron's theorem, which states that a positive matrix has a unique positive eigenvector with a single positive eigenvalue that is greater than the modulus of any other eigenvalue for the matrix. In reactor physics applications, the criticality eigenvalue represents the fundamental effective multiplication factor k_o that dominates the the nuclear system after all higher harmonic transients have died away (Nakamura, 1977).

Power-Aitken Iteration. In *PENTRAN*, the outer iteration fission source is based on the previously converged inner iteration flux estimate. The fundamental (criticality) eigenvalue k_o is determined from a variation of the Damped Power-Aitken eigenvalue iteration, where the newest eigenvalue is derived from an Aitken extrapolation (used successively after the 3rd iteration) and the relative difference between successive eigenvalue iterates. Note that due to their added parallel storage and synchronization requirements, Chebyshev acceleration methods were not considered. In any one outer iteration in *PENTRAN*, the criticality eigenvalue estimate for iteration t is updated using the following procedure on each processor, with the most recent fission source estimate Q_{fiss}^{t} (based on the most recent scalar flux) weighted with an arbitrary unit weighting vector:

$$\tilde{k}_o^{\,t} = k_o^{\,t-1} \frac{\left\langle Q_{fiss}^{t}, 1 \right\rangle}{\left\langle Q_{fiss}^{t-1}, 1 \right\rangle} \tag{2.75}$$

An Aitken extrapolation is, (by default) then performed (unless deactivated) if a minimum of 3 successive iterates are available, where

$$\delta\tilde{k}_o^{\,t} = -\frac{(\tilde{k}_o^{\,t} - k_o^{\,t-1})^2}{(k_o^{\,t-2} - 2k_o^{\,t-1} + \tilde{k}_o^{\,t})} \tag{2.76a}$$

$$k_o^{\,t} = \tilde{k}_o^{\,t} + \delta\tilde{k}_o^{\,t} \tag{2.76b}$$

The extrapolation correction $\delta\tilde{k}_o^{\,t}$ in equation (2.76) is zeroed in the final outer iteration as a consistency check to verify that the outer iteration is fully converged. In *PENTRAN*, it was also determined that a purely statistical convergence check of the last four (4) outer iterations should be assessed to guarantee that the fundamental eigenvalue has converged. **This was found to be essential in a parallel execution.**

Therefore, a minimum of four outer iterations are required to yield a solution to a criticality eigenvalue problem, regardless of the outer loop convergence criterion. This scheme is a Power-Aitken scheme, allowing convergence to the correct criticality eigenvalue, regardless of parallel domain decomposition applied. Note that the group window option or

the restart option, which can be leveraged to conserve memory in down-scatter-only transport problems when using the multigroup source iteration, are *not* available for criticality problems, which inherently require all groups to explicitly compute the fundamental eigenvalue.

Three Methods of Eigenvalue Reporting. *PENTRAN* reports three methods of computing criticality eigenvalues:

(i) Power iteration eigenvalue and outer source iteration tolerance; Example:

"System keff Single Mesh : 0.957541 tol= 5.0000E-05

(ii) Statistical average of last four power iteration eigenvalues with 2-sigma standard deviation; Example:

"System keff Avg Last4 : 0.957541 2sa= 0.000001"

(iii) Balance-derived eigenvalue, from assembly of integrated Production divided by Loss, integrated over the parallel processing grid with relative balance error; Example:

"System Balance (Prod/Loss) keff:0.957538 relbalerr= -3.4835E-06"

The example shown included actual results from a test problem, where it is evident that the end result for the criticality eigenvalue was 0.95754 based on the convergence.

In all cases, these three methods of reporting a criticality eigenvalue for the system should be consistent; if they are not, then this reduces the confidence that the problem has reached full convergence, and the results should be questioned beyond the normal data review process.

The key to the parallel scalability of the *PENTRAN* code lies in its internal structure; fundamentally, parallel execution and work allocation is fundamentally established by a decomposition weighting vector.

- A transport problem input deck is read and processed by each processor.
- Following several initialization sequences, the angles, energy groups, and spatial cells in the problem are automatically decomposed according to a user-specified *decomposition weighting vector*. This decomposition vector contains arbitrary weighting factors for angular, group, and spatial decomposition. This vector allows a user to "prioritize" the decomposition strategy used in a parallel execution of the problem *without* specifically assigning an exact number of processors (a user can also block decomposition in a particular variable, or lock-in any specific number of processors, if desired).
- *PENTRAN* auto-schedules the decomposition of the problem for the user at execution time onto *n* processors, *and self adjusts* the number of processes assigned for the problem being solved. However, this is performed within the restrictions allowed by the user's decomposition weights.
- If the weighting scheme specified by the user leads to an *odd number of processes* scheduled on an *even number of processors*, a halt statement for “processor utilization below 100%” is issued, with a message to the user to change the decomposition weights or the allocated number of processors.
- The number of *processes* must be greater than or equal to and divisible by the number of *processors*.
- Ultimately, the auto scheduling in *PENTRAN* leads to a 3-D Cartesian, virtual processor array topology, with angular, group, and spatial decomposition axes. The phase space of the problem is essentially projected onto this virtual processor topology during parallel execution.

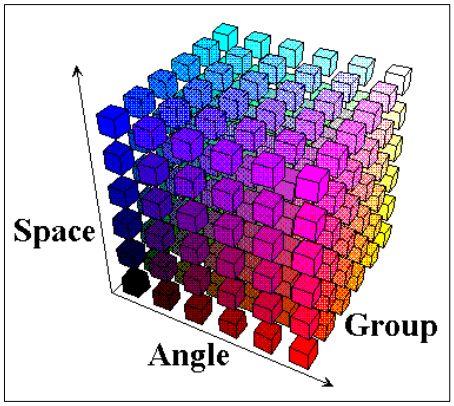


Fig 2.8 3-D Cartesian Virtual Processor Array

Central to any successful parallel code is a consistent and efficient communications structure. This is accomplished in *PENTRAN* through the use of *MPI*-generated custom *Communicators* constructed for selected levels of processors assigned among the angle-energy-space Virtual processor array.

Communicators. Efficient communications over an arbitrary domain decomposition are obtained from the use of complex inter-process communicator buffers (or *communicators* as they are called in *MPI*).

These communicators are generated on each process during the distribution of problem data over the virtual processor array completely transparent to the user. They enable communications among specific subgroups of processors in the virtual topology. For example, consider a scenario where angles, groups, and sweeps are fully decomposed on a processor array. If a scalar flux is required for all groups in a particular coarse mesh cell, the reduction operation to collect all of the angular fluxes for quadrature in their respective groups uses a communicator that links processors "owning" the groups and angles for that specific coarse cell. *PENTRAN* builds communicators to selectively communicate with selected processors containing:

- All angles for a specific coarse cell and energy group
- All energy groups and angles for a specific coarse cell
- All coarse cells and angles for a specific energy group

Communicator Limits. On some MPI implementations, an upper limit of a few thousand communicators may be in place; this limit often varies, dictated by the finite number of possible unique communicators available and system memory allocated for these tasks.

To retain maximum portability and scalability, *PENTRAN* will automatically minimize the number of unique communicators “built” during each problem execution, re-using communicator buffers through “aliasing” if they contact the same set of processors. Any excess communicator buffers that are not useable by the processor (null value) are immediately released back to the available *MPI* communicator buffer (memory) pool. Communications overhead is further minimized in *PENTRAN* by array packing routines. All array data transferred among processors is packed by the sending processor, then unpacked by the receiving processor after communications are completed. Embedded tags in the array pack ensure that data arriving from one processor to another processor are correct, maintaining message passing data integrity. Embedded tags offer unique parallel algorithm checks for code developers, because large scale parallel code debugging can be quite challenging; tags can be useful in detecting parallel execution errors, since new code and data features added that pass serial tests, but cause parallel data faults are usually flagged by these data checks during parallel execution.

In order to handle complete decomposition, processes in *PENTRAN* must be tracked. Imagine the case of a scatter and reflection out of one processor onto another, from ordinate to ordinate. With thousands of ordinates, millions of mesh cells, tens of energy groups, one must have precise mapping of parallel stored sata. This is accomplished in *PENTRAN* using *Process Mapping Arrays*.

Parallel Process Mapping Arrays and Message Passing:

- Following the assignment of the processors and communicators in the virtual processor array, phase space variables assigned to a particular processor are tracked and computed independently.

- Local variable dimensions (when necessary for memory conservation) and *process mapping arrays* are used on each processor. This allows all memory to be completely partitioned on each processor for all medium (and fine) mesh variables, including memory intensive angular, scalar flux, and current (*J*) variables.

- *Process mapping arrays* contain the global coarse mesh cell, energy group, angular sweep octant, and octant ordinate ($\hat{\Omega}$) indexes that are locally processed in locally dimensioned arrays.

- The 'kpmap' Array. A global *processor **"reference"** mapping array* (kpmap) is also maintained by each processor (allocated identically on each processor during problem setup), and permits access to the *processor number* containing any coarse cell, group, sweep octant, and octant ordinate. This permits "send" and "receive" processors to be readily identified if point to point message passing is required, as occurs in the case of reflective boundaries with angular decomposition, or with spatial decomposition.

- When message passing is performed, global index references are used during communications among processors. After messages are received, global variable array indexes are translated back to local indexes for storage.

- Messages and various communications are continuously cross-checked using the (kpmap) *processor **reference** mapping array,* the *process mapping arrays,* and the translation routine to verify data integrity.

- Because the *process mapping arrays* store global angular, energy, and spatial variable indexes for locally stored variables, any schemes that require specific ordering can be readily accomplished. Therefore, rearranging the order that global variable indexes are assigned and stored in the *process mapping arrays* alters *where and in what order* they are locally processed.

Memory Estimate/Processor. The bulk of memory (per processor) in *PENTRAN* is consumed by storing the cell centered and surface angular fluxes. Grid arrays only need be as large as required for the largest spatial grid in a single coarse mesh cell, the total number of *local* energy groups, and the total number of *local* angular sweep octants for the problem being solved. Therefore, larger problems can be solved directly in *PENTRAN* by adding more processors with increased decomposition. The amount of memory, in bytes, consumed by each processor prior to optimization by memory tuning, is *approximated* by

(2.77) Memory Bytes ~ λ maxcmc maxfmc maxglc maxswp maxqdm

where the parameters represent a scaling constant λ, the maximum local coarse meshes, fine meshes per local coarse mesh, local groups, local octant sweeps, and angles per octant (bound by angular quadrature method), respectively, for a processor. From experience, a few percent of processor memory should be reserved for system level processes, caching, etc, depending on the machine; therefore, the left hand side of equation (2.77) should be regarded as "free" memory per processor. The constant λ is typically between 50 and 60 for the single precision version of *PENTRAN*, depending on the compiler. This number could be far less with the automatic memory optimization in *PENTRAN* (discussed below). Note also Equation (2.77) assumes that the group window or restart options (which can provide a significant reduction of memory requirements) are *not* used. These options can save a large amount of memory for a fixed source problem.

Impacts to Parallel Performance. The amount of memory required for a specific calculation is initially estimated by the code when the parameters located at the top of an input deck are provided. <u>The user should avoid setting some parameters to values significantly greater than required for a problem.</u>

- Overall memory estimate. This is only an estimate provided by the code since it occurs prior to any optimization , and is compared to the *maxmem* parameter to determine if the memory required exceeds the available memory. Therefore, knowing the memory bound per node, one can set parameters to cover a general problem type, and determine how many processors are required for a problem, depending on the decomposition approach used. Memory Tuning occurs after process assignments have been made, and represents the true memory requirement, per processor, for the problem being considered.

- Generally, a cluster with 8 or 16 processors having at least 2 Gb of allocatable RAM per processor is considered a reasonable cluster. Note that this means one quad-core node should have ~8 Gb available, allocatable uniquely to each processor in 2 Gb parcels, for MPI distributed parallel problems.

- Parallel efficiency can be directly affected by parameter settings. Some automated help is available for properly tuning parameter settings to avoid problems in allocating more memory than needed, particularly for array buffers for parallel message passing.

Also, see the discussions below on "memory tuning" and "automated parameter repair".

- The following parameters directly affect message length (and therefore message passing time): ***maxleg, maxgrp, maxfmc*** (and therefore ***maxmmc*****) ,** and ***maxqdm*****.** The first two parameters are the maximum Legendre moment required and the maximum global number of energy groups required, respectively. The next parameter is the maximum number of fine mesh cells per coarse cell, and the last parameter is the number of directions per octant on the unit sphere for each fine mesh.

- The minimum total number of equations in a the transport matrix being solved in *PENTRAN* can be determined by the product of (total_fine_meshes * total_energy_groups * unit_sphere_directions). For example, a problem containing 100,000 fine meshes among 5 energy groups using S24 Legendre-Chebychev quadrature (a minimal whole body phantom medical physics simulation) totals 312 million equations, solved in a 312 million by 312 million element matrix. Of course, *PENTRAN* stores these equations, along with the accompanying overhead of flux and source moments, etc, across the allocated processor array. *PENTRAN* also has options of a group window (for fixed source problems, parallelizable within the window) for fixed source problems to minimize the memory requirements.

- Since it is conceivable that a heterogeneous parallel cluster could be used to run *PENTRAN* under *MPI*, parameters should be set to allow the code to fit within the total memory on the *minimum capacity machine*, since the same code must run on all machines simultaneously, albeit independently.

- Note that all important parameter settings and total memory demand are also provided in the run logfile for inspection by the user.

Parallel Scaling. Some variables in *PENTRAN* are dimensioned using an integral memory approach for efficiency, as they would provide a small memory savings, yet add appreciable message passing overhead.

- Energy group coarse grid data and coarse cell spatial mapping data are stored using *global* problem dimensions on each processor. Partitioned memory practices are applied when the memory savings potential is large, as in the case of medium/fine mesh scalar and angular fluxes.

- Overall, due to a partitioned memory structure, the *PENTRAN* code has true data parallelism for memory intensive arrays (the angular and scalar fluxes), and is therefore a fully scalable code.

- Larger problems require more processors, although a capability of complete phase space decomposition in the angular, energy, and spatial variables with discontinuous meshing offers great flexibility in how subdomains are created to render a parallel transport solution.

    - Angular decomposition is first on a unit sphere sweep octant basis (as dimensioned by the *maxswp* parameter, and afterwards by the individual ordinate number in each octant, set by the *maxqdm* parameter)

    - Energy group decomposition is on a group number basis (local groups dimensioned by the *maxglc* parameter, global groups dimensioned by the *maxgrp* parameter)

    - Spatial decomposition is accomplished on a coarse mesh basis (locally allocated coarse meshes are dimensioned by the *maxcmc* parameter, while global coarse meshes by the *maxgcm* parameter)

- The most efficient parallel decomposition scheme is, however, different for each problem, and is difficult to anticipate; some progress has been made in this area through problem and decomposition analysis by Patchimpattpong and Haghighat (2002).

Automated Parameter Repair. This feature was added to *PENTRAN* to assist the user in determining parameter settings that will be tuned to a particular parallel run. This is accomplished by having the user place negative values (e.g. -1, -2 …) for each element in **decmpv**, "locking in" the number of processors intended for angular, energy, and spatial decomposition, respectively, and then if the **maxmem parameter is set as a negative memory value**, **then the code will analyze the parameter settings and reset the Fortran90 parameters to the best values suited for the parallel run.** The resulting problem deck will then be ready for parallel execution on the total number of processors (which should equal to the absolute value of the product of each **decmpv** element).

While the automated parameter repair *is robust, it can fail if the maxarr parameter is set too low—this dimension governs the maximum count of numbers read in during input for any general input field. Should a problem occur, it is recommended that the user increase the* ***maxarr*** *value. Also, the* ***nctlim*** *parameter is the maximum count of FIDO characters read during input for any general input field. Both of these parameters are used to interpret the input deck so that parallel processor work assignments may begin, so it is important that these parameters span the number of entries/FIDO characters read for the largest containing variable (the largest block often is in BLOCK V--Source).*

Memory Tuning. *PENTRAN* incorporates an automatic memory tuning feature which optimizes the amount of memory allocated for large storage variables (e.g. angular fluxes, etc). This minimizes the memory required per processor, yet is consistent with a Fortran-90 architecture.

- In the original architecture, *PENTRAN* used a simple "upper-bound" parameter dimension for the number of fine meshes contained in each coarse mesh. This tended to "waste" memory when defining only a small number of fine meshes in a coarse mesh in some parts of the problem geometry.

- To facilitate minimizing memory use, *PENTRAN* has a "Large Coarse Mesh/Small Coarse Mesh" two-array system, where the large coarse meshes still use a dimension of the upper bound specified by ***maxfmc*.** However, prior to execution, *PENTRAN* performs a "memory tuning" function that analyzes the entire problem workload, looking at the average and standard deviation of the global spatial mesh in each coarse mesh, and allocates a "Small Coarse Mesh" dimension< ***maxfmc*** that best optimizes the amount of memory required to store the problem.

- This can save a sizeable amount of memory, as it "tunes" the memory allocation based on the spatial fine mesh distribution assigned to each machine. Therefore, each machine can, if spatial decomposition is selected, have a completely different mix of "large" and "small" coarse mesh sizes, depending on the work assignment for that processor.

- The "Memory Tuning" option has allowed researchers to solve extremely large problems through better use of machine memory. Also, if a problem does not fit on a machine with angular decomposition, it may well fit with spatial decomposition and load balancing activated, since a balanced load with memory tuning can flatten the memory required on each machine (note: this can introduce inefficiency by reducing the convergence rate, since load balancing affects the order of boundary value updates).

KEY NOTABLE ISSUES FOR THE USER

The user should pay attention to the following issues in solving a problem with *PENTRAN* in parallel:

- Be aware of parallel parameter limits; The Automated Parameter Repair feature for properly setting up minimum memory parallel allocations using parameters is a very useful one, with important rules
    - Use large enough **maxarr** and **nctlim** parameters to make sure the problem can be read in properly
    - **maxlin** should also span to at least the number of input lines in the problem input deck.
    - As an example, if always using 2 processors for angular decomposition (which may be good to do since it does not degrade convergence in Cartesian geometry with vacuum boundaries), the **maxswp** parameter (that defines the number of octants locally stored) can then be set to 4 rather than 8, since a maximum of just 4 sweep octants will be processed on each machine.
    - Also note that the number of medium meshes (**maxmmc**) should always be set equal to the number of fine meshes (**maxfmc**) per coarse mesh, as currently required for the Simpified Multigrid method (since these dimensions currently map to the same arrays).

- Note that coarse mesh boundaries define subdomains for parallel spatial decomposition, acceleration methods (rebalance and multigrid), and assignment of the numerical differencing scheme used.
    - *Avoid using small numbers of meshes inside each coarse mesh*, since a processor synchronization is required after each processor completes a coarse mesh (if reflective boundaries and/or spatial decomposition is used).
    - Use at least 1,000 fine meshes/coarse mesh; however, more is generally better--this makes the calculation increasingly coarse grained. A reasonable target is 8,000-10,000 fine meshes per coarse mesh. Large deviations of this number between coarse meshes may lead to load imbalance with spatial decomposition.
- TPMC is used on each surface of each coarse mesh connected to other coarse meshes. It is advisable to use discontinuous meshing where needed, since TPMC is always being paid for, and memory tuning will attempt to optimize storage.

- Use of angular and spatial (or both) decomposition can be beneficial to rebalance acceleration, which can offer a significant increase in convergence (where it would not on *one* processor). This is due to ADS sweep ordering/partitioning, and will often offset the convergence penalty incurred from spatial decomposition.

- DZ may incur more negative fluxes with increased spatial decomposition, and can exhibit non-convergence in parallel executions. In recent studies with optically thick problems, better performance was obtained by starting with DTW differencing as a first scheme in the adaptive algorithm, followed by application of EDI (by *PENTRAN*) as needed.

- Avoid setting the maximum allowable DTW weight (using the **dtwmxw** setting in Block 4) too low--a premature shift to EDI could degrade convergence in source regions, since the EDI scheme performs best in streaming cases. Still, if the maximum weight is still high and a shift is made, it is clear that EDI is required. For advanced users, don't forget that meshing can be locked in *or* upgradable within a coarse mesh, depending on the scheme used.

- If not using group decomposition, use the standard multigroup iteration method by setting **methit=1**. This method allows for the most efficient use of particle downscattering by continuously accumulating the scattering source through groups g - 1 during convergence of group g, which can be a great savings when performing a P_3 or higher calculation.

- With complete group decomposition, the Hiromoto-Wienke 1-level scheme may be best, selected with **methit=2.** (These iteration options are discussed in more detail in the next section).

- Be careful in using group decomposition on *few group* problems if the scattering ratios vary greatly. Since few group problems are strongly coupled, one processor could tie up the rest with many iterations in a group with a high scattering ratio. It can be shown through Fourier analysis that a high scattering ratio sets the upper limit of the spectral radius for the transport source iteration. This effect may be ameliorated by activating automatic load balancing and/or multigrid acceleration, although at the same time, load balancing can also inhibit convergence in some cases.

- Use of widely varying medium and fine grids can hinder acceleration from the multigrid--a ratio of two fine/medium mesh along each axis is generally recommended. In cases where multigrid is not as effective (e.g. low scattering ratios), use a low medium grid tolerance just to obtain a good first guess on the fine grid. Also, a medium grid tolerance that is set too small can be beyond the achievable infinity norm error due to truncation error of the medium mesh differencing, and may never converge.

- In fixed source problems with multigrid, the user is responsible for properly defining the source, including the total number of source particles. The meshing on the medium grid in the source region must be set so that the source is resolvable; *PENTRAN* will balance the volumetric source strength to be consistent for the slash multigrid iteration, provided there is a mesh it can alias the source to from the fine grid. A warning will occur is there is no resolvble source on the medium grid.

- Be aware of disk storage requirements. Scratch files for FIDO input and material mesh specification will typically require 5 bytes of disk space per mesh per processor; typically, this can be stored in /tmp space on each processor. A typical run with binary scalar flux outputs can require disk space for *O*{(total # of meshes) * maxgrp * 60 Bytes} bytes. (note this is if Memory Tuning is not considered).

- Also, exercise caution when selecting high scattering moment dumps or angular flux binary dumps--file space could be quickly saturated.

- *PENTRAN* should be executed in parallel to the most practical extent possible if the problem is large. Restricting the use of the code to a single machine (when not already constrained by memory due to storage requirements) forfeits the effort required to perform all of the complex mechanics necessary for complete phase space parallelization. Some of these tasks include problem decomposition and distribution routines, partitioned memory mapping, range checking, global-local and local-global mapping, and the overall coarse grid based, discontinuous mesh code structure.

- Although there are some logical by-passes, many tasks required for parallel execution in *PENTRAN* are also carried out in serial as well, and amount to what is considered *parallel overhead*. *PENTRAN* was developed with the intent of solving *large* problems on a distributed parallel cluster *that could not be solved in a practical time or fit under the memory and/or disk constraints of a single CPU*.

SOURCE ITERATION SCHEMES

Two source iteration schemes are available: a "multigroup" source iteration (the GOFMGM routine) and a Hiromoto-Wienke source iteration (the GOFCHM routine).

Multigroup Source Iteration. The GOFMGM "multigroup" routine (see Block IV, selected using **methit**=1) iterates to convergence sequentially through each group, from group g_i to g_j, where (i<j), with an added convergence confirmation of the local scattering source if upscattering or group decomposition is indicated.

- Progression to the next (local) group is not performed until convergence in the current group is obtained.

- The convergence of the scattering source is computed using a norm based on the relative change of the integral sum of the angular scattering source for all locally processed coarse mesh cells, groups and angles. This is used only in the event of energy group decomposition with this iteration scheme. Note that energy decomposition with this scheme is **not** recommended.

- Convergence on the scattering source is checked following reported convergence in all groups. Parallel execution with any level of decomposition using the "multigroup" source iteration scheme is available without restriction.

Hiromoto-Wienke One-Level Source Iteration. The GOFCHM routine was designed to be consistent with the one-level TPCS chaotic scheme of Hiromoto and Wienke (1989), and can be implemented (see Block IV, selected using **methit**=2) with any phase space decomposition strategy in *PENTRAN*. Excellent parallel speedups have been demonstrated using various combinations of decomposition in the angular, energy, and spatial variables using this scheme (Sjoden and Haghighat, 1996).

- In this Hiromoto-Wienke scheme, each processor completes a source iteration in a *single pass* through a number of *locally* processed groups.

- Group flux moments are updated by a summing reduction among participating processors for the next scattering source calculation, but convergence is tested only on locally processed groups; this is followed by a new iteration, if needed.

- On a *shared memory* machine, this iteration procedure is *truly* chaotic, as group flux moments are updated for all processors instantaneously, resulting in chaotic convergence among the various groups scheduled on different processors. That said, on a *distributed* memory architecture, this iteration scheme is "chaotic" *in name only* since the group flux moment reduction is required following a sweep, which effectively acts as a processor synchronization. However, since convergence is tested based on

locally processed groups, the user is able to identify the relative speed of convergence in each group, as *PENTRAN* reports when convergence is achieved on each processor.

- Computation for all processors must continue, however, until all tasks are converged, as message deadlocks will occur if one participating processor is deliberately stalled.

Note: In general, if group decomposition is *not* used, the multigroup scheme will provide for faster convergence to a solution, with some exceptions. First, if there is upscattering, tests have demonstrated that the Hiromoto-Wienke scheme works best, since all local groups are iterated through sequentially, updating the upscatter components continuously. Also, faster convergence with the Hiromoto-Wienke scheme resulted when the simplified multigrid acceleration was used for any type of problem (Sjoden and Haghighat, 1996). As more users apply the code to a variety of test problems, this contributes to the experience and bounds the efficiencies of the code.

Code Benchmarks. Several problem benchmarks computed for one, two, and three dimensional, multigroup, multi-region, anisotropic problems have been tested using *PENTRAN* with DZ and DTW differencing and parallel domain decomposition. In particular, exact agreement (within the convergence tolerance) was obtained between solutions from the *TWOTRAN-II* , *TWODANT*, *DORT*/*TORT*, and *THREEDANT* single CPU production codes and parallel *PENTRAN* solutions. Similar agreement was demonstrated for criticality and adjoint test problems compared with *TWOTRAN-II*. An independent criticality benchamark using Hansen-Roach cross sections was performed with *PENTRAN* against MCNP in multigroup mode. The criticality eigenvalues rendered by each were the same within the limits of the cross sections and problem data. When comparing with the two dimensional codes tested, *PENTRAN* calculations were performed in 3-D with reflective boundary conditions along one of the three (*x*-*y*-*z*) axes. With single variable and hybrid phase space decomposition strategies, 3-D test problems were solved using dedicated timing benchmarks using the Cornell Theory Center IBM-SP2. Significant speedups were demonstrated, with high estimates of parallel code fraction (between 95% and 98%, based on Amdahl's Law).

Validated Benchmarks. An experimental *PENTRAN* benchmark established by OECD/NEA modeling the complete Venus-3 reactor in 3-D has been performed. The Venus-3 reactor is owned and operated by SCK-CEN nuclear energy research laboratory, located in Mol, Belgium. This facility houses Venus-3, a zero power research reactor designed to test partial length fuel assemblies and various test fuels. The core includes sixteen "15x15" sub-assemblies (as opposed to the typical "17x17" type). A cross section slice of the z-level 2 *PENTRAN* spatial mesh distribution through the Venus-3 reactor model (generated by *PENMSH*) is provided in Fig 2.9 below. Although each assembly rack has fewer pins, the pin-to-pin lattice pitch is 1.26 cm-- resembling a conventional assembly. Therefore, the Venus-3 facility serves as a practical model of a PWR reactor. Comparing 370 experimental reaction rate measurements from nickel, indium, and aluminum dosimeters, 95% of the *PENTRAN* calculated-to-experiment (C/E) values were within +/-10%, with the remaining 5% to within +/-15%. Therefore, *PENTRAN* agreement with experimental flux measurements in the Venus-3 facility was excellent, and compared well with other codes (see Figs 2.10 to 2.15.4). This *PENTRAN* solution was rendered in only 1.5 hours on 32 SP2 machines. Other validated problems presented here include an OECD/NEA Kobayashi Benchmark, a Dry Storage Cask, a

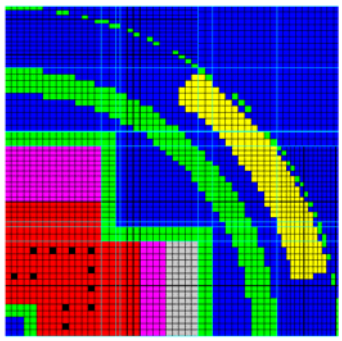


Fig 2.9 Venus-3 Z-Level 2 rendered by *PENMSH*

Prompt Gamma-Neutron Activation Analysis (PGNAA) device, a BWR Reactor, a He-3 detector system, an HEU Criticality Benchmark, a MOX 2-D & 3-D OECD/NEA benchmark, an X-ray room, and a CT Scan Simulation. Another application considered was a surreptitious WGPu problem as a Homeland Security demonstation.

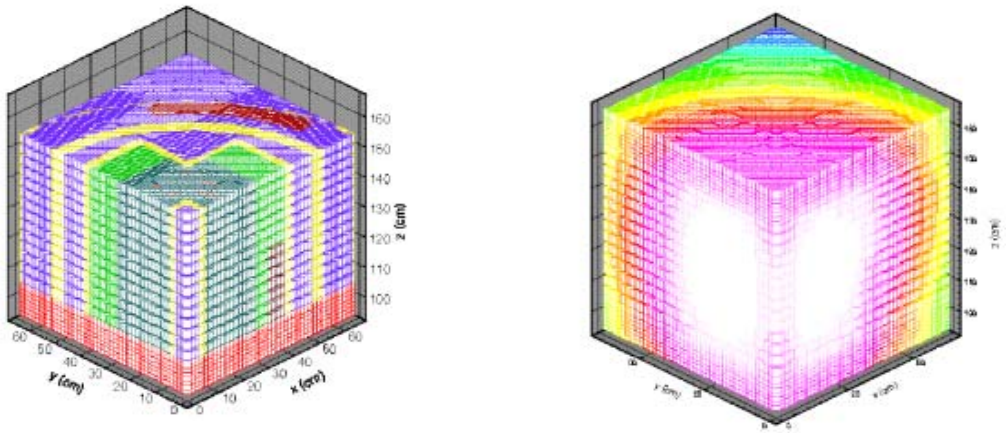


Fig. 2.10. PENTRAN™ simulation of VENUS-3 Reactor Dosimetry Benchmark Experiment, (Left) Discretized geometry, 85,000 cells, 26 Energy Groups, P3-S8 Discrete Ordinates, (Right) Group 1 Flux Solution. The 26 group calculation required 1.4 hours on 32 processors of an IBM-SP2 computer achieving 84% Ep. (Haghighat, Abderrahim, and Sjoden, 2000).

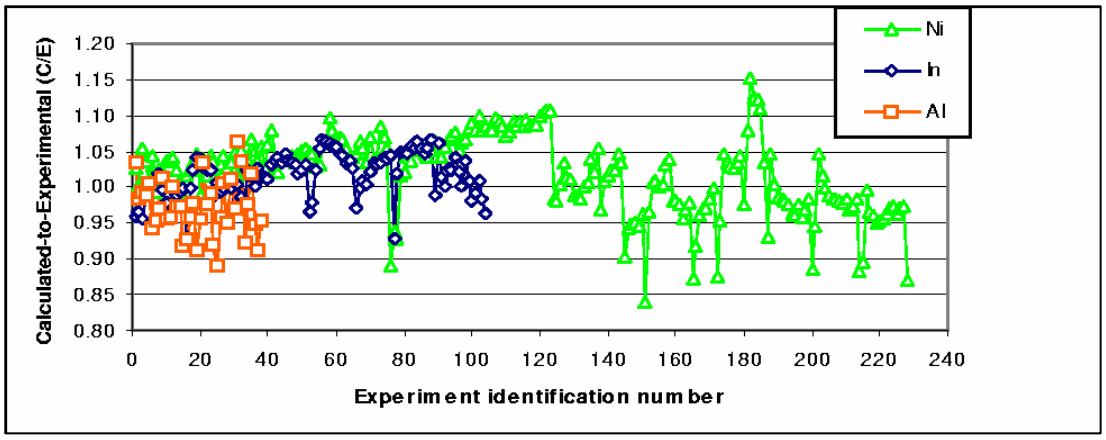


Fig. 2.11. VENUS-3 Calculation to Experiment (C/E) comparison of 370 experimentally measured neutron spectral reaction rate foils (using Ni, In, Al dosimeters) to reaction rates computed using PENTRAN (P3-S8 26 group-dependent) fluxes. Note 95% of the 370 C/E within +/-10% of experimental measurement (most were within 5%); 5% of the 370 C/E within +/-15% (associated with partial length fuel). (Haghighat, Abderrahim, and Sjoden, 2000).

