LA-UR-04-3400

Approved for public release;
distribution is unlimited.

Title: MCNP5 Tally Enhancements for Lattices
(aka Lattice Speed Tally Patch)

Author(s): Tim Goorley
X-5
Los Alamos National Laboratory

Submitted to:

Los Alamos
NATIONAL LABORATORY
Research Notes
APPLIED PHYSICS DIVISION
Diagnostics Applications Group, X–5
Phone:(505)667-8747—FAX:(505)665-3046

To/MS: Distribution
From/MS: T. Goorley
Phone/Email: 5-8417/jgoorley@lanl.gov
Symbol: X-5-RN(U)04-20
Date: 14 May 2004

MCNP5 Tally Enhancements for Lattices
(aka Lattice Speed Tally Patch)

Abstract

Tally and tracking modifications for MCNP4B were created by XTM in 1996 for the Harvard/MIT Boron Neutron Capture Therapy clinical trials team to greatly reduce (by factors of 100 or more) the wall-clock runtimes of their lattice calculations. The lattice speed tally enhancements (LSTE) have been revised and added to the MCNP5_LANL_1.20 and MCNP5_PROTON_1.16 threads. The revisions allow the code to recognize when most of the stringent requirements are met and, if so, will automatically use the tally enhancements. These requirements are: 1) a hexagonal lattice is present in the geometry, 2) none of the following are used: dxtran spheres, gaussian energy broadening, energy, angle or time bins, energy, angle or time bin multipliers, cell or surface flagging, several other tally types, time convolution, weight windows generator, and perturbations, and 3) all **F4** tally cells contain a lattice in their path. MCNP will not check three criteria, however: 1) if nested lattices are tallied over, 2) if a partial lattice index range is present on the **F4** card, and 3) if the entries for a cell's **fill** keyword include its own universe. If any of the three preceding conditions are met, then MCNP will likely crash but may yield wrong tally results, which are most likely all zeros.

Additionally, the new data card **spdtl**, with the keyword **force** or **off**, has been added to allow the user to force or prevent, respectively, the use of the modified tally routine. This allows the user to run a short test case with and without the enhancements and verify they are appropriate if the two runs yield the same tally results. Using **spdtl force** will also print comments about LSTE conflicts with other cards. To test the new executable, five regression suite test problems were modified to be able to run with the LSTE, and their tally results agree as expected. The new executable also passes the regression test suite.

Table of Contents

Introduction

This research note describes the incorporation of the lattice speed tally enhancements (LSTE) in MCNP5, which can greatly reduce wall clock runtimes for certain hexagonal lattice geometries. The code will recognize if many of the strict criteria that the LSTE were originally intended for have been met and will automatically use the modifications if so. This new MCNP5 executable also allows the user to force or prevent the use of the tally enhancements with the data card **spdtl** and the keyword **force** or **off**, respectively. Warning messages or comments are printed to the screen and output file concerning the LSTE enhancement usage.

The LSTE and new executable passed three kinds of tests to ensure the LSTE didn't introduce defects and operates as intended.

This document discusses how and why these modifications were developed, their proper usage, their implementation in MCNP5 and how they were tested. The "Background" section discusses why these modifications were developed, by whom, and how they were implemented. The "Lattice Speed Tally Usage" section contains six sub sections. The first subsection discusses the usage of the **spdtl** (SPeeDTaLly) card in MCNP5, which can control whether the enhancement is used or not. The next two subsections discuss the specific geometry and tally problem setup that the enhancement was designed for. The next two subsections deal with variations from this setup. The final subsections discusses other cards that cannot be used with the LSTE. The section "Changes to MCNP5 Source" details the variable introduced in MCNP5 and how they control whether the enhancement is used during the execution of the code. The next section, "Changes to MCNP5 Manual", mentions the changes to the MCNP5 Manual which should be implemented. The next section, "Testing", discusses how the new executable was verified. The "Results" section contains some comparisons of the speedup achieved from the LSTE. The "Future Improvements" section mentions some additional improvements, which could be implemented in the future if the need arises. Appendix A and B contain two conference papers on the enhancement. The original LSTE for MCNP4B are in Appendix C. Appendix D and E contain an example of the intended input and modified regression test suite problems, respectively.

Background

The lattice speed tracking and tally modifications were originally written by Dr. Gregg McKinney and Dr. Ken Adams in 1996 while they were in X-TM (now X-5), for the Harvard/MIT Boron Neutron Capture Therapy clinical trials program. The speedup obtained in their MCNP4B lattice calculations made treatment planning tractable and logistically feasible in a schedule which contained patient assessment, CT and MRI imaging, treatment planning, and neutron irradiation in a single week. The runtimes of their MCNP lattice problem, a 21 x 21 x 25 (each voxel is 1 cm^3) model, based on patient CT images, were effectively reduced by a factor of ~200 when the modifications were implemented. These runtime reductions, as well as the patch itself, have been presented in two conference papers, which are included in Appendix A and B. The second paper also discusses the testing and verification of the modifications, which were important since the modified version did not track the unmodified version.

As more countries initiated their own BNCT clinical trials, this patch (and the software to generate MCNP input decks from CT images) was distributed abroad without charge. I estimate roughly twelve research teams use this patched version of MCNP4B, most for patient treatment planning of neutron irradiations. Anyone with a large hexagonal lattice geometry and a few specific tally cards, however, would benefit from these modifications. Aside from clinical trials teams, several researchers are doing calculations with large lattices, ie. millions or tens of millions of voxels, based on human CT images, which also benefit from these modifications.

Many of the users of the existing patched version of MCNP4B complain that it is very unwieldy. Since modifications were applied at compile time, the executable would not run the regression test suite, resulting in several access violations and program crashes. This created problems for verification and validation efforts. Additionally, if the user wanted to use the patched version of the code, they had to continue to use MCNP4B. Since these calculations often use a surface source from a reactor criticality calculation, this **kcode** calculation had to be run with MCNP4B, instead of the most recent version of MCNP. Integrating the modifications into the LANL source would alleviate these problems and allow more users to access this functionality.

Lattice Speed Tally Enhancements (LSTE) Usage

While the LSTE dramatically reduces wall-clock runtimes, there are very few circumstances which can successfully utilize the enhancement. The following paragraphs address the LSTE usage and problem setup. The first section below discusses the **spdtl** card. The following two sections discuss the input geometry and tally cards that the enhancement was designed for. The following sections address altering the geometry and tally cards from that intended in the development of the LSTE. The final section lists which tally cards are prohibited with the lattice speed tally enhancements (either automatically or with the **spdtl force** card).

SPDTL Card

The use of the LSTE can be controlled with the **spdtl** card. The card is used in the data card section of the MCNP input deck and must have exactly one of two possible keywords. The keyword **off** will preserve normal behavior (i.e. slow), even if all the criteria are meet. The keyword **force** will force the use of the lattice speed tally enhancements, unless there is

not a hexagonal lattice present, in which case this card will be ignored. The **force** keyword will also cause comment messages to be printed for any conflict between the LSTE and other cards. Forcing the use of the lattice speed tally modifications may result in a crash, tallies which are all zero's or even silently wrong answers. Using **spdtl force** is discouraged, except to identify conflicts between the LSTE and other cards. If the **spdtl** card is not present, mcnp will check most of the possible conflicting cards or keywords are will use the lattice speed tally modifications if appropriate. MCNP will not check three criteria, however: 1) if nested lattices are used in a **F4** tally, 2) if a partial lattice index range is present on the **F4** card, and 3) if the entries for a cell's **fill** keyword include its own universe.

Intended Problem Geometry

The LSTE was written for use with a specific geometry. This geometry contained only a large lattice contained in a void sphere. An example of this geometry is show in Figure 1.

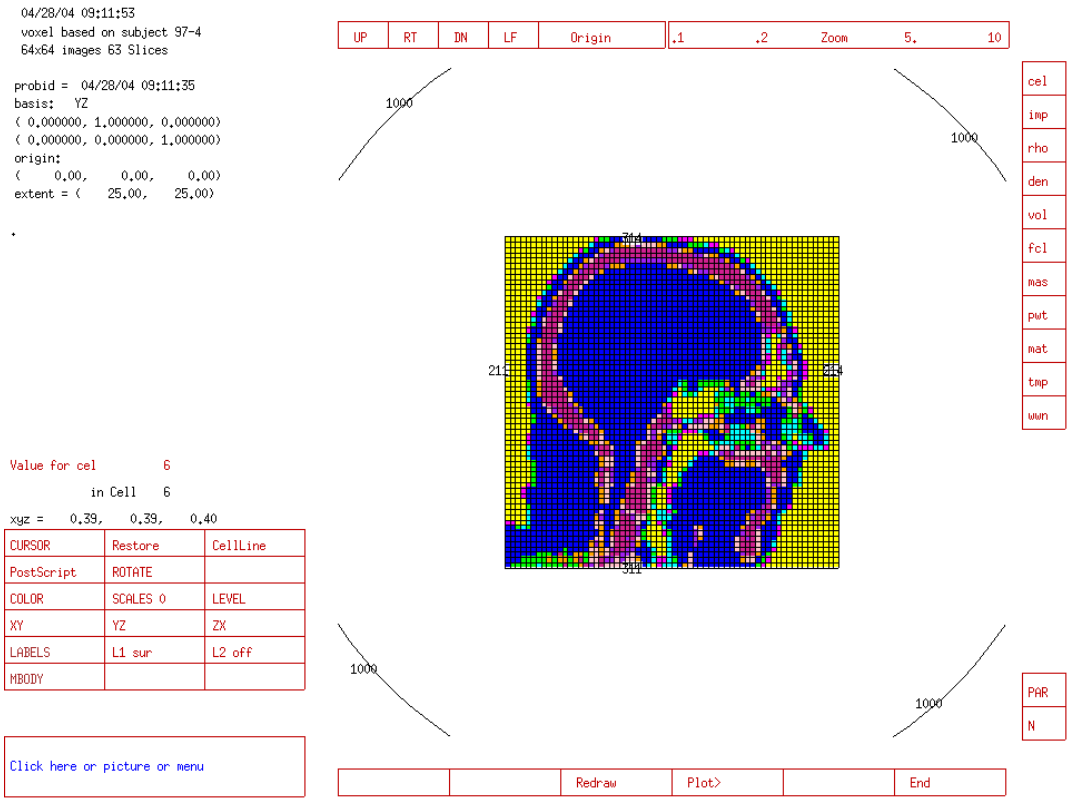


Figure 1. MCNP5 geometry plot of intended problem geometry: a hexagonal lattice contained in a sphere defining the problem space. This lattice is a 64x64x63 model of a human head.

Several restrictions apply to the lattice specifications. The universe which fills each lattice voxel was individually specified, i.e. **lat=1 fill= -10:9 –15:14 –5:4 2 5999r** for a 20x30x10 lattice of universe 2 cells. Simply saying **lat=1 fill=2** will construct the same geometry in MCNP5, but will not work properly with the LSTE and will result in silent wrong answers. For the intended problem geometry, the variation of the filling universes is what creates the voxel phantom. Additionally, none of these specified fill universes belong to the filling cell. In this example for cell 10, **10 100 –1.5 …. u=3 lat=1 fill=-10:9 –15:14 –5:4 2 3 4 1 5995r**, the **3** between the **2** and **4** references a universe, 3, which is the same universe that the fill keyword belongs to (**u=3**). This is allowed in MCNP and causes the corresponding lattice voxel to filled with that cell's material, material 100 in this case. If cell 10 didn't have a material and density, the corresponding lattice voxel would be a void. This was not intended to be handled by the LSTE, however. Finally, in the original problem geometry, all the lattice cells that are specified are available for transport, i.e. there is not another bounding surface which excludes some of the lattice. Since neutron doses are calculated in all lattice voxels, no variance reduction is used other than implicit capture. The importance for all cells is 1, except the outside cell, which is importance 0. For a complete example input deck for which the LSTE is fully compatible, see Appendix D.

Intended Tally Cards

The lattice speed tally enhancement was written for use with the **f4**, **sd4**, **fm4**, **de4** and **df4** cards, but, like the lattice geometry, there a few restrictions are placed on these cards. The **f4** card must reference a hexagonal lattice (i.e. brackets must be used), must reference the path to the lattice, and must reference exactly the entire lattice in which particles are transported. For example, the **f24:p (100<100[-8:7 -8:7 -8:7])** tally card is valid with the LSTE when the corresponding geometry has a cell with a fill index range entry of **fill= -8:7 –8:7 –8:7**. As per the usual tally format, the lattice may not be the first thing specified, so the **100[-8:7 –8:7 –8:7]** must be preceded by the **100<**, and the whole path must be in parenthesis. Even though the lattice is neither the highest nor the lowest level universe, this path is acceptable for use with or without the LSTE. MCNP5 will check to see if a lattice is referenced on the **f4** tally card and will not use the LSTE if no lattice is specified. MCNP5 will not check to see if the lattice index range or path is correct.

The **sd4** card, used to enter the volume of a lattice voxel, should be present. The fatal error "tally volumes or areas were not input nor calculated" will be issued by MCNP if the **sd4**

card is not present, and if the volume for each lattice voxel is not calculated. The original lattice speed tally was written for a voxel volume which happened to be 1.0.

