

Report: UCRL-ID-126455, Rev. 2
Code Release: UCRL-CODE-99034

TART98
A Coupled Neutron-Photon
3-D, Combinatorial Geometry
Time Dependent
Monte Carlo Transport Code

by
Dermott E. Cullen
University of California
Lawrence Livermore National Laboratory
P.O. Box 808
L-59
Livermore, CA 94550

tele: 925-423-7359
e. mail: cullen1@llnl.gov

November 22, 1998

Abstract

TART98 is a coupled neutron-photon, 3 Dimensional, combinatorial geometry, time dependent Monte Carlo radiation transport code. This code can run on **any modern computer**. It is a **complete system** to assist you with input preparation, running Monte Carlo calculations, and analysis of output results. TART98 is also **incredibly FAST**; if you have used similar codes, you will be amazed at how fast this code is compared to other similar codes. Use of the entire system can save you a great deal of time and energy.

TART98 is distributed on CD. This CD contains on-line documentation for all codes included in the system, the codes configured to run on a variety of computers, and many example problems that you can use to familiarize yourself with the system.

TART98 completely supersedes all older versions of TART, and it is strongly recommended that users only use the most recent version of TART98 and its data files.

NOTICE

This work was produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL. The rights of the Federal Government are reserved under Contract 48 subject to the restrictions agreed upon by the DOE and University as allowed under DOE Acquisition Letter 97-1.

DISCLAIMER

This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

NOTIFICATION OF COMMERCIAL USE

Commercialization of this product is prohibited without notifying the Department of Energy (DOE) or Lawrence Livermore National Laboratory (LLNL).

Acknowledgments

I thank the many users of earlier versions of TART who have supplied extremely useful feedback to me. Since the release of TART95, in July 1995, TART96, in November 1996, and TART97 in November 1997, the response from users in terms of feedback has been extremely useful in improving the code. These improvements have been in terms of correcting problems in the initial release of TART95, TART96 and TART97, and in terms of proposing new or improved options to meet the needs of users, now incorporated in TART98. I highly encourage all users to supply their feedback to me.

The TART98 System

This report is intended merely as a brief introduction to TART98. In particular no graphics results are presented in this report. The on-line documentation for the TART98 system codes, distributed on TART98 CD, has been coordinated to illustrate combined

use of the codes to make your job simpler and your work easier to accomplish, in particular extensive use of interactive graphics. If you have not used interactive graphics before you are only making your job harder and your tasks will take longer to accomplish.

Overview of This Report

This report describes all major changes in TART since TART95 [1], including changes in TART96 and TART97. As such this report supersedes the reports of TART96 [2] and TART97 [8]. However, the large TART95 report [1] is still the most comprehensive report on TART.

This report is divided into a number of parts, with each part describing one part of the TART98 CD system. The parts are,

Part 1: TART98 - Monte Carlo Calculations

Part 2: TARTCHEK - Check TART Input and Display TART Results

Part 3: TARTAID - Create TART Input

Part 4: EPICSHOW - Display Atomic and Nuclear data used by TART

Part 5: PLOTTAB - General Plotting Code to Display TART Output

Part 6: EDITOR - Text editor for use with TART

Part 7: Utility Codes - A collection of Useful Codes

I Strongly Recommend that you read the on-line documentation for all parts of this system, to get a better overall picture of how this entire code system fits together and can help you. The TART98 on-line documentation is in Microsoft Word 5.1 format and includes black and white as well as color graphic results. Only when you start using the codes in combination will you realize that this is a complete system that can really assist you in your work.

Computer Requirements

TART98 will run on any modern Computer, with at Least 8 Megabytes Memory and 30 Megabytes Disk Space. This puppy can run on virtually any computer; see, the below table of running times on a variety of computers.

TART98 CD

TART98 is distributed on CD. This CD contains on-line documentation for all codes included in the system, the codes configured to run on a variety of computers, and many example problems that you can use to familiarize yourself with the system.

TART Home Page

The TART home page is now located on the web at,

<http://reddog1.llnl.gov>

This site contains all of the TART documentation, as well as information and documentation related to TART and the nuclear and atomic data that it uses. This site is periodically updated, with newsletters, etc. If you are a TART user you should periodically check this site for the latest news.

TART Hot Line

Well, not exactly a hot line, but at least a place to turn to when you need help. If you have any difficulties setting up TART input, running it, or analyzing output, you can contact me at,

Telephone: 925-423-7359

E. Mail: cullen1@llnl.gov

Background

TART98 is a coupled neutron-photon, 3 Dimensional, combinatorial geometry, time dependent Monte Carlo transport code. The original **TARTND** has been used and distributed from Lawrence Livermore National Laboratory for many years. **TART95**, released in July 1995, was the first version of the code designed to be used on virtually any computer. **TART96** was designed to extend the general utility of the code to more areas of application, by concentrating on improving the physics used by the code. **TART97** further improved the physics, particularly with respect to newer neutron data, and more detailed neutron data. **TART98** further improved the physics, particularly with respect to newer photon data, and more detailed photon data. In addition **TART98** adds new input options, and greatly improved consistency checking designed to make the code more user friendly and to improve the reliability of results. **TART98 completely supersedes all older versions of TART, and it is strongly recommended that users only use the most recent version of TART98 and its data files.**

TART95

The objective of TART95 [1] was to develop a code that is as computer independent as possible. This objective was met by July 1995, when TART95 and its documentation were distributed for general use. At that time TART95 was operational on large mainframe computers, such as CRAY, and workstations, such as: SUN, SGI, HP, DEC Alpha, Meiko, and IBM RISC, as well as IBM-PC. Since that time it has become operational on additional types of computers, such as PowerMAC and even Laptop computers.

TART95 is written in such simple, computer independent FORTRAN, that it can now be easily implemented and used on virtually any computer.

TART96

Once the objectives of TART95 were met work began on TART96 [2]. The objective of TART96 was to extend the general utility of the code to more areas of application, by concentrating on improving the physics used by the code.