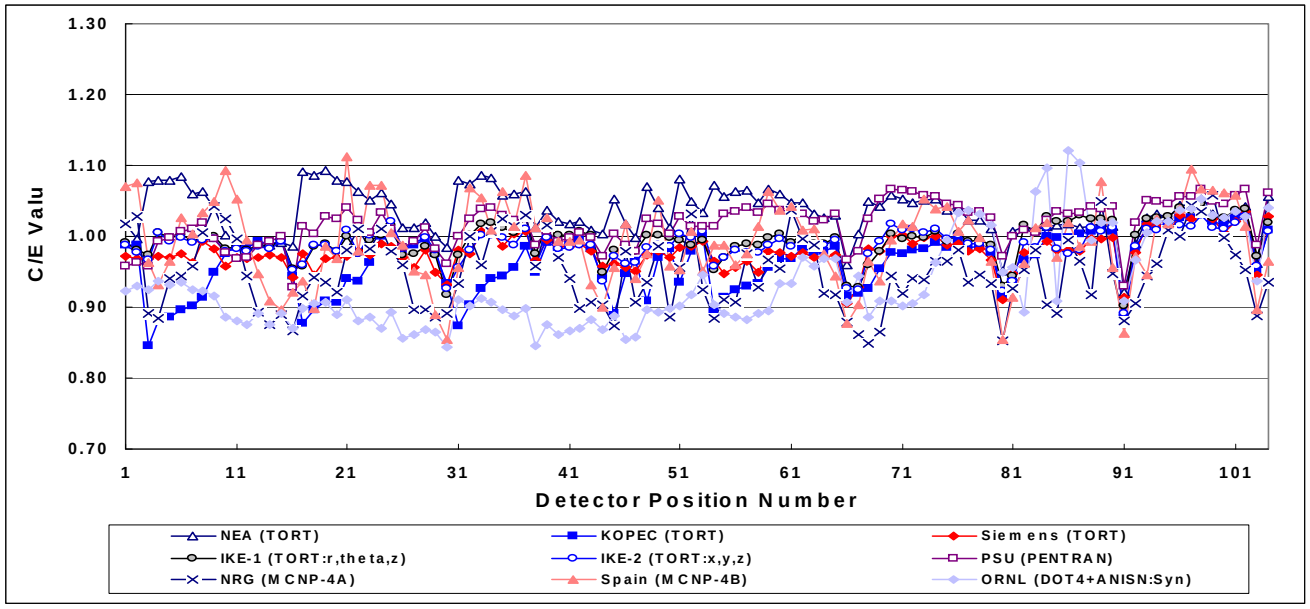


Fig. 2.12. VENUS-3 Calculation to Experiment (C/E) comparison for OECD/NEA "Venus-3" 3-D Benchmark Problem showing results for Equivalent Fission Flux for the In-115 inelastic scatter reaction; PENTRAN results cmpare well with other codes.(OECD/NEA Nuclear Science Publication: "Venus-1 and Venus-3 Benchmarks", 2000).

Participant	Code Used	Method	Problems	Mesh Width (cm)	Computer
Azmy, *et al.*	TORT	S_{16} LN[a]	Scattering cases only	10/9	Cray Y/MP 4 tasks
Konno	TORT	S_{16}	All cases	2	FUJITSU
	TORT with FNSUNCL3	S_{16} FCS[b]	All cases	2 0.25	AP3000/24
Alcouffe	PARTISN	S_8, FC[b]	All cases	2	SGI
Haghighat, *et al.*	PENTRAN	S_{20}, ADS[c]	No scattering	Variable (2-10)	IBM SP2
		S_{12}, ADS[c]	Scattering	1.111	IBM SP2
Zmijarevic	IDT	S_{16}, LC[d] Extrapolated	All cases	(10/9)	DEC Alpha 4100
Suslov, *et al.*	MCCG3D	RT[e], SC[f], DD[g]	All cases	2.5	SP2
Oliveira, *et al.*	EVENT	P_9, RT[e]	All cases	1.43-2	COMPAQ AXP 1000
Brown, *et al.*	ARDRA	P_{21}	All cases	1.04-2	IBM ASCI Blue-Pacific

[a] Linear nodal, [b] First collision source, [c] Adaptive differencing strategy using the DTW and EDW schemes, [d] Linear characteristic, [e] Ray tracing, [f] Step characteristic, [g] Diamond difference

Fig. 2.13. Kobayashi Benchmark Problem Set participants and Code methods (Kobayashi, et. al., 2000).

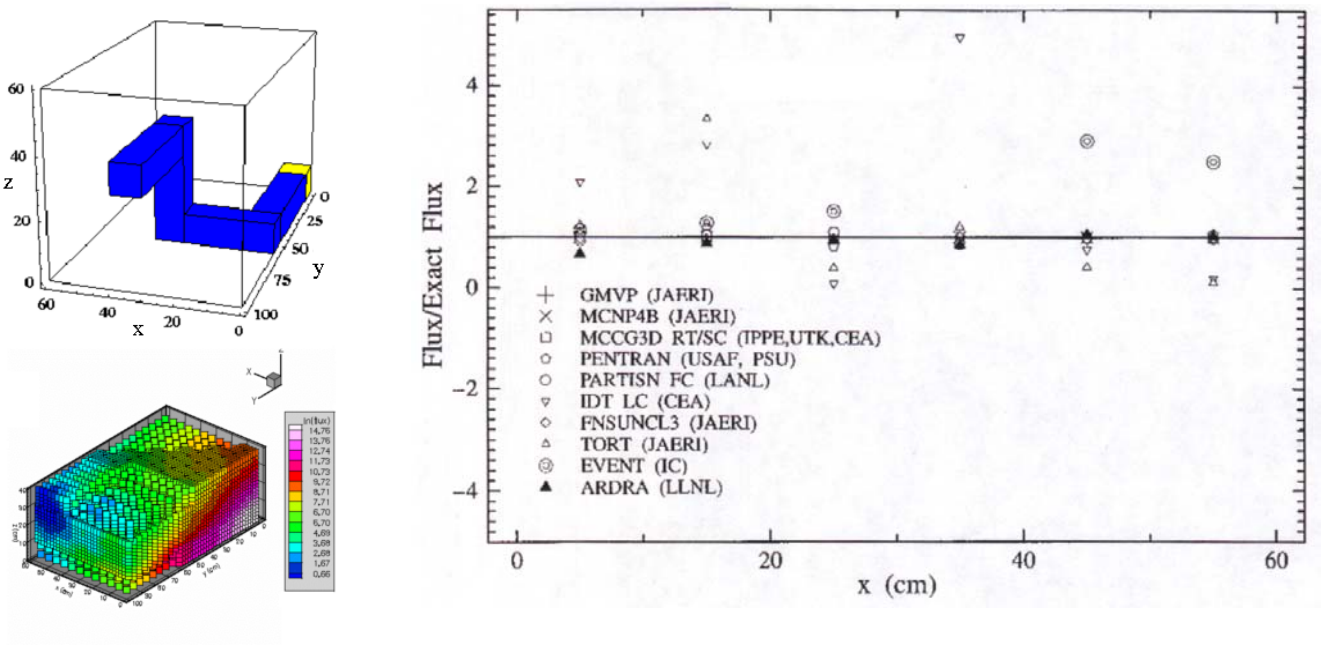


Fig. 2.14. Dog-Leg Void Duct Problem, mesh distribution/solution (Left); Calculation to Exact (C/E) comparison for AEN/OECD 3-D Kobayashi Dog-Leg Benchmark Problem with Zero Scattering at y=95 cm, z=35 cm (Right); PENTRAN was able to render an accurate problem solution without using a first collision source (see Fig 2.15.6 below), required by several others (Kobayashi, et al, 2000), and (Haghighat, Sjoden, and Kucukboayci, 2001).

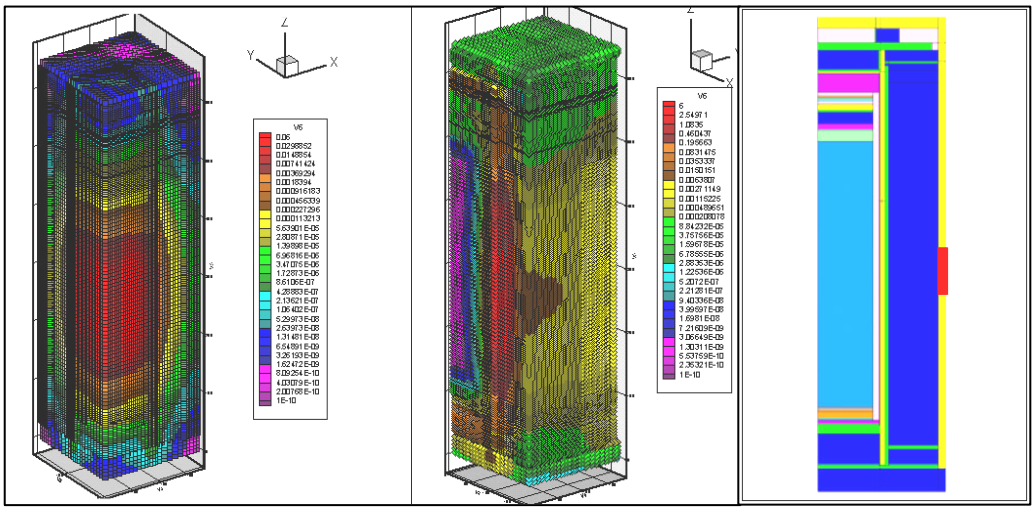


Fig. 2.15. PENTRAN™ Solution of Dry Storage Cask for Spent Fuel Storage. (Left) CASK Group 1, (Center) CASK Group 22 (thermal neutrons), and (Right) A³MCNP model of Dry Storage Cask, Height 610 cm, Shell O.D. 340 cm, Shell I.D. 187 cm, Loaded Weight 162.4 MT). This model consisted of 318,426 fine meshes solved with P3-S12 quadrature with 22 neutron and 18 gamma energy groups on 8 2.4GHz Intel Dual Xeon machines in 165 hours (Shedlock and Haghighat, 2004).

Dose Calculated Using Monte Carlo			Dose Ratio, Sn/MC	
A³MCNP Cont. Energy [(mRem/hr)/ n/cm²/s]	MCNP Cont. Energy [(mRem/hr)/ n/cm²/s]	MCNP Multigroup [(mRem/hr)/ n/cm²/s]	Multi-group	Cont. Energy
1.75E-04 (0.91%)	1.78E-04 (1.25%)	1.25E-04 (1.30%)	1.04	0.74
2.12E-04 (0.83%)	2.13E-04 (1.14%)	1.50E-04 (1.18%)	1.04	0.74
1.98E-04 (0.95%)	1.97E-04 (1.19%)	1.39E-04 (1.23%)	1.04	0.73
1.43E-04 (1.09%)	1.42E-04 (1.41%)	1.00E-04 (1.46%)	1.04	0.73

Fig. 2.16. Dry Storage Cask Dose comparison of PENTRAN™ , A³MCNP, and MCNP simulation results for Dry Storage Cask for Four Annular Segments Near Source Centerline (300 cm). Multigroup Monte Carlo and Sn results differ by only 5% (average). (Shedlock and Haghighat, 2004).

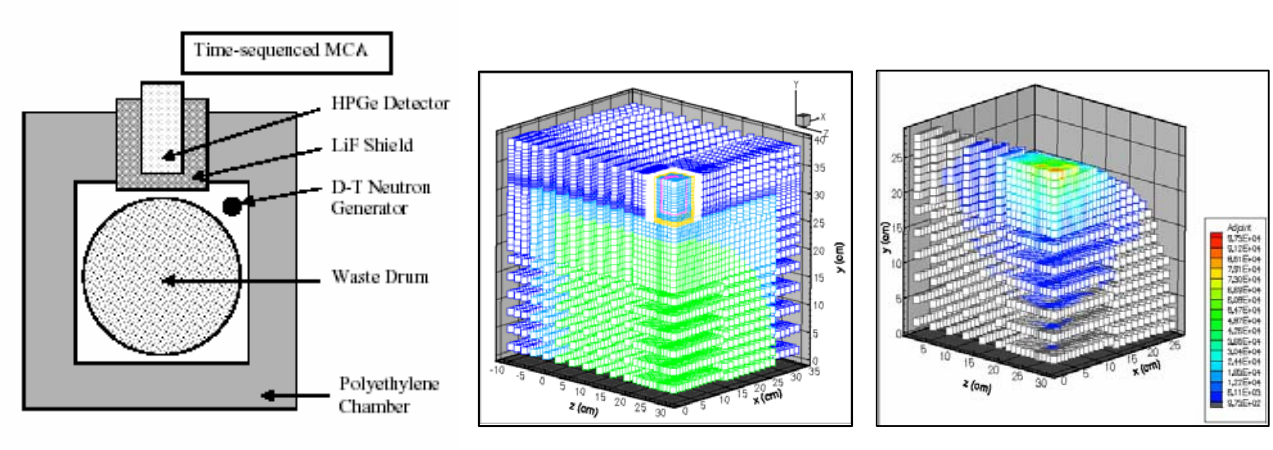


Fig. 2.17. Model of Waste Drum in Prompt Gamma-Neutron Activation Analysis (PGNAA) Device (Left); PENTRAN™ Sn Mesh distribution for waste drum in simulation for Westinghouse Corp. (Center); Adjoint function distribution for Pb gamma line at E= 1.023 MeV (Right). PENTRAN results were used to compute reaction rates and were in agreement with experimental results within experimental error (Petrovic, et. al., 1999).

Contaminant (Metal)	Max. Relative Difference (calculated-to-experiment)	Estimated Experimental Uncertainty	Low Limit of Detection (LLD)
Hg	12%	14%	9 ppm
Cd	6%	10%	115 ppm
Pb	19%	20%	4,400 ppm

Fig. 2.18. Comparison of Detector LLD for Model of Waste derived from calculated detector response using PENTRAN adjoint distributions folded with Monte Carlo MCNP n-gamma distribution with the experimental values. (Petrovic, et. al., 1999).

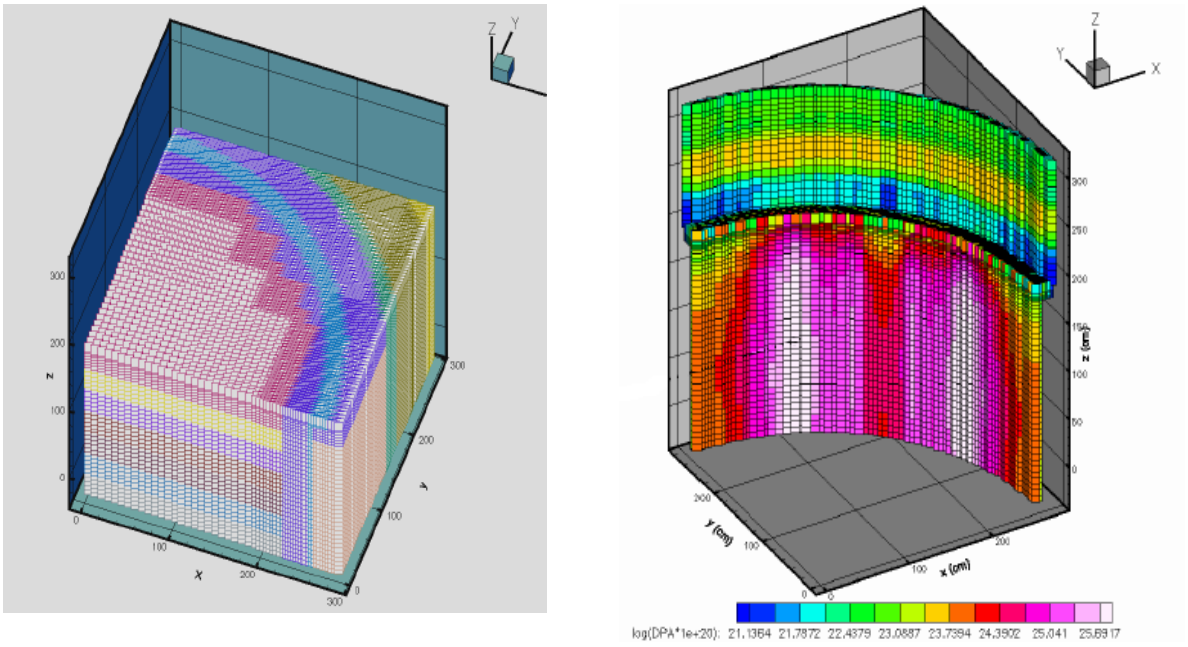


Fig. 2.19. BWR Core Shroud Problem. This is a simulation of the BWR reactor and Core Shroud Assembly, with baffles, jet pumps, steam voids, etc. (Left) The 67 Group P3-S8 coupled neutron-gamma calculation using 265,264 fine mesh cells was solved in 12 hours on 48 IBM-SP2 processors, 8 processors angular, 6 processors spatial decomposition. (Right) Displacement per Atom (DPA) in the core shroud shows intense radiation damage where fuel is close to the shroud. These results were verified independently by Monte Carlo computations; multigroup (BUGLE-96) PENTRAN values were within 5-15% of continuous energy MCNP values. (Kucukboyaci, et. al, 2000).

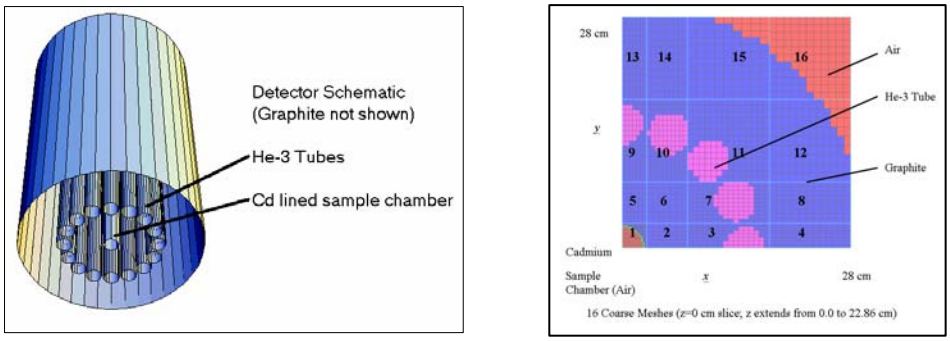


Fig. 2.20. Computational He-3 Detector Design Experience; this work focused on the determination of predicted neutron detector response accomplished using neutron importance derived from an adjoint discrete ordinates (SN) transport calculation. A hypothetical detector apparatus, intended to detect fast neutrons, was modeled using He-3 tubes with graphite moderation using the PENTRAN 3-D multi-group discrete ordinates parallel transport code system. (Left) He-3 fast neutron detector schematic with He-3 tubes positioned at 15 cm radius; (Right) PENTRAN™ mesh distribution of He-3 detector assembly at slice z=0 cm. (Sjoden, 2002).

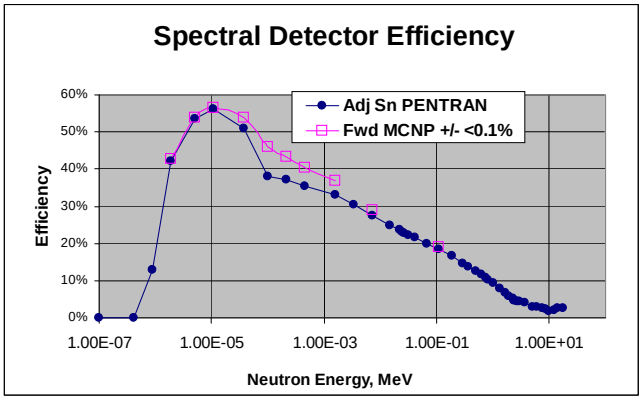


Fig. 2.21. He-3 Neutron spectrsl detector efficiency comparing mutiple MCNP & adjoint PENTRAN result. (Sjoden, 2002).

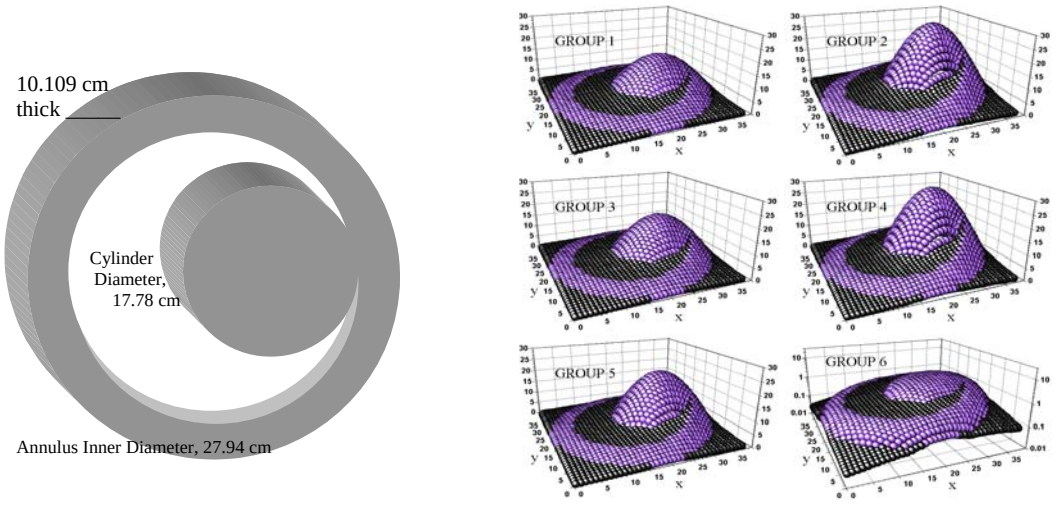


Fig. 2.22. HEU Annular Ring Criticality Benchmark Problem Geometry (Left) 6-Group calculation yielding k_{eff}=0.994, exact agreement with Multigroup MCNP (reported within +/- 0.001) (Sjoden and Haghighat, 1999).

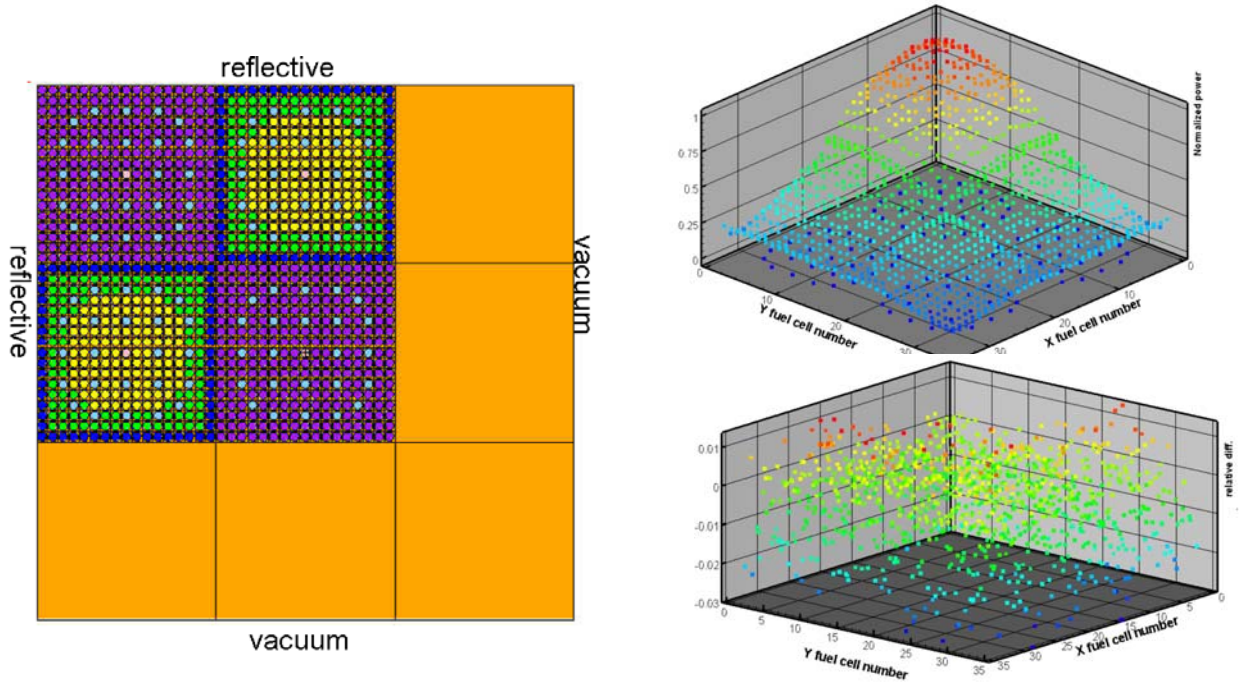


Fig. 2.23. OECD/NEA C5G7 MOX 2-D Benchmark Problem PENTRAN Mesh distribution with specific pins represented among 229,551 spatial meshes, S16quadrature (228 directions),7 groups (Left), PENTRAN power distribution with k_{eff}=1.18760, within <0.1% of MCNP (Top right), and relative power difference from MCNP (Bottom Right) average difference is 0.88%, compared to a statistical error reported in MCNP results that range from 0.4% to 1.24%. The 3-D unrodded case was also modeled with 946,080 spatial meshes and yielded k_{eff}=1.14323, within <0.09% of MCNP (Yi and Haghighat, 2004).

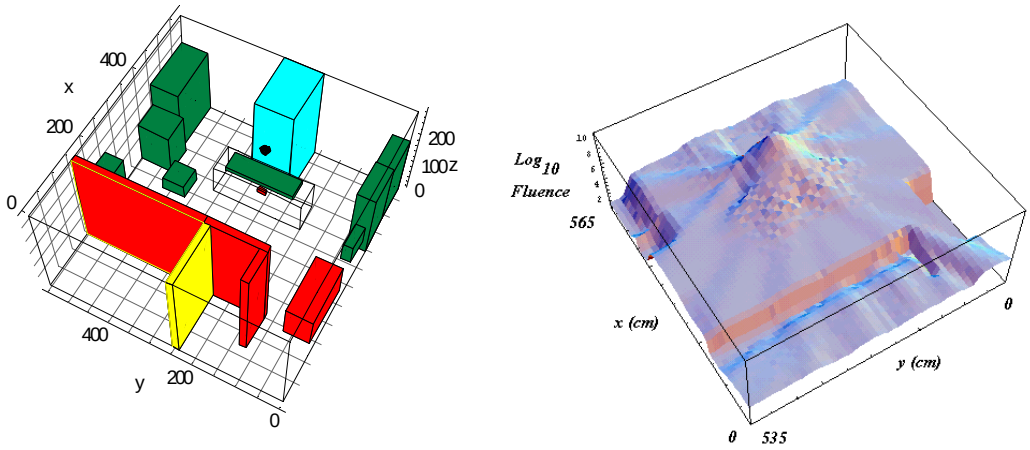


Fig. 2.24. Schematic for X-Ray Room S4 Simulation using PENTRAN (Left) X-ray dose for a 32mAs burst from an 80 keV Diagnostic imaging device (Right); in spite of some ray effects due to the low Sn order, dose results near the source were within measurement data uncertainty (Sjoden, Gilchrist, et al, 2000).

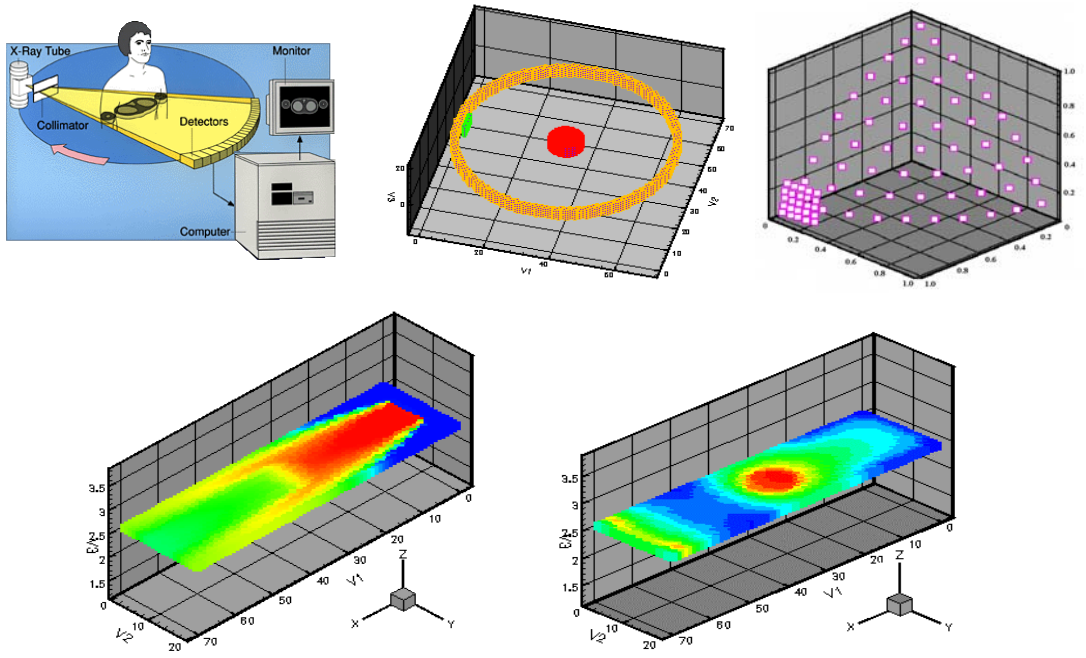


Fig. 2.25. CT Scan Simulation using PENTRAN with the built-in "Ordinate Splitting" (OS) feature to enhance a particular direction of interest and severely mitigate ray effects. The fundamental CT Scanner Method uses a fan-shaped beam and the uncollided flux is used for imaging (Top Left); the PENTRAN model (with air regions removed)showing the ring of detectors and phantom tissue body captured in (Top Center); S20 quadrature set depicted with 3 segments of OS (from 1 to 25 directions) enhancement on the ordinate near the centerline of the detectors (Top Right); High and Low energy (including down-scattered) photon fluxes (Bottom Left, Right, respectively). (Brown and Haghighat, 2000).

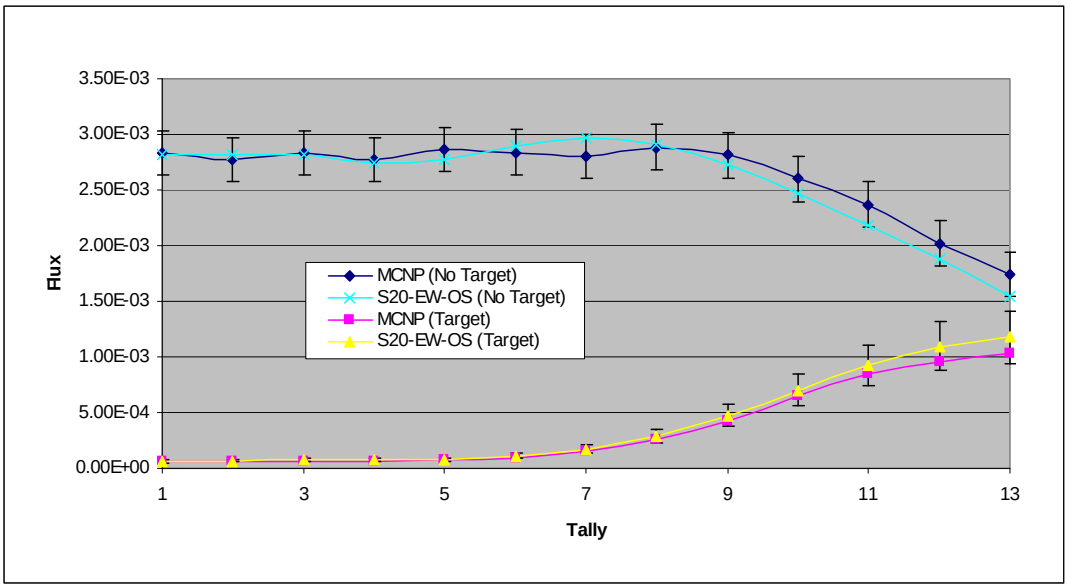


Fig. 2.26. CT Scan Simulation Results comparing PENTRAN fluxes with 3 segments of "Ordinate Splitting" (OS) and MCNP tallies (+/- 3% rel error) with, without target phantom in place. Without OS, S50 angular quadrature is required to yield equivalently accurate Sn results, highlighting the importance of the OS methodology (Brown and Haghighat, 2000).

