UCRL-ID-126455 Rev 2

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

D. E. Cullen

November 22, 1998

Lawrence Livermore National Laboratory

UCRL-ID-126455, Rev. 2

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.

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 of 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 additional 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 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 additional 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 the TART 68 fast critical assembly benchmark problems. All 68 fast criticality 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 (Seconds)	Ratio to TARTNP 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. Consider that since this $ 3,000 computer, has run 68 separate criticality problems in a total of 579 seconds, it means **on average each criticality problem is completed in less than 9 seconds!!! It boggles the mind.**

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 these 68 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 these 68 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, where 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 ding 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 **TARTAID**. 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: **TARTAID**, 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,

htp://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.

TARTAID is available for the first time with TART98 CD. This has already become a very popular code. I designed **TARTAID** 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 **TARTAID** many of the classic errors that **TARTCHEK** checks for cannot occur, and rather than complementing **TARTCHEK**, **TARTAID** 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,

xcubic: $(y-y0)^2 + (z-z0)^2 = R(x)^2$
$= a(x-x0)^3 + b(x-x0)^2 + c(x-x0) + d$
ycubic: $(x-x0)^2 + (z-z0)^2 = R(y)^2$
$= a(y-y0)^3 + b(y-y0)^2 + c(y-y0) + d$
zcubic: $(x-x0)^2 + (y-y0)^2 = R(z)^2$
$= a(z-z0)^3 + b(z-z0)^2 + c(z-z0) + d$

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,

xtorus: $[(x-x0)/a]^2 + [(r-c)/b]^2 = 1$
$r^2 = (y-y0)^2+(z-z0)^2$

ytorus: $[(y-y0)/a]^2 + [(r-c)/b]^2 = 1$
$r^2 = (x-x0)^2+(z-z0)^2$
ztorus: $[(z-z0)/a]^2 + [(r-c)/b]^2 = 1$
$r^2 = (x-x0)^2+(y-y0)^2$

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 size 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
zbelow 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.

TARTCHEK (Version 98-1) Update

The primary documentation for TARTCHEK is still Chapter 7 of the TART95 documentation. This is merely a brief update on new and improved features of TARTCHEK.

Distributed with the on-line documentation are four color Postscript figures, that will be discussed below. These figures are in standard Postscript format that can be printed on any color Postscript printer or viewed with any Postscript viewer, such as Ghostview. In order to fully understand the following discussion I suggest that you view the figures using a Postscript viewer as you read the following text. If you do not have a Postscript viewer you can download one from the Web for FREE!

Many TARTCHEK users are already aware of this code's ability to help you quickly verify input parameters for TART, particularly to verify geometry. **I STRONGLY RECOMMEND THAT YOU NEVER RUN TART WITHOUT FIRST USING TARTCHEK TO VERIFY YOUR INPUT.**

To verify TART input users have been using the Geometry page of TARTCHEK. What most users seem to still be unaware of is that TARTCHEK is designed to include many pages of options - Geometry is merely the first page of options that you see when the code starts. You can cycle through pages of options by using your mouse to click on the lowest option on each page. Here I'll briefly describe the next two pages of options.

Surfaces - the second page of options allows you to see your geometry in 3-D. New options allow you to make all low density zones invisible, shade objects, and slice them open so you can see inside of them. Best of all the new TARTCHEK ray tracing method runs about 200 times faster than the original method - that right - not 200 % faster - 200 times faster. Plots that used to take hours now take seconds. You will find that in just a few seconds or minutes you can rotate your geometry to look at it from many perspectives; by generating a series of plots you can walk around your geometry. In additional, as with any page of TARTCHEK options, you can obtain color Postscript output of anything that you see on your screen. The first two Postscript figures illustrates use of these features.

The first figure **(COG.ps)** is a 3-D view of the geometry for the first example problem from the COG manual. We have a spherical photon source to the lower left. There is a lead filter in the middle (here for convenience modeled as a cylinder; in the COG problem it is a rectangle). Finally there is a cylindrical detector to the upper right. Later we will return to this problem when I discuss overlaying your results on your geometry.

The next figure **(PULSED.ps)** shows the details of the neutron generator and spherical shell used in a Livermore pulsed sphere measurements. In this case I have used the new **SLICE** options to slice the geometry open, so that we can see inside of it. This is a very powerful option to let you examine your geometry in detail.

Flux Edit - The next page of options allows you to overlay the results of TART calculations directly onto your geometry; you can overlay energy deposition or flux. To use this option, after you have run a TART source problem (neutron or photon source) you can run the utility code **FLUXEDIT** that will read your TART output file TART.OUT and prepare results for use by TARTCHEK in the file **FLUXEDIT.OUT**. When you then run TARTCHEK if you use the same TART input TART.IN and FLUXEDIT output file FLUXEDIT.OUT (put them both in your TARTCHEK directory) you can then immediately see what your results look like. Analysis that used to take weeks or months can now be done in a few minutes. The third and fourth figures illustrate use of this feature.

With the third figure **(DEPOSIT.ps)** we return of the first COG example problem, discussed above. For this example I have used a photon source directed at the detector and having a 20 degree angular spread. Here I illustrate energy deposition. With the absolute scale at the lower right of the figure we can read directly off the plot how the deposition is varying. Note the high deposition in the lead filter (as we would expect in a high Z material), and the lower deposition in the detector. We can also see the spatial variation of deposition in the detector. With figures similar to this it is really very easy to design experimental set-ups. For example, you can quickly check to see if this is really the thickness of lead filter you should use to achieve a given response in the detector. You can also check on other features that might not otherwise be obvious to you. For example, note the relatively high deposition on the side of the lead filter closest to the detector and outside of the photon source. This indicates a lot of backscatter from the detector, which might lead you to re-position the lead filter relative to the detector - something you might never have noticed when reading a long output listing - here you can see it minutes after you have run the TART calculation.

The next two figures show the energy deposition in a water phantom due to 1 MeV **(PHANTOM1.ps)** and 10 MeV **(PHANTOM2.ps)** photons incident on the phantom from the left with a 1 degree angular spread. Here the geometry is modeled as a cylinder 30 cm high and 15 cm in radius. Recently a colleague asked me what I expected the difference to be in the energy deposition in a water phantom for a 1 MeV versus 10 MeV photon. Rather than try to explain I quickly used TARTAID to model his geometry using 100 by 100 zones in R by Z geometry; 10,000 zones. I then ran TART followed by FLUXEDIT, and minutes later we were looking at the results. By comparing results for 1 and 10 MeV photons the difference in the results were immediately obvious. Finally to understand why the results are different we used **EPICSHOW** to look at the oxygen photon cross sections; see the **EPICSHOW** on-line documentation. Using EPICSHOW to display the oxygen photon cross sections we could see that the total cross section at 1 MeV is considerably larger than at 10 MeV, so we expect more of the photons to collide and deposit their energy at 1 MeV, compared to 10 MeV. That's part of the picture; the rest is that the lower energy photons will scatter through larger angles and spread out more from the incident photon beam. The whole process, TARTAID, TART, TARTCHEK and EPICSHOW took only about 20 minutes = problem solved!

The bottom line is that by using the TART system codes in combination you can do more than just generate numbers; you can quickly improve your understanding of exactly what's happening in your problems, and more importantly you can improve your understanding of WHY it is happening. When I say quickly I mean QUICKLY! When you use this code system you will be amazed at how quickly you can accomplish results.

Recent Addition - Comparing Photons and Neutrons

The above comparison of 1 and 10 MeV photons incident on water has become very popular with readers, and they have asked for additional examples. For example, several readers have asked for results of 1 or 10 MeV neutrons incident on water. The next two figures show the energy deposition in a water phantom due to 1 MeV **(PHANTOM3.ps)** and 10 MeV **(PHANTOM4.ps)** neutrons incident on the phantom from the left with a 1 degree angular spread (exactly the same situation as in the case of the above photons).

From the TART output report we can see that for water at a density of 1 gram/cc the mean free path for photons is, 1 MeV: 14.15 cm, 10 MeV: 46.1 cm, and for neutrons it is, 1 MeV: 2.1 cm, 10 MeV: 9.4 cm. From the TART output report we can also determine that the expected energy loss per cm for photons is, 1 MeV: 0.031 MeV/cm, 10MeV: 0.17 MeV/cm, and for neutrons it is, 1 MeV: 0.176 MeV/cm, 10 MeV: 0.36 MeV/cm.

By comparing the photon and neutron results we can see that of the two the neutron cross sections are much higher, so that fewer neutrons are transmitted through the 30 cm thickness of water, e.g., with a mean free path of only 2.1 cm there is only a very small probability that any 1 MeV neutrons will be transmitted through the water. In addition the energy deposition (MeV/cm) is higher in the case of neutrons, so not only do the neutrons have more collisions per cm, they also lose more energy per cm of travel. We can see from the photon results that generally photons scatter through relatively small angles, so they more or less go where you point them. In contrast we can see from the neutron results that neutrons scatter through larger angles and can be very invasive, spreading their energy deposition over a relatively large area.

Recent Addition - Comparing Low and High atomic Number Materials

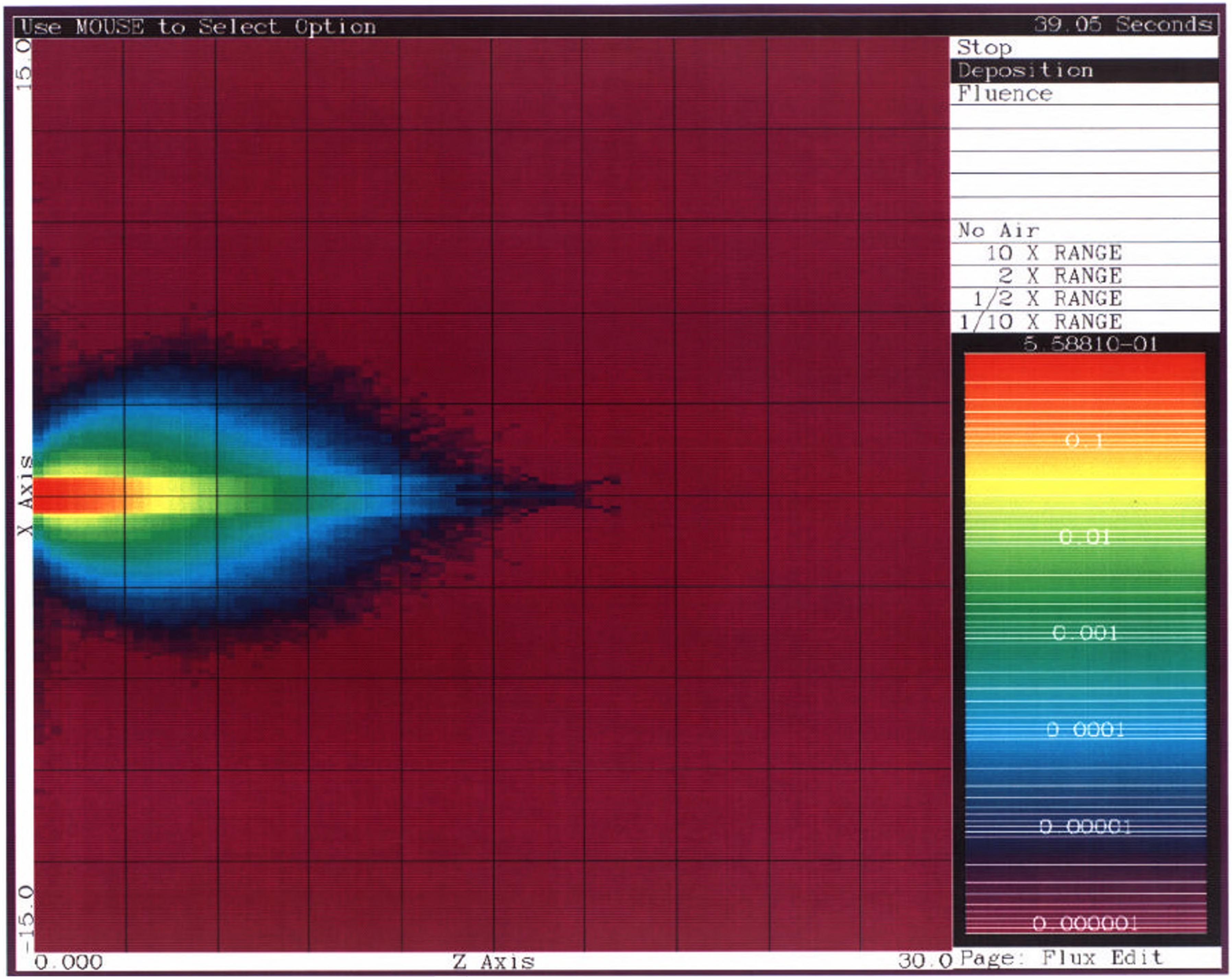
All of the above results are for water, here defined as simply two atoms of hydrogen for each atom of oxygen (H2O), normalized to an overall density of 1 gram/cc. Therefore in this case we have seen results for low atomic number materials: hydrogen, $Z = 1$, and oxygen, $Z = 8$. You can learn a lot by doing the same calculations for a high atomic number material, and comparing the results.

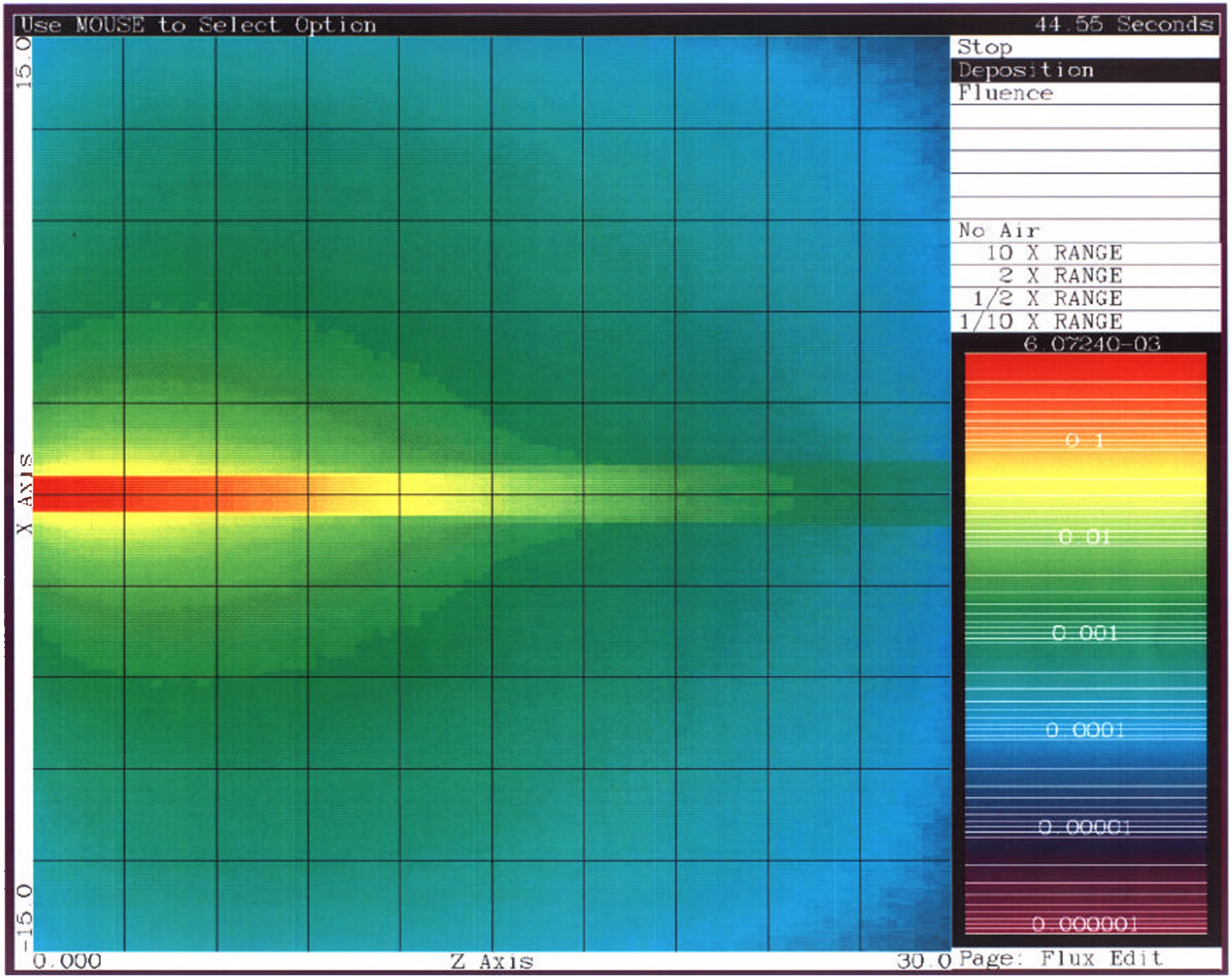
The following results using exactly the same geometry that we used in the above examples, namely a cylinder 15 cm in radius and 30 cm thick, and the same source distributions, namely, 1 or 10 MeV, photons or neutrons, with a one degree angular spread. The only difference will be that in this case we will use lead (Pb) at 11.35 grams/cc, instead of water at 1 gram/cc; this will show us results for a high atomic number material, $Z = 82$, that we can compare to the water results.

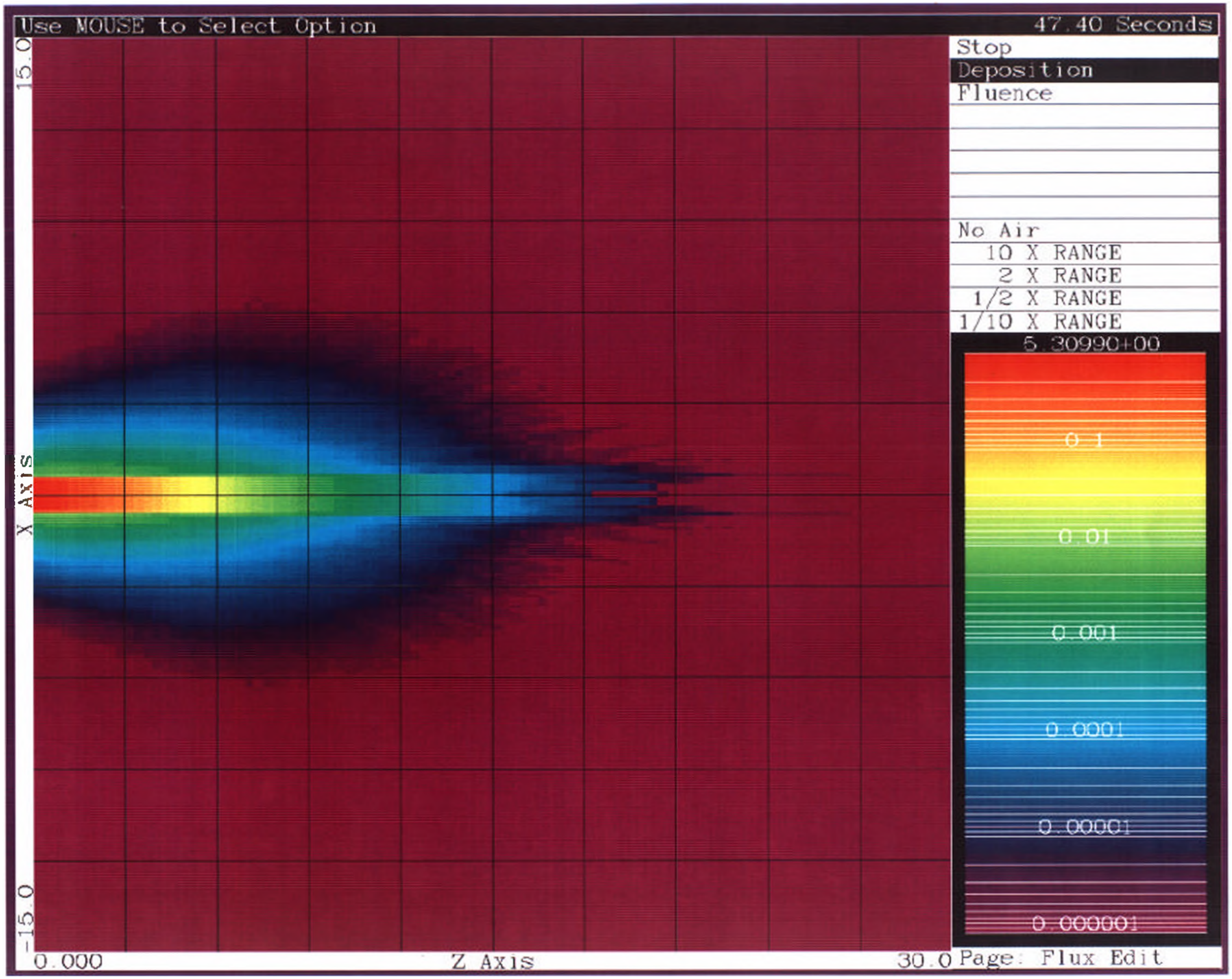
The next two figures are for 1 **(PHANTOM5.ps)** and 10 **(PHANTOM6.ps)** MeV **photons** incident. For a 1 MeV photon the mean free path is only 1.25 cm, and the expected energy loss is 0.44 MeV/cm, and for a 10 MeV photon the mean free path is 1.75 cm, and the expected energy loss is 5.1 MeV/cm. We can see from the figures that in this case the mean free paths are so small that virtually nothing gets through the 30 cm thickness of lead; everything is deposited in a fairly narrow volume about the original direction of the source. By comparing the water and lead results we can see that the photons cross sections increase rapidly with the atomic number (Z) of the target, so that high atomic number materials are very effective at stopping photons.

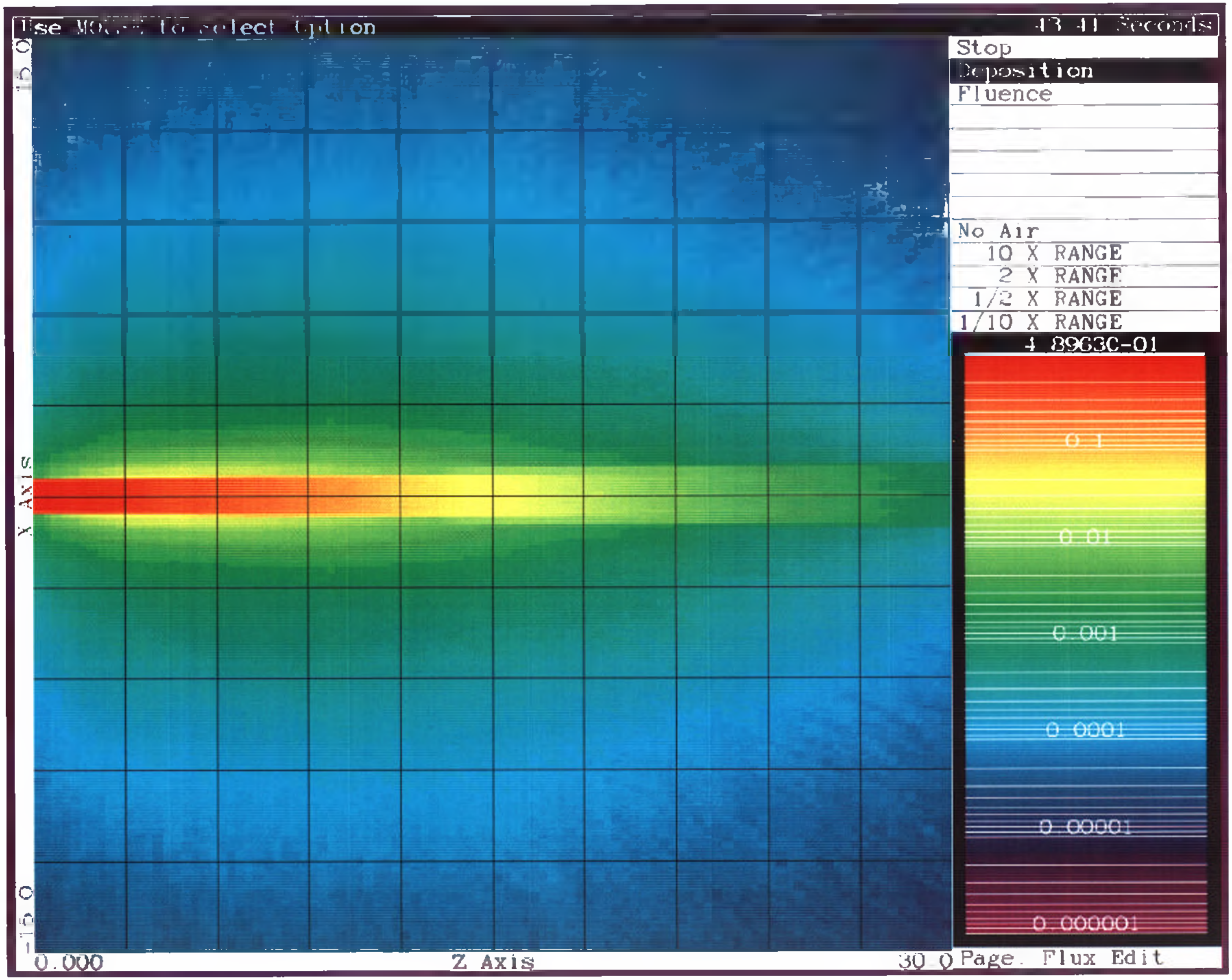
The next two figures are for 1 **(PHANTOM7.ps)** and 10 **(PHANTOM8.ps)** MeV **neutrons** incident. For a 1 MeV neutron the mean free path is about 5 cm, and the expected energy loss is only about 0.006 MeV/cm, and for a 10 MeV neutron the mean free path is also about 5 cm, and the expected energy loss is about 0.45 MeV/cm. By comparing the water and lead results we can see that they are fairly similar, indeed for 1 MeV neutrons it is easier for them to penetrate the lead than the water. This might seem surprising to you since for a thickness of 30 cm, we are comparing 30 grams of water to 340.5 grams of lead. But what these results illustrate is that the low atomic weight materials in water, particularly the hydrogen (Z=1) is very effective at moderating neutrons and reducing their energy to a sufficiently low value that they can be absorbed by the hydrogen. In contrast the high atomic weight of lead means that it is not a very good neutron moderator, so that neutrons will scatter around in the lead and spread out a lot before they are finally absorbed.

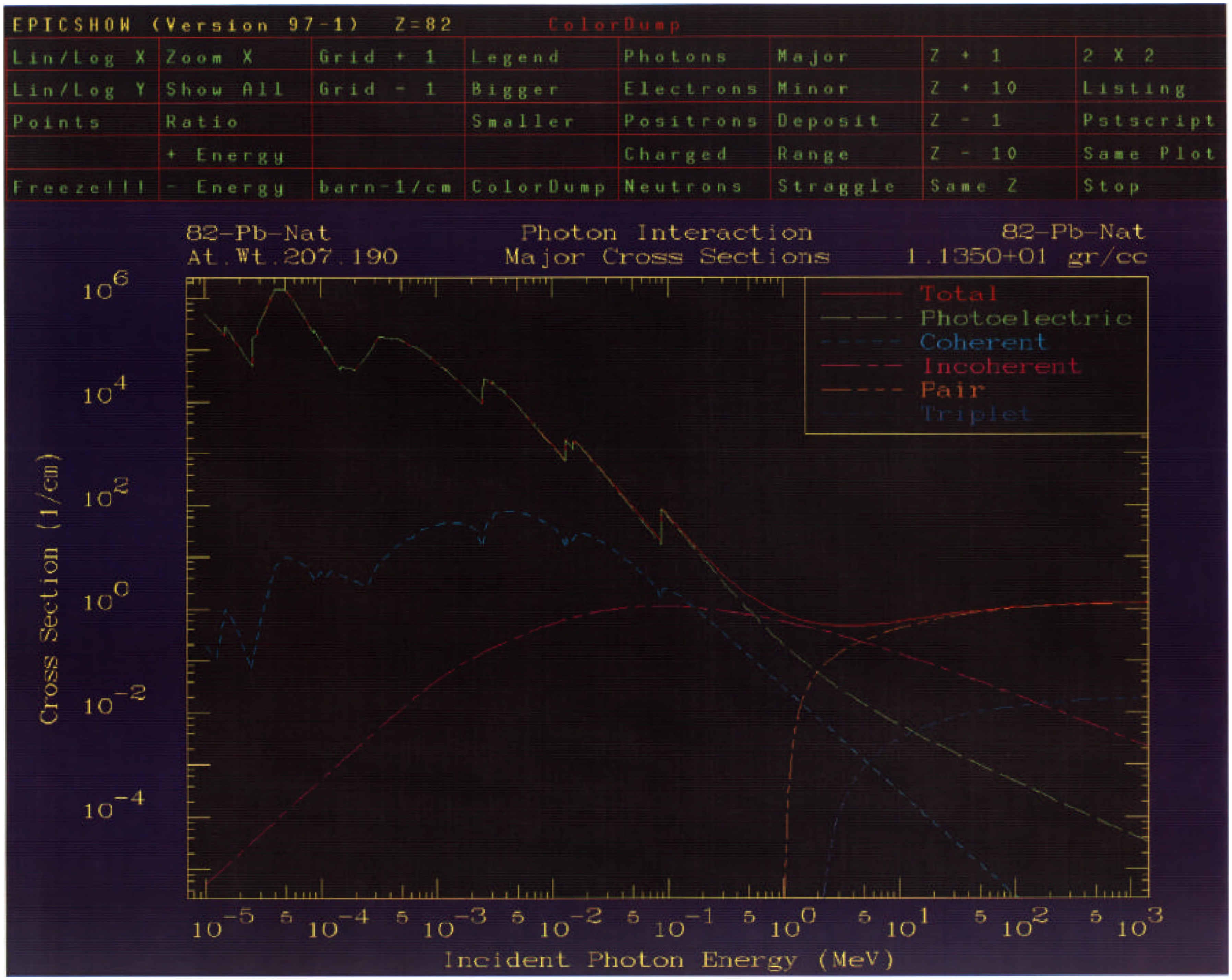
To further improve your understanding of the lead results you can also use EPICSHOW to see the lead photon cross sections (**PbPxc.ps**) and expected energy deposition (**PbPdep.ps**) as well as the lead neutron cross sections (**PbNxc.ps**) and expected energy deposition (**PbNdep.ps**). Note, for example, the relatively small lead neutron absorption cross section, which allows neutrons to transport very well through lead. Only after neutrons have scattered to lower energies can they be effectively absorbed.