The most important improvements include,

NEW NEUTRON 650 GROUP TREATMENT: for cross sections over the energy range 10^{-4} eV up to 1 GeV. Older versions of the code used a 175 group treatment from $1.309 \cdot 10^{-3}$ eV up to 20 MeV, with most of the groups concentrated at higher energy; this limited accurate use of the code to higher energy applications. In contrast the new 650 group treatment is designed to accurately treat the entire neutron energy range, thereby allowing the code to be used for a wider range of applications. As yet neutron data is only generally available up to 20 MeV, but as soon as higher energy data becomes available TART96 is ready to use it. If you are a fan of the older 175 group treatment, not be worry: TART96 can use either 175 or 650 groups - the choice is yours.

ENDF/B-VI CROSS SECTIONS: Older versions of the code only used the Livermore ENDL library, which is primarily designed for use in high energy applications. In contrast the ENDF/B-VI data is designed for general use at all energies [3]. Therefore using this data allows the code to be accurately used in a wider range of applications. If you are a fan of the older ENDL data, not be worry: TART96 can use either ENDL or ENDF/B-VI - the choice is yours.

IMPROVED THERMAL SCATTERING TREATMENT: The major advantages of the new thermal scattering treatment include: improved accuracy of sampling, and greatly improved speed of execution [4].

FURTHER IMPROVEMENTS IN COMPUTER INDEPENDENCE: TART95 was implemented on a variety of computers, but it required the use of a few routines that varied from one computer to another. On the basis of user feedback, most of this remaining computer dependence has now been eliminated, and TART96 is now so computer independent that it is almost trivial to implement it on any new type of computer that comes along.

TART97

Once the objectives of TART96 were met work began on TART97 [8]. The objective of TART97 was to extend the general utility of the code to even more areas of application, by concentrating on improving the physics, particularly with regard to improved neutron data, and extending input options used by the code.

The most important improvements include,

NO UPPER LIMITS: on anything you can define by input. Unlike earlier versions of

TART, that had a maximum allowed number of zones, surfaces, etc., TART97 has no limits at all. If you want to use a million zones, with a different material composition in each zone, or anything else you can think of, TART97 can handle it.

NO LOWER LIMITS: on anything you can define by input. With earlier versions of TART, that used fixed maximum limits, there was a lot of overhead when running small problems. For example, when TART96 started it used 50 megabytes of memory. TART97 starts with about 1 megabyte and expands to need the needs of each problem. Typical problems only use 3-4 megabytes of memory. I haven't tried it, but TART97 is now so compact it will probably run on the new Palm Top computers.

NEW LONG RUN RANDOM NUMBER GENERATOR: The new generator includes over 2,500 different random number sequences, each sequence a trillion (10^{12}) random numbers displaced from the preceding sequence. With a modern computer we can generate a trillion random numbers in about one day, if that's all a code is doing. With TART97 each random number sequence should take about 10 to 20 days to use a complete trillion number sequence. Therefore the currently available 2,500 sequences should keep you busy for years; and if you need more, just ask for them.

MULTIPROCESSING: If you have a large computer with say 256 processors and a gigabyte of memory, you can do the arithmetic yourself: for a typical 3-4 megabyte problem, you can run 250 copies of TART97 all at the same time, and use the new utility code **TARTSUM** to add all of the results together. This approach is completely computer independent, and in this example you can compress 250 days of work into a single day (more than a year of working days into one day). With the new random number generator, using different random number sequences for each run, you can make over 2,500 statistically independent runs and combine the results. You can do this simply by running the same problem with different random number sequences, either using multiprocessing, or any number of single processor computers that you have access to, or a single processor repeatedly, if you just want to run more histories to improve your results.

NEW INPUT OPTIONS: have been added to extend TART97's capabilities, as well as to simplify and make input more user friendly. These options include: **cubic and quartic (e.g., torus) surfaces**, new **rotation** and **spatial translation**, **surface cloning**, new neutron and photon **sources**, see, **Appendix: Summary of New Conventions and Options**.

IMPROVED INPUT CHECKING: to catch more input errors before the calculation begins. As described below, this checking is now incorporated in both TART97 and TARTCHEK. **WARNING** it is highly recommended that you always use TARTCHEK to check your input, before actually running TART97.

IMPROVED ANALYTICAL VOLUME CALCULATOR: in many cases the results that we are interested in are not results per zone, but rather results per unit volume (e.g. per cc). The improved analytical volume calculator greatly extends TART's ability to quickly calculate zone volumes. This will meet the needs of most applications. For

extremely complicated geometric shapes TART97 includes a very fast Monte Carlo volume estimator. Used in combination, the analytical and Monte Carlo volume calculators can quickly define the volume of all of your zones, regardless of how complicated your geometry may be.

MODULAR CODING: TART97 is also very **modular**, so that portions of the code can be used in other codes. For example, TART97 and TARTCHEK use exactly the same input and geometry package; this assures that when you use TARTCHEK to check your geometry, when you run TART97 it will interpret your geometry in exactly the same manner. Similarly these packages will soon be incorporated into EPIC: an Electron Photon Interaction Code.

THE LATEST NEUTRON AND PHOTON DATA: TART97 uses the latest **ENDF/B-VI, Release 4**, neutron data [5], and Evaluated Photon Data Library '97 (**EPDL97**), photon data [6]. As with past versions of TART, if you would prefer to use older data, the option is yours.

INTERNAL CONSISTENCY CHECKING: No code is perfect, and for any complicated code, such as TART, that has many possible paths through it, it is virtually impossible to manually check all possible paths. With TART97's new internal consistency checking, it does its own checking every time it is run. For example, every single array in the code is checked for misuse, in an attempt to find as many errors as possible. This procedure has already led to accelerated code development and improvements, and will continue to do so in the future.

GENERAL IMPROVEMENTS: TART97 is based on the older TARTND code, but required massive changes to the code to make it the modern, computer independent code that it is today. As such there were bound to be some growing pains with this essentially new code. Over the last two years, feedback from the many code users has led to general improvements in the code, both in terms of locating and correcting problem areas, as well as in adding and improving code options to meet the needs of users.