A **fm4** card must be present for each tally, and must be a single value multiplier, to use the LSTE. If the tally values do not need to be multiplied or changed, **fm4 1.0** should be used. Typically, for the problem the enhancement was designed for, the single entry on the FM card is used to multiply the tally result by the neutron beam source strength, causing the tally (usually flux per source particle) to yield the neutrons/cm^2s. MCNP5 will check to see if a **fm** card is present for each tally, and if not, will not use the LSTE. If the **fm4** card is not present, that tally's results will be 0.0 if the **spdtl force** card is used. MCNP5 will check to see if multiple entries are present on the **fm** card for each tally, and if so, will not use the LSTE. If multiple entries are used on the **fm4** card, to calculate reaction rates for example, and the **spdtl force** card is used, the results will be silently wrong or may cause a crash.

The **de4 df4** cards functional normally but must be present to use the LSTE. If the tally values do not need to be multipled by a set of dose conversion factors or response function, then two entries (over the entire energy range of the particle tallied over in the problem) should be give as 1.0. For example, **de4 1E-11 100** (newline) **df4 1 1** should be used for neutron tallies. Typically the **de4 df4** cards are used to convert neutron or photon fluxes in to kerma (~dose). If each tally does not have a **de4 df4** card, then MCNP will not use the LSTE. If the **spdtl force** card is used, that tally's values will be 0.0 if no **de df** cards are present for that tally.

Several other tallies are compatible with the LSTE. Since the **f5** tallies, the point and ring detectors, are not referenced in the skipped portion of the code, they are not impacted by the LSTE and operate as intended. The LSTE were designed developed for MCNP4B, and two new tally capabilities have been introduced since. The **fmesh** card and the radiography tallies (**fir, fic, fip**) do not use the tally routine which was modified and are thus compatible with the LSTE. Any legal tally modification card (**fm, de df, e, t, em, tm,** etc.) can be used with these tallies and will not interfere nor be affected by the LSTE, i.e. these cards produce the same tally output with or without the LSTE and the LSTE will not affect the speed of these tallies. Minor differences in the figure of merit (FOM) will be present if there is any tally which uses the LSTE. Using either of these other tally capabilities will slow down the code, however. The difference in run times will vary with the size and granularity of the mesh used with the tallies. Table 1 summaries the restrictions on the cards the LSTE are intended to be used with.

Table 1. Tally cards intended to be used with the lattice speed tally modifications.

Tally Related Data Cards Allowed	Comments
F4	Volume Averaged Tally -Every lattice location must be tallied over.
*DE DF cards (for F4)	Dose Response - Function as usual. Must be present
*SD (for F4)	Segment Divisor - Function as usual. Must be present.
*FM (for F4)	Tally Multiplier - Only simple multiplier allowed! Must be present.
F5, F5x, F5y, F5z	Point or Ring Detectors – Function as usual.
FMESH	Mesh Tally - Function as usual.
FIR, FIP, FIC	Radiography Tally - Function as usual.

* The DE DF SD and FM cards for F5, FMESH, FIR, FIP, and FIC cards are optional and fully functional.

Most other tally cards and F4 tally related cards are not intended for use with the LSTE. See the following section, "Cards Disallowed with the Lattice Speed Modifications" for more details.

Variations from the Intended Geometry

Major variations in this geometry are not expected to cause problems, since the main lattice speed tally modifications are only to one of the tally routines, tally.F90. Additional cells may be added to this geometry, including other lattices. Adding lattices within lattices is valid and transport will still be valid, but the tallies may not behave correctly. Using the **like ... but** card to copy an existing lattice is also valid for transport, but will cause difficulties if tallied over. Small deviations from this geometry have been tested, such as adding another lattice and material external to the lattice, and the output and tally files are as expected. See the discussion under "Testing" about variations in the geometries of the regression test suite which are appropriate for use with the LSTE. The are no checks in MCNP5 to determine the validity of the geometry when using the LSTE.

Variations in Intended Tally Cards

Unlike the geometry portion of the input deck, the tally cards are very sensitive to change when using the LSTE. Even small variations from the originally intended tally cards may result in tally values which are all zero's, an unexpected code crash, or even silent wrong answers. MCNP5 will check many of the criteria which the LSTE depend on and will enable or disable the modifications according to the cards or keywords present. Not all criteria are

checked, however, and the user is encouraged to run a short problem both with and without the LSTE and verify that the tally values exactly agree.

Some variations in the tally cards are have been tested and verified, however. The full path to a lattice cell can be given on its tally card, and not just the lattice itself. Individual cells which partially comprise a lattice voxel can also be tallied over correctly, and not just the entire lattice voxel. The entire lattice must still be tallied over, however. Extending the lattice tally range to voxels where particle transport will not occur results in a fatal error "[i]error in level x of f card y tally z." Using the **sd** card with entries other than 1.0 has been tested and works correctly. Tallying over nested lattices will partially work. Only the upper (outermost) lattice tally will be correct. See the "Testing" section for more details.

Tally Cards Disallowed with LSTE.

The modifications to tally.F90 effectively skip all of the coding (541 lines) and replace it with 22 lines of code. Therefore, most of the lines of code corresponding to angle, energy, or time bins or multipliers, perturbations, the weight windows multipliers, etc. have been skipped and, assumedly, will no longer work correctly. Coding for tally types other than the F4, fmesh, radiography tallies, and point and ring detectors is also skipped and they are prohibited when using the LSTE. Specific cards corresponding to the skipped code (and therefore not compatible with the LSTE) were identified by one of four ways: 1) skipped subroutine calls, 2) skipped global array assignment statements, 3) comments in skipped code, 4) running the code with the suspect test card. The list of known disallowed cards is given in Table 2. When these cards are used, the default for LSTE is switched to **off**, and the LSTE can only be enabled with the **spdtl force** card. Doing so, however, will probably result in a MCNP crash during execution.

Table 2. Cards explicitly disallowed with lattice speed tally.

Comment in code	Associated Cards	Associated index in cnm variable
! warn if tally made by both dxtran and non-dxtran particles.	DXT DXC	47 16
! gaussian energy broadening.	FT (GEB)	28
! find the energy bin and multiply by the energy bin multiplier.	E EM	30 37
! find the time bin.	T TM	31 38
! set jbd=2 if the tally is flagged.	CF SF	40 41
! set up tally index coefficients.	PERT	85
! branch according to tally type.	Fn	Types of 24
! find the segment bin. preserve lgc for track. (Firsts of Many comments about segments and bins)	FS SF	42 41
! multiply by the energy if there is a * on the f card.	*Fn	Option of 24
! do time convolution.	FM	34
! totals nest. (calls weight window generator)	WWG WWGE	71 72
! check about particle track plotting	VISED!	

Other than the weight windows generator, all the excluded cards are related to tallies. Cards related to plotting, physics, printing and cutoffs are not affected, and are allowed. While the **wwg** and **wwge** cards are not allowed, the weight windows card itself is still acceptable.

Words of Warning – Using the LSTE beyond the intended circumstances.

There are many checks in MCNP to make sure that the LSTE will be used appropriately. If any of the disallowed cards in Table 2 are present, the LSTE will not be used unless the **spdtl force** card is present. Using multiple entries on the **fm** card, or not having a **fm** or **de, df** cards for any tally will likewise prevent the LSTE from being used. If a hexagonal lattice is not present or every **f4** tally does not reference the lattice, the LSTE will not be used. MCNP will not check three criteria, however: 1) if nested lattices are tallied over, 2) if a partial lattice index range is present on the **F4** card, and 3) if the entries for a cell's **fill**

keyword include its own universe. It is up to the user to determine if these conditions are fulfilled.

As a robust test of the LSTE capability for a problem setup, users should run a short job without the **spdtl** card and another with **spdtl off**. The file without the **spdtl** card should use the LSTE, if the criteria have been met. In either case, a message to the screen and output file will indicate if MCNP used the LSTE or not. Enough particles should be run to give non-zero tally results in most of the lattice cells. If the output tally values agree for both runs, then it is acceptable to use the LSTE for a much longer history run. Some differences in runtimes and FOM values are expected. This test only needs to be performed for changes in problem geometry or tally cards.

Changes to MCNP5 Source

Eight subroutines were changed in MCNP5_LANL 1.20 thread. These changes relate to setting one of the five flags, or printing comment or warning messages, or the tally modifications themselves.

Five flags control the functionality of the lattice speed tally patch. The first, flag_speed_tally_ok, is set if the criteria have been meet to use the patch. It is initially set to 0, indicating no tests have been done yet. It is then set to 1 if a hexagonal lattice is present. This is set in nxtit1, which is in the first pass through the input deck. In newcrd and nextit, the second pass through the input deck, flag_speed_tally_ok is set to –1 if any of the cards or options preventing the use of the LSTE are present. The test to determine if the LSTE should be disallowed is dependant on flag_speed_tally_ok set to 1. This is done so that if flag_speed_tally_ok is still set to 0, its value is unchanged and lattice speed tally messages are not printed. The **lat** keyword will always be found in the first pass of the input deck and the disallowed cards are always checked in the second pass, so there is no conflict, ie. the flag is enabled before it can be disabled. Even if the data card **lat** is the last line in the input deck the check will still function properly. The flag is initialized in main.F90 to -1, after all the fixcom variables (including flag_speed_tally_used) are initialized to zero. The second flag is flag_speed_tally_force, and is used to indicate the keyword on the **spdtl** card. It is initialized to 0 in mcnp_global and set to 1 or –1 in newcd1 if **spdtl force** or **off**, respectively, is present.

The third flag, flag_speed_tally_used is set to 1 if the modifications are to be used, ie. if the requisite criteria are meet. It is also set to 1 if the **spdtl** card is present and the force option is used, but two warnings are printed if the criteria are not meet. The flag_speed_tally_used is

set to –1 if the speed tally is not used because the criteria are not meet. It is also set to –1 if the card **spdtl** is present and the **off** keyword is used. This logic is performed in rdprob.F90. The flag flag_speed_tally_used is added to the fixcom common so that it would be used appropriately in continue runs. It is the only card which needs to be added to fixcom, as the geometry cannot change in a continue run, nor can any prohibited card be added to the continue run.

The fourth and fifth flags were added to check if **fm** and **de df** cards are present for every f4 tally. The 2D integer arrays flag_speed_tally_fm and flag_speed_tally_de are declared in mcnp_global as allocatable. In the beginning of rdprob, both dimensions of both variables are allocated from 1: ntal, the number of tallies present in the problem, excluding **fmesh** tallies. Then they are set to zero. If a **f4** tally is encountered, the flag_speed_tally_fm(ital,2) and flag_speed_tally_de(ital,2) is set in newcrd to icn, a value corresponding to the user defined number of the **f4** tally (4, 14, 24, etc.). If a **fm de**, or **df** card is encountered, the flag_speed_tally_fm(ital,1) or corresponding de array is also set to icn. In rdprob, a loop over ntal is performed, and if flag_speed_tally_fm(I,1) or de is zero, then that means that the **f4** card does not have a corresponding **fm** or **de df** card. Then the LSTE are disabled, unless the **spdtl force** card is present, and in that case a comment message is printed stating which tally does not have a **fm** or **de df** card. These tests in rdprob and checks in newcrd are designed so that the legal use or absence of the fm de or df cards for a f5, fmesh, radiography tally or point or ring detector do not prevent the use of the LSTE. These changes are summarized in Table 3.