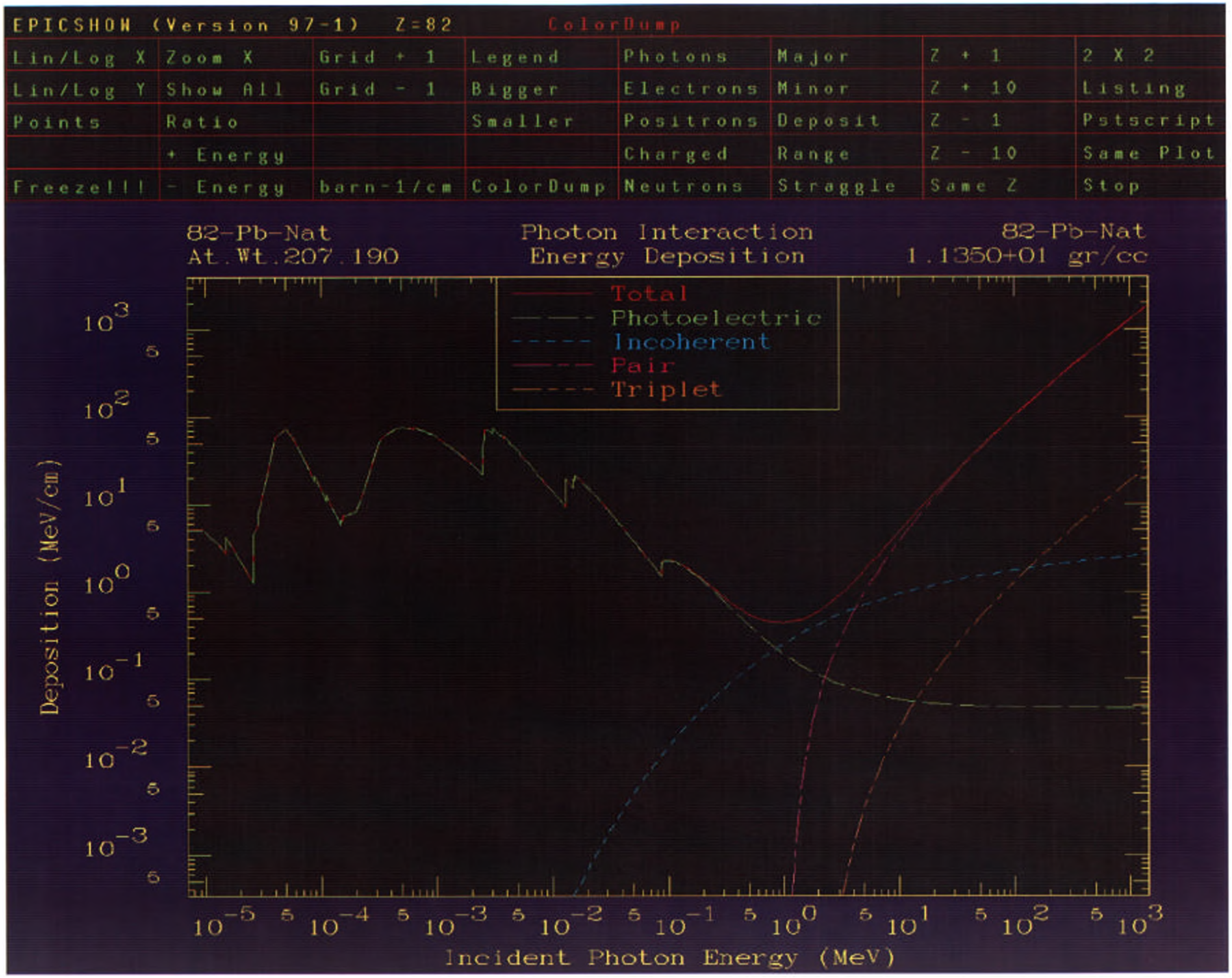

Use MOUSE to Select Option
39.05 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
5.58810-01
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X Axis
-15.0
0.000
Z Axis
30.0
Page: Flux Edit

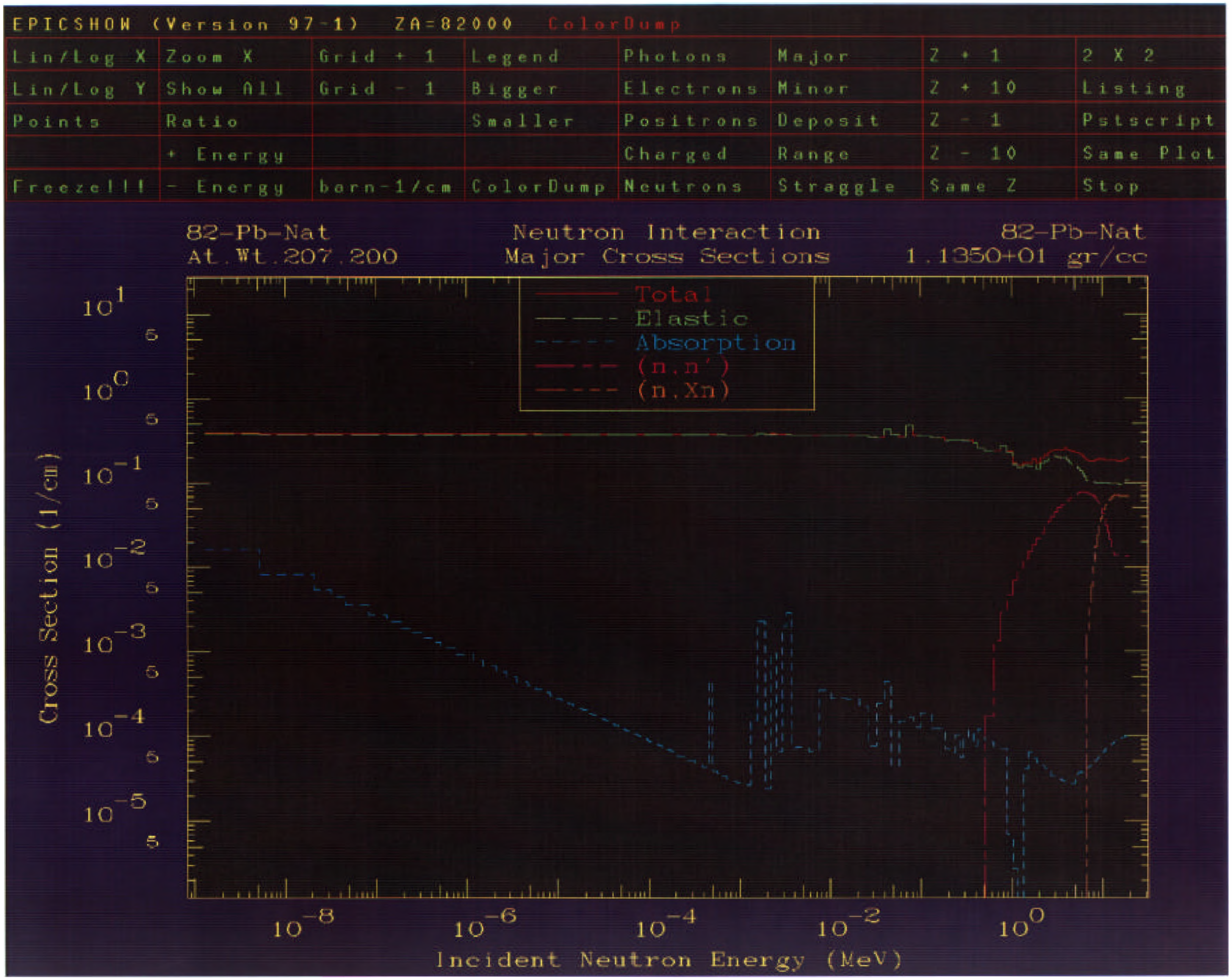

Use MOUSE to Select Option
44.55 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
6.07240-03
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X AXIS
-15.0
0.000
Z Axis
30.0
Page: Flux Edit

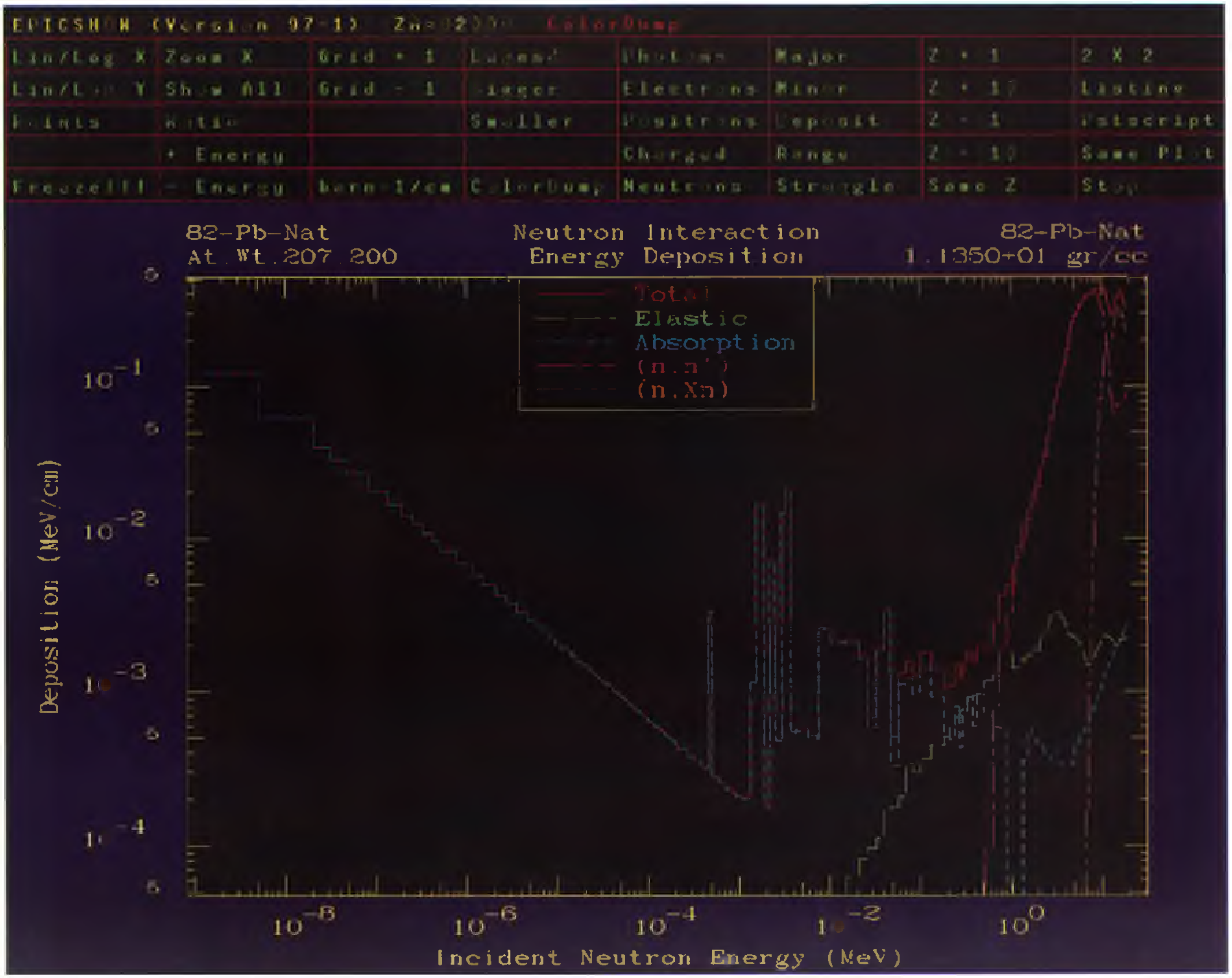

Use MOUSE to Select Option
47.40 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
5.30990+00
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X Axis
-15.0
0.000
Z Axis
30.0
Page: Flux Edit


Use MOUSE to select Option
43.41 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
4.89630-01
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X AXIS
-15.0
0.000
Z Axis
30.0
Page. Flux Edit


EPICSHOW (Version 97-1) Z=82
ColorDump
Lin/Log X
Zoom X
Grid + 1
Legend
Photons
Major
Z + 1
2 X 2
Lin/Log Y
Show All
Grid - 1
Bigger
Electrons
Minor
Z + 10
Listing
Points
Ratio
Smaller
Positrons
Deposit
Z - 1
Pstscript
+ Energy
Charged
Range
Z - 10
Same Plot
Freeze!!!
- Energy
barn-1/cm
ColorDump
Neutrons
Straggle
Same Z
Stop
82-Pb-Nat
At.Wt.207.190
Photon Interaction
Major Cross Sections
82-Pb-Nat
1.1350+01 gr/cc
Total
Photoelectric
Coherent
Incoherent
Pair
Triplet
Cross Section (1/cm)
10^{6}
10^{4}
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-5} 5 10^{-4} 5 10^{-3} 5 10^{-2} 5 10^{-1} 5 10^{0} 5 10^{1} 5 10^{2} 5 10^{3}
Incident Photon Energy (MeV)


EPICSHOW (Version 97-1) Z=82
ColorDump
Lin/Log X
Zoom X
Grid + 1
Legend
Photons
Major
Z + 1
2 X 2
Lin/Log Y
Show All
Grid - 1
Bigger
Electrons
Minor
Z + 10
Listing
Points
Ratio
Smaller
Positrons
Deposit
Z - 1
Pstscript
+ Energy
Charged
Range
Z - 10
Same Plot
Freeze!!!
- Energy
barn-1/cm
ColorDump
Neutrons
Straggle
Same Z
Stop
82-Pb-Nat
At.Wt.207.190
Photon Interaction
Energy Deposition
82-Pb-Nat
1.1350+01 gr/cc
Total
Photoelectric
Incoherent
Pair
Triplet
Deposition (MeV/cm)
Incident Photon Energy (MeV)


EPICSHOW (Version 97-1) ZA=82000 ColorDump
Lin/Log X
Zoom X
Grid + 1
Legend
Photons
Major
Z + 1
2 X 2
Lin/Log Y
Show All
Grid - 1
Bigger
Electrons
Minor
Z + 10
Listing
Points
Ratio
Smaller
Positrons
Deposit
Z - 1
Pstscript
+ Energy
Charged
Range
Z - 10
Same Plot
Freeze!!!
- Energy
barn-1/cm
ColorDump
Neutrons
Straggle
Same Z
Stop
82-Pb-Nat
At.Wt.207.200
Neutron Interaction
Major Cross Sections
82-Pb-Nat
1.1350+01 gr/cc
Total
Elastic
Absorption
(n,n')
(n,Xn)
Cross Section (1/cm)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
5
10^{-8}
10^{-6}
10^{-4}
10^{-2}
10^{0}
Incident Neutron Energy (MeV)


EPICSHOW (Version 97-1) Z=82000 ColorDump
Lin/Log X
Zoom X
Grid + 1
Legend
Photons
Major
Z + 1
2 X 2
Lin/Log Y
Show All
Grid - 1
Bigger
Electrons
Minor
Z + 10
Listing
Points
Ratio
Smaller
Positrons
Deposit
Z - 1
Postscript
+ Energy
Charged
Range
Z - 10
Same Plot
Freeze
- Energy
barn-1/cm
ColorDump
Neutrons
Straggle
Same Z
Stop
82-Pb-Nat
At.Wt.207.200
Neutron Interaction
Energy Deposition
82-Pb-Nat
1.1350+01 gr/cc
Total
Elastic
Absorption
(n,n')
(n,Xn)
Deposition (MeV/cm)
10^{-1}
10^{-2}
10^{-3}
10^{-4}
5
10^{-8}
10^{-6}
10^{-4}
10^{-2}
10^{0}
Incident Neutron Energy (MeV)

TARTAID (Version 98-1)

TART98 CD is the first version of the TART system that is distributed with **TARTAID**. This code is designed to allow you to quickly prepare TART input decks starting from scratch. Its most effective use is to define simple objects, in great detail using many spatial zones. You can then combine these simple objects into more complicated geometry.

For compatibility with earlier versions of TART and all existing older TART input decks, TART MUST use the same default parameters that it has always used. Today these default parameters are not necessarily the best choices to use, but to maintain compatibility that's what we are stuck with. **TARTAID** has no such limitations, and its default values are designed to be the best choices today.

In summary, **TARTAID** helps you with two of the most difficult problems facing anyone who is preparing TART input,
1) Accurately defining geometry.
2) Selecting the Best options to meet today's needs.

TARTAID is an interactive graphics code that uses a screen layout that is very similar to **TARTCHEK**. So if you are familiar with **TARTCHEK**, you will find it very easy to learn TARTAID, and to use it to good advantage.

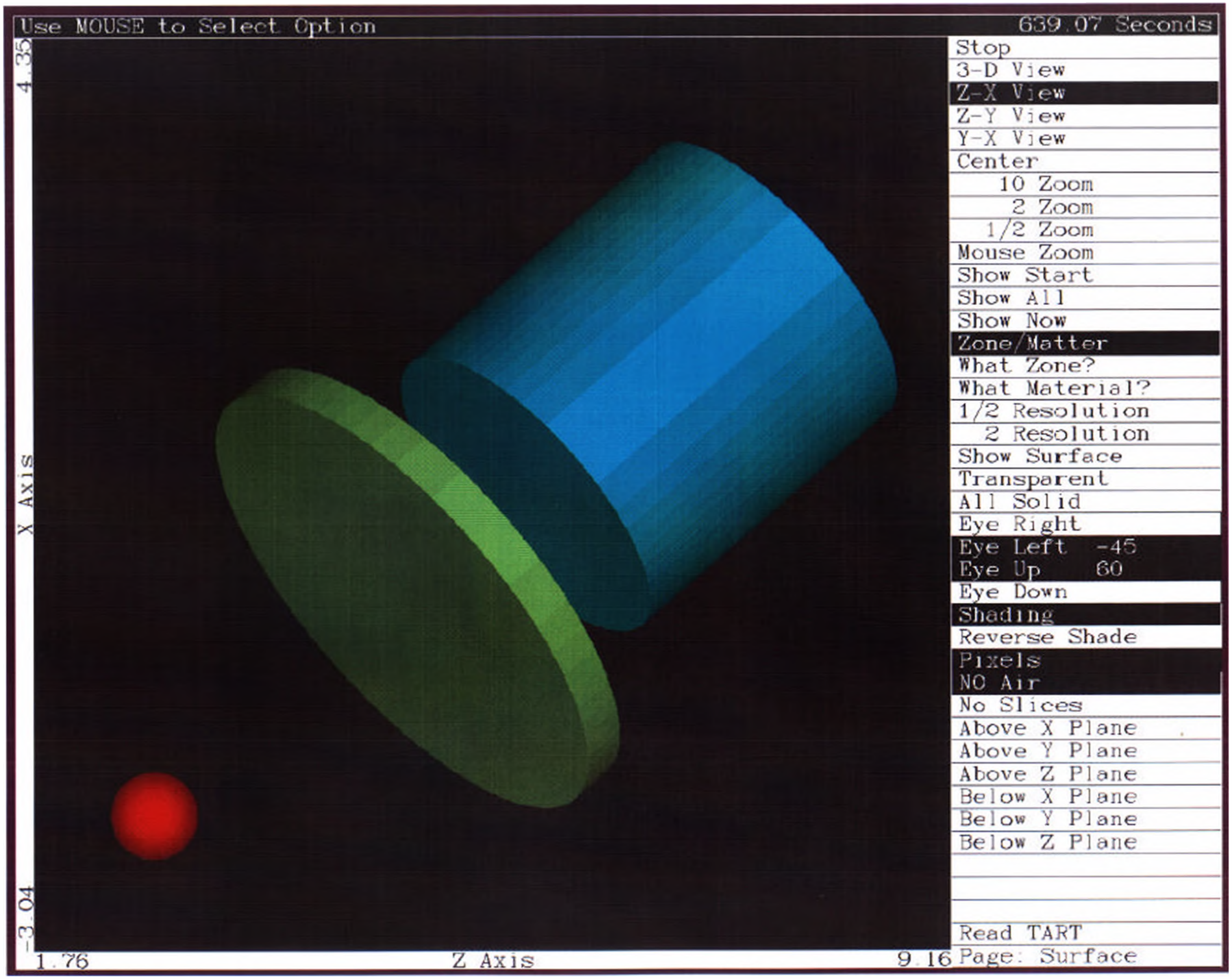
Overview

Creating a TART Input Deck requires the following steps,
1) Define General Running Conditions.
2) Define ALL Materials for your Problem.
3) Define your Spatial Zones.
4) Assign a Material to EACH Zone.
5) Define your Source.
6) Define Weights.
7) Define Tally and Output Options.

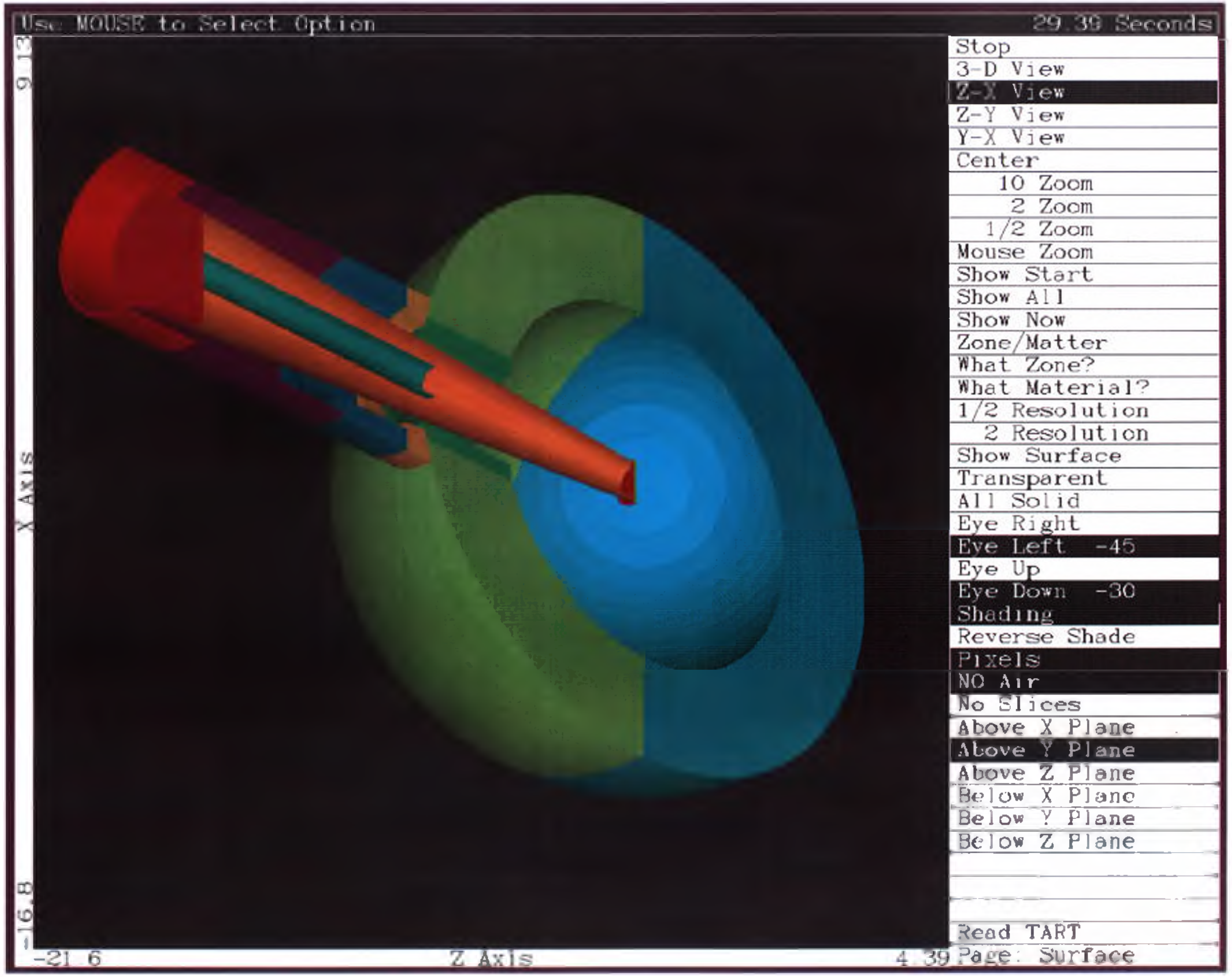
The presently distributed version of **TARTAID** ONLY does steps 1) through 4). Currently you have to do steps 5) through 7) manually. Fortunately, for most TART problems steps 1) through 4) involves the bulk of the work required to prepare TART input, and is by far the most error prone part of input preparation. So you can use **TARTAID** to quickly get you through these steps, and it is then usually much easier to deal with the remaining input preparation, see, the below example.

