Parametric Study of the Total System Life Cycle Cost of an Alternate Nuclear Waste Management Strategy Using Deep Boreholes

by

Taylor Allen Moulton

B.S., Nuclear and Radiological Engineering
University of Florida, 2006

Submitted to the Department of Nuclear Science and Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Nuclear Science and Engineering

at the
Massachusetts Institute of Technology
September 2008

Signature of Author: _____
Department of Nuclear Science and Engineering
August 8, 2008

Certified by: _____
Richard K. Lester
Professor of Nuclear Science and Engineering
Thesis Supervisor

Certified by: _____
Michael J. Driscoll
Professor Emeritus of Nuclear Science and Engineering
Thesis Reader

Accepted by: _____
Jacqueline C. Yanch
Chairman, Committee on Graduate Students
Department of Nuclear Science and Engineering

Parametric Study of the Total System Life Cycle Cost of an Alternate Nuclear Waste Management Strategy Using Deep Boreholes

by

Taylor Allen Moulton

Submitted to the Department of Nuclear Science and Engineering on
August 8, 2008 in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Nuclear Science and Engineering

Abstract

The Department of Energy recently submitted a license application for the Yucca Mountain repository to the Nuclear Regulatory Commission, yet even the most optimistic timetable projects that the repository will not now open until at least 2020. The Office of Civilian Radioactive Waste Management recently revised the official undiscounted total life cycle cost of the waste management system upward by $22B (2000$), an increase of nearly 40% over the previous estimate, published in 2001. In this thesis a waste management tool, named SNuFManager (**S**pent **Nu**clear **F**uel **Manager**), has been developed which deterministically simulates the stocks and flows of spent fuel in the United States and estimates annual expenditures based on the system's behavior. The tool allows policy makers to quickly and cheaply estimate the economic consequences of various decision alternatives under an array of scenarios in order to make quantitatively informed decisions and identify ways to mitigate or reverse recent increases in life cycle costs. The results are expressed in 2000 dollars, enabling a convenient comparison with the government's 2001 total system life cycle cost analysis.

For each year of delay beyond 2020 in opening the repository and transferring ownership of spent fuel to the federal government, the total waste management system life cycle cost is estimated to increase by another $330M (2000$). The model also estimates that switching from the current mined geologic repository approach to a deep borehole disposal strategy would reduce the undiscounted total system life cycle cost by $19.4B, or 32%. Assuming a 10% discount rate, the net present cost of the deep borehole strategy is 18% less than that of the mined geologic repository approach. Finally, the model illustrates the economic benefits of opening a centralized interim storage facility of significant capacity as soon as possible. For example, if a 40,000 metric tonne facility, comparable in scale to the proposed Private Fuel Storage Facility in Utah, was opened by 2020, and the mined repository was opened in the same year, the total life cycle cost would be reduced by $1.5B relative to the case with no interim storage. If, moreover, the opening date of the mined geologic repository were delayed until 2040 or 2060, the savings provided by interim storage increase dramatically, to $4.9 and $8.1B, respectively. The thesis concludes with a discussion of the political and strategic consequences of several key policy choices.

Thesis Supervisor: Richard K. Lester
Title: Professor of Nuclear Science and Engineering

Thesis Reader: Michael J. Driscoll
Title: Professor Emeritus of Nuclear Science and Engineering

Acknowledgements

There are so many people responsible for this work, but I have promised to keep this to a single page so I cannot possibly acknowledge everyone and for those I do, my brief comments reflect only a small fraction of my appreciation for their impact on my life. Without question, I must start by recognizing my thesis advisors, Professors Richard Lester and Michael Driscoll. In the absence of their deeply informed and constructive advice, the timely completion of this work would not be possible. They were able to focus my skills in such a way that I felt we were all truly learning together and provided a comfortable atmosphere in which mistakes were not beaten to death, but recognized as part of the learning process.

Of course, my family, my mother most of all, gives me unending love and support in a way that is careful to foster an independent and self-motivating attitude to never settle because one must continue to move to grow and survive. Through my family, I learned the lesson of personal responsibility – how to recognize that everything we do in life impacts others in a greater system and how to identify and stop behaviors that would unfairly burden others. My appreciation for this life lesson continues to grow as I witness it dwindling in human relationships, corporate practices, and government policies.

Perhaps it is because I grew up an only child, but I consider my best friends an integral part of my family. Of course I cannot comment on how all have helped me and there is no way to order their significance because I truly draw on their support in different ways at different stages of my life, but I should mention a few and for lack of a better system I will mention them in chronological order. Joey, you helped me think about the world in ways I never could have on my own at a time that was most critical in encouraging my open mindedness (zero years). Katie, I am shocked the more I learn about you and realize I should rely heavily on your advice as you are one of the few that thinks the way I do (straight). Jon, I love that I can always count on you to crack me up while knowing you will be there to help any time I really need to open up (cosmic bowling). Arash, there is no one in the world I could trust more to speak his mind and tell it like it is, or at least like you think it is, and I want you to know I will be there for you every time that behavior gets you in a fight (Jilax). Jennie, you have always given me a home away from home, even when we lived together, and I hope you can count on me to offer the same for you (IB-Psych). Elme, if it were not for you I would probably still believe this world was just black and white (lewe). Arlene, creces más hermosa cada vez tú me desafía (cobbler). Katie, your honesty in life should be a lesson to the ocean of illusory women (arm). Wanna give a shout out to the Godfather. If it wasn't for you I wouldn't be here today Mitch. I'd probably be dead. Face down in a drained pool somewhere.

I consider life a large classroom and I have been blessed with instructors and mentors who alternatively considered the classroom and the workplace an opportunity to gain perspective on life. I have had so many great teachers, but I must mention a few from high school, Dan Bertossa, Arup Guha, Rosalie Gwinn, and Renee Bell, as well as a couple of my college professors, Samim Anghaie, Jim Baciak, Wes Bolch, Ed Dugan, Alireza Haghighat, Glenn Sjoden, Jim Tulenko, and William Vernetson, who truly live to teach and do not end the lesson upon reaching an arbitrary academic subject boundary. In graduate school, all three of my System Dynamics professors, John Sterman, Paulo Gonçalves, and Kim Thompson, shared lesson plans I will reflect on for a lifetime that contained far more than just equations. In much the same way, my professional mentors, especially John Austin, Jon Earnhart, and Scott Palmtag, taught me so much more than a skill for the workplace.

The research was performed under appointment of the Office of Civilian Radioactive Waste Management Fellowship Program administered by Oak Ridge Institute for Science and Education under a contract between the U.S. Department of Energy and the Oak Ridge Associated Universities.

Finally, I would like to acknowledge one of the most influential literary pieces to my maturation. The Journey, by the American poet, Mary Oliver, seems to say just the right thing every few years or so when I am struggling to find the right direction in life or am worried about taking a big risk and tackling a new challenge. I can only hope it does the same for you.

The Journey by Mary Oliver

One day you finally knew
what you had to do, and began,
though the voices around you
kept shouting
their bad advice –
though the whole house
began to tremble
and you felt the old tug
at your ankles.
"Mend my life!"
each voice cried.
But you didn't stop.
You knew what you had to do,
though the wind pried
with its stiff fingers
at the very foundations,
though their melancholy
was terrible.
It was already late
enough, and a wild night,
and the road full of fallen
branches and stones.
But little by little,
as you left their voices behind,
the stars began to burn
through the sheets of clouds,
and there was a new voice
which you slowly
recognized as your own,
that kept you company
as you strode deeper and deeper
into the world,
determined to do
the only thing you could do –
determined to save
the only life you could save.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Thesis Objectives and Outline

This thesis presents a methodology for evaluating the scale of the nuclear waste management task from all the U.S. reactors through the back-end of the fuel cycle using a system dynamics framework. The physical stocks and flows of material, coupled with the costs of unit operations, determine the required cash flows so the combination of the economic analysis with the spent nuclear fuel management code gives a basic evaluation of the annual expenditures for a particular waste management strategy and therefore the total system life cycle cost. The development of the waste management tool gives policy makers great flexibility in analyzing the likely dynamics of different unloading strategies and, most importantly, the time profile of expenditures until DOE completes waste disposal.

Chapter 1 reflects on the history of the nuclear industry in a waste management perspective and introduces the magnitude of the challenge in dealing with the accumulation of spent nuclear fuel.

Chapter 2 discusses the approach taken to analyze the complex problem of tracking and controlling the stocks and flows of waste generated at many reactors across the nation. An example model of a single reactor generating spent fuel illustrates the basic principles of system dynamics that go into the development of a more general program written in Fortran90.

Chapter 3 reviews the basic expense modules used in this thesis. The generalized evaluation technique allows the decision maker to include the expense to the utility during wet storage in spent fuel pools and on-site dry storage, and the modularity provides flexibility in evaluating only a single aspect or in adding further considerations at a later date.

Chapter 4 explores the impact of policy decisions using the tools developed in this thesis. These tools provide policy makers realistic feedback into the effects of (a) delays in opening repositories, (b) benefits of deep borehole repository modularity, (c) alternative spent fuel pool unloading strategies, and (d) use of interim storage. In each case, the evaluation attempts to minimize the changes to the default strategy in an effort to parameterize the waste management evaluation and provide valuable analysis on the relative benefits of various strategies.

Chapter 5 quantitatively evaluates the significance of extending reactor operating licenses and qualitatively reviews the effects of increasing the average future spent fuel burnup and average future capacity factor. In doing so, this work provides a thorough analysis of the waste management problem from an economic perspective and highlights the benefits of a deep borehole strategy.

1.2 Rise of Nuclear Power and Accumulation of Used Fuel

Enrico Fermi and a team of 43 scientists from around the world initiated the first man-made fission chain reaction on December 2, 1942 in a racquets court underneath Stagg Field's west stands at the University of Chicago (1.1). A drawing of the Chicago Pile-1 reactor can be seen in Figure 1.1. This accomplishment would prove essential for the Manhattan Project and the development of nuclear weapons, but Fermi would later reflect on the historical event's significance in the rise of civilian nuclear power (1.2). He reported a vision of large "piles" (nuclear power reactors) providing tremendous amounts of safely controlled energy and being extremely competitive with traditional coal power plants, while acknowledging many of the challenges the industry still faces today including the general public's fear of nuclear power because of its association with nuclear weapons. However, Fermi's report hardly mentions the problem of waste accumulation, largely because he envisioned a scenario in which the high demand for nuclear fuel in the once-through fuel cycle would deplete uranium resources and economically force the U.S. to recycle material leaving the traditional reactors by the 1970s.

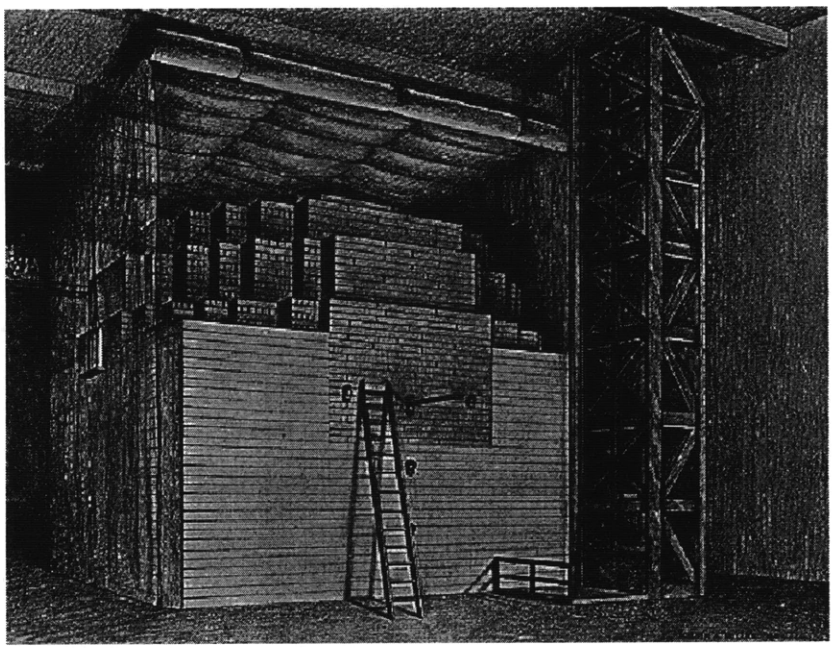


Figure 1.1. Drawing of Fermi's Chicago Pile-1 Nuclear Reactor Underneath Stagg Field (1.3).

The prediction of a large demand for nuclear power came true with the most rapid expansion in the U.S. occurring between 1972 and 1976, when the country added roughly 11 reactors per year, as shown in Figure 1.2. However, these were all thermal reactors (mostly light water reactors or LWRs) and the vision of closing the fuel cycle never came to pass. The LWR design was pursued in the U.S. and became

the most prevalent design worldwide. This was possible because of the enrichment technology developed during World War II, which enabled the utilization of high-pressure water as the coolant and moderator (1.4). The LWR designs are further classified as boiling water reactors (BWRs), direct-cycle systems operating at roughly 1,000 psi and 550°F with bulk boiling in the core, which account for approximately one-third of the U.S. fleet, and pressurized water reactors (PWRs), the majority of the U.S. fleet, which have an isolated primary loop with a pressure of roughly 2,200 psi and core outlet temperature of 600°F (1.5). Although these systems have fundamental differences in their design and operation, the fuel cycle characteristics are so similar that they are considered identical for the purposes of this study.

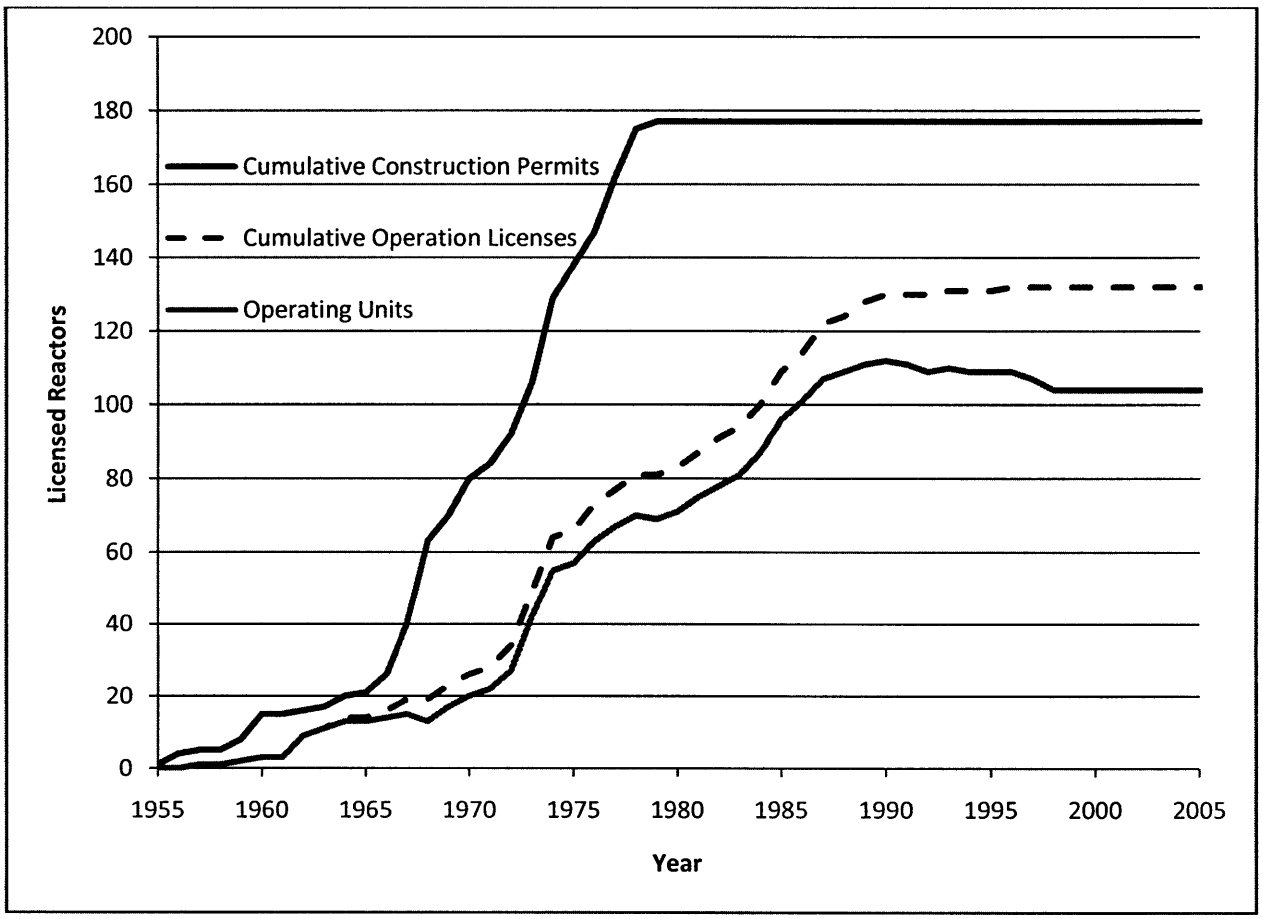


Figure 1.2. Cumulative Starts of Operational Commercial Nuclear Reactors in the U.S. (1.6).

Early reviews of uranium resources suggested that the supply is quite constrained, but each update reveals a greater supply than previously perceived, as shown in Figure 1.3, because private companies explore for uranium and doing so beyond a couple decades of demand is not economically prudent (1.7;

1.8). Investments in fast reactors necessary to close the fuel cycle were cost prohibitive in part because of the discovery of abundant uranium reserves (1.9).

Though a new nuclear power plant has not been licensed in decades, nuclear waste continues to accumulate in the once-through fuel cycle due to the 40 to 60 year plant lifetimes. Without reprocessing and in the absence of an operational waste repository, spent fuel pools are quickly reaching their capacity and utilities are forced to use costly on-site dry storage casks. Figure 1.4 shows the historical and Keystone Center projections of spent nuclear fuel discharges in the U.S. over the next half century (1.10). Notice that the current inventory amounts to nearly 60,000 MT and is accumulating at roughly 2,000 MT per year. The site specific spent fuel data is listed in Table 1.1.