FULL OPTIMIZATION: One general improvement worth noting, is that based on communications with a variety of FORTRAN compiler designers, TART97's has been re-designed to allow it to be compiled at the **highest level of optimization** on most computers, which can greatly reduce running time, without sacrificing accuracy.

TEMPERATURE DEPENDENT NEUTRON DATA: In the past TART has always used nominally room temperature (300 Kelvin) neutron cross sections. We can now prepare additional data files at virtually any temperature to meet programmatic needs [5].

TART98

Once the objectives of TART97 were met work began on TART98. The objective of TART98 was to extend the general utility of the code to even more areas of application, by concentrating on improving the physics, particularly with regard to improved photon

data, and extending input options used by the code. For details of the new photon treatment see, ref. [9].

It is worthwhile making one overall comment regarding the approach that I have used in implementing TART on every type of computer that I can get my hands on. Initially many people thought this couldn't be done. They were wrong. TART now runs on everything except my wristwatch (I'm working on that). In addition I'll mention that implementing TART on so many different types of computers has greatly improved the code. Each compiler has strengths and weaknesses, and by testing TART using as many different types of computers as possible has led to locating and eliminating as many different types of potential problems as possible. Isn't using all of these different types of computers time consuming and doesn't it delay development of a code like TART? Not at all. If anything, testing any code on as many different computers, using as many different compilers, as possible is probably the fastest way to test a code and improve its reliability.

The most important improvements include,

PHOTON 701 POINT TREATMENT: Incorporated in TART98 is a new photon 701 point treatment for cross sections over the energy range 100 eV up to 1 GeV. Older versions of the code used a 176 point treatment from 100 eV up to 30 MeV, with most of the points concentrated at higher energy; this limited accurate use of the code to higher energy applications. As with the 650 group neutron treatment, this new treatment of the photon cross sections is designed to accurately treat the entire energy range, allowing the code to be used for a wider range of applications.

PHOTON SCATTERING TREATMENT: TART98 includes an improved treatment of photon coherent and incoherent scattering. The new treatment has the advantage that it is both more accurate, and faster to use, than the older treatment.

MULTIPROCESSING: The TART utility codes MULTIPRO and TARTSUM are now routinely used to perform multiprocessing. This approach to multiprocessing is so simple, straightforward and general that it can be used by virtually all TART users. With this approach if you have a computer with thousands of processors you can use MULTIPRO to create everything that you need to use as many processors as you want and then average the results together using TARTSUM. Even if you don't have a computer with many processors, but you do have access to a number of computers, you can use all available computers to run problems (they don't even have to be the same type of computer), and again use TARTSUM to average all of the results together. This approach is completely computer independent, and in the example case of using 250 processors, you can compress 250 days of work into a single day (more than a year of working days into one day).

IMPROVED NO UPPER OR LOWER LIMITS: on anything you can define by input. Unlike earlier versions of TART, that had a maximum allowed number of zones, surfaces, etc., TART98 has no limits at all. For example, before TART97 the code was limited to a maximum of 1,000 spatial zones. Since then the spatial detail used in TART

problems has increased enormously. The largest TART98 problem that I know of involved 27 million (27,000,000) spatial zones. Of course TART98 can still accommodate even the simplest problem, such as a one zone spherical ball. In all cases from smallest to largest TART98 automatically sizes itself to accommodate each individual problem run.

NEW INPUT OPTIONS: have been added to TART98 to allow improved detail in photon tallies. In earlier versions of TART, photon output tallies were always limited to 50 energy bins. Starting with TART98 the user has the option to select 70 (the default), 175, or a full 700 energy bins for photon tallies and output, see, **Appendix: Summary of New Conventions and Options.**

IMPROVED INPUT CHECKING: to catch more input errors before the calculation begins. As described below, this checking is now incorporated in both TART98 and TARTCHEK. TART98 continues the TART traditions to support all older TART input. For example, if you have a twenty year old TART input problem, you will still be able to use it with TART98. However, TART98 and TARTCHEK are now much more clever at finding errors in TART input, with the result that you may find that TART input decks that ran earlier, will now cause TART to stop, with detailed ERROR messages asking you to correct your input before proceeding.

INTERNAL CONSISTENCY CHECKING: No code is perfect, and for any complicated code, such as TART, that has many possible paths through it, it is virtually impossible to manually check all possible paths. Starting with TART97 the code included new internal consistency checking; it did its own checking every time it ran. For example, every single array in the code is checked for misuse, in an attempt to find as many errors as possible. You wouldn't believe how effective this internal checking has been over the last two years at finding and allowing us to eliminate potential problems, resulting in a much more reliable code.

GENERAL IMPROVEMENTS: TART98 is based on the older TARTND code, but required massive changes to the code to make it the modern, computer independent code that it is today. As such there were bound to be some growing pains with this essentially new code. Over the last three years, feedback from the many code users has led to general improvements in the code, both in terms of locating and correcting problem areas, as well as in adding and improving code options to meet the needs of users.

Running Time

The below table presents results obtained using a collection of TART theoretical benchmark problems. All problems were run on each computer. This table summarizes timing results for the older TARTND code, that only runs on CRAY computers, as well as TART98 and TART95 on a variety of computers.

Code	Computer	Running Time	Ratio to TARTNP
------	----------	-----------------	--------------------

		(Seconds)	CRAY-YMP
TARTNP	CRAY-YMP	5396	1.0
TARTNP	CRAY-J90	7727	1.43
TART98	IBM-PC Pentium II/400	579	0.11
TART98	IBM-PC Pentium II/333	697	0.13
TART98	DEC-Alpha Model 5/300	712	0.13
TART98	IBM-PC Pentium II/266	855	0.16
TART98	IBM-PC Pentium Pro/200	1185	0.22
TART98	IBM-PC Lap Top/233	1301	0.24
TART98	Power-MAC 7500/275	1350	0.25
TART98	iMAC	1664	0.31
TART98	HP-735/125	1834	0.34
TART98	SUN E3000/166	2107	0.39
TART98	IBM-PC LapTop/133	2990	0.58
TART98	CRAY-YMP	4262	0.79
TART98	IBM-RISC RS-6000	5739	1.06
TART98	CRAY-J90	6095	1.13
TART98	Meiko CS-2/66	6225	1.15
TART98	SUN Sparc-20	6315	1.17
TART98	Power-MAC 7500/100	6446	1.21
TART98	SGI R4000/100	6953	1.29
TART95	CRAY-YMP	4912	0.91
TART95	HP-350	4322	0.80
TART95	DEC-Alpha	6130	1.14
TART95	SUN	9673	1.79
TART95	Meiko	9993	1.85
TART95	SGI	10157	1.88
TART95	IBM-RSIC	14838	2.75
TART95	IBM-PC 486DX2/66	18437	3.41

The TART95 results have been copied from the TART95 report.