TARTAID Creates a Complete TART Input Deck

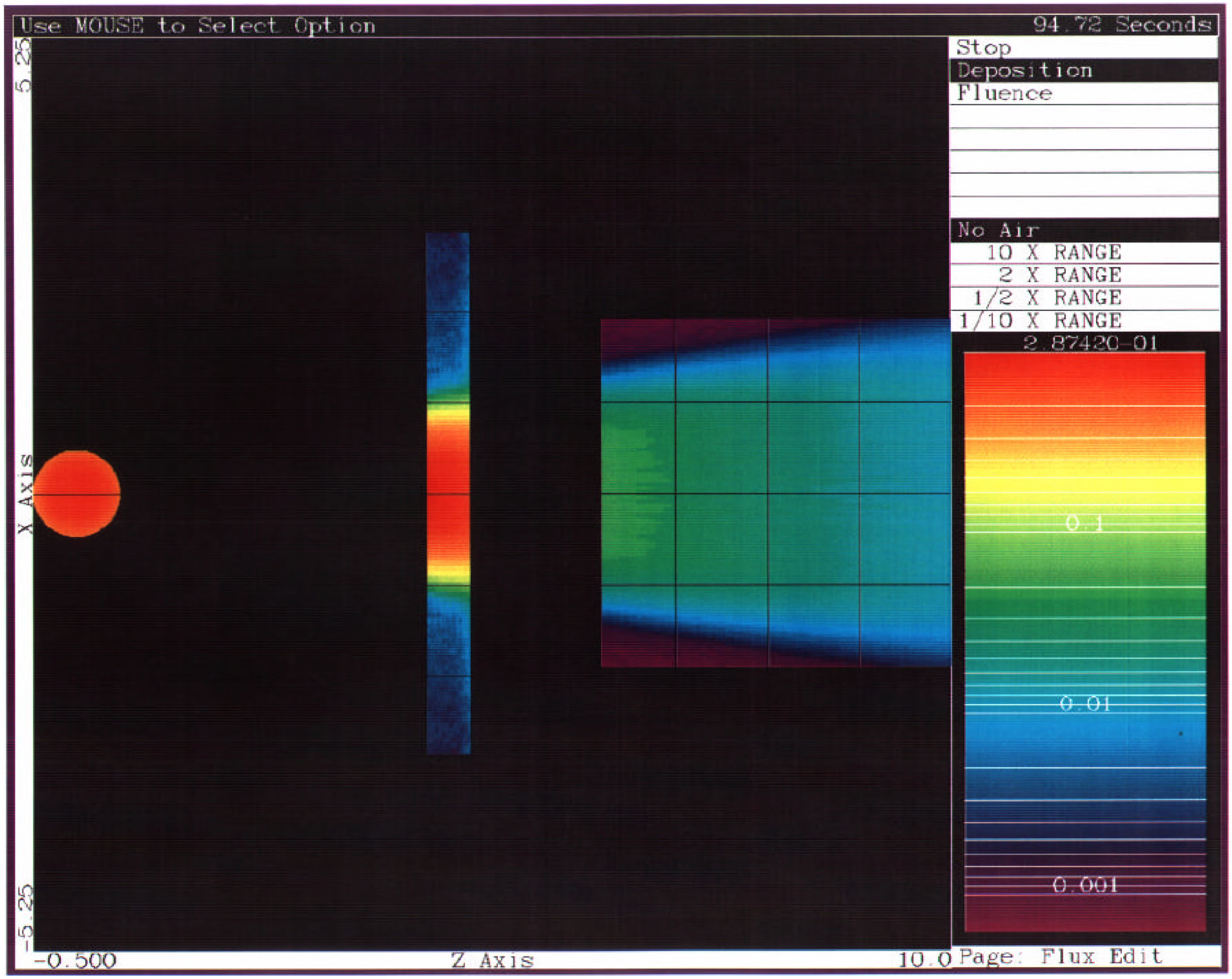
Even though TARTAID only allows you to define everything required for steps 1) through 4), it uses default parameters to define everything required to produce a complete TART input file, and at the end of each run it outputs the file as **TART.OUT**. You can

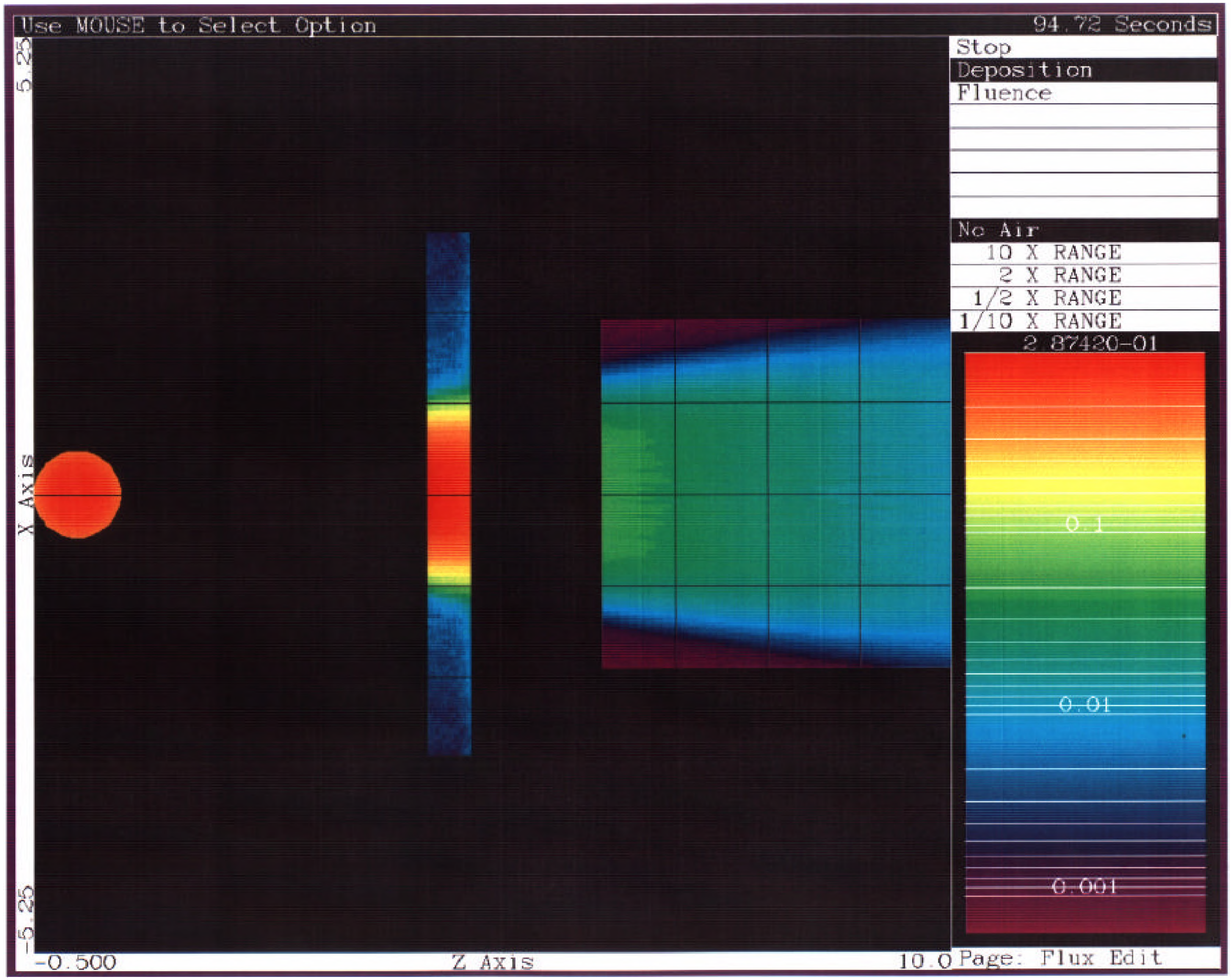


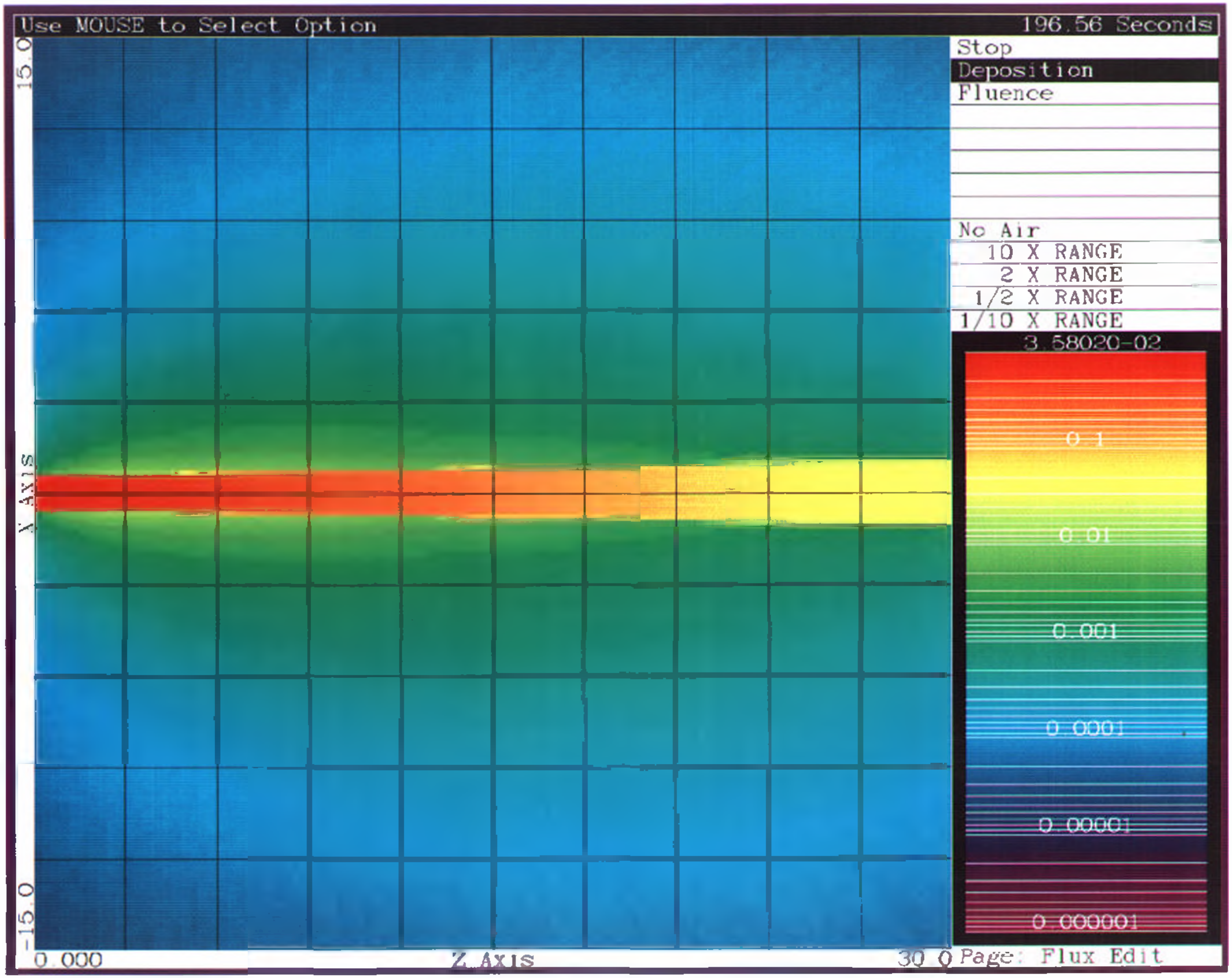
Eye Left -45
Eye Up 60

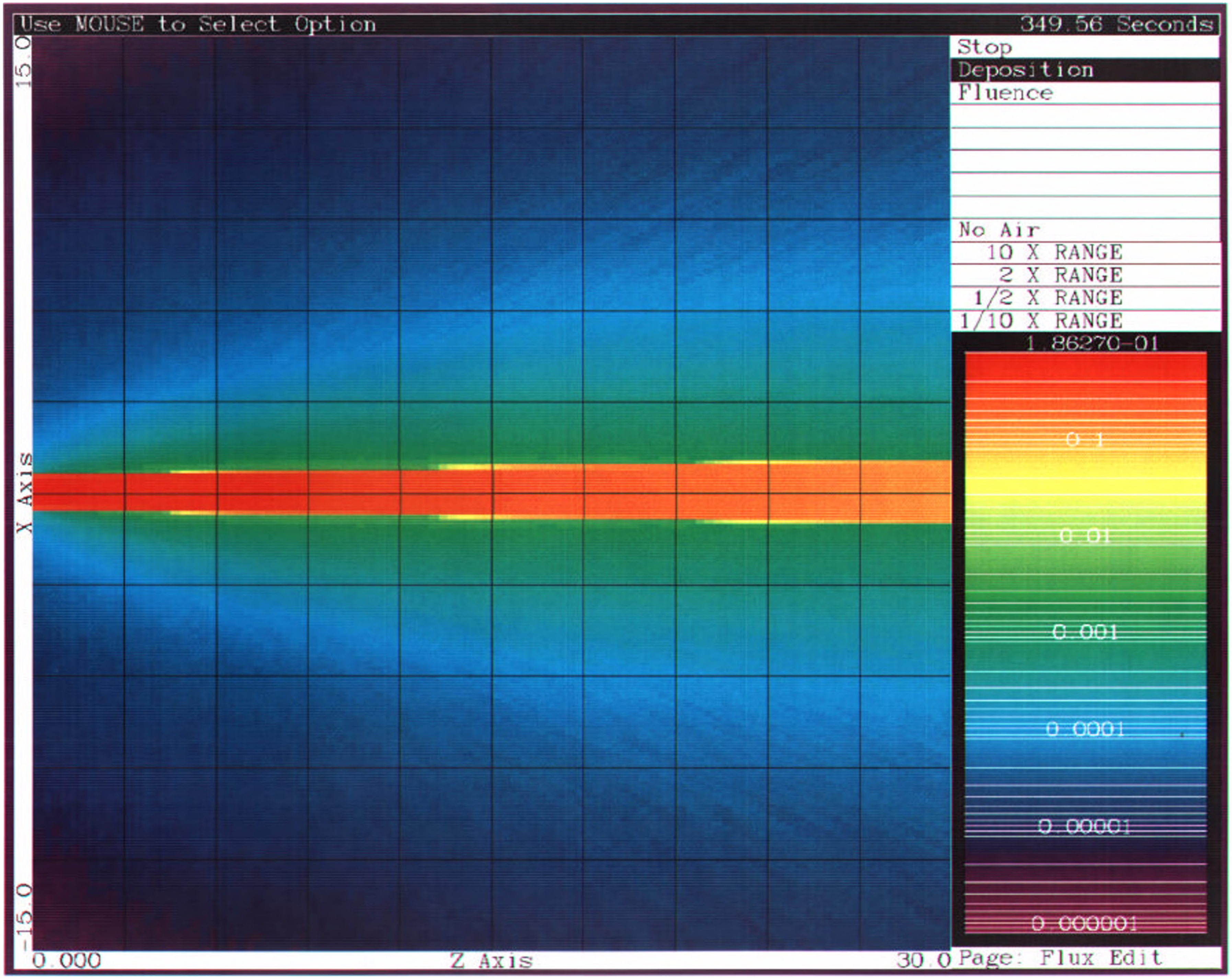

Use MOUSE to Select Option

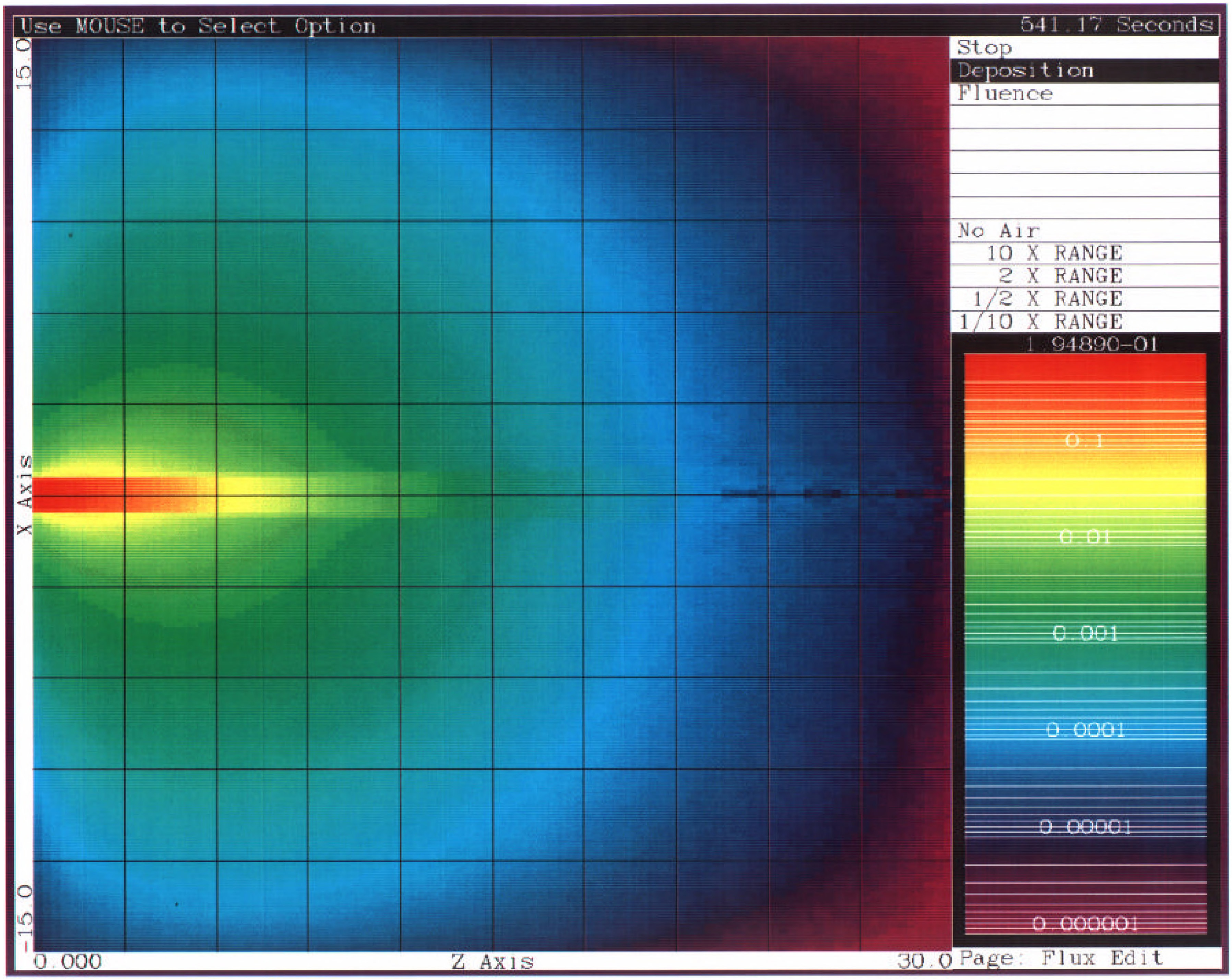
29.39 Seconds

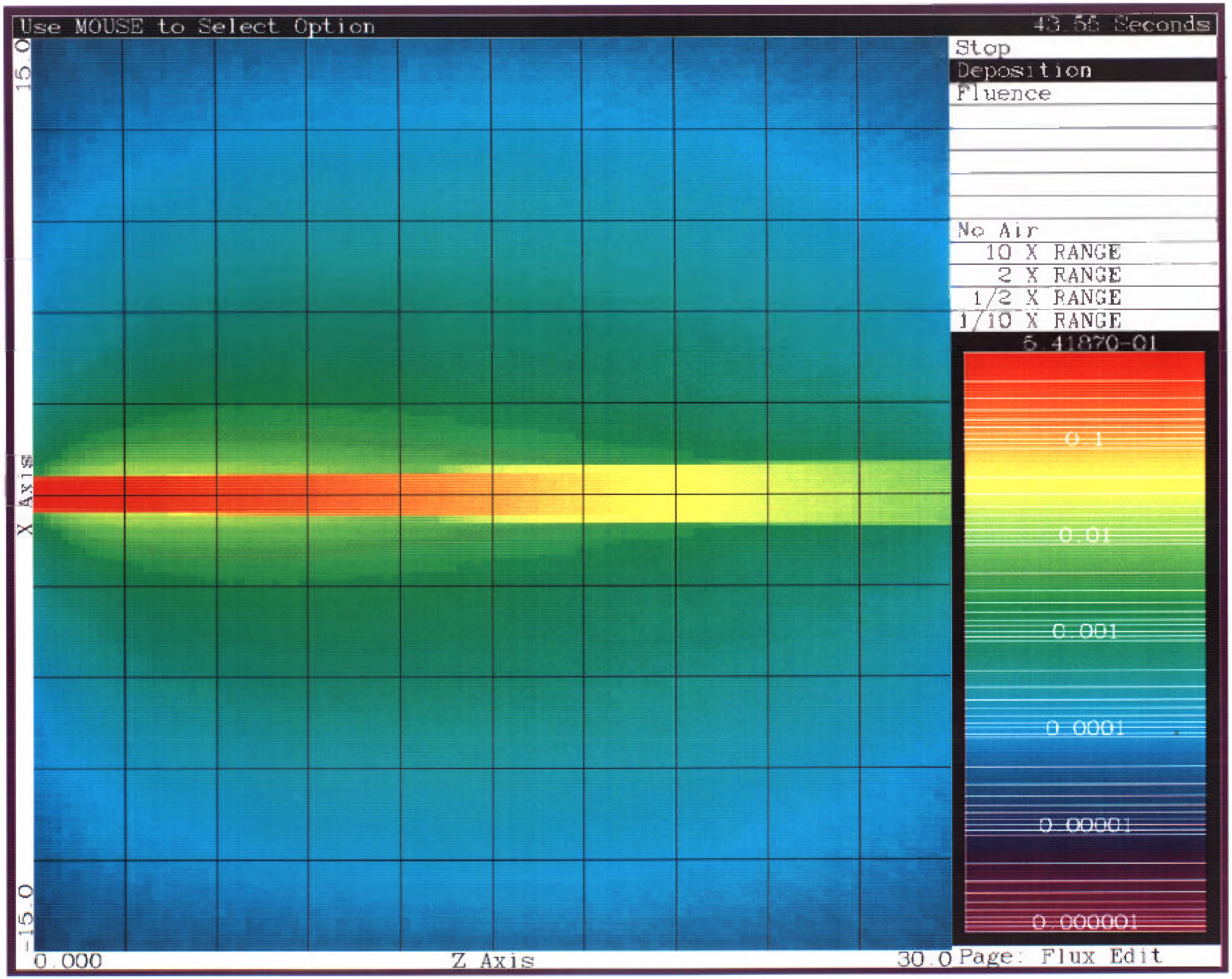

Use MOUSE to Select Option
94.72 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
2.87420-01
0.1
0.01
0.001
5.25
X Axis
-5.25
-0.500
Z Axis
10.0
Page: Flux Edit


Use MOUSE to Select Option
94.72 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
2.87420-01
0.1
0.01
0.001
5.25
X Axis
-5.25
-0.500
Z Axis
10.0
Page: Flux Edit


Use MOUSE to Select Option
196.56 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
3.58020-02
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X AXIS
-15.0
0.000
Z AXIS
30.0
Page: Flux Edit


Use MOUSE to Select Option
349.56 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
1.86270-01
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X Axis
-15.0
0.000
Z Axis
30.0
Page: Flux Edit


Use MOUSE to Select Option
541.17 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
1.94890-01
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X Axis
-15.0
0.000
Z Axis
30.0
Page: Flux Edit


Use MOUSE to Select Option
43.55 Seconds
Stop
Deposition
Fluence
No Air
10 X RANGE
2 X RANGE
1/2 X RANGE
1/10 X RANGE
5.41870-01
0.1
0.01
0.001
0.0001
0.00001
0.000001
15.0
X Axis
-15.0
0.000
Z Axis
30.0
Page: Flux Edit

then edit **TART.OUT** to define parameters for steps 5) through 7). In most cases this only involves defining the source specific to your problem, see, the below example.

Error Checking and Correction

There is a minimum of error checking and correction in the current code. Basically if you input anything that the codes recognizes as illegal, your input is ignored. For example, the density of all materials MUST be positive, and the radii of ALL cylinders and spheres MUST be positive - if you input contradicts this requirement it will be ignored, the you will be given another chance to enter your input.

However, if you input anything that the code finds acceptable, there is no way for you to later change it. For example if by mistake you input the density of water as 10.0 grams/cc, instead of 1.0, you are stuck with it. From a practical viewpoint this isn't as hard to deal with as you might think. Just get used to not worrying about such problems, since you will be able to easily correct it later by editing the output file **TART.OUT**. Once you get used to using **TARTAID** you will find that you can zip through it and create an entire TART deck in minutes. So if you make a major input error it is usually easiest to abort the code and start over. Below I'll discuss what's a major input error.

The Quickest Way to Learn to Use TARTAID, is to Use IT

So that's what we are going to do. The only thing that I will assume is that you are generally familiar with TART input parameters, so that as we go through each step using TARTAID, you can generally relate what we are inputting on the screen to how it will appear in a typical TART input deck.

WARNING - the biggest problem that I encounter when trying to teach users to use any interactive graphics code is to teach them to **RELAX and SLOW DOWN, THE COMPUTER WILL WAIT.** When users are confronted with a computer screen that is asking them for input they often make the mistake of feeling that the computer is pressuring them to immediately respond. It's NOT!!! Think about it: the dumb computer isn't pressuring you - it will sit there forever and wait for your input - you are in control. So take your time and slow things down to a pace that is comfortable for you.

WARNING - An IMPORTANT point to remember is that every time you run **TARTAID** it will output results into a file named **TART.OUT**. So if you don't want to lose your output from one run, before you run **TARTAID** again, remember to rename **TART.OUT**.

Now you start TARTAID running, and I'll talk you through creating your first TART input deck. After that I think you will find it to be easy to use the code on your own.

Screen 1: When you first start TARTAID running the first screen will identify the version of TARTAID that you are running - currently Version 98-1. Press ANY mouse key to proceed to the next screen. Note, if you report an error or problem to me it is important that I know what version of the code you are using.

Screen 2: The next screen merely repeats what I said above, namely,

Creating a TART Input Deck requires the following steps,
1) Define General Running Conditions.
2) Define ALL Materials for your Problem.
3) Define your Spatial Zones.
4) Assign a Material to EACH Zone.
5) Define your Source.
6) Define Weights.
7) Define Tally and Output Options.

This version of TARTAID ONLY does steps 1) through 4).
Currently you have to do steps 5) through 7) Manually.

Press ANY Mouse Key when you are ready to Start.

Screen 3: On the next screen you MUST define General Running Conditions. All of these conditions have default values, EXCEPT for the last input parameter, defining your type of geometry. I've tried to keep this input as simple and as short as possible. When you are given choices usually responses are only one letter, e.g., N for Neutron, P for Photon, Y for Yes, N for No, etc.. In ALL cases these responses are case insensitive, e.g., you can use upper case N or low case n, for Neutron. If you make a mistake and the code doesn't accept your input, it will remove your input, display a black area and wait for you to input again. If you make a mistake, but the code accepts it, don't worry about it; you can fix it later in the file **TART.OUT**.

Here's what you will see on the screen - only you will see it one line at a time, and you are expected to input something before preceding to the next line. Here each line prompts you to define a TART general running condition, and tells you what the default will be if you press the ENTER key without first inputting ANY characters.

```
Problem Name..............................(Test)
Output Box..................................(T32)
C=Criticality/S=Source.....................(C )
Source Neutron or Photon..................(N)
Track Neutron/Photon/Both.................(B)
Thermal Scattering Y/N?....................(Y)
Temperature MeV....................(2.53E-8)
Self-Shielding Y/N?.........................(Y)
Fluorescence Y/N?..........................(Y)
Number of Batches...........................(20)
Histories per Batch........................(5000)
Starting Zone Number........................(1)
Starting Surface Number.....................(1)
```

```
Geometry 1 = Planar
         2 = Cylindrical
         3 = Spherical
         4 = R-Z
Select Geometry 1 to 4......................(4)
```

In most cases it is recommended that you use the default values, i.e., just press ENTER to proceed to the next line. Remember if you change your mind later it is trivial to change any of these parameters in the file **TART.OUT**. So don't worry too much about what you enter here. The only things to consider at,

1) **Do you want to run a criticality or source problem?** If you say criticality [enter c or C] your TART input deck will include a critcalc input line. If you say source [enter s or S] your TART input deck will not include a critcalc input line. If you say criticality, you can then only have a neutron source, and can only track neutrons - no photons. In this case you will not be given a choice of some of the following input parameters; all options to do with photons will be defined for you.

2) **For source problems do you want a neutron or photon source?** You are not allowed to specify both.

3) **For source problems what do you want to track?** This only applies if you want to run a neutron source problem. This is the only case in which you can decide to track only neutrons or neutrons and photons (photons produced by neutron interactions).

4) **Starting Zone and Surface numbers?** For now use the default values. Later we will return to this input when we get to a more complicated problem.

5) **Geometry?** This current version only treats fairly simply 1-D or 2-D geometry.

For now let's skip through all of this input and use the default value for everything until we come to geometry. To do this just press your ENTER key repeatedly until you come to Geometry. At Geometry press your ENTER key again. You will see a black area displayed, indicating that your input was illegal, and giving you another chance to enter a legal parameter. In this case you MUST enter a number from 1 to 4 followed by your ENTER key. As soon as you enter an acceptable value (1 through 4) for geometry the code proceeds to the next screen. For this example, enter 4 followed by ENTER, to select R-Z geometry.

Screen 4: On the next screen you will define ALL of the materials you want to use in your problem. You will do this in EXACTLY the manner used by TART. For each material you will define,
1) The overall density of the material in grams/cc.
2) Whether the constituent "fraction" is defined in terms of atoms or grams (A or G).
3) Then you select a constituent, i.e., 1-H-1, for hydrogen, or 26-Fe-Nat, for iron. You select a constituent by positioning your mouse and clicking on the material.
4) The then define the constituents "fraction"

You can repeat steps 3) and 4) as many times as necessary to define ALL of the constituents of a material.

Let's start,

1) Click on "Define Material" - the right of the screen will go black.
2) The upper left says "Material Density (grams/cc)" - type 1.0 followed by ENTER to define an overall density of 1.0 grams/cc.
3) The upper left says "Fractions in Grams/cc or Atoms/cc (enter G or A)" - type A followed by ENTER, so that each constituent will be defined by atoms/cc.
4) Next position your mouse over "1-H-1" and click - to select hydrogen.
5) The upper left says "Fraction (Atoms/cc)" - type 2 followed by ENTER.
6) Next position your mouse over "8-O-16" and click - to select oxygen
7) The upper left says "Fraction (Atoms/cc)" - type 1 followed by ENTER.
8) To end the material definition move your mouse into the black area to the right and click.

Congratulations - you just successfully defined water as having a density of 1 gram/cc and composed of 2 atoms of hydrogen for each 1 atom of oxygen. Note that the only absolute value is the overall density in grams/cc. All of the fractions will be normalized for you by TART. For example, in this case you could have said there is 1 atom of hydrogen to 0.5 atoms of oxygen, or any other two numbers that are in a ratio of 2:1.

Note, that as you entered each line of input defining your material, the code interpretation of your input is listed to the right. By the time you have finished defining water this will say,

```
1.0
Atoms/cc
 1-H-1
2
 8-O-16
1
```

If you don't remember what you input or you aren't sure how the code interpreted your input, you should occasionally check this list.

WARNING - Let me stop here and mention the most probable problem you will have defining materials - you will forget the order of the input and try to do something with your mouse while the code is expecting you to be typing numbers - this can be very frustrating. You can avoid this by **ALWAYS READ WHAT THE TOP, LEFT HAND LINE OF THE SCREEN SAYS** - this will always tell you what the code expects you to be doing. My experience has been that after selecting "Define Material" and then defining the Density, I forget that I MUST next define whether constituent "fractions" are in Grams or Atoms (G or A). I find myself banging away with my mouse trying to define the first constituent and I am getting frustrated because the stupid code isn't responding. In order to avoid this frustration all I have to do is **READ WHAT THE TOP, LEFT**

HAND LINE OF THE SCREEN SAYS, which is this case says,
"Fractions in Grams/cc or Atoms/cc (enter G or A)"

Let's define one more material,
1) Click on "Define Material" - the right of the screen will go black.
2) The upper left says "Material Density (grams/cc)" - type 7.8 followed by ENTER to define an overall density of 7.8 grams/cc.
3) The upper left says "Fractions in Grams/cc or Atoms/cc (enter G or A)" - type G followed by ENTER, so that each constituent will be defined by grams/cc.
4) Next position your mouse over "26-Fe-Nat" and click - to select iron.
5) The upper left says "Fraction (Atoms/cc)" - type 1 followed by ENTER.
6) Next position your mouse over "6-C-12" and click - to select carbon
7) The upper left says "Fraction (Atoms/cc)" - type 0.02 followed by ENTER.
8) To end the material definition move your mouse into the black area to the right and click.