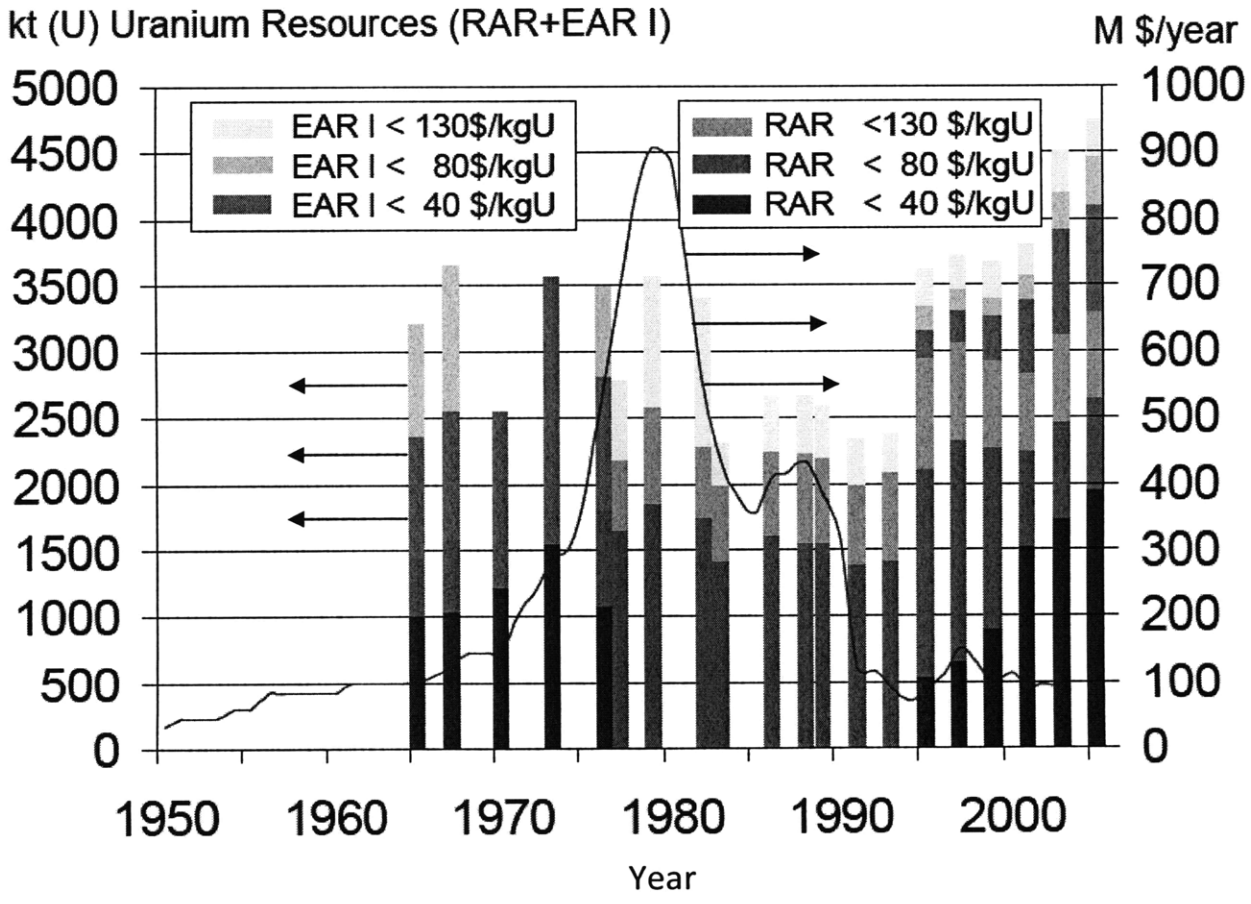
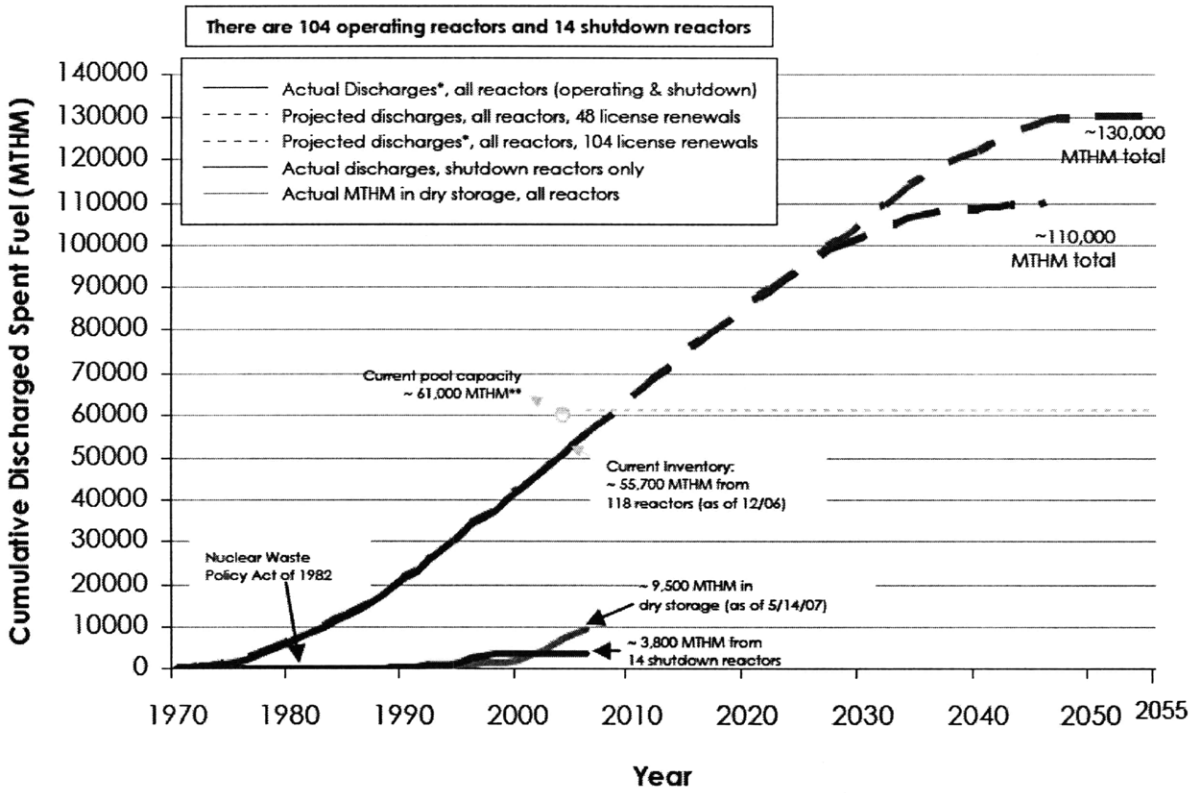


Figure 1.3. Uranium Resources and Annual Exploration Expenditure (1.11).



Sources: * Based on actual discharge data as reported on RW-859's through 12/31/02, and projected discharges, in this case, based on 104 license renewals.
** Represents the aggregate industry pool capacity based on pool capacities provided in 2002 RW-859 (less FCR) and supplemented by utility storage plans. However, the industry is not one big pool and storage situations at individual sites differ based on pool capacities versus discharges into specific pools.

Figure 1.4. Historical and Projected Spent Nuclear Fuel Discharges (1.10).

Table 1.1. U.S. 2002 Spent Fuel Pool Data (1.12).

Site	Type	Power (MWe)	License Expiration	Assemblies Stored	Assembly Capacity
Arkansas Nuclear One 1	PWR	836	2034	779	968
Arkansas Nuclear One 2	PWR	858	2018	738	988
Beaver Valley 1	PWR	821	2016	876	1,627
Beaver Valley 2	PWR	831	2016	580	1,088
Big Rock Point	BWR	67	2027	0	441
Braidwood 1	PWR	1,161	2026	1,485	2,984
Braidwood 2	PWR	1,154	2027	--	--
Browns Ferry 1	BWR	1,155	2013	4,536	6,942
Browns Ferry 2	BWR	1,118	2014	--	--

Browns Ferry 3	BWR	1,118	2016	2,160	3,471
Brunswick 1	BWR	872	2016	976 BWR 160 PWR	1,803 BWR 160 PWR
Brunswick 2	BWR	811	2014	947 BWR 144 PWR	1,839 BWR 144 PWR
Byron 1	PWR	1,163	2024	1,786	2,984
Byron 2	PWR	1,131	2026	--	--
Callaway	PWR	1,125	2024	1,118	2,642
Calvert Cliffs 1	PWR	825	2034	1,348	1,830
Calvert Cliffs 2	PWR	835	2036	--	--
Catawba 1	PWR	1,129	2024	944	1,419
Catawba 2	PWR	1,129	2026	836	1,418
Clinton 1	BWR	1,022	2026	1,580	2,672
Columbia	BWR	1,107	2023	1,904	2,658
Comanche Peak 1	PWR	1,150	2030	1,273	3,373
Comanche Peak 2	PWR	1,150	2033	--	--
Cook 1	PWR	764	2014	2,198	3,613
Cook 2	PWR	834	2017	--	--
Cooper Station	BWR	1,000	2014	1,537	2,366
Crystal River 3	PWR	1,060	2016	824	1,357
Davis-Besse	PWR	882	2017	749	1,624
Diablo Canyon 1	PWR	1,087	2021	908	1,324
Diablo Canyon 2	PWR	1,087	2025	828	1,324
Dresden 1	BWR	197	1978	0	720
Dresden 2	BWR	850	2009	2,954	3,537
Dresden 3	BWR	850	2011	2,744	3,537
Duane Arnold	BWR	565	2014	1,912	3,152
Enrico Fermi 2	BWR	830	2025	1,708	4,608
Farley 1	PWR	839	2017	1,050	1,407
Farley 2	PWR	1,089	2021	961	1,407
Fitzpatrick	BWR	813	2014	2,460	2,797
Fort Calhoun	PWR	478	2033	839	1,083
Grand Gulf 1	BWR	480	2024	3,160	4,348
Haddam Neck	PWR	1,207	2007	1,019	1,172
Harris 1	PWR	900	2026	2,793 BWR 1,021 PWR	5,304 BWR 3,080 PWR
Hatch 1	BWR	856	2034	5,015	6,026
Hatch 2	BWR	883	2038	--	--
HB Robinson 2	PWR	710	2010	344	544

Hope Creek	BWR	1,049	2026	2,376	4,006
Humboldt Bay	BWR	63	1976	390	390
Indian Point 1	PWR	257	1974	160	756
Indian Point 2	PWR	951	2013	1,078	1,374
Indian Point 3	PWR	979	2015	833	1,345
Kewaunee	PWR	511	2013	904	1,205
La Crosse	BWR	48	1987	333	333
LaSalle County 1	BWR	1,111	2022	4,106	8,059
LaSalle County 2	BWR	1,111	2023	--	--
Limerick 1	BWR	1,134	2024	4,665	7,586
Limerick 2	BWR	1,134	2029	--	--
Maine Yankee	PWR	860	1997	1,170	2,019
McGuire 1	PWR	1,100	2021	1,107	1,463
McGuire 2	PWR	1,100	2023	1,125	1,463
Millstone 1	BWR	641	1998	2,884	3,229
Millstone 2	PWR	871	2015	1,020	1,306
Millstone 3	PWR	1,130	2025	654	1,860
Monticello	BWR	578	2010	1,342	2,237
Nine Mile Point 1	BWR	565	2009	2,524	4,086
Nine Mile Point 2	BWR	1,120	2026	1,932	4,049
North Anna 1	PWR	925	2038	1,410	1,737
North Anna 2	PWR	917	2040	--	--
Oconee 1	PWR	846	2013	926	1,312
Oconee 2	PWR	846	2013	--	--
Oconee 3	PWR	846	2014	488	825
Oyster Creek	BWR	619	2009	2,556	3,035
Palisades	PWR	730	2011	649	896
Palo Verde 1	PWR	1,243	2024	948	1,034
Palo Verde 2	PWR	1,243	2025	948	1,033
Palo Verde 3	PWR	1,247	2027	851	1,034
Peach Bottom 2	BWR	1,116	2033	2,908	3,819
Peach Bottom 3	BWR	1,093	2034	2,997	3.819
Perry 1	BWR	1,235	2026	2,088	4,020
Pilgrim 1	BWR	653	2012	2,274	3,859
Point Beach 1	PWR	512	2010	1,353	1,502
Point Beach 2	PWR	518	2013	--	--
Prairie Island 1	PWR	522	2013	1,135	1,386
Prairie Island 2	PWR	522	2014	--	--
Quad Cities 1	BWR	855	2012	6,116	7,554

Quad Cities 2	BWR	855	2012	--	--
R.E. Ginna	PWR	498	2009	976	1,879
Rancho Seco	PWR	966	2008	0	1,080
River Bend 1	BWR	710	2025	2,148	2,680
Salem 1	PWR	839	2016	992	1,632
Salem 2	PWR	839	2020	812	1,632
San Onofre 1	PWR	436	1992	207	216
San Onofre 2	PWR	1,092	2013	1,166	1,542
San Onofre 3	PWR	1,080	2013	1,117	1,542
Seabrook	PWR	1,070	2026	624	1,236
Sequoyah 1	PWR	1,080	2020	1,700	2,316
Sequoyah 2	PWR	1,155	2021	--	--
South Texas 1	PWR	1,125	2027	627	1,969
South Texas 2	PWR	1,126	2028	627	1,969
St. Lucie 1	PWR	1,251	2016	1,352	1,706
St. Lucie 2	PWR	1,251	2023	900	1,584
Summer Unit 1	PWR	966	2022	812	1,712
Surry 1	PWR	810	2032	827	1,044
Surry 2	PWR	815	2033	--	--
Susquehanna 1	BWR	1,105	2022	4,240	5,680
Susquehanna 2	BWR	1,111	2024	--	--
Three Mile Island 1	PWR	802	2014	898	1,990
Trojan	PWR	1,095	1992	780	1,408
Turkey Point 3	PWR	693	2032	916	1,404
Turkey Point 4	PWR	693	2033	938	1,404
Vermont Yankee	BWR	510	2012	2,671	3,353
Vogtle 1	PWR	1,152	2027	1,639	3,574
Vogtle 2	PWR	1,149	2029	--	--
Waterford 3	PWR	1,075	2024	960	2,104
Watts Bar 1	PWR	1,125	2035	321	1,610
Wolf Creek 1	PWR	1,165	2025	925	2,642
Yankee Rowe	PWR	167	1991	173	721
Zion 1	PWR	1,040	1998	2,226	3,012
Zion 2	PWR	1,040	1998	--	--

It has been decades since the last nuclear power plant construction permit was licensed, as Figure 1.2 shows. However, there is much optimism in the nuclear industry about the possibility of a renaissance if an acceptable waste management strategy becomes operational. Driven by growing energy needs in

combination with environmental concerns over fossil fuel alternatives, "renaissance" scenarios are projected in several studies including the U.S. Department of Energy's *Annual Energy Review 2006* and MIT's *Future of Nuclear Power* (1.6; 1.13). Plausible nuclear power deployment scenarios in combination with rising capacity factors, shown in Figure 1.5, could leave the United States with 200,000 to over 300,000 MT of used nuclear fuel in the next half century. This uncertainty requires a well-developed waste management strategy that is robust enough to accommodate significant variability in demand.

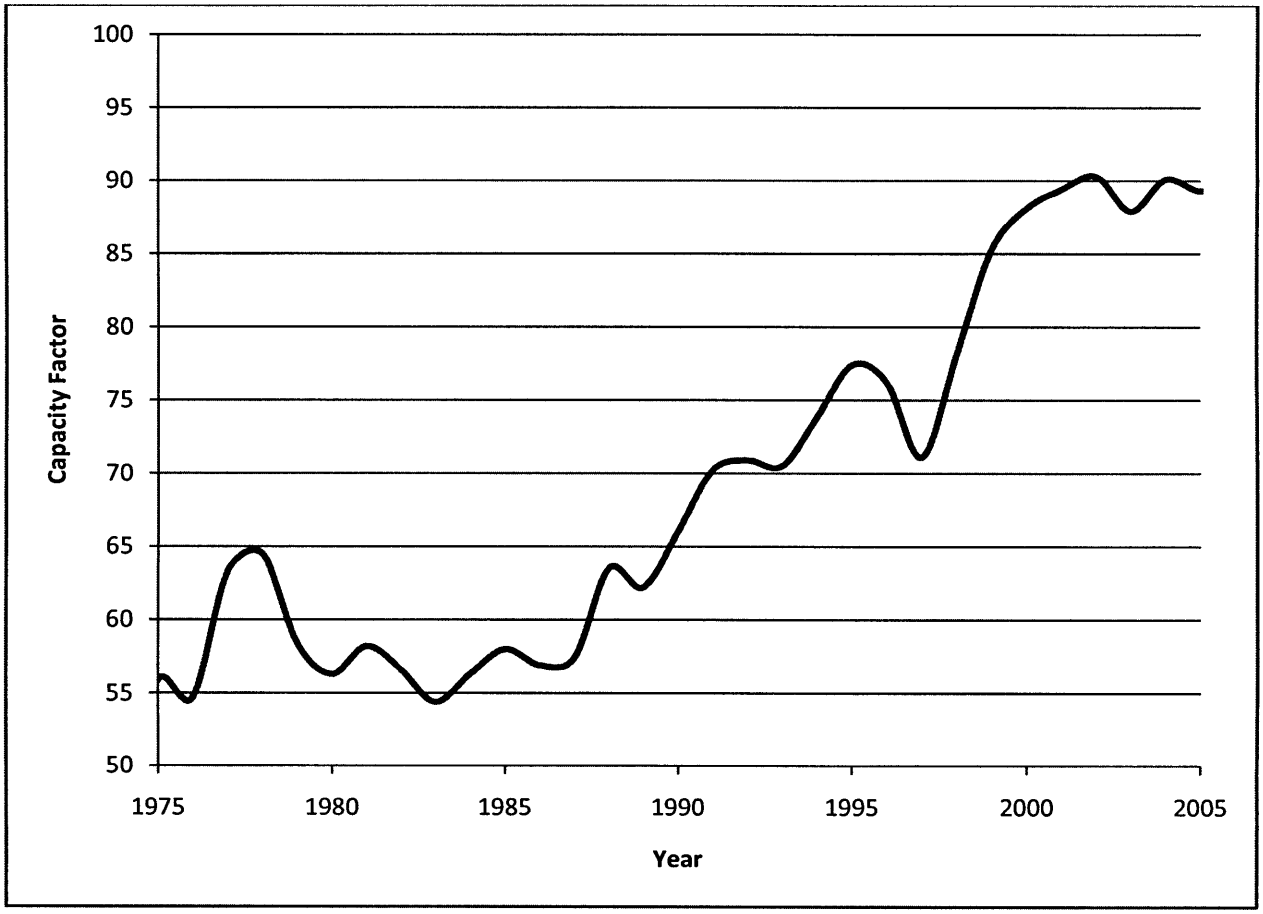


Figure 1.5. Capacity Factors of the U.S. Commercial Nuclear Plants in the Last 30 Years (1.6).