CT Scan Timing Results	CPU Time	Memory Req.
MCNP	3000 sec	-
S20- Level Symmetric (55 dir/octant)	216 sec (1 processor, interactive)	88 Mb
S50- Equal Weight (325 dir/octant)	640 sec (20 processors) (4/1/5)	2417 Mb
S20- Equal Weight with Ordinate Splitting (79 dir/octant)	451 sec (1 processor, interactive)	204 Mb

Fig. 2.27. CT Scan Simulation Timing Results comparing PENTRAN times and specifications with MCNP Monte Carlo simulation times (+/- 3% rel error) (Brown and Haghighat, 2000).

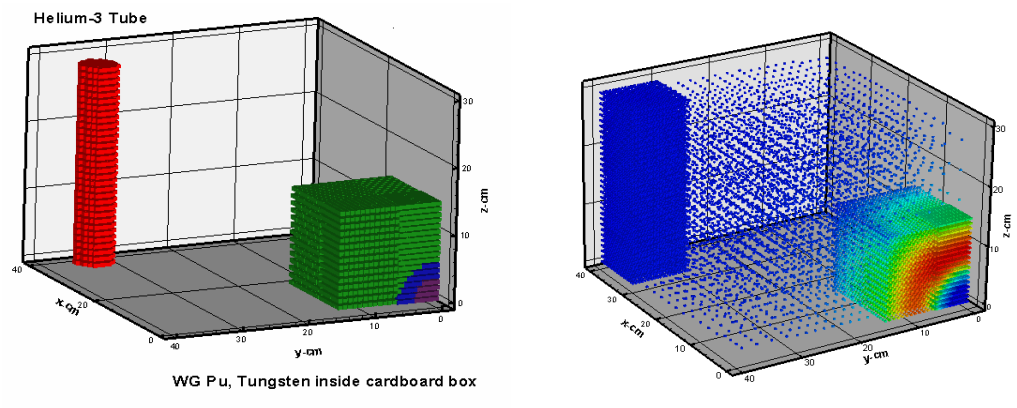


Fig. 2.28. Homeland Security "Pu-Ball-in-a-Box" Problem: (Left) Surreptitious shipment of 4.0 kg of alpha phase plutonium metal (19.8 g/cc) as an inner sphere of weapons grade plutonium surrounded by an outer spherical shell of 3 cm of tungsten metal. The ball was placed at the center of a cardboard box filled with craft packing paper, simulated in our PENTRAN™ transport model with 60% density cellulose to simulate packing paper/cardboard. The Pu is sub-critical at keff=0.843 (yielding M=5.37, ML=3.61). (Right) thermal neutron flux colored onto mesh distribution; the He-3 tube is hidden by the fine mesh of air surrounding the tube; the majority of neutrons that interact with the He-3 tube are well below 1 keV. (Sjoden, 2004).

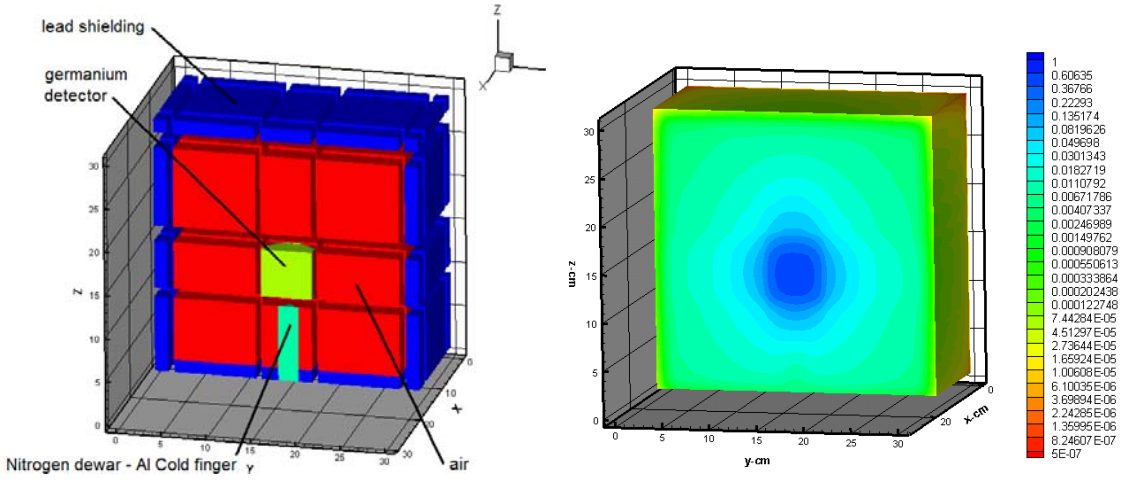


Figure 2.29. (Left) PENTRAN Cutaway/exploded coarse/fine mesh detail of germanium detector Sn model. A hyper-fine gamma energy structure isolating 90 gamma lines with Sn adjoint transport was used to profile the detector efficiency for gamma energies spanning from 50 keV to 3 MeV. Any level of detector geometric detail can be represented using PENTRAN (Sjoden and Ghita, 2008). (Right) Relative efficiency contours for 400-410 keV gamma rays in a standard Germanium detector. Values can be multiplied by 0.000154 to yield space dependent absolute efficiency before electronic processing (Sjoden and Ghita, 2008).

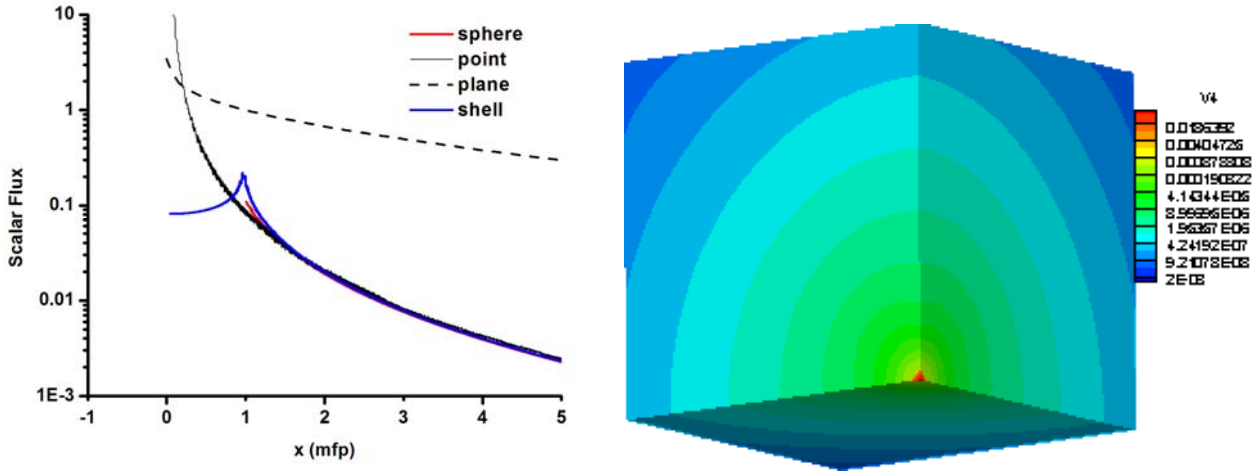


Figure 2.30. (Left) PENTRAN solutions for the case (c=0.9, g=0.9, L=12) to Ganapol's 'TIEL' Benchmark problems [Ganapol, 2006]. The goal of these challenging problems was to provide a set of analytical benchmarks in infinite media for a variety of sources (Plane, Sphere, Shell, Point) that were highly anisotropic, using a Henyey-Greenstein (H-G) scattering kernel requiring up to L=24 scattering order. Ganapol solved these through quadrature of a Fourier Transform inversion, so that these problems can unequivocally be used as standards of comparison. PENTRAN solutions were accurate to < 0.7% difference relative to the reference solutions, and verified accuracy and asymptotic convergence using 3-D Legendre-Chebychev quadratures to N=64 and 3-D anisotropic scattering up to L=24 (Ghita, Sjoden, and Ghita, 2007). (Right) Isotropic point source solution in a highly anisotropic medium Modeled as a 12 x 12 x 12 mfp box divided into 86,000 fine cells of 5 mm (each side) cells, except for center region (0.5 mm on each side in the first coarse mesh. The source cell (at the origin) was modeled as one single fine mesh cell of 0.5 mm size (Ghita, Sjoden, and Ghita, 2007).

Overall, *PENTRAN* has proven to be a robust and accurate computational tool for a wide variety of applications in nuclear science and engineering.

Several problems were profiled for parallel performance; these results are presented here. Most results are based on selected problems presented in the previous section.

			All results converged to a 1.0E-5 relative tolerance.					
Run	PENTRAN Decomposition Strategy	# Procs Angle	# Procs Group	# Procs Space	#Total Procs P	Sp= Speed- up Factor	Ep= Efficiency (Sp /P, %)	Comm Overhead: % wall-clock
1	Angular	2	1	1	2	1.9	95	5
2*	Group	1	3	1	3	2.7	90	9
3	Spatial	1	1	3	3	2.59	86	12
4	Angular	4	1	1	4	3.26	82	16
5	Angular-Group	2	3	1	6	4.78	80	22
6	Angular	8	1	1	8	4.3	54	32
7	Spatial	1	1	9	9	5.49	61	21
8	Angular-Group	4	3	1	12	6.72	56	42
9	Angular-Spatial	4	1	3	12	5.76	48	43
10	Angular	16	1	1	16	6.14	38	53
11	Angular-Spatial	2	1	9	18	8.74	49	32
12	Angular-Group-Spatial	2	3	3	18	8.74	49	34
13	Angular-Group	8	3	1	24	7.96	33	64
14	Angular-Spatial	8	1	3	24	6.56	27	53
15	Spatial	1	1	27	27	8.74	32	37
* Average of 2-batch and interactive runs at CNSF								

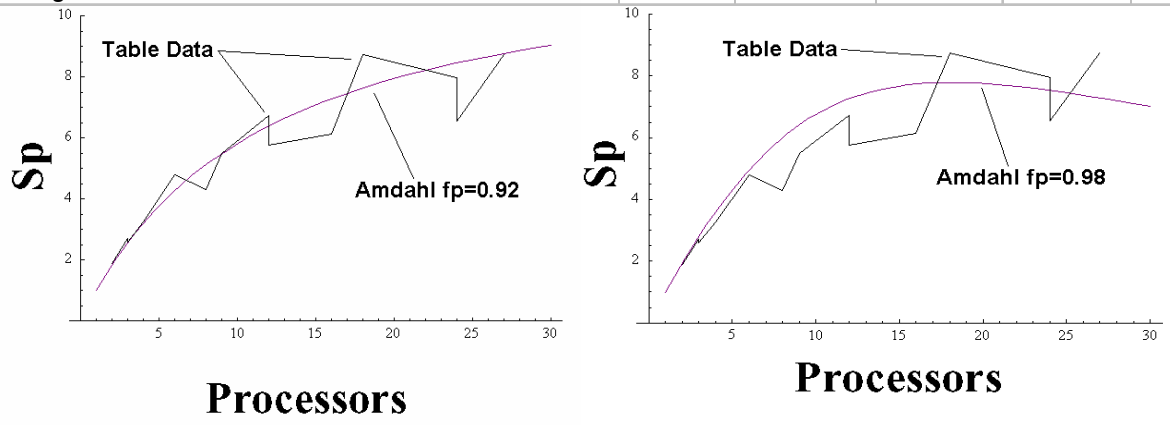


Fig. 2.31. PENTRAN Parallel Scalability Tests for a simple "Box in a Box" Sn simulation with upscattering cross sections. Results depict various decomposition methods tested, and the speedup curves depict speedup projected with an Amdahl performance curve (See Section 2.1), assuming a worst case zero communication overhead (Left) and then assuming an estimated linear scaling of communications overhead ratio of Tc/Ts=0.072 on 24 processors (Right). These tests reveal that the parallel fraction for PENTRAN depends on the problem and decomposition, and is in the range of 0.92 to 0.98. Other testing on larger problems yielded parallel fractions estimated at 0.975 (Sjoden and Haghighat, 1996) and (Sjoden, 1997).

VENUS-3 *PENTRAN* Parallel Performance study

Case	No. of processors	Domain Decompostion Algorithm (A/G/S)[1]	Wall-Clock time (min)[2]	Speedup	Efficiency (%)
1	4	4/1/1	551.8	1.00	-
2	8	8/1/1	311.9	1.77	88
3	16	8/1/2	153.3	3.60	90
4	32	8/1/4	84.3	6.54	82

[1]*(A/G/S) refers to the number of angular, group, and spatial subdomains*

[2]*Time is obtained in a BATCH mode*

Fig. 2.32. PENTRAN Parallel Scalability Tests for the Venus-3 OECD/NEA Benchmark Problem (Haghighat, Abderrahim, and Sjoden, 2000).

BWR Reactor Simulation, *PENTRAN* Scalability study

Case	No. of Directions	No. of Processors	Domain Decomposition (A/G/S)[1]	Wall-Clock time per iteration (s)
1	24	6	1/1/6	30.12
2	48	12	1/1/12	33.28
3	80	24	4/1/6	29.52
4	169	48	8/1/6	36.12

[1](A/G/S) refers to the number of angular, group, and spatial subdomains

Fig. 2.33. *PENTRAN* Parallel Scalability Tests for the BWR Reactor Problem (Kucukboyaci, et. al, 2000).

Fuel Storage Cask Simulation*

Model	# CPU	Dose Ratio to Reference	Run Time (hrs)	Speedup	Values/ Hour
Unbiased Multigroup	8	0.70	362	1.0	19
PENTRAN 'Large' Model	8	0.74	165	2.2	42,100
PENTRAN 'Small' Model	8	0.74	123	2.9	35,000

**Parallel wall-clock times, speedup, and calculated values per hour of two PENTRAN storage cask model calculations as compared to Monte Carlo MCNP calculations*

Fig. 2.34. *PENTRAN* Parallel Run comparisons with computations needed in Monte Carlo for the Spent Fuel Cask Problem. Note that Monte Carlo simulation yielded tallies at a small percentage of locations compared to the complete solution of the phase space rendered using *PENTRAN* (Shedlock and Haghighat, 2004).

3. *PENTRAN* INPUT

PENTRAN INPUT PROCESSING

All input to the *PENTRAN* code is performed using an input file read and processed by each independent processor. The filename is always input by the user on processor 1, and if the execution is parallel, processor 1 can be set to broadcast the input filename to all other processors (if this feature is enabled at compile time). Parallel input is reported to be the fastest means of initializing problem data on distributed memory machines, and typically involves less overhead than processing the input data on one process with message passing data to all other processors (Gerner, 1995).

For best results, a 'ppen' parallel *PENTRAN* script should be used, the name of the input deck in parallel is converted to 'prb.pen'; the parallel job script (see the *Appendix*) permits the user to use a unique filename, and the script manipulates file names to 'prb.pen' and back again to the user's name seamlessly). With the ppen script, parallel execution on a problem called *testproblem.f90* with 8 processors of machine 'mycluster' is accomplished with:

mycluster/home/user > ppen *testproblem f90 8*

Note that a single blank space should separate each field, and the "period" ('.') separating the file prefix and suffix is NOT used. If a period is inserted, the script will not perform as desired; also, the location of the parallel executable code, assumed to be reachable by all processors running MPI, should be set as indicated in the 'ppen' script.

To accomplish parallel *FIDO* data input on each processor that is completely consistent with *FORTRAN* character/numeric data restrictions, a small (typically 5-50 kb) scratch file (denoted with a '*.dat*' suffix) is created. To avoid file I/O conflicts in a distributed file system, a uniquely named scratch file is generated and used independently by each processor; the scratch file name (*fileprefix(1:5)+ processor# +*'.dat') is based on the processor number. (A uniquely *named* scratch file is necessary because a *FORTRAN* compiler may create a default *un-named* scratch file, 'fort.2,' for all processes, even during parallel execution). Although independently naming scratch files may not be necessary on all clustered workstation file systems, the current treatment prevents having to handle system-unique implementations, and prevents "fixes" that in some cases might violate strict *ANSI FORTRAN* programming. These *FIDO* scratch ('*.dat'*) files are only used for *FIDO* input processing and communicator minimization, and can be deleted after problem execution.

Similarly, another set of independent scratch files is created for material specification in each coarse mesh. All material scratch files ('*.mat'*) can be deleted following execution. On processor 1, the material file (containing data identical to other '*.mat*' files) is intended for archive and is stored using an input file *prefix*+'.M1'.

The input deck for *PENTRAN* is designed in a block structure, emulating the block structure used in other deterministic code packages (O'Dell, et al, 1995). After the code parameters for establishing the memory on each processor are read in, subsequent input for each block is performed via the so-called free-field *FIDO* inidcators, which includes control characters avialable for abbreviated input syntax; these are discussed in more dtail in the next section.

It is true that most of the issues surrounding the determination of correct parameter settings and *FIDO* commands are eliminated if using the *PENMSH-XP* automated mesh and input generation code (covered in a separate code manual, developed at the University of Florida). However, should one wish to understand memory allocation issues, or tune an exisitng deck to implement a different decomposition, some discussion is necessary. Overall, the input deck is composed of :

```
PENTRAN CODE PARAMETERS FOR THIS PROBLEM
maxmem,  maxpcs,  maxgcm,  maxxsg
 2000       8        4        0
maxcmc,  maxcrs,  maxmmc,  maxmed,  maxfmc,  maxfin
   2        2      100      100      100      100
maxgrp,  maxglc,  maxswp,  maxqdm,  maxmat,  maxleg
   2        1        4       10        1        3
maxsrc,  maxslc,  maxcmr,  maxlin,  maxarr,  nctlim
   4        2        4      204     40000      20
----------------------Start Problem Deck-----------------------
Problem Header card
Title Card 1
        :
Title Card 10
Block 1: General Problem Parameters  T
Block 2: Geometry  T
Block 3: Cross Section Parameters T
(Optional) Cross Sections (T required if xsecs within input deck)
Block 4: Control Options  T
Block 5: Sources  T
Block 6: Boundary Conditions  T
Block 7: Print Controls T
```

Fig. 3.1. Outline of PENTRAN Parameters/Block Input; in this example, the parameters indicate that 8 processors will be used with decomposition among angular (2 processors), energy(2 processors), and spatial cells (2 processors).

The input deck portion reserved for the parameter cards falls between the first line of each input deck, beginning with "*PENTRAN* CODE PARAMETERS FOR THIS PROBLEM." The parameter portion ends with with "----------Start Problem Deck----------". In reality, the labels are "dummy" lines and can contain ANY ASCII phrase, as are the lines that contain the names of the parameter labels that serve as a guide for the user. More detail on parameters and what they mean are presented in the next section.

In 1997, *PENTRAN* was modified to allow for adjustable-sized arrays according to *ANSI FORTRAN-90 standards*, which avoids the issue of code recompilations for specific numbers of processors, etc, and permits more efficient use of dynamic memory. The following is an example of a parameter card set that should appear *at the very top of each PENTRAN input deck.* The user must enter the code parameters in a prescribed order for each parameter. Note that above each card image of integers is a text "dummy" comment line naming the parameters below it.

Note that the "Code Parameters" section at the very top of the problem input is the *only* part of the input deck that will *not* utilize FIDO notation. To Automatically have *PENTRAN* correct the Parameters specified for the parallel run, see the discussion in "Automated Parameter Repair."

Table 3.1 Sample of Code Parameters at Top of Input Deck:

```
 PENTRAN CODE PARAMETERS FOR THIS PROBLEM
maxmem,  maxpcs,  maxgcm,  maxxsg
 2000       8        4        0
maxcmc,  maxcrs,  maxmmc,  maxmed,  maxfmc,  maxfin
   2        2      100      100      100      100
maxgrp,  maxglc,  maxswp,  maxqdm,  maxmat,  maxleg
   2        1        4       10        1        3
maxsrc,  maxslc,  maxcmr,  maxlin,  maxarr,  nctlim
   4        2        4      204     40000      20
-----------------Start Problem Deck-------------------
```

Desciption of Each Code Memory Parameter

```
maxmem  Max Per-Process-allocated Memory,Mb
maxpcs  Max Processors Allowed for problem
maxgcm  Max Global Coarse Meshes in problem
maxxsg  Max Restart Total Groups used
maxcmc  Max Locally stored Coarse Meshes
maxcrs  Max Coarse Meshes along any axis
maxmmc  Max Local Medium Meshes/Coarse mesh
maxmed  Max Medium Meshes along any axis
maxfmc  Max Local Fine Meshes/Coarse mesh
maxfin  Max Fine Meshes along an axis
maxgrp  Max Global number of Energy Groups
```

```
maxglc  Max Locally stored Energy Groups
maxswp  Max Locally stored Sweep Octants
maxqdm  Max Quadratures Angles/Octant
maxmat  Max Number of Material (xsec) Types
maxleg  Max Legendre Scattering (xsec)Order
maxsrc  Max Global number of Fixed Sources
maxslc  Max Locally stored Fixed Sources
maxcmr  Max No. of Contiguous CMR/PCR Cells
maxlin  Max Number of Lines in Input Deck
maxarr  Max Entries in Any Vector Variable
nctlim  Max Number of FIDO Chars/Variable
```

Note the following details are important to parameter settings, particularly when there are references to global or local variables. Global parameters refer to dimensions across the entire problem, and local variables refer to parameter settings the depend on parallel processing decomposition. Again, note that there is an Automated Parameter Repair feature in *PENTRAN* that will correct these settings for the user if the parallel decomposition is defined with the decomposition weighting vector.

- **In general, all parameters should be set no higher than necessary to minimize overhead.**

- [+]**maxmem** may be set beyond the physical per-processor memory, but allocation past the physical memory will result in disk swap, which will ruin parallel performance; it is recommended that more processors be allocated rather than exceed physical memory.

- [+]**maxpcs** should not be set higher than the number of processors being used for the calculation, as this parameter sets up memory buffers (along with other parameter data) that enable parallel communication among all machines; setting this too high will allocate more memory than is required for parallel message passing, resulting in wasted resources and longer message passing packets.

- **maxgcm** is the number of global coarse meshes in the problem handled among the aggregate of all processors.

- **maxxsg** can normally be set to zero. It is only used to allocate memory to handle restart cross sections in a restart procedure to continue a fixed source transport problem. A restart procedure run enables one to perform a fixed source calculation in partial steps, where blocks of groups of downscattered radiation can be computed in separate runs. If using a limited size parallel machine, the restart feature enables one to perform a fixed source calculation of a very large problem with subsets of energy groups executed in separate steps (see the section describing 'Restart').

- [+]**maxcmc** is set to be the local number of course mesh cells handled on each processor. This should be less than **maxgcm** if spatial decomposition is used.

- **maxcrs** sets the dimension for the maximum stride along any one axis (e.g. stride along x, y, or z) for the number of coarse meshes.

- **maxmmc** should always be set to **maxfmc.**

- **maxmed** sets the dimension for the maximum stride along any one axis for the medium mesh interval (e.g. stride along x, y, or z) in a coarse mesh.

- **maxfmc** is the upper limit of the maximum total number of fine mesh cells contained in any one coarse mesh, considering coarse mesh cells over the entire problem.

- **maxfin** sets the dimension for the maximum stride along any one axis for the fine mesh interval (e.g. stride along x, y, or z) in a coarse mesh.

- **maxgrp** is the maximum number of energy groups handled in the problem.

- [+]**maxglc** is the maximum number of energy groups locally computed by each processor. For example, if using group decomposition for a 4 group problem (**maxgrp**=4), and executing group decomposition on 4 processors, then set **maxglc**=1.

Note that if using the group window, maxglc should be the number of groups handled in the window, typically 1 group.

- [+]**maxswp** is the maximum dimension for the number of sweep octants locally stored on each machine. If the run is a serial one, this number must be 8. Note that if using 2 processors in angular decomposition, then this number would be 4, since (2 * 4 = 8).

- **maxqdm** determines the maximum dimension of octant angles based on the order of the quadrature, where the maximum number of angles per octant **maxqdm**=(N(N+2)/8) for a given S_N order; e.g. S_8 requires that **maxqdm**=10.

- **maxmat** determines the dimension of the maximum number of materials, and therefore the number of cross sections (macroscopic) read into the problem.

- **maxleg** determines the maximum dimension for the number of Legendre moments that are to be used in the cross section set

- **maxsrc** determines the maximum dimension for the total number of fixed sources that are to be established in the problem. One source per coarse mesh cell can be defined.

- [+]**maxslc** determines the maximum local dimension for the number of fixed sources that are to be stored locally in the problem. Without spatial decomposition, based on unit numbers of coarse mesh cells, then **maxslc** = **maxsrc.** However, with spatial decomposition, it is possible that the number of local sources need not be as large as **maxsrc**. Again, one source type per coarse mesh cell can be defined.

- [+]**maxcmr** sets the parameter dimension for the number of coarse meshes contained in the largest size zone for coarse mesh partial current rebalance acceleration.

- [+]**maxlin** sets the dimension for the maximum number of lines in an input deck. This should be set at least as large as the number of lines in the *PENTRAN* input deck.

- **maxarr** sets the dimension for the largest array that will be encountered, considering any field in the input deck

- **nctlim** sets the dimension for the maximum number of FIDO characters that will be encountered , considering any field in the input deck.

*The PENMSH-XP code will include the necessary proper parameter settings in constructing the input file (the 'input deck') for a 3-D model. For optimum performance, precise parameter settings for parallel exzecution must be made by the user, or adjusted using 'Automatic Parameter Repair' for parallel hybrid options using angular, energy, or spatial

decomposition. The settings that typically require adjustment after the input deck is generated by PENMSH-XP are flagged in the above descriptions using '+'.

AUTOMATIC PARAMETER REPAIR

To enable *PENTRAN* to correctly set the F90 parameters at the top of the code deck:

- Step 1: Change the number of Mb of memory under **maxmem** from a positive integer to a negative integer
- Step II: set the decmpv vector using all negative numbers (locking in the intended decomposition). NOTE: Automatic parameter repair will not be fully effective unless each entry in decmpv is < 0.
- Example:
    - After setting the number of maxmem in the top of the parameter section from 2048 to -2048 Mb, change the decomposition vector (**decmpv** in Block I) to indicate a negative number of processors for each potential parallel decomposition, e.g. **decmpv=-2 -1 -3** which will indicate 2 processors for angular decomposition, one processor for group decomposition (no group decomposition), and 3 processors for spatial decomposition, with a total no. of processors as ABS((-2)(-1)(-3))=6.
    - The negative **maxmem** value signals the code to perform a parameter correction based on the problem's settings read in.
    - Then the code will automatically set up the correct parameters to minimize memory allocation as follows: the total number of processors **maxpcs**=6, the maximum sweeps per processor **maxswp**=4, **maxglc**=1 if group window option, etc, and **maxcmc** will be corrected for the maximum local number of spatial coarse meshes on each processor.
    - NOTE: While the Automatic Parameter Repair feature is reasonably robust, the parameters **maxarr** and **maxlin** must be acceptable values (large enough to accommodate reading in the input deck /FIDO arrays) in order for the code to properly analyze the input deck.

FLUX MOMENT PRECONDITIONING

PENTRAN now incorporates a new option that allows the user to precondition the 0^{th} and 1^{st} flux moments in the initial S_N source iteration sweep using an effective initial guess to the polar angle flux moments to provide acceleration to the S_N source iteration. To invoke this procedure, the preconditioned flux moment files must be present in the local execution directory, and must be set as indicated below. **WARNING: if present, these files will be used regardless of the problem if they are present in the local NFS directory:**

- Filename "precflx1a0" is the Group 1 0^{th} moments in ASCII for the first ~½ of the coarse meshes in the problem, free format as follows:

```
0.958276          <- k-effective (if fixed source problem, enter 1.0 here)
1       9         <- Starting, Ending Coarse Mesh (CM) numbers (1 to 9) for this
file
1    1125         <- Coarse Mesh number 1, 1125 fine meshes in this CM
8.50036           <- fine mesh #1 flux moment CM#1
8.44759           <- fine mesh #2 flux moment CM#1
...
9.76354           <- fine mesh #1125 flux moment, CM#1
2     342         <- Coarse Mesh number 2, 342 fine meshes in this CM
7.43762           <- fine mesh #1 flux moment CM#2
6.97540           <- fine mesh #2 flux moment CM#2
...
```

- Filename "precflx1b0" is the Group 1 0^{th} moments in ASCII for the last ~½ of the coarse meshes in the problem, free format as follows:

```
0.958276          <- k-effective (if fixed source problem, enter 1.0 here)
10      18        <- Starting, Ending Coarse Mesh (CM) numbers (10 to 18) for this
file
10     725        <- Coarse Mesh number 10, 725 fine meshes in this CM
20.0036           <- fine mesh #1 flux moment CM#10
18.4476           <- fine mesh #2 flux moment CM#10
...
19.6354           <- fine mesh #725 flux moment, CM#10
11     650        <- Coarse Mesh number 11, 650 fine meshes in this CM
7.43762           <- fine mesh #1 flux moment CM#11
6.97540           <- fine mesh #2 flux moment CM#11
```

- Other 0^{th} Moment Filenames:

  "precflx2a0" is the Group 2 0^{th} moments in ASCII, similar format as "precflx1a0"
  "precflx2b0" is the Group 2 0^{th} moments in ASCII, similar format as "precflx1a0"
  . . .

- Other optional 1^{st} Moment Filenames:
  "precflx1a1" is the Group 1 1^{st} moments in ASCII, similar format as "precflx1a0"
  "precflx1b1" is the Group 1 1^{st} moments in ASCII, similar format as "precflx1a0"
  . . .

Note these files can be automatically generated from a previous transport run using the *REPRO* tool, based on data extraction selection from *PENDATA*. Executing *REPRO after PENDATA* then processes output files automatically to yield preconditioned solution files ('precflxXXX') where, if present in the local directory of execution, will be used by *PENTRAN* in subsequent runs to yield a significant iterative acceleration.

LARGE PROBLEM RESTART PROCEDURE

This procedure was developed to allow one to perform an extremely large computation by breaking up energy groups in downscatter only fixed source problems.

In the Restart procedure, flux moments are read as results from previous batch groups, and are used to calculate the transfer scattering source inside the code with restart for the new group batch.

A new batch of groups will start with the medium grid if the simplified multigrid option is on, although there is only a fine grid transfer source (qinscf). In such cases, the 'closest mesh' approach is used by projecting the fine grid source back onto medium grid from the input fluxes.

The cross sections are read in for the global total number of groups to be computed based on the **maxxsg** parameter, since this parameter is set to the ceiling of the number of groups to be considered among **all** multiple restart calculations.

Restart Example: In a 20 group (total) calculation, if there are ten groups (Group 1 to Group 10) in the **first restart batch, then the** binary flux moment files from the initial calculation must be renamed and locally available (see below), and the next 10 energy groups (Groups 11 to 20) require calculation.

- **This requires all 20 groups to be loaded in from the cross section file for the second batch**, and therefore the maxxsg parameter must be set as **maxxsg=20**.
- One may use ngroup=10, 1, 10 to indicate 10 groups starting at group 11 and proceeding through group 20, 1 group in the window, with groups 1 to 10 loaded as restart from binary files.
- Note that the **same parallel decomposition and number of processors must be used** to allow the binary files to be read by the correct processor.
- **Depending on the processor ID#, the binary flux moments must be named as: input filename + '.r' + processor#.** These restart files have the same format as dumped flux moments that are stored as input filename + '.f' + processor#--these must be renamed for a restart run by replacing the 'f' with an 'r.'
- The **combfm** routine can be used to combine flux moment files for multiple pass restarts. Note: binary files can be very large for large 3-D problems, especially so for P_3 or higher calculations.

FIDO INPUT CONTROL CHARACTERS

The *FIDO* syntax was originally implemented in several codes at the national laboratories. The following *general rules* apply for an input deck:

- 1 to 79 column line (card) images, no "special columns," no special ordering, data delimiters are a blank space or comma

- All entries following a slash (/) are ignored, and Input is made according to the following syntax: *varname=number1 number2…*

- Named variables must be input *followed immediately by an "=" and the first entry in the array field, with no spaces between the "=" and the first array element*.