Congratulations - you just successfully defined a simple form of steel as having a density of 7.8 gram/cc and composed of 1 gram of iron for each 0.02 grams of carbon. This is a simple form of steel with 2 % by weight of carbon. Again note that the numbers 1 and 0.02 can be entered as any values as long as the 50:1 ratio is correct.

Note, again that as you entered each line of input defining your material, the code interpretation of your input is listed to the right. By the time you have finished defining steel this will say,

```
7.8
Grams/cc
 26-Fe-Nat
1
  6-C - 12
0.02
```

That's as many materials as we need for the steps that follow, so we can now proceed to the next step. However, before we move on, remember here we only defined two materials, and each material had only two constituents. But in a real problem you can define as many materials as you need, and each material can have any number of constituents.

To move on to the next screen select "Finished".

Screen 5: On the next screen you will define ALL of the surfaces of your zones. Since we selected R-Z geometry our surfaces can be planes (Z) and cylinders (R). You can define one simple surface by entering as single number, e.g., 4.3. Or you can define many surfaces at once by entering a lower and upper limit and the number of surfaces you want, e.g., 3.2/9.6/3, says define surfaces at 3.2, 6.4, and 9.6 = 3 surfaces equally spaced between 3.2 and 9.6. If you use ranges there MUST be 3 fields and they MUST be separated by / - if your input isn't in this form it will be ignored and the code will erase your input and wait for your next input.

If you make a mistake and enter something illegal the code will merely ignore your input and cycle back to waiting for correct input. Illegal input includes,
1) Non-positive radius.
2) Non-positive number of surfaces.

The most common error at this point is to think you are defining zones directly, rather than the surfaces that will bound your zones. For example, remember if you want **10** zones equally spaced along the Z axis between 0 and 10 cm, you MUST define **11** planes. However, if you want **10** zones equally spaced radially between 0 and 10 cm, you MUST define **10** cylinders - not **11**, as for planes.

WARNING - Before proceeding let me again stop and point out that this is **THE MOST IMPORTANT STEP IN USING TARTAID TO CREATE TART INPUT**. Defining geometry for any Monte Carlo code that uses 3-D, Combinatorial geometry is VERY DIFFICULT. Therefore if you can avoid difficulties by using **TARTAID** you will have saved yourself a lot of time and effort.

So during this step if you make any mistakes immediately consider aborting and starting over. What's a major mistake? It is a major mistake if you define the wrong number of surfaces. For example, if you want to define 10 cylinders and by mistake you define 100 or 1,000 or even 10,000, IMMEDIATELY stop and start over. The number of zones defined in your output file **TART.OUT** is based on how many surfaces you define. So if you get the number of surfaces wrong, your output file will be useless. What's a minor mistake? Surprisingly the wrong positions of surfaces can be a minor mistake. If you don't have too many surfaces, say 10 or so, you can always later edit **TART.OUT** to move your surfaces to any position that you want. However, if you have many surfaces, it is better to stop and start over.

Let's see how quickly we can define a 10 by 10 R-Z grid of zones,
1) Click on "Planes"
2) Type 0/10/11 followed by ENTER - 11 planes will be displayed equally spaced along the Z axis between Z = 0 and 10 cm.
3) Click on "Cylinder"
4) Type 1/10/10 followed by ENTER - 10 cylinders will be displayed equally spaced radially.

Congratulation - you just defined a 10 by 10 (100 zones) R-Z grid of zones. Couldn't be easier, right? If you wanted 100 by 100 (10,000 zones) R-Z grid of zones it isn't any more difficult - your input would have been 0/10/101 and 1/10/100 - a piece of cake.

For this example, you have now defined all surfaces, so you can move to the next screen by selecting "Finished".

Screen 6: On the next screen you will assign a material to each zone. To save time the code starts out by assigning the first material that you defined earlier (water) to each zone - to save yourself time, remember this point, and plan ahead when you use TARTAID for your real problems. Always define the material that is in most of your zones as the first material that you define as **TARTAID** input.

If your problem only included one material, there is nothing else for you to do and you can select "Finished". In this example, earlier we defined two materials (water and steel) so that you can learn how to assign a material to a zone.

To start assigning materials select "Assign Material" and the code will display a list of the materials you defined earlier, where each material is identified by its first defined constituent, and each material is assigned a different color. In this example the list will look like,

1-H -1	(in RED)
26-Fe-Nat	(in GREEN)

Note that to start all zones are RED, indicating that water has been assigned to all zones. To assign steel to some of the zone click on the 26-Fe-Nat GREEN area - the selected material will be identified in the upper left. Now to assign steel to any zone, position your mouse within the zone and click. You can do this as many times as you like and when you are finished assigning this material move your mouse into the black area to the right and click.

This will bring you back to where you started on this page. You can next again select "Assign Material" to make more changes to the material assigned to each zone, or "Finished".

At this stage if you make a mistake and assign the wrong material to a given zone you can correct yourself by continuing to "Assign Material" until you have everything the way you want it. While you are doing this you can change the material assigned to any given zone any number of times until you are satisfied.

Once you select "Finished" the code will output your **TART.OUT** file and it will then terminate. That's it - when you get this to this point you have used **TARTAID** to create a complete TART input deck.

Manually Editing TART.OUT

The last step required is to manually edit **TART.OUT** to define a source specific to this problem. The example TART input deck that we just created is listed in the appendix. These few simple steps that we followed resulted in a complete TART input deck, in this case 253 lines long.

This includes a default definition of the source as,
1) a neutron source (**sentl 1**).
2) a sphere of radius 1 cm centered on the Z axis and at Z = 5 cm along the Z axis (the default is a source at the center of the defined space) (**source3**)
3) a fission spectrum energy distribution (**sentl 4**)
4) isotropic in direction (**sentl 6 and 7**)

Let's modify **TART.OUT** and then demonstrate that the result is a complete TART input deck by using it as input to TART98. As output by **TARTAID, TART.OUT** is already a complete TART input deck, so why do we have to modify it? The only thing that we MUST change is that we accepted the default to run a criticality problem, but there aren't any fissile materials in our problem, so if you try to run it we will get a strange result; try it if you want to see.

Let's change this from a criticality problem to a source problem. To do this we have to make only two changes,

1) First, comment out or delete the "critcalc" line.
2) Change **sentl 2 2000** to **sentl 2 20**

For criticality problems the number of batches (**sentl 1**) is set to a large number to insure convergence is reached before the maximum number of batches (2000) is exceeded; so we assume the code will never really run this number of batches. However, for source problems the code will run exactly the number of batches we tell it to run. If you want to run 2000 batches, that's fine, and no change is required. For our example problem let's run only 20 batches by changing **sentl 2**.

Just so that you get used to doing this, next let's modify the source to be a point source on the Z axis just slightly inside the R-Z geometry, with a 14.1 MeV monoenergetic energy, monodirectional, straight up the Z Axis. As an exercise modify **TART.OUT** to include,

```
* a neutron point source just slightly inside the R-Z geometry
source1  1   0. 0. 1.0e-05
* 14.1 MeV monoenergetic energy
sentl  4   14.1
* monodirectional straight up the Z axis
sentl   6   0.0
sentl   7   1.0
```

Now you can use this file as input to **TART98**, but first let's change its name, since **TART.OUT** is the default name of the **TART98** output file.

The Real Power of TARTAID

The example that we went through above was relatively simple, but hopefully it got you used to two important concepts,

1) You can use **TARTAID** to quickly produce a TART input deck.
2) You can then modify the deck **TART.OUT**.

The next thing to get used to is that the deck **TART.OUT** starts with a few run parameters, followed by longer lists of surfaces and zone definitions, followed by a short section defining the source and more run parameters. Most of a typical **TARTAID** output file defines surfaces and zones for a simple 1-D or 2-D geometry. This is the most difficult part of any TART input deck to produce.

The real power of TARTAID is that you can use it to define any number of TART decks, each defining a relatively simple 1-D or 2-D geometry, and you can then edit the decks together to define more complicated geometry. In order to do this let's go back to a few input parameters that we skipped earlier. On **Screen 3** you can define,

Starting Zone Number........................(1)
Starting Surface Number.....................(1)

In order to be able to simply combine TARTAID decks all you have to do is,
1) Insure that the various parts of the geometry do not overlap.
2) Insure that the surface and zone numbers of various parts of the geometry do not overlap. This is where the above input can be used.

See if you can put together this problem,
1) A cobalt sphere 0.5 cm in radius centered at Z = 1.0,
2) A lead cylindrical filter 2.0 cm in radius between Z = 4.0 and 4.5 cm,
3) A silicon detector 2.0 cm in radius between Z = 6.0 and 10.0 cm

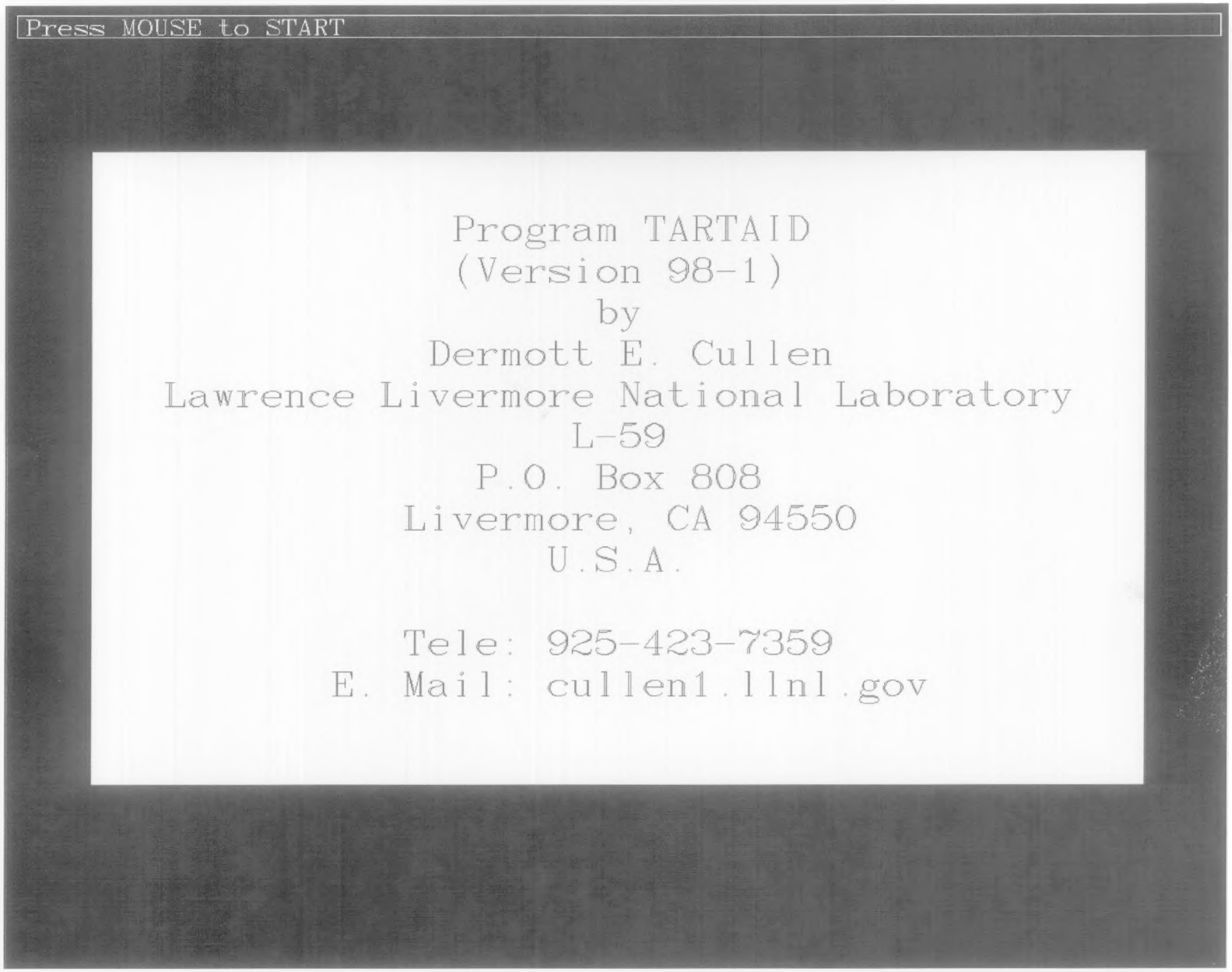
If you represent each of these three objects as a single spatial zone it should be relatively simple for your to manually create a TART deck. But if you do this you won't learn anything about the spatial distribution of flux or energy deposition in these objects.

So try using **TARTAID** to define a number of TART decks to describe each object in detail. Use 100 radial zones in the cobalt sphere, and 100 by 100 R-Z zones in the lead filter and silicon detector. You can then patch the output results together to define a complete problem. The only zones that you will have to manually define are air zones between the three objects, and vacuum zones surrounding everything; just a hand full of zones. The result will be a problem with over 20,000 zones, and after you run **TART98** you can immediately run **FLUXEDIT** and then use **TARTCHEK** to see your results overlaid on your geometry. With this approach within a few minutes you will see your results, and you should never have to actually look at the long **TART98** output report.

Start by using **TARTAID** to define the cobalt sphere. Here you can use the default starting surface and zone numbers. When you finish this **TARTAID** run, first rename **TART.OUT** to **DECK1**. Look in this file and find out what is the largest surface and zone number used. Next, define the lead filter. Here use starting surface and zone numbers that are larger than those used to define the cobalt sphere. When you finish this **TARTAID** run, first rename **TART.OUT** to **DECK2**. Look in this file and find out what is the largest surface and zone number used. Next, define the silicon detector. Here use starting surface and zone numbers that are larger than those used to define the lead filter. When you finish this **TARTAID** run, first rename **TART.OUT** to **DECK3**. Look in this file and find out what is the largest surface and zone number used. The last step is to patch these three decks together and add a few surrounding air zones between objects and vacuum zones surrounding everything, using zone numbers higher than those already used, and you are finished.

Example TARTAID Output File (TART.OUT)

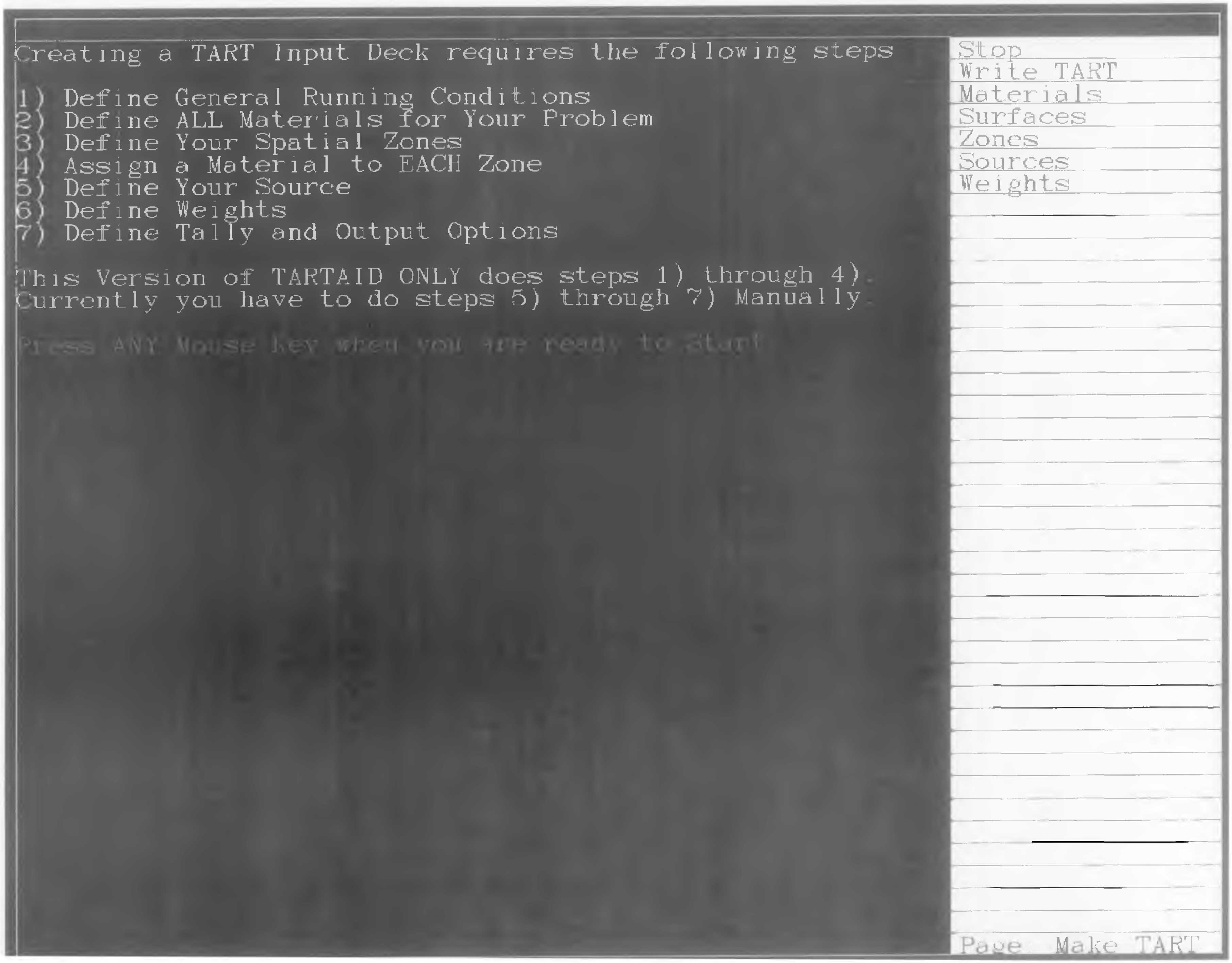
```
name Test
box T32 Test
* ===============================================================================
*
* TART Input Deck Generated Using Program TARTAID (Version 98-1)
*
* ===============================================================================
* ===============================================================================
*
* R-Z Geometry
*
* ===============================================================================
* ===============================================================================
*
* Criticality Problem
* Settle Cycles (15)
* Repetitions (1)
* Standard Deviation (1%)
* Time Step in Shakes (0) Dynamic Problems ONLY
*
* ===============================================================================
critcalc  15 1 10.
* ===============================================================================
*
* Surfaces Definitions
*
* Surface Keyword, #, and Parameters
*
* ===============================================================================
zplane   1 0.
zplane   2 1.
zplane   3 2.
zplane   4 3.
zplane   5 4.
zplane   6 5.
zplane   7 6.
zplane   8 7.
zplane   9 8.
zplane   10 9.
zplane   11 10.
cylz     12 1.
cylz     13 2.
cylz     14 3.
cylz     15 4.
cylz     16 5.
cylz     17 6.
cylz     18 7.
cylz     19 8.
cylz     20 9.
cylz     21 10.
* ===============================================================================
*
* Material Definitions
*
* matl   # grams/cc Followed by list of ATOM Fractions   and ZA
* matlwp # grams/cc followed by list of WEIGHT Fractions and ZA
*
* ===============================================================================
matl     1 1. 2. 1001 1. 8016
matlwp   2 7.8 1. 26000 0.02 6012
* ===============================================================================
*
* Assign Material to Zones
*
* Material # Followed by Zones Material is Assigned to
*
* ===============================================================================
matz  1 1  thru  44
matz  1 46  thru  55
matz  1 57  thru  66
matz  1 68  thru  77
matz  1 79  thru  99
matz  2 45
matz  2 56
matz  2 67
matz  2 78
matz  2 100
* WARNING - Only legal for exterior, non-re-entrant zones
matz  0 101  thru  103
* ===========================================================================
*
* Neutron Minimum Energy by Zone.
* If Thermal Scattering (sentl 39) is used emin is the
* Temperature in each zone in MeV (2.53d-08)
*
* ===========================================================================
emin  2.53e-8 1  thru  103
* ===========================================================================
*
* Surfaces Bounding Zones
*
* Zone # Followed by Signed Surfaces Bounding Zone
*
* ===========================================================================
jb  1 -1 2 12
jb  2 -2 3 12
jb  3 -3 4 12
jb  4 -4 5 12
jb  5 -5 6 12
jb  6 -6 7 12
jb  7 -7 8 12
jb  8 -8 9 12
jb  9 -9 10 12
jb  10 -10 11 12
jb  11 -1 2 -12 13
jb  12 -2 3 -12 13
jb  13 -3 4 -12 13
jb  14 -4 5 -12 13
jb  15 -5 6 -12 13
jb  16 -6 7 -12 13
jb  17 -7 8 -12 13
jb  18 -8 9 -12 13
jb  19 -9 10 -12 13
jb  20 -10 11 -12 13
jb  21 -1 2 -13 14
jb  22 -2 3 -13 14
jb  23 -3 4 -13 14
jb  24 -4 5 -13 14
jb  25 -5 6 -13 14
jb  26 -6 7 -13 14
jb  27 -7 8 -13 14
jb  28 -8 9 -13 14
jb  29 -9 10 -13 14
jb  30 -10 11 -13 14
jb  31 -1 2 -14 15
jb  32 -2 3 -14 15
jb  33 -3 4 -14 15
jb  34 -4 5 -14 15
jb  35 -5 6 -14 15
jb  36 -6 7 -14 15
jb  37 -7 8 -14 15
jb  38 -8 9 -14 15
jb  39 -9 10 -14 15
jb  40 -10 11 -14 15
jb  41 -1 2 -15 16
jb  42 -2 3 -15 16
jb  43 -3 4 -15 16
jb  44 -4 5 -15 16
jb  45 -5 6 -15 16
jb  46 -6 7 -15 16
jb  47 -7 8 -15 16
jb  48 -8 9 -15 16
jb  49 -9 10 -15 16
jb  50 -10 11 -15 16
jb  51 -1 2 -16 17
jb  52 -2 3 -16 17
jb  53 -3 4 -16 17
jb  54 -4 5 -16 17
jb  55 -5 6 -16 17
jb  56 -6 7 -16 17
jb  57 -7 8 -16 17
jb  58 -8 9 -16 17
jb  59 -9 10 -16 17
jb  60 -10 11 -16 17
jb  61 -1 2 -17 18
jb  62 -2 3 -17 18
jb  63 -3 4 -17 18
jb  64 -4 5 -17 18
jb  65 -5 6 -17 18
jb  66 -6 7 -17 18
jb  67 -7 8 -17 18
jb  68 -8 9 -17 18
jb  69 -9 10 -17 18
jb  70 -10 11 -17 18
jb  71 -1 2 -18 19
jb  72 -2 3 -18 19
jb  73 -3 4 -18 19
jb  74 -4 5 -18 19
jb  75 -5 6 -18 19
jb  76 -6 7 -18 19
jb  77 -7 8 -18 19
jb  78 -8 9 -18 19
jb  79 -9 10 -18 19
jb  80 -10 11 -18 19
jb  81 -1 2 -19 20
jb  82 -2 3 -19 20
jb  83 -3 4 -19 20
jb  84 -4 5 -19 20
jb  85 -5 6 -19 20
jb  86 -6 7 -19 20
jb  87 -7 8 -19 20
jb  88 -8 9 -19 20
jb  89 -9 10 -19 20
jb  90 -10 11 -19 20
jb  91 -1 2 -20 21
jb  92 -2 3 -20 21
jb  93 -3 4 -20 21
jb  94 -4 5 -20 21
jb  95 -5 6 -20 21
jb  96 -6 7 -20 21
jb  97 -7 8 -20 21
jb  98 -8 9 -20 21
jb  99 -9 10 -20 21
jb  100 -10 11 -20 21
jb  101 -11
jb  102 1
jb  103 -21
* ===============================================================================
*
* Neutron Source Definitions
*
* ===============================================================================
*
* Neutron source position (Point at 0,0,0)
*
* WARNING - Position UNDEFINED - Using DEFAULT
* Spherical Volume nz ri ro z0
* Near the center of the geometry
source3   1 0. 1.0000000149 5.
*
* Neutron source energy (0=Fission spectrum)
*
* WARNING - Energy UNDEFINED - Using DEFAULT
*  4) Neutron Source Energy (0.0)
sentl  4 0.
*
* Neutron source direction (Isotropic, Cosine=-1.0 to +1.0)
*
* WARNING - Angular Distribution UNDEFINED - Using DEFAULT
*   6) Neutron Source Cosine Interval (2.0)
sentl  6 2.
*   7) Neutron Source Cosine Start (-1.0)
sentl  7 -1.
*
* Neutron source initial time (0.0)
*
* ===============================================================================
*
* Tally and Output Options
*
* ===============================================================================
*   5) Neutron tally type (3)
sentl    5    3
*  33) Photon tally type (3)
sentl   33    3
* ===============================================================================
*
* Running Conditions
*
* ============================================================================
*   1) Source and Tracked Particles (0)
sentl    1    1
*   2) Batches of Particles (20)
sentl    2       2000
*   3) Particles per Batch (5000)
sentl    3       1000
*  20) Multiband Self-Shielding (0)
sentl   20    1
*  25) Fluorescence (0)
sentl   25    0
*  39) Thermal Scattering (0)
sentl   39    1
end
```



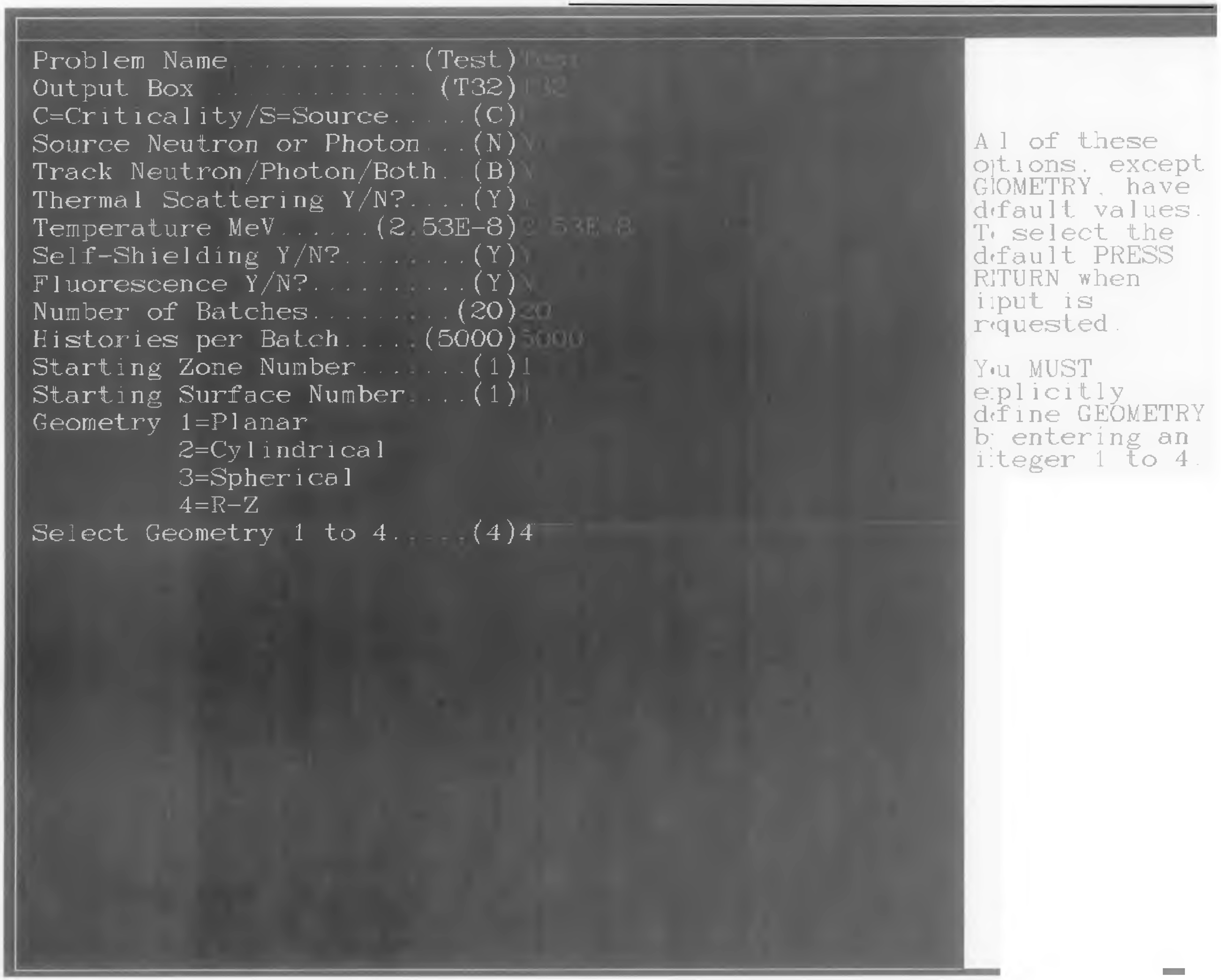
Press MOUSE to START

Program TARTAID
(Version 98-1)
by
Dermott E. Cullen
Lawrence Livermore National Laboratory
L-59
P.O. Box 808
Livermore, CA 94550
U.S.A.