1.3 History of the United States Nuclear Waste Management Policy

As revealed by the Fermi paper, *The Future of Atomic Energy*, there was little focus on nuclear waste management at the dawn of nuclear power as efforts concentrated on maturing reactor technology to provide safe and reliable electricity (1.2). Perhaps the so-called "back-end" of the nuclear fuel cycle was largely ignored because the popular belief was that the U.S. would quickly close the fuel cycle and greatly reduce the volume of waste requiring disposal. Whatever the reason, the majority of the early waste management research came in the form of demonstration fast reactors and reprocessing facilities. The few waste disposal investigations conducted were largely unfocused and some went without sufficient scientific development to fairly assess their viability. Today's policy makers would prefer to first quantify the attractiveness of waste management strategies in a value tree approach as shown in Figure 1.6, which implies the best waste management strategies would require (a) superior functionality, (b) sufficient safety, and (c) competitive economics. With this framework in mind, we will consider some of the early disposal concepts and step through the waste management history.

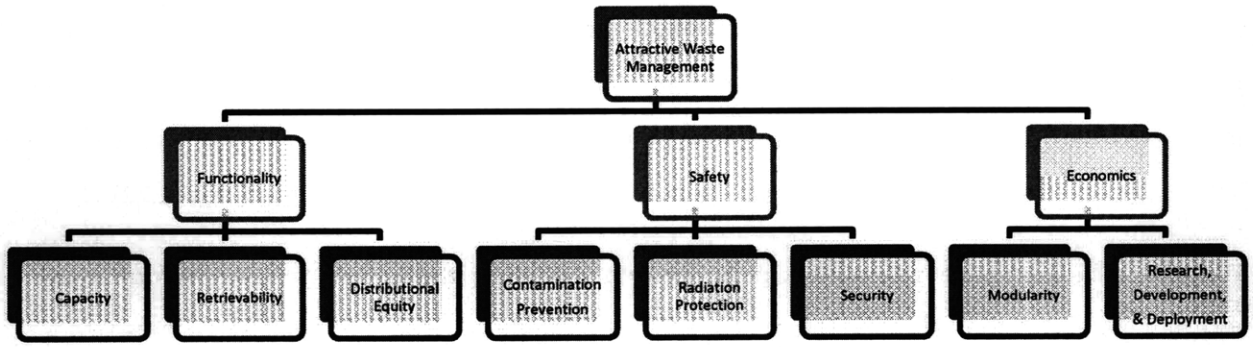


Figure 1.6. Decision Making Value Tree Showing Components of Attractive Waste Management.

While the success of waste management methods affects all nations, there is little international agreement on waste management policy and differing preferences on disposal alternatives. This international debate is reflected in the history of domestic waste solutions including disposal in space, deep seabed, ice sheets, and various geologic repository concepts (1.14). The ultimate goal of waste management is to safely separate the possible interactions of harmful radioactive waste with the human population. While the space solution would completely remove the waste from the environment, the extraordinary costs and relatively high launch risks made the concept impractical (1.15). Deep seabed disposal was considered technically viable, but the 1972 "London Dumping Convention" and the so-called "Law of the Sea," initiated in the 1950s, highly developed in the 1970s and 1980s, and finally

entered into force in 1994, ultimately banned the option even though the law was strongly contested by the U.S. government (1.16). Similarly, technical concerns about global climate variability and international treaties prevented the development of ice sheet disposal. These less accepted approaches were reviewed in some detail in 1974, but more than three decades later their technical feasibility is still in question (1.17; 1.18).

In 1970 the Atomic Energy Commission (AEC) established an official waste management policy in the publication of Appendix F to 10 CFR Part 50, but several point to the ineffectual formulation of the early policy (1.19). Shortly thereafter, the AEC promoted and developed a detailed strategy analyzing the safe utilization of long-term dry storage in the Retrievable Surface Storage Facility (RSSF), but within a few years of its conception the government reversed its support of the RSSF program, Congress replaced the AEC with the Energy Research and Development Administration and then the Department of Energy, and President Carter expressed the need for a waste management policy that avoided transferring the waste burden onto future generations (1.20; 1.21). Although the president rejected the concept of indefinite monitored dry storage as the only option, the fact that spent fuel pools were filling forced utilities to resort to this on a smaller scale at individual reactor sites. Utilities will likely continue this practice and vendors will support these operations, but this routine is expensive and interests opposed to the expansion of nuclear power use the current dry storage practice as evidence the industry has little impetus toward the implementation of a socially acceptable waste management solution (1.22).

The remaining alternative is a stable geologic repository, of which the buried salt bed host formation received the early approval of the National Academy of Sciences (1.21). After political conflicts eliminated possible sites in Michigan and Ohio, in 1970 resources were dedicated to the development of a salt mine near Lyons, Kansas as the nation's primary high level waste repository. Almost as quickly as the AEC announced the site, the discovery of facility integrity compromises from old oil and gas exploration excluded its use, forcing the AEC to abandon the project in 1972. Other nations such as Germany and the Netherlands still pursue disposal in salt and such facility concepts continue to promise adequate long-term radioactivity protection (1.23; 1.24). Even recently, the Waste Isolation Pilot Plant in New Mexico began receiving the government's defense related transuranic waste in a stable salt formation created 250 million years ago by the evaporation of the Permian Sea.

The 1974 Schneider and Platt report also reviewed various deep borehole disposal options, one of which Figure 1.7 illustrates, summarizing how the oil drilling experiences could be applied to nuclear waste management (1.17). In this design, sufficiently aged radioactive waste would be transported to a

repository zone that could accommodate enough holes to store the nation's entire accumulated waste, or somewhat fewer holes that were sufficient enough to store just a region's waste. The depth of each hole should allow the fuel to stay significantly below regions of significant geologic activity and provide a sizeable radiological transportation barrier above the fuel canisters. The immediately recognizable benefits of a deep borehole approach include a nearly continuous modularity of waste capacity and the ability to perform fully functional demonstration facilities at a relatively low investment (1.25). The modularity allows costs to accrue later in the future when the repository needs exist and allows the distribution of repositories to more closely reflect the distribution of beneficiaries of nuclear power without a significant increase in total system cost and, in fact, at a significant discount in transportation costs.

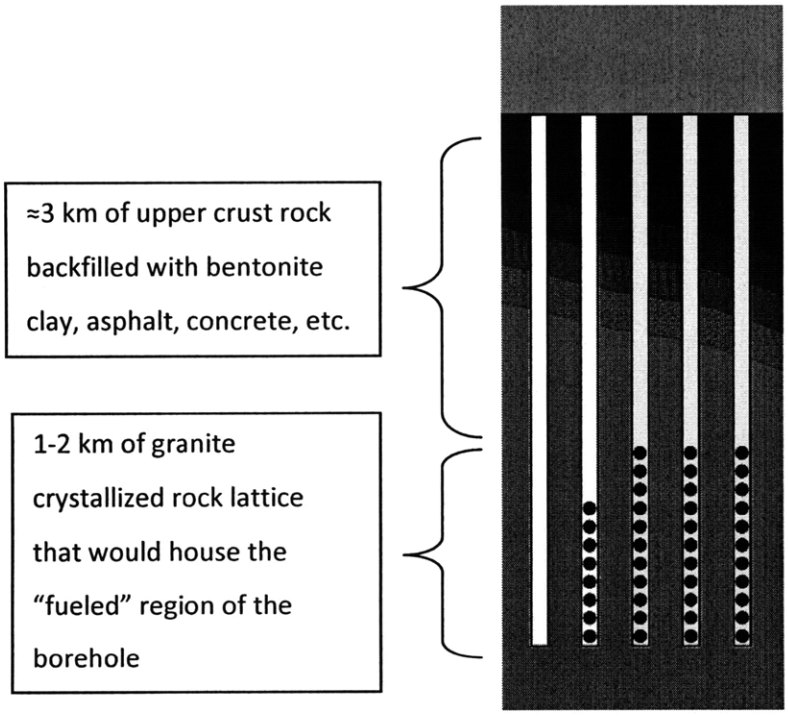


Figure 1.7. Deep Borehole Illustration Showing Fueled and Backfilled Regions (1.26).

The signing of the 1982 Nuclear Waste Policy Act (NWPA) legally required these investigations into repository designs and contractually obligated the Department of Energy (DOE), through the Office of Civilian Radioactive Waste Management (OCRWM), to begin receiving waste in 1998 in exchange for a one mill per electrical kilowatt-hour (0.1 ¢/kWh_e) fee charged to the utilities operating nuclear power plants and paid for by the electricity consumers. DOE retroactively charged this fee for nuclear power generated before the signing of the law and agreed to proceed with the development of two facilities,

one east and one west of the Mississippi, each with a legislated capacity of 70,000 MTHM in order to enforce the requirement of multiple repositories and avoid burdening any single state with the nation's spent fuel. Political battles over siting forced Congress to pass the 1987 Amendments and renege on developing an eastern repository. The new policy forced the OCRWM to completely focus efforts on the Yucca Mountain Project located near the Nevada Test Site and about 90 miles from Las Vegas, Nevada (1.21; 1.27; 1.28).

Volcanic eruptions 10 to 15 million years ago formed the successive layers of ash shaping Yucca Mountain. The repository, shown in Figure 1.8, sits as an underground network of access tunnels and emplacement drifts 1,000 feet below the surface and 1,000 feet above the water table (1.29; 1.30). The dry climate makes Yucca Mountain appealing because the scarcity of water reduces corrosion rates and presumably allows many of the engineered barriers to function and the total system to safely contain the radioactive materials for up to a million years and keep the reasonably maximally exposed individual under the 350 millirem dose limit (1.31; 1.32). Though Congress limited Yucca's capacity to 70,000 MTHM, many suspect the repository can technically handle expected U.S. waste accumulation without a nuclear renaissance, but expanding the allowed capacity requires an additional amendment to the NWPA (1.33).

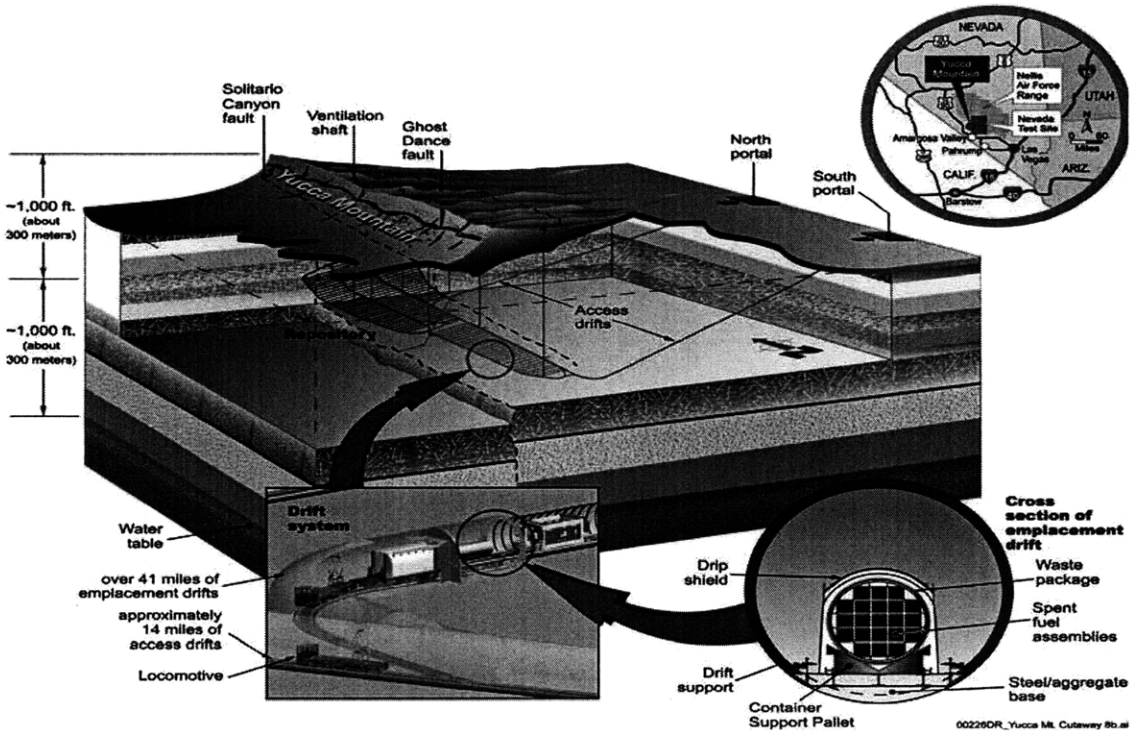


Figure 1.8. Yucca Mountain Repository Illustration and Basic Overview (1.29).

Clearly DOE did not meet its contractual obligation to take possession of utility spent fuel starting in 1998. However, OCRWM expects Yucca Mountain will continue as scheduled and could begin receiving fuel as early as 2020. A changing presidential administration leaves the fate of the nuclear industry, in particular the waste management policy, up in the air. Whatever the outcome of the next presidential election, the OCRWM faces overwhelming challenges of a damaged public perception from a history of false-starts in U.S. nuclear waste management, summarized in Table 1.2, which illustrates a bad habit of choosing "winners" in repository concepts before adequate technical analysis identifies the best design alternative. We should expect a waste management strategy dictated by political winds will change with the election cycles and continue the delays experienced thus far. The economic effects of these delays grow larger each year and the problem simultaneously impedes the "renaissance" movement (1.19; 1.34).

Table 1.2. U.S. High Level Waste Management Condensed History (1.35; 1.21; 1.24).

Year	Significant Event or Policy Initiative
1946	Atomic Energy Commission assumes control of nuclear energy regulation
1957	National Academy of Sciences recommends high level waste disposal in salt formations
1963 - 1967	Studies conducted on salt vault to be located near Lyons, Kansas
1970	AEC tells Idaho that all waste currently stored there will be sent to Lyons facility by 1980
1972	AEC discards Lyons salt mine project and promotes Retrievable Surface Storage Facility
1974	Nuclear Regulatory Commission takes over regulation after AEC shuts down
1975	RSSF turned down, geologic disposal and Waste Isolation Pilot Plant promoted
1980	Carter Administration rejects WIPP
1981	Department of Energy reinstates WIPP for government transuranic waste
1982	Nuclear Waste Policy Act --development of east and west repositories --government must begin accepting spent nuclear fuel in 1998
1987	NWPA Amendments redirects repository efforts entirely on Yucca Mountain site
2002	Bush recommends the Yucca Mountain site
2008	Office of Civilian Radioactive Waste Management submits YMP license application to NRC

Like a bus that arrives late to its stop only to find a larger than normal crowd requiring a longer boarding time and causing further delays, the problem of accumulating nuclear waste drives a vicious reinforcing feedback loop that also inhibits the expansion of nuclear power, illustrated in Figure 1.9. Interim storage offers policy makers an opportunity to reevaluate waste management options while removing the

management burden from the utilities. However, many oppose the idea on the grounds that interim storage without an agreeable ultimate waste solution allows the government to utilize surface dry storage indefinitely while engineers and scientists debate facility concepts. To prevent this, some states legally prohibit further development of nuclear power until the government moves forward with a permanent solution (1.36). In the mean time, utilities continue to sue the government for billions of dollars because of the failure to take possession of the spent nuclear fuel as promissed.

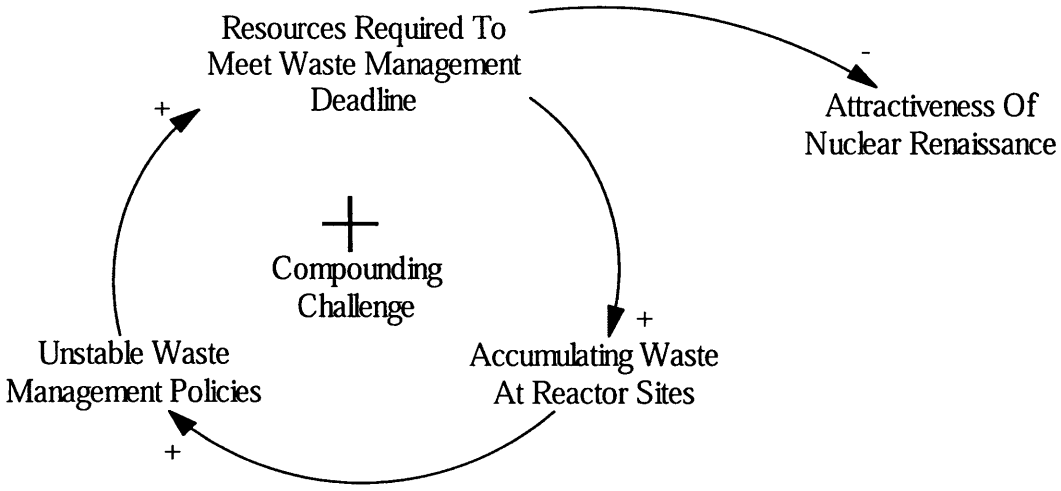


Figure 1.9. Vicious Reinforcing Loop Illustrating Compounding Nature of Waste Accumulation.