When we compare the three codes all run on the same CRAY-YMP, we find that compared to the older TARTND code, TART95 was about 9 % faster, and TART98 is about 21 % faster. So that not only has TART98 been extended for more general uses, these extensions were accomplished with no lose in running time efficiency, i.e., TART98 is actually faster than TART95.

You should also note the advantage of TART95 and TART98 over the older TARTND in terms of their ability to be used on virtually any computer. For example, even a Laptop computer runs TART98 over four times as fast as TARTND on a CRAY-YMP, and on a basically \$ 3,000 IBM-PC Pentium-II, 400 MHz, TART98 runs about nine times faster than TARTND does on a multi-million dollar CRAY-YMP.

Why is Monte Carlo Used so much Today?

The last point to note from these comparisons is how far we have come in terms of available inexpensive computer power in the three years between the release of TART95 and TART98. When TART95 was released the fastest IBM-PC then available took 18,437 seconds to run this collection of theoretical problems. Even then we could foresee

the potential of an inexpensive computer being able to run these problems in only about 3.4 as much time as it took on a CRAY-YMP. But I don't think anyone could foresee that just three years later we now have available IBM-PCs that can run this collection of problems in only 579 seconds; almost nine times faster than a CRAY-YMP. Compared to the PCs of only three years ago, not only does today's PC run these problems almost 32 times faster, but it does it at about half the cost.

Think about what a difference in running time of a factor of 32 means. A major expense of any scientific project is your salary, so time is money and it can be expensive - or inexpensive, depending on how you spend it. Consider that only three years ago if it took an entire 9 to 5, 8 hour working day, to run a TART problem on an IBM-PC, today it would take less than 15 minutes to run the same problem, i.e., a factor of 32 faster. It should be noted, that this tremendous increase in available inexpensive computer power is one of the reasons that the use of Monte Carlo has expanded so much in recent years. Problems that we thought too time consuming to be practical just a few years ago, have now become routine.

Why is TART so FAST?

Some users make the mistake of assuming that since TART is so much faster than other codes that perform the same types of calculations, the results based on other codes must be better than those based on TART. When you use TART you will find that its results are just as accurate as those of other codes. So why is TART so fast?

There isn't any big secret to TART's speed: TART includes the three most important things necessary for generally efficient and accurate programming:

EXPERIENCE! EXPERIENCE! EXPERIENCE!

It is as simple as that. TART is based on over 30 years of continuous use and improvement. During this time roughly 80 work years of physicist/programmer time, and hundreds of work years of user experience, were incorporated into the code that we have today. To illustrate why TART is so much faster and still as accurate as other codes, I'll mention just a few points.

First is the use of multi-group data, including the multi-band method to account for self-shielding [1, 7], as used by TART, compared to continuous energy cross sections used to other codes. Results using continuous energy cross sections have to be better, right? Not always! This is only true if you run a calculation for extremely long times so that you accurately sample ALL of the continuous energy cross sections. This is almost never done, and I know of no code that explicitly includes an estimate of the uncertainty in its results based on the enormous variation in continuous energy cross sections. In comparison, TART's approach is designed for the real world, and incorporates not only the best nuclear and atomic data, but also the best nuclear and atomic engineering.

For example, if we look at the U-238 cross sections we see capture cross sections that vary by roughly four orders of magnitude, and we can see that it is composed of very

narrow resonances with relatively large energy intervals between resonances, i.e., the ratio of resonance spacing to width is about 100 to 1. This data is VERY DIFFICULT to sample on a continuous energy basis. Indeed if you try it you will find that in order to obtain even a fairly accurate estimate of the average cross sections and distance to collision you would have to sample billions of histories. I don't know of any code that uses continuous energy cross sections that actually does this. They simply supply you with the "best" possible cross sections and assume that this will solve your problems. TART takes it a step further: not only does TART use the "best" cross sections, but also uses the "best" nuclear engineering. Again, consider the U-238 cross sections. Anyone who has taken a course in reactor physics knows that in this case the neutron flux will self-shield and we know the form of the self-shielding. Therefore we do not need all of the nitty-gritty details of each and every narrow capture resonance in order to perform an accurate transport calculation. Think about it: people have been successfully designing nuclear reactors for over 50 years, and yet only fairly recently have detailed cross sections become available. So how did people design their reactors? They did what TART now does: combine the "best" currently available nuclear data with the "best" nuclear reactor theory. In the case of TART the use of the multi-band method to account for resonance self-shielding [1, 7] allows it to use multi-group, rather than continuous energy cross sections, resulting in rapid convergence of calculations, compared to code that use continuous energy cross sections and take forever to converge. Most important for users to understand is that this is done with virtually no lose in accuracy in the TART calculations, indeed it is fair to say that since for reasonable running times the TART results converge and those of other codes do not, from the pragmatic viewpoint of obtaining accurate answers in a reasonable amount of time, the TART results are better.

I should also mention the unresolved resonance region, where by definition we do not know the cross sections on a continuous energy basis, but it can be accurately treated by the multi-band method used by TART. Ask yourself: what do codes that claim to use continuous energy cross sections do in the unresolved resonance region?

A second example of why TART is so fast is its treatment of geometry. Compared to other codes TART uses a very strict geometry, which places an additional burden on the user in terms of input preparation. But the pay off is that the input is easier to check and correct (using TARTCHEK) to improve reliability, and when the code starts to run it FLYS!!!