Tele: 925-423-7359
E. Mail: cullen1.llnl.gov



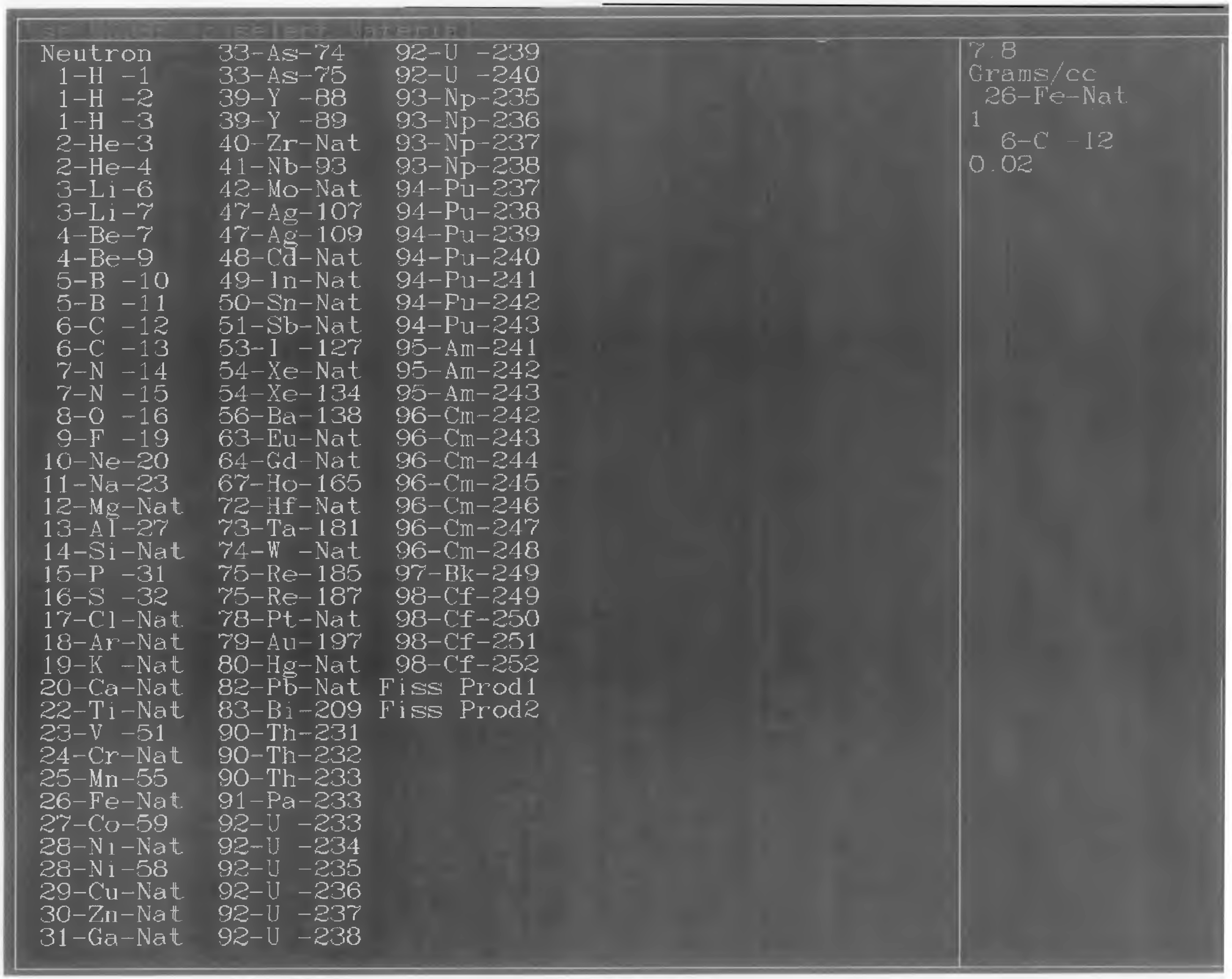
```
Creating a TART Input Deck requires the following steps

1) Define General Running Conditions
2) Define ALL Materials for Your Problem
3) Define Your Spatial Zones
4) Assign a Material to EACH Zone
5) Define Your Source
6) Define Weights
7) Define Tally and Output Options

This Version of TARTAID ONLY does steps 1) through 4).
Currently you have to do steps 5) through 7) Manually.

Press ANY Mouse key when you are ready to Start!
```



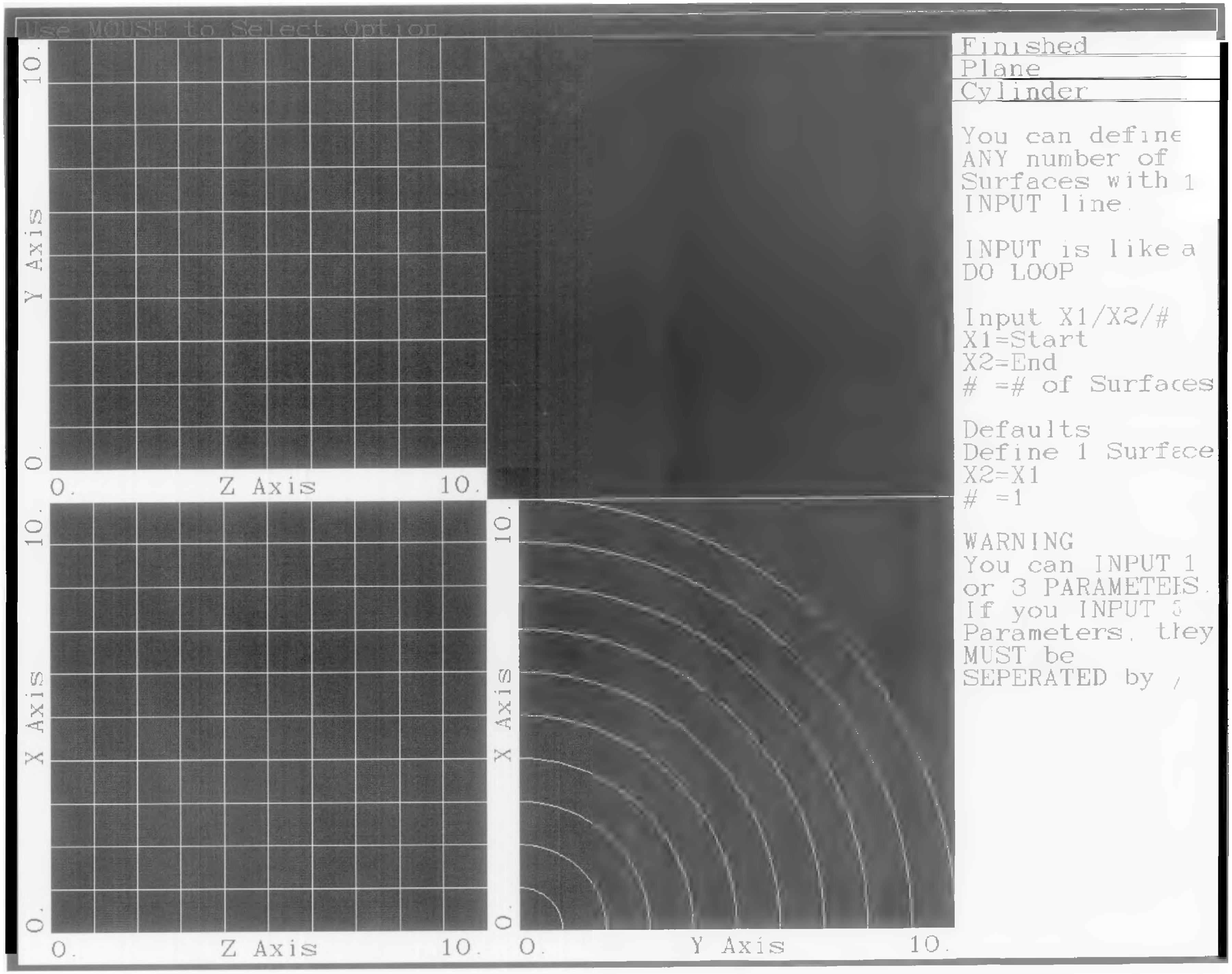
```
Problem Name............(Test)Test
Output Box .............(T32)T32
C=Criticality/S=Source.....(C)C
Source Neutron or Photon...(N)N
Track Neutron/Photon/Both..(B)N
Thermal Scattering Y/N?....(Y)Y
Temperature MeV......(2.53E-8)2.53E-8
Self-Shielding Y/N?........(Y)Y
Fluorescence Y/N?..........(Y)Y
Number of Batches.........(20)20
Histories per Batch.....(5000)5000
Starting Zone Number.......(1)1
Starting Surface Number....(1)1
Geometry 1=Planar
         2=Cylindrical
         3=Spherical
         4=R-Z
Select Geometry 1 to 4.....(4)4
```

All of these options, except GEOMETRY, have default values. To select the default PRESS RETURN when input is requested.

You MUST explicitly define GEOMETRY by entering an integer 1 to 4.



```
[illegible] Select Material
Neutron     33-As-74    92-U -239                     7.8
  1-H -1    33-As-75    92-U -240                     Grams/cc
  1-H -2    39-Y -88    93-Np-235                      26-Fe-Nat
  1-H -3    39-Y -89    93-Np-236                     1
  2-He-3    40-Zr-Nat   93-Np-237                       6-C -12
  2-He-4    41-Nb-93    93-Np-238                     0.02
  3-Li-6    42-Mo-Nat   94-Pu-237
  3-Li-7    47-Ag-107   94-Pu-238
  4-Be-7    47-Ag-109   94-Pu-239
  4-Be-9    48-Cd-Nat   94-Pu-240
  5-B -10   49-In-Nat   94-Pu-241
  5-B -11   50-Sn-Nat   94-Pu-242
  6-C -12   51-Sb-Nat   94-Pu-243
  6-C -13   53-I -127   95-Am-241
  7-N -14   54-Xe-Nat   95-Am-242
  7-N -15   54-Xe-134   95-Am-243
  8-O -16   56-Ba-138   96-Cm-242
  9-F -19   63-Eu-Nat   96-Cm-243
 10-Ne-20   64-Gd-Nat   96-Cm-244
 11-Na-23   67-Ho-165   96-Cm-245
 12-Mg-Nat  72-Hf-Nat   96-Cm-246
 13-Al-27   73-Ta-181   96-Cm-247
 14-Si-Nat  74-W -Nat   96-Cm-248
 15-P -31   75-Re-185   97-Bk-249
 16-S -32   75-Re-187   98-Cf-249
 17-Cl-Nat  78-Pt-Nat   98-Cf-250
 18-Ar-Nat  79-Au-197   98-Cf-251
 19-K -Nat  80-Hg-Nat   98-Cf-252
 20-Ca-Nat  82-Pb-Nat  Fiss Prod1
 22-Ti-Nat  83-Bi-209  Fiss Prod2
 23-V -51   90-Th-231
 24-Cr-Nat  90-Th-232
 25-Mn-55   90-Th-233
 26-Fe-Nat  91-Pa-233
 27-Co-59   92-U -233
 28-Ni-Nat  92-U -234
 28-Ni-58   92-U -235
 29-Cu-Nat  92-U -236
 30-Zn-Nat  92-U -237
 31-Ga-Nat  92-U -238
```



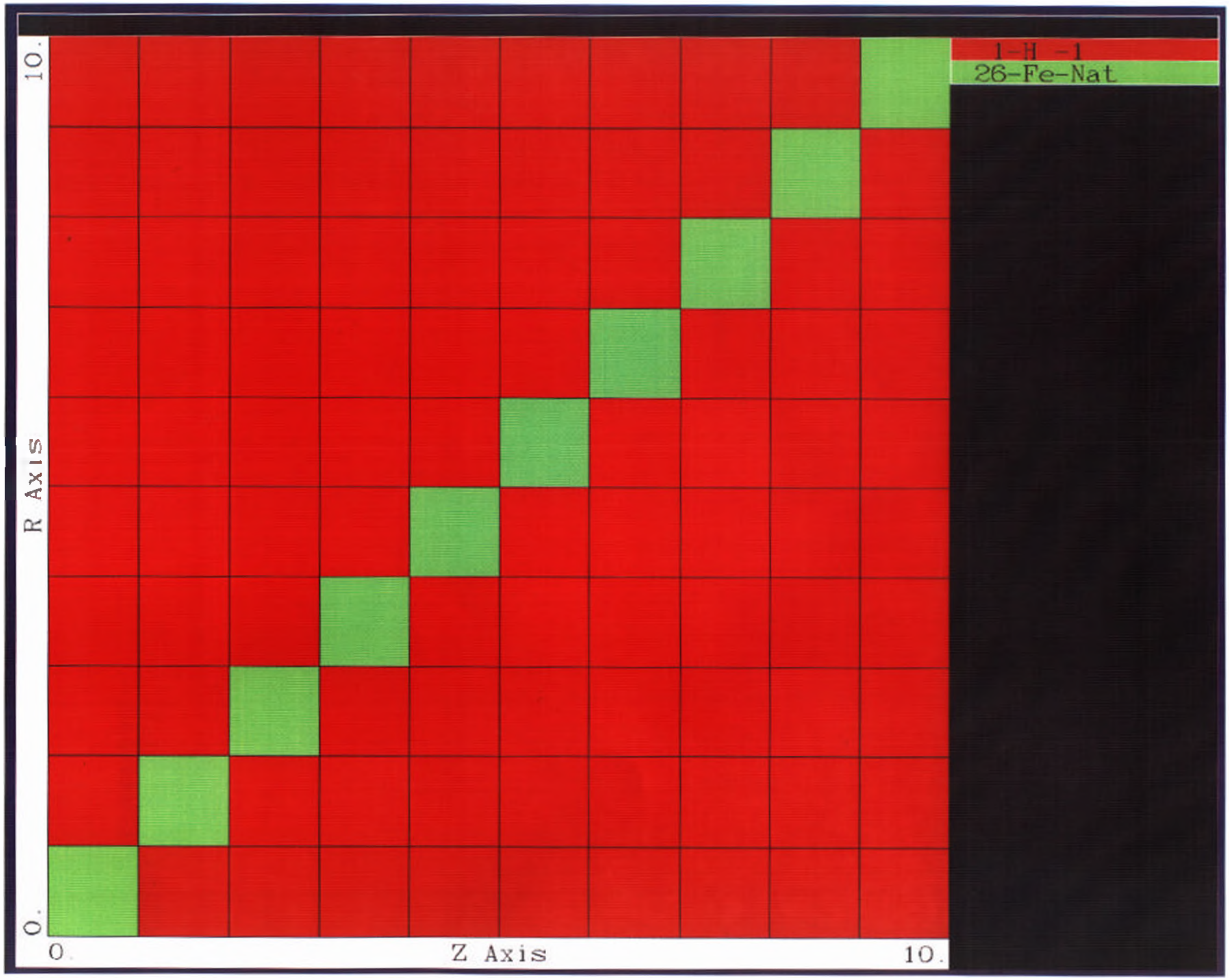
You can define
ANY number of
Surfaces with 1
INPUT line.

INPUT is like a
DO LOOP

Input X1/X2/#
X1=Start
X2=End
=# of Surfaces

Defaults
Define 1 Surface
X2=X1
=1

WARNING
You can INPUT 1
or 3 PARAMETERS.
If you INPUT 3
Parameters, they
MUST be
SEPERATED by /


10.
R Axis
0.
0.
Z Axis
10.
1-H -1
26-Fe-Nat

PROGRAM EPICSHOW
A Computer Code to Allow
Interactive Viewing of the EPIC
Data Libraries
(Version 98-1)

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

tele: 925-423-7359
FAX: 925-422-9560
E.Mail: cullen1@llnl.gov

1998 Update

This document has been updated (October 13, 1998) to describe the 1998 EPICSHOW update to calculate mixtures of material, e.g., water. For details, see the below description of this option and how to define mixtures of materials.

Abstract

EPICSHOW is an interactive graphics code that allows users to view and interact with neutron, photon, electron and light charged particle data. Besides on screen graphics the code provides hard copy in the form of tabulated listings and Postscript output files. The code has been implemented on UNIX, IBM-PC, Power MAC, and even Laptop computers. It should be relatively easy to use it on almost any computer.

All of the data included in this system is based on the Lawrence Livermore National Laboratory Data Bases and the neutron and photon data is used in the TART98 Monte Carlo transport code system.

Overview

For many years the data used at Lawrence Livermore National Laboratory has been documented in the UCRL-50400 report series. Anyone who has subscribed to this series now has an entire bookshelf of reports.

The problem with publishing data in book form is: 1) by the time the books are actually published the data can be out-of-date, 2) books allow users to see the data, but they do not have the data on-line for use in their applications, 3) publishing books is expensive.

As an alternative to the traditional UCRL-50400 series, EPICSHOW is offered as an interactive computer graphics code. Compared to books, a code offers the following advantages: 1) the data files can be maintained up-to-date and periodically distributed, 2) users can not only see the data, they can interact with it and obtain it on their computer in a form that can be easily used in applications, 3) compared to the cost of publishing books, maintaining and distributing this code is relatively inexpensive.

Introduction to EPICSHOW

EPICSHOW is a mouse driven interactive graphics codes. There is no keyboard or any other type of input - ONLY the mouse is used and ALL mouse keys (left, center, or right) are treated the same. There are no commands to remember - every option is on the screen and to use any option all you need do is click your mouse in an option box.

You will find that EPICSHOW is so user friendly that you do not even need this report to efficiently use the code. The easiest way to learn how to use EPICSHOW is to just start using it. Feel free to click away with your mouse to select as many options as you want, and see what happens. You will find that in no time at all you will be an EPICSHOW expert.

The remainder of this report defines computer requirements, documents the sources of the data used in this system, and then briefly describes ALL current options. That's it - that's all you need to use this system and understand where the data came from.

Bottom line: HAVE FUN!!!

Computer Requirements

1 Megabyte of Memory
7 Megabytes of Disk Space
a Mouse
a Color screen is highly recommended, but not an absolute necessity

The code is written in standard FORTRAN that can run on any computer. It is distributed with a very simple graphics interfaces for use on UNIX, IBM-PC, Power MAC, and even Laptops. The code is so standard and the graphics interface so simple, that it should be relatively easy to use it on almost any computer.

Data Contents

EPICSHOW includes data for neutrons, photons, electrons and light charged particles. For neutrons data is included for all materials included in the Livermore ENDL data base; it is included in the form of 175 group (energy) averaged TART98 data over the energy range 0.0013 eV to 20 MeV. For all other types of projectiles, data is included for each element, Z = 1 through 100, over the energy range 10 eV to 1 GeV; in the case of photons high energy analytical extensions are provided to extend the data to 100 TeV.

Data References

All of the data included in this system is based on the Lawrence Livermore National Laboratory Data Bases. Below is a list of the primary references for each type of data. If you use any of our data in your applications these are the references you should quote to document the sources of the data.

Neutrons

The neutron data is based on the Evaluated Nuclear Data Library (ENDL). In this system it is presented in the form of TART98 175 group (energy) averaged data; this is the data used by the Monte Carlo code TART98. The primary references to ENDL include,

"The LLL Evaluated Nuclear Data Library (ENDL): Evaluation Techniques Index, and Descriptions of Individual Evaluations," by R.J. Howerton, et al, (Sept. 1975), UCRL-50400, Vol. 15, Part A, Lawrence Livermore National Laboratory.

"The LLL Evaluated Nuclear Data Library (ENDL): Graphs of Cross Sections from the Library," by R.J. Howerton, et al, (Oct. 1978), UCRL-50400, Vol. 15, Part B, Rev. 1, Lawrence Livermore National Laboratory.

"Tabular and Graphical Presentation of 175 Neutron-Group Constants Derived from the LLL Evaluated Nuclear Data Library (ENDL)," by E.F. Plechaty, et al, (Oct. 1978), UCRL-50400, Vol. 16, Rev. 2, Lawrence Livermore National Laboratory.

Photons

The photon data is based on the Evaluated Photon Data Library (EPDL). Atomic relaxation data to define fluorescence is from the Evaluated Atomic Data Library (EADL). The primary references to EPDL and EADL include,

"The 1989 Livermore Evaluated Photon Data Library (EPDL)," by D.E. Cullen, et al., (March 1990), UCRL-ID-103424, Lawrence Livermore National Laboratory

"Tables and Graphs of Photon-Interaction Cross Sections from 10 eV to 100 GeV Derived from the LLNL Evaluated Photon Data Library (EPDL), Part A: Z=1 to 50, Part B: Z=51 to 100," by D.E. Cullen, et al., (October 1989), UCRL-50400, Vol. 6, Rev. 4, Lawrence Livermore National Laboratory

"Tables and Graphs of Atomic Subshell and Relaxation Data Derived from the LLNL Evaluated Atomic Data Library (EADL)," by S.T. Perkins, et al, (October 1991), UCRL-50400, Vol. 30, Lawrence Livermore National Laboratory

Soon to be added is the latest Livermore photon interaction data, "EPDL97: the Evaluated Photon Data Library, '97 Version, by Dermott E. Cullen, John H. Hubbell , Lynn Kissel, (September 1997), UCRL--50400, Vol. 6, Rev. 5, Lawrence Livermore National Laboratory

It should also be noted that EPDL is the International Standard adopted for the ENDF/B-VI data library as well as the data used by the Monte Carlo code TART98.

Electrons

The electron data is based on the Evaluated Electron Data Library (EEDL). Atomic relaxation data to define electron emission is from the Evaluated Atomic Data Library (EADL); see, above. The primary references to EEDL include,

"Tables and Graphs of Electron-Interaction Cross Sections from 10 eV to 100 GeV Derived from the LLNL Evaluated Electron Data Library (EEDL), Z=1-100," by S.T. Perkins, et al, (November 1991), UCRL-50400, Vol. 31, Lawrence Livermore National Laboratory.

Charged Particle Data

This data is from the TRIM code.

Screen Layout

The screen is divided into two parts: 1) the lower part is used to display data, 2) the upper part is used to identify the version of the code you are using and to display options. Currently forty (40) option boxes are displayed at the top of the screen, although as yet not all are used.

Code Option Summary

Here's a brief description of the options in the order that they appear on your screen - details on using these options are included later in this report.

Lin/Log X	- Switch between linear or log scaling for the X Axis.
Lin/Log Y	- Switch between linear or log scaling for the Y Axis.
Points	- Switch between showing and not showing data points.
Freeze!!!	- Switch between freezing/not freezing the screen.

Zoom X	- Use mouse to define lower/upper X limits for next plot.
Show All	- Reverse the effect of zoom and show the entire X range.
Ratio	- Display data AND ratio of all curves to the first curve.
+Energy	- For photons increase upper energy limit by a decade.
-Energy	- For photons decrease upper energy limit by a decade.

Grid + 1	- Include more details in the grid.
Grid - 1	- Include less details in the grid.
Mixture	- Read mixture from the file EPICSHOW.INP (1998 Update)
barns-1/c	- switch between units of barns and 1/cm.
==========	==========
Legend	- Switch between showing and not showing the legend.
Bigger	- Make characters bigger.
Smaller	- Make characters smaller
ColorDump	- Create color Postscript copy of current screen
==========	==========
Photons	- Display Photon Data
Electrons	- Display Electron Data
Positrons	- Display Positron Data - Not yet implemented
Charged	- Display Light Charged Particle Data
Neutrons	- Display Neutron Data
==========	==========
Major	- Display Major Cross Sections
Minor	- Display Minor Cross Sections
Deposit	- Display Energy Deposition Data
Range	- Display Range Data - Only Electrons and Charged
Straggle	- Display Straggling Data - Not yet implemented
==========	==========
Z + 1	- Position 1 Evaluation UP in the Periodic Table
Z + 10	- Position 10 Evaluations UP in the Periodic Table
Z - 1	- Position 1 Evaluation DOWN in the Periodic Table
Z - 10	- Position 10 Evaluations DOWN in the Periodic Table
Same Z	- Display the same data again
==========	==========
2 X 2	- Switch between displaying 1 or 4 plots per screen
Listing	- List the data currently being displayed
Pstscript	- Switch between creating/not creating Postscript output
Same Plot	- Display the same data gain
Stop	- Terminate execution
==========	==========

Getting Started

When the code starts it identifies itself; press ANY mouse key to continue. The code next displays Photon, Major cross sections for hydrogen.

Selecting Exactly the Data you Want

From this point on you can select any options that you want, in any order that you want.

For example you can select,

Projectile by selecting: Photons, Electrons, Charged or Neutrons - Positrons are not yet implemented.

Target by selecting Z+1, Z+10, Z-1,Z-10, to quickly position to anyplace in the periodic table.

Data Type by selecting Major, Minor, Deposit and Range - Straggling is not yet implemented.

Once you have the data that you want on the screen you can interact with it to display it in exactly the form you want - you can set,

Scaling using Lin/Log X/Y
Points to decide whether or not you want to display points
Energy Range using Zoom X
Ratio to decide whether or not to display ratios
Grid to include more or less details
Units use barn-1/cm to select microscopic or macroscopic units
Character size using Bigger and Smaller

With these options you can customize each plot to exactly meet your needs.

Once you have the data you want you can obtain hard copy output,

Listing will list ALL of the data for the **Projectile/Target** that you are **presently** displaying. You can output data for as many projectile/target combinations that you wish. When the code finishes all listing output will be in a file named **EPICSHOW.LST**. Note, the listing includes ALL of the data over the entire energy range, not just the energy range you may have selected Zoom X; experience has shown that it is easier to edit the output listing of the data than to restrict output to a Zoomed energy range.

ColorDump will produce a color Postscript file of the entire screen that you are **presently** displaying. See below for details on implementing this option.

Pstscript will produce black and white Postscript files of the data that you display **in the future** - not what is on the screen when you select this option - everything that you display in the future, until you again select this option to turn it off. See below for details on using this option.

EPICSHOW.INP Format for Mixtures

The 1998 and later versions of EPICSHOW can calculate mixtures of materials, e.g., WATER. Mixtures are defined in a file named **EPICSHOW.INP**, which is a simple text file that you can edit to define your materials. The format of this file is,

Line	Columns	Format	Description
1	1-10	A10	Name of Mixture (displayed on plots)
2	12-22	E11.4	Density of Mixture (grams/cc)
3	1-11	I11	ZA of Constituent (ZA = 1000*Z + A)
	12-22	E11.4	Grams/cc of Constituent
	23-33	E11.4	Atoms/cc of Constituent

Line 3 can be repeated any number of times to define all of the constituents of a mixture. The definition of the mixture is terminated by a blank (not zero, blank) line; see the below example input.

For each material as input you can define grams **OR** atoms for **ALL** constituents, but you cannot use **BOTH** grams **AND** atoms in the same material definition. For example, for Water it is convenient to define it as 2 atoms of Hydrogen for each atom of Oxygen, i.e., define the Atoms of each constituent. In contrast for Steel you may wish to define the grams of each constituent. In all cases the Grams or Atoms of all constituents as input are relative and the sum will be normalized to the input Density of Mixture (grams/cc).

EPICSHOW.INP may contain definitions for any number of materials, one after the other. Each time you select the "Mixture" option the next mixture will be read from EPICSHOW.INP.

If any errors are encountered while reading EPICSHOW.INP, the current and all subsequent requests to use the Mixture option will be ignored. In this case, see the file EPICSHOW.LST for details of the error(s).

Below is an example EPICSHOW.INP file defining mixtures. In this case we define Water at a density of 1.0 grams/cc, composed of 2 atoms of hydrogen (1001) for each 1 atom of oxygen (8016) = H2O. This is followed by definitions of Hydrogen and Oxygen at the density that each is a constituent of Water at 1.0 grams/cc. This input can be used to first see the composite data for Water, and to then see how each constituent contributes to the composite data.

```
Water                                   Title - 10A1
           1.0000E+00                   Overall grams/cc   - 11X,E11.4
       1001            2.0000E+00       ZA,grams/cc,atoms  - I11,2E11.4
       8016            1.0000E+00
                                        (BLANK LINE TERMINATED MIXTURE)
Hydrogen
           0.11191
       1001            2.0000E+00
                                        (BLANK LINE TERMINATED MIXTURE)
Oxygen
           0.88809
       8016            2.0000E+00
                                        (BLANK LINE TERMINATED MIXTURE)
```

Available Combinations

Not all possible combinations of projectile and type of data are included in this system. For example, ranges are only included for electrons and light charged particles; not for neutrons and photons. Only deposition and range data, no cross sections, are included for light charged particles.

You will find that other combinations are also not included. If you try to select a combination that is not included the code will not respond. There is no warning - it simply doesn't respond - and you will quickly get used to this and move on to examine other data.

Atomic and Nuclear Data

This system includes atomic data for everything except neutrons. Atomic data is all elemental and is identified by the atomic number (Z) of the element, e.g., 1-H- Nat, means hydrogen (Z=1) with its natural elemental abundance (Nat). The Nuclear data for neutrons is for either a naturally occurring elemental mixture of isotopes, or for a specific isotope. Neutron data is identified by both atomic number (Z) and atomic weight (A), e.g., 92-U-238, means uranium (Z=92), isotope 238.

Microscopic vs. Macroscopic Units

All of the basic data in this system is in microscopic units per atom. Be aware that in defining macroscopic units the atomic weight and the density shown on the plot are used. STP elemental conditions are used in ALL cases. For gases the density of an element that you are using in your applications can be VERY DIFFERENT from STP elemental conditions.

Detailed Instructions

As stated above, the easiest way to learn how to you this code is to just start using it, i.e., click on any options and see what happens. Only a few of the options needs more detailed instructions; these include,

Postscript

Output starts when you select this option AND continues until you again select it to turn it off. If you do not use the option carefully you can end up with an ENORMOUS number of Postscript output files that you don't really want = a Postscript file for every single display that appears on your screen.