Chapter 1 References

1.1. **Office of the Assistant Secretary for Nuclear Energy.** *The First Reactor - 40th Anniversary Commemorative Edition (DOE/NE--0046).* Washington, DC : U.S. Department of Energy, 1982.

1.2. **Fermi, Enrico.** *The Future of Atomic Energy (MDDC--1).* Oak Ridge, TN : U.S. Atomic Energy Commission (now Department of Energy and Nuclear Regulatory Commission), 1946.

1.3. **Office of History and Heritage Resources.** Production Reactor (Pile) Design, 1942. *The Manhattan Project: An Interactive History.* [Online] U.S. Department of Energy. [Cited: May 24, 2008.] http://www.cfo.doe.gov/me70/manhattan/reactor_design.htm.

1.4. **Lamarsh, John R and Baratta, Anthony J.** *Introduction to Nuclear Engineering.* 3rd Edition. Upper Saddle River, NJ : Prentice Hall, 2001. ISBN 0-201-82498-1.

1.5. **Murray, Raymond L.** *Nuclear Energy: An Introduction to the Concepts, Systems, and Applications of Nuclear Processes.* 5th Edition. Woburn, MA : Butterworth-Heinemann, 2001. ISBN 0-7506-7136-X.

1.6. **Energy Information Administration.** *Annual Energy Review.* Washington, DC : U.S. Department of Energy, 2006.

1.7. *Fuel and Operational Economics.* **Pilat, Ed E.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2007. 22.251 Systems Analysis of the Nuclear Fuel Cycle.

1.8. **Nuclear Energy Agency.** *Uranium 2007: Resources, Production, and Demand.* International Atomic Energy Agency, Organization for Economic and Cooperative Development. s.l. : OECD Publishing, 2007. ISBN 9789264047662.

1.9. *Lecture on "Fast Reactors".* **Driscoll, Michael J.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2007. 22.251 Systems Analysis of the Nuclear Fuel Cycle.

1.10. **Keystone.** *Nuclear Power Joint Fact-Finding.* Keystone, CO : The Keystone Center, 2007.

1.11. **Energy Watch Group.** *Uranium Resources and Nuclear Energy.* 2006. EWG-Series No 1/2006.

1.12. **U.S. Department of Energy.** *Nuclear Fuel Data.* [Database] Washington, DC : s.n., 2002. RW-859.

1.13. **Massachusetts Institute of Technology.** *The Future of Nuclear Power.* Cambridge, MA : MIT Press, 2003.

1.14. *High-Level and Long-Lived Radioactive Waste Disposal.* **Angino, Ernest E.** 4320, s.l. : American Association for the Advancement of Science, December 2, 1977, Science, Vol. 198, pp. 885-890.

1.15. **Cochran, Robert G and Tsoulfanidis, Nicholas.** *The Nuclear Fuel Cycle: Analysis and Management.* 2nd Edition. La Grange Park, IL : American Nuclear Society, 1999. ISBN 0-89448-451-6.

1.16. *Into the Abyss: International Regulation of Subseabed Nuclear Waste Disposal.* **Kaplan, Robert A.** 3, s.l. : The University of Pennsylvania Law Review, January 1991, Vol. 139, pp. 769-800.

1.17. **Schneider, Keith J and Platt, Anthony M.** *High-Level Radioactive Waste Management Alternatives.* Richland, WA : Battelle Pacific Northwest Laboratories, 1974. vols. 1-4. BNWL-1900.

1.18. **Idaho National Laboratory.** *Advanced Fuel Cycle Cost Basis.* Idaho Falls, ID : US Department of Energy, 2008. INL/EXT-07-12107.

1.19. **Vance, Scott A.** *The Ramifications of a Delay in the National High-Level Waste Repository Program.* Nuclear Science and Engineering. Cambridge, MA : Massachusetts Institute of Technology, 1988. SM Thesis.

1.20. **Atlantic Richfield Hanford Co.; Kaiser Engineers.** *Retrievable Surface Storage Facility Conceptual System Design Description.* 1977. ARH-LD-140.

1.21. *Nuclear Waste Management.* **Lester, Richard K.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2008. 22.812 Managing Nuclear Technology.

1.22. *Used Nuclear Fuel-- What will we do with it?* **Hanson, Alan S.** Cambridge, MA : Massachusetts Institute of Technology, 2006. Nuclear Science and Engineering Seminar Series.

1.23. **Nuclear Energy Agency.** *Advanced Nuclear Fuel Cycles and Radioactive Waste Management.* Paris, France : Organization for Economic Cooperation and Development, 2006. ISBN 92-64-02485-9.

1.24. **Saling, James H and Fentiman, Audeen W.** *Radioactive Waste Management.* 2nd Edition. New York, NY : Taylor & Francis, 2002. ISBN 1-56032-842-8.

1.25. *The Use of Deep Boreholes For High Level Nuclear Waste Disposal.* **Driscoll, Michael J.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2007. 22.77 Nuclear Waste Management.

1.26. *The Feasibility of Deep Geological Boreholes for Disposal of Minor Actinides.* **Pierpoint, Lara M.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2007. 22.251 Systems Analysis of the Nuclear Fuel Cycle.

1.27. *High Level Wastes of the Nuclear Fuel Cycle.* **Kazimi, Mujid S.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2007. 22.251 Systems Analysis of the Nuclear Fuel Cycle.

1.28. *Nuclear Waste Disposal.* **Tulenko, James S.** Gainesville, FL : UF Nuclear and Radiological Engineering Department, 2005. ENU5186 Nuclear Fuel Cycles.

1.29. *The Role of Science in Assessing the Long Term Performance of a Geologic Repository at Yucca Mountain, Nevada.* **Dyer, R J.** Las Vegas, NV : Department of Energy, 2006. Office of Civilian Radioactive Waste Management Graduate Fellowship Conference.

1.30. **Cochran, Robert G and Tsoulfanidis, Nicholas.** *The Nuclear Fuel Cycle: Analysis and Management.* 2nd Edition. La Grange Park, IL : American Nuclear Society, 1999. ISBN 0-89448-451-6.

1.31. *Repository Requirements and Design.* **Nutt, W M.** Argonne, IL : Argonne National Laboratory, 2007. Nuclear Engineering Division Summer Seminar Seriers.

1.32. *Risk-Informed Regulation for a High-Level Waste Repository at Yucca Mountain.* **Ghosh, S T.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2007. 22.39 Integration of Reactor Design, Operations, and Safety.

1.33. *The Yucca Mountain Repository for Nuclear Waste.* **Sproat, Edward F.** Cambridge, MA : Massachusetts Institute of Technology, 2007. Nuclear Science and Engineering Seminar Series.

1.34. **Siegel, Matthew J.** *Economic Ramifications of a Delay in the National High-Level Waste Repository Program.* Nuclear Science and Engineering. Cambridge, MA : Massachusetts Institute of Technology, 1989. SM Thesis.

1.35. **Macfarlane, Allison M and Ewing, Rodney C.** *Uncertainty Underground: Yucca Mountain and the Nation's High-Level Nuclear Waste.* Cambridge, MA : MIT Press, 2006. ISBN 0-262-63332-9.

1.36. *Nuclear Energy.* **Holdren, John P.** Cambridge, MA : Harvard John Fitzgerald Kennedy School of Government, 2008. ENR302 Energy Policy: Technologies, Systems, and Markets.

Chapter 2:
Waste Management Evaluation Methodology

Introduction

This chapter illustrates the use of System Dynamics principles to map a simplified waste management system's spent fuel stocks and flows. The Vensim software, developed by Ventana Systems, assists users in building these basic models and simulating the system's behavior. A series of charts illustrate the results one might expect to see in a waste management system that only considers a single reactor generating nuclear waste under a hypothetical scenario that includes the use of wet storage, onsite dry storage, interim storage, and a final repository. This chapter and the use of this simplified model serves as a proof-of-principle in using the System Dynamics concepts to characterize the transport of spent nuclear fuel through the waste management system.

2.1 Stocks and Flows in a System Dynamics Framework

Established in the 1960s, the field of System Dynamics attempts to explain unexpected behaviors created by structures of complex systems. In real systems, cause and effect appear distant in space and time yet policy makers assume unrealistically tightly coupled relationships. System Dynamics fundamentally believes models only require stocks and flows to define all systems and that closed causal links of variable dependence define loops of behavior feedback. Properly defined models contain feedbacks that many decision makers fail to appreciate and which identify modes of policy resistance that would lead to ineffective results. These models represent "low-cost laboratories" providing invaluable insight and intuition for open-minded problem solvers (2.1).

A distinction exists between dynamic complexity and combinatorial complexity. The former manifests itself as a system with competing feedback loops and long time delays, which makes predicting system performance challenging. Massive systems containing many variables characterize combinatorial complexity. Models of this type provide challenges in solving and understanding, but analysts should avoid relying upon system dynamics software to explore these problems because they require intense logic structures that may overwhelm and confuse the model. Analysts only benefit from system dynamics software when the model illustrates the complex relationships effectively.

To demonstrate how systems thinking applies to waste management, we look to the important stocks defining the problem for a single reactor. Stock variables pass the "snapshot test" in which the level remains countable even if one observes a picture of the system and pauses time. In this case, fuel in the reactor, used fuel in the spent fuel pool, used fuel in dry storage, used fuel in transportation, used fuel in interim storage, and used fuel in repositories represent stocks of interest. Figure 2.1 illustrates the basic stock structure of this system where the cloud feeding into the reactor represents a condition of no constraint on supply. In other words, this report assumes the operation of nuclear power plants remains constant until the license expiration and that externalities, such as a possible uranium resource scarcity, do not affect the fleet's generation of power. One should recognize the strictly enforced consistency among the units of each connected stock. Our choice of stocks raises an important point about aggregation that not every stock needs tracking. In our example, we need not track material in transit explicitly. The flow of material between stocks gives sufficient information about the dynamics because spent fuel generally remains in transit for only a few days or weeks and a one-year simulation time step completely glosses over that level of detail. On the other hand, the stock of fuel in the spent

fuel pool needs to explicitly distinguish fuel that the reactor recently discharged from fuel with sufficient cooling time for handling, shown in Figure 2.2.

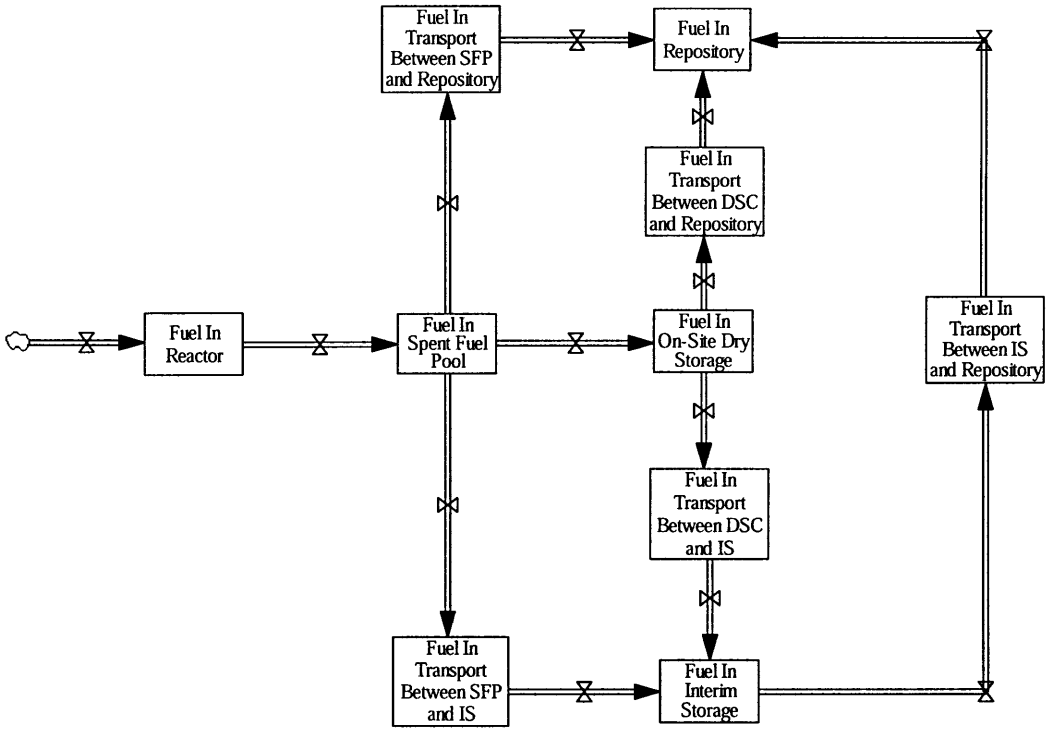


Figure 2.1. Basic Waste Management Stock Structure Including Material In Transit.

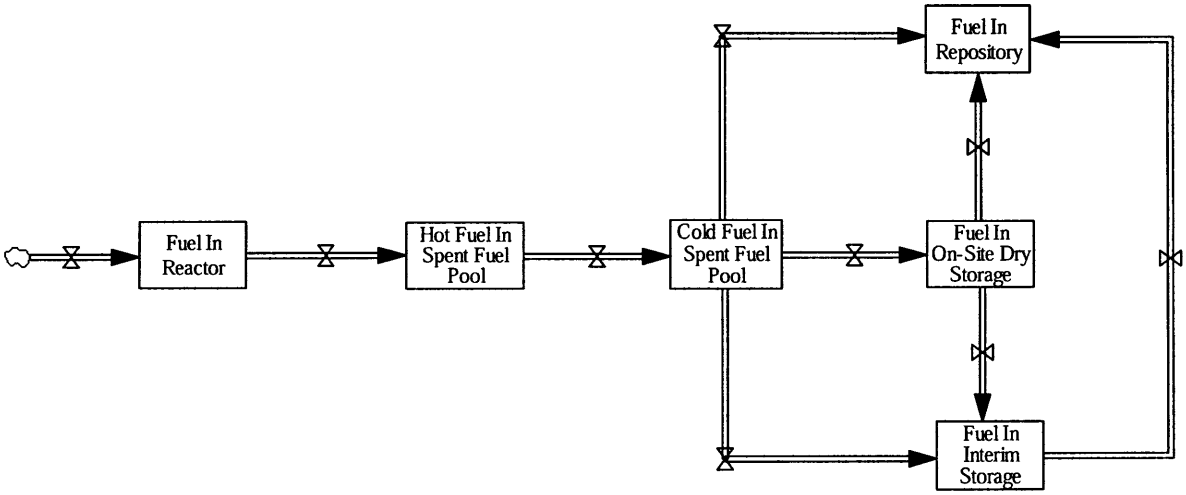


Figure 2.2. Revised Waste Management Stock Structure With Relevant Detail.

The valves between stocks represent rates of flow and, like the rate at which water flows through a sink faucet, these quantities fail the "snapshot test" and have no measurable value to an observer who pauses time by taking a single picture. In a traditional system dynamics model, the stocks in the system define the flows and the flows integrate to define the stocks, however most functional and useful models include auxiliary variables, such as time constants, and include flow definitions based on other flows, as illustrated in Figure 2.3. Even in this simplified representation of the stock and flow structure with causal link feedbacks, the detail begins to cloud the communication of the system, making interpretation of system response challenging.

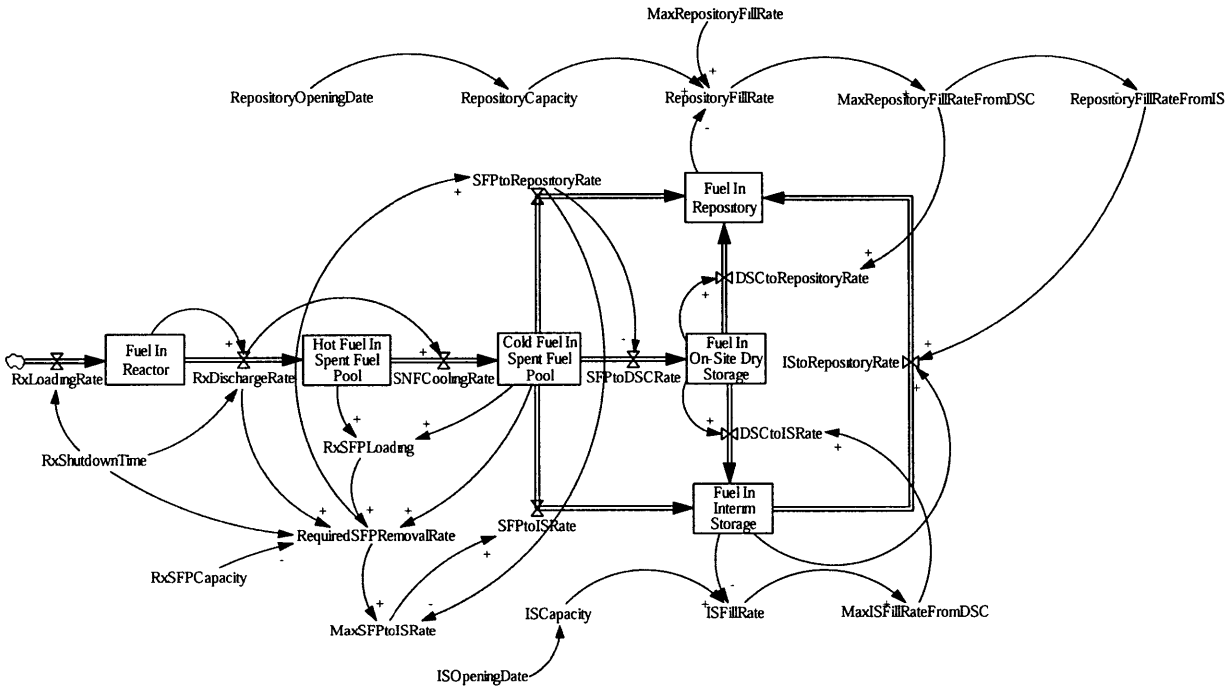


Figure 2.3. Simplified Stock and Flow Structure Including Causal Links.