Table 3. Changes to MCNP5 Source (as of the MCNP5_LANL_1-23 thread in cvs)

Routine	Line Number	Effect
Fixcom.F90	104 281 311	Declare flag_speed_tally_used as integer (1 or –1) Add flag_speed_tally_used to fixcom common. Equivalence Flag_speed_tally_used to jfixcm
Main.F90	137	Initialize flag_speed_tally_used to -1
Mcnp_global.F90	167-183	Declare flag_speed_tally_ok = 0 Declare flag_speed_tally_force = 0 Declare flag_speed_tally_fm 2D integer arrays & allocatable Declare flag_speed_tally_de 2D integer arrays & allocatable
Newcd1.F90	224-232	Handle SPDTL card and keyword.
Newcrd.F90	262-271, 273-290, 291-298, 418, 450 455-457 538-546 615-617	Check if prohibited tally modifying cards are present. Check if prohibited tally cards are present. Set fm, de check Check if [appears on f4 tally line. Set check for fm, de df presence. Check for prohibited dxtran Check for prohibited wwg and wwge Check for prohibited pert card
Nextit.F90	515-517 1558-1555	Change flag_speed_tally_ok to –1 if more than 1 entry on FM Handle SPDTL card and warning message.
Nxtit1.F90	479	Change flag_speed_tally_ok to 1 if hexagonal lattice present.
Rdprob.F90	24-28 225-240 241-260	Allocate and zero flag_speed_tally_fm and de. Do check for fm, de df cards over all (f4) tallies. Issue comment/warning on use & appropriateness of lattice tally. See Table 4 for warning text. Set flag_speed_tally_used according to values of flag_speed_tally_ok and flag_speed_tally_force.
Tally.F90	22 542-563	If test on flag_speed_tally_used to skip most of tally routine Lattice speed tally patch

The lattice speed tally modifications which replace the majority of tally.F90 in MCNP5_LANL and MCNP5_PROTONS are shown in the text below.

```
!   this is the special speed tally treatment for lattices
800 lx=lo+2               ! From: do all tallies that inc this cell, surface
    ta=wgt                          !   include weight of particle in tally
    j=-mfl(1,nint(udt(7,lev-1))) ! effectively from do surface or cell bins
    n8=nint(udt(8,lev-1))
    n9=nint(udt(9,lev-1))
    n10=nint(udt(10,lev-1))
    do ml=1,itds(lo)    !From: do all tallies that inc this cell, surface
      ital=itds(lx-1)
      n1=itds(lx)
      ir=n8-laf(j+1,1)+laf(j+1,2)*(n9-&
        & laf(j+2,1)+laf(j+2,2)*(n10-laf(j+3,1)))+1
      j7=ktal+jptal(5,ital)+iptal(2,5,ital)*(itds(lx+ir)-1)+1
      tb=dosef(ta)                  ! Include DE, DF cards into tally result
      td=tb*dr*tds(iptal(5,2,ital)+1)
      if(tal(j7) .eq. 0._dknd)then
       jtls=jtls+1
       if(jtls .le. ktls) tal(ktal+nmxf*mxf+jtls)=j7
      endif
      tal(j7)=tal(j7)+td
    lx=lx+n1+2
   enddo
   return
```

During the execution of MCNP5, the warnings given in Table 4 will be printed to the screen and the output file, depending on the input deck. Additionally, every time a condition in the input deck which would prevent the use of the LSTE occurs, a comment message is printed to the screen and the output file if **spdtl force** is in the input deck.

Table 4. Comments & warnings issued in conjunction with lattice speed tally enhancements.

Routine	Message	Condition
Newcrd1	Fatal. spdtl card must have exactly one keyword, either force or off.	Spdtl card present in input deck, but no / wrong keyword present.
Rdprob	Comment. using lattice speed tally modifications.	flag_speed_tally_ok == 1 flag_speed_tally_force == 0 or 1 (i.e. OK to use mods, either forced or no spdtl card.)
Rdprob	Comment. lattice speed tally modifications ok to use, but have been turned off.	flag_speed_tally_ok == 1 flag_speed_tally_force == -1 (i.e. OK to use mods, but 'spdtl off' present)
Rdprob	Comment. lattice speed tally modifications will not be used.	flag_speed_tally_ok == -1 flag_speed_tally_force == -1 or 0 (i.e. Not OK to use mods, no spdtl or 'spdtl off')
Rdprob	Warning. using lattice speed tally even though not appropriate. Warning. Silent wrong answers or crash may result.	flag_speed_tally_ok == -1 flag_speed_tally_force == 1 (i.e.) Not OK to use mods, 'spdtl force' present.)
Nextit	Warning. spdtl card present, but no lattice. spdtl ignored.	Spdtl card present. No lattice card present or lat /= 1 Flag_speed_tally == 0 Flag_speed_tally_force = -1 or 1
	---- No message ----	flag_speed_tally_ok == 0 flag_speed_tally_force == 0 (i.e.) No lattice & No spdlt card.

If the LSTE is used, some warning messages will not be printed, since they are skipped in the tally routine. These warning messages relate to user and time bins, which should not be present if the lattice speed tally enhancements are being used.

Table 5. Comments & warnings not issued if lattice speed tally enhancements used.

Routine	Message
Tally	Warning. Tally not scored beyond last user bin
Tally	Warning. Tally not scored beyond last time bin
Tally	Warning. Tallyx is in an endless loop.

Changes to MCNP5 Manual

The MCNP5 Manual should be changed to reflect the modifications to the MCNP5 source. The following changes should be made to the manual:

1) Page 3-77, Add to the end of the table: SPDTL Lattice Speed Tally page 3-114

2) Page 3-86, after n, pl.Si, Ci, #, Ii listing. Add new line:" Use: Consider using the SPDTL card."

3) Page 3-114 Immediately prior to the line F. Material Specification, add the SPDTL card description.

SPDTL Lattice Speed Tally Enhancement

Form: SPDTL x

x = off or force (one entry is required)

Default: The lattice speed tally enhancement will be enabled by default if MCNP strict criteria are meet.

Use: Optional.

The data card spdtl, with the keyword force or off, will allow the user to force or prevent, respectively, the use of the lattice speed tally enhancement. This allows the user to run a short test case with and without the enhancements and verify they are appropriate if the two runs yield the same tally results. Using spdtl force will also print comments about lattice speed tally enhancement conflicts with other cards.

The lattice speed tally enhancements will greatly reduce the runtime of certain problems, namely large lattices used for voxel phantoms. This enhancement will only work under certain conditions, which MCNP will try to detect. If any of the following criteria are not met, then the lattice speed tally enhancement will not be used unless the spdtl force card is used. Using the spdtl force card to run the lattice speed tally enhancement is discouraged, since it may result in a program crash, tally values which are all zero's, or silent wrong answers.

Criteria which must be met for MCNP to automatically (and appropriately) use the lattice speed tally enhancement:
a) A hexagonal lattice must be present in the geometry.
b) All F4 tallies contain a hexagonal lattice.
c) None of the following cards are used: dxt, dxc, f1, f2, *F4, f6, f7, f8, +f8, pert, wwg, wwge
d) None of the following cards are used to modify a F4 tally: ft, e, em, t, tm, cf, sf, fs, c.
e) All F4 tallies have an associated fm4 card which contains only a single digit multiplier.
f) All F4 tallies have associated de df cards.

The following criteria are not checked by MCNP. It is up to the user to make sure the input deck meets these criteria:
g) Nested lattices are not tallied over.
h) The entries for a cell's fill card do not include that cell's own universe number.
i) The full lattice index range is given on every lattice on each f4 tally card.

For more information, see the LANL Research Note X.

4) Appendix A, page A-10 and A-12 (in the tally section). Add the new line "SPDTL prevent or force lattice speed tally enhancements " and add the appropriate links to page 3-114.

5) Index-9 Add SPDTL with reference to 3-114. Index-6 Add Lattice Tally Enhancements with reference to 3-114.

Testing

The LSTE modifications were tested in three different ways. The first was to run the regression test suite, which by default meet none of the LSME criteria, and check for differences. The second test was to modify the five **lat=1** regression test suite problems so that they are usable with the LSTE, and then compare output with and without the **spdtl off** card. The third method was to use a BNCT input deck and again compare the output with and without the **spdtl off** card.

Running the regression test suite with the LSTE executable was a success. None of the test problems meet the criteria to use the LSTE, so the executed code should not have changed, except for the comment messages that the LSTE will not be used. These warning messages were only be present in the five problems that use a hexagonal lattice: inp15, inp16, inp17, inp24 and inp38.

As a secondary test, these five input decks were changed so that they would use the LSTE. Cards which cannot be used with the LSTE were commented out. Other cards were added or modified to run with the LSTE. Table 6 summarizes the input files, modifications and what is tested. The modified regression test suite problems are found in Appendix E. These cards are discussed in more detail in the following paragraph.

Table 6. Regression test suite modifications to run with LSTE

Input file	Modifications	Purpose
Inp15	Comment out f2 sd2. Add `fill=-3:7 -4:3 -2:2 3 439r` to cell 6. Add `f4:n (6<6[ -3:7 -4:3 -2:2 ]) sd4 1, fm4 1, de4, df4` cards	Test tracking and tallying over lattice which is partially obscured by other cells.
Inp16	Comment out `f2, e2, f6:p, dxt:p` cards Add `fill=-1:1 -1:1 0:0 3 8r` card to cell 4. Add `f4:p (5<5[0:1 0:3 0:0]<3), sd4 1, fm4 1, de4 df4` cards.	Test tallies over multiple lattices. Test tallies of cells within lattices. Test fill card without lattice indicies.
Inp17	Comment out `fq4, f16, f6, f7, f5 *f8` cards Add `f4:n (3<3[-2:2 -2:2 -2:2]), de4 df4 cards. Sd4` card not added	Test tallies with and without sd card. Test in conjunction with kcode problem.
Inp24	Comment out f6, sd6 Changed fill universes to other than own Add f4:n card	Test of lattices within lattices. Test of fill card with own cell universe specified.
Inp38	Comment out cards associated with tallies f4, f14, f24, f34, f44, f54. Add new f4, f14, f24 cards	Test of lattices within lattices. Test of fill card with own cell universe specified.
Cube17	New input deck	Test LSTE with fmesh, f5, fir cards. Test c, fs, e fm cards with these tallies (in addition to LSTE f4 tallies). Test e0 card. Test warning messages with these cards.

Modified regression test suite problems 15, 16, 17 were able to track and yield identical tally results (other than figure-of-merit related items) with and without the LSTE. These test problems exercises the LSTE for geometries which obscure portions of a complete rectilinear lattice, multiple lattices (but not nested), and **kcode** problems. These input decks also tested tallying over certain cells which comprise a lattice cell. All tests were successful.

The inp24 geometry was fairly complex, with nested lattices, and some lattice cells being filled by their own universe number, in this case they were filled with water. A portion of this geometry is shown in Figure 3. Once the tally cards which exclude the use of the LSTE were removed, and the appropriate **fm, sd** and **de df** cards were added, the problem was run with LSTE. The lattice cells being filled by their own universe caused the code to crash. When these cells were replaced by a dummy water cell with a different universe, the problem ran. Unfortunately, only the tallies which corresponded to the lowest level universe (lattice cell 7) were correct. The nested lattice produced silent wrong answers.

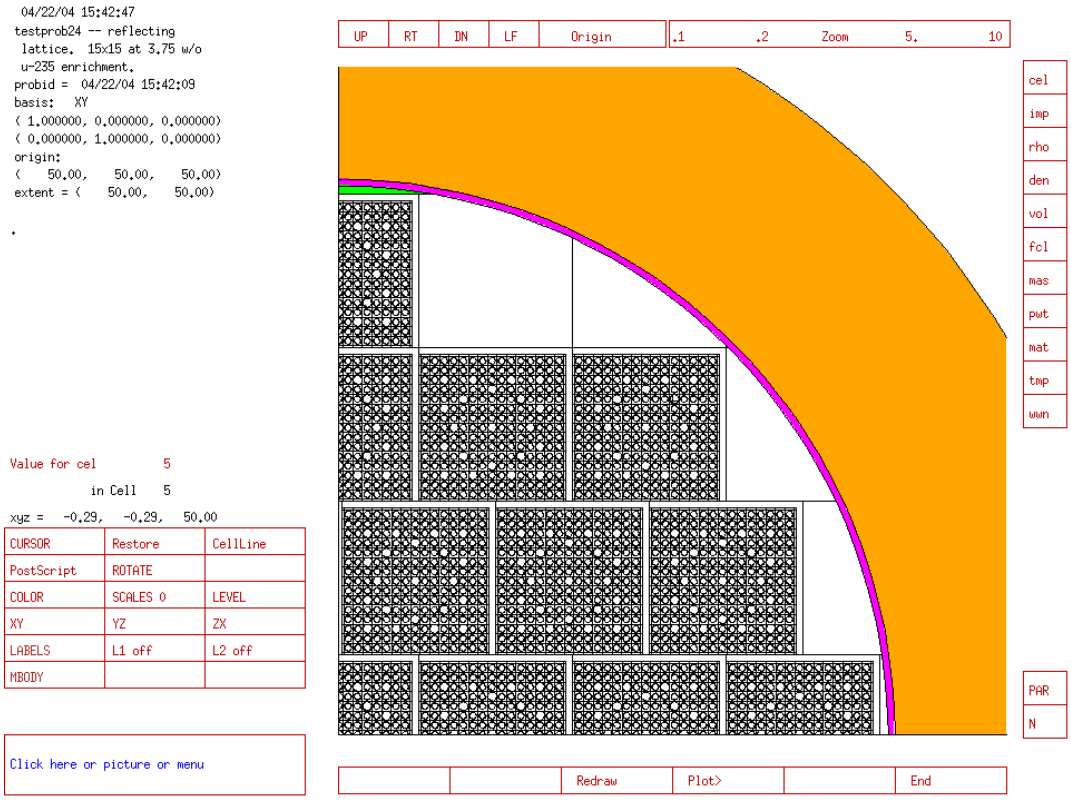


Figure 3. MCNP geometry plot (pz = 50) of inp24. There are nested lattices and lattice cells which are filled by their own universe (ie. void).

Test problem inp38 is similar to inp24. The geometry of this problem contains nested lattices, with some lattice cells being filled by their own universe number. Figure 4 shows this geometry. Once the tally cards which exclude the use of the LSTE were removed, and the appropriate **fm, sd** and **de df** cards were added, the problem was run with LSTE. The lattice cells being filled by their own universe caused the code to crash. When these water cells were replaced by a dummy water cell with a different universe, the problem ran. Only the lowest level lattice, cell 4, tallies were correct, however. The upper-level lattice, cell 3, had a mix of correct, wrong and zero tally values. These differences can be shown by running the input problem in Appendix D with and without the **spdtl off** card.