- *Any* spacing option can be used for *subsequent array elements* (with or without commas, intermixed slashes, etc). No spaces or commas should be used immediately following a *FIDO* control character (see table).

- No distinction is made between *real* and *integer* data, although *real entries used for integer fields are automatically rounded*. Scientific notation can use upper or lower case "e" for the exponent.

- A Block Terminator " T " is required (case sensitive) at the end of each data block (**do not use "t"**). The block terminator should immediately follow one or more spaces after (but always on the same line as) the last field data entry.

Summary of *FIDO* Array Control Characters

Control Character	*Syntax*	*Description*
R-Repeat	nRd	Repeat data d n times
I-Interpolate	nId d+1	Interpolate n items between d,d+1
C-sCale	nCd	Scale n previous items by d
F-Fill[+]	Fd	Fill the remainder of array with item d
Y-string repeat[+]	nYm	Repeat m strings n times
L-Log Interpolate	nLd d+1	Log Interpolate n items between d,d+1
Z-Zero	nZ	enter Zero n times
S-Skip[++]	nS	Skip the next n data items
A-pointer set	nA	set pointer to nth data item in array
Q-Q repeat	nQm	repeat the last m entries n times
G-G repeat	nGm	same as Q but change sign every repeat
N-N repeat	nNm	same as Q but invert order every repeat
M-M repeat	nMm	same as N but change sign every repeat
X-check entries	nX	check the number of entries against n
&-string skip[+]	&	skips to the end of the string

[+] Identified but not Currently supported in *PENTRAN*

[++] Data items that contain *FIDO* control characters count as a single entry

TROUBLESHOOTING INPUT READ ERRORS

The following are typical issues that can lead to errors when *PENTRAN* is reading an input deck across all processors:

- *dos2unix:* When moving between Windows and UNIX/LINUX platforms, dos2unix or unix2dos commends in LINUX should be executed as appropriate. Failure do do this can commonly cause input read failure and or unexplained input errors

- The number of lines in any input deck, including all comments, is **maxlin** lines (e.g. *maxlin=1000).* Note: the *PENMSH-XP* code will provide a recommended upper limit, and these will be repaired by the code if the value is exceeded.

- The number of *FIDO* control characters in a given array field is typically set to a maximum of the parameter **nctlim** (e.g. *nctlim=200*). The user should increase the number of *FIDO* characters to be read per array field as needed using the **nctlim** parameter. Note: the *PENMSH-XP* code will provide a recommended upper limit, and these will be repaired by the code if the value is exceeded.

- *Exceeding column 79:* Read errors may be encountered if input is specified beyond the 79 column limit in the input deck.

- *Inserting spaces after the block 'T':* On some computers, we have observed that if a block appears to be correct but still encounters read errors/fatal errors, the user should add one (or more) spaces following the 'T' block terminator.

- *Interpreted FIDO:* how the *FIDO* processes the input is reported in a logfile (designated with an 'L#', where the '#' is a processor number) generated on one or more processors, depending upon the 'loglevel' setting indicated in the problem header. If fatal errors are encountered, the cause should be identified in the logfile. (See the *loglevel #* phrase/option in the Header card of an input deck).

- *Check the previous Block:* Block errors encoutered in a block that seem completely correct may in fact be due to an improper or multiple parameter entry in the *previous block or previous variable*.

The problem header card comes immediately after the input parameter field, is required, and is a maximum of 79 columns, primarily serving as a problem label.

The header card includes control options for a special purpose logfile and output file control when a "loglevel" string is present, as described below.

- Logfiles display *processed FIDO* input, as well as all warning and error messages, as-read cross section data, problem decomposition information, rebalance acceleration matrix solution data, processor iteration progress, and a process **Decomposition Mapping Table** for all processes, cells, groups, and angles.

- The **Decomposition Mapping Table** provides the user with an outline of where parallel
- output data is located, again depending on the automatic processor assignment.

- **Disabling all logfiles is *not* recommended.** A logfile can, depending on the optional "loglevel" string located somewhere in the header card, be generated on one or all processes; see the table below.

At least one logfile is required for parallel data post processing using PENDATA

Summary of *loglevel* string appearing in Header Card on Log/Output Files

loglevel string	*Effect*
loglevel 0	*Disables* all logfiles.
No loglevel	Same as loglevel 1 ***"Default"***
loglevel 1	*Enables* logfile on ***processor 1 only***
loglevel 2	*Enables* logfiles on ***all processors***
loglevel 3	*Enables* logfile *and* prints CMR-PCR/SR factors, xsec reads on ***processor 1 only***
loglevel 4	*Enables* logfile *and* prints CMR-PCR/SR factors, xsec reads on ***all processors***
loglevel all	*Enables* logfiles, CMR-PCR/SR factors, xsec reads, *and* data output, ***all processors***

Additional Notes on 'loglevel' settings:

- Iteration progress will **only** be provided by a processor with an active logfile during execution, echoed to the terminal and written to the logfile.

- A **loglevel 4** string *forces **all** processors to print a log file* with CMR(PCR)/SR results (if rebalancing is activated).

- A **loglevel all** string is similar to **loglevel 4**, but *also forces **all** processors to print an output file*.

- *CAUTION:* The user should exercise caution with these settings when scaling to a large number of processors, as 3-D file outputs may be voluminous (especially with complete angular decomposition), possibly saturating all available disk space.

- Each log file is saved using the prefix of the input filename + '.L' + processor#; each output file is saved using the input filename prefix+'.'+processor#. For example: on processor 1 using input file *test.pen*, the logfile would be saved under *test.L1*, and the output file would be saved as *test.1.*

- More discussion on output is made under Block 7 (print options).

- On most parallel systems, processor numbers (alias task or rank numbers) are numbered starting from zero. In the *PENTRAN* code, all processors are assumed to be numbered from 1 to n. Therefore, all machine rank identifications (e.g. resulting from an MPI_COMM_RANK call to identify the process rank) are shifted by +1 for *reporting purposes*. When referring to the process rank *within* MPI commands for actual message passing, reported ranks are converted back to machine ranks by adding -1).

(10) Title Cards. There are 10 title cards; these are REQUIRED, 1-79 columns each, which can be used for problem description and documentation. If unused, these must REMAIN as the first 10 lines immediately following the *header card*.

4. *PENTRAN* INPUT BLOCKS

Below the parameter list, there are a total of *seven* required input blocks in a *PENTRAN* input deck.

- Block 1: includes general problem input parameters; all fields are required for code execution, and can be defined *in any order*. Twelve possible parameters in this block include: **ngeom, ngroup, isn, nmatl, ixcrs, jycrs, kzcrs, decmpv, lodbal, timcut, tolmgd, modadj.**

- Block 2: includes problem geometry input parameters; all fields are required for code execution, and can be defined in any order. Twelve possible parameters in this block include: **xmesh, ixmed, ixfine, ymesh, jymed, jyfine, zmesh, kzmed, kzfine, nmattp, flxini, mathmg.**

- Block 3: includes problem macroscopic material cross section parameters; all fields are required for code execution, and can be defined in any order. Ten possible parameters in this block include: **lib, legord, legoxs, nxtyp, ihm, iht, his, ihng, chig, nxcmnt.**

- Block 4: includes execution control options. Fields can be defined in any order. Twelve possible parameters in this block include: **nprtyp, nrdblk, tolin, tolout, maxitr, methit, methac, ncoupl, ndmeth, nzonrb, dtwmxw, nquit.**

- Block 5 includes source definition options. Fields can be defined in any order. Eleven possible parameters in this block include: **nsdef, nscmsh, ssnrm, sref, serg, smag, spacpf, omegap, scalsf, rkdef, kextrp.**

- Block 6 includes boundary conditions. All input fields are required and can be defined in any order. Six possible parameters in this block include: **ibback, ibfrnt, jbeast, jbwest, kbsout, kbnort.**

- Block 7 includes print controls. All input fields are optional and can be defined in any order. No specification for any print controls sets maximum printing. Ten possible parameters in this block include: **nxspr, ngeopr, nsumpr, meshpr, nfdump, nsrcpr, nsdump, nmatpr, nadump**, and **njdump.**

Summary of Block 1 Inputs

Variable	Decription	Syntax Example
ngeom	Geometry identifier	ngeom=3d
ngroup	Energy groups in calc/window/above restart	ngroup=*Grps, Window_Grps, Restart_Grps*
isn	Order of 3-D quadrature	isn=*Sn_order, #Dirs_OS, OS_indices, OS_Segments*
nmatl	Number of materials in problem	nmatl=*Number_of_Materials*
ixcrs	Number of Coarse x mesh zones	ixcrs=*Number_of_CoarseX*
jycrs	Number of Coarse y mesh zones	jycrs=*Number_of_CoarseY*
kzcrs	Number of Coarse z mesh zones	kzcrs=*Number_of_CoarseZ*
decmpv	Decomposition vector	decmpv=*Angular_weight*,
lodbal	Automatic load balancing	lodbal=0 (off) or =1 (on)
timcut	Wall-Clock time cutoff limit, minutes	timcut=*Wall-clock_minutes*
tolmgd	Multigrid tolerance variable	tolmgd=*value*
modadj	Adjoint transport mode	modadj=0 (forward) or =1 (adjoint)

Termination of the Block with a T is required

- The **ngeom** variable, at present, must always be =3d.

- The **isn** variable is a vector. The first number is the Sn order, where if isn(1)>0, theis uses level symmetric, and isn(1)<0, this uses P_n-T_n. The remaining elements of the vector **isn** are directly related to omega splitting, which splits an ordinate into multiple directions surrounding the original ordinate with a redistribution of the individual ordinate weight; this is used to mitigate ray effects. The second number (isn(2)) is the number of directions desired for "Omega Splitting" (OS), followed next by the specific ordinate numbers (referenced from the (+,+,+) octant, based on the number of total ordinates in each octant) selected for splitting. Final entries are the segment orders corresponding to each OS ordinate, respectively. The segment order….

    - Example: isn=-8, 2, 2 3, 2 3 is interpreted by *PENTRAN in order of appearance* as
        - -8 indicates S8 (<0) P_n-T_n based quadratures (80 directions)
        - 2 indicates 2 directions are intended for ordinate splitting
        - 2 3 indicates ordinate numbers 2 and 3 will be split
        - 2 3 indicated that the ordinates will be subdivided with segmentation levels 2 and 3, respectively

This results in a total of 42 directions per octant, where of the normal 10 directions per octant for S8, ordinate #2 is split into 9 equal-weight ordinates (2 segments), and ordinate #3 is split into 25 equal weight segments as follows in Fig 3.4.1:

```
S8 3D Pn-Tn Weighted  Angular Quadrature
 Number of Omegas per Octant  :  42
 ABS Minimum Direction Cosine :  .130158469080925

        (+,+,+) Direction Cosines(in sampling order):

 Omega     w (weight)      mu (x-axis)     eta (y-axis)      xi (z-axis)
 ----- ---------------- ---------------- ---------------- ----------------
    1  0.012653566896915 .197285875678062 .197285875678062 .960289835929871
    2  0.001544312806800 .506257951259613 .153571680188179 .848598062992096
    3  0.001544312806800 .488767504692078 .202454149723053 .848598062992096
    4 …
 … 9  0.001544312806800 .623374402523041 .258210152387619 .738059580326080
   10  0.001544312806800 .595063686370850 .318068206310272 .738059580326080
   11  0.000555952603463 .251446872949600 .435518771409988 .864348232746124
   12  0.000555952603463 .222424209117889 .451031684875488 .864348232746124
   13 …
 …33  0.000555952603463 .266709595918655 .643893957138062 .717123806476593
   34  0.000555952603463 .224025875329971 .659958958625793 .717123806476593
   35  0.000555952603463 .180382817983627 .673197925090790 .717123806476593
   36  0.013071110472083 .821784138679504 .220196396112442 .525532424449921
   37  0.013071110472083 .601587772369385 .601587772369385 .525532424449921
   38  0.013071110472083 .220196425914764 .821784138679504 .525532424449921
   39  0.011333867907524 .964143216609955 .191780015826225 .183434635400772
   40  0.011333867907524 .817361176013947 .546143293380737 .183434635400772
   41  0.011333867907524 .546143233776093 .817361235618591 .183434635400772
   42  0.011333867907524 .191779926419258 .964143276214600 .183434635400772

             Omega Sampling Order: Sn Angles

          S8                xi
        (+ + +) Octant       1
        m-Level Diagram    2   3
          (Sweep 2)      4   5   6
                       7   8   9  10
                     mu            eta


             Omega Splitting Segments:

                       x--x--x
           9:1         |  |  |
         Equal weight  x- 2--x  Ordinate:  2,  2 Segment(s)
          Splitting    |  |  |
                       x--x--x

                       x--x--x--x--x
                       |  |  |  |  |
                       x--x--x--x--x
             25:1      |  |  |  |  |
         Equal weight  x--x- 3--x--x   Ordinate:  3,  3 Segment(s)
          Splitting    |  |  |  |  |
                       x--x--x--x--x
                       |  |  |  |  |
                       x--x--x--x--x


       GENERAL OCTANT Sweeping Assignments

  Sweep   mu   eta   xi   Start  Sweep   mu   eta   xi   Start
  ----- ----- ----- ----- -----  ----- ----- ----- ----- -----
    1     -     -     -    FWN     2     +     +     +    BES
    3     -     -     +    FWS     4     +     +     -    BEN
    5     +     -     -    BWN     6     -     +     +    FES
    7     +     -     +    BWS     8     -     +     -    FEN
```

Fig 3.4.1: Example for isn card with ordinate splitting outlined in detail.

• The maximum allowed *level symmetric* quadrature is S_{20} (**isn**=20). Higher order quadratures using P_n-T_n (Legendre-Chebychev) are permitted from S_2 to arbitrary order (provided there is adequate memory). To select P_n-T_n quadratures, use an Sn order < 0; for example, S_{20} P_n-T_n is accessed using (isn=-20). Note: the *maxqdm* parameter must be set to **isn*(isn+2)/8** with no OS.

• **Arbitrary Quadrature Set.** If the file "**quadset.pen**" exists in the local problem execution directory, then the quadrature set contained in this file will **be read to replace any quadrature specified for the isn order indicated.** Note that for the isn order specified for the problem, one must match the number of entries in the first octant (+,+,+) in the file. A warning message will be generated by the code in the logfile if this occurs: "WARNING: File quadset.pen detected: Over-riding (+,+,+) quadrature set." The user should take care to eliminate the "quadset.pen" file unless specifically intending to use it as a quadrature set. If more ordinates are required, the isn value should be increased until enough ordinates can be represented, entering zeros for those ordinates with no weight in the calculation.

  • **Example:** set isn=8. This requires **maxqdm=10.** Therefore, ten ordinates representing the (+,+,+) octant should be entered.

  • Then the free field format of the file "quadset.pen" for the example is as follows: ordinate number, normalized weight (w_m), μ_m, η_m, ξ_m, as shown below. Note there is no header or comment line:

```
    1  0.012653566896915 .197285875678062 .197285875678062 .960289835929871
    2  0.001544312806800 .506257951259613 .153571680188179 .848598062992096
    3  0.001544312806800 .488767504692078 .202454149723053 .848598062992096
    4  0.001544312806800 .466569960117340 .249386876821518 .848598062992096
    5  0.001544312806800 .578393042087555 .175453618168831 .796666502952576
    6  0.001544312806800 .558410465717316 .231301203370094 .796666502952576
    7  0.001544312806800 .533050060272217 .284921228885651 .796666502952576
    8  0.001544312806800 .645681738853455 .195865422487259 .738059580326080
    9  0.001544312806800 .623374402523041 .258210152387619 .738059580326080
   10  0.001544312806800 .595063686370850 .318068206310272 .738059580326080
```

Fig 3.4.2 Contents of a sample file "**quadset.pen**" for isn=8

• The **ixcrs, jycrs, kzcrs** variables are the number of coarse mesh cells projected along the x, y, and z axes, respectively, which yield **(ixcrs*jycrs*kzcrs)** total number of coarse mesh cells.

• The **decmpv** "decomposition vector" allows the user to specify the priority at which problem decomposition occurs for each variable, optimizing decomposition (within the restrictions of the assigned weights) during parallel execution. The input is an ordered triple of weight factors.

The following special rules apply to **decmpv** weight factors:

decmpv rule i. A **negative value** over-rides any optimization and assigns the absolute value of the negative weight as the number of processes for decomposition in that variable; if

this number exceeds the number of available processors, scaling is as detailed in (decmpv rule iii) below.

decmpv rule ii. A **zero value** blocks any parallel decomposition scaling for that variable.

decmpv rule iii. A **positive value** scales normally based on the decmpv weighting vector. See the procedure described below. If **no clear variable has decomposition priority**, then the priority follows angular, group, and then space.

- In effect, **decmpv** defines the user desired aspect ratio of the 3-D processor array for the problem. During execution, these weights are then compared with the *actual number* of S_N directions (angles), energy groups, coarse mesh cells required, and *number of parallel processors* executing.

- Decomposition procedure using **decmpv:**
  - Consider a 3-D virtual rectangular parallelepiped processor array, of dimensions *A*G*S*.
  - *A* processors are devoted to angular decomposition, *G* processors are devoted to group decomposition, and *S* processors are devoted to spatial (coarse mesh level only) decomposition, with a total number of processors totalling (*A*G*S*).
  - The **decmpv** vector allows one to prioritize which variable is decomposed first, second, and third with then systematically performed by *PENTRAN based on the weighting system imposed by the user* with **decmpv**.
  - It is usually best to lock in a set number of processors for one variable, and then let *PENTRAN* scale the other variables. For example, if a problem has 80 directions (**isn**=8), 2 energy groups, and 4 coarse meshes, one decmpv strategy could be decmpv=-2 1 0.5.
  - Calling for 8 processors at execution, 2 processors would be locked in for angular decomposition, followed by maximizing group decomposition to 2 processors, followed by maximizing spatial decomposition with the remaining pool of processors.
  - Note that in this case, the decomposition priority is allocated to energy groups, since it carries the largest (most positive) weighting factor. *PENTRAN* will attempt to autoscale to an assigned number of processors (as in the above example) to a problem that is consistent with a user's specified weighting vector.
- Also a consideration in assigning decmpv weights is that *PENTRAN* always breaks the angles up into sweep octants, with a subsequent number of directions omega per octant. This will affect the way angular decomposition is applied for a given number of processors, as there are ***two*** levels of decomposition that follow processor assignments in angular decomposition: there are always 8 octants, and *(isn*(isn+2)/8)* directions $\hat{\Omega}$

per octant. Therefore, octants should be decomposed using an evenly numbered processor assignment (see the next paragraph).

- The **decmpv** weights must be chosen carefully by the user, as setting a decomposition vector component too high will block possible scaling of useful processor work to another decomposition variable. **If the processor utilization in each decomposition is less than 100%, execution halts. An integer number of processes must evenly divide so that integer portions can be evenly distributed on allocated processors at execution time. This is imposed due to the possibility of MPI communication synchronization failures among specific processors that are assigned odd numbers of angles, groups, or coarse mesh cells.**

- **ngroup** is actually a 3-element vector variable which fundamentally declares the group structure of the calculation. The first field is total number of energy groups required in the current calculation. The second and third fields are both optional, and *only used for downscatter fixed source problems*. Criticality computations or fixed source calculations with upscatter *cannot* take advantage of the group window or restart.

  - ngroup(1) indicates the total number of energy groups required in the current calculation.

  - ngroup(2) indicates the group width (span) of the group window. The group window is a minimum of one (1) group wide. A window of one group therefore uses only a single memory location through which to cycle all energy groups, rather than saving angular fluxes in each locally computed group. Group parallel can be performed on the span of the group window, if desired.

  - ngroup(3) indicates the **total number** of **restart groups** converged from a previous run; note that this requires binary flux moment files to be available--see Restart notes below). If positions (2) and (3) are not set, then position (2) is set to the number of groups in ngroup(1), and position (3) is set to zero.

- A wall-clock time cutoff via the **timcut** variable insures a "safe" problem stop and data dump after the specified wall clock time is exceeded (in minutes). *If **timcut=0**, no cutoff time is activated*. The **timcut** option gaurantees that the user will have final results if the execution time is limited for any reason. Execution is halted after all processors signal a "wall time exceeded" synchronization, only after an iteration is complete, whereupon data is dumped. Note that since no intermediate data dumps are performed to maximize parallel efficiency, this feature provides a "safety net" if wall clock times are limited in a batch queue structure. This is also another means of job control, in addition to solution tolerances and maximum iterations (Block 4).

  - Example: timcut=40 will force the code to cease computations, dump output files, and halt gracefully, provided there is enough time allocated to the queue. Recommendation: set timcut to cease execution 10 minutes before queue time limit (if running on a timed job queue).

- Load balancing is handled by the code automatically if selected by the user. (The automatic load balancing option is selectable via the **lodbal=1** switch in Block 1). For example, if a coarse mesh contains significantly more medium/fine meshes relative to other coarse meshes, a sequential coarse mesh assignment may lead to an imbalanced processor workload; one processor effectively drags down the rest while it finishes calculations, and others sit idle waiting for message completion. With load balancing, the workload for each coarse mesh is evaluated and ranked, and cells with the densest medium/fine meshing and/or highest within group scattering ratio are paired with cells having the sparsest meshing and/or lowest scattering ratio. Coarse mesh cells and/or energy groups are then reracked according the workload in each, thus attempting to force each processor to carry the same computational load. While this spreads work (more) evenly, there are drawbacks:

1. An important disadvantage of automatic load balancing is that it may limit the effectiveness of the automatic red-black coloring feature (Block 4), as it is possible only "red" cells are assigned to a processor based on the computational load.

2. The rerack of cells may also inhibit problem convergence with regard to angular flux sweep progression.

- The **tolmgd** variable defines the grid structure to be used in solving the transport problem. **tolmgd<0** indicates the fine grid **only** is used in the iteration sequence. **tolmgd=0** indicates the medium grid **only** is used in the iteration sequence. Note that the **mathmg** array sets how materials on the medium grid are used, either using a "closest approach" or homogenized by volume assignment (see Block 2). **tolmgd>0** indicates a simplified multigrid sequence is used, where a solution is converged on the medium mesh grid to a relative local tolerance equal to **tolmgd**, whereupon the values are projected to the fine grid (where medium grid values are overwritten with projected fine grid values to conserve memory).

- Because converged medium grid values are projected and overwritten, there is no return to the medium grid –hence, the "simplified" or "slash" multigrid algorithm. The user is free to determine the tolerance for **tolmgd** and all grid structures. However, if too small a tolerance is chosen, the time spent converging to the less accurate medium mesh may outweigh the benefit of using two grids. If the **tolmgd** tolerance is set too loosely, the maximum benefit of using the multigrid acceleration will not be realized to provide an effective pre-conditioning of the fine grid values. Experience has shown that **tolmgd** should fall somewhere between 20 and 200 times the fine grid tolerance, but should also be no greater than the truncation error of the medium grid.

- Since coarser grids require more iterations to converge (in spite of being fewer in number), the difference between grids should not vary significantly from a factor of two in any one direction. If the problem is a criticality problem, the outer iteration tolerance used for the medium grid is implicitly determined from (**tolmgd/tolin*tolout**), but is also restricted from being any greater than **tolmgd**. Note that no differencing is based on the coarse mesh grids.

• Based on testing already performed, multigrid performance is enhanced when the Hiromoto-Wienke source iteration scheme (**methit=2**) is selected, especially in the case of criticality problems. See Block 4.

• To solve adjoint transport problems, the procedure has been somewhat automated in *PENTRAN*. By setting **modadj=1**, forward cross sections are reversed internally, with full automatic transposition of the scattering matrix, with $\nu\sigma_{f\,g}$ and χ_g also transposed internally. However, the user must recognize that:

1. Group G is reported as Group 1
2. Group 1 is reported as Group G (groups are reported in reverse order)
3. Directions $\hat{\Omega}$ are implicitly $-\hat{\Omega}$
4. A Group G *Adjoint* Source is input/reported as a Group 1 Source
5. A Group 1 Adjoint Source is input/reported as a Group G source

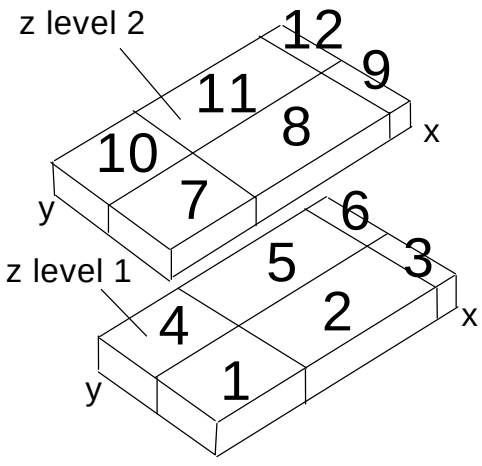
• A warning message is printed in each output file describing the effect of the adjoint calculation on the output data. Essentially, as long as the user has properly defined the source, the adjoint calculation proceeds automatically; the user must take heed of the warning message in analyzing the adjoint function output.

Summary of Block 2 Inputs

Variable	Decription	Syntax Example
xmesh	coarse x mesh bdys	xmesh=*Coarse_xBdy_1*,
ixmed	medium x mesh intervals in coarse mesh	ixmed=*medium_x_in_Coarse_Cell_1*..
ixfine	fine x mesh intervals in coarse mesh	ixfine=*fine_x_in_Coarse_Cell_1*, ...
ymesh	coarse y mesh bdys	ymesh=*Coarse_yBdy_1*,
jymed	medium y mesh intervals in coarse ymesh	jymed=*medium_y_in_Coarse_Cell_1*..
jyfine	fine y mesh intervals in coarse mesh	jyfine=*fine_y_in_Coarse_Cell_1*, ...
zmesh	coarse z mesh bdys	zmesh=*Coarse_zBdy_1*,
kzmed	medium z mesh intervals in coarse mesh	kzmed=*medium_z_in_Coarse_Cell_1*..
kzfine	fine z mesh intervals in coarse mesh	kzfine=*fine_z_in_Coarse_Cell_1*...
nmattp	nmattp card for each coarse mesh	nmattp=*coarse mesh #, material#_in_fine-mesh_1*..
flxini	initial scalar flux in each coarse mesh	flxini=*initial_flux_in_Coarse_Cell_1*..
mathmg	sets material/homogenization for med grid	mathmg=*Setting_in_Coarse_Cell_1*.. 0: use closest matl; 1: use homog matl

Termination of the Block with a T is required

- **All three dimensional cells are numbered sequentially** according to a (z(y(x))) integral loop count, where coarse cells are numbered by proceeding along all x, incrementing y, proceeding until the limit of x and y cells are reached, then incrementing z, and so on.
- **This numbering scheme is also used throughout the hierarchy of cells, where medium and fine cells within each coarse mesh follow this mapping scheme.** For example, consider that ixcrs=3, jycrs=2, kzcrs=2, or 12 total cells. Coarse cell numbers progress along x, then y, then z from 1 to 12, as shown in Fig 3.3.



Cell Numbering Progression (z(y(x)))

- Benefits of the sequential numbering scheme, **used for all grid hierarchies** (coarse, medium, and fine grids) are: (1) it allows for a more compact, sequential dimensioning of spatial arrays, reducing the span of arrays in memory; (2) looping through spatial variables uses a one dimensional index; and (3) surrounding cells are easily identified using forward and reverse translation mapping functions, enabling efficient problem setup.
- Medium (fine) mesh intervals along each

axis (e.g. *x, y*, and *z* in **ixmed, jymed, kzmed**, … respectively) must be assigned for each coarse mesh number, in order, using the prescribed mesh numbering scheme. Again, *Taylor Projection Mesh Coupling* (*TPMC*) is used to couple transport sweeps across cell surface interfaces where medium (fine) meshes may be discontinuous.

- An **nmattp** card must be present for each coarse mesh number. The first entry in each **nmattp** card is the coarse mesh cell number, followed by the material number assigned for each fine mesh contained in the coarse mesh cell using the order of the standard sequential mesh numbering scheme (proceeding along all x, incrementing y, …). The material number must directly correspond to the rank order of the material in the cross section input deck. Material numbers must not exceed the number of materials (**nmatl)** defined in Block 1.

- The **flxini** array, if non-zero for each coarse mesh, is energy group-weighted by **chig** (in Block 3), where **chig** is the fission spectrum group probability distribution variable. The **flxini** values are weighted by **chig** *even if there is no fission present*. Note: fission is *only* computed if the problem type (**nprtyp** in Block 4) is set for a criticality eigenvalue problem. If a problem is solely a fixed source problem, any fission cross sections read in.

- The **mathmg** array sets up how materials are treated on the medium grid, and requires a value for each coarse mesh number, assigned according to the sequential numbering scheme. If set to zero for a coarse mesh cell, this indicates that the material defined on the medium grid shall use the "closest approaching material" defined on the fine grid as a medium grid material specification. If set to unity, the cross sections on the medium grid are homogenized by volume based on the fine grid material specification for that coarse mesh cell. If material boundaries can be approximately resolved, the closest approaching material setting may perform best.

- A geometry file is always generated (unless deactivated in Block 7) as *fileprefix*+'.geo'. This file contains a *Mathematica*™ graphics command deck to automatically render a full color, 3-D geometry image of the problem. This is useful for viewing/verifying problem geometries. Read the *filename*.geo file into *Mathematica*™ directly as a text file, or cut and paste portions of it from a text editor. Using *Mathematica*™, coarse meshes can be made invisible for viewing other coarse meshes buried within the problem structure. See the cover of this manual for examples. *Mathematica*™ is available from Wolfram Research, Inc for Unix-x stations, PC-Windows, and Mac operating systems. A very powerful workstation may be required to render a suitable 3-D image for very large problems.

- Translations between sequential cell numbers and **ixcrs**, **jycrs**, and **kzcrs** coordinate position are made by calling the CELMAP and CIVMAP routines in *PENTRAN*.

Summary of Block 3 Inputs

Variable	Decription	Syntax Example
lib	cross section library location	lib=cards *(input card images from input deck)*
legord	Legendre scattering order	legord=*Scattering_order* (0,1,3,5, etc, < legoxs)
legoxs	Legendre scattering order of cross sections to be read	legoxs=*Cross section scattering_order* (0,1,3,5, etc)
nxtyp	cross section type (See Notes)	nxtyp=*xsec_type* (0,1,2,3,4,5,6,7, or 8)
ihm	number of rows of xsec data	ihm=*position_of_last_row*
iht	total xsec row position	iht=*total_xsec_row_position*
ihs	within group scatter xsec	ihs=*g->g_xsec_row_position*
ihng	position of last neutron scatter xsec prior to coupled gamma xsecs	ihng=*last_neutron_xsec_row_position*
chig	group fission probabilities for each material (1,2,...), by (group...)	chig= *mat1_Grp1_prob, mat1_Grp2_prob, ...*
nxcmnt	number of cross section comment cards	nxcmnt=*number_of_79_col_cardsin xsec entries*

Termination of the Block with a T is required

- The library input **lib** can be from a datafile or on card images in the input deck.

- Cross sections read directly from the input deck should immediately follow Block 3, and must be terminated with a Block "T" terminator on the same line as, and immediately following the last cross section entry.

- Zero fields should be used to "pad" null values in the scattering matrix, as applicable.

- All filenames assume a path with an 8 character filename prefix, followed by a 3 character (maximum) filename suffix ('.xxx').

- The Legendre order **legord** must be no greater than **isn**-1 (from Block 1) to properly integrate Legendre expansions using level-symmetric quadratures.

- The Legendre scattering order of the cross sections **legoxs** can equal or exceed the the **isn** value, as long as the scattering order called for in computations is less than **isn**.

- Both **legord** and **legoxs** must be zero or odd (to be able to represent peaked scattering situations with odd moment expansions). Violating these rules results in an execution halt.

- The cross section type parameter **nxtyp** allows for 9 different cross section formats (0-8).

- The format refers to row or column input, ASCII or BINARY data types, and whether or not the cross sections include the (2*l*+1) normalization factor for Legendre moments.