To avoid this problem I STRONGLY RECOMMEND: 1) Get the plot on the screen that you want to make a Postscript output of, 2) select Pstscript to turn the option ON, 2) select Same Plot - immediately below Pstscript - this will make another plot of what is currently on the screen - including Postscript output, 3) select Pstscript again to turn the option OFF. If you follow this recommendation Postscript output will only be turned on while you are producing the plots you want and you will end up with ONLY the Postscript output that you want.

When you finish running EPICSHOW you will find a series of Postscript files,

PLOT0001.ps
PLOT0002.ps

etc., one file for each plot. You can print these files on any Postscript printer and view them using any Postscript viewer, such as Ghostview. These files contain only plots of the data; not the options from the top of the screen.

Warning - Every time that you run EPICSHOW and generate Postscript output the code uses the same names for the files. So if you want to save any files from a previous run be sure that you rename then before running the code again.

ColorDump

The above applies to black and white output. For color Postscript output you can use the ColorDump option - only if you get on the Web - search for ImageMagik - and download a copy of the code import for your computer - and put import in the directory where you run EPICSHOW. If you do not have import EPICSHOW will run fine, but this option will be ignored.

Unlike the Pstscript option, ColorDump only applies to the screen currently being displayed - it only produces one screen dump and you need not worry about it staying on and producing more screen dumps.

When you finish running EPICSHOW you will find a series of Postscript files,

SCRN0001.ps
SCRN0002.ps

etc., one file for each plot. You can print these files on any color Postscript printer and view them using any Postscript viewer, such as Ghostview. These files contain a screen dump of the entire screen, including both options at the top of the screen and the data in the lower part of the screen.

Warning - Every time that you run EPICSHOW and generate Postscript output the code uses the same names for the files. So if you want to save any files from a previous run be sure that you rename then before running the code again.

Same Z and Same Plot

Both of these options do exactly the same thing = refresh the screen by repeating the last plot. Be aware that in 2 X 2 mode this will produce another sub-plot - not a plot of everything on the screen.

2 X 2

The code can display either one full screen plot, or four half size plots. The 2 X 2 option is used to switch back and forth between one and four plots per screen. The 2 X 2 option is convenient when you want to compare data all on the same plot. For example, if you want to compare similar data for four different targets, or all of the data for the same target, e.g., electrons major, minor, energy deposition and range. The possibilities are limited only by your imagination.

If you have a small screen and you use the 2 X 2 option you may have difficulty clearly seeing all of the data. However, on any system Postscript output using the 2 X 2 option will be clear and easy to read.

Warning - If you want a 2 X 2 Postscript output be sure the Pstscript option is ON before you display the first sub-plot on the screen.- remember that the Pstscript option will only produce Postscript output for data that you display in the future, after you have selected the Pstscript option. So if you are using the 2 X 2 option and select the Pstscript option after already plotting a few sub-plots on the screen, the Postscript output **WILL NOT** include the sub-plots that were on the screen before you selected the Pstscript option.

Freeze!!!

Normally as you select each option the screen will immediately respond. If you select Freeze!!! you lock the screen and it will not change until you again select Freeze!!! Once you have unlocked the screen the code will again respond to each command as you select it. For example, if you are at Z=1 and quickly want to go to the other end of periodic table: 1) select Freeze!!!, 2) select Z+10 a number of times - note, your position in Z is always defined on the top line of the screen next to the code identification, so you always know where you are - but nothing else on the screen will change, 3) select Freeze!!! This will release the screen and you can start displaying data.

Freeze!!! and 2 X 2 can be used in combination to get four sets of data from anyplace in the periodic table all on the screen at the same time. For example, try to get the photon major cross sections on the screen for Z = 3, 26, 82 and 94, all at the same time, and you'll see what I mean - using Freeze!!! and 2 X 2 it can be done.

WARNING - Freeze!!! cannot be used to queue a number of plots to display. It can only be used a change as many options as you want for the next plot that you request after you again select Freeze!!! to release the screen.

Points

All of the data included in this system are in tabulated form - no models are used to define data. If you want to see where the data points are tabulated use this option.

Ratio

When this option is used the first curve is considered to be a standard, and the ratio of all other curves to this standard is displayed. This is helpful if you want a quantitative estimate of the ratio of individual cross sections to the total.

Energy

This option only applies to photon data. Above it was stated that all data in this system is tabulated and models are not used. This is true up to 1 GeV. Above this energy analytical expressions are used to define all photon data.

Grid

If you want to try to read values off the plots it helps to add more details to plots. For example, even the first plot displayed is over 12 log decades in cross sections and it is hard a see specific cross section ranges. This is much easier if you add grid detail.

barn-1/cm

This option can be used to display data in either the original data units of barn (microscopic) units or 1/cm (macroscopic) units.

Be aware that in defining macroscopic units the atomic weight and the density shown on the plot are used. STP elemental conditions are used in ALL cases. For gases the density of an element that you are using in your applications can be VERY DIFFERENT from STP elemental conditions.

Character Size

Depending on the size of your screen you may want to make the characters larger or smaller. Also for 2 X 2 output you may want to make the characters larger.

Listing

The output listing for each combination of data is generally in three parts: 1) major data, 2) minor (detailed) data, and 3) macroscopic data. A portion of an example is shown below.

Warning - be aware that this output includes macroscopic derived data. Some users of this system have not checked their output listings in detail and have ended up wasting their time deriving macroscopic data that is already available to them in the output listing.

```
===================================================================================================
Atomic Weight 207.190                     82-Pb-Nat                              1.1350+01 grams/cc
                                  Electron Interaction Data
Energy    Cross Sections (barns)-----------------------------------Energy Loss per Collision (MeV)
(MeV)     Total     Ionize    Brems.    Excite    Elastic   Transport Ionize    Brems.    Excite
===================================================================================================
1.0000-05 2.6290+09 1.3106+08 4.8698+03 8.7576+06 2.4892+09  2.4892+09 6.6445-06 2.1615-06 6.2969-06
1.2500-05 2.4483+09 1.7847+08 5.0513+03 2.6756+07 2.2431+09 2.1032+09 7.2252-06 2.5863-06 6.7137-06
1.2589-05 2.4419+09 1.8016+08 5.0578+03 2.7452+07 2.2343+09 2.0895+09 7.2472-06 2.6018-06 6.7300-06


7.9433+02 5.0670+06 1.3190+06 1.6125+03 1.6149+06 2.1315+06 3.2035-02 3.8557-04 2.5925+01 1.0952-05
8.0000+02 5.0669+06 1.3190+06 1.6131+03 1.6148+06 2.1315+06 3.1720-02 3.8569-04 2.6105+01 1.0952-05
1.0000+03 5.0655+06 1.3189+06 1.6265+03 1.6145+06 2.1305+06 2.0621-02 3.8987-04 3.2463+01 1.0953-05
===================================================================================================
Atomic Weight 207.190                     82-Pb-Nat                              1.1350+01 grams/cc
                                  Electron Interaction Data
Energy    Ionization Shell Cross Sections (barns)--------------------------------------------------
(MeV)     K         L1        L2        L3        M         N         O         P         Q
===================================================================================================
1.0000-05                                                                       1.3106+08
1.2500-05                                                                       1.7847+08
```

```
1.2589-05                                                                                 1.8016+08


7.9433+02 3.2770+01 1.5391+02 2.0094+02 4.3601+02 1.1170+04 1.8337+05 7.6211+05 3.6151+05
8.0000+02 3.2787+01 1.5391+02 2.0094+02 4.3601+02 1.1170+04 1.8337+05 7.6211+05 3.6151+05
1.0000+03 3.3373+01 1.5392+02 2.0098+02 4.3608+02 1.1172+04 1.8337+05 7.6202+05 3.6148+05
====================================================================================================
Atomic Weight 207.190                               82-Pb-Nat                        1.1350+01 grams/cc
                                        Electron - Derived Quantities
Energy    Total Cross Section-Mean Free Path------Range-----Energy Loss ( Mev/cm)--------------------
MeV       barns     1/cm      cm        cm*cm/gr  cm        Total     Ionize    Brems.    Excite
====================================================================================================
1.0000-05 2.6290+09 8.6729+07 1.1530-08 7.6413+06           3.0547+01 2.8728+01 3.4724-04 1.8192+00
1.2500-05 2.4483+09 8.0768+07 1.2381-08 7.1161+06           4.8465+01 4.2539+01 4.3097-04 5.9259+00
1.2589-05 2.4419+09 8.0556+07 1.2414-08 7.0975+06           4.9167+01 4.3072+01 4.3411-04 6.0948+00


7.9433+02 5.0670+06 1.6715+05 5.9825-06 1.4727+04 2.8605+00 1.3964+03 1.6777+01 1.3791+03 5.8345-01
8.0000+02 5.0669+06 1.6715+05 5.9826-06 1.4727+04 2.8645+00 1.4065+03 1.6782+01 1.3892+03 5.8342-01
1.0000+03 5.0655+06 1.6711+05 5.9843-06 1.4723+04 2.9914+00 1.7594+03 1.6962+01 1.7418+03 5.8336-01
```

Using the Data in Your Applications

For neutron and photon applications it is recommended that you use the TART98 Monte Carlo transport code. This is a 3-D, Combinatorial geometry, time dependent, general purpose, Monte Carlo code that runs on any computer and is designed to satisfy the needs of a wide variety of applications. It uses the EPICSHOW neutrons and photon data and is now available Worldwide through local computer code centers. It is strongly recommended that if you have applications that only involve neutrons and/or photons you should use TART98 to solve your problems, rather than trying to re-invent the wheel by writing your own code.

If you prefer to do it yourself you can,

1) If you only need a few materials use EPICSHOW to retrieve the data you need to EPICSHOW.LST and edit it to meet your needs. The format (shown above), is simple tabulated fixed field data with each field ten (10) columns wide in FORMAT E format - WARNING - each field is defined by each 10 columns - not by tabs, blanks or anything else.

2) If you have more general needs use the original tabulated data distributed with the UNIX version of the code. There are a series of files named ???.TAB, where ??? defines the contents of the file, e.g., PHOTON.TAB, ELECTRON.TAB, etc. Each file contains all of the data of a given type for the entire periodic table. As with the EPICSHOW.LST output, these tables have headings describing each type of data, and the tables are in fixed field data in exactly the same 10 characters per field format as found in EPICSHOW.LST.

What Computers will EPICSHOW Run on?

This code has been implemented and used on UNIX, IBM-PC, Power MAC, and even Laptop computers. It is written in such standard FORTRAN and uses such a simple graphics interface that it should be relatively easy to use on any computer.

If you interface this code to a different type of computer it is recommended that you send copies of everything to the author for inclusion in future distributions of this code. This will save you time in the future.

Customizing the Code

All of the names for the options that appear at the top of the screen are in a file named PAGE.DAT. If you do not like the names I've defined, feel free to edit PAGE.DAT to include the names that you prefer. Be VERY CAREFUL not to change the format of this file; if you do, the results obtained running EPICSHOW can be unreliable.

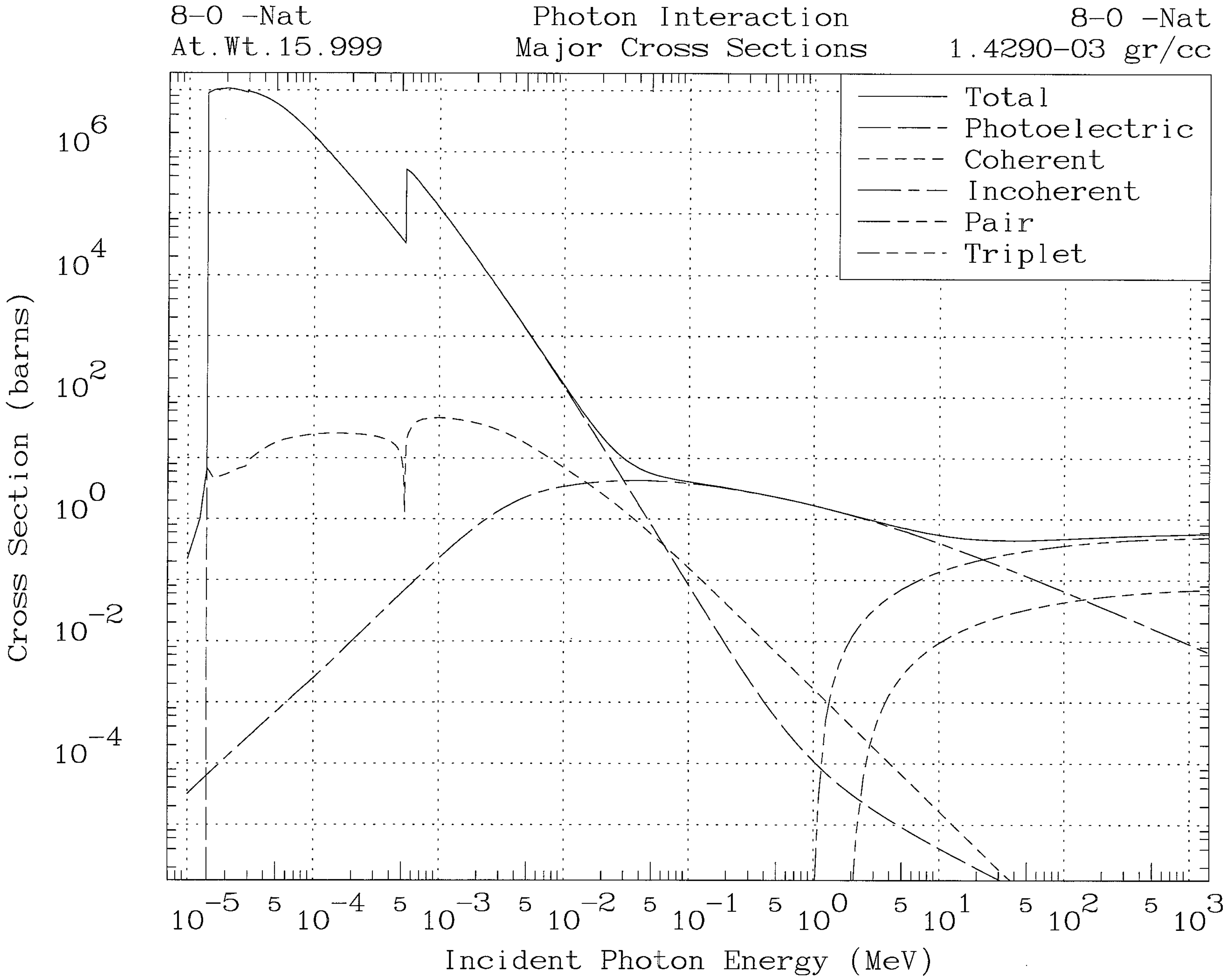
Code Installation

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.

When you receive this code you will find a file named **README** - be sure that you read this file as you proceed with installation.

Example EPICSHOW Output

When you first run EPICSHOW take a look at the oxygen photon cross sections. This data is related to the results shown in the **TARTCHEK** update documentation (TARTCHEK.DOC) where energy deposition in a water phantom is discussed. There the question was asked: what's the difference in the energy deposition due to 1 versus 10 MeV photons. From a plot of the oxygen cross sections we can see that the total cross section at 1 MeV is considerably larger than at 10 MeV, so we expect more of the photons to collide and deposit their energy at 1 MeV compared to 10 MeV. That's part of the picture; the rest is that the lower energy photons will scatter through larger angles and spread out more from the incident photon beam.


8-0 -Nat
At.Wt.15.999
Photon Interaction
Major Cross Sections
8-0 -Nat
1.4290-03 gr/cc
Total
Photoelectric
Coherent
Incoherent
Pair
Triplet
Cross Section (barns)
10^{6}
10^{4}
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-5} 5 10^{-4} 5 10^{-3} 5 10^{-2} 5 10^{-1} 5 10^{0} 5 10^{1} 5 10^{2} 5 10^{3}
Incident Photon Energy (MeV)

PROGRAM PLOTTAB: A Code Designed to Plot Continuous and/or Discrete Physical Data
(VERSION 98-1)

Abstract

PLOTTAB is designed as a general purpose plotting utility code to plot continuous and/or discrete physical data for use in almost any application. It is designed to be easily used by your application codes to produce your output results in a form that can be immediately used by PLOTTAB to allow you to see your results.

It produces on screen graphics as well as Postscript formatted output files that can be viewed or printed on any Postscript printer. The code is designed to be easily used on any computer - not only today's computers, but also anything that comes along in the future. So you can be assured that once you start using PLOTTAB your graphics problems are over - not just today, but well into the future.

Introduction

PLOTTAB is designed as a simple plotting code that can be used on virtually any computer and graphics device to plot continuous and/or discrete physical data for use in almost any application. It is designed to be easily used by your application codes to produce your output results in a form that can be immediately used by PLOTTAB to allow you to see your results.

It produces on screen graphics as well as Postscript formatted output files that can be viewed or printed on any Postscript printer. The code is designed to be easily used on any computer - not only today's computers, but also anything that comes along in the future.

Only Do the Job Once

Best of all you can be assured that unlike other computer graphics codes that quickly come and go, PLOTTAB will be here not only today to meet your needs, but also into the future to meet your needs. The code has been conservatively designed to not only run on virtually any of today's computers, but also be to easily implemented on any new computers that come along in the future. I have now been using PLOTTAB and its predecessors for almost 20 years, and during that time as each new computer has come along PLOTTAB has been a complete plug-in that has smoothly and effortlessly moved from one computer to the next.

With this approach you can be assured that once you start using PLOTTAB your plotting problems are over, not only for today, but also into the future. You only have to do the job once to modify your codes to produce output in the PLOTTAB input format, and then you can be assured that you will be able to produce graphic output well into the future.

Concentrate on Your Applications

You will find that in order to use PLOTTAB you do not have to be a graphics expert, nor do you have to spend much time learning how to use the code. This allows you to concentrate on your applications, instead of worrying about how you are going to plot your results. Once you have started using PLOTTAB you can take graphics output for granted; something that you never have to worry about or spend much time on ever again. It will simply always be there when you need it.

What Computer Can You USE?

The non-interactive version can be run on ANY computer. The non-interactive code is written in standard FORTRAN and outputs standard formatted Postscript files that can be printed on any Postscript printer, or viewed with any Postscript viewer, such as Ghostview.

The interactive version can be run on UNIX, IBM-PC, Power MAC and even Laptop computers. The interactive code is identical to the non-interactive code, except that a very simple graphics interface for on screen graphics is loaded with the code. The code is distributed with graphic interfaces for UNIX, IBM-PC, and Power MAC. However, it should be very simple to interface this code for on screen graphics on any computer, e.g., it took me 3 days to write the IBM-PC interface and 2 days for the Power MAC version.

Reading and Interpreting Data

All data is read by this code in character form and internally translated into integer or floating point form as needed. This means: 1) it's difficult to get the code to crash by improperly defining input. 2) your input can be in quite general form and will still be properly interpreted by the code, e.g., 14, 1.4+1, 14E+00, 1.4D+01, are all 14 as far as this code is concerned. 3) the code can distinguish between **BLANK** and **ZERO** input - **WARNING** - when this documentation says **BLANK**, it means **BLANK** - in this case nothing else will be interpreted correctly as input.

Data Formats

The formats of the continuous and discrete physical data read by this code are designed to be very simple, so that any of your computer codes can be simply modified to produce output results in the PLOTTAB input format.

Continuous Data Format

The continuous data includes a one line title, followed by a series of (x,y) coordinates, one per line. Each "curve" of continuous data is terminated by a **blank** line. One curve can be followed by another, starting with the one line title, another followed by a series of (x,y) coordinates and terminated by a **blank** line. The input to this code can include any number of such "curves", one curve after the other in the continuous data input file

PLOTTAB.CUR. Each pair of (x,y) coordinates are in fixed fields each 11 columns wide, corresponding to **FORTRAN 2E11.4 format**. For example,

```
Example Curve # 1
 17       43.0
 19       37.0
       .
       .
 71       12.9
                    (BLANK LINE ENDS CURVE)
Example Curve # 2
       .
       .
```

WARNING - again, I'll stress that a **BLANK LINE** means **completely blank** in the first 22 columns - **NOT zero**, which is considered by the code to be perfectly legal input as part of a "curve".

Discrete Data Format

The format of the discrete data is very similar to the continuous data. Each set of discrete points starts with a one line title, followed by a series of points, and ends with a **blank** line. Each point is defined by an (x,y) coordinate plus uncertainties in both x and y - each point is defined by six values: x, -dx, +dx, y, -dy and +dy, one point per input line. The input to this code may include any number of sets of discrete points, one set after the other in the discrete data input file **PLOTTAB.PNT**. Each point of six values is in fixed fields each 11 columns wide, corresponding to **FORTRAN 6E11.4 format**. For example,

```
Example Set # 1
 17       1.2      2.4   43.0     17.2   12.1
 19       1.6      2.6   37.0     15.8    9.3
       .
       .
 71       8.2     10.7   12.9      7.2    2.3
                    (BLANK LINE ENDS CURVE)
Example Set # 2
       .
       .
```

Note, uncertainties -dx, +dx, -dy and +dy are always interpreted as positive (sign ignored) in the same units as the data (not % or fraction or anything else), and they are interpreted x +/- dx - not a minimum, average and maximum, e.g., for the first x value 17 1.2 2.4, means an average value of 17 with errors extending -1.2 below 17, and +2.4 above 17 - there are no implied units - everything is absolute - below you will learn how to physically interpret and identify your data.

Note, this format allows for zero, or blank, error fields, as well as asymmetric errors. If the errors are symmetric you must define both of them separately.

WARNING - again, I'll stress that a **BLANK LINE** means **completely blank** in the first 66 columns - **NOT zero**, which is considered by the code to be perfectly legal input as part of a set of discrete points.

Interpretation of the Data

When you generate data in these formats using one of your application codes you do not have to decide in advance how they will actually be interpreted and appear on a plot. This is controlled by an input file named **PLOTTAB.INP**. This file defines how many curves and/or sets of discrete points will appear on each plot, allows for a two line title to appear at the top of each plot, x and y axis labels and scaling (linear or log scaling), and x and y ranges (in case you do not want to plot all of the data on a single plot). Using these options you can customize each plot to exactly meet your needs. As in the case of the continuous and discrete point files, almost all input fields in **PLOTTAB.INP** are 11 columns wide corresponding to **FORTRAN E11.4 or I11 format**.

How Do You Produce Output

Let's first see how simple it is to update any of your codes to produce output that can be read as input to PLOTTAB. Assume that you have a code that is calculating lots of results that until now you have had to wade through by hand to see what your results mean. Here's what you add to your code to output the results in a form that PLOTTAB can read as input,

```
C-----LOOP OVER CURVES
      DO ICURVE = 1,NCURVE
C-----PRINT FIRST LINE TITLE
      WRITE(16,1600) TITLE(ICURVE)
 1600 FORMAT(A40)
C-----PRINT DATA POINTS
      DO IPOINT = 1,NPOINT(ICURVE)
      WRITE(16,1610) X(IPOINT,ICURVE),Y(IPOINT,ICURVE)
 1610 FORMAT(2E11.4)
C-----END OF POINT LOOP
      ENDDO
C-----PRINT BLANK LINE FOR END OF CURVE
      WRITE(16,1620)
 1620 FORMAT(30X,'(BLANK LINE ENDS CURVE)')
C-----END OF CURVE LOOP
      ENDDO
```

That's all you have to add to your application codes. We just output any number of curves (defined by **NCURVE**), each with its own title line to identify it (defined by **TITLE**), each with any number of points defining each curve (defined by **X, Y**), and each curve ended with a **blank** line. What could be simpler? The above example is for outputting continuous curves, but outputting discrete sets of points is no more difficult. The only difference in the above coding would be where we output two numbers for each (x,y) point, for discrete points we would output six numbers to include the x and y uncertainty in the order x -dx +dx y -dy +dy.

That's it!!! You are now an expert at producing output that can immediately be read by PLOTTAB to show you your results. That's all you need to know about producing output. Once you start using this very simple approach never again you will have to wade through piles of results trying to figure out what they mean. Instead you can immediately see your results, and you will see them on plots that are of high enough quality to be accepted by journals for publication without any modifications.

Interpreting and Plotting Your Data

Let's assume that you have now produced some output results that you want to plot, i.e., you have your results in the correct format, described above, in a file named **PLOTTAB.CUR** for continuous curves and in **PLOTTAB.PNT** for discrete points. First let me make it clear that you don't need both; you can plot continuous and/or discrete data in any combination. For the following example I'll assume both merely so that I can discuss how to control both.

Now we will edit the control file **PLOTTAB.INP** to tell PLOTTAB how to interpret and plot your data. Below is an example **PLOTTAB.INP** that I recently used to produce a plot. At first this may look complicated, but let me point out that although I have now been using PLOTTAB and its predecessors for almost 20 years, I still have only one **PLOTTAB.INP** and every time I want to produce a plot all I do is edit a few things and bingo! Out come the plots I want. This is the approach that I suggest you also use - don't start from scratch - start with the PLOTTAB.INP file distributed with the code and modify it to meet your needs.

```
    0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
```

Let me first cover the input things that I change for each new plot that I want. Above are eight (8) lines from PLOTTAB.INP that are used to produce one plot and I'll discuss the important parameters from top to bottom. I've highlighted the parameters I will discuss so that you can more easily find them.

The first important thing to tell PLOTTAB is how many continuous curves and/or discrete sets of points to put on the plot. This is done on the second line, where my input says to display **3** continuous curves and **2** sets of discrete data. Regardless of how many curves and sets of points you have produced in PLOTTAB.CUR and PLOTTAB.PNT you can use these 2 input fields to tell PLOTTAB how many to actually read and put on the next plot. For example, I can use the above input to produce a plot with 3 curves on the plot, or I can easily change the input (as we will see below) to display the curves one at a time on a series of plots.

Next we want to physically describe what data we are plotting. This is done using the next four (4) input lines. These are,

1) A label for the x axis - in this example **`Neutron Energy (MeV)`**
2) A label for the y axis - in this example **`Cross Section (barns)`**
3-4) A two line title to appear at the top of the plot - in this example

```
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
```

In most cases that's all I modify - the code takes care of everything else - so I can immediately run PLOTTAB and get the plots I want. That's all you need to know to successfully use PLOTTAB to generate plots for you.

Note, with this approach the code does not have to have any idea what the physical significance of the data is, and any data can be put into the PLOTTAB input format and plotted. Physical interpretation of the data is all in your hands - by changing plot titles and axis labels you are free to interpret the data any way that you see fit - and you can easily produce plots to specifically meet your needs.

Let me briefly cover the meaning of the other input fields, just in case you want to get fancy.

The first input line, again shown below, defines the (x,y) dimensions of the plot, how many sub-plots to put on each plot, and how large to make the characters. Reading left to right the below line says the plot size is x=0 to 13.5, and y=0 to 10. This will give you a full page plot on 8-1/2 by 11" paper, and except for special purpose you won't want to change this. One case in which you will want to change them is if your plotter doesn't use inches, but rather use cm, mm, anything - no problem - just change these once for your system and you probably will never have to change them again. The next 2 fields say that each page will contain 1 plot in the x direction and 1 y in the y direction = one single, full page plot. If I would like 6 plots on each page I could change these to 3 plots in the x direction and 2 in the y direction. The last field, 1.5, says to make the characters 1-1/2 times normal size.

```
    0.00000    13.50000    0.00000    10.00000          1          1 1.5
```

On the next line you already know about the first two fields. The remaining fields, from left to right, allow you to:
1) add a border around each plot
2) add an (x,y) grid - the code has 6 built-in grids
3) plot the ratio of everything to the first curve
4) change the thickness of all lines drawn

```
          3          2          0          0          0          0
```

You already know about the next four lines, so all we need discuss is the last two lines. These two lines are used to control the x and y features of the plot: the first line is for x and the second for y. Again from left to right, the six (6) fields on each line control,

1) the lower x limit
2) the upper x limit
3) should discrete point x errors be plotted = no(0), yes (1)
4) x scaling = automatic (0), linear (1), log(2)
5) round x limit to avoid touching edge of plot = yes(0), no(1)
6) show legend box = yes(0), no(1)

1) the lower y limit
2) the upper y limit
3) should discrete point y errors be plotted = no(0), yes (1)
4) y scaling = automatic (0), linear (1), log(2)
5) round y limit to avoid touching edge of plot = yes(0), no(1)
6) show data points on curves = no(0), yes(1)

```
                                            0          0          0          0
                                            0          0          0          0
```

WARNING - let me again stress that this code can tell the difference between **BLANK** and **ZERO** - for example, on the above two input lines the **BLANK** x and y lower and upper limits means scale the plot to show all of the data as read. In contrast, if one of these fields contained **0.0** the specified limit would be forced to be zero, regardless of the range of the data read.

Again, that's it!!!If you want to get fancy you can use these parameters to customize plots to obtain almost any output that you want to meet any specific need.