In this thesis, the general approach to the model definitions breaks the fuel management into stages. First, the reactor discharges fuel into the spent fuel pool and the cooling process commences. At this point, one must monitor the capacity of the spent fuel pool and determine when the site must unload old fuel to make room for future fuel. The second stage works backward and basically states that if the repository exists and capacity exists to receive fuel, planners must first send fuel from required spent fuel pool discharges, then from onsite dry storage, interim storage, and finally from the remaining fuel cool enough for handling in the spent fuel pool. The third stage ultimately follows the same logical structure, but for the interim storage loading. The fourth and final stage sends the remaining required

spent fuel discharge into onsite dry storage casks. Appendix A details a basic working model for a single reactor waste management system, including documentation of the important stocks and flows in this system with sample values for illustration purposes provided in Table 2.1. This model was created in Vensim, a System Dynamics programming software developed by Ventana Systems.

Table 2.1. Summary of Basic Model Parameters Assumed in Illustrative Vensim Model.

Variable	Value	Units
Simulation Start Time	0	year
Reactor Shut Down Time	50	year
Fuel In Reactor Core When Online	100	MT
Minimum Spent Fuel Pool Cooling Time	10	year
Hot Fuel Initially In Spent Fuel Pool	250	MT
Cold Fuel Initially In Spent Fuel Pool	100	MT
Spent Fuel Pool Capacity	500	MT
Required Free Capacity During Reactor Operation*	100	MT
Reactor Loading/Unloading Rate	25	MT/year
Interim Storage Opening Time	35	year
Interim Storage Capacity	500	MT
Interim Storage Maximum Loading Rate	100	MT/year
Repository Opening Time	40	year
Repository Initial Capacity	1,000	MT
Repository Maximum Loading Rate	50	MT/year
Repository Expansion Time	65	year
Repository Expanded Capacity	2,000	MT

Figure 2.4 and 2.5 show the level of stocks and flows in the reactor and its spent fuel pool. Notice the amount of fuel in the reactor remains constant for the first 50 years because the level of the fuel in the core of the reactor exists in dynamic equilibrium, the reactor loading rate equals the reactor discharge rate. This simulation hypothesized a reactor shuts down at time equal to 50 years, at which point the spent fuel pool must unload as quickly as possible, while still allowing the spent fuel to cool for a specified 10 years. Notice that the level of fuel in the spent fuel oscillates between 325 and 400 MT. This shows how the system tries to reach equilibrium, while simultaneously reconciling the required free capacity of the spent fuel pool during reactor operation. In this simulation, we require that the spent

* 100 MT corresponds to the mass of the fuel in the core.

fuel pool must be able to accommodate a full core discharge at any time and that the spent fuel pool discharge increments equal the total mass in the core, but a modeler can easily modify these requirements to fit any conditions of interest. When the spent fuel pool reaches 400 MT, the reactor site must discharge spent fuel to a repository, interim storage, or onsite dry storage.

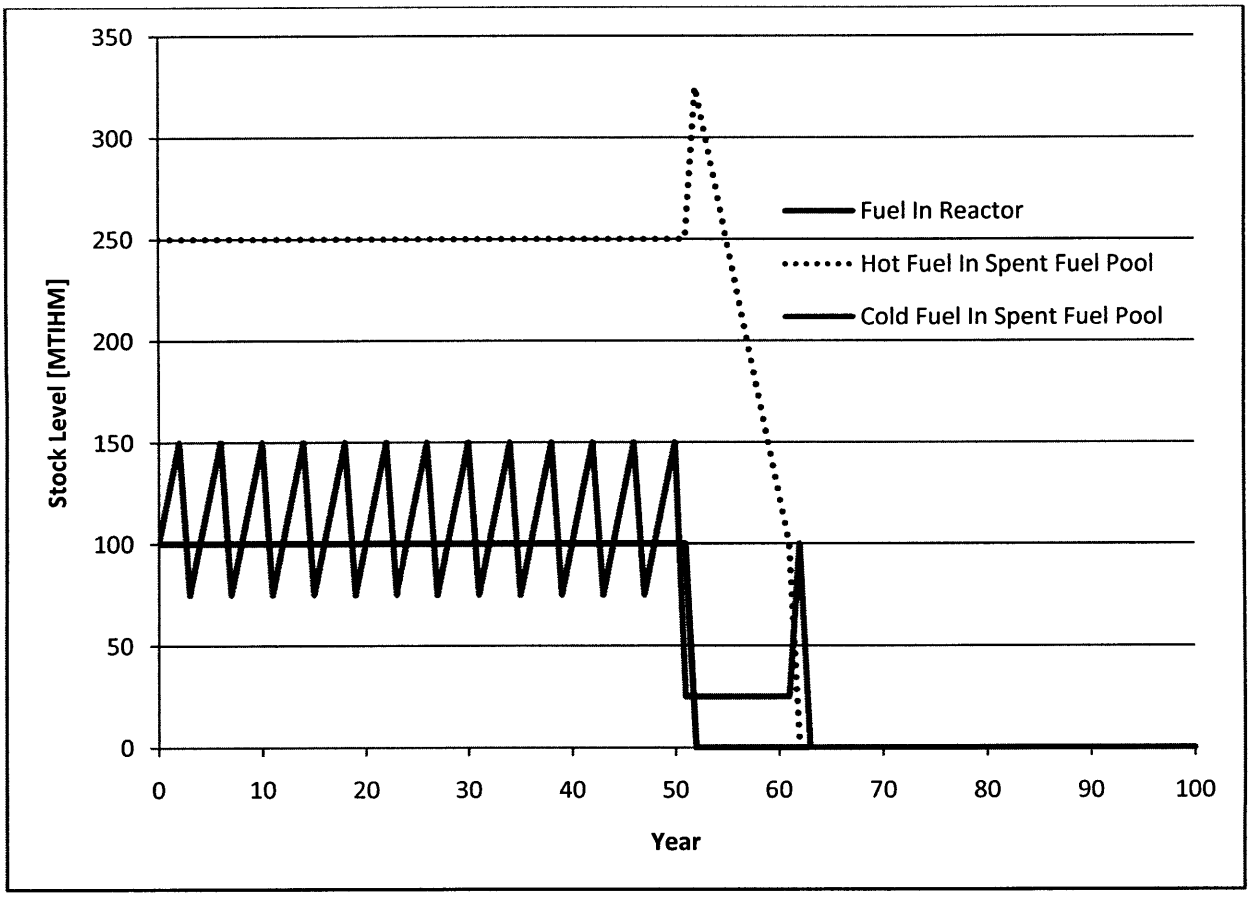


Figure 2.4. Sample Levels of Fuel in the Reactor and Spent Fuel Pool.

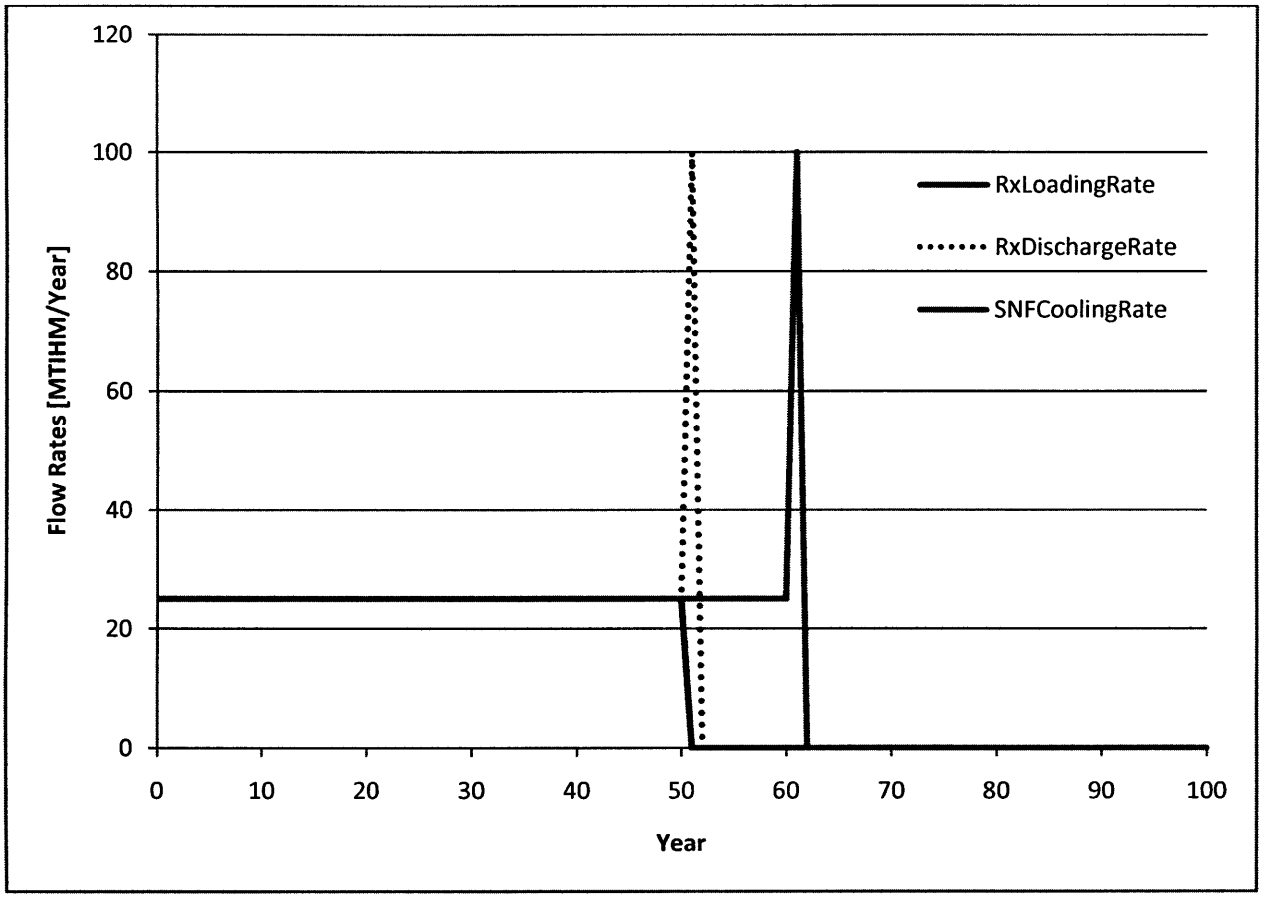


Figure 2.5. Sample Flow Rates into the Reactor and Spent Fuel Pool.

The second stage deals with the filling of the repository, shown in Figures 2.6 and 2.7. Notice the repository opens in year 40 and temporarily levels off at year 60 because the repository reaches the 1,000 MTIHM initial capacity. This capacity doubles to 2,000 MT at the year 65 and so the repository loading rate resumes at that time. Although not shown, the managers could conceive of a situation in which the maximum fill rates change over time. Figure 2.7 illustrates the constant fill rate by showing how the different inflows to the repository trade off over time in order to sum to the desired 50 metric tonne per year value. The model defines the decision process to determine from which stock to load the repository and already one begins to see the combinatorial complexity that exists in a system of 104 active reactors feeding into a limited number of interim storage and repository sites. The level of the stock in the repository never reaches the capacity expansion of 2,000 MT because the single reactor system only generates 1,725 MT of spent nuclear fuel before shutting down. One can imagine a scenario in which the stock reaches this capacity and waste managers must continue utilizing interim storage sites and onsite dry storage indefinitely.

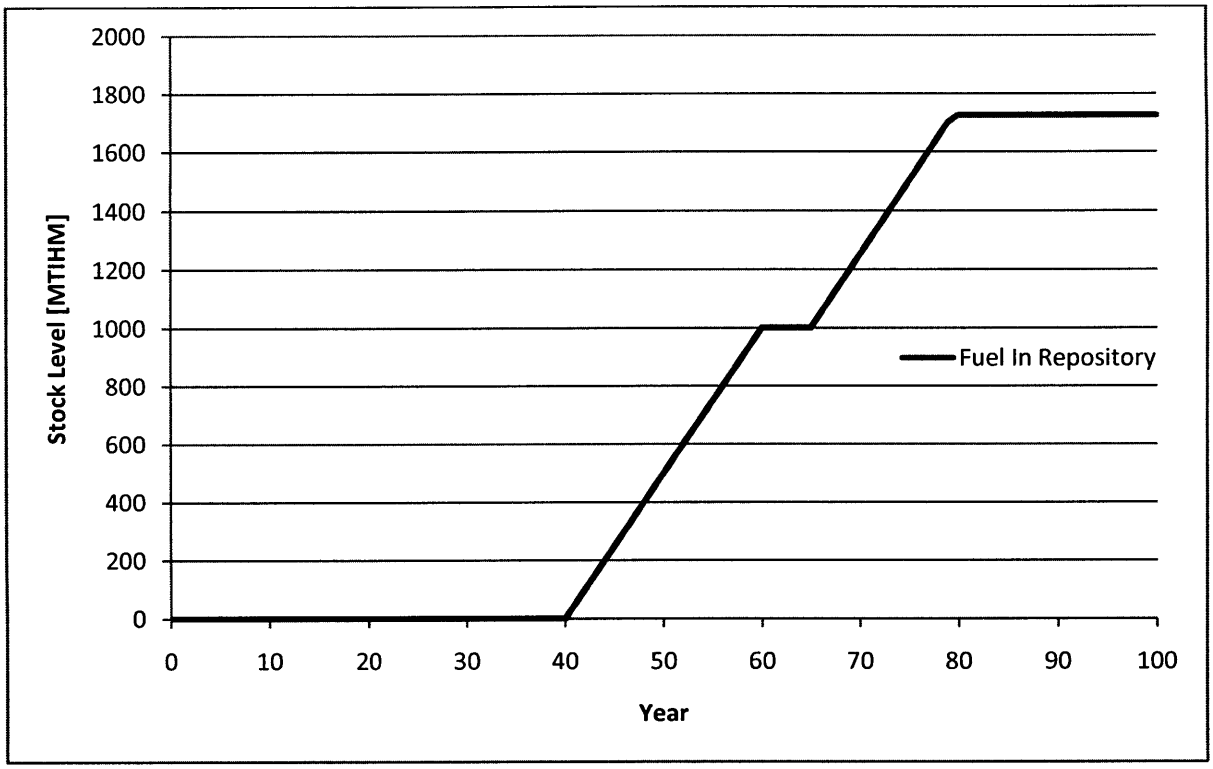


Figure 2.6. Sample Level of Used Fuel in the Repository.

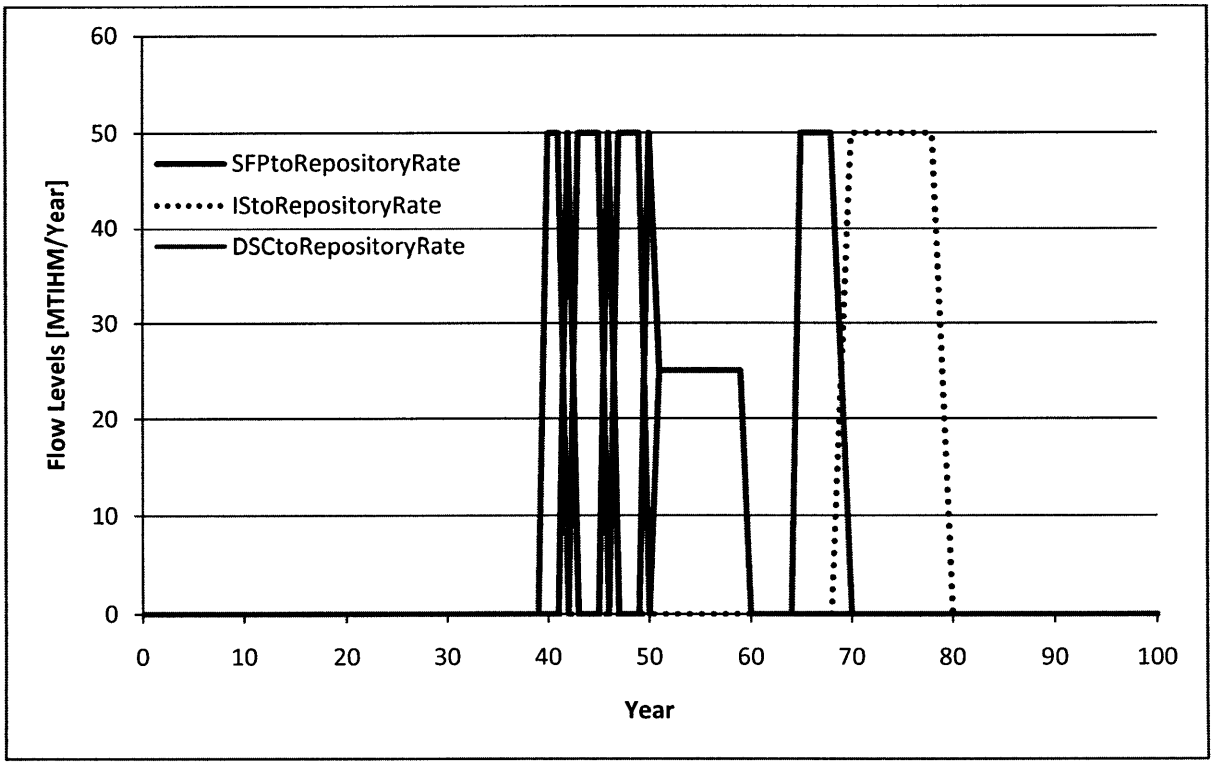


Figure 2.7. Sample Flow Rates into the Repository.

Figures 2.8 and 2.9 illustrate the interim storage dynamics. Notice the facility opens in the year 35, has a lower capacity of 500 MT, and a higher maximum fill rate of 100 MT/year. In this example, the interim storage fills to capacity by the time the repository opens and begins depleting its supply to the repository after the spent fuel at the reactor site gets removed. Notice that the sum of the flows into interim storage never exceeds 100 MT/year in Figure 2.9. The repository operates in maximum receiving mode when not at capacity, but the interim storage site acts as a buffer to the repository and can theoretically oscillate between a loading mode and an unloading mode as needed.