- The available formats are listed as follows:

  - 0: STANDARD (row) form: NO, Legendre consts NOT pre-multiplied
  - 1: STANDARD (row) form: YES, Legendre consts ARE pre-multiplied
  - 2: NON-STD (col) form: NO, Legendre consts NOT pre-multiplied
  - 3: NON-STD (col) form: YES, Legendre consts ARE pre-multiplied
  - 4: STANDARD (row) BINARY FILE form: NO, Legendre consts NOT pre-multiplied
  - 5: STANDARD (row) BINARY FILE form: YES, Legendre consts ARE pre-multiplied
  - 6: NON-STD (col) BINARY FILE form: NO, Legendre consts NOT pre-multiplied
  - 7: NON-STD (col) BINARY FILE form: YES, Legendre consts ARE pre-multiplied
  - 8: GIP-ORNL BINARY FILE form: YES, Legendre consts ARE pre-multiplied

- There is no difference between ASCII and BINARY file read *formats* in the standard row and non-standard column form; BINARY data is assumed to be stored in the same relative order as the ASCII form. The GIP-ORNL format assumes that a binary file, generated by the GIP program for mixing materials, generated the cross section file. (The GIP-ORNL format reads blocks of data for all materials and Legendre orders by energy group, which differs from the STANDARD and NON-STANDARD formats).

- To be compatible with binary file formats, all cross sections are input and stored as single precision.

- Examples of the above file formats for a typical 7-group set of cross sections are below. In all cases, the **iht**, **ihs**, and **ihm** parameters are indicated, as applicable. Since there are no gamma groups, ihng=0; this parameter is in place as an indicator. Note the structure of each is repeated for every scattering moment.

  - STANDARD FORMAT (rows) With UP *and* DOWN Scatter: iht=3, ihs=10, ihm=16

```
...nxcmnt comment cards (check file for compliance if lib=file:filename) ...

siga1 rnsigf1 sigt1 sig7->1 sig6->1 ...sig2->1 sig1->1    0       0       0...
siga2 rnsigf2 sigt2    0    sig7->2 ...sig3->2 sig2->2 sig1->2    0       0...
siga3 rnsigf3 sigt3    0       0    ...sig4->3 sig3->3 sig2->3 sig1->3    0...
siga4 rnsigf4 sigt4    0       0    ...sig5->4 sig4->4 sig3->4 sig2->4 sig1->4...
siga5 rnsigf5 sigt5    0       0    ...sig6->5 sig5->5 sig4->5 sig3->5 sig2->5...
siga6 rnsigf6 sigt6    0       0    ...sig7->6 sig6->6 sig5->6 sig4->6 sig3->6...
siga7 rnsigf7 sigt7    0       0    ...   0    sig7->7 sig6->7 sig5->7 sig4->7...
```

➢ STANDARD FORMAT (rows) With DOWN Scatter only: iht=3, ihs=4, ihm=10

```
 ...nxcmnt comment cards (check file for compliance if lib=file:filename) ...

 siga1 rnsigf1 sigt1 sig1->1    0       0       0       0       0      0
 siga2 rnsigf2 sigt2 sig2->2 sig1->2    0       0       0       0      0
 siga3 rnsigf3 sigt3 sig3->3 sig2->3 sig1->3    0       0       0      0
 siga4 rnsigf4 sigt4 sig4->4 sig3->4 sig2->4 sig1->4    0       0      0
 siga5 rnsigf5 sigt5 sig5->5 sig4->5 sig3->5 sig2->5 sig1->5    0      0
 siga6 rnsigf6 sigt6 sig6->6 sig5->6 sig4->6 sig3->6 sig2->6 sig1->6   0
 siga7 rnsigf7 sigt7 sig7->7 sig6->7 sig5->7 sig4->7 sig3->7 sig2->7 sig1->7
```

➢ NON-STANDARD FORMAT With UP and DOWN Scatter: iht=3, ihs=10, ihm=16

```
...nxcmnt comment cards (check file for compliance if lib=file:filename) ...
 siga1   siga2   siga3   siga4   siga5   siga6   siga7
 rnsigf1 rnsigf2 rnsigf3 rnsigf4 rnsigf5 rnsigf6 rnsigf7
 sigt1   sigt2   sigt3   sigt4   sigt5   sigt6   sigt7
 sig7->1    0       0       0       0       0       0
 sig6->1 sig7->2    0       0       0       0       0
 sig5->1 sig6->2 sig7->3    0       0       0       0
 sig4->1 sig5->2 sig6->3 sig7->4    0       0       0
 sig3->1 sig4->2 sig5->3 sig6->4 sig7->5    0       0
 sig2->1 sig3->2 sig4->3 sig5->4 sig6->5 sig7->6    0
 sig1->1 sig2->2 sig3->3 sig4->4 sig5->5 sig6->6 sig7->7
    0    sig1->2 sig2->3 sig3->4 sig4->5 sig5->6 sig6->7
    0       0    sig1->3 sig2->4 sig3->5 sig4->6 sig5->7
    0       0       0    sig1->4 sig2->5 sig3->6 sig4->7
    0       0       0       0    sig1->5 sig2->6 sig3->7
    0       0       0       0       0    sig1->6 sig2->7
    0       0       0       0       0       0    sig1->7
```

➢ NON-STANDARD FORMAT With DOWN Scatter: iht=3, ihs=4, ihm=10

```
 ...nxcmnt comment cards (check file for compliance if lib=file:filename) ...
 siga1   siga2   siga3   siga4   siga5   siga6   siga7
 rnsigf1 rnsigf2 rnsigf3 rnsigf4 rnsigf5 rnsigf6 rnsigf7
 sigt1   sigt2   sigt3   sigt4   sigt5   sigt6   sigt7
 sig1->1 sig2->2 sig3->3 sig4->4 sig5->5 sig6->6 sig7->7
    0    sig1->2 sig2->3 sig3->4 sig4->5 sig5->6 sig6->7
    0       0    sig1->3 sig2->4 sig3->5 sig4->6 sig5->7
    0       0       0    sig1->4 sig2->5 sig3->6 sig4->7
    0       0       0       0    sig1->5 sig2->6 sig3->7
    0       0       0       0       0    sig1->6 sig2->7
    0       0       0       0       0       0    sig1->7
```

- **chig** is the group fission probability for each material, and must correspond to each material number (in the order the materials are read in). Therefore, **chig** must be input as a vector of length **ngroup*nmatl**. Note that in the absence of fission, **chig** is **still used** as a weighting factor for the initial guess of scalar flux in each coarse mesh (**flxini**--see Block 2). *For this reason,* **chig** *is <u>un-normalized</u>. The user should insure when fission is present, group fission fractions sum to 1.0. The data for chig are produced by the GMIX gross section mixer code.*

- **nxcmnt** is the number of comment cards preceeding each Legendre order cross section set. (This setting has no effect on GIP-ORNL cross section formats). If capital letters are used in the cross section comment fields, the comments should be preceeded by a slash to avoid confusion with *FIDO* control characters during input processing.

- ***Adjoint* cross sections are *autoimatically transposed internally*** by setting the **modadj** parameter--see Block I inputs. By transposing the cross sections (inside *PENTRAN*), a forward code can be used to solve for the adjoint function, with proper attention to source definitions and fission parameters. Formats for up- and down-scatter and down scatter only are provided here for completeness.

-

STANDARD *ADJOINT* TRANSPOSED FORM,UP-DOWN Scatter: iht=3,ihs=10, ihm=16

```
siga7 rnsigf7 sigt7 sig7->1 sig7->2 ...sig7->6 sig7->7    0       0       0...
siga6 rnsigf6 sigt6    0    sig6->1 ...sig6->5 sig6->6 sig6->7    0       0...
siga5 rnsigf5 sigt5    0       0    ...sig5->4 sig5->5 sig5->6 sig5->7    0...
siga4 rnsigf4 sigt4    0       0    ...sig4->3 sig4->4 sig4->5 sig4->6 sig4->7...
siga3 rnsigf3 sigt3    0       0    ...sig3->2 sig3->3 sig3->4 sig3->5 sig3->6...
siga2 rnsigf2 sigt2    0       0    ...sig2->1 sig2->2 sig2->3 sig2->4 sig2->5...
siga1 rnsigf1 sigt1    0       0    ...   0    sig1->1 sig1->2 sig1->3 sig1->4...
```

- STANDARD *ADJOINT* TRANSPOSED FORM, DOWN Scatter: iht=3,ihs=4, ihm=10

```
siga7 rnsigf7 sigt7 sig7->7    0       0       0       0       0      0
siga6 rnsigf6 sigt6 sig6->6 sig6->7    0       0       0       0      0
siga5 rnsigf5 sigt5 sig5->5 sig5->6 sig5->7    0       0       0      0
siga4 rnsigf4 sigt4 sig4->4 sig4->5 sig4->6 sig4->7    0       0      0
siga3 rnsigf3 sigt3 sig3->3 sig3->4 sig3->5 sig3->6 sig3->7    0      0
siga2 rnsigf2 sigt2 sig2->2 sig2->3 sig2->4 sig2->5 sig2->6 sig2->7   0
siga1 rnsigf1 sigt1 sig1->1 sig1->2 sig1->3 sig1->4 sig1->5 sig1->6 sig1->7
```

Summary of Block 4 Inputs

Variable	Decription	Syntax Example
nprtyp	problem type classification	nprtyp=*problem_type* ([-maxsrc, 0, maxsrc])
nrdblk	automatic red-black coloring switch	nrdblk =0 (off) or =1 (on)
tolin	inner local flux iteration tolerance	tolin=*tolerance* OR tolin=CM1_tol, CM2_tol…
tolout	outer criticality tolerance, med grid multiplier	tolout=*tolerance, med_grid_multiplier*
Maxitr	max inner & outer iters/group, k-inner iter limit	maxitr=*max_no_of_iterations, criticality_inner_limit*
Methit	source iteration method	methit=*method_no* =1 (“Multigroup”) or =2 (“Hiromoto-Wienke”)
methac	acceleration method, csda parameters	methac=*method_no* (0,1,2,3,4,5, or 6), csda parameters
Ncoupl	*Taylor Projection (TPMC*) coupling order	ncoupl=*coupling order* (0 or 1)
ndmeth	differencing method by coarse mesh no	ndmeth= *Coarse1_diffmeth, Coarse2_diffmeth, …* set to 0(DD), 1(DZ), 2(DTW), 3(EDI), 4(EDW)
nzonrb	zones, damping, skip iterations for methac setting	nzonrb=*number_of_zones, damping_factor, skip_Iterations*
dtwmxw	DTW parameter vector for adaptive differencing settings (3)	dtwmxw=*maximum_DTW_weight, min_optical_thick, qfratio; default settings dtwmxw=0.95, 0.02, 1.00*
nquit	# iters a non-converging group is stopped	nquit=*number_of_iterations*

Termination of the Block with a T is required

nprtyp determines the type of transport problem to be solved:

- **nprtyp**=0 indicates a criticality eigenvalue problem (no fixed sources)

- **nprtyp** >=1 corresponds to 1 or more fixed sources, equal to the number of fixed sources in the problem. There is one fixed volumetric and planar source permitted for each coarse mesh--see Block 5. This input requires sdef cards, defined in Block 5.

- **nprtyp** <=-1 is a combination of the first two options: a criticality k-eigenvalue *with* fixed sources present

- the number of fixed sources is limited by the **maxsrc** parameter in *PENTRAN*

tolin defines the maximum local relative flux error for convergence.

- Note if one value is supplied, then that value is used globally for all coarse mesh cells

- If a vector of values are supplied, then individual tolin values are supplied to each coarse mesh cell. Note that if more than one value is entered, the number of entries must match the total number of coarse mesh cells.

- The vector of tolin values for local convergence enables the inifinity norm to be relaxed in specific local regions by coarse mesh designation.

tolout defines the relative tolerance on the *k*-effective system eigenvalue, and is only used in a criticality calculation.

- Solution convergence progression, including intermediate *k*-effective values and relative tolerances, are stored in logfiles (see *probname.K1* file, other log files, if used (see the loglevel indicator in the Header card).

- The final system *k*-effective from each method (power iteration, statistical mean, and balance-derived) is reported in the output file. Note these values should be fully consistent to within the outer convergence criterion.

- Note that while a solution may be converged based on the local relative convergence criteria **tolin**, the solution *must* satisfy the scalar balance equation to be converged. The integral system balance is collected and computed on each processor during problem output. If the net balance reported is not on the order of the solution tolerance (e.g. within an order of magnitude of the solution tolerance, sometimes a bit more if the problem is fully decomposed in parallel), then convergence has not been reached.

maxitr is a vector containing iteration limits.

- The first entry is a standard inner iteration limit.
- The second (optional) entry is an inner iteration limit for criticality problems. A recommended value for criticality eigenvalue problems is 10 inner iterations
- Note the first entry iteration limit holds for both the inner and outer loop tolerances. It is recommended to set the **maxitr** variable to be based on the inner iteration loop. A data dump occurs if **maxitr** is exceeded. The wall clock time (**timcut** in Block 1) can be used to control execution for the outer loop, if necessary.

methit refers to the algorithm for the source iteration sequence, either a 1 or a 2

- the "*multigroup*" (methit=1) method—standard
- the "one-level scheme" (methit=2) of Hiromoto and Wienke (1989). Either scheme may be used with any phase space decomposition strategy, although multigrid acceleration seems to perform best with the Hiromoto-Wienke scheme (see Sjoden and Haghighat, 1996).

methac is a vector input variable, where the first position refers to the rebalance acceleration method used. At present, damped Partial Current Coarse Mesh Rebalance (CMR) and System Rebalance (SR) are available, based on coarse mesh cells. Damping restricts oscillations and divergence in the rebalance (Rhoades, 1981). Note that rebalance is used to scale only the scalar flux (as opposed to the angular fluxes), and is performed to avoid additional reductions if angular decomposition is used. Options for methac are:

i. methac=0: No Acceleration
ii. methac=1 : *SR* only (based on nzonrb coarse mesh cells-see below)
iii. methac=2 : *PCR* only (based on nzonrb coarse mesh cells-see below)
iv. methac=3 : Alternating *PCR/SR* (based on nzonrb coarse mesh cells-see below)
v. methac=4, 5, or 6: Same as 1,2,3, respectively, but utilizing global synchronization.

- methac=1,2, or 3 settings use processor communications for rebalancing (using group-wise processor communicators), and can be used with any source iteration scheme.

- methac=4 ,5, or 6 requires an intermediate step of global process communications (which can be significantly more expensive), and can only be used with methit=2. This distinction between the use of communications in rebalancing methods was necessary due to limitations encountered on some parallel system implementations of the MPI standard.

- *PCR* rebalancing is performed using a direct Cholesky-LU factorization of the group rebalance matrix (the rebalance matrix is stored as an augmented system in an array, and is replaced in memory by Cholesky-factorized lower and upper triangular matrices).

- There is a limit on the size of the augmented rebalance matrix. Although the rebalance matrix is dimensioned as *n* x *(n+1)* of single precision numbers. The number of contiguously numbered coarse mesh cells, (e.g. a block of coarse meshes) that can be rebalanced at one time is limited by the **maxcmr** parameter in *PENTRAN*; as a result, multiple "blocks" of rebalance operations may be required, depending on how many coarse meshes are defined for a problem.
- A direct solution (rather than an iterative one) is used for rebalance in *PENTRAN* because efficient synchronization required, especially with complete phase space decomposition.

- The **nzonrb** vector permits the user to specify:

  - (1) how many coarse meshes should be considered in a rebalance zone (up to a current maximum of maxcmr), loaded into position 1;
  - (2) the damping factor to be used with partial current rebalance, where *dampf* <1 is under-damped, *dampf* =1 is critically damped, and *dampf* >1 is *over*-damped (not recommended), loaded into vector position 2

  - (3) the iteration *period* wherein rebalance is skipped, loaded into position 3 of the nzonrb vector.

  - If the execution is parallel, a check is made to determine if the locally assigned coarse meshes (determined by the nzonrb(1) setting) belong to the zone of coarse meshes undergoing rebalance; if not, rebalance is bypassed, saving execution time.

  - *The user should be aware that setting* **nzonrb** *too small (covering a small block of coarse meshes) may cause iteration instability and divergence of the solution, as may using overdamping--of course, this is problem dependent.*

- **Collapsed Source Diffusion Acceleration (CSDA).** Subsequent vector positions for **methac** are for CSDA (collapsed source diffusion acceleration) sources.

  - Note: at present, this is an unproven method by which to accelerate the transport solution, and is not recommended for use at this time. This is a completely experimental feature where an analytical diffusion approximation is used in each energy group to preset flux iterates in *PENTRAN* based on a collapsed source from integration over problem geometry.

  - The vector positions for this are as follows, starting with methac position (2) entry: csda init Off=0/Point=1/Cosine=2), csda xcenter (0=auto), csda ycenter

(o=auto), csda zcenter (o=auto), inner iterate # for application use of csda, outer iterate # for application of csda.

- Zero entries for "center" values force *PENTRAN* to determine an average source location based on the distribution. "Point" refers to a collapsed point source; "Cosine" refers to a cosine source distribution.

ncoupl defines the order of coupling for Taylor Projection Mesh Coupling.

- Setting ncoupl=0 forces *oth* order, while setting ncoupl=1 implements first order coupling.

- A zero setting forces all coupling coefficients to be zero, thus invoking only a direct flux assignment/simple flow balance at each interface during a transport sweep.

- Note that restricting the coupling to *oth* order does not save significant computational work.

ndmeth defines the spatial differencing method to be used

- It is a vector containing the differencing method to be used in each coarse mesh cell.

- At present, setting this =0 selects DD (with no fixup, no upgrade); =1 selects DZ (can be upgraded to DTW, EDI); =2 in each coarse mesh selects DTW (can be upgraded to EDI); =3 selects EDI, while =4 selects EDW.

- An upgradeable diamond (DZ) setting will automatically use DTW differencing if a negative (set to zero) flux fixup is detected during any angular flux sweep. From there, if a DTW maximum weight (in any direction) is greater than **dtwmxw**, then the DTW method is upgraded to EDI. This allows EDI to take over from DTW, since high weights in DTW tend to occur in thicker cells with streaming, where EDI is more accurate.

- A negative differencing number locks that method (with the exception of DD, already locked using a zero setting), blocking upgrade options; for example, setting **ndmeth** to -1 in a coarse mesh cell locks DZ differencing regardless of fixup calls, blocking any upgrade. For more information on differencing, refer to Chapter 2 of this document.

dtwmxw is the keyword variable where the user can optionally apply a specification vector containing three entries in a specified order (dtwmxw, edwmno, qfratio).

- The first position (internal code variable dtwmxw) defines the maximum accepted weight for DTW for adaptive differencing that, if reached, is used to trigger an upgrade to EDI differencing.

- The second position in the vector defines the minimum optical thickness (internal code parameter edwmno) to consider for adaptive differencing; if this minimum optical thickness ($\sigma \Delta h$) is not achieved, then the DTW scheme is used by default.
- The third field specifies the allowed ratio of the cell angular source density to the cell angular collision density (internal code variable *qfratio*, default of 1.00); if the cell has a ratio less than *qfratio*, the cell is not dominated by a radiation source term, and is therefore a cell wherein radiation is streaming, and adaptive differencing is allowed. Otherwise, the DTW scheme is used. Cells where strong sources are present yield a concave down spatial profile, and therefore are more accurately represented using the DTW scheme. Hence, the *qfratio* setting is actually a ratio of the local angular source density and product of average angular flux and total cross section. Therefore, if onewants adaptive differencing to be applied more often, then this value should be raised.
- The Default settings for dtwmxw are: **dtwmxw**=0.95, 0.02, 1.00
  The ranges for settings are: dtwmxw=[0.5,1.0],N/A,[>0.0]
  This means that the linear maximum weight where DTW will be upgraded to EDI is 0.95; the optical thickness of the cell for upgrade to EDI is 0.02; the *qfratio* is 1.00, so that an effective angular source to product of angular flux ratio and total cross section must be less than 1.00 to enable an upgrade to EDI.

nrdblk=1 can be set to engage automatic red-black coloring based on problem coarse meshes, used only to accelerate parallel spatial decomposition.

- Coarse meshes are re-ordered in a "stacked checkerboard" sequence to use the most recent iteration angular fluxes as soon as they are available, to the greatest extent possible. Note that use of this setting when not performing parallel spatial decomposition can (in some cases) hinder convergence, and should be avoided.

- A warning message is issed if nrdblk is engaged without spatial decomposition. Also note that the effectiveness of red-black coloring may be limited by automatic load balancing, especially if only "red" (or "black") cells are allocated to one processor due to load reracking (see Block 1).

nquit sets the maximum number of iterations permitted to stop a non-converging group when that group fails to converge based on a specific location (coarse and medium or fine mesh number) in the problem.

- The defailt value is nquit=4 if no value is specified. After failure, the group is set as "converged" with warnings.

Summary of Block 5 Inputs

Variable	Decription	Syntax Example
nsdef	Source type of each source	nsdef=*source1_type , source2_type, …* (0=volumetric, 1=planar source)
nscmsh	Coarse mesh number of each source	nscmsh=*coarse_mesh#_source1, …*
ssnrm **	Surface Source normal vector	ssnrm=*u1,v1,w1,u2,v2,w2, …*
sref	Arbitrary reference coordinate for each source	sref=*x1,y1,z1,x2,y2,z2, …*
serg	Source energy distribution with respect to energy groups (*1*…ngroup), each source	serg=*g1_prob_source1, g2_prob_source1, …*
smag	Source integral magnitude, each source	smag=*source1_magnitude, source2_magnitude …*
spacpf *	Source fine mesh spatial dist, specified	spacpf=*src#, grp#, #cells, cell1_prob, cell2_prob …*
omegap *	Source angular dist, specified by octant no	omegap=*src#, grp#, octant#,* Ω_{1_prob},… Ω_{n_prob}.
scalsf *	Source fine mesh scale factor, specified	scalsf=*src#, grp#, scale_factor …*
rkdef	Criticality eigenvalue estimate for system	rkdef= k*eff_estimate*
kextrp *	Aitken extrapolation switch for k_{eff}	kextrp= *0 (none)/1 (extrapolate, default)*

*Optional
**Required for each source if nsdef=1 detected

Termination of the Block with a T is required

In the event of a criticality problem with no fixed sources, *only* the **rkdef** card is required.

rkdef is the user-supplied initial guess for the integral system k_{eff} (criticality eigenvalue). In the event of a fixed source problem with no fission, the **rkdef** card is not required.

Regarding **kextrp**, if a criticality problem, the Aitken extrapolation discussed in Section 2.10 can be disabled by setting kextrp=0. Otherwise, the code defaults to using extrapolation of the eigenvalue to help accelerate convergence.

All sources are defined as isotropic with equal probability in each fine spatial mesh ***unless modified*** by **omegap** and/or **spacpf**. To be considered "active," these probability distributions must assign values to have an integral probability magnitude >1.E-15.

The medium mesh source is automatically constructed using a "nearest neighbor" approach from the fine grid source definition. If the medium grid density is too sparse to resolve a source distribution based on the fine grid, a warning message is issued.
nsdef defines either a volumetric source (#/cm³/s) or a planar (#/cm²/s) source. One of each is possible in each coarse mesh cell. The number of entries in **nsdef** must equal the absolute value of **nprtyp** (see Block 4).

nscmsh is a vector providing the reference location of sources (in order of source number) by coarse mesh number. The number of entries in **nscmsh** must equal the absolute value of **nprtyp** (see Block 4).

sref permits an arbitrary reference coordinate to be specified for each source, in order of source number. The number of entries in **nsref** must equal the absolute value of **nprtyp*3** (see Block 4).

serg is a vector containing the energy group probability distribution for each source, in order of source number. The number of entries in **serg** must equal the absolute value of **nprtyp*ngroup** (see Blocks 1, 4).

smag defines the integral source magnitude (over all variables), in order of source number. The number of entries in **smag** must equal the absolute value of **nprtyp** (see Block 4). Negative smag entries activate source normalization for the source number.

ssnrm is only required if there are **nsdef** entries equal to 1, indicating planar sources. *In that case,* **ssnrm** *entries are required for* <u>*all*</u> *sources*, although they are not used in the case where sources are volumetric.

- The **ssnrm** defines a vector for each source pointing from the center of the coarse mesh where the source is located. *PENTRAN* turns this vector into a *unit vector*, and then assigns a planar source to the surface of the 3-D coarse mesh boxoid normal to the largest component of that unit vector.

- If the surface is on a shared *interior* problem boundary, a similar planar source is implicitly defined for the adjacent mesh cell sharing the common boundary. *Since only* ***one*** *surface source can be defined for one coarse mesh, the coarse mesh containing the implicitly defined planar source cannot contain any other planar source.* This restriction was made due to memory limitations.

- Therefore, planar sources on interior surfaces cannot use discontinuous meshing between coarse mesh cells on the common surface where the source is located. Note that a spatial distribution can also be assigned to a volumetric source(s) to simulate a planar source, although volumetric sources are assumed to be averaged at the cell center.

- The number of entries in **ssnrm** must equal the absolute value of **nprtyp*3** (see Block 4).

Source particles from planar sources located at system boundaries entering into the system show up *implicitly* in boundary currents of particle balances. Planar sources defined on *interior* surfaces will be indicated in the system balances as "source particles."

omegap is an *optional variable* where the user can specify an angular probability distribution for any coarse mesh source.

- *If this variable **is not defined** for the source number*, an equal (isotropic) angular probability is assigned for the source.

- *If this variable **is defined** for the source number,* it must be specified to indicate the source number, group number, angular octant number, and corresponding probability for each angle in the octant *only for the sources that require a non-uniform angular probability distribution.*

- *If the group number is entered as a negative group number, the angular probability function is systematically applied to all energy groups*. Each angular probability distribution is un-normalized.

- Probabilities for the total number of angles in any specified octant are required, and it is the user's responsibility to supply the correct number. Probabilities for octants not defined are implicitly assumed to be zero.

- A numbering pattern to guide probability assignment for sweep octants (and angles in each sweep octant) is provided in the Appendix; a complete quadrature set can be found in output files as a reference.

spacpf is an *optional variable* where the user can specify a fine mesh spatial distribution to a any coarse mesh source.

- *If this variable **is not** defined for the source number*, an equal spatial probability is assigned for the source.

- *If this variable **is** defined for the source number,* it must be specified to indicate the source number, group number, number of sequentially numbered meshes, and corresponding mesh spatial probabilities *only for the sources that require a non-uniform spatial probability distribution.*

- *If the group number is entered as a negative group number in* **spacpf***, the spatial probability function is systematically applied to <u>all</u> energy groups*.

- If a **scalsf** is declared, then this is used for all groups as well (see below).

- *Probabilities for cells not defined are assumed to be zero.* Note that if an un-normalized distribution is specified (the default), a warning message is reported.

For a planar boundary source, each **spacpf** probability is still referenced by fine sequential cell number--although probabilities in cells not on the coarse mesh boundary technically have no effect on the boundary cell source probabilities. For clarity, the user should still include zeros for cell location probabilities not on the boundary surface. Again, the sequential numbering scheme described in Block 2 for coarse meshes also applies to fine (and medium) meshes contained within each coarse mesh. (Note that cell probabilities are automatically transferred to the implicitly defined planar source for the adjacent coarse mesh on the common boundary--this is why discontinuous meshing on the boundary surface containing the source is not permitted).

scalsf is an optional scale factor available for applying a scalar multiplier to any ***spatial*** probability distribution. To be "active," the magnitude of the scale factor must be greater than 1.E-50. The source number, group number, and positive scale factor, in order, are required for each applicable source. Note: if the **spacpf** distribution applied to all groups (so that the group number is entered as a negative number), then the scale factors defined in **scalsf** only need to be entered for group 1, and will be automatically apllied throughout all groups.

Summary of Block 6 Inputs

Variable	Decription	Syntax Example
ibback	Global "back" (-x) surface boundary condition	ibback=*type, group1_albedo, group2_albedo,…*
ibfrnt	Global "front" (+x) surface boundary condition	ibfrnt=*type, group1_albedo, group2_albedo,…*
jbeast	Global "east" (-y) surface boundary condition	jbeast=*type, group1_albedo, group2_albedo,…*
jbwest	Global "west" (+y) surface boundary condition	jbwest=*type, group1_albedo, group2_albedo,…*
kbsout	Global "south" (-z) surface boundary condition	kbsout=*type, group1_albedo, group2_albedo,…*
kbnort	Global "north" (+z) surface boundary condition	kbnort=*type, group1_albedo, group2_albedo,…*

Termination of the Block with a T is required

Note the name of each boundary condition begins with the axis normal to each surface. Global boundary condition names correspond to global system boundaries with a right handed 3-D Cartesian coordinate system:

- "back" (-x)
- "front" (+x)
- "east" (-y)
- "west" (+y)
- "south" (-z)
- "north" (+z)

Boundary types (illustrated with **ibback,** but can apply to any of the boundary variables):

- **ibback**=0 is a *vacuum* boundary. *No albedo factors are required*.
- **ibback**=1, … is an albedo boundary--***ngroup** albedo factors are required immediately following the type* (the "1").
    - For perfect spectrally reflective boundaries, all group albedo factors should be *unity* (1.0).
    - Hint: One can often represent a complex boundary "wall" geometry with a small model, and with the detailed J+/J- information available on individual coarse meshes in the output, derive a spectrally dependent set of albedo factors for a larger model (Sjoden, et al, 2000).
    - Gray/White boundaries are **not** supported. Periodic boundaries are **not** supported.

Summary of Block 7 Inputs

Variable	Decription	Syntax Example
nxspr	Print cross section tables in output	nxspr=0 (off) or =1 (on)
ngeopr	Print a *Mathematica*™ readable geometry file	ngeopr=0 (off) or =1 (on)
nsumpr	Print local coarse mesh summary tables in output	nsumpr=0 (off) or =1 (on)
meshpr	Vector list of coarse mesh cells with formatted output	meshpr=0 (none) or *coarse_mesh#, …*
nfdump	Binary data dump of mesh scalar fluxes/moments	nfdump=0 (off)or =(*legord_dumped+1*)
nsrcpr	Print source/distribution tables in output	nsrcpr=0 (off) or =1 (on), =2 (detailed)
nsdump	Binary data dump of mesh scalar sources	nsdump=0 (off) or =1 (on)
nmatpr	Print a material map in output files for local coarse cells	nmatpr=0 (off) or =1 (on)
nadump	Vector list of Binary data dump for angular fluxes	nadump=*coarse_mesh#, dump_type, …*
njdump	Binary data dump for flux and net fine mesh currents (Jnet)	njdump=0 (off) or =1 (on)

Termination of the Block with a T is required

Default settings are as follows: nxspr=1, ngeopr=2, nsumpr=1, meshpr=0, nfdump=1, nsrcpr=1, nsdump=1, nmatpr=1, nadump=0, njdump=0. These settings are used if the parameter is not listed in the input deck.

ngeopr can be set to values between [0,2], and prints increasingly detailed information about the problem geometry.

- If **ngeopr=1**, only coarse mesh geometry information is output; if **ngeopr=2**, then medium/fine mesh details are printed.

- If no **meshpr** field is specified, full formatted output, incuding six-face partial and net currents, are output for all coarse and medium (fine) meshes. **A negative coarse mesh number supresses the fine mesh flux and partial current outputs, yielding formatted output of coarse mesh computed scalar fluxes and currents only. An example of the type of data from a negative setting for coarse mesh 53 from the cardd**

- It is not recommended that medium (fine) mesh data be specified for many coarse meshes with **meshpr**, since flux moments are accessible from binary files (see **nfdump**), automatically managed using the *PENDATA* utility. Still, it is helpful to request data from a few sample coarse meshes, since optical thickness, current balance, and other helpful data are printed into the output file.

```
 MESH/HOMOGENIZED OPTICAL DATA  Group   1  Coarse Cell   53  Dom Matl   1

Mesh  Cells xDiv  dex(cm)  xMFPH yDiv  dey(cm)  yMFPH zDiv  dez(cm)  zMFPH
----  ----- ---- -------- ------ ---- -------- ------ ---- -------- ------
Crs       1   1    6.00007499.94   1    2.00002499.98   1   12.0000*******
Fine    160  10    0.6000 749.99   4    0.5000 624.99   4    3.00003749.97

Coarse Center = (  15.0000,   1.0000,  22.0000 )  Volume =  1.4400E+02 cm3
Normal Area to :   x =  2.4000E+01  y =  7.2000E+01  z =  1.2000E+01 cm2
Hsiga_g = 6.45916E+01  Hnsigf_g = 0.00000E+00  Hsigt_g = 6.53327E+01 1/cm
Hsig_g->g = 7.41082E-01 1/cm    Hchi_g = 0.00000E+00   Group Homog c=0.011

     COARSE BOUNDARY DATA  Group   1  Coarse Cell   53  Dom Matl   1
                     Final Error Norm=  8.8637E-07

 Crs  Boundary     Phi|         J+|         J-|       J Net|   Phi|BdySrc
 Bdy     cm      #/cm2/s      #/cm2/s     #/cm2/s    #/cm2/s   Avg #/cm2/s
----- -------- ----------- ----------- ----------- ----------- -----------
x-1/2  12.0000  6.5255E-05  1.2858E-07  1.2261E-05 -1.2132E-05  0.0000E+00
x+1/2  18.0000  6.5325E-05  1.2699E-05  1.2825E-07  1.2570E-05  0.0000E+00
y-1/2   0.0000  3.4330E-19  0.0000E+00  2.6472E-19 -2.6472E-19  0.0000E+00
y+1/2   2.0000  1.0960E-02  5.4601E-07  8.9992E-03 -8.9987E-03  0.0000E+00
z-1/2  16.0000  1.1105E-04  2.6579E-05  1.3937E-07  2.6440E-05  0.0000E+00
z+1/2  28.0000  4.3798E-05  2.9356E-05  7.8157E-09  2.9348E-05  0.0000E+00
```

Fig 3.4.2: Example of Coarse mesh based detail available (derived from the 'cardd.3' file, output from processor 3 of 8, profiled in the Appendix), useful for partial currents, etc. Note that fine mesh net current data (with fluxes) can be obtained by setting the **njdump** variable.