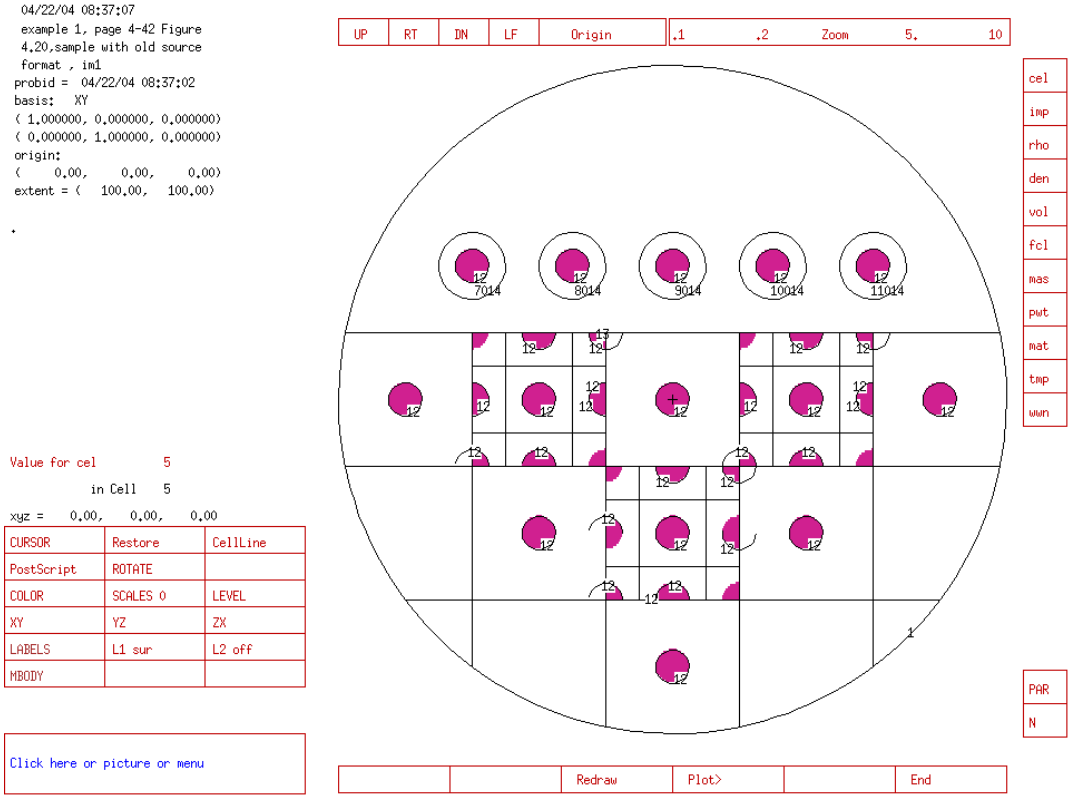


Figure 2. Geometry plot of regression test problem inp38. Nested lattices are present. This causes the LSTE to result in silent wrong answers.

The LSTE was also tested with a simplified BNCT input deck, which is given in Appendix D. It contains f4, f5, mesh and radiography tallies, all of which yield the same tally results with and without the LSTE. It also tests the appropriate use and checking of the fm, de df cards when used legally by the point detectors. The input deck and executable performed correctly in both sequential, mpi, omp, and mpi+omp runs, and also with continue runs.

Results

A typical BNCT problem was used for some simple timing studies with the modified version of MCNP5. For this case, 1 million source particles were used for initial scooping calculations, and 10 million source particles were used for the final run during treatment planning. Wall clock run-times on a Dell Precision 350 (two 2.0 GHz Pentium Xenon Processors, 1 Gbyte of RAM) for these source neutron problems are shown in Table 7. In all cases, the mctal and output files only differ with regard to the time of the run.

Table 7. Decrease in runtimes for BNCT example case.

File	Cutoff	Nps w/ Patch	NPS w/o Patch	Effective Decrease in Runtime
BNCT1	Ctme 2	183625	264	695
BNCT2	Nps 50000	1.44	854.03	593
BNCT1	Nps 1M	11.43	6957.8	609

These results also show that the wall clock runtime improvement is a function of the number of histories run, in addition to slight changes in problem geometry. Usually both source neutron and photon input decks are run, but the neutron input decks are the time limiting step.

Future Improvements

There are a few additional improvements that could be implemented. The tally.F90 modifications could be reviewed for what tally cards/functionality could easily be added without sacrificing the speedup. Other portions of the code might also be optimized to speed up certain classes of problems. The checks for the LSTE that MCNP does not do could be implemented. Checks could be added to determine if nested lattices have been tallied over, the full index range is tallied over, and that fill entries do not use their own universe.

Summary

The lattice speed tally enhancements will greatly reduce the run times of a certain few lattice geometries. MCNP5 is able to detect if most of these circumstances are met, and will automatically use the enhancement if applicable. Using the **force** keyword will cause the LSTE to be used, and print out potential conflicts between the LSTE and some cards. The card **spdtl off** will prevent the LSTE from being used and is useful for comparing short runs with and without the LSTE to confirm a given input deck is compatible with the LSTE. Incorporating the LSTE with the LANL MCNP source will enable more users to have access to this functionality and allow current users to upgrade to MCNP5.

References

1) R.G. Zamenhof, E. Redmond II, G.R. Solares, D.Katz, S. Kiger, and O.K. Harling, "Monte Carlo based treatment planning for boron neutron capture therapy using custom designed models automatically generated from CT data," *International Journal of Radiation Oncology, Biology & Physics*. Vol. **35**, no. 383, 1996.

2) M.R. Palmer, J.T. Goorley, W.S. Kiger, III, P.M. Busse, K.J. Riley, O.K. Harling, and R.G. Zamenhof. Treatment Planning and Dosimetry for the Harvard-MIT Phase I Clinical Trial of Cranial Neutron Capture Therapy. *International Journal of Radiation Oncology, Biology & Physics*. Vol. **53**, no. 5., pp 1361-1379, 2002.

Appendix A: M&C 1998 ANS Conference Paper

This conference paper was the first of two conference papers discussing the speed tally patch. It describes the patch as well as the speed improvements resulting from the patch and implementation of MCNP4B in parallel using PVM on two Windows NT computers. Citation: Goorley, T, G. McKinney, K. Adams, G. Estes, "MCNP Speed Advances in BNCT" *Proceedings of the American Nuclear Society Radiation Protection and Shielding Conference*, Vol II. Nashville, TN, April 1998, pp. 95-98.

MCNP Speed Advances for Boron Neutron Capture Therapy

J. Tim Goorley (Massachusetts Institute of Technology)
Gregg McKinney, Ken Adams, Guy Estes (Los Alamos National Laboratory))

ABSTRACT

The Boron Neutron Capture Therapy (BNCT) treatment planning process of the Beth Israel Deaconess - MIT team relies on MCNP to determine dose rates in the subject's head for various beam orientations. In this time consuming computational process, four or five potential beams are investigated. Of these, one or two final beams are selected and thoroughly evaluated. Recent advances greatly decreased the time needed to do these MCNP calculations. Two modifications to the new MCNP4B source code, lattice tally and tracking enhancements, reduced the wall-clock run times of a typical one million source neutrons run to one hour twenty five minutes on a 200 MHz Pentium Pro computer running Linux and using the GNU FORTRAN compiler. Previously these jobs used a special version of MCNP4A created by Everett Redmond, which completed in two hours two minutes. In addition to this 30% speedup, the MCNP4B version was adapted for use with Parallel Virtual Machine (PVM) on personal computers running the Linux operating system. MCNP, using PVM, can be run on multiple computers simultaneously, offering a factor of speedup roughly the same as the number of computers used. With two 200 MHz Pentium Pro machines, the run time was reduced to forty-five minutes, a 1.9 factor of improvement over the single Linux computer. While the time of a single run was greatly reduced, the advantages associated with PVM derive from using computational power not already used. Four possible beams, currently requiring four separate runs, could be run faster when each is individually run on a single machine under Windows NT, rather than using Linux and PVM to run one after another with each multiprocessed across four computers. It would be advantageous, however, to use PVM to distribute the final two beam orientations over four computers.

1. INTRODUCTION

Boron Neutron Capture Therapy (BNCT) is an experimental bimodal cancer treatment utilizing neutron irradiation and a tumor seeking ^{10}B loaded pharmaceutical. The BNCT treatment planning process of the Beth Israel Deaconess Medical Center (BIDMC) - Massachusetts Institute of Technology (MIT) team relies on the Monte Carlo N-Particle (MCNP1) transport code to determine dose rates throughout a model of the subject's head, or other body part. The model is constructed using patient specific CT data with the aid of

1 MCNP is a trademark of the Regents of the University of California, Los Alamos National Laboratory.

treatment planning software, MacNCTPlan[2,3]. This program allows the medical physicist to combine the thermal and fast neutron, structural and induced gamma, and ^{10}B dose rates calculated by MCNP with appropriate relative biological effectiveness (RBE). It also allows the comparison of RBE dose rates to normal tissue, sensitive structures and tumor from scoping run results of four or five potential beam orientations. Of these, one or two final beams are selected and thoroughly evaluated. Both the scoping and final runs are time intensive computational tasks, limiting the practicality of the BNCT treatment planning process. Increased computational efficiency and running tasks in parallel has greatly decreased the time needed to do these MCNP calculations.

2. BACKGROUND

2.1 MCNP

There are several reasons MCNP is used for the dose rate calculation. It has the ability to accurately represent the subject's head, the irradiation beam's spatial, angular and energy distributions, the flux depression caused by neutron absorption, and the detailed transport and thermalization of epithermal neutrons. MCNP also operates on a variety of computer platforms, including Windows NT, Windows 95 and Linux, all of which operate on PCs. MCNP is a benchmarked reliable code, created and maintained by the Code Integration Group at Los Alamos National Labs.

2.2 Parallel Virtual Machine

The ability to link computers and run tasks in parallel requires additional software. Parallel Virtual Machine (PVM), created and maintained by Oak Ridge National Laboratory, allows a heterogeneous network of computers to be linked together[4]. PVM is accessed by a series of routines used by other programs that allow them to spawn subtasks and pass messages on linked computers. MCNP has included the appropriate PVM calls since MCNP4A, and has additional PVM load balancing routines in MCNP4B. It has been reported for workstation clusters that MCNP speedup using PVM multiprocessing roughly scales with the number of processors used[5].

2.3 Linux

Although MCNP works on a variety of operating systems used by PCs, only Linux supports the parallel version of MCNP, mcnp.pvm. Linux is a UNIX based operating system for PCs, easy to install and maintain. Initial tests showed that the Linux GNU FORTRAN compiler (G77) produced an executable twenty percent slower than the Lahey FORTRAN 90 (LF90) compiler under the Windows NT operating system. Since the runtime increase was significantly smaller than the decrease associated with PVM speedup, Slackware Linux 2.0.29 and PVM 3.3.11 were installed on the BIDMC-MIT group's computers.

2.4 MCNP with PVM

The installation of Linux and PVM is reasonably straight forward. Both are freeware and obtainable via ftp from ftp.cdrom.com and netlib2.cs.utk.edu respectively. MCNP4B is obtainable from the Radiation Safety Information Computational Center (RSICC), and the Linux installation file, fix4b.970723, can be obtained from www-xdiv.lanl.gov/ XTM /world. During the MCNP installation process, PVM should be enabled in MCSETUP, option 5.1[6]. The path for the Linux PVM libraries should be entered. The resulting compiled and linked executable should be named mcnp.pvm and placed in the subdirectory pvm3/bin/LINUX in the users home directory.

3. DESCRIPTION

MCNP can represent the subject's head in a variety of ways. The BIDMC-MIT team converts subject CT data into an arrangement of 1 cm^3 voxels, each a homogenized combination of air, bone, tissue and/or tumor. The model uses 11025 one cm^3 voxels arranged in a twenty-one by twenty-one by twenty-five cm parallelepiped. In MCNP, each voxel can be represented as a standard 3-D cell, or the entire arrangement can be modeled as a lattice.

3.1 Non Lattice Geometry Model

The standard cell geometry of MCNP fully describes the location, material, and boundaries of each of the 11025 cells. The neutron and gamma BNCT problems require forty-six and forty-three Mbytes of memory, respectively. These memory requirements are large enough to prevent MCNP from operating within a Windows NT DOS window. The standard MCNP 4B lattice version decreases these memory requirements to nineteen and fourteen Mbytes, but it prohibitively increases wall clock runtimes.

3.2 Lattice Geometry Model

The lattice geometry model is more complex, requiring multiple "universe" levels. The lattice consists of a single rectangular parallelepiped, defined by the user, repeated infinitely in all directions. Only a portion of this lattice universe level is used for particle transport. The bounding "window" is defined in the higher universe, where the window is filled by the lattice universe. The lattice is filled with other cells that contain the appropriate material information.