For example, TART insists that the users define every space point to be within a spatial zone. Other codes do not insist on this, so why does TART? The first reason is that without insisting on this it is not possible to check the input parameters for errors; checking is now simple and straightforward using TARTCHEK, and greatly improves the reliability of the input. Next, when TART runs it greatly accelerates tracking. How can a few holes in the geometry make such a big difference? Consider a simple problem involving 1000 spatial zones with each zone bounded by 6 surfaces. When a particle enters a spatial region that is not defined in the problem, i.e., is a "hole", the code has to track (ray trace) to the nearest bounding surface to determine what zone it will next enter. In this example it has to ray trace to the 6 bounding surfaces of each of the 1,000 spatial zones, to determine which of these surfaces is closest to the particle in its direction of travel, i.e., it has to ray trace to 6,000 surfaces. In contrast, with TART where a particle is

always within a defined zone, in this example, we are inside one of the zones and we have to track (ray trace) to the nearest boundary of the zone. This only involving tracking to each of the bounding surfaces of this one zone, i.e., ray trace to 6, rather than 6,000 surfaces. No wonder TART geometry is so much faster to track through. How much of an effect does this really make? TART and TARTCHEK use exactly the same geometry package. In the original method used by TARTCHEK to display 3-D objects, TARTCHEK used a general ray tracing technique that did not take advantage of TART geometry. When TARTCHEK was updated to take advantage of TART geometry the ray tracing to display 3-D objects ran up to 200 TIMES FASTER - not 200 % - 200 TIMES!!! Pictures that took hours or all night to produce could suddenly be done in minutes or seconds. The difference was dramatic. You can see for yourself; use TARTCHEK to display 3-D views of your geometry and you will be amazed at how fast it can do it - and remember in doing it, it is using EXACTLY the same routines that TART uses to track through 3-D geometry. No wonder TART is so fast.

These are but a few examples of why TART runs so much faster than other codes, with essentially no lose in accuracy. Try it for yourself and see what you think.

As related to reliability, I'll also mention in passing the danger of using the default of other codes, that assume that whatever volume you have not explicitly defined is vacuum that the code can freely transport through. My experience has been that when a problem has an undefined volume in it, well over 90 % of the time it is because it is an error. Other codes sweep this under the rug and make it appear that nothing is wrong, usually resulting in the wrong answer. In contrast TARTCHEK and TART98 will quickly find these volumes and ask you to explicitly define them. This approach greatly improves the reliability of the TART input.

What Code should you be using?

TART98 completely supersedes all older versions of TART, and it is strongly recommended that users only use the most recent version of TART98 and its data files. How do you know if you have the most recent version of the code and its data files? As soon as the code starts to run it identifies the version you are running and the dates of its data files. Below is the beginning of the code output report. Note, the code version: **TART 98-5, Dec. '98**, and the date of three data files is **09/19/97**, and the fourth is **07/04/98**. Note, also the newer 566 groups for the neutron data and 701 points for the photon data. If you are using an older version of the code or its data files, it is strongly recommended that you obtain the most up-to-date code and data; see, the below section on **Availability**.

TART98 - Neutron-Photon Monte Carlo Transport (TART 98-5, Dec. '98)

```

I/O Files Opened for Entire Run
=====
Definition                               Filename      Unit  Date
=====
TART Input Parameters.....TART.IN          2
TART Output Listing.....TART.OUT           3
TART Input Scratch File.....TART0IN0.TMP      33
Neutron Interaction Data File.....TARTND         7  09/19/97
Photon Interaction Data File.....GAMDAT          8  07/04/98
Neutron Induced Photon Production File...TARTPPD         9  09/19/97
Multi-Band Parameter File.....NEWCROSS        10  09/19/97

Neutron Interaction Data.  566 Groups 1.0000D-10 to 2.0000D+01 MeV
Photon Interaction Data..  701 Points 1.0000D-04 to 1.0000D+03 MeV

```

Utility Codes

In addition to the TART98 code you should also be aware of the utility codes distributed with TART98; of particular note are **TARTCHEK** and **TARТАID**. One of the most difficult tasks that you will face in using any 3-D combinatorial Monte Carlo code is to correctly define input parameters for the code, particularly to correctly define geometry. This is what **TARTCHEK** is designed to help you with. It is an interactive graphics code that will allow you to view and check your input parameters before you run **TART98**. Even we so called "experts" on **TART** find that using **TARTCHEK** can greatly reduce the amount of time that we have to spend on input preparation, and even what is more important, greatly improve the reliability of our input parameters. In addition the TART98 CD system includes a new code: **TARТАID**, which will allow you to interactively create TART input decks from scratch.

TARTCHEK can also help you analyze results by overlaying flux or energy deposition on your geometry. Instead of spending days or weeks wading your way through a thick output listing trying to understand the results, using **TARTCHEK** a few minutes after

you finish a TART98 calculation you can “see” the results overlaid on your geometry. Not only will this save you time, it can improve your overall understanding of the results, by showing you the “big picture” of how flux, deposition, etc., in each zone is related to that in all other zones. This is something that is very difficult to “see” regardless of how long you stare at an output listing. If you are not using **TARTCHEK** you are only making your job more difficult, and you don't know what you are missing.

TARTAID is another code you should be aware of. In addition to **TARTCHEK**, which can be used to check existing TART input, and display TART results, the TART98 CD system also includes **TARTAID**. This code is designed to help you create TART input from scratch. It is particularly helping to define very detailed geometry, involving many spatial zones. For example to create a TART input deck involving 10,000 or even 100,000 spatial zones, takes only minutes using **TARTAID**.