- The value of **nfdump** specifies the order of the angular flux moments to be dumped; setting this =1 indicates that only *zeroth order moments* (scalar fluxes) will be dumped. The maximum is (**legord**+1), which would dump all moments through order **legord**.

- If one desires only scalar fluxes from one portion of the problem, it may be better to select the appropriate coarse mesh using **meshpr**, since **nfdump** yields (binary) moment data for the entire problem phase space. Again, binary files are dumped by each processor only for the phase space computed on that processor. Data from multiple processor files can be automatically gathered with the *PENDATA* utility.

- Binary files generated by **nfdump** and **nsdump** have file input prefixes + '.**f***processor#*' and '.**s***processor#*', respectively. Note that since medium (fine) mesh data are stored locally, the output on each processor number is dependent on coarse mesh and group processor assignment.

- **nadump** permits the cell centered fine mesh angular fluxes from a particular coarse mesh to be dumped to binary file format (see *PENDATA* option 8).

- For **nadump**, each coarse mesh number listed must be followed *immediately* by an angular flux ***dump type***, which specifies the hemisphere of angular fluxes to dump for each medium/fine mesh in the coarse mesh, as follows:

Dump types for angular fluxes using **nadump***:*

- indicates Type 1: - x hemisphere, sweep octants 1, 3, 6, 8
- . . . . . .Type 2: +x hemisphere, sweep octants 2, 4, 5, 7
- . . . . . .Type 3: - y hemisphere, sweep octants 1, 3, 5, 7
- . . . . . .Type 4: +y hemisphere, sweep octants 2, 4, 6, 8
- . . . . . .Type 5: - z hemisphere, sweep octants 1, 4, 5, 8
- . . . . . .Type 6: +z hemisphere, sweep octants 2, 3, 6, 7
- . . . . . .Type 7: complete sphere, sweep octants 1 through 8

- Binary files generated by **nadump** have file prefixes + '.**a**processor#'.

- **WARNING**! Be very careful of how much angular flux data you ask for with **nadump** -- this requires a great deal of disk space! Since angular fluxes can be dumped based on a coarse mesh, you may want to set up a "thin wall" coarse mesh to obtain a hemisphere of exiting angular fluxes to use for coupling/input for another calculation. Since angular data is stored in parallel, where it is only locally available to those processors working on their assigned portion of the phase space, you will need to gather angular flux data from multiple processor files—this can be automatically gathered with the *PENDATA* utility.

njdump yields a set of binary files that contain flux and net current over the fine mesh contained in a particular coarse mesh to be dumped to binary file format; see *PENDATA* option 9.

- Binary files generated by **njdump** have file prefixes + '.**j**processor#'.

5. APPENDIX

LEVEL SYMMETRIC QUADRATURE SETS

Level symmetric quadrature weights are given as point weights; these *initially* sum to 1.0 *for each octant*. Following initial assignment in *PENTRAN*, all weights are then multiplied by 1/8 for normalization on the unit sphere.

- The number of Ω's per octant is (isn*(isn+2)/8), with (isn/2) *distinct* □'s, assuming isn is the discrete ordinates quadrature order.
- Point weights derived from level weights for S14 and above can vary due to more than one positive real root possible to satisfy the criteria for level symmetry; therefore, point weights derived here for S14 and above may or may not differ from those found elsewhere.
- All quadratures in *PENTRAN* were derived with a numerical precision of at least 1.0D-15. A more detailed *PENTRAN* quadrature listing can be obtained using the *PENQUAD* utility.

Example: **S6 Level Symmetric** ***PENTRAN* Ω Sampling order:**

(μ>0, η>0, ξ>0)

```
      ξ                    ξ
       1                    1
      2 2                  2 3
     1 2 1                4 5 6
μ         η          μ         η
```

6*8/8 = 6 Ω's per octant 6/2 = 3 unique μ's

PENTRAN Assignment of S6 Ω's:

Ω #	μ=μm	η=μm	ξ=μm	w=wm
1	1	1	3	1
2	2	1	2	2
3	1	2	2	2
4	3	1	1	1
5	2	2	1	2
6	1	3	1	1

❑ S2 Level Symmetric Quadrature

S2	ξ	ξ
(+ + +) Octant	1	1
m-Level Diagram	μ η	μ η
(Sweep 2)		

wm(1)=1.D0
μm(1)=0.5773502691896257D0

❑ S4 Level Symmetric Quadrature

S4	ξ	ξ
(+ + +) Octant	1	1
m-Level Diagram	1 1	2 3
(Sweep 2)	μ η	μ η

wm(1)=0.3333333333333333D0

μm(1)=0.3500211745815406D0
μm(2)=0.8688903007222013D0

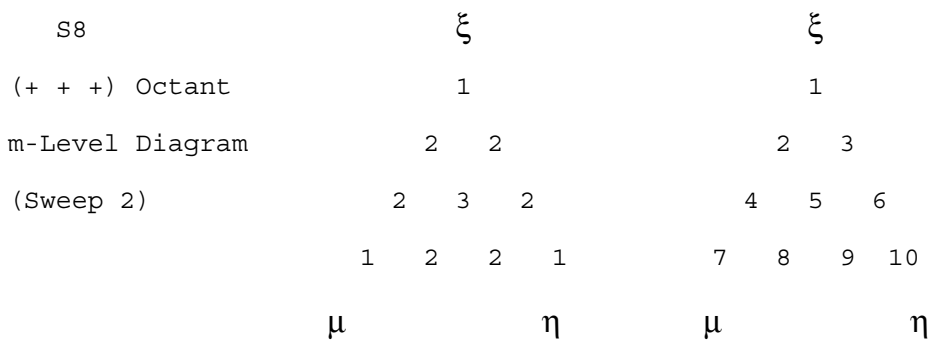
❑ S6 Level Symmetric Quadrature

S6	ξ	ξ
(+ + +) Octant	1	1
m-Level Diagram	2 2	2 3
(Sweep 2)	1 2 1	4 5 6
	μ η	μ η

wm(1)=0.1761261308633819D0 wm(2)=0.1572072024699513D0

μm(1)=0.2666354015167032D0 μm(2)=0.6815077265365472D0
μm(3)=0.9261809355174899D0

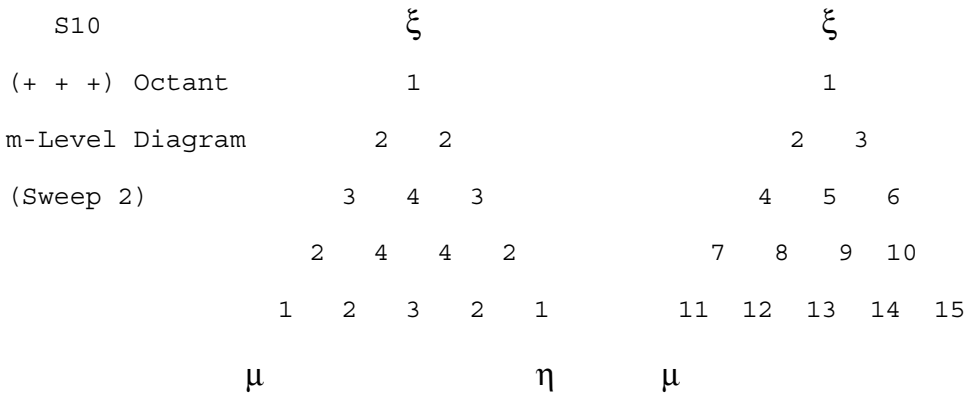
❑ S8 Level Symmetric Quadrature



wm(1)=0.1209876543209866D0 wm(2)=0.0907407407407413D0
wm(3)=0.0925925925925926D0

μm(1)=0.2182178902359909D0 μm(2)=0.5773502691896258D0
μm(3)=0.7867957924694435D0 μm(4)=0.9511897312113425D0

❑ S10 Level Symmetric Quadrature



wm(1)=0.0893031479843530 2D0 wm(2)=0.0725291517123689 0D0

wm(3)=0.0450437674364028 8D0
wm(4)=0.0539281144878397 1D0

μm(1)=0.1893213264780056D0 μm(2)=0.5088817555826185D0
μm(3)=0.6943188875943850D0 μm(4)=0.8397599622366860D0
μm(5)=0.9634909811104704D0

❑ S12 Level Symmetric Quadrature

```
     S12                    ξ                          ξ
(+ + +) Octant              1                          1
m-Level Diagram           2   2                      2   3
(Sweep 2)               3   4   3                  4   5   6
                      3   5   5   3              7   8   9  10
                    2   4   5   4   2          11  12  13  14  15
                  1   2   3   3   2   1      16  17  18  19  20  21
                          μ                          η      μ                          η
```

wm(1)=0.07076258997008411D0
wm(2)=0.05588110156489365D0
wm(3)=0.03733767375882513D0
wm(4)=0.05028190106005664D0
wm(5)=0.02585129165575492D0

μm(1)=0.1672126528227026D0
μm(2)=0.4595476346425931D0
μm(3)=0.6280190966421315D0
μm(4)=0.7600210148336660D0
μm(5)=0.8722705430257244D0
μm(6)=0.9716377192513620D0

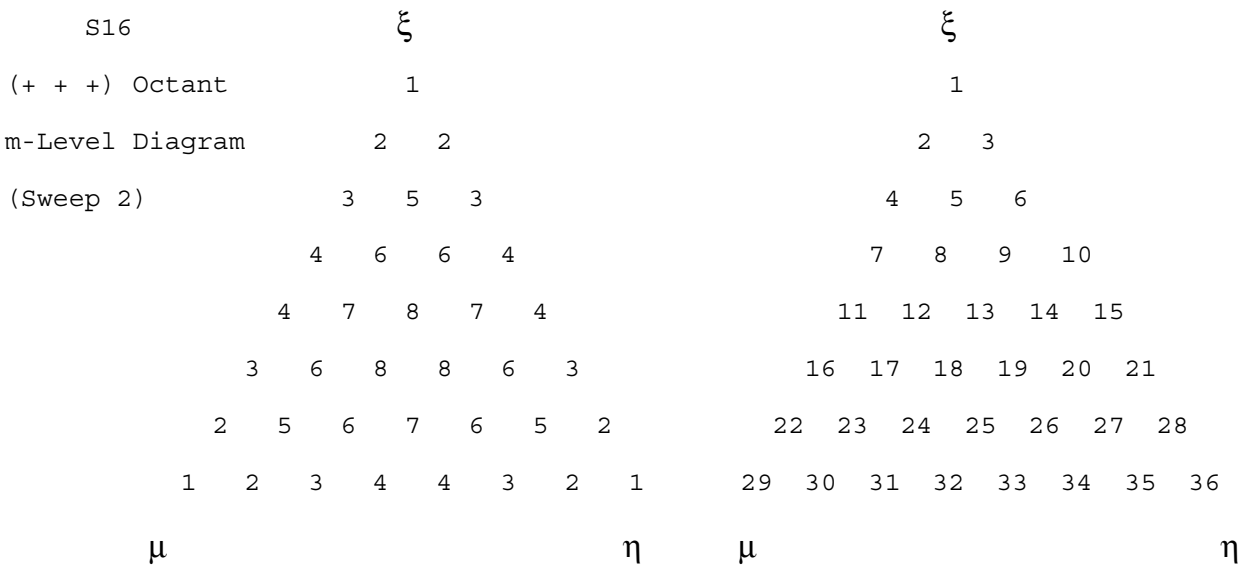
❑ S14 Level Symmetric Quadrature

```
     S14                    ξ                          ξ
(+ + +) Octant              1                          1
m-Level Diagram           2   2                      2   3
(Sweep 2)               3   5   3                  4   5   6
                      4   6   6   4              7   8   9  10
                    3   6   7   6   3          11  12  13  14  15
                  2   5   6   6   5   2      16  17  18  19  20  21
                1   2   3   4   3   2   1  22  23  24  25  26  27  28
                          μ                          η      μ                          η
```

wm(1)=0.05799704089709301D0
wm(2)=0.04890079763671375D0
wm(3)=0.02214970797116879D0
wm(4)=0.04070085313525794D0
wm(5)=0.03938673868440395D0
wm(6)=0.02455175510137073D0
wm(7)=0.01213253759421592D0

μm(1)=0.15198586146I1801D0 μm(2)=0.4221569823048237D0
μm(3)=0.5773502691896257D0 μm(4)=0.6988920867758852D0
μm(5)=0.8022262552313840D0 μm(6)=0.8936910988743190D0
μm(7)=0.9766271529257242D0

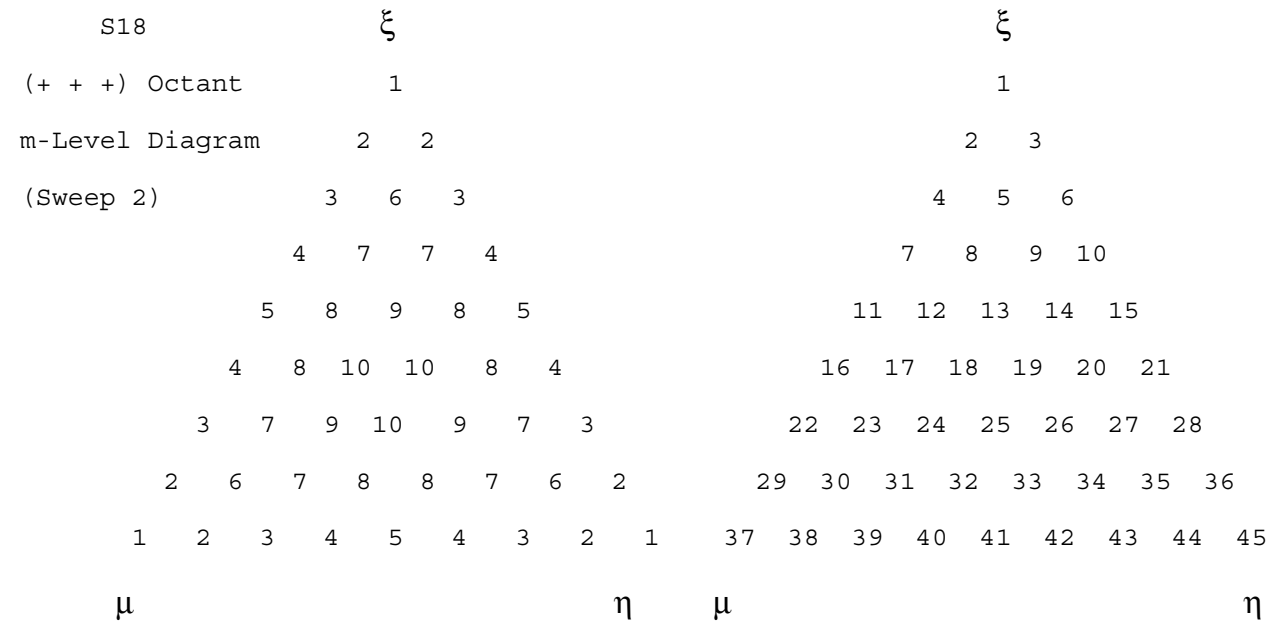
❑ S16 Level Symmetric Quadrature



wm(1)=0.04898723915796233D0 wm(2)=0.04132959786990422D0
wm(3)=0.02244759769020243D0 wm(4)=0.02440567883044038D0
wm(5)=0.03361864689378468D0 wm(6)=0.01567390172962426D0
wm(7)=0.03692573110461228D0 wm(8)=0.00608816393663137D0

μm(1)=0.138956875067641 6D0 μm(2)=0.3922892614447836D0
μm(3)=0.5370965613008739D0 μm(4)=0.6504264506287802D0
μm(5)=0.7467505736146995D0 μm(6)=0.8319965569100706D0
μm(7)=0.9092855009437586D0 μm(8)=0.9805008790117792D0

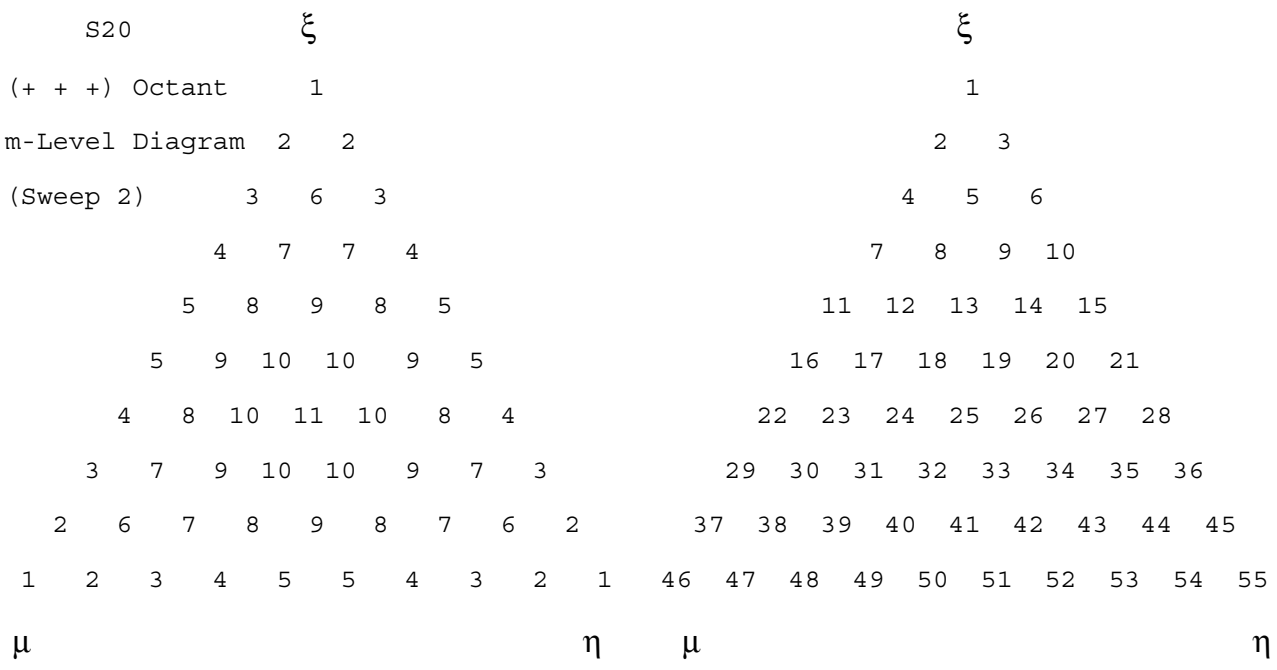
❑ S18 Level Symmetric Quadrature



wm(1)=0.0422646448821714 9D0
wm(2)=0.0376127473855226 5D0
wm(3)=0.0066907320068974 2D0
wm(4)=0.0391919328951441 7D0
wm(5)=0.0042549971742542 1D0
wm(6)=0.0423661901429511 6D0
wm(7)=0.0092396276440937 6D0
wm(8)=0.0156647508615558 5D0
wm(9)=0.0136576046459212 8D0
wm(10)=0.0139903149016074 8D0

μm(1)=0.1293445045421084D0
μm(2)=0.3680438160525554D0
μm(3)=0.5041651517249193D0
μm(4)=0.6106625499349821D0
μm(5)=0.7011668842525139D0
μm(6)=0.7812561994964660D0
μm(7)=0.8538662066922110D0
μm(8)=0.9207680210618902D0
μm(9)=0.9831276612370913D0

❑ S20 Level Symmetric Quadrature



wm(1)=0.0370210490616941 5D0
wm(2)=0.0332842165365488 8D0
wm(3)=0.0139670148926500 8D0
wm(4)=0.0290851323204875 4D0
wm(5)=0.0062319300460547 4D0
wm(6)=0.0262166700044418 5D0
wm(7)=0.0022875393888147 6D0
wm(8)=0.0363991290209142 4D0
wm(9)=0.0089905960145254 3D0
wm(10)=0.0029760691215602 7D0
wm(11)=0.0109570787522563 8D0

μm(1)=0.1206033430392688D0
μm(2)=0.3475742923164429D0
μm(3)=0.4765192661438829D0
μm(4)=0.5773502691896257D0
μm(5)=0.6630204036531319D0
μm(6)=0.7388225619100911D0
μm(7)=0.8075404016607585D0
μm(8)=0.8708525837599884D0
μm(9)=0.9298639389547678D0
μm(10)=0.9853474855580162D0

ANGULAR OCTANT SWEEPING ASSIGNMENTS

❑ Octant numbers are assigned on the unit sphere with the signs given for each direction cosine. Also listed are the starting corners where sweeping originates in a Cartesian system. For a more detailed quadrature list, use the *PENQUAD* utility.

Summary of *PENTRAN* GENERAL OCTANT Sweeping Assignments

Sweep	μ	η	ξ	Start	Sweep	μ	η	ξ	Start
1	-	-	-	FWN	2	+	+	+	BES
3	-	-	+	FWS	4	+	+	-	BEN
5	+	-	-	BWN	6	-	+	+	FES
7	+	-	+	BWS	8	-	+	-	FEN

B=Back(-x) F=Front(+x) E=East(-y) W=West(+y) S=South(-z) N=North(+z)

PENQUAD: SUPPLEMENTAL LEVEL SYMMETRIC ANGULAR DATA

In the event the user requires direct access to level symmetric angular quadrature data for a particular quadrature set, or would like to print out the entire quadrature set for assigning an angular source distribution in *PENTRAN*, the *PENQUAD* utility has been developed. This utility outputs all of the quadrature data for any level-symmetric set in *PENTRAN* from Sn=S_2 to S_{20}, including all evaluated Legendre polynomials through P_7, depending on the user-specified options. A sample output from the *PENQUAD* utility for S_4 quadrature and P_1 Legendre functions is given below ("aphi" is the azimuthal angle in radians):

```
PENQUAD S 4 Level Symmetric Quadrature in PENTRAN 4.xxbeta with P1 Legendre
 i Sweep Omega    w           mu          eta           xi          aphi
 1   1     1  4.16667E-02 -3.50021E-01 -3.50021E-01 -8.68890E-01 -1.95375E+00
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)= 8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 2   1     2  4.16667E-02 -8.68890E-01 -3.50021E-01 -3.50021E-01 -2.35619E+00
   P(0)= 1.00000E+00 P(1)=-8.68890E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 3   1     3  4.16667E-02 -3.50021E-01 -8.68890E-01 -3.50021E-01 -2.75864E+00
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)= 8.68890E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 4   2     1  4.16667E-02  3.50021E-01  3.50021E-01  8.68890E-01  1.18785E+00
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)=-8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 5   2     2  4.16667E-02  8.68890E-01  3.50021E-01  3.50021E-01  7.85398E-01
   P(0)= 1.00000E+00 P(1)= 8.68890E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 6   2     3  4.16667E-02  3.50021E-01  8.68890E-01  3.50021E-01  3.82950E-01
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
       P(1,1)*COS(1*aphi)=-8.68890E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 7   3     1  4.16667E-02 -3.50021E-01 -3.50021E-01  8.68890E-01  1.95375E+00
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)=-8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 8   3     2  4.16667E-02 -8.68890E-01 -3.50021E-01  3.50021E-01  2.35619E+00
   P(0)= 1.00000E+00 P(1)=-8.68890E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
 9   3     3  4.16667E-02 -3.50021E-01 -8.68890E-01  3.50021E-01  2.75864E+00
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)= 8.68890E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
10   4     1  4.16667E-02  3.50021E-01  3.50021E-01 -8.68890E-01 -1.18785E+00
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)= 8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
11   4     2  4.16667E-02  8.68890E-01  3.50021E-01 -3.50021E-01 -7.85398E-01
   P(0)= 1.00000E+00 P(1)= 8.68890E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
12   4     3  4.16667E-02  3.50021E-01  8.68890E-01 -3.50021E-01 -3.82950E-01
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
```

```
       P(1,1)*COS(1*aphi)=-8.68890E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
13   5     1  4.16667E-02  3.50021E-01 -3.50021E-01 -8.68890E-01 -1.95375E+00
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)= 8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
14   5     2  4.16667E-02  8.68890E-01 -3.50021E-01 -3.50021E-01 -2.35619E+00
   P(0)= 1.00000E+00 P(1)= 8.68890E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
15   5     3  4.16667E-02  3.50021E-01 -8.68890E-01 -3.50021E-01 -2.75864E+00
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
       P(1,1)*COS(1*aphi)= 8.68890E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
16   6     1  4.16667E-02 -3.50021E-01  3.50021E-01  8.68890E-01  1.18785E+00
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)=-8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
17   6     2  4.16667E-02 -8.68890E-01  3.50021E-01  3.50021E-01  7.85398E-01
   P(0)= 1.00000E+00 P(1)=-8.68890E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
18   6     3  4.16667E-02 -3.50021E-01  8.68890E-01  3.50021E-01  3.82950E-01
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)=-8.68890E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
19   7     1  4.16667E-02  3.50021E-01 -3.50021E-01  8.68890E-01  1.95375E+00
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)=-8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
20   7     2  4.16667E-02  8.68890E-01 -3.50021E-01  3.50021E-01  2.35619E+00
   P(0)= 1.00000E+00 P(1)= 8.68890E-01
       P(1,1)*COS(1*aphi)= 3.50021E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
21   7     3  4.16667E-02  3.50021E-01 -8.68890E-01  3.50021E-01  2.75864E+00
   P(0)= 1.00000E+00 P(1)= 3.50021E-01
       P(1,1)*COS(1*aphi)= 8.68890E-01
       P(1,1)*SIN(1*aphi)=-3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
22   8     1  4.16667E-02 -3.50021E-01  3.50021E-01 -8.68890E-01 -1.18785E+00
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)= 8.68890E-01
 i Sweep Omega    w           mu          eta           xi          aphi
23   8     2  4.16667E-02 -8.68890E-01  3.50021E-01 -3.50021E-01 -7.85398E-01
   P(0)= 1.00000E+00 P(1)=-8.68890E-01
       P(1,1)*COS(1*aphi)=-3.50021E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
 i Sweep Omega    w           mu          eta           xi          aphi
24   8     3  4.16667E-02 -3.50021E-01  8.68890E-01 -3.50021E-01 -3.82950E-01
   P(0)= 1.00000E+00 P(1)=-3.50021E-01
       P(1,1)*COS(1*aphi)=-8.68890E-01
       P(1,1)*SIN(1*aphi)= 3.50021E-01
```

PENDATA: AUTOMATED POST-PROCESSING OF PARALLEL OUTPUT

The *PENDATA* utility has been developed to permit the user to gather and have selective access to any data field generated by a parallel run. In a simple sense, the *PENDATA* utility removes most of the pain of handling large, massively parallel binary and ASCII output data files from *PENTRAN*. *PENDATA* can *automatically* process *all* output files or a single output file. Fully automated data processing progresses by reading parallel storage information from the decomposition mapping table located at the end of any logfile from a parallel run. Automated or single file selection and gather of any output data, including Coarse Mesh Summary data (from ASCII output files) and any binary stored output data, can be performed using *PENDATA*. The opening screen of *PENDATA* offers the following data manipulation choices, where the user must input an option number [1 to 9]:

```
                PENDATA 8.2   SINGLE PRECISION
            Data Post-Processor for PENTRAN Outputs
               Supporting  PENTRAN  Parallel  Sn
                           G. Sjoden
                         A. Haghighat
                      HSW Technologies LLC
                           Mar 2008

 # SCALAR  FLUX Options:
 1 : Get COARSE MESH SUMMARY  Data from one '.1'  OUTPUT FILE
 2 : Get Binary FLUX MOMENT   Data from one '.f1' OUTPUT FILE
 3 : Get Binary SCALAR SOURCE Data from one '.s1' OUTPUT FILE

 4 : Get Logfile & AutoMap PARALLEL COARSE MESH SUMMARY Data
 5 : Get Logfile & AutoMap PARALLEL Binary FLUX MOMENT  Data
 6 : Get Logfile & AutoMap PARALLEL Binary SOURCE       Data

 # ANGULAR FLUX/CURRENT Options:
 7 : Get Binary ANGULAR FLUX Data from one '.a1' OUTPUT FILE
 8 : Get Logfile & AutoMap PARALLEL Binary ANGULAR FLUX Data
 9 : Get Logfile & AutoMap PARALLEL Binary FLUX/CURRENT Data

 For binary outputs (no labels), use -1,-2,-4, or -5
 Enter Option #:
```

A simple, self explanatory menu format follows each option, showing what data fields can be selectively stripped and automatically gathered.

If an automated processing is selected, at least one logfile *(e.g. probname.L1)* ***from a processor participating in the parallel run must be located with all other output files in a common directory*.**

PENDATA will prompt for all required user input, and then report progress as output files are scanned, data is stripped, and collection is made in ASCII table (column) format. The user has freedom to name the output files in *single* file processing.

Automated data output files processed and stored by *PENDATA* based on the post-processing choice number:

- Option 4: *fileprefix*.**crs**
- Option 5: *fileprefix*.**flx**
- Option 6: *fileprefix*.**src**
- Option 8: *fileprefix*.**ang**
- Option 9: *fileprefix*.**fjn**
- **Binary output for all of these variables is possible by entering negative option numbers; no labels of the data will occur in the binary cases, and binary formats mirror formatted data.**

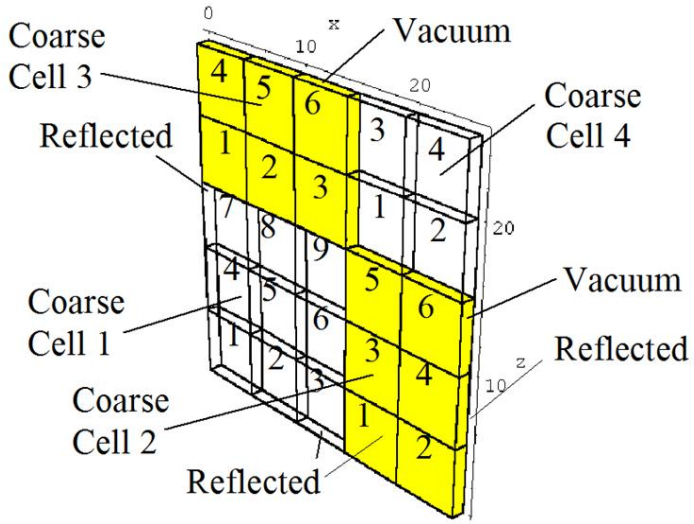
Note: In binary flux moment files (Options (2) or (5)), moments are reported by *PENDATA* beginning with a column entitled "flux moments," with flux moments written in columns in the order of scalar, all cosine, and all sine angular moments. Columns therefore include $\phi_{0,}\phi_1,\phi^1_{C1,}\phi^1_{S1},\phi_2,\phi^1_{C2},\phi^2_{C2},\phi^1_{S2},\phi^2_{S2}$, etc, as given in the theory section, etc.

Note that source files only yield *scalar* sources, and use of a *single precision PENTRAN* version requires a *single precision PENDATA* version for binary data compatability.