The MCNP source code can be greatly optimized for certain lattice applications. To this end, tracking and tallying optimizations were developed for the BNCT problem. The tally optimization removes extraneous energy bins and tally modifiers, while retaining the necessary tally multipliers and DE, DF cards that are used to convert cell averaged neutron fluxes to dose rates. Tracking was made more efficient by removing checks for generalized geometries and specializing only to the lattice geometry enclosed in a parallelepiped. These tracking and tally enhancements can be applied to the MCNP4B source code with an install.fix file and the standard compilation procedure can be used to create a specialized MCNP executable called MCNPBNCT. These optimizations reduce the runtimes by factors of two for the tracking modifications and fifty for the tallying modifications. This reduces the executation times of the lattice model below that of the standard cell model. Furthermore, MCNPBNCT is compatible with the PVM multiprocessing feature, allowing for further reductions in runtimes.

4. RESULTS

The MCNPBNCT executable was compared against the standard MCNP4B release using non lattice geometry, and against the previous executable MCNPNEDH, an optimized lattice version of MCNP4A created by Everett Redmond II. The time decreases were quantified and the dose rates were verified against the previously accepted code.

4.1 Execution Times

The combination of the tracking and tallying enhancements and PVM significantly reduced wall clock runtimes. The single beam scoping evaluation of both the neutron and gamma components, previously totaling one hundred fifty minutes on one 200 MHz Pentium Pro running MCNPNEDH, was reduced to fifty nine minutes using two 200 MHz Pentium Pros. The runtimes for MCNPNEDH, MCNP4B with the lattice and non lattice geometry, and the PVM enabled MCNP4B lattice model with two 200 MHz Pentium Pro's are shown in Fig. 1. Both MCNP4B lattice models use the tracking and tallying optimizations.

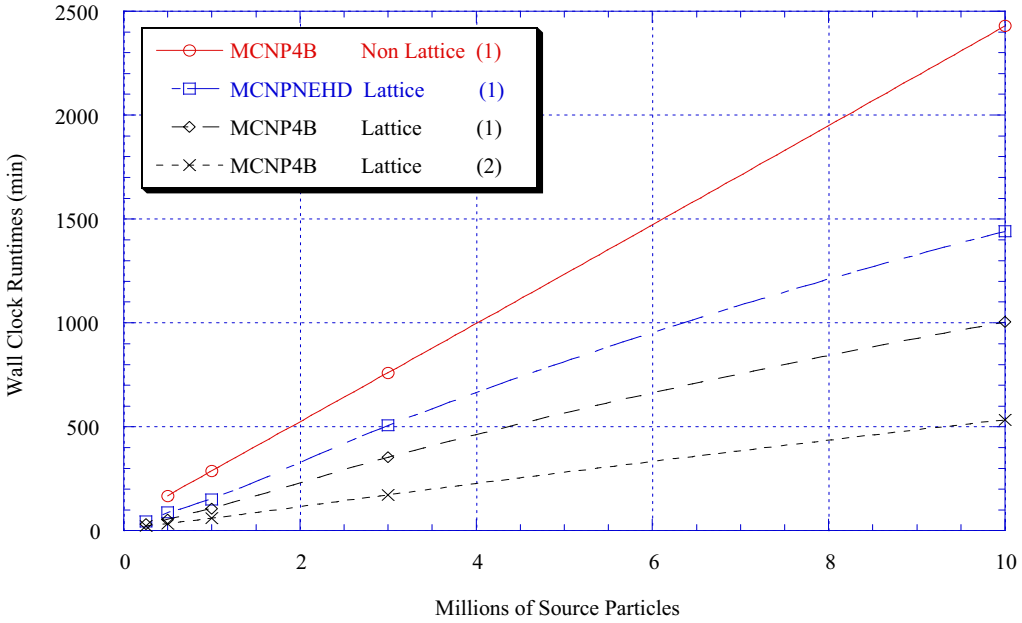


Figure 1. Total wall clock runtimes for a single beam evaluation. The error associated with each point is a few minutes. The numbers in parentheses in the key are how many CPUs were added to the virtual machine.

Scoping runs are typically one or three million source particles, and final evaluations are ten million source particles. This reduces the statistical error from 5% to 1.5% in the regions of interest, in and around the maximum dose and tumor locations.

4.2 Single Task Verification

Before this new version was used during the BNCT treatment planning process, its calculated dose rates were verified with the previous version. This is especially necessary since the lattice enhanced version will not run the MCNP test suite. When ten million source particles were run, the two versions agreed within two percent for cells with RBE dose rates greater than 1 RBE cGy/min, and within four percent for lower dose rate cells.

5. CONCLUSIONS

While using MCNPBNCT with PVM does greatly reduce runtimes for a single beam, actual treatment planning involves multiple beams, typically four. Currently each beam requires a separate MCNP run and the MacNCTPlan computer system consists of only two 200 MHz Pentium CPUs. Thus the number of cases to be run exceeds the number of CPUs and the potential speedup from multiprocessing with PVM is negated (i.e., it is more efficient to run the four cases on 2 CPUs without PVM than to run one case at a time with PVM across 2 CPUs). Due to these CPU limitations, treatment planning uses the non PVM MCNPBNCT executable generated by LF90 on Windows NT. With this executable a four-beam analysis can be completed in 176 minutes instead of 214 minutes required for the LINUX executable. In a similar fashion, the two final runs are most efficiently run under Windows NT without PVM. If a single beam evaluation is needed, then using the LINUX version of MCNPBNCT with PVM will reduce runtimes by nearly a factor of two.

To alleviate this efficiency dependence on number of tasks and operating systems, and to further reduce BNCT treatment planning time, more computers should be added to the virtual machine. It has been shown that the decrease in wall clock runtime is proportional to

the number of linked CPUs. Accordingly, if ten 200 MHz Pentium Pros formed a parallel virtual machine, the one million source scoping run could be reduced to seventeen minutes. This would greatly enhance the practicality of treatment planning, easing scheduling burdens.

6. ACKNOLEDGEMENTS

This research was supported by a U.S. Department of Energy (Office of Health and Environmental Research) grant to Los Alamos National Laboratory. The authors would like to thank the BIDMC-MIT BNCT team, including Stead Kiger, Dr, Dr. Guido Solares, and Dr. Robert Zamenhof for their support.

7. REFERENCES

1. J.F. Briesmeister, Ed., "MCNP - A General Monte Carlo N-Particle Transport Code," v. 4B, Los Alamos National Laboratory report LA-12625-M Version 4B (March 1997).

2. W. S. Kiger, III, et. al. "MacNCTPlan: An Improved Macintosh-Based Treatment Planning Program," *Trans. Am. Nuc. Soc*., 75, 38, (1996).

3. R.G. Zamenhof, et. al. "Monte Carlo-Based Treatment Planning for Boron Neutron Capture Therapy Using Custom Designed Models Automatically Generated from CT Data*." Int. J. Radiation Oncology Biol. and Phys.*, 35 [2], 383-397 (1996).

4. Al Geist, et. all. "PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing" Cambridge, MA. The MIT Press. (1994)

5. G. W. McKinney et al., "Multiprocessing MCNP on an IBM RS/6000 Cluster," *Trans. Am. Nucl. Soc.,* 68, 212 (1993).

6. G. W. McKinney, "A Practical Guide to Using MCNP with PVM," *Trans. Am. Nucl. Soc.,* 71, 397 (1994).

Appendix B: LaJolla 1998 BNCT Conference Paper

This is the second of the two conference papers on the speed tally patch. It describes the patch, the improvements resulting from it, and discusses the differences between results run with and without the patch. Since the speed tally modifications prevent the code from tracking, the calculated dose for each of the 11,025 voxels in the geometric model were compared and found to have acceptable differences. Citation: Goorley, T, G. McKinney, K. Adams, G. Estes, "MCNP Enhancements, Parallel Computing and Error Analysis for BNCT" in Frontiers in Neutron Capture Therapy. Ed. M.F. Hawthorne, K. Shelly, and R.J. Wiersema. Plenum Publishers, New York, 2001, pp. 599-604.

MCNP ENHANCEMENTS, PARALLEL COMPUTING AND ERROR ANALYSIS FOR BNCT

T. Goorley[1], G. McKinney[2], K. Adams[2], G. Estes[3]

[1]Nuclear Reactor Laboratory, Massachusetts Institute of Technology
[2]X-CI, Los Alamos National Laboratory
[3]X-TM, Los Alamos National Laboratory

1. INTRODUCTION

The Boron Neutron Capture Therapy (BNCT) treatment planning procedure used by the Harvard/MIT BNCT Program relies on MCNP2 to calculate dose rates throughout a patient specific model for NCT irradiations. Since MCNP transport calculations are a time consuming portion of the treatment planning process, their acceleration greatly improves treatment planning efficiency. Source code augmentations and implementation of parallel computing on Windows NT computers have greatly decreased the time needed for these dosimetry calculations. A statistical uncertainty analysis verified that the appropriate number of source particles, which varies with the irradiation beam orientation, was being used for treatment planning.

2. MCNP ENHANCEMENTS

MCNP is a well benchmarked, general purpose radiation transport program with extensive capabilities.[1] Some of these capabilities are useful for BNCT dose calculations, including modeling the spatial, energy and angular distributions of the irradiation beam, and calculating the flux depression caused by explicitly increased boron concentrations in the modeled tumor.[2] MCNP can also calculate the dose rates from the thermal and fast neutron reactions in tissue, primarily from nitrogen absorption and hydrogen recoil respectively, neutron absorption by boron, and gamma ray interactions, to every voxel of a lattice model based on the patient specific CT data.[3,4] Some of the other features included in MCNP are not needed for these BNCT calculations, and their removal significantly reduces run time. The enhancements assume that the lattice is a rectangular parallelepiped, and simplifies lattice tracking calculations. The lattice tallying calculations are also accelerated by removing energy bin

[a] MCNP is a trademark of the Regents of the University of California, Los Alamos National Laboratory.

tallies, although the DE and DF cards, which are used in BNCT dose rate calculations to convert energy dependent neutron or photon fluxes to KERMA rates, are kept intact. The tracking improvements account for a factor of 1.2 runtime reduction for neutrons, while the tallying improvements account for a factor of 190. The executable created from the combination of these two modifications is called MCNPBNCT. While MCNPBNCT will not pass the MCNP test suite, which is used to ensure that MCNP was properly installed, MCNPBNCT was verified by running the same input deck as the unmodified MCNP and comparing the results, which were within statistical uncertainty for the converged result from 10 million particles.

Table 1 shows wall-clock runtimes, in minutes, associated with one beam orientation for Harvard/MIT subject 97-4. The asterisks denote estimated times, while the other values are actual times with errors of a few seconds. The wall-clock runtime is slightly greater than the sum of the MCNP startup time, cp0, and the final MCNP runtime, ctm. Wall clock run times are important to clinical treatment planning situations, where the time between dose rate data analysis is a limiting factor.

Table 1. Single CPU MCNP Wall-Clock Run Times in minutes for 10^4 or 10^6 Source Particles

	NPS=10,000		NPS=1,000,000	
Voxel Geometry	Neutron	Gamma	Neutron	Gamma
Lattice	839	86	83,000*	8,400*
Lattice with Tracking Enhancement	704	86	69,000*	8,400*
Lattice with Tally Enhancement	4.4	2.8	86.9	20.7
Lattice with Both Enhancements	4.4	3.2	85.6	20.4

* Indicates an estimate of runtime, based on linear scaling of ctm times from shorter run.

As can be seen from Table 1.1, the dominant time savings comes from modifications of the tally routine in MCNP for this calculation. Only for runs with more than 3 million particles do the tracking modifications contribute to runtime reduction when combined with the tally modification. For shorter runs, the wall-clock runtimes for MCNPBNCT, as shown on the last line in table 1.1, are dominated by the setup times (cp0): 2.73 minutes for neutron source particle runs and 2.36 minutes for gamma source particle runs. The estimated wall clock run times were made by adding the startup time to the linearly scaled particle tracking time. The patch files needed to create MCNPBNCT are available from the first author.

3. PARALLEL COMPUTING WITH MCNP

MCNP has the capability to simultaneously run over a heterogeneous computing network, using a program called Parallel Virtual Machine (PVM), developed at Oak Ridge National Laboratory.[5,6,7] Both software programs were recently modified to allow IBM PCs running the operating systems Windows NT to be linked together, along with the other supported systems (UNIX Sun Solaris, SGI Irix, PC Linux, etc.), allowing the MCNP transport calculation runtimes to be reduced by a factor equivalent to the number of computers being used.[8]

The primary obstacle in the development of a Windows NT parallel version of MCNP was the mixed language programming, where real numbers, integers and character strings had to be passed correctly between MCNP, written in FORTRAN, and PVM, written in C. This required compilers able to compile each program's source code, and a common linker able to link together each compiler's object files. The recent release of Digital Visual FORTRAN 5.0,

compatible with Microsoft Visual C/C++ 4.0, fulfilled these requirements. Although several difficulties were encountered, attributed to the beta status of version of PVM 3.4 beta 6 for Windows, modifications were developed that allow MCNP4B to effectively pass the MCNP parallel test suite.

Table 2 shows the wall clock runtimes, in minutes, for MCNPBNCT running in parallel under Windows NT on up to four 200 MHz Pentium Pro computers. The third and fourth CPU used were on a dual CPU motherboard in one computer.