Multiple Plots

Let's now generalize to more than one plot. Don't worry there isn't much to generalize. For more than one plot you basically have two options:

1) If the layout of each plot is the same you need merely copy the last four lines as many times as you want, changing any parameters that you want on these lines. You can do this as many times as you want for as many plots as you want. For example here's a generalization of our above input for three plots - in this case all I did was copy the last four lines twice and change the titles for Li^6, Al^{27} and U^{238} - this assumes that in PLOTTAB.CUR I have 3 curves for each plot (at least 9 curves), and that in PLOTTAB.PNT I have 2 sets of discrete points for each (at least 6 sets).

```
    0.00000   13.50000    0.00000   10.00000          1          1 1.5
          3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
Aluminum-27 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
Uranium-238 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
```

2) If the layout of each plot is different you can follow the eight (8) line input for one plot by a **BLANK** (not 0, BLANK) line, and then add eight lines for the next plot. You can do this as many times as you want for as many plots as you want. For

example here's a generalization of our above input for three plots using this second method - in this case the input is EXACTLY equivalent to what I have shown above using the first method - use either method, or a combination of the two, in any order that you want - the choice is your's.

```
0.00000   13.50000    0.00000   10.00000         1          1 1.5
         3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Lithium-6 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0

    0.00000   13.50000    0.00000   10.00000          1          1 1.5
         3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Aluminum-27 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0

    0.00000   13.50000    0.00000   10.00000          1          1 1.5
         3          2          0          0          0          0
Neutron Energy (MeV)
Cross Section (barns)
Uranium-238 Major Cross Sections
From the Livermore ENDL Cross Section Library
                                0          0          0          0
                                0          0          0          0
```

You might wonder why I wrote out the names instead of just using Li^6, Al^{27} and U^{238} - if you prefer this form, no problem - see the documentation within the code on how to use sub and super-scripts, as well as Greek characters and other interesting goodies that I cannot cover in detail in a brief introduction.

Data is Read in Order

In controlling the flow of curves from PLOTTAB.CUR and sets of discrete data points from PLOTTAB.PNT remember each time you use input in PLOTTAB.INP to tell it to read data from these files the code continues to read further and further into each file. For example, in the above case the first plot will contain the first three curves from PLOTTAB.CUR and first two sets of points from PLOTTAB.PNT. The second plot will contain curves four through six from PLOTTAB.CUR and sets of points three and four from PLOTTAB.PNT, etc. It's your responsibility to insure that you have properly ordered the curves and sets of points, and properly arrange their grouping on successive plots.

Terminating Plotting

Execution terminates when the are no more requests for plots in PLOTTAB.INP, or no more data to plot from PLOTTAB.CUR and PLOTTAB.PNT. Note, regardless of how many curves and set of points you have in PLOTTAB.CUR and PLOTTAB.PNT only as many plots will be produced as you define in the control file PLOTTAB.INP. Also if one input stream of data (curves or sets of points) is exhausted, but the other isn't, the code will continue producing any plots that you request. For example, for the above three plots if there are 9 curves in PLOTTAB.CUR, but only 2 sets of points in PLOTTAB.PNT, the first plot will contain 3 curves and 2 sets of points (which exhausts the sets of points), and the following two plots will each contain 3 curves and 0 sets of points.

Editing Files

All of the files PLOTTAB.CUR, PLOTTAB.PNT and PLOTTAB.INP are simple text files, so you can use any editor to edit them. For example, after you have produced output in PLOTTAB.CUR if you can want to change anything just open the file and do it. You can change titles for the curves, deletes curves, rearrange the order of curves, anything you want to do to meet your needs.

How Do You Define Input Parameters Before You Have Seen ANY Plots

It very nice to have all of these options to select x and y scaling, x and y ranges, etc. But how can you possibly know what options to select before you have seen plots of you data? The answer is: you don't! PLOTTAB is supplied in two versions: an interactive version that only produces on-screen output, and a non-interactive version that only produces Postscript hardcopy output.

What I recommend is that you start by not selecting any special options and first run the interactive version. This version will allow you to interactively select many of the options described above, e.g., x and y scaling, x range, etc. Then after you have seen your data and played with it you can decide what options you want to set to produce your final Postscript output.

As I use PLOTTAB well over 90 % of the plots that I look at are generated by the interactive code and I never even bother to generate Postscript output. In this way PLOTTAB can be used very simply to quickly look at enormous amounts of data. In this case I don't care what the x and y axis labels are or what the two lines at the top of the plot say - I know how to physically interpret the data I'm looking at - so I just use my existing PLOTTAB.INP file and all I have to tell it is how many curves and sets of discrete points to display on each plot, and I quickly start looking at my results.

Only when I see something of interest that I need a hardcopy of do I bother making any of the changes I have described above. If you also use this pragmatic approach you can save yourself a lot of time and energy and use PLOTTAB much more effectively in your work.

How Do You Remember ALL of the Options

You don't - or at least, I don't. Whenever I want to find out what a given field in the PLOTTAB.INP input parameters means I run PLOTTAB, immediately kill it (use

CONTROL C) and then look in the output file PLOTTAB.LST that contains an interpretation of all of the input parameters. For example, for the following input,

```
    0.00000    13.50000     0.00000    10.00000          1           1 1.5
          3           0           0           1          0           0
Neutron Energy (MeV)
Production, Absorption and Leakage
Production, Absorption and Leakage
For Test Problem
                                  0          -2          1           0
                                  0          -2          1           0
```

Here's the interpretation of the above input from the output file PLOTTAB.LST. As you can see there is a line by line and field by field interpretation of the input parameters in exactly the order they appear in the input. For example, if I can't remember which input field controls plotting ratios, from the below listing I can see that it is the fifth field of the second input line.

```
=======================================================================
 PLOT TABULATED DATA (PLOTTAB VERSION 97-1)
=======================================================================
 DESCRIPTION OF PLOTTER AND FRAME LAYOUT
 ----------------------------------------------------------------------
 X DIMENSIONS (X-MIN TO X-MAX).....   .00000+ 0 TO   1.35000+ 1
 Y DIMENSIONS (Y-MIN TO Y-MAX).....   .00000+ 0 TO   1.00000+ 1
 PLOTS PER FRAME (X BY Y)..........           1 BY           1
 CHARACTER SIZE MULTIPLIER.........       1.500
=======================================================================
 READ AND PLOT (FOR EACH PLOT).....     3 CURVES
 SETS OF POINTS PER PLOT........... NONE
 SHOULD BORDER BY PLOTTED.......... NO
 TYPE OF GRID......................DASHED GRID (COARSE)
 SHOULD RATIOS BE PLOTTED.......... NO
 LINE THICKNESS...................   0
 ----------------------------------------------------------------------
 X AXIS LABEL AND UNITS............ Neutron Energy (MeV)
 Y AXIS LABEL AND UNITS............ Production, Absorption and Leakage
 ----------------------------------------------------------------------
 ----------------------------------------------------------------------
 PLOT TITLE
.......................................................................
 Production, Absorption and Leakage
 For Test Problem
 ----------------------------------------------------------------------
 REQUESTED X RANGE................. PLOT ALL POINTS
 PLOT X ERROR BARS................. NO
 X PLANE ON PLOTS (IF POSSIBLE)....LOG (NO INTERPOLATION)
 ROUND X LIMITS.................... NO
 LEGEND BOX ON PLOT................ YES
 ----------------------------------------------------------------------
 REQUESTED Y RANGE................. PLOT ALL POINTS
 PLOT Y ERROR BARS................. NO
 Y PLANE ON PLOTS (IF POSSIBLE)....LOG (NO INTERPOLATION)
 ROUND Y LIMITS.................... NO
 SHOW DATA POINTS.................. NO
 ----------------------------------------------------------------------
 X LIMITS (PLANE)..................  1.02360-10 TO   1.99760+ 1 (LOG)
 Y LIMITS (PLANE)..................  2.02610- 5 TO   1.57230+ 7 (LOG)
 ----------------------------------------------------------------------
 CONTINUOUS CURVES
.......................................................................
```

```
 INDEX POINTS DESCRIPTION
.................................................................
     1    566 Production
     2    566 Absorption
     3    498 Leakage
```

Postscript Output Files

When you run the non-interactive version of the code it will produce a series of Postscript output files, one plot per file. The files will be named,
PLOT0001.ps
PLOT0002.ps
.
.

These postscript files can be printed on any Postscript printer, or viewed with any Postscript viewer, such as Ghostview; note, Ghostview is available FREE on the web.

WARNING - every time you run the code it uses the same file names. So if you want to save any file make sure you rename it before you again run the code.

Interpolation Along Curves

When you start using PLOTTAB you will find that your results can look quite different depending upon how you display the data, e.g., linear or log scaling in the x or y direction.

By default PLOTTAB assumes that between tabulated points of curves it should interpolate assuming linear interpolation in x and y. If your data is not linearly interpolable you can get some very strange looking results when PLOTTAB interpolates in say log/log x/y scaled plots. PLOTTAB ALWAYS uses the defined interpolation to show the TRUE shape of each curve between tabulated points in each possible combination of linear and log x and y scaling. For example, if I have only two tabulated points at x=1 and x=100, using standard linear interpolation this will be a straight line on a linearly scaled x and y plot. But if you use log x and/or y scaling PLOTTAB will display a curved line corresponding to the TRUE (assuming linear interpolation) shape of the curve.

You can control how PLOTTAB interpolates in x and y separately by defining the interpolation to be linear or log in each dimension. Above I briefly described how to control the x and y scaling of each plot = automatic (0), linear (1), log(2). In this case, any non-negative values specify scaling with **linear** interpolation. To control interpolation, any negative values specify scaling and the same type of interpolation; the only meaningful negative value is -2 = **log** scaling and interpolation.

An important point to note, is that generally if you are using tabulated data in your applications it is important how the data is interpreted not only at the points that you tabulate data, but AT EVERY SINGLE POINT along the curve, e.g., integrals and interpolate values depend crucially on EXACTLY HOW YOU INTERPOLATE.

This PLOTTAB option can supply you with value information that you can use in your applications. For example, if you assume your data is linearly interpolable, but you see funny bumps and cusps in the results when you plot it in various combinations of linear and log, x and y scaling, you better think again about your assumption, because your data is not smoothly linearly interpolable. In this case you should consider either adding more, closer spaced points along your curves, or try PLOTTAB's log interpolation and see if this solves your problem. It ONLY SOLVES YOUR PROBLEM if you are willing to interpret your data in your applications using more complicated log interpolation. What should you do? The choice is yours. But whatever you do, be sure that you use PLOTTAB to check your final results to insure that you are not incorrectly interpreting your data in your applications.

You are NOW a PLOTTAB Expert

Sorry, that's about all I can quickly teach you about PLOTTAB. If you now understand how to produce output from your codes to be used as input to PLOTTAB, and how to edit PLOTTAB.INP to define the layout of each plot, you are now an expert, and you can immediately start generating plots. If you want to get even fancier, see the following documentation for more details - there's a lot more that this code can do to meet your specific needs than I have been able to cover in this brief introduction.

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.

When you receive this code system you will find it arranged in a file directory structure. At each level of the directory you will find a file named **README** - be sure that you read all such files as you proceed with installation and testing.

Additional Documentation

This report is designed to be used on-line and as such has been restricted to simple text in Microsoft Word 5.1 format. In particular a minimum of graphic output is included; only one simple example is included below. For additional documentation, including many example plots see the earlier documentation,

PROGRAM PLOTTAB: A Code Designed to Plot Continuous and/or Discrete Physical Data, by Dermott E. Cullen, UCRL-ID-110240, (March 1992), Lawrence Livermore National Laboratory

Latest Documentation

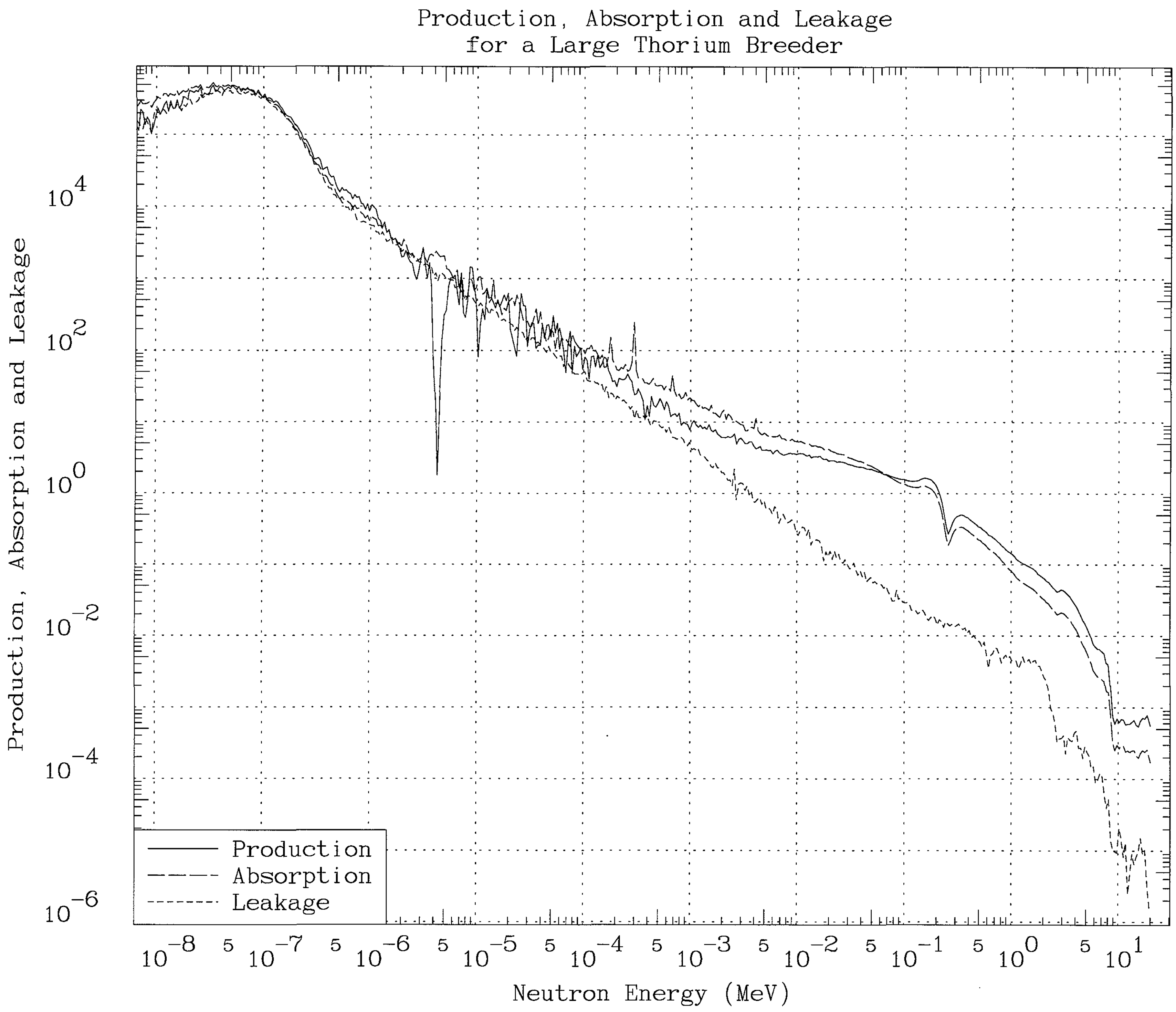
This code is designed to be self-documenting, in the sense that the latest documentation for the code is included as comment lines at the beginning of the code. Printed documentation, such as this report, is periodically published and consists mostly of a copy of the comment lines from the beginning of the code. The user should be aware that the comment lines in the code are continuously updated to reflect the most recent status of the code and these comment lines should always be considered to be the most recent documentation for the code and may supersede published documentation, such as this report. Therefore users are advised to always read the documentation within the actual code.

Example PLOTTAB Problem

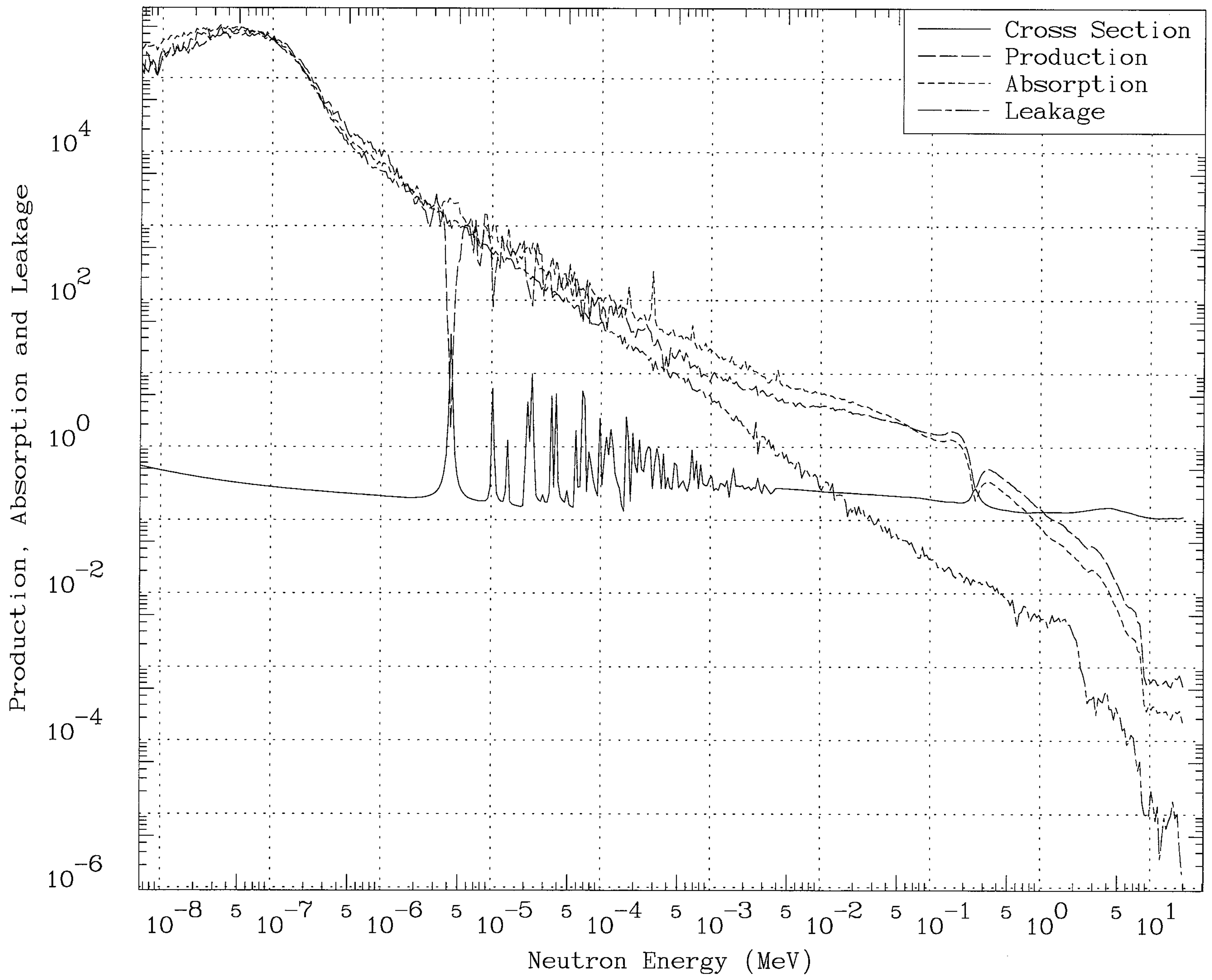
When you run the PLOTTAB test problem you will see the two figures discussed here. After running TART98 the utility code BALANCE was used to read the TART98 results and output them in the PLOTTAB format. The first figure you see when you run PLOTTAB illustrates the results for the neutron balance of the system: neutron production, absorption and leakage. The results, from higher to lower energy, show a fission spectrum, slowing down spectrum and finally a thermal Maxwellian. Based solely on this figure it is difficult to understand some features of the spectrum, such as the minimum in production at a few eV. Why is there is minimum and why isn't there also a minimum in the absorption at this point? This is explained below.

By adding the total cross section to the second figure, when you run PLOTTAB you can see that the minimum in the production corresponds to a maximum in the cross section due to a capture resonance, i.e., classical self-shielding where an increase in cross section results in a corresponding decrease in flux so that the product is about constant. How do I know this is a capture resonance? Note, the production, but not the absorption has a minimum. Self-shielding theory tells me that since absorption is smooth the product of the flux and absorption cross section must be almost the entire reaction rate. In contrast the minimum in production tells me the fission cross section is smooth, so that the product of the reduced self-shielded flux and smooth cross section results in a minimum in production.

Hopefully these results illustrate how the TART98 system codes can be used in combination to produce not just numbers, but rather physical understand. By using TARTCHEK to check input, TART98 to run your calculations, the utility codes to process the results, and PLOTTAB and TARTCHEK to immediately show you your results, you can save a great deal of time and learn a lot more than you ever will by wading through piles of output. For example, in this simple case the TART98 output file was over 32,000 lines long. But I learned everything I wanted to know about this system without ever having to look at this output. In addition you have EPICSHOW to independently check results compared to the cross sections TART98 is using.


Production, Absorption and Leakage
for a Large Thorium Breeder
Production, Absorption and Leakage
10^4
10^2
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8} 5 10^{-7} 5 10^{-6} 5 10^{-5} 5 10^{-4} 5 10^{-3} 5 10^{-2} 5 10^{-1} 5 10^0 5 10^1
Neutron Energy (MeV)
Production
Absorption
Leakage

Production, Absorption and Leakage
for a Large Thorium Breeder



Red's NATURAL Editor
A Program Designed to Edit FORTRAN Programs
(Version 98-1)

Abstract

This program allows the user to edit files using a completely natural method. You don't need a lot of time to learn how to use this program; it's as simple as using a typewriter. Even while you are reading this report you can start editing files.

Although it is as simple as using a typewriter, this is a full screen editor, so that you can overwrite (replace) anything, delete or insert characters or lines, copy or move lines, find or change anything. All of this can be done without using any complicated combinations of key strokes (as you are forced to memorize with other editors).

This program is written in standard FORTRAN. Although it was designed and initially implemented on an IBM-PC, it can be easily adapted for use on any graphics terminal, and now been implemented on UNIX workstations, IBM-PC, PowerMAC, and even Laptop computers; it is distributed with complete graphics interfaces for all of these types of computers.

Full Screen Editor

If you want to edit a file simply position the cursor to where you want to start editing and start typing - there are no control characters or anything else to remember - what could be more natural. If you want to replace any text in the file simply type over it. If you want to insert text in the middle of a file press the INSERT key and then start typing. If you want to delete a character simply press the DELETE key. It's that simple and natural.

Specific to FORTRAN

There are now many much more powerful editors available. What makes this editor useful is that it uses FORTRAN syntax rules. For example, if you want to change a variable named X to Y, you can do it without worrying about anything except a FORTRAN variable named X being changed to Y. It won't change a variable X1, nor will it change 5X in FORMAT statements. Also the layout of the screen forces you to be constrained to simple FORTRAN rules that work on any computer.

Requirements

The program is written in standard FORTRAN that can easily to adapted to run on virtually any computer. To use this code you need,

1) At least 8 megabytes of memory
2) A keyboard
3) A color screen is highly desirable, but not required.

Limitations

This program is primarily designed to edit FORTRAN programs. As such its major limitation is that each line may not be longer than 72 characters; if any lines exceeds this length, only the first 72 characters will be read and output when you close the file.

Although this program is primarily designed for use with FORTRAN programs, it is a general editor that can be used to type letters or reports or create tables of data (see, the below section on tabs, which can be used to simplify creating tables).

WARNING - the only real limitation is 72 character per line; this is sufficient for many applications. However, you should be WARNED that if you open an existing file that has more than 72 character on any line, all characters past the 72-th will NOT be read and if you then SAVE the file these characters will be lost. Before editing a file it is always a good idea to first make a backup copy. If you do this you can always recover your original file should you truncate longer lines, or inadvertently make incorrect changes to a file.

As distributed the program is limited to handle files of up to 100,000 lines in length. If you need to edit longer files and you have more memory available the program can be easily modified to edit larger files. If you do not have 8 megabytes of memory available and only edit smaller files you can easily modify the program to use less memory to only edit smaller files.

Minimum File Size

This program will minimize the disk storage requirements for your files. When files are written to disk each line will be output only up to the last non-blank character on the line. If a line is completely blank only one blank character will be output. Using this convention you may be surprised at how much your disk storage requirements can be reduced.

Screen Layout

The screen layout is 30 lines by 80 columns. The definition of the lines are,

1	= Program identification to LEFT and messages to RIGHT
2	= HOME to LEFT, number of lines in file being edited to RIGHT
3-30	= Up to 28 lines of text from the file being edited

You should watch the messages to the RIGHT on line 1. If you are not doing anything special this message will continuously remind you how to get on-line HELP - H in Home for HELP. If you are INSERTING, COPYING, MOVING or DELETING this message will remind you what you are doing, e.g., COPY Mode on ON. It is particularly important to watch this message if you make a mistake - Syntax ERROR - Command IGNORED. In this case you can either try the command again or use the on-line HELP to review the available commands and then retry.

The definition of columns are,

1- 6	= Sequence number of lines from the file being edited
7-78	= Up to 72 columns of text from the file being edited
78-80	= Not used

Identifying Modified Text

On a color screen for an original line from a file the sequence number will be RED and the text will be WHITE on a black background. If you modify any line the sequence number will be YELLOW in all cases. Any text that you modify will be shown in a different color than WHITE as long as it stays on the screen. If you move to a different page of text all lines than have been modified will still be indicated by a YELLOW sequence number, but the text will be WHITE.

Case Sensitivity

Text to actually be included in a file is case sensitive, i.e., the program distinguishes between upper and lower case letters. The same is true of the character strings used in find or change commands.

Commands to the program are NOT case sensitive. For a find command you can type either f or F, for a change command either c or C; similarly for all of the control commands.

Summary of What you have to know

The following sections describe in detail what you have to know in order to use this editor. What you have to know is,
1) How to open, save and close files that you want to edit
2) How to position yourself anywhere in a file
3) How to edit a file by replacing, inserting or delete characters
4) How to edit a file by copying, moving or deleting lines
5) How to edit a file by finding or changing character strings

That's basically everything that you need to know. It is so easy to use this editor that it is recommended you now start the editor running and as you read the following sections try each option as it is described. If you follow this recommendation you will find that by the time you have finished reading this report you will be familiar enough with the editor to feel comfortable editing files. If you need a quick reminder of any program option simply use the on-line HELP facility while you are running the program.

REMINDER - If you follow this recommendation to now start the editor you can open and edit any text file. To learn the options in this editor feel free to make any changes to the file that you want. When you are finished if you do not want to SAVE these changes (i.e., you do not want to really modify the file on disk) use the ESCAPE option described in the next section.

Opening, Saving and Closing Files

When you start the program it will display a list of options,

Existing File..........
Create File............
Set Tabs..............
Display Help File....
Terminate (STOP)...

Existing File.....	= open an existing file to be edited
Create File.......	= create a file to be edited
Set Tabs..........	= set tabs and then return to this menu (see, details below)
Display HELP File.	= display on-line HELP file and then return to this menu
Terminate (STOP)..	= execution is terminated

Use UP and DOWN ARROWS to position to the option you wish to select and then press ENTER (or RETURN).

If you select either Existing File or Create File the program will then prompt you for the filename,

Enter Filename....

Type the filename exactly. If you make any mistakes while typing you can use the DELETE key to delete characters. When you are finished press ENTER (or RETURN).

The file will be opened or created and you can now start editing. While you are editing the file, all editing is done within the memory of the computer and changes to the actual file on disk are only made when you tell the code to update the disk file. If you are making many changes to a file it is recommended that you periodically SAVE these changes in the actual file on disk. This is done by pressing the HOME key to move the cursor to HOME and then pressing the S key (S = SAVE). When you are finished editing you can return to the initial list of options by pressing either,

END = SAVE the current file with all editing changes and then restart the program.

ESCAPE = Immediately restart the program WITHOUT SAVING the file. This option can be used to advantage if you have somehow made erroneous changes to the file that you want to cancel. Since using this option will cause you to lose all changes that you have made, you are asked to

CONFIRM ESCAPE by Pressing C

If you press C or c the program will then immediately restart. If you press any other key the program will assume you pressed ESCAPE by mistake, ignore this command and allow you to continue editing.

WARNING - Using the ESCAPE option will NOT undo any changes that you have made to the file on disk if you have used the SAVE option, described above. When you use the SAVE option the current file is copied from memory to disk and replaces the original file; similarly if you use the END option. When you use the ESCAPE option the file is NOT copied from memory to disk. Therefore the file on the disk will either be the original file (if you have not used the SAVE option) or the last version of the file that you SAVED.

Positioning within a File

Your current position in a file is defined by a cursor, which on a color screen is shown in GREEN. The following section describes how to position the cursor anywhere within a file.

A file being edited can be thought of as a number of pages of text, with each page being up to 28 lines in length (the number that will fit on the screen). To position within a file or on a screen,

1) Use **ARROW** keys to position LEFT, RIGHT, UP or DOWN within a page.

2) Use **PAGE UP** or **PAGE DOWN** to position to the preceding (toward the start of the file) or following (toward the end of the file) page of the file.

3) Position the cursor to **HOME** by pressing the HOME key and the use **PAGE UP** or **PAGE DOWN** to position to the beginning or end of the file.

4) Anywhere in columns 1-6 pressing **T** will cause the line where the cursor is currently located to be moved to the **TOP** of the screen.

5) On any line position the cursor to the first column and **type any number** followed by ENTER (or RETURN) to position that line number to the top of the screen, e.g., typing 1234 ENTER, will cause the 1234-th line of the file to be positioned to the top of the screen.

6) Positioning the cursor to **HOME** and **typing any number** followed by ENTER (or RETURN) has the exactly the same effect as 5), above.