You should also be away of the new utility codes **MULTIPRO** and **TARTSUM**, which will allow you to easily run many TART problems simultaneously, and then add together results from any number of TART problems, and produce a combined output file in **EXACTLY** the same format as any other single TART problem output file. Our computers are getting faster and faster, but we are running into the speed of light problem, where we can only get so much work done using a single processor. TART98's approach to multiprocessing allows us to avoid this limit, in the sense that we can now compress the work that used to take many days, into a single day. This is true on either multiprocessing computers or a group of single processors computers. Just run your problems on ANY computer(s), using as many processors as you have access to, and **TARTSUM** will combine the results for you. Note, since the combined output file produced by **TARTSUM** is in **EXACTLY** the same format as any other single TART problem output file, if you are one of the many TART users who have utility codes to further process TART output results - not to worry - your utility codes will work on the combined file, exactly the same way they work on the results of a single TART run.

Documentation

Although TART98, supersedes all earlier versions of TART, the most complete documentation for TART is still,

TART95: A Coupled Neutron-Photon Monte Carlo Transport Code, Lawrence Livermore National Laboratory, UCRL-MA-121319, July 4, 1995, by D. E. Cullen, A.L. Edwards and E.F. Plechaty

This document, as well as all other TART documentation, is now available on-line at the TART website,

<http://reddog1.llnl.gov>

Availability

At Livermore, for copies of the system, contact me. Outside of Livermore, contact your local computer code center - within the United States, the Radiation Safety Information Computational Center (RSICC), Oak Ridge National Laboratory (e. mail: jib@ornl.gov), outside of the United States, the OECD Nuclear Energy Agency/Data Bank (NEA/DB), Paris, France (e. mail: sartori@nea.fr).

Code Installation

The code is distributed with detailed instructions concerning installation and testing of the code. These instructions are periodically updated for distribution with the code, to insure that the instructions are as up-to-date as possible, and exactly correspond to the version of the code that you will be implementing and using. As such, installation instructions will not be included here.

References

- [1] "TART95: A Coupled Neutron-Photon Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-MA-121319, July 1995, by D.E. Cullen, A.L. Edwards and E.F. Plechaty
- [2] "TART96: A Coupled Neutron-Photon 3-D, Combinatorial Geometry Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-ID-126455, November, 1996, by D.E. Cullen.
- [3] "The 1996 ENDF/B Pre-Processing Codes," The International Atomic Energy Agency, Vienna, Austria, IAEA-NDS-39, Rev. 9, November 1996, by D.E. Cullen.
- [4] "THERMAL: A Routine Designed to Calculate Neutron Thermal Scattering," Lawrence Livermore National Laboratory, UCRL-ID-120560-Rev-1, Sept. 1995, by D.E. Cullen.
- [5] "A Temperature Dependent ENDF/B-VI, Release 4 Cross Section Library," Lawrence Livermore National Laboratory, UCRL-ID-127776, by D.E. Cullen.
- [6] "EPDL97: the Evaluated Photon Data Library, '97 Version," Lawrence Livermore National Laboratory, UCRL--50400, Vol. 6, Rev. 5, by D.E. Cullen.
- [7] "Nuclear Cross Section Preparation", by D.E. Cullen, Chapter 1, Volume I, "Handbook of Nuclear Reactions Calculations," editor Yigal Ronon, CRC Press, Boca Raton, Florida (1986).
- [8] "TART97: A Coupled Neutron-Photon 3-D, Combinatorial Geometry Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-ID-126455, Rev. 1, November, 1997, by D.E. Cullen.
- [9] "A Simple Model of Photon Transport", by D.E. Cullen, Nuclear Instrumentation and Methods in Physics Research B101 (1995) pp 499-510. The original extended form of this paper is now available on-line at the TART website, <http://reddog1.llnl.gov>

Appendix: Summary of New Conventions and Options

To help explain and illustrate the use of the new options the TART98 CD distribution includes example input decks. I encourage you to use TARTCHEK to look at these examples - particularly using 3-D views, so you can see them better.

Biggest Changes for TART98 vs. TART97

In terms of physics, the biggest change is in the treatment of photons. Compared to TART97, TART98 includes a much more detailed representation of photon cross sections, and a greatly improved treatment of coherent and incoherent scattering. TART98 also includes expanded tally and output options for photon results.

TARТАID is available for the first time with TART98 CD. This has already become a very popular code. I designed **TARТАID** as somewhat of a complement to **TARTCHEK**, to deal with the problem of preparing and checking TART input before it is used in actual TART calculations. What I didn't foresee is that if you are using **TARТАID** many of the classic errors that **TARTCHEK** checks for cannot occur, and rather than complementing **TARTCHEK**, **TARТАID** is somewhat replacing it.

The other important change is EXPERIENCE!!! Again, I cannot stress how important this is for any code. In the case of TART each successive version of the code includes the operating experience of the many people who are now using the code. With each passing version of TART reliability and accuracy are improved, mostly based on feedback from users - such as yourself.

Biggest Changes for TART97 vs. TART96

There is no limit on input parameters. You can have any number of surfaces, zones, bounding surfaces, materials, sources, e.g., you can have a million zones, with a different material in every zone, if you want to do burnup calculations.

This also means there is no lower limit. Earlier versions of TART were dimensioned to handle large problems. Because of this the code would start at about 50 MB and then decrease in size. This caused startup problems on smaller computers. TART97 starts at about 1 MB and increases to meet the needs of your specific problem; most problems will only use 3-4 MB.

Any input line can now be continued onto any number of continuation lines. With earlier versions of TART some input, particularly complicated sources, could not be continued from one line to another, which made input preparation difficult. You will find that being able to continue any input line, it is much easier to prepare input. Some of the following new options, such as cloning, rotation and spatial translation, were recommended by TART users, and are also designed to simplify preparation of TART input. If you have ideas to even further simplify input preparation, I'd love to hear them.