Table 2. Parallel MCNPBNCT Wall-Clock Runtimes.

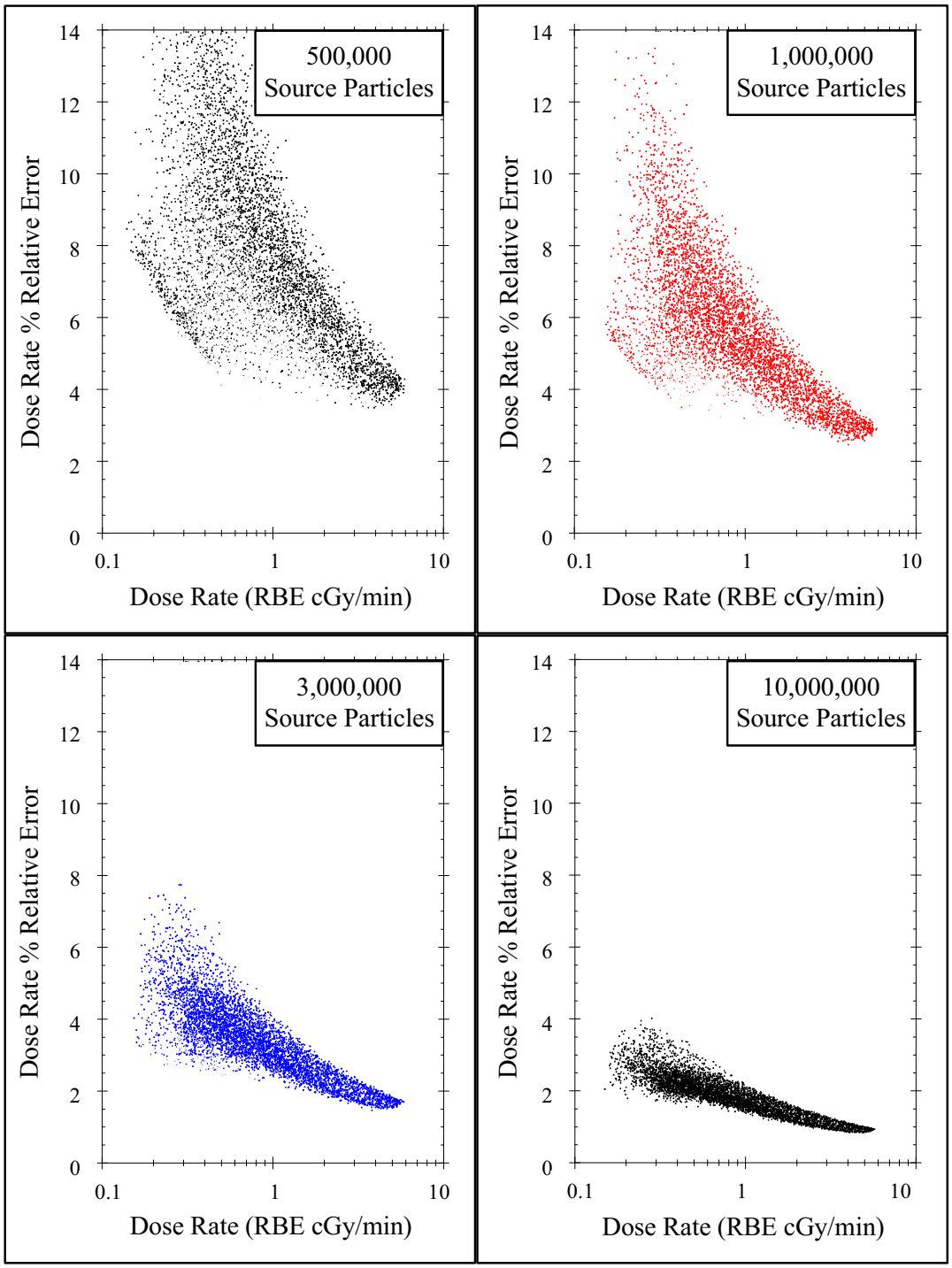
Number of CPUs used	Number of Spawned Tasks	NPS = 10,000	NPS = 1,000,000
1	Not run in Parallel	7.6	106
2	2	15	99
3	2	8	64
3	3	12	59
4	3	8	44

Various options in PVM and MCNP can be used to reduced the total wall-clock runtime of the dose rate calculation. Although the MCNP master process, which performs startup calculations and directs spawned tasks, typically does not run any particles, it does use a portion of the computational power, constantly checking to see whether the subtasks have finished. The master task can be run on its own CPU, when the number of spawned tasks is less than the number of linked CPUs available. As shown in lines three and four of Table 2, it is more efficient to have the master on its own CPU only for short runs. Since tasks spawned by MCNP are assigned sequentially through the PVM configuration listing, it is possible to make a PVM configuration of dual CPU mother board computers followed by single processor computers, and have MCNP assign two processes only to the dual CPU computers. In MCNP4B, the tasks command line option allows the optional use of a negative number of spawned processes, which disables PVM load balancing[9], providing some runtime reduction for a homogeneous computing network. The MCNP PRDMP card controls the communication frequency between spawned and master processes.[1]

4. VOXEL DOSE RATE STATISTICAL UNCERTAINTY ANALYSIS

In addition to calculating the dose rates to each voxel, MCNP also calculates the statistical uncertainty associated with each voxel for each dose component. It is important that high dose regions, particularly around the region of interest, such as the tumor and peak dose rate location, have statistical uncertainties low enough to ensure statistically significant comparisons between different beam orientations. Typically the statistical uncertainty associated with these high dose region is 5% for scooping runs, and 1% for final planning runs. Figure 1 shows the dose rate and associated statistical uncertainty for various number of source particles for Harvard/MIT BNCT Subject 97-3. Each dot represents a single voxel.

Figure 1 Dose Rate Statistical Uncertainty for Various Number of Source Particles.



As expected, higher dose rate regions have more particles passing through them, and have lower statistical uncertainties. When the region of interest is near the beam entrance, it will be in a high dose rate region, and will require fewer number of source particles (~1 Million) to

achieve a 5% uncertainty. When the entrance location is located far from the region of interest, as in the case of an contralateral beam, between 1 and 3 million particles must be run to achieve the same amount of statistical uncertainty. An example of a statistically significant contribution from a contralateral beam to a primary beam's high dose rate region is: the dose rate to the region of interest from a contralateral beam is 0.5 RBE cGy/min +/- 0.05 RBE cGy/min, while the primary beam is in 5 RBE cGy/min +/- 0.25 RBE cGy/min If the primary beam calculation had been run with fewer source particles, its peak dose rate statistical uncertainty would be the same as the second beam's dose rate contribution to that point.

5. CONCLUSIONS

This paper presents three approaches to reduce the total wall clock runtimes of MCNP dose rate calculations. MCNP modifications which removed unnecessary calculations, particularly unnecessary energy bin tallying, significantly reduced wall-clock runtimes. Modifying the existing PVM capability within MCNP to work with Windows NT allowed additional office computers to be used during treatment planning calculations, further reducing the calculation time. A statistical uncertainty analysis was used to determine whether excessive number of source particles were being run, but the appropriate number was already being used. The total wall clock runtime of a typical one million source particle beam dose rate calculation running on four 200 MHz Pentium Pro CPUs was reduced to forty-four minutes.

6. ACKNOLEDGEMENTS

This research was supported by grants from the U.S. Department of Energy to the Radiation Transport Group at Los Alamos National Laboratory, with subsequent subcontracts to Harvard/MIT. The authors would like to thank the Harvard/MIT BNCT team, including Stead Kiger, Dr. Matthew Palmer and Dr. Robert Zamenhof for their support. The MCNP enhancements for BNCT were, in part, first proposed by LANL in a paper presented at the Fifth International Symposium on Neutron Capture Therapy[10].

7. REFERENCES

1. J.F. Briesmeister, Ed., "MCNP - A General Monte Carlo N-Particle Transport Code," v. 4B, Los Alamos National Laboratory report LA-12625-M Version 4B, March 1997.

2. R.G. Zamenhof, E.L. Redmond II, G.R. Solares, D. Katz, W.S. Kiger III, and O.K. Harling. "Monte Carlo-Based Treatment Planning for Boron Neutron Capture Therapy Using Custom Designed Models Automatically Generated from CT Data*." Int. J. Radiation Oncology Biol. and Phys.*, 35[2]:383-397, 1996.

3. T. Goorley, G.W. McKinney, K. Adams, G.P. Estes. "MCNP Speed Advances for Boron Neutron Capture Therapy." Trans. ANS Rad. Prot. Shield. Div. Topical Conf. Apr. 19-23 1998, **2,** 95**.**

4. W.S. Kiger III, R.G. Zamenhof, G.R. Solares, E.L. Redmond II, and C. Yam. "MacNCTPlan: An Improved Macintosh-Based Treatment Planning Program," *Trans. Am. Nuc. Soc*., 75:38, 1996.

5. Al Geist, A. Benguelin, J. Dongarra, R. Manchek, W. Jiang, and V. Sunderman. "PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing," The MIT Press, Cambridge, 1994.

6. J. Dongarra, A. Geist, R. Manchek, and V. Sunderam, "Integrated PVM Framework Supports Heterogeneous Network Computing", *Comp. Phys*. 7:166-175, 1993.

7. G. W. McKinney, "Parallel Processing Monte Carlo Radiation Transport Codes," Pro. 8th ICRS, Arlington, TX, Apr 24-28 1994.

8. G. W. McKinney et al., "Multiprocessing MCNP on an IBM RS/6000 Cluster," *Trans. Am. Nucl. Soc.,* 68:212, 1993.

9. G. W. McKinney, "A Practical Guide to Using MCNP with PVM," *Trans. Am. Nucl. Soc.,* 71:397, 1994.

10. Guy P. Estes and William M. Taylor, "Potential MCNP Enhancements for NCT", Advances in Neutron Capture Therapy, Albert H. Soloway, Roth F. Barth and David E. Carpenters (Eds.), Plenum Press, 1993.

Appendix C: Original MCNP4B Speed Tally Patch

```
*/ ------------------------------------------------------------- track
*/ Speed up the lattice tracking.                    02/18/97  (KJA/GWM)
*i,tr.22                                                          <23367>
      if(ll.eq.2)go to 380
*d,tr.258,tr.261                                              <23611-23614>
  320 if(il.eq.0)go to 380
      if(nlt.eq.0)go to 380
      call chkcel(ic,3,j)
*d,tr.266                                                         <23619>
      if(dl(ll).eq.huge)go to 380
*d,tr.275                                                         <23628>
*d,tr4b.14,tr.285                                             <23640-23641>
      if(ll.gt.0)go to 10
      jk=-mfl(lmfl+1,nint(udt(7,1)))
      if(nint(udt(8,1)).eq.laf(jk+1,1))go to 10
      if(nint(udt(9,1)).eq.laf(jk+2,1))go to 10
      if(nint(udt(10,1)).eq.laf(jk+3,1))go to 10
      if(nint(udt(8,1)).eq.laf(jk+1,2)-laf(jk+1,1)-1)go to 10
      if(nint(udt(9,1)).eq.laf(jk+2,2)-laf(jk+2,1)-1)go to 10
      if(nint(udt(10,1)).eq.laf(jk+3,2)-laf(jk+3,1)-1)go to 10
      dl(ll)=huge
*/
*/ ------------------------------------------------------------- tally
*/ Speed up the lattice tallying.                    11/04/96  (GWM/GWM)
*d,ty4b.1,ty.358                                              <30491-30868>
      li=lipt
      lj=ljpt
      lx=lo+2
      do 10 ml=1,itds(lo)
      ta=wgt
      ital=itds(lx-1)
      n1=itds(lx)
      j=-mfl(lmfl+1,nint(udt(7,lev-1)))
c      ir=nint(udt(8,lev-1))-laf(j+1,1)+laf(j+1,2)*(nint(udt(9,lev-1))-
c     & laf(j+2,1)+laf(j+2,2)*(nint(udt(10,lev-1))-laf(j+3,1)))+1
      ir=iii-laf(j+1,1)+laf(j+1,2)*(jjj-
     1 laf(j+2,1)+laf(j+2,2)*(kkk-laf(j+3,1)))+1
      j7=ktal+jptal(lj+5,ital)+iptal(li+2,5,ital)*(itds(lx+ir)-1)+1
      ta=dosef(ta)
      td=ta*dr*tds(iptal(li+5,2,ital)+1)
      if(tal(j7).eq.0.)then
       jtls=jtls+1
       if(jtls.le.ktls)tal(ktal+nmxf*mxf+jtls)=j7
      endif
      tal(j7)=tal(j7)+td
   10 lx=lx+n1+2
*/
```

Appendix D: An Example of Voxel Phantom Geometry.