6. SAMPLE PROBLEMS

SIMPLE FIXED SOURCE INPUT DECK (FIXED SOURCE PROBLEM)

This is a simple, homogeneous 3-D fixed source problem that is reflected along each boundary of the 1 cm width y-axis to simulate 2-D behavior. The figure below depicts the geometry, and local coarse and fine cell numbering schemes used in *PENTRAN*; cells are alternating in transparency for illustration of the geometry. In addition, there are four coarse mesh cells, with a uniform source placed throughout each coarse mesh in the geometry. The cross sections are included in the input using the "cards" option for illustration.



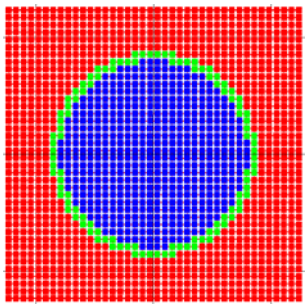

```
PENTRAN CODE PARAMETERS FOR THIS PROBLEM
  maxmem,   maxpcs,   maxgcm,   maxxsg
       80         1         4         0
  maxcmc,   maxcrs,   maxmmc,   maxmed,   maxfmc,   maxfin
        4         2       100       100       100       100
  maxgrp,   maxglc,   maxswp,   maxqdm,   maxmat,   maxleg
        2         2         8        10         1         3
  maxsrc,   maxslc,   maxcmr,   maxlin,   maxarr,   nctlim
        4         4         4       204      4000        20
-----------------Start Problem Deck--------------------
P3 SAMPLE PROBLEM--From M. Hunter     loglevel 3
1
2  For PENTRAN
3  XZ P3 (P1 used) Scattering Problem
4  -5x5 medium mesh, 25.0 cm by 25.0 cm, 2 groups, P3 scatter xs
5  -S8 level symmetric quadrature, vacuum boundaries on front
6   and north bounds, reflective boundaries on back and south
7   bounds, (with reflective on east and west y axis bounds)
8  -Source is fixed uniformly distributed in group 1 only
9  -Supplied by M. Hunter
0
/-----------------------------------------------------
```

```
/  General Notes:
/    --input for columns 1-79 only
/    --EACH FIRST parameter value MUST immediately
/    follow the EQUALS sign, with ANY format thereafter!!
/    --Anything following a slash is ignored as a comment
/    --Block Terminator T MUST be placed on the SAME LINE
/    as the LAST DATA ENTRY for a given block
/  Block 0  (ABOVE:REQUIRED Inputs)
/     1 problem HEADER card
/     10 problem TITLE cards follow
/
/--------------- BLOCK I ----------------------------
   ngeom=3d    /3D geometry
  ngroup=2 1    /2 energy groups
  isn=8
   nmatl=1     /1 material
   ixcrs=2     /2 coarse meshes in x direction
   jycrs=1     /1 coarse meshes in y direction
   kzcrs=2     /2 coarse meshes in z direction
   lodbal=0    /0 No Load balancing
   timcut=0    /0 No wall-time (minutes) cutoff
   tolmgd=-0.015    / multigrid tolerance
   modadj=0    / forward
  decmpv=-1 -1 -1   T   / Decomposition weight vector
/
/--------------- BLOCK II ---------------------------

  xmesh=0.0,15.0,25.0
/  ixmed=2 1 2 1      ixfine=3 2 3 2
    ixmed=3 2 3 2      ixfine=3 2 3 2

  ymesh=0.0,1.0
  jymed=4R1          jyfine=4R1

  zmesh=0.0,15.0,25.0
  kzmed=2 2 1 1      kzfine=3 3 2 2
/   kzmed=3 3 2 2      kzfine=3 3 2 2

  nmattp=1 9R1       nmattp=2 6R1
  nmattp=3 6R1       nmattp=4 4R1

  flxini=4R9.0     mathmg=0 0 0 0    T
/
/-----------------------------------------------------
/   cross section library type--nxtyp
/     0 STANDARD (row) form:  NO, Legendre consts NOT pre-multiplied
/     1 STANDARD (row) form: YES, Legendre consts ARE pre-multiplied
/     2 NON-STD  (col) form:  NO, Legendre consts NOT pre-multiplied
/     3 NON-STD  (col) form: YES, Legendre consts ARE pre-multiplied
/     4 STANDARD (row) BINARY FILE form:  NO, Legendre consts NOT pre-multiplie
/     5 STANDARD (row) BINARY FILE form: YES, Legendre consts ARE pre-multiplie
/     6 NON-STD  (col) BINARY FILE form:  NO, Legendre consts NOT pre-multiplie
/     7 NON-STD  (col) BINARY FILE form: YES, Legendre consts ARE pre-multiplie
/     8 GIP-ORNL       BINARY FILE form: YES, Legendre consts ARE pre-multiplie
/
/ ------------- BLOCK III (CROSS SECTIONS) -----------
/  This problem includes cross sections in the input deck, not as a separate file
   lib=cards
   legord=1   legoxs=3
   nxtyp=0
   ihm=5   iht=3   ihs=4
   ihng=0
   chig=1.0 0.1
   nxcmnt=4    T
/   (slash used in col1 to allow use of upper case)
/   Material 1 Two-Group P0 xsections
/     siga1     nusigf1    sigt1     sig_1->1    --------
/     siga2     nusigf2    sigt2     sig_2->2    sig_1->2
```

```
    1.39306E-02 0.00000 1.22839E-01 5.21631E-02 0.00000
    1.32329E-02 0.00000 1.23884E-01 5.18270E-02 9.05113E-03
/
/   Material 1 Two-Group P1 xsections
/     siga1     nusigf1    sigt1     sig_1->1    --------
/     siga2     nusigf2    sigt2     sig_2->2    sig_1->2
    0.00000E+00 0.00000 0.00000E+00 4.61583E-02 0.00000
    0.00000E+00 0.00000 0.00000E+00 4.49890E-02 4.48667E-03
/
/   Material 1 Two-Group P2 xsections
/     siga1     nusigf1    sigt1     sig_1->1    --------
/     siga2     nusigf2    sigt2     sig_2->2    sig_1->2
    0.00000E+00 0.00000 0.00000E+00 3.94276E-02 0.00000
    0.00000E+00 0.00000 0.00000E+00 3.78602E-02 2.14926E-03
/
/   Material 1 Two-Group P3 xsections
/     siga1     nusigf1    sigt1     sig_1->1    --------
/     siga2     nusigf2    sigt2     sig_2->2    sig_1->2
    0.00000E+00 0.00000 0.00000E+00 3.31849E-02 0.00000
    0.00000E+00 0.00000 0.00000E+00 3.14823E-02 1.58553E-03  T
/            *** END OF XSECTION DATA ***
/
/------------- BLOCK IV (CONTROL OPTIONS) ---------------
  ncoupl=1
  nprtyp=4
  nrdblk=0
   tolin=0.000001
  tolout=0.000001
  maxitr=180
  methit=1   ndmeth=4R-1   nzonrb=4 /1  / nquit=4
  methac=3 T
/
/------------ BLOCK V (Sources) -------------------------
  nsdef=4R0
  nscmsh=1           2          3          4
    sref=7.5 .5 7.5  20 .5 7.5  7.5 .5 20  20 .5 20
    serg=1 0 3Q2
    smag=1.0         1.0        1.0        1.0   T
/
/  this shows how the omegap option is used
/     omegap=1,-1,1,10R1 1,-1,2,10R1
/            1,-1,3,10R1 1,-1,4,10R1
/            1,-1,5,10R1 1,-1,6,10R1
/            1,-1,7,10R1 1,-1,8,10R1  T
/
/------------- BLOCK VI (BOUNDARY CONDITIONS) -----------
/ var   type  GroupAlbedos   var   type GroupAlbedos
  ibback=1    1 1            ibfrnt=0
  jbeast=1    1 1            jbwest=1   1 1
  kbsout=1    1 1            kbnort=0 T
/
/-------------- BLOCK VII (PRINT/OUTPUT CONDITIONS) -------
/
   nxspr=0    nmatpr=1   ngeopr=2
   nsrcpr=0   nsumpr=1   meshpr=1 4
   nfdump=1   nsdump=1   nadump=4,7    T
/
/-----------------------------------------------------
```

WESTINGHOUSE PIN INPUT DECK (K_{EFF} PROBLEM)

This is a pin demonstration problem that was used to benchmark PENBURN, the burnup and depletion module supporting the *PENTRAN* code system. It serves as a good example of a criticality problem.



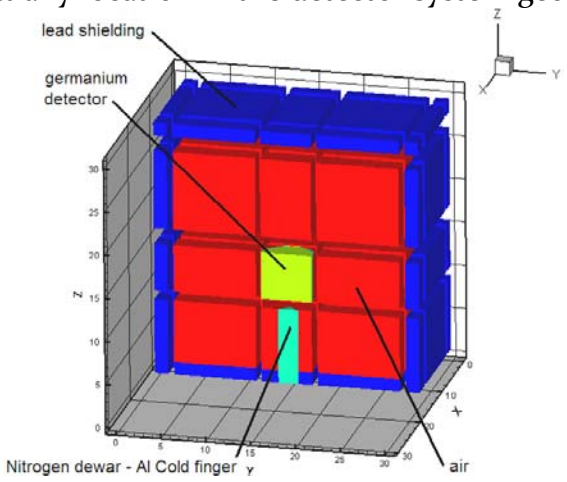
```
   PARAMETERS FOR MEMORY ALLOCATION using F90:
    maxmem,    maxpcs,    maxgcm,    maxxsg
      2000          8          1          3
    maxcmc,    maxcrs,   maxmmc,   maxmed,   maxfmc,   maxfin
         1          1       8000         40       8000        40
    maxgrp,    maxglc,   maxswp,   maxqdm,   maxmat,   maxleg
         3          3          1         10          3          1
    maxsrc,    maxslc,   maxcmr,   maxlin,   maxarr,   nctlim
         0          0          1        189     53400       132
/-----------------Start Problem Deck---------------
whpin                                                    loglevel 2
generated by PENMSHXP version 1.5a
Total Number of Fine Meshes:  8000
Total Number of Coarse Meshes:  1
Number of zlevs:  1
Number of coarse mesh per z lev: 1
6
7
8
9
10
/
/-------------BLOCK I (GENERAL PROBLEM info.)-----------
/
ngeom=3d
modadj=0
ngroup=3
isn=8
nmatl=3
ixcrs=1
jycrs=1
kzcrs=1
lodbal=0
timcut=0.
tolmgd=-0.200
decmpv=-8 -1 -1  T
/
/-----------------BLOCK II(geometry)-----------------
/
/ x coarse-mesh position
/
xmesh=-6.3250E-01  6.3250E-01
/
/ x fine mesh distribution for zlev=  1
```

```
/
ixfine=40
/
/ x medium mesh distribution for zlev=  1
/
ixmed=40
/
/ y coarse-mesh position
/
ymesh=-6.3250E-01  6.3250E-01
/
/ y fine mesh distribution for zlev=  1
/
jyfine=40
/
/ y medium mesh distribution for zlev=  1
/
jymed=40
/
/ z coarse-mesh position
/
zmesh= 0.0000E+00  1.0000E+01
/
/ z fine mesh distribution for zlev=  1
/
kzfine=5
/
/ z medium mesh distribution for zlev=  1
/
kzmed=5
/
/  material distribution for zlev=  1
/
    nmattp=1
257R3 6R2 31R3 4R2 4R1 4R2 26R3 3R2 10R1 3R2 23R3 2R2 14R1 2R2 21R3 2R2 16R1
2R2 19R3 2R2 18R1 2R2 17R3 2R2 20R1 2R2 16R3 2 22R1 2 15R3 2R2 22R1 2R2 14R3 2
24R1 1Q40 2 13R3 2R2 24R1 2R2 12R3 2 26R1 3Q40 2 12R3 2R2 24R1 2R2 13R3 2 24R1
2 14R3 1Q40 2R2 22R1 2R2 15R3 2 22R1 2 16R3 2R2 20R1 2R2 17R3 2R2 18R1 2R2
19R3 2R2 16R1 2R2 21R3 2R2 14R1 2R2 23R3 3R2 10R1 3R2 26R3 4R2 4R1 4R2 31R3
6R2 257R3 4Q1600
flxini=1000.000
mathmg=0    T
/
/ ------------- BLOCK III (CROSS SECTIONS) -----------
/
lib=file:whpin.xs
legord=1  legoxs=1
nxtyp=1
ihm=6
iht=3  ihs=4
ihng=0
/
/    Material   1  uo2              Chi Values
/    For whpin.xs        Grps   1 -  3  ntemp = t
chig=6.93859E-01 3.06141E-01 2.12672E-10
/
/    Material   2  zr               Chi Values
/    For whpin.xs        Grps   1 -  3  ntemp = t
 0.00000E+00 0.00000E+00 0.00000E+00
/
/    Material   3  h2o              Chi Values
/    For whpin.xs        Grps   1 -  3  ntemp = t
 0.00000E+00 0.00000E+00 0.00000E+00
nxcmnt=2    T
/
/------------- BLOCK IV (CONTROL OPTIONS) --------------
/
ncoupl=1
```

```
nprtyp=0
nrdblk=0
tolin=5.00E-05
tolout=5.00E-05
dtwmxw=0.95
maxitr=200    20
methit=1
/
/  Starting or selected differencing scheme,for each coarse-mesh, for z-level=   1
/ penmsh default is 2
ndmeth=4
nzonrb=1  0.999  0
methac=1      T
/------------------BLOCK V(source)------------------
/
rkdef=1.000   T
/
/------------- BLOCK VI (BOUNDARY CONDITIONS) ---------
/
/ var   type  Group albedos
ibback=1   3R1
ibfrnt=1   3R1
jbeast=1   3R1
jbwest=1   3R1
kbsout=1   3R1
kbnort=1   3R1        T
/
/------------- BLOCK VII (PRINTING CONDITIONS) ---------
/
/
nxspr=0 nmatpr=1 ngeopr=1 nsrcpr=0 nsumpr=1
meshpr=-1
nfdump=2 nsdump=1 njdump=0
nadump=0     T
```

GERMANIUM 'CARDD' DETECTOR ADJOINT (WITH PENMSH-XP INPUT FILES)

For this problem we have included the penmsh-xp automatic mesh generation utility files, which build a problem mesh. The problem is illustrated in Figure 2.29 of this document. The problem is a detector adjoint that will yield the inherent detector efficiency of any gamma ray in the system at any location in the detector system geometry.



The details of automatic problem input deck generation and source projection are best presented in the *PENMSH-XP* manual; however, for illustration, we have included the minimal inputs required for the *PENMSH-XP* software. In only a few minutes, one can build a 3-D model and render graphics slices using the *DISLIN* library (generating .png files) and a 3-D geometry plot (set up for the TecPlot™ Graphics package). A material balance feature, scaling, source placement, 3-D object placement, etc are all available in the *PENMSH-XP* utility for use with *PENTRAN*.

The following files are processed by *PENMSH-XP* and represent the model called 'CARDD':

'PENMSH.INP' file: a file detailing an overall system geometry and parameters
'CARDD1.INP': file describing the first z-level mesh structure for a model
'CARDD2.INP': file describing the second z-level mesh structure for a model
'CARDD3.INP': file describing the third z-level mesh structure for a model
'CARDD4.INP': file describing the fourth z-level mesh structure for a model
'CARDD.SPC': file describing the energy spectrum for 90 gamma energy groups

When processed by *PENMSH-XP*, these files create a 'CARDD.f90' file, which is a *PENTRAN* input deck that can be further editied for specific user-desired settings.

PENMSH.INP FILE FOR 'CARDD' MODEL

```
cardd
/no. coarse z-levels, no. of materials, imath
4,4,0
/z-level coarse mesh boundaries
0., 10.0, 16.0, 28.0, 30.0
/max. number of fine z-mesh per coarse z-level
16 16 16 16 16
/fine-to-med grid ratio along x.., y.., z.. in each coarse z-level
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
/source format, # x-src mesh, # y-src mesh, # z-src mesh, ngrp, sn, pn
-1,1,1,1,90,8,1
/xsec type, xsec #comment cards, xsec Leg order, ihm
2,2,1,93
/Bdy conds: ibback(-x),ibfrnt(+x),jbeast(-y),jbwest(+y),kbsout(-z),kbnort(+z)
0,0,0,0,0,0
/source material id (if iso <0, need that number of sources)
3
/source magnitude(s)
1.0
```

1ST Z-LEVEL FOR PENMSH-XP UTILITY: CARDD1.INP

```
/ncx, ncy, maxfinz  (maxfinz < 0, add z-fine per cm below y-fine)
5, 5,-16
/ x-fine mesh per cm (# seq along cm rows of x)
 4 12 10 12 4
 4 12 10 12 4
 4 12 16 12 4
 4 12 10 12 4
 4 12 10 12 4
/ y-fine mesh per cm (# seq ... y)
4 4 4 4 4
12 12 12 12 12
10 10 16 10 10
12 12 12 12 12
4 4 4 4 4
/ z-fine mesh per cm (# seq ... z)
 7  7  7  7  7
 7  7  7  7  7
 7  7  7  7  7
 7  7  7  7  7
 7  7  7  7  7
/ cm bounds along x-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/cm bounds along y-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/ cm type, each cm (<0 =look for overlay)
 1 1 1 1 1
 1 -1 -1 -1 1
 1 -1 -1 -1 1
 1 -1 -1 -1 1
 1 1 1 1 1
/ number of matl regns per cm;
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
/material id for coarse mesh 1
1
/material id for coarse mesh 2
1
/material id for coarse mesh 3
1
/material id for coarse mesh 4
1
/material id for coarse mesh 5
1
/material id for coarse mesh 6
1
/material id for coarse mesh 7
4
/material id for coarse mesh 8
4
/material id for coarse mesh 9
4
/material id for coarse mesh 10
1
```

```
/material id for coarse mesh 11
1
/material id for coarse mesh 12
4
/material id for coarse mesh 13
4
/material id for coarse mesh 14
4
/material id for coarse mesh 15
1
/material id for coarse mesh 16
1
/material id for coarse mesh 17
4
/material id for coarse mesh 18
4
/material id for coarse mesh 19
4
/material id for coarse mesh 20
1
/material id for coarse mesh 21
1
/material id for coarse mesh 22
1
/material id for coarse mesh 23
1
/material id for coarse mesh 24
1
/material id for coarse mesh 25
1
/ overlay cm 7 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1.., yhi1..)
1
1
1
0., 30., 0., 30.0, 0.0, 2.0
/ overlay cm 8 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1.., yhi1..)
1
1
1
0., 30., 0., 30.0, 0.0, 2.0
/ overlay cm 9 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1.., yhi1..)
1
1
1
0., 30., 0., 30.0, 0.0, 2.0
/ overlay cm 12 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1..,
yhi1..)
1
1
1
0., 30., 0., 30.0, 0.0, 2.0
/ overlay cm 13 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1..,
yhi1..)
2
1 2
1 2
0., 30., 0., 30.0, 0.0, 2.0
14., 16., 14., 16.0
/ overlay cm 14 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1..,
yhi1..)
1
```

```
1
1
0., 30., 0., 30.0, 0.0, 2.0
/ overlay cm 17 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1..,
yhi1..)
1
1
1
0., 30., 0., 30.0, 0.0, 2.0
/ overlay cm 18 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1..,
yhi1..)
1
1
1
0., 30., 0., 30.0, 0.0, 2.0
/ overlay cm 19 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1..,
yhi1..)
1
1
1
0., 30., 0., 30.0, 0.0, 2.0
```

2ND Z-LEVEL FOR PENMSH-XP UTILITY: CARDD2.INP

```
/ncx, ncy, maxfinz  (maxfinz < 0, add z-fine per cm below y-fine)
5, 5,-16
/ x-fine mesh per cm (# seq along cm rows of x)
 4 12 10 12 4
 4 12 10 12 4
 4 12 16 12 4
 4 12 10 12 4
 4 12 10 12 4
/ y-fine mesh per cm (# seq ... y)
4 4 4 4 4
12 12 12 12 12
10 10 16 10 10
12 12 12 12 12
4 4 4 4 4
/ z-fine mesh per cm (# seq ... z)
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
/ cm bounds along x-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/cm bounds along y-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/ cm type, each cm (<0 =look for overlay)
 1 1 1 1 1
 1 1 1 1 1
 1 1 -1 1 1
 1 1 1 1 1
 1 1 1 1 1
/ number of matl regns per cm;
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
/material id for coarse mesh 26
1
/material id for coarse mesh 27
1
/material id for coarse mesh 28
1
/material id for coarse mesh 29
1
/material id for coarse mesh 30
1
/material id for coarse mesh 31
1
/material id for coarse mesh 32
4
/material id for coarse mesh 33
4
/material id for coarse mesh 34
4
/material id for coarse mesh 35
1
/material id for coarse mesh 36
1
/material id for coarse mesh 37
4
/material id for coarse mesh 38
4
/material id for coarse mesh 39
```

```
4
/material id for coarse mesh 40
1
/material id for coarse mesh 41
1
/material id for coarse mesh 42
4
/material id for coarse mesh 43
4
/material id for coarse mesh 44
4
/material id for coarse mesh 45
1
/material id for coarse mesh 46
1
/material id for coarse mesh 47
1
/material id for coarse mesh 48
1
/material id for coarse mesh 49
1
/material id for coarse mesh 50
1
/ overlay cm 38 (#over, shape1 4=sphere, mat#1.., xlo1.., xhi1.., ylo1.., yhi1)
2
2 2
2 3
12., 18., 12., 18.0
12.1, 17.9, 12.1, 17.9
```

3RD Z-LEVEL FOR PENMSH-XP UTILITY: CARDD3.INP

```
/ncx, ncy, maxfinz  (maxfinz < 0, add z-fine per cm below y-fine)
5, 5,-16
/ x-fine mesh per cm (# seq along cm rows of x)
 4 12 10 12 4
 4 12 10 12 4
 4 12 16 12 4
 4 12 10 12 4
 4 12 10 12 4
/ y-fine mesh per cm (# seq ... y)
4 4 4 4 4
12 12 12 12 12
10 10 16 10 10
12 12 12 12 12
4 4 4 4 4
/ z-fine mesh per cm (# seq ... z)
 4  4  4  4  4
 4  4  4  4  4
 4  4  4  4  4
 4  4  4  4  4
 4  4  4  4  4
/ cm bounds along x-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/cm bounds along y-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/ cm type, each cm (<0 =look for overlay)
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
/ number of matl regns per cm;
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
/material id for coarse mesh 51
1
/material id for coarse mesh 52
1
/material id for coarse mesh 53
1
/material id for coarse mesh 54
1
/material id for coarse mesh 55
1
/material id for coarse mesh 56
1
/material id for coarse mesh 57
4
/material id for coarse mesh 58
4
/material id for coarse mesh 59
4
/material id for coarse mesh 60
1
/material id for coarse mesh 61
1
/material id for coarse mesh 62
4
/material id for coarse mesh 63
4
/material id for coarse mesh 64
```

```
4
/material id for coarse mesh 65
1
/material id for coarse mesh 66
1
/material id for coarse mesh 67
4
/material id for coarse mesh 68
4
/material id for coarse mesh 69
4
/material id for coarse mesh 70
1
/material id for coarse mesh 71
1
/material id for coarse mesh 72
1
/material id for coarse mesh 73
1
/material id for coarse mesh 74
1
/material id for coarse mesh 75
1
```

4TH Z-LEVEL FOR PENMSH-XP UTILITY: CARDD4.INP

```
/ncx, ncy, maxfinz  (maxfinz < 0, add z-fine per cm below y-fine)
5, 5,-16
/ x-fine mesh per cm (# seq along cm rows of x)
 4 12 10 12 4
 4 12 10 12 4
 4 12 16 12 4
 4 12 10 12 4
 4 12 10 12 4
/ y-fine mesh per cm (# seq ... y)
4 4 4 4 4
12 12 12 12 12
10 10 16 10 10
12 12 12 12 12
4 4 4 4 4
/ z-fine mesh per cm (# seq ... z)
 4  4  4  4  4
 4  4  4  4  4
 4  4  4  4  4
 4  4  4  4  4
 4  4  4  4  4
/ cm bounds along x-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/cm bounds along y-axis (in seq ...x)
0., 2.0, 12.0, 18.0, 28.0, 30.0
/ cm type, each cm (<0 =look for overlay)
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
 1 1 1 1 1
/ number of matl regns per cm;
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
/material id for coarse mesh 1
1
/material id for coarse mesh 2
1
/material id for coarse mesh 3
1
/material id for coarse mesh 4
1
/material id for coarse mesh 5
1
/material id for coarse mesh 6
1
/material id for coarse mesh 7
1
/material id for coarse mesh 8
1
/material id for coarse mesh 9
1
/material id for coarse mesh 10
1
/material id for coarse mesh 11
1
/material id for coarse mesh 12
1
/material id for coarse mesh 13
1
/material id for coarse mesh 14
```

```
1
/material id for coarse mesh 15
1
/material id for coarse mesh 16
1
/material id for coarse mesh 17
1
/material id for coarse mesh 18
1
/material id for coarse mesh 19
1
/material id for coarse mesh 20
1
/material id for coarse mesh 21
1
/material id for coarse mesh 22
1
/material id for coarse mesh 23
1
/material id for coarse mesh 24
1
/material id for coarse mesh 25
1
```

```
    12.755700
     7.528220
     4.897690
     3.361800
     2.401790
     1.771920
     1.343690
     1.042930
8.256040E-01
6.647150E-01
5.431070E-01
4.495090E-01
3.763110E-01
3.182580E-01
2.716340E-01
2.337620E-01
2.026860E-01
1.769500E-01
1.554560E-01
1.373650E-01
1.220310E-01
1.089500E-01
9.771730E-02
8.802300E-02
7.961080E-02
7.228340E-02
6.584970E-02
6.019590E-02
5.519450E-02
5.076640E-02
4.682600E-02
4.330530E-02
4.015610E-02
3.732440E-02
3.477280E-02
3.246720E-02
3.037850E-02
2.847840E-02
2.675290E-02
2.517780E-02
2.373680E-02
2.241620E-02
2.120150E-02
2.008880E-02
1.905990E-02
1.810910E-02
1.702390E-02
1.602650E-02
1.344990E-02
1.133660E-02
1.059710E-02
9.572350E-03
8.618260E-03
8.055610E-03
7.800980E-03
7.440710E-03
7.103430E-03
6.892680E-03
6.595830E-03
6.316800E-03
6.141790E-03
5.972450E-03
5.663070E-03
```

```
5.373160E-03
5.238050E-03
4.985470E-03
4.639900E-03
4.430850E-03
4.232520E-03
3.944120E-03
3.764140E-03
3.651710E-03
3.544880E-03
3.432820E-03
3.331390E-03
3.190230E-03
3.074720E-03
2.988750E-03
2.932340E-03
3.060820E-03
3.419640E-03
3.698670E-03
4.039550E-03
5.392230E-03
6.845250E-03
7.331700E-03
1.008930E-02
1.275910E-02
1.559440E-02
1.824990E-02
```

```
PENTRAN CODE PARAMETERS FOR THIS PROBLEM
  maxmem,  maxpcs,  maxgcm,  maxxsg
    2100        8      100       90
  maxcmc,  maxcrs,  maxmmc,  maxmed,  maxfmc,  maxfin
      25        5     2560       32     2560       64
  maxgrp,  maxglc,  maxswp,  maxqdm,  maxmat,  maxleg
      90        1        4      136        4        1
  maxsrc,  maxslc,  maxcmr,  maxlin,  maxarr,  nctlim
       1        1      100      611  2304000    10000
-----------------Start Problem Deck-------------------
cardd                                                    loglevel 2
generated by PENMSHXP version 1.5a
Total Number of Fine Meshes:  48000
Total Number of Coarse Meshes:  100
Number of zlevs:  4
Number of coarse mesh per z lev: 25
6
7
8
9
10
/
/-------------BLOCK I (GENERAL PROBLEM info.)-----------
/
ngeom=3d
modadj=1
ngroup=90    1
isn=-30
nmatl=4
ixcrs=5
jycrs=5
kzcrs=4
lodbal=0
timcut=0.
tolmgd=-0.200
decmpv=-2 -1 -4  T
/
/------------------BLOCK II(geometry)------------------
/
/ x coarse-mesh position
/
xmesh= 0.0000E+00  2.0000E+00  1.2000E+01  1.8000E+01  2.8000E+01  3.0000E+01
/
/ x fine mesh distribution for zlev=  1
/
ixfine=4 12 10 12 4
       4 12 10 12 4
       4 12 16 12 4
       4 12 10 12 4
       4 12 10 12 4
/
/ x fine mesh distribution for zlev=  2
/
       4 12 10 12 4
       4 12 10 12 4
       4 12 16 12 4
       4 12 10 12 4
       4 12 10 12 4
/
/ x fine mesh distribution for zlev=  3
/
       4 12 10 12 4
       4 12 10 12 4
       4 12 16 12 4
       4 12 10 12 4
```

```
       4 12 10 12 4
/
/ x fine mesh distribution for zlev=  4
/
       4 12 10 12 4
       4 12 10 12 4
       4 12 16 12 4
       4 12 10 12 4
       4 12 10 12 4
/
/ x medium mesh distribution for zlev=  1
/
ixmed=2 6 5 6 2
      2 6 5 6 2
      2 6 8 6 2
      2 6 5 6 2
      2 6 5 6 2
/
/ x medium mesh distribution for zlev=  2
/
      2 6 5 6 2
      2 6 5 6 2
      2 6 8 6 2
      2 6 5 6 2
      2 6 5 6 2
/
/ x medium mesh distribution for zlev=  3
/
      2 6 5 6 2
      2 6 5 6 2
      2 6 8 6 2
      2 6 5 6 2
      2 6 5 6 2
/
/ x medium mesh distribution for zlev=  4
/
      2 6 5 6 2
      2 6 5 6 2
      2 6 8 6 2
      2 6 5 6 2
      2 6 5 6 2
/
/ y coarse-mesh position
/
ymesh= 0.0000E+00  2.0000E+00  1.2000E+01  1.8000E+01  2.8000E+01  3.0000E+01
/
/ y fine mesh distribution for zlev=  1
/
jyfine=4 4 4 4 4
       12 12 12 12 12
       10 10 16 10 10
       12 12 12 12 12
       4 4 4 4 4
/
/ y fine mesh distribution for zlev=  2
/
       4 4 4 4 4
       12 12 12 12 12
       10 10 16 10 10
       12 12 12 12 12
       4 4 4 4 4
/
/ y fine mesh distribution for zlev=  3
/
       4 4 4 4 4
       12 12 12 12 12
       10 10 16 10 10
       12 12 12 12 12
```

```
       4 4 4 4 4
/
/ y fine mesh distribution for zlev=  4
/
       4 4 4 4 4
       12 12 12 12 12
       10 10 16 10 10
       12 12 12 12 12
       4 4 4 4 4
/
/ y medium mesh distribution for zlev=  1
/
jymed=2 2 2 2 2
      6 6 6 6 6
      5 5 8 5 5
      6 6 6 6 6
      2 2 2 2 2
/
/ y medium mesh distribution for zlev=  2
/
      2 2 2 2 2
      6 6 6 6 6
      5 5 8 5 5
      6 6 6 6 6
      2 2 2 2 2
/
/ y medium mesh distribution for zlev=  3
/
      2 2 2 2 2
      6 6 6 6 6
      5 5 8 5 5
      6 6 6 6 6
      2 2 2 2 2
/
/ y medium mesh distribution for zlev=  4
/
      2 2 2 2 2
      6 6 6 6 6
      5 5 8 5 5
      6 6 6 6 6
      2 2 2 2 2
/
/ z coarse-mesh position
/
zmesh= 0.0000E+00  1.0000E+01  1.6000E+01  2.8000E+01  3.0000E+01
/
/ z fine mesh distribution for zlev=  1
/
kzfine=7 7 7 7 7
       7 7 7 7 7
       7 7 7 7 7
       7 7 7 7 7
       7 7 7 7 7
/
/ z fine mesh distribution for zlev=  2
/
       10 10 10 10 10
       10 10 10 10 10
       10 10 10 10 10
       10 10 10 10 10
       10 10 10 10 10
/
/ z fine mesh distribution for zlev=  3
/
       4 4 4 4 4
       4 4 4 4 4
       4 4 4 4 4
       4 4 4 4 4
```

```
       4 4 4 4 4
/
/ z fine mesh distribution for zlev=  4
/
       4 4 4 4 4
       4 4 4 4 4
       4 4 4 4 4
       4 4 4 4 4
       4 4 4 4 4
/
/ z medium mesh distribution for zlev=  1
/
kzmed=3 3 3 3 3
      3 3 3 3 3
      3 3 3 3 3
      3 3 3 3 3
      3 3 3 3 3
/
/ z medium mesh distribution for zlev=  2
/
      5 5 5 5 5
      5 5 5 5 5
      5 5 5 5 5
      5 5 5 5 5
      5 5 5 5 5
/
/ z medium mesh distribution for zlev=  3
/
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
/
/ z medium mesh distribution for zlev=  4
/
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
      2 2 2 2 2
/
/  material distribution for zlev=  1
/
    nmattp=1
112R1
    nmattp=2
336R1
    nmattp=3
280R1
    nmattp=4
336R1
    nmattp=5
112R1
    nmattp=6
336R1
    nmattp=7
144R1 864R4
    nmattp=8
120R1 720R4
    nmattp=9
144R1 864R4
    nmattp=10
336R1
    nmattp=11
280R1
    nmattp=12
120R1 720R4
```

```
    nmattp=13
87R1 2R2 13R1 4R2 11R1 6R2 10R1 6R2 11R1 4R2 13R1 2R2 87R1 87R4 2R2 13R4 4R2
11R4 6R2 10R4 6R2 11R4 4R2 13R4 2R2 174R4 4Q256 2R2 13R4 4R2 11R4 6R2 10R4 6R2
11R4 4R2 13R4 2R2 87R4
    nmattp=14
120R1 720R4
    nmattp=15
280R1
    nmattp=16
336R1
    nmattp=17
144R1 864R4
    nmattp=18
120R1 720R4
    nmattp=19
144R1 864R4
    nmattp=20
336R1
    nmattp=21
112R1
    nmattp=22
336R1
    nmattp=23
280R1
    nmattp=24
336R1
    nmattp=25
112R1
/  material distribution for zlev=  2
/
    nmattp=26
160R1
    nmattp=27
480R1
    nmattp=28
400R1
    nmattp=29
480R1
    nmattp=30
160R1
    nmattp=31
480R1
    nmattp=32
1440R4
    nmattp=33
1200R4
    nmattp=34
1440R4
    nmattp=35
480R1
    nmattp=36
400R1
    nmattp=37
1200R4
    nmattp=38
5R4 2 4R3 2 8R4 2 8R3 2 5R4 2 10R3 2 3R4 2 12R3 2 2R4 14R3 4 2 14R3 2 64R3 2
14R3 2 4 14R3 2R4 2 12R3 2 3R4 2 10R3 2 5R4 2 8R3 2 8R4 2 4R3 2 5R4 9Q256
    nmattp=39
1200R4
    nmattp=40
400R1
    nmattp=41
480R1
    nmattp=42
1440R4
    nmattp=43
1200R4
    nmattp=44
```

```
1440R4
    nmattp=45
480R1
    nmattp=46
160R1
    nmattp=47
480R1
    nmattp=48
400R1
    nmattp=49
480R1
    nmattp=50
160R1
/  material distribution for zlev=  3
/
    nmattp=51
64R1
    nmattp=52
192R1
    nmattp=53
160R1
    nmattp=54
192R1
    nmattp=55
64R1
    nmattp=56
192R1
    nmattp=57
576R4
    nmattp=58
480R4
    nmattp=59
576R4
    nmattp=60
192R1
    nmattp=61
160R1
    nmattp=62
480R4
    nmattp=63
1024R4
    nmattp=64
480R4
    nmattp=65
160R1
    nmattp=66
192R1
    nmattp=67
576R4
    nmattp=68
480R4
    nmattp=69
576R4
    nmattp=70
192R1
    nmattp=71
64R1
    nmattp=72
192R1
    nmattp=73
160R1
    nmattp=74
192R1
    nmattp=75
64R1
/  material distribution for zlev=  4
/
    nmattp=76
```

```
64R1
    nmattp=77
192R1
    nmattp=78
160R1
    nmattp=79
192R1
    nmattp=80
64R1
    nmattp=81
192R1
    nmattp=82
576R1
    nmattp=83
480R1
    nmattp=84
576R1
    nmattp=85
192R1
    nmattp=86
160R1
    nmattp=87
480R1
    nmattp=88
1024R1
    nmattp=89
480R1
    nmattp=90
160R1
    nmattp=91
192R1
    nmattp=92
576R1
    nmattp=93
480R1
    nmattp=94
576R1
    nmattp=95
192R1
    nmattp=96
64R1
    nmattp=97
192R1
    nmattp=98
160R1
    nmattp=99
192R1
    nmattp=100
64R1
flxini=37R0.00 1.0 62R0.00
mathmg=100R0    T
/
/ ------------- BLOCK III (CROSS SECTIONS) -----------
/
lib=file:cardd.xs
legord=1  legoxs=1
nxtyp=2
ihm=93
iht=3  ihs=4
ihng=0
chig=1.0000E+00 89R0.0000E+00 3Q90
nxcmnt=2    T
/
/------------- BLOCK IV (CONTROL OPTIONS) --------------
/
ncoupl=1
nprtyp=1
nrdblk=0
```

```
tolin=1.00E-03
tolout=1.00E-05
dtwmxw=0.95
maxitr=50    10
methit=1
/
/  Starting or selected differencing scheme,for each coarse-mesh, for z-level=
/
ndmeth=2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
/
/  Starting or selected differencing scheme,for each coarse-mesh, for z-level=
/
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
/
/  Starting or selected differencing scheme,for each coarse-mesh, for z-level=
/
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
/
/  Starting or selected differencing scheme,for each coarse-mesh, for z-level=
/
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
       2 2 2 2 2
nzonrb=100  0.999  1
methac=3      T
/-----------------BLOCK V(source)-----------------
/
nsdef=0
nscmsh=38
sref=3R0.000
serg=1.276E+01 7.528E+00 4.898E+00 3.362E+00 2.402E+00 1.772E+00 1.344E+00
1.043E+00 8.256E-01 6.647E-01 5.431E-01 4.495E-01 3.763E-01 3.183E-01
2.716E-01 2.338E-01 2.027E-01 1.769E-01 1.555E-01 1.374E-01 1.220E-01
1.089E-01 9.772E-02 8.802E-02 7.961E-02 7.228E-02 6.585E-02 6.020E-02
5.519E-02 5.077E-02 4.683E-02 4.331E-02 4.016E-02 3.732E-02 3.477E-02
3.247E-02 3.038E-02 2.848E-02 2.675E-02 2.518E-02 2.374E-02 2.242E-02
2.120E-02 2.009E-02 1.906E-02 1.811E-02 1.702E-02 1.603E-02 1.345E-02
1.134E-02 1.060E-02 9.572E-03 8.618E-03 8.056E-03 7.801E-03 7.441E-03
7.103E-03 6.893E-03 6.596E-03 6.317E-03 6.142E-03 5.972E-03 5.663E-03
5.373E-03 5.238E-03 4.985E-03 4.640E-03 4.431E-03 4.233E-03 3.944E-03
3.764E-03 3.652E-03 3.545E-03 3.433E-03 3.331E-03 3.190E-03 3.075E-03
2.989E-03 2.932E-03 3.061E-03 3.420E-03 3.699E-03 4.040E-03 5.392E-03
6.845E-03 7.332E-03 1.009E-02 1.276E-02 1.559E-02 1.825E-02
smag=1.88000E+03
spacpf=1   -1        2560
6R0.00000E+00 4R5.31915E-04 10R0.00000E+00 8R5.31915E-04 7R0.00000E+00
10R5.31915E-04 5R0.00000E+00 12R5.31915E-04 3R0.00000E+00 14R5.31915E-04
2R0.00000E+00 14R5.31915E-04 0.00000E+00 64R5.31915E-04 0.00000E+00
14R5.31915E-04 2R0.00000E+00 14R5.31915E-04 3R0.00000E+00 12R5.31915E-04
5R0.00000E+00 10R5.31915E-04 7R0.00000E+00 8R5.31915E-04 10R0.00000E+00
4R5.31915E-04 12R0.00000E+00 8Q256 4R5.31915E-04 10R0.00000E+00 8R5.31915E-04
7R0.00000E+00 10R5.31915E-04 5R0.00000E+00 12R5.31915E-04 3R0.00000E+00
14R5.31915E-04 2R0.00000E+00 14R5.31915E-04 0.00000E+00 64R5.31915E-04
0.00000E+00 14R5.31915E-04 2R0.00000E+00 14R5.31915E-04 3R0.00000E+00
```

```
12R5.31915E-04 5R0.00000E+00 10R5.31915E-04 7R0.00000E+00 8R5.31915E-04
10R0.00000E+00 4R5.31915E-04 6R0.00000E+00   T
/
/------------- BLOCK VI (BOUNDARY CONDITIONS) ---------
/
/ var   type  Group albedos
ibback=0
ibfrnt=0
jbeast=0
jbwest=0
kbsout=0
kbnort=0    T
/
/------------- BLOCK VII (PRINTING CONDITIONS) ---------
/
/
nxspr=0 nmatpr=1 ngeopr=1 nsrcpr=0 nsumpr=1
meshpr=98I-1 -100
nfdump=1 nsdump=1 njdump=0 nadump=0   T
```