The new input Options

Cubic

xcubic nb x0 y0 z0 d c b a
ycubic nb x0 y0 z0 d c b a
zcubic nb x0 y0 z0 d c b a

a cubic, rotationally symmetric about an axis - the equations are,

$$\begin{aligned} \text{xcubic: } (y-y_0)^2 + (z-z_0)^2 &= R(x)^2 \\ &= a(x-x_0)^3 + b(x-x_0)^2 + c(x-x_0) + d \\ \text{ycubic: } (x-x_0)^2 + (z-z_0)^2 &= R(y)^2 \\ &= a(y-y_0)^3 + b(y-y_0)^2 + c(y-y_0) + d \\ \text{zcubic: } (x-x_0)^2 + (y-y_0)^2 &= R(z)^2 \\ &= a(z-z_0)^3 + b(z-z_0)^2 + c(z-z_0) + d \end{aligned}$$

nb - Surface Number
x0, y0, z0 - Center coordinates
d, c, b, a - Coordinates of the cubic

The radius along one axis is represented as a cubic. By defining zones using a cubic and planes perpendicular to the axis of the cubic, you can reproduce almost any surface, using different cubic parameters to apply along different intervals of the axis; exactly as we think in terms of performing cubic spline fits.

You can reproduce almost any surface depending on a, b, c and d - spheres, ellipses, cylinders, cones, parabola, hyperbola, plus more complicated shapes.

WARNING it is R^2 , NOT R, that is represented by a cubic.

Example problem: NEWCUBIC.IN

Torus

xtorus nb x0 y0 z0 a b c
ytorus nb x0 y0 z0 a b c
ztorus nb x0 y0 z0 a b c

a torus aligned with an axis - the equations are,

$$\begin{aligned} \text{xtorus: } [(x-x_0)/a]^2 + [(r-c)/b]^2 &= 1 \\ r^2 &= (y-y_0)^2 + (z-z_0)^2 \end{aligned}$$

$$\text{ytorus: } [(y-y_0)/a]^2 + [(r-c)/b]^2 = 1$$

$$r^2 = (x-x_0)^2 + (z-z_0)^2$$

$$\text{ztorus: } [(z-z_0)/a]^2 + [(r-c)/b]^2 = 1$$

$$r^2 = (x-x_0)^2 + (y-y_0)^2$$

nb - Surface Number
 x0, y0, z0 - Center coordinates
 a, b, c - Coordinates of the torus

If a = b, it is a circular torus, otherwise it is an elliptical torus.

Example problem: NEWTORUS.IN

Rotation about the X, Y or Z axis

xrotate ang is1 thru is2

xrotate ang is1 is2 is3.....

yrotate ang is1 thru is2

yrotate ang is1 is2 is3.....

zrotate ang is1 thru is2

zrotate ang is1 is2 is3.....

A rotation of surface(s) about an axis by a clockwise angle **ang** (degrees) looking up the axis. Rotation is about the ORIGIN - not the center of the surface. Note, this differs from **surfp** and **srotate** input, which can only be used to rotate **surfr** input about the center of the surface.

ang - angle of rotation in degrees
 is1 thru is2 - rotate surface numbers is1 thru is2
 is1 is2 is3... - rotate the listed surface numbers

Surfaces can be rotated one or more times, and the rotation is cumulative and order dependent.

Any linear or quadratic surface may be rotated. Cubic and torus MAY NOT be rotated (at least yet).

WARNING - these options are executed immediately when they are read from a TART input deck. Therefore all surfaces to be rotated MUST be defined before they can be rotated, and the order of rotations is important.

WARNING - for TARTCHEK users, the lower, left hand plot, is looking at the front of your geometry in the (z,x) plane, looking UP THE Y AXIS. The upper, left hand plot, is looking down at the top of your geometry in the (z,y) plane, looking DOWN THE X AXIS. The lower, right hand plot, is looking at the right hand side of your geometry in the (y,x) plane, looking DOWN THE Z AXIS. As a result, a clockwise rotation about the

y axis will appear clockwise in the lower, left hand plot. However, a clockwise rotation about the x axis will appear COUNTERCLOCKWISE in the upper, left hand plot, and a clockwise rotation about the z axis will appear COUNTERCLOCKWISE in the lower, right hand plot. This isn't an error - it is merely a result of your perspective when viewing TARTCHEK displays.

Example problems: NEWHEX.IN, NEWROT.IN (1 rotation) and NEWROT2.IN (2 rotations)

Translation of Spatial Coordinates

addxyz xadd yadd zadd is1 thru is2
addxyz xadd yadd zadd is1 is2 is3.....

Add (x,y,z) to the current center of surfaces.

xadd, yadd, zadd - add to the current (x0, y0, z0) center coordinates of surfaces
 is1 thru is2 - add to surface numbers is1 thru is2
 is1 is2 is3... - add to the listed surface numbers

Any surface may be translated, any number of times - and the results are cumulative.

This can be used to translate an entire object or objects to a new location, by translating all bounding surfaces by the same amount. It also simplifies input by allowing you to ignore the final position of a collection of surfaces, and input them as if they are at the origin - then later "add" their final center coordinates.

WARNING - these options are executed immediately when they are read from a TART input deck. Therefore all surfaces to be spatially translated MUST be defined before they can be spatially translated.

Try: Adding this to any of the example input

Cloning (Duplicating) Surfaces

clones ns is1 thru is2
clones ns is1 is2 is3.....

Clone (copy) a surface any number of times. Surface ns is copied to define surfaces is1 thru is2, or is1 is2 is3.....

ns - surface number to clone (MUST be defined)
 is1 thru is2 - make surface numbers is1 thru is2 identical to surface ns
 is1 is2 is3.. - make the list of surface numbers identical to surface ns

Limitations: surface number ns MUST be defined BEFORE it can be cloned (copied). The surface numbers is1 thru is2 or is1 is2 is3... MUST NOT be defined.

Any surface may be cloned, any number of times.

This option can be used to minimize input preparation when you have a number of identical surfaces that will finally be located at different locations. You can input a surface once, clone it, and then later translate and/or rotate the clones to their final locations.

WARNING - these options are executed immediately when they are read from a TART input deck. Therefore the surface to be cloned (ns) MUST be defined before it can be cloned.

Example Problems: NEWHEX.IN, NEWROT.IN and NEWROT2.IN

Reduced, Reflecting Geometry

xabove x0
yabove y0
zabove z0

xbelow x0
ybelow y0
zbellow z0

For users who only want to model 1/2, 1/4 or 1/8 of symmetric geometry, these options can be used to: 1) define additional x, y and/or z planes, 2) add these planes as boundaries of ALL zones, 3) add additional, reflecting zones on the "other" side of the planes.

x0, y0, z0 - a plane perpendicular to the axis is defined at one of these coordinates.