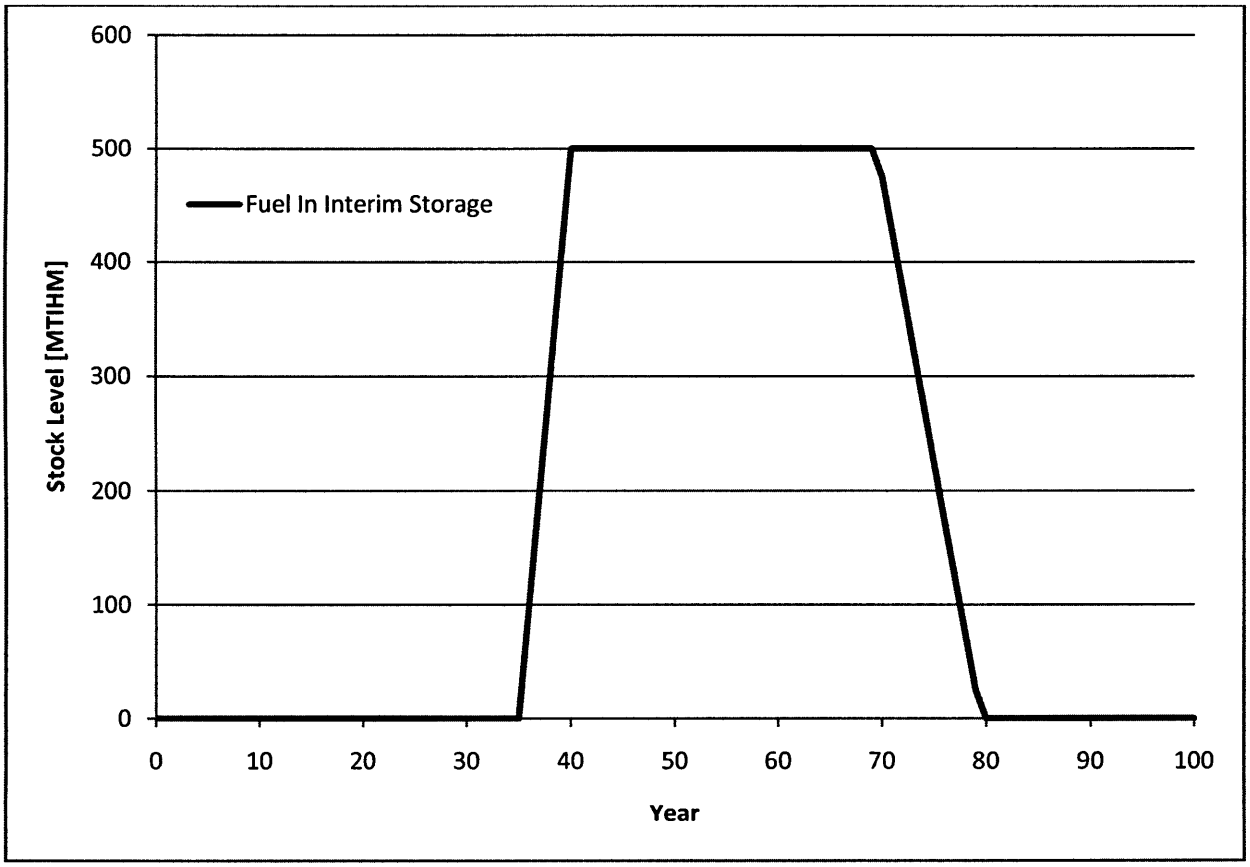


Figure 2.8. Sample Level of Used Fuel in Interim Storage.

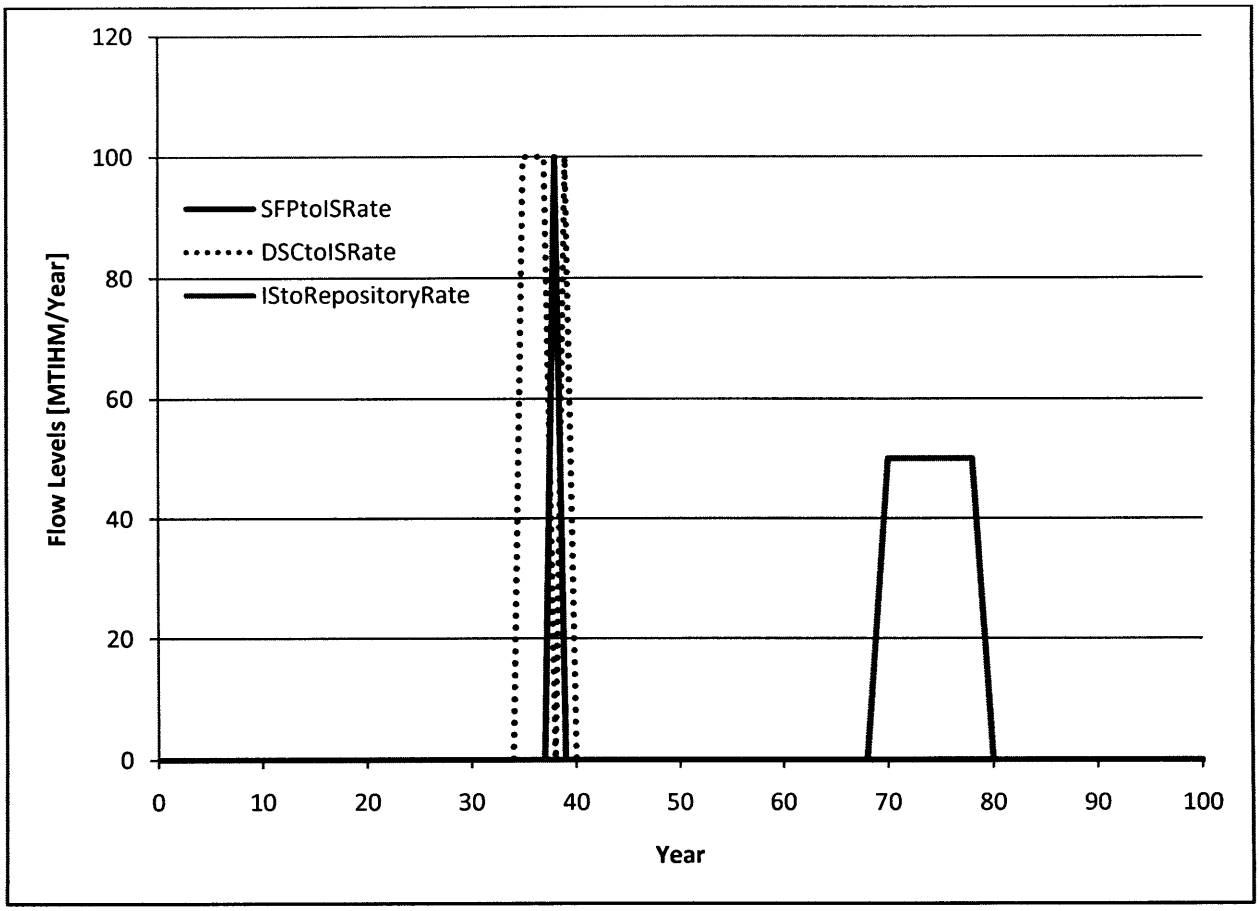


Figure 2.9. Sample Flow Rates In and Out of Interim Storage.

Similar to the interim storage, the onsite dry storage casks operate as a buffer to both the repository and the interim storage facility. The amount of fuel in dry storage fluctuates more drastically, as shown in Figures 2.10 and 2.11, because unloading algorithm assumes the use of centralized interim storage and repository storage can be done much more cost effectively.

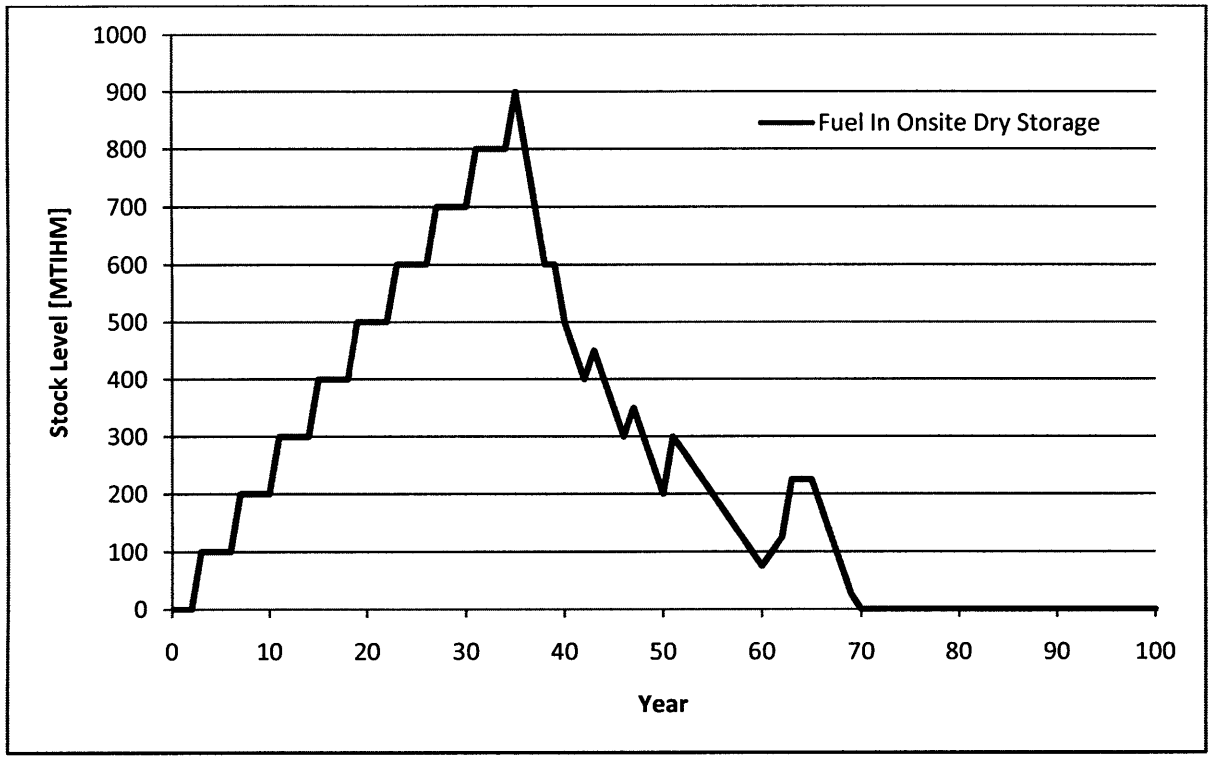


Figure 2.10. Sample Level of Used Fuel in Onsite Dry Storage Casks.

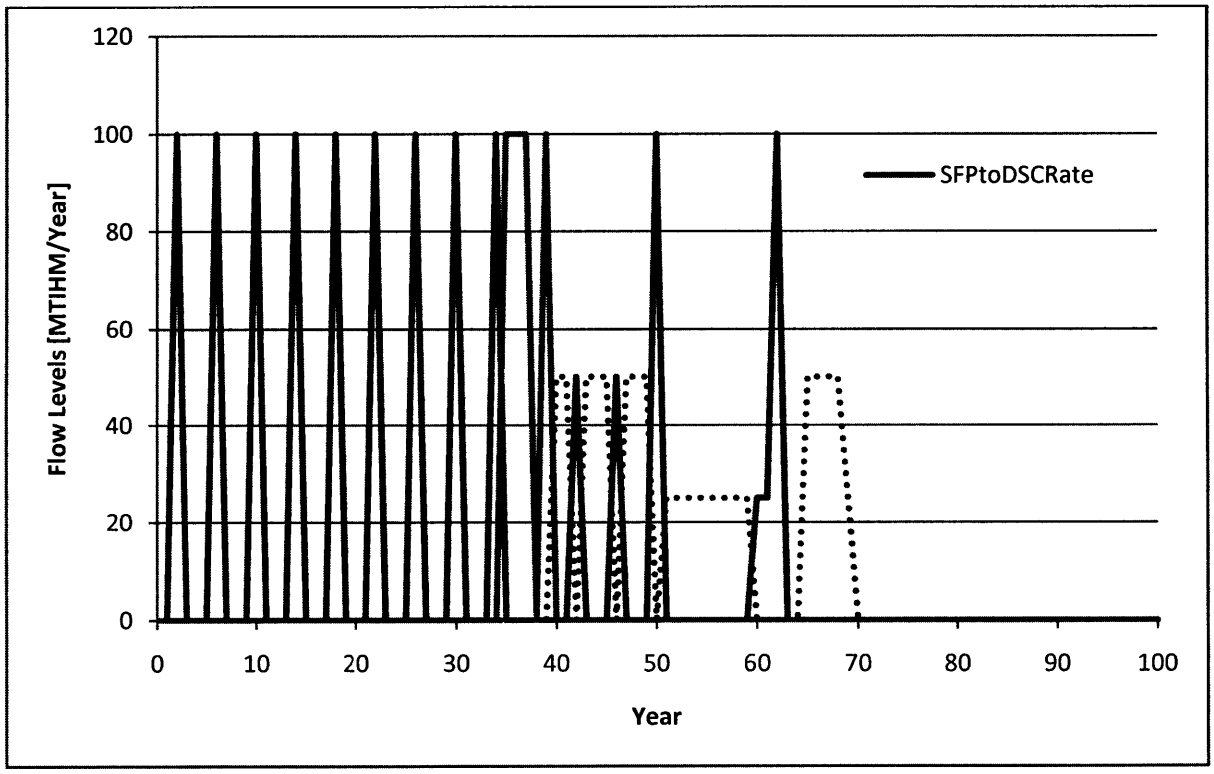


Figure 2.11. Sample Flow Rates In and Out of Onsite Dry Storage Casks.

Figures 2.4 to 2.11 illustrate the necessary structure and behavior that a complete waste management program must capture. Although some dynamic complexities exist, the combinatorial complexities of the U.S. waste accumulation problem far outweigh the dynamic challenges and so the situation likely extends beyond the useful application of the Vensim System Dynamics software. The full problem must manage fuel from over 100 reactors at roughly 75 reactor sites and constantly optimize the unloading strategy based on the status of the sites themselves. Part of this work includes the development of a fundamental computer language waste management program, named SNuFManager (**S**pent **Nu**clear **F**uel **Manager**), using Fortran90. The program solves a generalized problem with dynamic memory allocation to fill reactor data arrays of any size based on a user defined input file. Appendix B reveals the commented Fortran program and its subroutines, Appendix C a basic user's manual for creating input files, and Appendix D provides sample input files used in this report.

Although the general problem illustrated by the Vensim model accurately depicts the management strategy, SNuFManager must include a few additional layers of complexity. For instance, if three reactors once operated on a single site and two no longer produce electricity, an intelligent program identifies the reactor site as possessing an "online" status because a utility experiences the consequences of higher shutdown waste monitoring expenses only when all reactors on site no longer operate. The program drastically simplifies the problem by assuming all reactors on a given site freely share the site's spent fuel pool storage capacity.

The next layer of complexity comes from the logical operations that continuously prioritize fuel unloading patterns into the repository and interim storage facility. Although Chapter 4 discusses possible benefits of more complex unloading patterns, the most primitive algorithm simply ranks the reactors from oldest to youngest and unloads their spent fuel in that order. This strategy completely neglects the effort to explicitly quantify the site and reactor characteristics, such as the occupancy fraction of the spent fuel pools, the amount of dry storage in use, and the proximity to reactor shutdown that could inform the unloading patterns. This depth in strategy made possible by SNuFManager illustrates the gains in efficiency over the Vensim model when dealing with the management of systems with high combinatorial complexity.

2.2 Modularizing the System Expenses

The complex waste management analysis only shows value when combined with an economic assessment. This work fundamentally approaches the problem by modularizing the expenses as shown in Figure 2.12 and follows the strategy used by the Advanced Fuel Cycle Cost Basis report (2.2). As a matter of practicality, this thesis exercises a restrictive assumption that the conditioning and packaging process for dry storage contains no difference to the process for transportation and that once packaged for dry storage, spent fuel needs no further conditioning for subsequent transportation. In other words, this assumption essentially assumes exclusive use of the so-called "dual purpose" casks and considerably simplifies the problem. Relaxing this assumption merely requires adequate time with little expected impact on the conclusions of this study. Also note the repository conditioning and packaging module ultimately ignores the possibility of widespread use of so-called "multi-purpose" casks intended for dry storage, transportation, and final geologic disposal, which the Department of Energy recently endorsed. In much the same way as editing the assumed use of dual purpose casks, time alone stands in the way of relaxing the assumption against the use of multi-purpose containers should the need arise.

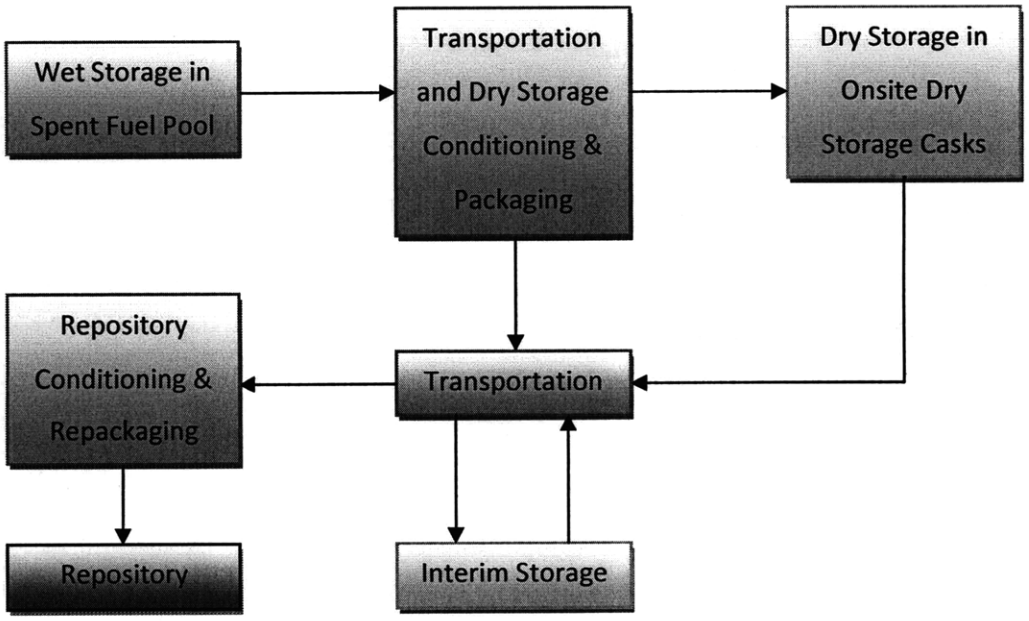


Figure 2.12. Economic Modules and Cost Flow Sheet.