PARALLEL LINUX EXECUTION SCRIPT FOR *BASH*

Below is the *PENTRAN* parallel job script for execution on Linux clusters running a bash command shell.

Example: To run a problem called *fuelcask.f90* on 8 machines, the command line is: "ppen fuelcask f90 8" (note the '.' Between the prefix and suffix is not included).

```
#!/bin/sh
#
# ppen parallel PENTRAN job script
#
# G. Sjoden, Sep 2001, June 2008
# B. Dionne, June 2004
#
# syntax: 'ppen problemfile 3LtrSuffix #procs'
#                   $1          $2       $3
#
#    problemfile.3LtrSuffix contains PENTRAN deck
#    #procs is the number of processors used
#
#    For Declaring Executable locations
# ------------------------------------------------
#
#
     PENTRANcode=/home/sjoden/pentran/penmpf90
#
#
#    For Converting Files as needed
# ------------------------------------------------
echo
echo Running PARALLEL PENTRAN on $3 Processors ...
echo $1 files converted to prb prefix scratch files in run ...
echo
#    lamclean -v
     cp $1.$2 prb.pen
     sleep 1
#
#    Document MPI BUFFERING OPTIONS
#------------------------------------------------
#    --Standard MPI
     mpirun           -np  $3 $PENTRANcode
#
#    --Medium Buffer Client MPI
#    obsolete: mpirun -c2c -nger -c  $3 $PENTRANcode
#    mpirun -ssi rpi tcp -nger -c  $3 $PENTRANcode
#
#    --High Buffer MPI (slower but better buffering)
#    mpirun -lamd -nger -c $3 $PENTRANcode
#
#    For Executing Code Sequences
# ------------------------------------------------
```

```
#
#    mpirun           -np  $3 $PENTRANcode
#    obsolete: mpirun -c2c -nger -c  $3 $PENTRANcode
#    mpirun -ssi rpi tcp -nger -c  $3 $PENTRANcode
#    mpirun -lamd -nger -c $3 $PENTRANcode
echo
echo PARALLEL PENTRAN Finished.
#
#    For Converting prb files back to $1 files
# ------------------------------------------------
echo
echo Removing Scratch files *.?at
echo prb.pen ...
echo
     sleep 1
     cp prb.pen $1.$2
     rm *.?at prb.pen
#    lamclean -v
echo
echo Cleaning Up Parallel Job files ...
echo
#
# For Cleaning Up Processor : unique references
# ------------------------------------------------
#
proc=1
while [ $proc -le $3 ]
do
#
if [ -f ssnsrcp   ];
then
  sed 's/prb/'$1'/g' ssnsrcp > ssnsrcpnew
  mv ssnsrcpnew ssnsrcp
fi
#
if [ -f prb.$proc ];
then
  sed 's/prb/'$1'/g' prb.$proc > $1.$proc
  rm prb.$proc
fi
#
if [ -f prb.L$proc ];
then
  sed 's/prb/'$1'/g' prb.L$proc > $1.L$proc
  rm prb.L$proc
fi
#
if [ -f prb.f$proc ];
then
  mv prb.f$proc $1.f$proc
fi
#
if [ -f prb.s$proc ];
then
  mv prb.s$proc $1.s$proc
fi
#
if [ -f prb.j$proc ];
```

```
then
  mv prb.j$proc $1.j$proc
fi
#
if [ -f prb.a$proc ];
then
  mv prb.a$proc $1.a$proc
fi
proc=`expr $proc + 1`
done
#
if [ -f prb.M1 ];
then
  mv prb.M1 $1.M1
fi
#
if [ -f prb.K1 ];
then
  sed 's/prb/'$1'/g' prb.K1 > $1.K1
rm prb.K1
fi
#
if [ -f prb.geo ];
then
  sed 's/prb/'$1'/g' prb.geo > $1.geo
rm prb.geo
fi
# ------------------------------------------------------
#
echo
echo Job $1 Complete
echo
```

7. REFERENCES

Alcouffe, R., E. Larsen, W. Miller, and B. Wienke, "Computational Efficiency of Numerical Methods for the Multigroup, Discrete Ordinates Neutron Transport Equations: The Slab Geometry Case," ***Nuclear Science and Engineering***, 71:111-127 (1979).

Barbieri, K., "Introduction to the SP2 at CTC," ***Proc Conference on Parallel Programming on the IBM SP-2***, Cornell Theory Center, Cornell University, Ithaca, N.Y. (1995).

Brown, J. F., and A. Haghighat, "A *PENTRAN* Model for a Medical Computed Tomography Device," ***Proc of the 10th International Meeting on Radiation Protection and Shielding (RPS2000)***, Spokane, WA (2000)

Barbucci, P., and F. DiPasquantonio, "Exponential Supplementary Equations for Sn Methods: The One-Dimensional Case," ***Nuclear Science and Engineering***, 63: 179-187, (1977).

Barnett, A., J. Morel and D. Harris, "A Multigrid Acceleration Method for the One-Dimensional SN Equations with Anisotropic Scattering," ***Nuclear Science and Engineering***, 102: 1 (1989).

Bell, G. and S. Glasstone. ***Nuclear Reactor Theory***, Malabar, Florida: Krieger, 1985.

Boltzmann, L., ***Lectures on Gas Theory*** (English translation from original German manuscript (1872)), Berkely: University of California Press, 1964.

Brandt, A., "Multi-level Adaptive Solutions to Singular Perturbation Problems, in P.W. Hernker (ed.), ***Numerical Analysis of Singular Perturbation Problems***, Academic Press, New York, 1979, pp. 53-142.

Briesmeister, J., ed., "MCNP-A General Monte Carlo Code for Neutron and Photon Transport, Version 4b," ***Los Alamos National Laboratory***, 1998.

Chandy, K., and J. Misra, ***Parallel Program Design--A Foundation***, Reading, MA: Addison-Wesley, 1988.

Chilton, A., J. Shultis, and R. Faw, ***Principles of Radiation Shielding***, Englewood Cliffs, New Jersey: Prentice Hall, 1984.

Dorr, M., and E. Salo, "Performance of a Neutron Transport Code with Full Phase Space Decomposition on the Cray Research T3D," ***Proc American Nuclear Society Int'l Conference on Mathematics and Computations, Reactor Physics, and Environmental Analyses***, Portland, Oregon (1995).

Flynn, M., "Some Computer Organizations and their Effectiveness," ***IEEE Trans Comput***, C-21:948-960 (1972).

Freeman, T. and C. Phillips, ***Parallel Numerical Algorithms***, New York: Prentice Hall, 1992.

Ganapol, B., **The Analytical Benchmark Library for Nuclear Engineering**, OECD/NEA Press, Paris, France 2007 (In Press)

Gerner, J., "Scalability Issues," ***Proc Conference on Parallel Programming on the IBM SP-2***, Cornell Theory Center, Cornell University, Ithaca, N.Y. (1995).

Ghita, M., G. Sjoden and G. Ghita, "Parallel *PENTRAN* Solutions of the "TIEL" Steady State Neutron Transport Benchmark Problems", ***Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications*** (M&C + SNA 2007), Monterey, CA, April 15-19, 2007.

Ghita G., G. Sjoden, and J. Baciak, "A Methodology for Experimental and 3-D Computational Radiation Transport Assessments of Pu-Be Neutron Sources," **Nuclear Technology**, Vol 159, pp. 319-328, September 2007.

Gropp, W., E. Lusk, and A. Skjellum, ***Using MPI: Portable Parallel Programming with the Message Passing Interface***, Cambridge, Massachusetts: MIT Press, 1994.

Haghighat, A., "New Spatial Parallel Sn Algorithms for One Dimensional Spherical Geometry," ***Transactions of the American Nuclear Society***, 65: 206-207 (1992).

Haghighat, A., M. Hunter, and R. Mattis, "Iterative Schemes for Parallel Sn Algorithms," ***Proc 1994 American Nuclear Society Reactor Physics Conference***, I: 423, Knoxville, Tennessee (1994).

Haghighat, A., M. Hunter, and R. Mattis, "Iterative Schemes for Parallel Sn Algorithms in a Shared Memory Computing Environment," ***Nuclear Science and Engineering***, 121: 103-113, (1995).

Proc 1st International Workshop/Training Course on Transport Methodologies and Uncertainty Estimation for PWR Vessel Fluence and BWR Shield/Shroud Dose Calculations, A. Haghighat, ed., Penn State University, (1995).

Haghighat, A., G. Sjoden, and M. Hunter, "Parallel Algorithms for the Linear Boltzmann Equation Based on Complete Phase Space Decomposition," ***(Invited) SIAM Annual Meeting***, Kansas City, MO,(1996).

Haghighat, A., H. Ait Abderrahim, and G. Sjoden, "Accuracy and Parallel Performance of *PENTRAN* Using the VENUS-3 Benchmark Experiment," ***ASTM STP-1398: Reactor***

Dosimetry, John G. Williams, David W. Vehar, et. al, Eds., American Society For Testing and Materials, West Conshohocken, PA, 2000.

Haghighat, A., G. E. Sjoden, V. N. Kucukboyaci, "Parallel Performance and Robustness of *PENTRAN*™ for Simulation of Real-Life Shielding Problems," ***Radiation Protection 2000 Proceedings***, p. 275-284, Spokane, Washington, September 2000.

Haghighat, A., G. E. Sjoden, and V. Kucukboyaci, "Effeciveness of *PENTRAN*'s Unique Numerics for Simulation of the Kobayashi Benchmarks," ***Special Issue, Progress in Nuclear Energy Journal,*** "3D Rad. Transport Benchmarks for Simple Geometries with Void Region," 39, Nr. 2 2001, ISSN 0149-1970.

Hill, T. R., "ONETRAN: A discrete Ordinates Finite Element Code for the Solution of the One-Dimensional Multigroup Transport Equation," ***LA-5990-MS***, Los Alamos Scientific Laboratory (1975).

Hiromoto, R., and B. Wienke, "A Chaotic, Single-Pass Inner-Iteration Scheme for the Discrete Ordinate Method Sn,," ***Proc 1989 American Nuclear Society Mathematics and Computation Conference***, 70: 1-17, Santa Fe, New Mexico (1989).

Hoermann, H., "Application of Parallel Computing Systems to Nuclear Engineering Problems," ***Proc Int'l Topical Meeting on Advances in Mathematical Methods for the Solution of Nuclear Engineering Problems***, II:519-527, Munich, Federal Republic of Germany (1981).

Hunter, M., and A. Haghighat, "Combined Spatial/Angular Domain Decomposition Sn Algorithms for Shared Memory Parallel Machines," ***Proc Joint Int'l Conference on Mathematical Methods and Supercomputing in Nuclear Applications***, 2:112-123, Karlsruhe, Germany (1993).

James, M., G. Smith, and J. Wolford, ***Applied Numerical Methods for Digital Computation with FORTRAN and CSMP (2nd Edition)***, New York, Harper and Row, 1977.

Kallinderis, Y., "Numerical Treatment of Grid Interfaces for Viscous Flows," ***J. Comp Phys*** , 98: 129-144 (1992).

Kobayashi, K., "A Proposal for 3-D Radiation Transport Benchmarks for Simple Geometries with Void Region," ***Dept of Nuclear Engineering, Kyoto University, Yoshida, Sakyoku, Kyoto, Japan, presented as a Technical Memorandum submitted to the OECD/NEA***, winter 1997.

Kobayashi, K., et. al., "3-D radiation Transport Benchmarks for Simple Geometries with Void Region," ***OECD/NEA Nuclear Science Publication,*** Paris, France (2000).

Kucukboyaci, V., A. Haghighat, G. Sjoden, and B. Petrovic, "Modeling of the BWR for Neutron and Gamma Fields Using *PENTRAN*," ***ASTM STP-1398: Reactor Dosimetry***, John

G. Williams, David W. Vehar, et. al, Eds., American Society For Testing and Materials, West Conshohocken, PA, 2000.

Lathrop, K., "Spatial Differencing of the Transport Equation: Positivity vs. Accuracy," ***J. Comp Phys***, 4: 475-498 (1969).

Lee, C., "The Discrete Sn Approximations to Transport Theory," ***LA-2595***, Los Alamos Scientific Laboratory, (1962).

Lewis, E. E. and W. F. Miller. ***Computational Methods of Neutron Transport***, LaGrange Park, Illinois: American Nuclear Society, 1993.

Longoni, G. and A. Haghighat, "Development of new quadrature sets with the Ordinate Splitting technique", ***Proceedings of the ANS International Meeting on Mathematical Methods for Nuclear Applications,*** Salt Lake City UT, 2001.

Longoni, G., June 2004, Private Communication.

Longoni, G., G. Sjoden, A. Haghighat, "Development and Applications of the SPL Methodology for a Criticality Eigenvalue Benchmark Problem," ***Nuclear Mathematical and Computational Sciences: A Century in Review, A Century Anew***, Gatlinburg, TN, Am Nuc Soc, LaGrange Park, IL, April 2003.

Longoni, G., A. Haghighat, and G. Sjoden, "Development and Application of the Multigroup Simplified P_3 (SP_3) Equations in a Distributed Memory Environment," ***PHYSOR 2002 Proceedings—International Conference on New Frontiers of Nuclear Technology***, Seoul, Korea, October 2002.

Mathews, K., G. Sjoden, and B. Minor "Exponential Characteristic Spatial Quadrature for Discrete Ordinates Radiation Transport in Slab Geometry," ***Nuclear Science and Engineering***, 118: 24-37 (1994).

Mattis, R., and A. Haghighat, "Parallel Sn Algorithms on Workstation Networks," ***Proc 8th Int'l Conference on Radiation Shielding***, I: 558-563 (1994).

Miller, W., "Generalized Rebalance: A Common Framework for Transport Acceleration Methods," ***Nuclear Science and Engineering***, 65:226-236 (1978).

Nakamura, S., ***Computational Methods in Engineering and Science***, New York, Wiley and Sons, 1977.

Nowak, P., E. Larsen, and W. Martin, "Multigrid Methods for Sn Problems," ***Transactions of the American Nuclear Society***, 55: 355-356 (1987).

O'Dell, R., F. Brinkley, D. Marr, and R. Alcouffe, ***"DANTSYS - One, Two, and Three Dimensional Multigroup Discrete Ordinates Transport Code System,"*** Los Alamos National Laboratory, (1995).

OECD/NEA Nuclear Science Series, “Prediction of Neutron Embrittlement in the Reactor Pressure Vessel, Venus-1 and Venus-3 Benchmarks,” ***OECD/NEA Nuclear Science Publication***, Paris, France (2000).

Petrovic, B., and A. Haghighat, "Boundary Conditions Induced Oscillations in the Sn Solutions of the Neutron Transport Equation," ***Proc Int'l Conf on Mathematics and Computations, Reactor Physics, and Environmental Analyses***, Portland, OR, April 30-May 4 1995, Vol I: 382-391, La Grange Park, IL, American Nuclear Society (1995a).

Petrovic, B., and A. Haghighat, "Analysis of Inherent Oscillations in Multidimensional SN Solutions of the Neutron Transport Equation," ***Nuclear Science and Engineering***, 124: 31-62, (1996).

Petrovic, B., and A. Haghighat, "New Directional Theta-Weighted Sn Differencing Scheme," ***Transactions of the American Nuclear Society***, 73: 195-197 (1995b).

Petrovic, B., and A. Haghighat, "New Directional Theta-Weighted Sn Differencing Scheme and its Application to Pressure Vessel Fluence Calculations," ***Proc 1996 Radiation Protection and Shielding Topical Meeting***, Falmouth, MA, Vol I, 3-10 (1996).

Petrovic, B., and A. Haghighat, "New Directional Theta-Weighted (DTW) Differencing Scheme and Reduction of Estimated Pressure Vessel Fluence Uncertainty," ***Proc 9th International Symposium on Reactor Dosimetry***, 746-753, Prague, Czech Republic, September 1996.

Petrovic, B., A. Haghighat, M. Mahgerefteh, and J. Louma, "Validation of Sn Transport Calculations for Pressure Vessel Fluence Determination at Penn State," ***Proc 8th Int'l Conference on Radiation Shielding***, I: 558-563 (1994).

Phillips, R., and F. Schmidt, “Multigrid Techniques for the Numerical Solution of the Diffusion Equation,” ***Numerical Heat Transfer***, 7, 251-268 (1984).

Plower, Thomas, Private Communication (2007).

Reed, "The Effectiveness of Acceleration Techniques for Iterative Methods in Transport Theory," ***Nuclear Science and Engineering***, 45:245-254 (1971).

Rhoades, W., “Improvements in Discrete-Ordinates Acceleration," ***Transactions of the American Nuclear Society***, 39: 753-755 (1981).

Rhoades, W., and R. Childs, ***“TORT/DORT: Two- and Three-Dimensional Discrete Ordinates Transport," CCC-543,*** ORNL Radiation Shielding Information Center, Oak Ridge, TN (1991).

Rhoades, W., and W. Engle, "A New Weighted Difference Formulation for Discrete Ordinates Calculations," ***Transactions of the American Nuclear Society***, 27: 776-777 (1977).

Sewell, G. ***The Numerical Solution of Ordinary and Partial Differential Equations***, New York, Academic Press, 1988.

Shedlock, D. and A. Haghighat, "Neutron Analysis of Spent Fuel Storage Installation Using Parallel Computing and Advanced Discrete Ordinates and Monte Carlo Techniques," ***Proceedings of the ICRS-10 – RPS 2004 Conference,*** Madeira Island, Portugal, May 2004.

Sjoden, G., "Exponential Characteristic Spatial quadrature for Discrete Ordinates Neutral Particle Transport in Slab Geometry," ***M.S. Thesis, AFIT/GNE/ENP/92-M10***, Air Force Institute of Technology (1992).

Sjoden, G. and A. Haghighat, "The Exponential Directional Weighted (EDW) Differencing Scheme in 3-D Cartesian Geometry," ***Proceedings of the Joint International Conference on Mathematics and Supercomputing for Nuclear Applications, Saratoga Springs***, New York, Vol II: 1267-1276, October 1997.

Sjoden, G., and A. Haghighat, "*PENTRAN*, A 3-D Scalable Transport Code with Complete Phase Space Decomposition," ***Transactions of the American Nuclear Society***, 74, 181-183 (1996).

Sjoden, G. E., "*PENTRAN*: A Parallel 3-D SN Transport Code with Complete Phase Space Decomposition, Adaptive Differencing, and Iterative Solution Methods," ***Ph.D. Thesis in Nuclear Engineering***, The Pennsylvania State University, May 1997.

Sjoden, G., and A. Haghighat, "Taylor Projection Mesh Coupling Between 3-D Discontinuous Grids for Sn," ***Transactions of the American Nuclear Society,*** 74, 178-179 (1996).

Sjoden, G., and A. Haghighat, "A New Adaptive Differencing Strategy in the *PENTRAN* 3-D Parallel Sn Code," ***Transactions of the American Nuclear Society,*** 75, (1996).

Sjoden, G., and A. Haghighat, "A Simplified Multigrid Acceleration in the *PENTRAN* 3-D Parallel Code ," ***Transactions of the American Nuclear Society,*** 75, (1996).

Sjoden, G. and A. Haghighat, "The Exponential Directional Weighted (EDW) Differencing Scheme in 3-D Cartesian Geometry," ***Proceedings of the Joint International Conference on Mathematics and Supercomputing for Nuclear Applications***, Saratoga Springs, New York, Vol II, pp.1267-1276, Oct 1997.

Sjoden, G. and A. Haghighat, "Advanced 3-D Parallel Discrete Ordinates Methods for Criticality Safety Calculations," ***Proceedings of the 1999 Mathematics and Computation Conference on Reactor Physics and Supercomputing for Nuclear Applications***, Madrid, Spain, September 1999.

Sjoden, G. and A. Haghighat, “New EDW Sn Differencing Scheme Compared to Monte Carlo for Zero-C Problems--Accuracy and Cost,” ***Transactions of the American Nuclear Society***, Boston, Massachusetts, June 5-11 1999.

Sjoden, G. and R. N. Gilchrist, D.L. Hall, and C. A. Nusser, “Modeling a Radiographic X-ray Imaging Facility with the *PENTRAN* Parallel Sn Code,” ***PHYSOR 2000 Proceedings***, Pittsburgh, PA, May 2000.

Sjoden, G. E., “Deterministic Adjoint Transport Applications for He-3 Neutron Detector Design,” **Annals of Nuclear Energy**, 29, 1055-1071, 2002.

Sjoden, G. E., “Applications of *PENTRAN* to a High Dose Rate Brachytherapy Problem,” **Transactions of the American Nuclear Society**, Vol 88, San Diego Meeting, American Nuclear Society, June 2003.

Sjoden, G.E., “Performance of *PENTRAN* Using a Heterogeneous Cluster For Selected Medical Physics Problems,” **Transactions of the American Nuclear Society**, Vol 90, American Nuclear Society, June 2004.

Sjoden, G., “An Efficient Exponential Directional Iterative Differencing Scheme for 3-D Sn Computations in XYZ Geometry,” **Nuclear Science and Engineering**, Vol 155, pp. 179-189, 2007.

Stamm'ler, R. and M. Abbate, ***Methods of Steady State Reactor Physics in Nuclear Design***, New York, Academic Press, 1983.

Wagner, J. and A. Haghighat, “Application of the Discrete Ordinates Adjoint Function to Accelerating Monte Carlo Reactor Cavity Dosimetry Calculations," ***Proc 1996 Radiation Protection and Shielding Topical Meeting***, Falmouth, MA, Vol I, 345-352 (1996).

Walters, W., T. Wareing, and D. Marr, "The Non-Linear Characteristic Scheme for X-Y Geometry Transport Problems," ***Proc Int'l Conf on Mathematics and Computations, Reactor Physics, and Environmental Analyses***, Vol I, American Nuclear Society, Portland, OR, April 30-May 4 1995.

Wareing, T. and R. Alcouffe, "An Exponential Discontinuous Scheme for x-y-z Geometry Transport Problems," ***Proc 1996 Radiation Protection and Shielding Topical Meetin***g, Vol II: 597-604, American Nuclear Society, Falmouth, MA, 1996.

Werner, W., "Application of Parallel Computing Systems to Nuclear Engineering Problems," ***Proc Int'l Topical Meeting on Advances in Mathematical Methods for the Solution of Nuclear Engineering Problems***, II:509-519, Munich, Federal Republic of Germany (1981).

Wienke, B., R. Hiromoto, and R. Brickner, "Parallel Discrete Ordinates Algorithms on Distributed and Common Memory Systems," ***Transactions of the American Nuclear Society***, 55: 321-322 (1987).

Wienke, B., and R. Hiromoto, "Parallel Sn Iteration Schemes," ***Nuclear Science and Engineering***, 90: 116-123 (1985).

Yavuz, M. and C. Aykanat, "Spatial Domain Decomposition Applied to Linear Discontinuous Sn Methods," ***Transactions of the American Nuclear Society***, 66: 274 (1992).

Yavuz, M. and E. Larsen, "Iterative Methods for Solving x-y Geometry Sn Problems on Parallel Architecture Computers," ***Nuclear Science and Engineering***, 112: 32-42 (1992).

Yavuz, M. and E. Larsen, "Spatial Domain Decomposition Methods for Discrete Ordinates Problems," ***Proc. 1989 American Nuclear Society Mathematics and Computation Conference***, 69, Santa Fe, New Mexico (1989).

Yi and Haghighat, "PENMSH-XP: A 3-D Mesh Generator for *PENTRAN*, Version 1.5b" University of Florida, (2008).

www.hswtech.com