"above" means transport above this plane - the reflecting zone is below this plane.

"below" means transport below this plane - the reflecting zone is above this plane.

With earlier versions of TART in order to accomplish this you had to include the surface of the reflecting zone explicitly as a bounding surface of every zone. With this new input option this is automatically done for you.

For TARTCHEK users, to see the effect of inserting these planes, use the above/below options on the "Surface" page.

WARNING: These planes are inserted into the geometry AFTER ALL input has been read - they CANNOT be rotated or transformed in ANY way. It is suggested that as a reminder to yourself, you always locate these options at the end of your TART input deck after all other geometric input parameters have been defined.

Try: Adding this to any of the example input

New Sources

These sources can be used to sample sources from irregularly shaped zones. Unlike the other sources, these sources reject a sample if it is not inside a zone number in the range nz1 through nz2. These three new sources are for a sphere, cylinder, or rectangular box. For sampling select whichever of these shapes corresponds “best” to the shape of your actual zone numbers nz1 through nz2.

RESTRICTIONS

- 1) nz1 MUST be less than or equal to nz2.
- 2) If none of 10,000 consecutive samples from the defined volume is within zone numbers nz1 through nz2, it is assumed you made a mistake and the code will terminate. This prevents the code from going into an infinite loop of sampling and rejecting forever.

source19 nz1 thru nz2 ri ro [x0 y0 z0]
s19 nz1 thru nz2 ri ro [x0 y0 z0]
s19g nz1 thru nz2 ri ro [x0 y0 z0]

A spherical shell source of inner radius ri, and outer radius ro, centered at x0, y0, z0. Use source19 or s19 for neutrons, and s19g for photons.

nz1 - lowest zone number to sample from
 nz2 - highest zone number to sample from
 ri - inner radius of sphere
 ro - outer radius of sphere
 x0, y0, z0 - center of the sphere (optional, defaults to 0, 0, 0)

source20 nz1 thru nz2 z1 z2 ri r0 [x0 y0]
s20 nz1 thru nz2 z1 z2 ri r0 [x0 y0]
s20g nz1 thru nz2 z1 z2 ri r0 [x0 y0]

A cylindrical shell source, aligned with the z axis, extending along the z axis from z1 to z2, of inner radius ri, and outer radius ro, centered at x0, y0. Use source20 or s20 for neutrons, and s20g for photons.

nz1 - lowest zone number to sample from
 nz2 - highest zone number to sample from
 z1 - lower z limit of cylinder
 z2 - upper z limit of cylinder
 ri - inner radius of cylinder
 ro - outer radius of cylinder
 x0, y0 - center of the sphere (optional, defaults to 0, 0)

Note, for a cylinder aligned with an axis other than the z axis, use sentl 30 (neutrons) or sentl 43 (photons) to rotate the coordinates.

source21 **nz1 thru nz2 x1 x2 y1 y2 z1 z2**
s21 **nz1 thru nz2 x1 x2 y1 y2 z1 z2**
s21g **nz1 thru nz2 x1 x2 y1 y2 z1 z2**

A rectangular box in (x,y,z), extending in x from x1 to x2, in y from y1 to y2, and in z from z1 to z2. Use source21 or s21 for neutrons, and s21g for photons.

nz1 - lowest zone number to sample from
nz2 - highest zone number to sample from
x1 - lower x limit of box
x2 - upper x limit of box
y1 - lower y limit of box
y2 - upper y limit of box
z1 - lower z limit of box
z2 - upper z limit of box

There are no examples of these sources included here.

Changes in sentinels

Photon tally bin sentinel

sentl 20 (0)

This sentinel was not used in earlier version of TART.

Starting with TART98 this sentinel can be used to define the Photon tally bin sentinel. If 0, the default, 70 photon tally bins will be used. If 1, there will be 175 photon tally bins. If 2, there will be a full 700 photon tally bins. For neutrons, see **sentl 46**.

Do not limit the energy range of transport and scoring

sentl 8 and 9

These sentinels define the minimum neutron (sentl 8) and photon (sentl 9) energy below which particles cannot transport.

DO NOT use these, unless you really want to limit the minimum energy of neutrons and photons. TART will now use the minimum energy of the data read from the data files - which for neutrons differs for 175 and 650 groups.

sentl 13 and 14

These sentinels define the minimum neutron (sentl 13) and photon (sentl 14) energy below which particles cannot tally (contribute to output results).

Similar to sentl 8 and 9 above - DO NOT use these, unless you really want to limit the minimum editing energy of neutrons and photons.

sentl 15 and 16

These sentinels define the maximum neutron (sentl 13) and photon (sentl 14) energy above which particles cannot tally (contribute to output results).

Similar to sentl 8 and 9 above - DO NOT use these, unless you really want to limit the maximum editing energy of neutrons and photons. Note, soon TART will be extended to higher energies, so get used to not using these options now.

New random number sequence selection

sentl 12

The code now has 2,510 sequences, one trillion (10^{12}) samples apart. Input 0 (the default) to 2509 will use the selected sequence. Any other input is a fatal ERROR.

WARNING - this replaces the earlier definition of this sentinel, where the random number seed was entered; seen. TART95 documentation.

Highly Recommended Options

For compatibility with earlier versions of TART, by default the following options are turned off, unless the user specifies by input that they be turned on. It is Highly Recommended that you turn on ALL of the following options.

sentl 20

For neutron problems turn on resonance self-shielding. This can make problems run 20 to 30 % longer, but without accounting for self-shielding the results can be completely unreliable.

sentl 25

For photon problems turn on fluorescence. If no photons get down to low energies, this will have no effect on running time. However, if they do, this option is REQUIRED to obtain reliable answers.

sentl 39

For neutron problems turn on thermal scattering. If no neutrons get down to thermal energies, this will have no effect on running time. However, if they do, this option is REQUIRED to obtain reliable answers.