These are the only controls that you need to position yourself anywhere within the file that you are editing.

Replacing, Inserting and Deleting Characters

This is a full screen editor, so that once you have positioned the cursor over a character in the file you can,

REPLACE - the character simply by typing over it. If you type over a character it is replaced and the cursor advances one position to the right, so you can continue replacing characters.

INSERT - characters by first pressing the INSERT key and then continuing to type. As you type each character will be inserting into the line and the cursor and all following characters will be shifted to the right.

You can continuing inserting characters as long as you want, including continuing to the next and following lines; if you press RETURN (or ENTER) the line will be divided at that point and a new line started.

You terminate inserting by using any cursor re-positioning command, e.g., ARROW UP, DOWN, LEFT or RIGHT, PAGE UP or DOWN, etc.

You cannot use INSERT to extend the length of any line beyond 72 characters; your input will be ignored once a line is full.

DELETE - the character by pressing the DELETE key. The character will be deleted and all following characters on the line will be shifted to the left. You can delete as many characters as you wish by repeatedly pressing the DELETE key. If you delete all of the characters on a line the line itself will be deleted.

REMINDER - For a full screen editor ENTER (or RETURN) is an acceptable character, which will cause a new line to be started. Do not use ENTER (or RETURN) to try and re-position the cursor - experience has shown that for people who are used to using other editors, but not full screen editors, this is probably the most difficult thing to remember when using this editor.

Copying, Moving and Deleting Lines

This program can be used to copy, move or delete one or a series of lines.

The syntax for all three of these options is very similar. Position the cursor to column 1 of the line where you want to start and press **C** (for **Copy**), **M** (for **Move**) or **D** (for **Delete**). Next position the cursor to column 1 of the line where you want to end and again press **C**, **M** or **D** (it must be the same character you used on the starting line - if it isn't you will get the error message Syntax ERROR - Command IGNORED).

Once you start one of these commands the sequence number of the starting line will be replaced by the word COPY, MOVE or DELETE and the message to the RIGHT on line 1 will indicate what you are doing, e.g., COPY Mode is ON.

If you are **DELETING** lines, as soon as you press D on the end line the entire range of lines between start and end will be deleted.

If you are **COPYING** or **MOVING** lines you must next position the cursor to column 1 of the line that you want the lines to a copied or moved **BEFORE**; remember at the beginning of a line the cursor is in front of the line and the COPY or MOVE will be to **BEFORE** the line where the cursor is currently positioned. To complete the COPY or MOVE press **INSERT**.

If you only want to COPY, MOVE or DELETE one line the starting and ending lines can be the same line; press C, M or D twice on the same line.

To position the cursor from the first to last line of the range and to indicate the INSERT point for COPY and MOVE you can only use,

1) UP and DOWN ARROWS to position within a page

2) PAGE UP or PAGE DOWN to position to different pages

3) type a line number starting in column 1 of any line to position to that line

If you try to position the cursor using any other commands a message will appear to the RIGHT on line 1,

Syntax ERROR - Command IGNORED

and your command, including the starting line, will be ignored.

If you only want to COPY, MOVE or DELETE a relatively small number of lines which will be COPIED or MOVED to a relatively nearby place in the file you can easily use the above cursor positioning commands without any planning.

However if a large number of lines are involved and they must be moved a relatively long distance in the file you should plan what you intent to do by first positioning through the file and noting the sequence number of the start, end and insert lines. Once you know these sequence numbers you can easily position to the points involved and quickly complete any of these operations.

Inserting or Deleting a Line

In addition to the commands described above you can INSERT or DELETE individual lines by positioning the cursor in column 1 of any line and pressing,

INSERT - to create a blank line with the cursor positioned in the first text column BEFORE the line where the cursor is located.

DELETE - to delete the line where the cursor is positioned. If you want to delete just a few lines this may be more convenient for you to use then the more general DELETE command described above.

Finding and/or Changing Text

This program will allow you to find or change text within a file.

To define a **find** command use the HOME key to position the cursor to HOME and then type **f or F** followed by the character string that you wish to find followed by ENTER (or RETURN). A find command is terminated by a ENTER or a blank. If you wish to find anything that includes blanks enclose the command in quotes. For example, you can type either,

```
f this
```

or

```
f 'this is a test find'
```

To define a **change** command use the HOME key to position the cursor to HOME and then type **c or C** followed by the character string that you wish to find followed by the character string that you wish to change it to. A change command is treated as a find followed by a change. The find and change parts of the command are each terminated by a blank or ENTER. If either character string includes blanks it should be enclosed in quotes. For example you can change XOLD to XNEW using the following change command,

```
c XOLD XNEW
```

If you define a change command as described above the program will only change the next occurrence of the character string found by searching forward in the file from the current cursor position. If you wish to change every occurrence of a character string you can include the word ALL at the end of the change command, e.g.,

```
c XOLD XNEW all
```

Since the program is primarily designed to edit FORTRAN programs it will distinguish between FORTRAN variables and general text strings. When a find or change command is defined if it is NOT enclosed in quotes this code assumes you want to find or change a FORTRAN variable. If it is enclosed in quotes it is assumed you are defining a general text string. For example, the find command,

```
f editor
```

will only find the word editor, whereas,

```
f 'editor'
```

will find editor, editors, editorial, etc. - anything in which the character string editor occurs.

You cannot use **CHANGE** to extend the length of any line beyond 72 characters; if a change commands would cause the new line length to exceed 72 characters the cursor will be positioned to the point where the change is required, but the change will not be performed. In this case the message on line 1 will say, Line TOO Long for CHANGE. If all changes are performed successfully the message on line 1 will tell you the number of changes made, e.g., 17 Changes.

Repeating Find or Change Commands

This program remembers the **last find and change commands** that you define (as described above). To repeat the last find command press FUNCTION KEY 1. To repeat the last change command press FUNCTION KEY 2. For Example, if you wish to step through a file finding all occurrences of XOLD you can first define this find command (see, preceding section) and then merely repeatedly press FUNCTION KEY 1 to find the next occurrence; the program will notify you when it reaches the end of the file without finding another occurrence.

What this program remembers is the last find command that you defined OR the last two parts of a change command = find followed by change. For example, if the last find or change command was a find, this will define what the program will find next if you press FUNCTION KEY 1. However, it will also define the first half of the change command (the find part) if you press FUNCTION KEY 2. Conversely if the last find or change command was a change, this will define what the program will change next if you press FUNCTION KEY 2. However, it will also define what the program will find next if you press FUNCTION KEY 1.

This convention can be used to good advantage. If you are not sure if you want to change every occurrence of a character string in a file, you can define a change command to change only the next occurrence of the string in a file. You can then use FUNCTION KEY 1 to position to the next occurrence of the string in the file. If you want to then change it, simply press FUNCTION KEY 2. If you do not want to change it, press FUNCTION KEY 1 again to proceed to the next occurrence of the string. Following this procedure you can quickly step through an entire file only using FUNCTION KEYS 1 and 2, to selectively decide whether or not you want to change each individual occurrence of a character string.

Display Column Numbers

This program can display column numbers for columns 1 through 72 of the text. This option can be helpful if you want to determine exactly what columns any text is located in, or how long a given string of text is.

Press HOME to position the cursor to HOME and then press # to display column numbers 1 through 72 on line 2 of the screen.

Using Tabs

It is convenient to use tabs to minimize the number of key strokes that you must use for input and to position text in specific columns. Therefore this program can read files containing tabs and you can input tabs from your keyboard.

However, the output file from this program will not contain tabs. Any tabs read from an input file or the keyboard will be replaced by blank characters to position the next input character at the next tab position on a line.

This program has built-in tabs every 7 columns, e.g., inputing a tab in any column between 1 and 6 will position the next character to be in column 7. Similarly, inputing a tab in any column between 7 and 13 will position the next character to be in column 14. This continues out to the last tab in column 70.

When you start the program you have the option to change the position of the built-in tabs. If you wish to do this, on the first screen position the cursor (using UP and DOWN ARROWS) to "Set Tabs" and press ENTER (or RETURN). The program will display column numbers 1 through 72 on the second line of the screen and the current tab positions, indicated by T on the third line. The cursor is initially positioned at the beginning of the second line. Use the SPACE BAR to position the cursor to your next tab position and press the T key; this will set the next tab position in this column. Continue to insert as many tabs as you wish. Press ENTER (or RETURN) when you have defined ALL of your tab positions. This procedure is very similar to what you would do to set tabs on a typewriter. Your defined tab positions will remain until you restart the program; at which time the tabs will be reset to their built-in default positions of every 7 columns.

Once you have started editing a file you cannot change tab positions while still editing. However, it is easy enough to simply END your editing, which will save your edited file and restart the program with the first screen. You can then "Set Tabs" and resume editing your file.

Adding Lines to the End of a File

To add lines to the end of a file position the cursor anywhere after the last line of the file and press INSERT. The program will create a line at the end of the file and position the cursor to the first column of this file, so that you can immediately start typing and continue as long as you like.

Characters Echoed by this Program

Any time you press a keyboard key and the program understands what you are trying to do it will always echo your action either by showing the character you typed or taking some action. If you press a key that the program does not understand it will ignore your action.

For example, at HOME and in columns 1 through 6, there are only a few keys that the program understands (see, summary of commands by position, below). Similarly, in columns 7 through 78 (1 through 72 of the data), you can only use keys that correspond to normal key codes (1 through 255); if you press any other key it will be ignored.

Simply by watching the screen you will immediately know whether or not the program understands what you are trying to do. If the program does not respond to your input and you do not understand why, use the on-line HELP facility to review the available commands.

On Line Help

This program comes with a file named HELP. This is a simple text file that you can read, typed to the screen or print. In addition at any time this file may be viewed using the editor by positioning the cursor to HOME and keying H (for HELP).

The first page of this file is a summary of all the commands that you can use with this editor - that's how simple this editor is - all of its commands can be displayed on only one page!

When using the on-line HELP the message displayed to the RIGHT of the first line says: Q to Quit, otherwise MORE. This means that if you have seen enough of this file, press Q to return to editing. If you want to see more of the file press any other key to see the next page.

See, the appendix for a copy of the on-line help file HELP.

Summary of Commands by Position

The following table summarizes ALL of the commands that you can use depending on the current location of the cursor. This table is identical to the first screen that you will see if you use the on-line HELP facility.

```
* - Anywhere
    * - HOME      = Position cursor to HOME
    * - END       = SAVE the file and terminate
    * - ESCAPE    = Terminate WITHOUT saving the file
    * - Page Up   = Move a Page toward the BEGINNING of the file
    * - Page Down = Move a Page toward the END of the file
    * - ARROWS    = Move cursor in the indicated direction
    * - F1        = Repeat last FIND
    * - F2        = Repeat last CHANGE
* - At HOME
    * - Page Up   = Move to the BEGINNING of the file
    * - Page Down = Move to the END of the file
    * - F         = FIND
    * - C         = CHANGE
    * - S         = SAVE the current file and continue
    * - #         = Display Column Numbers
    * - Numbers   = Position to this line NUMBER
* - In Columns 1 through 6
    * - Delete    = DELETE a Line
    * - Insert    = INSERT a Line
    * - D         = DELETE RANGE
    * - C         = CHANGE RANGE
    * - M         = MOVE RANGE
    * - T         = Move current line to the TOP of the screen
    * - Numbers   = Position to this line NUMBER
* - In Columns 7 through 78 (1 through 72 of the data)
    * - Delete    = DELETE a Character
```

```
- Insert     = Start to INSERT Characters
```

That's It

Unlike other editors where you need a 300 or 400 page manual describing how to use it, this editor is so simple that this documentation is all you need to use it. Everything has been kept as simple and natural as possible, so that you can start editing even while you are reading this document.

That's it! There is nothing more that you need to use this editor. Have fun and please contact the author if you have any comments or suggestions, either pro or con. Based on experience, it is feedback from users that leads to the most significant improvement in programs such as this one. So that any comments you have would be most appreciated.

Appendix B: A copy of the on-line Help file HELP

The following is a copy of the on-line Help file HELP as of the date of this report; for the most up-to-date version see the copy of this file that you received with the program.

This is a simple text file that you can read, type on your screen or print. When you are running this program you can also access this file by pressing the HOME key to position the cursor to HOME and then pressing H (H = HELP).

You can use this standard on-line HELP file or if you wish you can modify it to meet your own needs - you are free to include anything that you wish in this file that will be of assistance to you while you are running this editor.

If you do decide to modify this file you should understand the structure of the file. It is divided into pages of text, with up to 28 lines on each page. The end of each page is defined by a line of the form,

------------------------------(PAGE)------------------------------

This is a signal to the program that only the lines above this line should be included on the current screen. The next screen will start at the line immediately following this line; this line is not displayed on the screen.

The first page of the standard file distributed with this program is a summary of all commands that you can use - this usually serves as a quick reference in case you forget one or more of the commands. It is recommended that you leave this page in place for your use.

The following pages give more details on the use of this program. You may want to edit these pages to meet your needs, e.g., different people think different ways, so that you may prefer to explain the use of the commands in the form that you can more easily understand and follow when you use this file for on-line HELP.

You do not have to follow this "page" convention if you decide to modify this file. This program will display up to the next 28 lines from the file. If it encounters an end of page line (as described above) before displaying 28 lines, it is merely a signal to stop displaying lines on the current screen, and then continue with the next line on the next screen. If it doesn't find an end of page line it will merely keep displaying lines, up to 28 at a time. This end of page line is merely a convenience that can be used to group together and display similar information on the same screen, but it is not necessary for you to use it.

```
* - Anywhere (Summary of Commands by Position)
    * - HOME      = Position cursor to HOME
    * - END       = SAVE the file and terminate
    * - ESCAPE    = Terminate WITHOUT saving the file
    * - Page Up   = Move a Page toward the BEGINNING of the file
    * - Page Down = Move a Page toward the END of the file
    * - ARROWS    = Move cursor in the indicated direction
    * - F1        = Repeat last FIND
    * - F2        = Repeat last CHANGE
* - At HOME
    * - Page Up   = Move to the BEGINNING of the file
    * - Page Down = Move to the END of the file
    * - F         = FIND
    * - C         = CHANGE
    * - S         = SAVE the current file and continue
    * - #         = Display Column Numbers
    * - Numbers   = Position to this line NUMBER
* - In Columns 1 through 6
    * - Delete    = DELETE a Line
    * - Insert    = INSERT a Line
    * - D         = DELETE RANGE
    * - C         = CHANGE RANGE
    * - M         = MOVE RANGE
    * - T         = Move current line to the TOP of the screen
    * - Numbers   = Position to this line NUMBER
* - In Columns 7 through 78 (1 through 72 of the data)
    * - Delete    = DELETE a Character
    * - Insert    = Start to INSERT Characters
--------------------------(PAGE)--------------------------------
* - Closing and Saving Files
    * - END       = SAVE the file and terminate
    * - ESCAPE    = Terminate WITHOUT saving the file
    * - At HOME S = SAVE the file and then return to editing
* - Positioning with a file
    * - UP, DOWN, LEFT or RIGHT ARROWS within page
    * - PAGE UP or PAGE DOWN for next or preceding page
    * - At HOME, PAGE UP or PAGE DOWN for beginning or end of file
    * - In Column 1 T to position cursor line to TOP of screen
    * - In Column 1 any number to position to this line number
    * - At HOME any number to position to this line number
* - Full Screen Editor
    * - In Columns 7 through 78 (1 through 72 of the data)
        * - Replace - type over character
        * - Insert - Press INSERT and continue typing
            * - End Insert using cursor motion, e.g., any ARROW
        * - Delete - Press DELETE
* - Copying, Moving and Deleting Lines
    * - Starting Line Column 1 C (Copy), M (Move) or D (Delete)
    * - Ending Line Column 1 C (Copy), M (Move) or D (Delete)
        * - for Delete starting to end immediately deleted
        * - for Copy or Move
            * - At INSERTION Line Column 1 press INSERT
* - Inserting or Deleting a Line
- In Column 1 press INSERT or DELETE
-------------------------(PAGE)--------------------------------
* - Finding Text
    * - At HOME press F immediately followed by the string to find
        * - Enclose in quotes if string includes any blanks
            * - e.g., f 'this is a test'
* - Changing Text
    * - At HOME press C immediately followed by the string to find
        and what to replace it with
        * - Enclose in quotes if string includes any blanks
            * - e.g., c 'this is a test' 'this is not a test'
        * - Follow with ALL to change ALL occurrences
            * - e.g., c 'start with this' 'change to this' all
* - Repeating Last Find or Change
    * - Find - Press FUNCTION KEY 1
    * - Change - Press FUNCTION KEY 2
* - Display Column Numbers
    * - At HOME press #
* - Defining TABS
    * - ONLY possible at program start or restart (end editing)
    * - Select 'Set Tabs'
        * - Use space bar to space to column
        * - Use T to define TAB positions
        * - Repeat for each TAB
        * - ENTER (or RETURN) at end
-------------------------(PAGE)--------------------------------
* - Requirements (on IBM-PC)
    * - 8 megabyte of memory
    * - keyboard
    * - Color screen desirable
* - Limitations
    * - 100,000 lines per file
    * - 72 characters per line
* - Screen layout
    * - line     1 - Program identification and messages
                     * - for messages watch the upper
                         right hand corner of the screen
    * - line     2 - Home and number of lines in the current file
    * - lines 3-30 - Lines from the file you opened (28 lines)
    * - cols. 1- 6 - Line number from the file
    * - cols. 7-78 - Line from the file (up to 72 characters)
```

Utility Codes: Version 98-1 Summary

TART is distributed with a number of utility codes that are designed to edit TART output for criticality or source problems. The primary reference to these utility codes remains Chapter 8: Utility Codes, of the TART95 documentation (included in the TART on-line documentation). Below is a brief introduction and update for each code.

Changes in TART98 Output Format

In order to accommodate many more zones the TART output format has been changed starting with TART98; earlier only three digit zone number were accommodated by the format; this has been extended to five digits.

As a consequence all of the TART utility codes distributed with TART98 CD have been updated to recognize the new output format. **WARNING** - do not use older versions of the TART utility codes; they have been completely superseded.

Multiprocessing

WARNING - Currently MULTIPRO and TARTSUM can only be used for source, not criticality, problems.

MULTIPRO
Prepare input to start any number of TART runs. After ALL runs have finished, TARTSUM can be used to average the results of any number of runs.
TARTSUM
Combine results of a number of TART runs. This can be used to improve the statistical accuracy of your results by adding additional results. When using multiprocessing this can quickly improve results.

Criticality Problems

CRITEDIT
Summarize static reactivity results in tabular form.
BALANCE
Edit criticality production, absorption and leakage spectra for an entire system to PLOTTAB input format. Use PLOTTAB to see your results.
PATH
Edit pathlength (flux) spectra for individual spatial zones. Use PLOTTAB to see your results.

Source Problems

FLUXEDIT
Edit flux and energy deposition for each spatial zone to PLOTTAB input format. Use TARTCHEK to overlay the results on your geometry.
EDIT1112
Edit binary output to tabular form.
PLOT1112
Edit binary output to PLOTTAB input format. Use PLOTTAB to see your results.

Input Parameters

If there are input parameters, they will be in a file with the same name as the utility codes, plus an extension of .INP, e.g., for the BALANCE code, BALANCE.INP. Each input file contains a complete, up-to-date description of all input parameters. As such, input parameters will not be described here.

Input TART Results

TART.OUT is read by CRITEDIT, BALANCE, PATH, and FLUXEDIT. This is the TART default file name for TART output.

TART.OUT is written by TARTSUM. It reads a number of TART output files, and creates TART.OUT, containing the combined results.

EDIT1112 and PLOT1112 read binary output files created using TART tally type 11 or 12. These are TART options to score particles entering a zone and output their coordinates to a binary file for later use.

Multiprocessing

WARNING - Currently MULTIPRO and TARTSUM can only be used for source, not criticality, problems.

MULTIPRO

MULTIPRO prepares input to be used to start any number of TART source problems, and to later combine the results using TARTSUM. This can be used to improve the statistical accuracy of your results by adding additional results. When using multiprocessing this can quickly improve results.

MULTIPRO reads ONE TART input problem, named TART.IN, and produces ANY number of copies of the input deck where each copy differs from the original ONLY by the random number sequence used (sentl 12) and binary tally type name (sentl 51).

MULTIPRO also produces a batch file named TART98.BAT that can be used to start all of the input decks that it created. The contents of TART98.BAT look like this,

```
tart98-5 IN000001 OUT00001 > LST00001 &
tart98-5 IN000002 OUT00002 > LST00002 &
tart98-5 IN000002 OUT00002 > LST00003 &
                    .
                    .
                    .
tart98-5 IN000049 OUT00049 > LST00049 &
tart98-5 IN000050 OUT00050 > LST00050 &
```

Where in this example,
IN000001 through IN000050 are 50 copies of the TART input (created by MULTIPRO)
OUT0001 through OUT00050 are 50 TART output listings (created by TART)
LST00001 through LST00050 are 50 dummy files for normal screen output (created by TART)

On UNIX and LINUX systems the & at the end of each line will run each problem in the background, thereby releasing your terminal to immediately continue on to start the next run, without delaying until the preceding run is completed. Essentially all 50 runs will be started at the same time.

On other systems, such as IBM-PC/Windows, currently & is ignored, and the problems are run sequentially one after another. Therefore on other systems the advantage of the simultaneous multiprocessing possible on UNIX and LINUX systems is currently not possible. However, note that the procedures described here can still be used to advantage by using any number of IBM-PC/Windows computers to simultaneously run problems for you, and to then use TARTSUM to add the results together.

MULTIPRO also produces an input file that can be used with TARTSUM to add all of the results together. For example, in the case described above where 50 problems are run, MULTIPRO will produce a file named TARTSUM.INP. This file will contain the filenames of the 50 TART output listings to add together. For the above example this file would contain,

```
OUT000001
OUT000002
OUT000003
    .
    .
    .
OUT000049
OUT000050
```

Input Parameters: **MULTIPRO.INP**

To use MULTIPRO, edit **MULTIPRO.INP** to define how many problems you want to run, and which random number sequence to start with; each successive problem will be assigned the next random number sequence is sequential order, and type,

MULTIPRO

On any UNIX or LINUX sysyem when MULTIPRO finishes follow these steps,
1) Make sure TART98.BAT is executable [chmod 777 TART98.BAT] and execute it to start all of the problems running.
2) Use "top" to monior execution of the problems.
3) When ALL of the problems have finished execute TARTSUM to average all of your results together and produce one final TART output file named TART.OUT.

It is as simple as that. Used properly this can be an extremely simple and yet powerful tool that can greatly accelerate how much work you can accomplish in a short period of time. For example, if you have available one or more of the currently available computers that have thousands of processors, you could use all of the processors to accelerate your work by factors of thousands.

TARTSUM

TARTSUM will combine results of a number of TART runs. This can be used to improve the statistical accuracy of your results by adding additional results. When using multiprocessing this can quickly improve results.

The TARTSUM code reads a series of entire TART output files, for **any number of source problems**, and creates a combined file named TART.OUT. All of the output files MUST correspond to EXACTLY the same TART input problem(s) - the ONLY thing that you are allowed to change is the random number sequence (sentl 12) [to make the results statistically independent], and binary tally type name (sentl 51)..

Input Parameters: **TARTSUM.INP**

To use TARTSUM, edit **TARTSUM.INP** to define the output you want, be sure that all of the files you want it to read are in the same directory, and that NONE are named TART.OUT (the combined output file name), and type,

TARTSUM

You will obtain results in a file named TART.OUT, that is in EXACTLY the same format as any other TART output file. Therefore, if you have any codes that process TART output - not to worry - they will work on the combined file, just as they work on any other TART output files.

Criticality Problems

CRITEDIT

CRITEDIT reads an entire TART output file, TART.OUT, for **any number of criticality problems**, and produces a table summarizing the results of the calculations.

Input parameters: **NONE**

To use CRITEDIT, be sure that the file you want it to read is named TART.OUT, and type,

```
CRITEDIT
```

You will obtain results both on your screen as the code runs, and in an output file named CRITEDIT.LST (CRITEDIT list).

There are two primary use for CRITEDIT, both are illustrated in the TART95 documentation,
1) Summarize the results of a variety of different criticality calculations.
2) Summarize the results of running the same criticality calculation a number of times. This can be used to further check on and improve the statistical accuracy of results

BALANCE

For any criticality calculation the multiplication of the system (K-eff), is defined as a BALANCE between the production of neutrons and loss from the system due to absorption and leakage. BALANCE will allow you to see the results for any system as a function of neutron energy. Results include a spectrum of the incident energy at which neutron interactions produced neutrons (not the fission spectrum produced), as well as spectra of the energy at which neutrons were absorbed or leaked from the system. By comparing these three curves it is very simple to see what energy ranges are important. For example, those energy ranges where production exceeds the sum of absorption and leakage will tend to make a system super-critical, and those energy ranges where production is less than the sum of absorption and leakage will tend to make it sub-critical. The integral over energy of these results defines the multiplication of the system (K-eff).

The BALANCE code reads an entire TART output file, TART.OUT, for **one criticality problem**, and produces output in the PLOTTAB input format, so that you can immediately see your results.

Input Parameters: **BALANCE.INP**

To use BALANCE, edit **BALANCE.INP** to define the output you want, be sure that the file you want it to read is named TART.OUT, and type, BALANCE

You will obtain results in a file named PLOTTAB.CUR, which can be moved to your PLOTTAB directory and immediately plotted; see, the PLOTTAB documentation for details.

PATHC

BALANCE is designed to allow you to see overall features of an entire system. In contrast, PATHC allows you to see the features of individual spatial zones. It will edit pathlength (flux) spectra as a function of energy for individual spatial zones, and allow you to optionally renormalize the results per unit energy and/or unit volume.

Input Parameters: **PATHC.INP**

To use PATH, edit **PATHC.INP** to define the output you want, be sure that the file you want it to read is named TART.OUT, and type,

PATHC

You will obtain results in a file named PLOTTAB.CUR, which can be moved to your PLOTTAB directory and immediately plotted; see, the PLOTTAB documentation for details.

WARNING - the name of this utility code has been changed to PATHC from the earlier name PATH. The earlier name caused problems on some systems where attempts to run a code named path by typing,

path

resulted in changing the system defined "path".

Source Problems

FLUXEDIT

FLUXEDIT is designed to edit flux and energy deposition for each spatial zone to PLOTTAB input format, which can be used by TARTCHEK to overlay the results on your geometry.

The FLUXEDIT code reads an entire TART output file, TART.OUT, for **one source problem**, and produces output in the PLOTTAB input format, so that you can immediately see your results.

Input Parameters: **NONE**

To use FLUXEDIT, be sure that the file you want it to read is named TART.OUT, and type,

```
FLUXEDIT
```

You will obtain results in a file named FLUXEDIT.OUT, which can be moved to your TARTCHEK directory and immediately plotted. See the chapter on **TARTCHEK** in the TART95 documentation to learn how to overlay your results on your geometry.

EDIT1112

TART tally types 11 and 12 for neutrons or photons allows users the option of tallying particles when they enter individual spatial zones; they are then terminated. For each particle the output includes all spatial, energy, time, direction, and the number of collisions from source. This option is used by many TART users for special applications, such as detector response problems.

EDIT1112 was initially designed to read a binary file created by TART and produce tabulated results that users can then use in their applications. However, currently it is more often used by users within their special purpose application codes. EDIT1112 is incorporated into their codes and is used to read the coordinates for one particle at a time from a binary file, and the users then do whatever they want with them.

The EDIT1112 code reads **a single binary** TART output file, for **a single source problem**, and creates a file of tabulated results. The user defines by input the file name of the output tabulated result.

Input Parameters: **EDIT1112.INP**

To use EDIT1112, edit **EDIT1112.INP** to define the output you want, be sure the binary file you want it to read is in the same directory, and type,

```
EDIT1112
```

You will obtain tabulated results in the file you specified by input.

WARNING - EDIT1112 will only read **a single binary** file, even though TART may create an entire family of binary files. This is done because initial tests of the code illustrated that reading and tabulating the results for an entire family of files can result in enormous output files. To use an entire family of file, see PLOT1112 below.

PLOT1112

PLOT1112 is merely one example of a special purpose code built on top of EDIT1112. PLOT1112 reads **a family of binary** files created by TART, summarize the results in a file name defined by user input, and also produce results versus time, in a file named

TIME.CUR, versus energy, in a file named **ENERGY.CUR**, and versus the direction cosine with respect to the Z axis, in a file names **ZCOSINE.CUR**. Each of these files is in the PLOTTAB format, so that you can use PLOTTAB to see your results. Each file contains PLOTTAB "curves" for total results, as well as results for each number of collisions that particles had between source and entering the tally zone, e.g., uncollided, first collided, second collided, etc., results.

Input Parameters: **PLOT1112.INP**

To use PLOT1112, edit **PLOT1112.INP** to define the output you want, be sure the entire family of binary files you want it to read is in the same directory, and type,

```
PLOT1112
```

You will obtain a summary of results in the file you specified by input, as well as **TIME.CUR** and **ENERGY.CUR**, that can be used with PLOTTAB to see your results. Note, **TIME.CUR** and **ENERGY.CUR** correspond to the standard PLOTTAB input file **PLOTTAB.CUR**. By moving either to your PLOTTAB directory and renaming it **PLOTTAB.CUR** you can immediately see your results. See, the PLOTTAB documentation for details.