Summary

The simple Vensim model proves the ability to use the concepts of System Dynamics in modeling the stocks and flows of spent fuel from the reactor to ultimate geologic disposal and illustrates the types of behavior expected in the complete system. The use of Vensim must end at the conceptual phase because while System Dynamics modeling packages handle dynamic complexity well, use of a fundamental programming language better suits the challenges of combinatorial complexity seen in the national waste management program. After completing the stock and flow logic, the detailed spent fuel tracker will apply the simple economic modules to estimate annual expenditures used to determine the total system life cycle cost.

Chapter 2 References

2.1. **Sterman, John D.** *Business Dynamics: Systems Thinking and Modeling for a Complex World.* Boston, MA : Irwin McGraw-Hill, 2000. ISBN 0-07-231135-5.

2.2. **Idaho National Laboratory.** *Advanced Fuel Cycle Cost Basis.* Idaho Falls, ID : US Department of Energy, 2008. INL/EXT-07-12107.

Chapter 3:
System Module Details

Introduction

This chapter provides the background on the economic modules with their estimated operation levelized costs that are required to generate the waste management total system life cycle costs. Each section presents the rationale for choosing the input shown in Appendix D, which most often relies on the open literature. In many cases the proprietary nature of vendor fees precludes a highly accurate calculation of the costs and so high and low values accompany the nominal estimates. Because the 2001 report on the Yucca Mountain Project total system life cycle cost inspired this study, this work exclusively expresses costs in constant 2000$ for consistency.

3.1 Wet Storage in Spent Fuel Pools

During construction of nuclear power plants, designers made allowances for discharged fuel storage in spent fuel pools, which were intended to easily store more than 10 years (generally 4.4 cores) of waste, allowing the fission products to decay enough to make handling feasible (3.1). These pools typically contain more than 50,000 ft^3 (≈45 feet long, ≈30 feet wide, ≈40 feet deep) of water and vertically stored the assemblies in the lower 12 to 14 feet in a square array originally loosely packed with a 21 inch center-to-center assembly distance. As pools quickly filled, designers utilized borated storage racks and more elaborate management techniques that allowed for much denser packing of the assemblies in 9 inch arrays, as shown in Figure 3.1. Now spent fuel pool management relies on the burnup experienced in the core to determine the fissile content of each assembly and place the assemblies in optimized patterns to avoid an uncontrolled criticality accident.

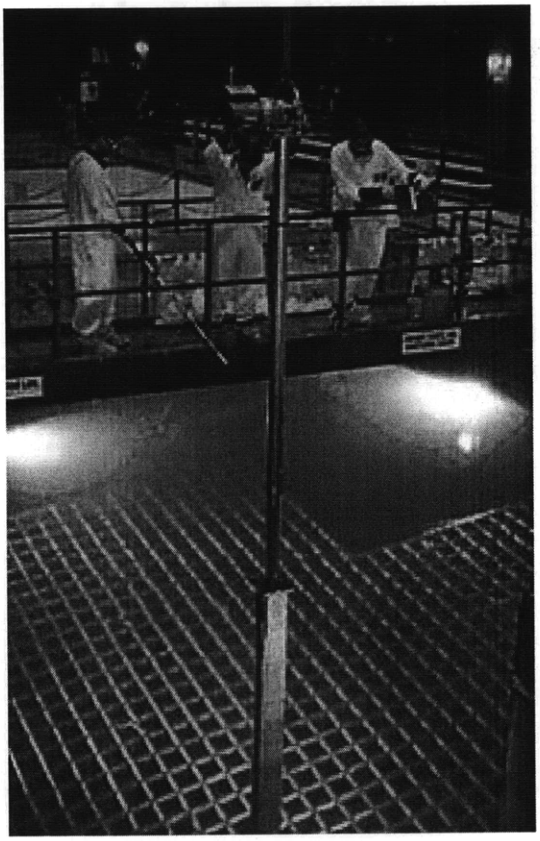


Figure 3.1. Spent Fuel Pool Management at Nuclear Power Plant (3.2).

The price of electricity includes the costs of spent fuel pool management during the normal operating lifetime of the reactor. However, a complete waste management evaluation requires consideration of the cost of active monitoring and maintenance of the spent fuel pool after reactor shutdown. The open literature lacks accurate cost estimates for post-shutdown wet spent fuel storage, but one study presented in 1993 suggests utilities must spend over $10 million a year to properly manage nuclear waste in spent fuel pools of shutdown reactors (3.3).

The Bowser report suggested the cost of wet spent fuel storage after shutdown remained relatively constant for a wide range of pool capacities. This suggests that an intelligent waste management strategy should avoid the use of wet storage at shutdown reactors, but that if a utility needs to store fuel in the spent fuel pool, no incentive exists to minimize the amount of fuel stored in the pool. The cost of operation and maintenance of spent fuel pools depends largely on the cost to secure the facility, manage the operation, and maintain the functionality of the pool and its water treatment systems. The relatively flat cost of wet storage as a function of loading comes about because none of these cost components scale with the amount of fuel in the spent fuel pool. Table 3.1 identifies several estimates reported in the open literature of the cost of wet storage at shutdown reactor facilities.

Table 3.1. Documented Annual Operation and Maintenance Cost Estimates of Spent Fuel Pools.

Cost (M$/year)	Source	Capacity (MT HM)	Notes
10.6	(3.3)	N/A	1993 study, assume 1993$ ($12.6M in 2000$)
2-10[*]	(3.4)	N/A	Assume 2000$[†]
5[‡]	(3.5)[§]	N/A	1998$ ($5.3M in 2000$)

Conventional wisdom suggests utilities must spend more on wet storage of spent nuclear fuel than dry storage. A rudimentary estimate of the operation and maintenance cost for a spent fuel pool of a shutdown reactor might include dedicated security, general management, two environmental staff, five engineering support, two health physics employees, and a somewhat arbitrary $2.5M a year to cover pool supply and circulation maintenance as well as debris control. Given that a typical power plant today

[*] Low value of $2M per annum is unreasonably low and not considered an accurate reflection of wet storage costs.
[†] Prices quoted in the 2008 INL report are said to represent 10 to 20 year market conditions.
[‡] Inferred from stated $9M for both wet and dry storage and $4M for just dry storage.
[§] Referenced "Office of Civilian Radioactive Waste Management, 1998. Analysis of the Total System Life Cycle Cost of the Civilian Radioactive Waste Management Program. DOE/RW-0510, Department of Energy,December." Unconfirmed.

spends roughly $10M annually on security and absent any justification to significantly modify this figure, utilities should dedicate the same resources to secure large quantities of spent fuel after shutdown. Assuming a budget for today's employees, including overhead such as materials, security upgrades, and employee benefits: $250,000 per manager, $150,000 per environmental monitor, $200,000 per engineer, and $200,000 per health physicist, the annual operation and maintenance requirements might total $14.2M, or $11.3M in 2000$. Table 3.2 outlines a summary of the expected expenditures for spent fuel pool storage.

Table 3.2. Wet Storage Cost Summary.

Variable	Nominal	Low	High
Initial Capital	$0M*	$0M	$0M
Online O&M	$0M/year†	$0M/year	$0M/year
Shutdown O&M	$11.3M/year	$5.3M/year	$12.6M/year

* Assumption that the capital cost of the spent fuel pool is already covered in the initial capital cost of the plant.
† Assumption that the O&M during normal operation of the power plant is included in the electricity rate charged to customers.

3.2 Conditioning and Packaging for Transportation and Dry Storage

Once removed from the cooling pool, the spent nuclear fuel must go through several stages of preparation before loading into multipurpose canisters. Spent fuel operating staff use these metal canisters for structural support and array spacing as opposed to the "overpack" or multipurpose casks used to house the canisters and act as an outer radiation protection barrier. This conditioning module includes the costs associated with lowering the canisters and their surrounding casks into the spent fuel pools, transferring the used fuel assemblies into the flooded containers, capping and backfilling the containers with helium, vacuum sealing and welding the containers, and moving the containers to onsite dry storage pads or immediately preparing them for offsite transportation. This module also includes the cost of the inner canisters and outer multipurpose casks. Figure 3.2 shows the two typical dry storage cask designs and Figure 3.3 illustrates the underwater loading process for the casks. In Figure 3.3, notice the number of people directly involved in the handling and cask loading process.

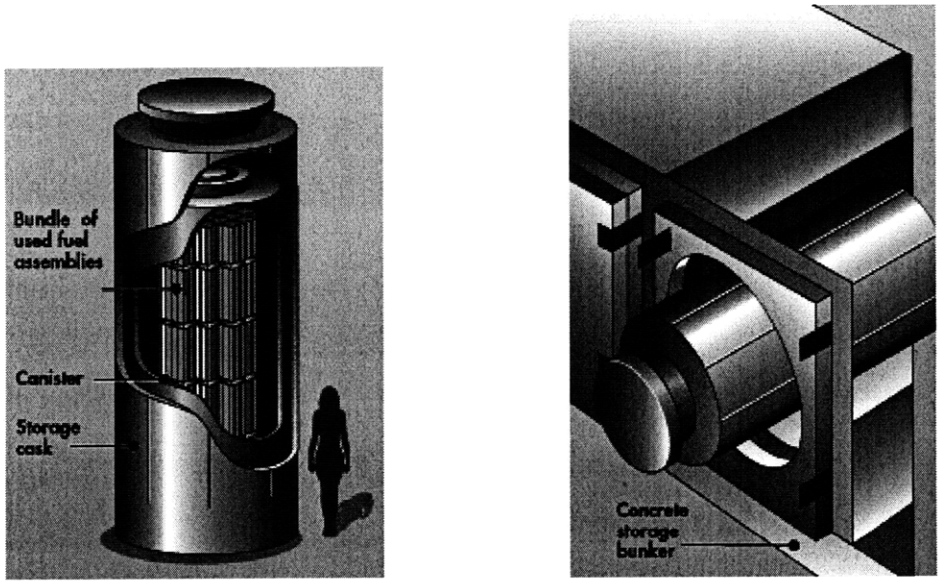


Figure 3.2. Vertical and Horizontal Spent Nuclear Fuel Dry Storage Systems (3.6).

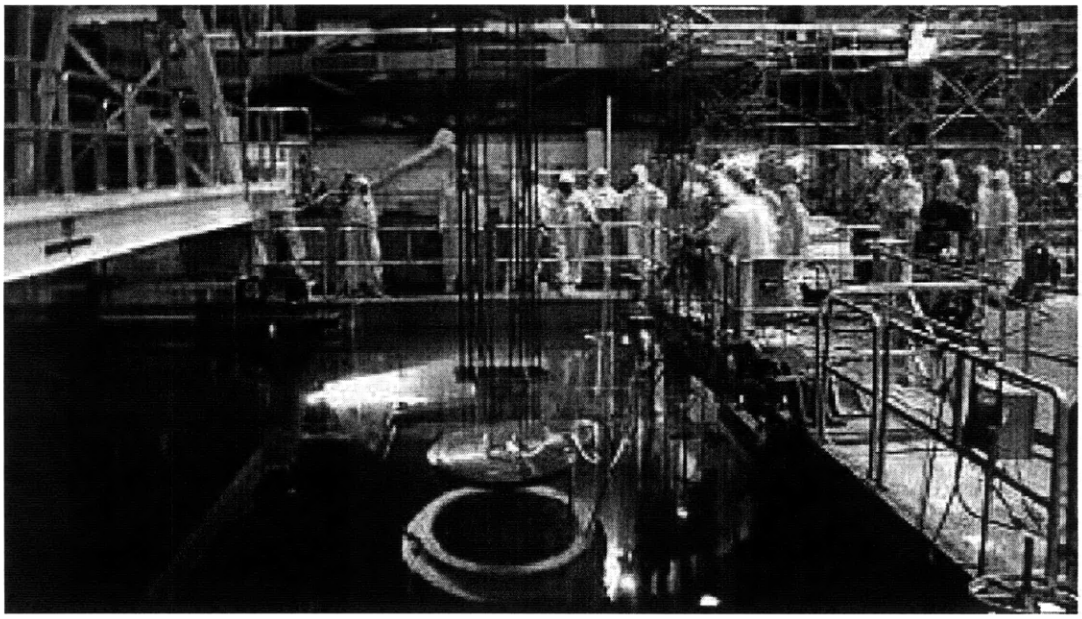


Figure 3.3. Underwater Loading of Dry Storage Cask Before Sealing and Backfilling with Gas (3.4).

As with many of the economic modules, proprietary concerns make accurate pricing challenging. One can fundamentally build the cost of handling in terms of (a) canister and cask costs, (b) heavy equipment costs, (c) personnel, and (d) logistics and planning. The Idaho National Laboratory Fuel Cycle Cost Basis report suggests a likely range of $50 - $130/kg for conditioning and packaging (3.4). This estimate includes all costs for the conditioning module except the outer cask capital expense. The Fairlie report identified cost estimates of several cask systems as ranging from $35 - $68/kg, stating the more expensive prices represent multipurpose casks capable of both dry storage and transportation (3.7)*. After escalation to reflect 2000$, Table 3.3 shows the combination of these estimates and summarizes the spent nuclear fuel initial conditioning costs at the reactor site.

Table 3.3. Transportation and Dry Storage Conditioning and Packaging Cost Summary.

Variable	Nominal†	Low‡	High
Conditioning/Packaging	$150/kg	$91/kg	$209/kg

* 1994$ which equals 79 in 2000$ for the premium multipurpose casks, Referenced "Wisconsin Public Service Commission (1994) Final Environmental Impact Statement: Point Beach Nuclear Power Plant Project. PSC Docket 6630-CE-197, Madison, Wisconsin, US, August 1994." Unconfirmed.

† The nominal price simply reflects the average of the low and high price estimates.

‡ Low price utilizes the low cask estimates because some additional savings might be expected for large contracts.

3.3 Dry Storage in Onsite Dry Storage Casks

The Onsite Dry Storage module considers the costs specific to storing nuclear fuel at reactor sites after discharged reactor fuel fills available spent fuel pool capacity. As mentioned in the wet storage module discussion, designers intended spent fuel pools to act as a cool-down buffer to allow the high radiation levels from freshly discharged fuel to decay. The capacity of spent fuel pools originally ranged from 10 to 20 years to meet this purpose and can never handle the 60 years of core discharges corresponding to the expected operational lifetime of the majority of the U.S. fleet. Utilities increasingly rely on dry storage of their accumulating nuclear waste in order to sustain the operation of the nuclear industry. In 2006, dry storage contracts existed for 67 U.S. reactors and vendors expect this number to increase to 93 by 2012, eventually leading to the addition of 2,000 MT of spent nuclear fuel into reactor dry storage facilities by the year 2015 (3.8). Engineers have designed dry storage casks to function under extreme conditions and utilities can license their use for 20 years under 10 CFR Part 72 to safely maintain a radiation protection barrier, but the Nuclear Regulatory Commission suggests a more likely operable lifetime for dry storage casks on the order of 100 years or more.

The casks themselves come in various designs, but all work in essentially the same way using gas and concrete or metal to replace water as the coolant and radiation barrier. The helium, or sometimes nitrogen, atmosphere inside the canisters minimizes oxidation during storage. Figure 3.4 shows the placement of vertical dry storage casks at onsite storage pads. As a reminder, this module does not include the cost of the cask or the heavy equipment, which the conditioning and packaging module covers because this study assumes the use of multipurpose casks able to act as both a storage cask and transportation overpack.

The dry storage module primarily focuses on the initial capital expense required to begin onsite dry storage as well as the annual operation and maintenance costs associated with monitoring and managing the spent nuclear fuel. A distinction exists between dry storage at a reactor site during normal operation of the power plant and dry storage at a shutdown site in which no reactors continue to operate. In the former, the normal revenue from selling electricity covers major expenses like the cost of security, while a shutdown reactor facility must bill all expenditures to spent fuel management. Table 3.4 summarizes previous literature estimating onsite dry storage costs for both online and shutdown reactor facilities. One must pay close attention to the prices quoted by sources in the open literature because they rarely include details on how the authors arrived at the given values. Note that each of the numbers quoted in Table 3.4 include the purchasing cost of the storage casks.

Figure 3.4. Onsite Storage Pads Supporting Vertical Dry Storage Casks (3.4).

Table 3.4. Documented Cost Estimates of Onsite Dry Storage of Spent Nuclear Fuel.

Cost ($/kgHM)	Source	Capacity (MT HM)	Duration (years)	Notes
280	(3.3)	230	20	Shutdown dry storage estimate for Sacramento Municipal
120	(3.4)	1,200*	N/A	Includes casks, averaged pre- and post-shutdown†
90-210	(3.9)	N/A	N/A	Based solely on discussion with cask manufacturers
120	(3.10)	1,000	40	At reactor during normal operation of plant
250	(3.10)	1,000	40	At reactor after decommissioning of power plant
35-68	(3.7)‡	N/A	N/A	Exclusively the cost of the dry storage cask
68-100	(3.7)§	(3.7)500	300 20	At operating reactor site

*Based on stated 20 MT/year discharge rate and assuming 60 year operational lifetime.
† This is assumed based on vague statements describing the author's arbitrary judgments made.
‡ Referenced "Wisconsin Public Service Commission. Final Environmental Impact Statement: Point Beach Nuclear Power Plant Project. Madison, WI. 1994. PSC Docket 6630-CE-197." Unconfirmed.
§ Referenced "Supko 1995." Unconfirmed.