The following is an example of three 16x16x16 hexagonal lattices of two materials. With different constituent materials, it would serve as an example of how to set-up a voxel phantom in MCNP.

```
message: o=cube27.o r=cube27.r mctal=cube27.m c=cube27.c

 lattice based Tissue Cube 16 x 16 x 16
c     The fill card should be fill= -n/2:n/2-1 -n/2:n/2-1 z/2:z/2-1
c        Where n is the number of pixels per row and z Axial Slices
 100    0   -113 112  -213 212  -313 312
       lat=1  fill=  -8:7  -8:7   -8:7   8 1999r 9 1799r
       10 295r  u=100 $ # of fill entries=16x16x16=4096=2000+1800+296
  1      1  -1.04700E+00     -70  u= 8
  2      2  -0.001           -70  u= 9
  3      3  -1.              -70  u=10
 101   0 111 -114 211 -214 311 -314 fill=100
 201  like 101 but trcl=( 50 0 0 )
 300    0   -113 112  -223 222  -313 312
       lat=1  fill=  -8:7  -8:7   -8:7   10 4095r  u=300
 301  0  111 -114 221 -224 311 -314 fill=300
 102   2 -1.0 #201 #301 -1000 ( -111: 114: -211: 214: -311: 314)
 103   0   1000
c     BLANK LINE

c     BLANK LINE
c   n = number of pixels per row of square image
c   R = resolution of image (mm/pixle)
c   z = number of axial slices
c   Z = height of axial slice
  111  px    -12.5056   $ Should be -nR/2  lattice boundary
  112  px      0.0      $ Should be 0.0    single voxel boundary
  113  px      1.5632   $ Should be  R     single voxel boundary
  114  px     12.5056   $ Should be  nR/2  lattice boundary
  211  py    -12.5056
  212  py      0.0
  213  py      1.5632
  214  py     12.5056
  221  py     37.4944
  222  py     50.0
  223  py     51.5632
  224  py     62.5056
  311  pz    -12.40000   $ Should be -zZ/2
  312  pz      0.0
  313  pz      1.55
  314  pz     12.40000   $ Should be  zZ/2
 1000   so  50.56417E+01
   70   so  5.56417E+01
c     BLANK LINE

c     BLANK LINE
mode    n p
prdmp   2j 1 1
phys:p  100 0
phys:n   20.0 0.0
imp:n     1     8r  0
```

```
imp:p     1     8r  0
print   110
m1     1001.50c -0.1058466  6012.50c -0.1395933  7014.50c -0.0184255
       8016.50c -0.7269068 15031.50c -0.0039054 17000.50c -0.0014019
      19000.50c -0.0039054  5010.50c -0.0000150
mt1     lwtr.01t  $ Use S(alpha,beta) data for n transport in tissue
m2     1001.50c 1
mt2     lwtr.01t
m3     8016.50c 1
c Example Source definition, two beams which intersect lattices
c  Note that if vex were 1.00 0 0, and dir=1 then the particles are parallel
c  to the lattice grid, potentially resulting in 'Zero Lattice Hit' error
sdef  x=-15.00   y = d2 z= 0.00000000  dir=d3  rad=d1 erg=10.0
      vec=   1.00000000   0.00000000   0.00100000
      axs=   1.00000000   0.00000000   0.00000000
si1   0  5
si2  L 0.00 50.00
sp2      1    1
si3   H   0.9  1.0  $ Foreward beam, almost parallel to vec direction
sp3     0    1
c
#         de14           df14  $ DE DF cards must be present, otherwise
          1.00E-11       1     $  the lattice tally results will be all
          100.0          1     $  zeros.
#         de24           df24  $ These DE DF cards result in no change to
          1.00E-3        1     $  the tallies, ie. the Fx4: tallies will
          15.0           1     $  report flux.
#         de64           df64  $ Use DE DF cards to convert flux to Kerma
          1.00E-11       1
          100            1
#         de74           df74
          1.00E-8        1
          100            1
fm14        1.5E+13 $ FM card for F4 must be present and must have only
fm24        1.5E+13 $   1 entry to use lattice speed tally modifications.
fm64        1.5E+13 $   Use single digit multiplier to take into account
fm74        1.5E+13 $   source strength
fm54  (-1 3 -1)     $ FM card for FMESH card operates as normal.
e15 0.5 y10  $ e bin card can be used w/ radiography, point & ring, not f4
e5    0.5  10
e0    0.5  10
c F#4 tallies must have full path to lattice and must cover entire lattice
c   used for particle transport to use lattic speed tally enhancement
f14:n   (100<100[ -8:7  -8:7   -8:7])
f24:n   (100<100[ -8:7  -8:7   -8:7])
f64:p  (100<100[ -8:7  -8:7   -8:7])
f74:n  (300<300[ -8:7  -8:7   -8:7])
f5:p   50 50 50 1.0
f35:n  52 52 52 1.0
c FMESH card does not affect lattice speed tally - use fmesh normally
fmesh54:n geom=xyz origin -20 -20 -20 imesh -12 12 20 iints 1 24 1
       jmesh -12 12 20 jints 1 24 1   kmesh -12 12 20 kints 1 24 1
fir15:n 100 0 0   0   50 0 0   0   0   0  $ Radiography Tally
c15   -50.0 3i 50.0 $ c  card can be used with fir, fip or fic, not f4
fs15  -50.0 3i 50.0 $ fs card can be used with fir, fip or fic, not f4
sd14    3.787571072 $ enter volume of single lattice voxel
sd24    3.787571072
sd64    3.787571072
```

```
sd74    3.787571072
 vol     no
nps    500
c spdtl  force $ or OFF to use/not use Lattice Speed Tally Eatch
c This input deck should use the LSTE automatically, w/o spdtl force
```

Appendix E: Modified Regression Test Problems.

The following input files are modifications of the regression test suite. They have been modified so that the lattice speed tally modifications will be applied automatically. They track and produce the same tally results with either spdtl off or spdtl force, except for the tallies over the nested lattices in problems 24 and 38.

```
INP15
testprob15 -- test filled lattice and skewed lattice.
1     1  -.6 -1                                  imp:n=1
2     0  1 -2 -4  fill=1 (-6 -6.5 0)                 imp:n=1
3     0  2 -3 -4  *fill=2 (-7 5 0 30 60 90 120 30 90)   imp:n=2
4     0  2 3 -4   *fill=2 (4 8 0 15 105 90 75 15 90)    imp:n=2
5     0  4                                       imp:n=0
6     0  -5                fill=-3:7  -4:3 -2:2 3 439r
         u=1   lat=1       imp:n=1
7     3  -2.7  -11 12 -13 14 -15 16   u=2   lat=1       imp:n=1
8     2  -.8   -17                u=3             imp:n=1
9     0  17                      u=3             imp:n=1

1     sy -5 3
2     py  0
3     px  0
4     so  15
5     box  -1.5 -1 -3  3 0 0  0 2 0  0 0 6
11    p  1 -.5 0 1.3
12    p  1 -.5 0 -1.3
13    py  .5
14    py  -.5
15    pz  3
16    pz  -3
17    sq  1 2 0 0 0 0 -1 .2 0 0

sdef  pos 0 -5 0 erg d1 rad d2
si1   0 10
sp1   0 1
si2   3
sp2   -21
c f2:n  3
c sd2   1
f4:n  (6<6[ -3:7  -4:3 -2:2 ])
sd4   1
fm4 1
de4 1E-11 20
df4 1    1
m1    4009.60c 1
m2    6012.40c 1
m3    13027.40c 1
drxs
nps   2000
print 72 128 160 161 162
prdmp 2j -1
c spdtl off
```

```
INP16
testprob16 -- test general source in a lattice.
1  0  1:-3:-4:5:6:-7  imp:p=0
2  0  -2 3 4 -5 -6 7   imp:p=1  fill= 1 (-25 0 0)
3  0  -1 2 4 -5 -6 7  imp:p=1  fill=2 (0 -20 0)
4  0  -11 12 -14 13  imp:p=1  lat=1  u=1 fill=-1:1 -1:1 0:0
    3 3 3   3 3 3   3 3 3
5  0  -15 16 -18 17  imp:p=2  lat=1  u=2
c       interrupt card
        fill=0: &
1 0:3 0:0 4 4 4(5 0 0) 4 4 5 4 4
6     1 -.9  21:-22:-23:24  imp:p=1  u=3
7     1 -.9  19  imp:p=1  u=4
8     2 -18  -21 22 23 -24  imp:p=1  u=3
9     1 -.9  20(31:-32:-33:34) 9  imp:p=1  u=5
11    2 -18  -19  imp:p=1  u=4
12    1 -.9   -9  imp:p=1  u=5
13    2 -18  -20  imp:p=1  u=5
15    2 -18  -31 32 33 -34  imp:p=1  u=5

1     -3 px 50
2 px 0
3 -1 px -50
4     -5 p 0.00000000001 1 0.000 -20
5   -4 p 0.00000000001 1 0.000 20
6+    pz 60
7+    pz -60
9     s 5 5 3 .5
11    px 8.334
12    px -8.334
13    py -6.67
14    py 6.67
15    px 25.0000001
16    px -.0000001
17    py -.0000001
18    py 10.0000001
19    c/z 10 5 3
20    c/z 10 5 3
21    px 4
22    px -4
23    py -3
24    py 3
31    px 20
32    px 16
33    py 3
34    py 6

mode p
m1    6012.50m .4 8016.50m .2
m2    92238.50m .98  92235.50m .02
sdef  erg fcel d1  cel d6  x fcel d11  y fcel d13  z fcel d15 &
    rad fcel d17  ext fcel d19  pos fcel d21  axs fcel d23
ds1   s d2 d3 d4 d5
# sp2 sp3 sp4 sp5 &
-2 -2 -2 -2 &
     1.2 1.3 1.4 1.42 &
si6   s d7 d8 d9 d10
sp6   .65 .2 .1 .05
si7   l -2:4:8
sp7   1
si8   l 3:5(0 0 0):11 3:5(1 0 0):11 3:5(0 1 0):11 3:5(1 1 0):11
      3:5(0 2 0):11 3:5(0 3 0):11 3:5(1 3 0):11
sp8   1 1 1 1 1 1 1
si9   l 3:5(1 2 0):13
sp9   1
si10  l 3:5(1 2 0):15
```

```
sp10  1
ds11  s d12 0 0 d25
ds13  s d14 0 0 d26
ds15  s d16 0 0 d16
ds17  s 0 d18 d18 0
ds19  s 0 d20 d20 0
ds21  s 0 d22 d22 0
si22  l 10 5 0
sp22  1
ds23  s 0 d24 d44 0
si24  l 0 0 1
sp24  1
si44 s 45 46
sp44 .5 .5
si45 l 0 0 1
sp45 1
si46 l 1 1 1
sp46 1
# sp12 si12 si14 sp14 si16 sp16 si18 sp18 si20 sp20 si25 sp25 si26 sp26
0 -46 -17 0 -60 0 0 -21 -60 0 16 0 3 0
1 -4   17 1  60 1 3   1  60 1 20 1 6 1
c f2:p  2
c e2    .1 1 20
c f6:p  2 4 6 8   7 11 12 13 15
c sd6   1 1 1 1   1  1  1  1  1
c print 10 70 128 170
print
c fq6  f e
c f5:p -48 -18 -58 0
c dxt:p 30 5 3 0.6 0.9
nps   2000
mgopt f 12
prdmp 2j -1
elpt:p 0 4r .05 .06 .05 .06 3r
phys:p 1
cut:p 30
f4:p  (5<5[0:1 0:3 0:0 ])  (15<5[0:1 0:3 0:0 ]<3) (11<5[0:1 0:3 0:0 ]<3)
sd4       1.0                  1.0                 1.0
fm4 1
# de4   df4
  1E-3   1
  100    1
f14:p  (4<4[-1:1 -1:1 0:0 ])  (6<4[-1:1 -1:1 0:0 ]<2) (8<4[-1:1 -1:1 0:0 ]<2)
sd14       1.0                  1.0                 1.0
fm14 1
# de14   df14
  1E-3   1
  100    1
spdtl off
```

```
INP17
testprob17 -- kcode in a rectangular finite lattice.
c     finite lattice fails in mcnp4.2
3 0 -8 7 -10 9 -12 11  lat=1  imp:n=1 &
                fill=-2:2 -2:2 -2:2 2 124r
4     0            -13:14:15          u=2    imp:n=1
5 &
2 &
 .100691    13 -14 -15 #6      u=-2   imp:n=1
6     1 &
  .0983726   16 -17 -18         u=-2   imp:n=1

7     px -17.15
8     px  17.15
9     py -17.15
10    py  17.15
11    pz -16.5
12    pz  16.5
13    pz  -9.5
14    pz   9.5
15    cz  10.15
16    pz  -8.839
17    pz   8.839
18    cz   9.489

mode n p
kcode 30 1 4 40
fcl:n 0 0 1
m1    1001.50m .057776  7014.50m .002131  8016.50m .037403 &
92238.50m .0009846 &
92235.50m .1000062
m2    1001.50m .053702  6012.50m .033564  8016.50m .013425 &
92236.50m .05
f4:n (3<3[-2:2 -2:2 -2:2])
fm4 1
de4 1E-11 20
df4 1     1
c fm4 ((1 2 (102)(101))(1.0 -1 2 0.000502)) (-1 1 -1) (-1 1 -2 -4) &
c  (-1 1 -3) (-1 1 -5) (-1 1 301)
c fq4 m e
c f16:p 6
c f6:n 6
c f7:n 6
c f5:p 0 0 0 0
c *f8:n (6<3[0 0 0])
mgopt f 42
print 128
prdmp 2j -1
phys:n 100 0.01
spdtl force
c sd4 3.88242E+04 $ sd card not needed!  Correct volumes calculated.
```

INP24

```
testprob24 -- reflecting lattice.  15x15 at 3.75 w/o u-235 enrichment.
1    1 -10.182 -1 u=2
2    2 -.001 1 -2 u=2
3    3 -6.55 2 -3 u=2
4    4 -1.0 3 u=2
5    4 -1.0 -14:15 u=3
6    3 -6.55 14 -15 u=3
7    4 -1.0 -4 +5 -6 +7 u=1 lat=1 fill=-8:8 -8:8 0:0
     21 17r 2 14r 21 21 2 14r 21 21 2 2 3 2 2 3 2 2r 3 2 2
     3 2 2 21 21 2 6r 3 2 6r 21 21 2 3r 3
     2 4r 3 2 3r 21 21 2 2 3 2 8r 3 2 2
     21 21 2 14r 21 21 2 2r 3 2 2r 3 2 2r
     3 2 2r 21 21 2 14r 21 21 2 2 3 2 8r 3
     2 2 21 21 2 3r 3 2 4r 3 2 3r 21 21
     2 6r 3 2 6r 21 21 2 2 3 2 2 3
     2 2r 3 2 2 3 2 2 21 21 2 14r 21 21 2 14r 21 17r
8    0 -8     -10     -12 u=4 fill=1
9    5 -7.9 8:10 u=4
10   4 -1.0 -8     -10     +12 u=4
11   4 -1.0 -16 +9         u=5 lat=1 fill=0:6 0:0 0:0 4 3r 21 2r
12   0 +28 +29 -19 -17 +13 -18 fill=5
13   0 +28 -19 +17 -31 +13 -18 fill=5 (-11.5 23 0)
14   0 +28 -19 +31 -32 +13 -18 fill=5 (-23 46 0)
15   0 +28 -19 +32 -33 +13 -18 fill=5 (-69 69 0)
16   4 -1.0 +28 -19 +33 +13 -18
17   4 -1.0 +28 +29 -19 -24 +18
18   6 -7.9 (+28 +29 +19 -20 +23 -25):(+28 +29 -19 +23 -13)
     :(+28 +29 -19 +24 -25)
19   7 -7.088254305 (+28 +29 +20 -21 +22 -25):(+28 +29 -20 +22 -23)
20   0 -28:-29:+21:-22:+25
21   0   -34 u=21

1    cz  .464693
2    cz  .483743
3    cz  .535940
4    px  .71501
5    px -.71501
6    py  .71501
7    py -.71501
8    px  11.0
9    px -11.0
10   py  11.0
12   pz  400.903
13   pz  34.0
14   cz  .652018
15   cz  .690118
16   px  12.0
17   py  12.0
18   pz  439.0
19   cz  82.25
20   cz  83.25
21   cz  116.35
22   pz  0.0
23   pz  33.0
24   pz  447.9
25   pz  485.9
*28  px  0.0
*29  py  0.0
31   py  35.0
32   py  58.0
33   py  81.0
34   so  500.0

imp:n 1 18r 0 1
nonu 1 18r 0
```

```
kcode 200 .7 1 3 4500 0
ksrc 1.5 1.5 217.4515
m1   92235.50d 1.31964e20   92237.50d 1.31964e20 92238.40c 2.15905e22
m2    8016.40c 1.00000000
m3   41093.40c 1.
m4   1001.60c .666666667    8016.40c .333333333
m5   26058.60c -.68874500    5010. -.00178200    5011.40c -.00721800
m6   26058.60c -.69500000
m7   26058.60c .830266962    6012.40c .133437328
mt4  lwtr.01t
drxs
prdmp 2j -1
c f6:n  8 12 13 14 15 $ heating in mat=0 cells kills mcnp4.2
c sd6   1 4r
c 0  7[ 3 -8 0] <    8 < 11[0 0 0] <   12
f4:n  (12<11<11[0:6 0:0 0:0]<8<7<7[-8:8 -8:8 0:0])
sd4 1                          $ Tally results silently wrong!
fm4 1
de4 1E-11 100
df4   1    1
f14:n  (7<7[-8:8 -8:8 0:0])       $ Tally results agree spdtl
sd14 1
fm14 1
de14 1E-11 100
df14   1    1
c f24:n  (11<11[0:6 0:0 0:0]<8<7) $ Paths to lattice cell 11 result
c sd24 1                        $   in crash.
c fm24 1
c de24 1E-11 100
c df24   1    1
print
c spdtl off
```

```
INP38
example 1, page 4-42 Figure 4.20,sample with old source format , im1
   1  0     -1  -2   3  13          fill=4
   2  0     -1  -2   3 -13          fill=1
   3  0     -4   5  -6   7     u=1    lat=1
     fill=-2:2 -2:0 0:0   15 15 3 15 15   15 3 2 3 15   3 2 3 2 3
   4  0     -8   9 -10  11     u=2    lat=1
        fill=-1:1 -1:1 0:0  3 3 3  3 3 3  3 3 3
   5  1 -.1 -12                u=3
   6  0     12                u=3
   7  0    -14  -2   3        u=4   fill=3       trcl=(-60 40 0)
   8  like 7 but                              trcl=(-30 40 0)
   9  like 7 but                              trcl=(0 40 0)
  10  like 7 but                               trcl=(30 40 0)
  11  like 7 but                               trcl=(60 40 0)
  12  0     #7 #8 #9 #10 #11    u=4
  13  0     -99  (1:2:-3)
  14  0      99
  15  0     -99              u=15

   1  cz  100
   2  pz  100
   3  pz -100
   4  px   20
   5  px  -20
   6  py   20
   7  py  -20
   8  px   10
   9  px  -10
  10  py   10
  11  py  -10
  12  cz    5
  13  py   19.9
  14  cz   10
  99  so  142

mode   n
imp:n   1  12r  0 1
m1     1001  2   8016  1
mt1    lwtr.01t
print
prdmp    j j -1
c       total n source strength = 4.882e6  n/s per assembly, 3 assemblies
sdef   cel=d5  pos=0 0 0  axs=0 0 1  ext=d3  rad=d4
sc3    axial distribution uniform
si3    -100 100
sp3    -21 0
sc4    radial distribution proportional to r (uniform in volume)
si4    5.0
sp4    -21 1
sc5    sample in cell 5, 5 different ways using new source format
si5   l 1:7:5 1:8:5 1:9:5 1:10:5 1:11:5                        $ 1
       2:3(0 -2 0):5    2:3(-1 -1 0):5    2:3(1 -1 0):5       $ 1
       2:3(-2 0 0):5    2:3(0 0 0):5      2:3(2 0 0):5        $ 1
       2:3(0 -1 0):4(-1 -1 0):5    2:3(0 -1 0):4(0  -1 0):5   $ 1
       2:3(0 -1 0):4( 1 -1 0):5    2:3(0 -1 0):4(-1  0 0):5   $ 1
       2:3(0 -1 0):4( 0  0 0):5    2:3(0 -1 0):4( 1  0 0):5   $ 1
       2:3(0 -1 0):4(-1  1 0):5    2:3(0 -1 0):4( 0  1 0):5   $ 1
       2:3(0 -1 0):4( 1  1 0):5                               $ 1
       2:3(-1 0 0):4(-1 -1 0):5    2:3(-1 0 0):4(0  -1 0):5   $ 1
       2:3(-1 0 0):4( 1 -1 0):5    2:3(-1 0 0):4(-1  0 0):5   $ 1
       2:3(-1 0 0):4( 0  0 0):5    2:3(-1 0 0):4( 1  0 0):5   $ 1
       2:3(-1 0 0):4(-1  1 0):5    2:3(-1 0 0):4( 0  1 0):5   $ 1
       2:3(-1 0 0):4( 1  1 0):5                               $ 1
       2:3(1  0 0):4(-1 -1 0):5    2:3(1  0 0):4(0  -1 0):5   $ 1
       2:3(1  0 0):4( 1 -1 0):5    2:3(1  0 0):4(-1  0 0):5   $ 1
```

```
      2:3(1  0 0):4( 0  0 0):5   2:3(1  0 0):4( 1  0 0):5   $ 1
      2:3(1  0 0):4(-1  1 0):5   2:3(1  0 0):4( 0  1 0):5   $ 1
      2:3(1  0 0):4( 1  1 0):5                              $ 1
      (5<7 8 9 10 11<1)                                     $ 2
      (5<3[0 -2 0]<2)  (5<3[-1 -1 0]<2)  (5<3[1 -1 0]<2)      $ 2
      (5<3[-2 0 0]<2)  (5<3[0 0 0]<2)    (5<3[2 0 0]<2)       $ 2
      (5<4[-1 -1 0]<3[0 -1 0]<2)  (5<4[0  -1 0]<3[0 -1 0]<2)  $ 2
      (5<4[ 1 -1 0]<3[0 -1 0]<2)  (5<4[-1  0 0]<3[0 -1 0]<2)  $ 2
      (5<4[ 0  0 0]<3[0 -1 0]<2)  (5<4[ 1  0 0]<3[0 -1 0]<2)  $ 2
      (5<4[-1  1 0]<3[0 -1 0]<2)  (5<4[ 0  1 0]<3[0 -1 0]<2)  $ 2
      (5<4[ 1  1 0]<3[0 -1 0]<2)                              $ 2
      (5<4[-1 -1 0]<3[-1 0 0]<2)  (5<4[0  -1 0]<3[-1 0 0]<2)  $ 2
      (5<4[ 1 -1 0]<3[-1 0 0]<2)  (5<4[-1  0 0]<3[-1 0 0]<2)  $ 2
      (5<4[ 0  0 0]<3[-1 0 0]<2)  (5<4[ 1  0 0]<3[-1 0 0]<2)  $ 2
      (5<4[-1  1 0]<3[-1 0 0]<2)  (5<4[ 0  1 0]<3[-1 0 0]<2)  $ 2
      (5<4[ 1  1 0]<3[-1 0 0]<2)                              $ 2
      (5<4[-1 -1 0]<3[1  0 0]<2)  (5<4[0  -1 0]<3[1  0 0]<2)  $ 2
      (5<4[ 1 -1 0]<3[1  0 0]<2)  (5<4[-1  0 0]<3[1  0 0]<2)  $ 2
      (5<4[ 0  0 0]<3[1  0 0]<2)  (5<4[ 1  0 0]<3[1  0 0]<2)  $ 2
      (5<4[-1  1 0]<3[1  0 0]<2)  (5<4[ 0  1 0]<3[1  0 0]<2)  $ 2
      (5<4[ 1  1 0]<3[1  0 0]<2)                              $ 2
      (5<7 8 9 10 11<1) (5<3[u=3]<2)  (5<4[u=3]<3[u=2]<2)     $ 3
      (5<7 8 9 10 11<1) (5<3[u=3]<2)  (5<4[u=3]<3[u=2]<2)     $ 4
      (5<7 8 9 10 11<1) (5<3[u=3]<2)(5<4[-1:1 -1:1 0]<3[u=2]<2) $ 5
sp5    1. 4r   1. 5r                  $ 1
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 1
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 1
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 1
      1. 4r   1. 5r                  $ 2
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 2
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 2
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 2
      1. 4r   1. 5r                  $ 3
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 3
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 3
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 3
      1. 4r   1. 5r                  $ 4
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 4
      .25 .5 .25  .5 1. .5   .25 .5 .25    .25 .5 .25  .5 1. .5   .25 .5 .25    $ 4
      1. 4r   1. 5r                  $ 5
      .25 .5 .25  .5 1. .5   .25 .5 .25   $ 5
      .25 .5 .25  .5 1. .5   .25 .5 .25    .25 .5 .25  .5 1. .5   .25 .5 .25    $ 5
nps     10000
c     volume of unit lattice cells, 5=15,707.963
c       6 full through cell 3,u3=304,292.0, 3,u2=64,292.037
c       6 full through cell 1=47,123.89
c f4:n   5  6  (5  6  3)                              $ 3 bins
c sd4    361283.16  2832876.11  7126461.27 $ first 2 are slightly less
c f14:n  (5<3) (5<(3[-2:2 -2:0 0]))                    $ 2 bins
c sd14   282743.34  1r                  $ slightly less
f4:n  (3<3[-2:2 -2:0 0:0]<2)   $ Silent Wrong Answers!
sd4    1.0
fm4    1.0
de4    1E-11  10
df4      1     1
f14:n (4<4[-1:1 -1:1 0:0])     $ Tally results match - spdtl off/force
sd14    1.0
fm14    1.0
de14    1E-11  10
df14     1     1
f24:n  (4<4[-1:1 -1:1 0:0]<3<3[-2:2 -2:0 0:0]<2) $ Silent Wrong Answers!
sd24    1.0
fm24    1.0
de24    1E-11  10
df24     1     1
c spdtl off
```