Again, the values quoted in Table 3.4 include the cost of the dry storage casks themselves, while this module only considers the initial capital expense required to store spent fuel in dry storage casks as well as the annual operation and maintenance cost associated with monitoring the fuel onsite.

Quoted initial capital costs of an onsite dry storage facility in the open literature include the cost of the concrete storage pads, the heavy machinery necessary to load and move the casks, additional security requirements, and the licensing expenses. Table 3.5 lists some estimates for the total up-front costs of dry storage without providing details the utilities and vendors consider proprietary. The initial capital cost depends strongly on the specific conditions at the reactor site, but the values depicted in Table 3.5, likely indicate a realistic range of possible expenses.

Table 3.5. Documented Initial Capital Costs Estimates of Onsite Dry Storage Facilities.

Cost (M$)	Source	Capacity (MT HM)	Notes
8-12	(3.10)*	N/A	1998 study, assume 1998$ ($8.5M - $12.7M in 2000$)
9	(3.11)	N/A	1998$, capacity not specified ($9.5M in 2000$)
12.4	(3.3)	230	1993 study, assume 1993$ ($14.8M in 2000$)
9-18	(3.9)†	N/A	2001 study, assume 2001$ ($8.8M - $17.5M in 2000$)
16.5	(3.7)	500	Inferred from cask-only costs and total system costs, includes O&M‡

Without significant justification to do otherwise, this study assumed $12M, the arithmetic mean§ of the values provided in Table 3.5, accurately represents the initial capital cost for dry storage and the high value of $17.5M and low value of $8.5M should allow for a reasonable measure of pricing sensitivity. Additionally, the conditioning and packaging module includes the high cost of much of the heavy machinery and equipment also accounted for in the quotes above. One presumes the equipment cost represents a significant fraction of the initial capital expense associated with onsite dry storage and for the purposes of the cost simulation, this study assumed the equipment accounted for half the values quoted in Table 3.5.

---

* Referenced "Supko, Eileen M. Minimizing Risks Associated with Post-Shutdown Spent Fuel Storage and LLW Disposal. 1998." Unconfirmed.

† Reference another 2001 Macfarlane study. Unconfirmed.

‡ Determined using the roughly $33/kg difference between cask-only cost and total system cost and the given onsite dry storage capacity of 500 MT.

§ (8.5 + 12.7 + 9.5 + 14.8 + 8.8 + 17.5)/6; notice the final data point of $16.5M is ignored because it includes O&M

The annual operation and maintenance expenses conclude the required input to model the dry storage costs in the waste management economic simulator. These annual expenditures reflect the cost to monitor, secure, and maintain engineering integrity to protect against environmental release. The utility only needs to bill a relatively modest amount to waste management expenses during the normal operation of the power plant due to the availability of so many of the staff required to perform waste management functions. However, when all reactors at a particular site shut down, the staffing requirements persist and significantly increase the waste management operation and maintenance costs for shutdown reactor dry storage. Table 3.6 outlines the limited data published detailing the annual maintenance costs for onsite dry storage.

Table 3.6. Documented Operation and Maintenance Estimates of Onsite Dry Storage Facilities.

Cost (M$/year)	Source	Capacity (MT HM)	Notes
0.47-0.75	(3.11)*	N/A	1998$ ($0.50M - $0.79M in 2000$), online reactor site
4-9	(3.11)†	N/A	1998$ ($4.2M - $9.5M in 2000$), shutdown reactor site
2.6	(3.3)	N/A	1990 study, assume 1990$ ($3.4M in 2000$), shutdown
3-4	(3.10)‡	N/A	1998 study, assume 1998$ ($3.2M - $4.2M in 2000$), shutdown

The absence of available data for annual dry storage operation and maintenance requires some degree of best estimation to ensure the use of reasonable numbers. For the case of dry storage at online reactor sites, this study assumes dry storage monitoring and maintenance requires no more than four additional staff with an average pay of $200,000 per staff in today's dollars including overhead. Converting this cost to 2000$ implies a normal operating site requires no more than $0.64M in additional annual spending to cover the dry storage monitoring, which falls reasonably within the range quoted in Table 3.6. After shutdown, the monitoring might require somewhat fewer resources than those given to onsite spent fuel storage after shutdown because of the complete containment of the nuclear waste. For argument's sake, we assume roughly half the security requirements because of the significantly reduced proliferation risk and consequences of malicious attack. After subtracting the additional water circulation allowance and three engineering staff because of the lack of moving parts in

* Referenced "TRW Systems Analysis, 1993." Unconfirmed.
† Referenced "TRW Environmental Safety Systems Inc., 1998." Unconfirmed.
‡ Referenced "Supko, Eileen M. Minimizing Risks Associated with Post-Shutdown Spent Fuel Storage and LLW Disposal. 1998." Unconfirmed.

the system, we suggest the annual operation and maintenance requirements might total $6.35M, or $5.1M in 2000$, which lies well within the range of the estimates shown in Table 3.6.

Table 3.7 gives a summary of the dry storage cost estimates in 2000$ and provides an expected range of reasonable values for sensitivity analysis. Recall, the initial capital expense reflects half of the values presented in Table 3.5 in an effort to avoid double counting much of the heavy machinery costs already considered in the conditioning and packaging module.

Table 3.7. Dry Storage Cost Summary.

Variable	Nominal	Low	High
Initial Capital	$6.0M	$4.2M	$8.8M
Online O&M	$0.64M/year	$0.50M/year	$0.79M/year
Shutdown O&M	$5.1M/year	$3.4M/year	$9.5M/year

3.4 Rail Transportation

Transportation costs vary widely depending on the nature of the spent fuel requiring moving and the geographic variables in the nation of interest. Many values quoted in the open literature include economic aspects already considered in other modules of this report. With this in mind, Table 3.8 contains a list of published transportation estimates and shows a range between $14 and $101 per kilogram in 2000$. The higher prices include the additional loading costs associated with single-purpose transportation casks as well as the high prices of the casks themselves. This study avoids most of these costs, due to the global assumption of the exclusive use of multipurpose casks capable of dry storage and transportation, thereby avoiding the need to pay high handling charges in many of the economic modules. The U.S. figure presented in the 1994 Nuclear Energy Agency report most closely resembles the actual methodology used in this module and estimates the average price of spent nuclear fuel transportation at under $20 per kilogram of initial heavy metal (3.12).

Table 3.8. Documented Spent Nuclear Fuel Transportation Cost Estimates.

Cost ($/kgHM)	Source	Notes
20-80	(3.12)	1991$ ($25 - $101/kg in 2000$), transportation within the European area
11*	(3.12)	1991$ ($14/kg in 2000$), British estimates
33†	(3.12)	1991$ ($42/kg in 2000$), Swedish estimates
67‡	(3.12)	1991$ ($85/kg in 2000$), German estimates
13	(3.12)	1991$ ($16/kg in 2000$), Canadian estimates, very low burnup fuel§
15	(3.12)	1991$ ($19/kg in 2000$), U.S. Estimate in NEA report**
72	(3.13)	1998$ ($76/kg in 2000$), U.S. Estimate in 1998 TSLCC of Yucca Mountain††
67	(3.13)	2000$, U.S. Estimate in 2000 TSLCC of Yucca Mountain
50-57	(3.11)	1998$ ($53 - $60/kg in 2000$), U.S. Estimates in Macfarlane report‡‡
76-106	(3.4)	2007$ ($63 - $88/kg in 2000$), U.S. Estimates in Idaho National Laboratory report
40-60	(3.14)	Expressed in 2000$

* Numbers presented in British Pounds and converted using 1991 exchange rate of £ 1 = 1.77 USD
† Numbers presented in Swedish Krona and converted using 1991 exchange rate of 1 SKr = 0.166 USD
‡ Numbers presented in German Marks and converted using 1991 exchange rate of 1 DM = 0.605 USD
§ Note that the Canadian fuel discharged from CANDU reactors generally had 1991 burnup levels of 8,330 MWD/MT compared to the upper burnups of 45,500 MWD/MT experienced in the PWR fleet at the same time.
** Used the isolated shipping costs of $1.432B and divided by the given assumption of 96,300 MT waste
†† Used the "National Transportation Costs" for 1983 to 2041 and divided by the stated loading of 83,800 MT
‡‡ Used the transportation costs of $153-740 M + $86 M + $3912 M and divided by 83,800 MT

The transportation module of this study attempts to evaluate the total shipment costs by using the levelized capital costs for reusable components, handling costs, and the basic costs associated with transporting the spent fuel assemblies on the rail lines. Although the conditioning module encases the nuclear waste in multipurpose casks ready to ship, handlers must apply one reusable impact limiter, the large end-caps shown in Figure 3.5, to both ends of the cask before transportation begins. Table 3.9 identifies three different quotes for pairs of impact limiters, which essentially act as an additional safety measure to prevent hull breach in case of an accident in transit. One can easily calculate the per-shipment-cost of reusable components assuming an additional 10% above the cost of the impact limiters for items such as the fittings to secure the cask to the railcar and assuming a useful lifetime long enough to provide the equivalent of 50 roundtrips of an average duration from reactor site to national repository.

Table 3.9. Estimated Costs for a Pair of Transportation Impact Limiters.

Cost (M$)	Source	Associated Cask	Notes
1.64	(3.4)*	HI-STAR	2007$ ($1.40M in 2000$), earlier quote had pair at $2.84M
0.60	(3.4)	NAC-UMS	2007$ ($0.50M in 2000$)
0.58	(3.4)	NAC-STC	2007$ ($0.48 M in 2000$)

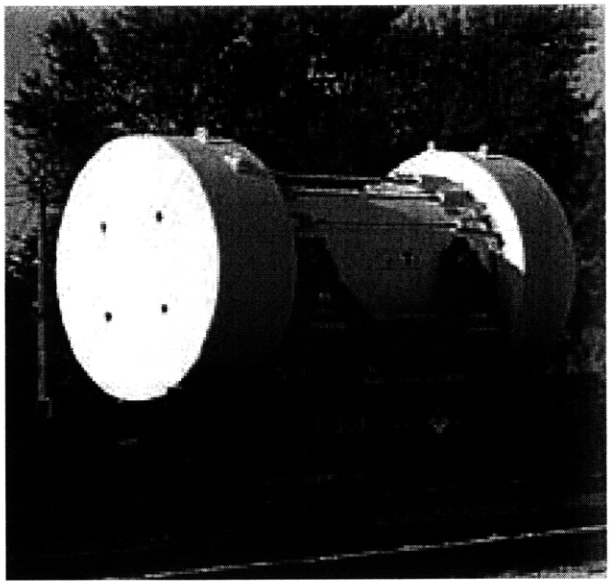


Figure 3.5. Spent Nuclear Fuel Transportation Cask on Rail Car with Impact Limiters (3.15).

* Series of informal quotes collected by Sandia National Laboratory in 2001 and 2003.

Most analysts consider the handling costs per cask a relatively minor component, only $15,000 to $16,300[*], of the overall transportation expense because the already packaged and conditioned casks require little work to simply load onto and unload from the railcar (3.4; 3.16). The more obvious shipment charges cover the moving costs, roughly $0.0106 to $0.0139 per kilometer per kilogram of initial heavy metal in the assembly, and various fees associated with shipping sensitive material across the country, estimated at just $0.0012[†] per kilometer per kilogram of heavy metal. This shipping tariff ignores the additional state fees possibly totaling as much as the regular tariff itself, which Table 3.10 includes in the summary of shipping charges.

Table 3.10. Shipping Cost Summary.

Variable	Nominal	Low	High
Equipment[‡]	$2.92/kgHM[§]	$1.00/kgHM	$2.92/kgHM
Handling	$1.48/kgHM[**]	$1.42/kgHM	$1.54/kgHM
Moving	$0.0037/km/kgHM	$0.0034/km/kgHM	$0.0039/km/kgHM
Fees	$0.0018/km/kgHM	$0.0012/km/kgHM	$0.0024/km/kgHM

The waste management simulator attempts to simplify the transportation analysis by using cost per kilogram of initial heavy metal economic numbers in the program input and making assumptions about the distances traveled. The referenced transportation documents spend a significant time analyzing details about the transportation process, including identifying prospective interim storage and repository locations, shown in Figure 3.6 (3.4; 3.16). We accept the ins and outs of selecting a repository and interim storage site discussed in other papers and recognize the political arguments against these sites and the insufficient technical review to truly recommend those locations. However, going through the minutiae of the siting process does not add to the economic or policy analysis considered in this study. With these disclosures in mind, Table 3.11 outlines the regional transportation costs calculated using the average distances between origin and destination for each shipping route, listed in Table 3.12. Table 3.11 prominently displays the nominal transportation costs and shows both the low and high estimates in parentheses.

---

[*] These values were originally reported in different year dollars and are translated here as 2000$

[†] Shipping tariff of $0.1056/tonne-km of cargo, must multiply by 11 to get the tariff per tonne-km of initial HM.

[‡] Includes 10% equipment costs above the impact limiters

[§] The nominal equipment cost is assumed to be the high cost for consistency with the two referenced studies

[**] The nominal handling, moving, and fee costs are simply taken as the average of the high and low values

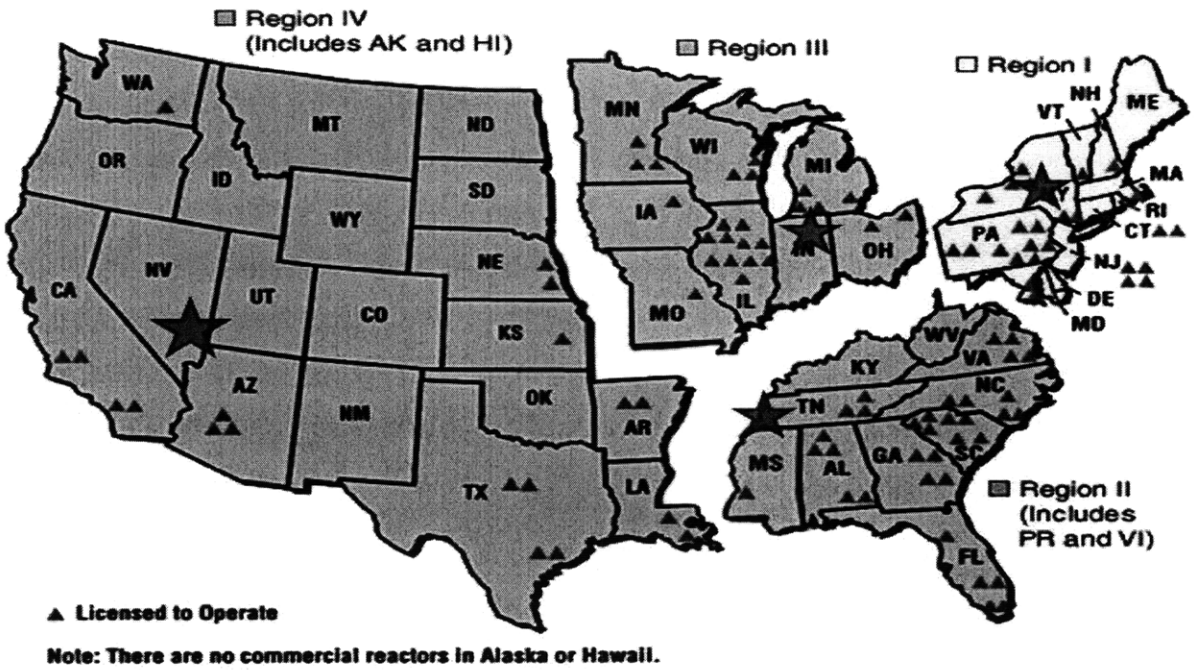


Figure 3.6. Nuclear Regulatory Commission Region Map with Storage Facility Locations* (3.16).

Table 3.11. Regional Transportation Cost Summary Per Kilogram of Initial Heavy Metal.

From\To	I	II	III	IV
I	\$7/kg $\binom{4}{7}$	--	--	\$26/kg $\binom{21}{30}$
II	--	\$10/kg $\binom{7}{10}$	--	\$24/kg $\binom{19}{27}$
III	--	--	\$7/kg $\binom{5}{8}$	\$20/kg $\binom{15}{22}$
IV	--	--	--	\$14/kg $\binom{11}{16}$

* The proposed regional facility in Nevada (Region IV) is assumed to be the site of a national repository should only one be pursued.

Table 3.12. Site-Specific Transportation Distances To Both Regional and National Storage Sites*.

Site	NRC Region	Location	Kilometers to Regional Facility	Kilometers to National Facility
Arkansas Nuclear One	II	Russellville, AR	354	2,363
Beaver Valley	I	Shippingport, PA	686	3,434
Big Rock Point	III	Charlevoix, MI	745	3,317
Braidwood	III	Will County, IL	352	2,721
Browns Ferry	II	Huntsville, AL	301	3,011
Brunswick	II	Southport, NC	1,270	4,078
Byron	III	Ogle County, IL	455	2,623
Callaway	III	Callaway County, MO	571	2,371
Calvert Cliffs	I	Lusby, MD	700	3,917
Catawba	II	York, SC	1,017	3,669
Clinton	II	Clinton, IL	282	2,763
Columbia	IV	Richland, WA	1,381	1,381
Comanche Peak	IV	Somervell County, TX	2,232	2,232
Cook	III	Bridgman, MI	333	2,858
Cooper Station	IV	Nemaha County, NE	2,045	2,045
Crystal River 3	II	Crystal River, FL	1,228	4,015
Davis-Besse	III	Oak Harbor, OH	438	3,156
Diablo Canyon	IV	San Luis Obisbo, CA	906	906
Dresden	III	Morris, IL	352	2,697
Duane Arnold	III	Palo, IA	674	2,419
Enrico Fermi 2	III	Monroe County, MI	447	3,164
Farley	II	Dothan, AL	719	3,428
Fitzpatrick	I	Oswego, NY	163	3,822
Fort Calhoun	IV	Fort Calhoun, NE	2,026	2,026
Ginna	I	Rochester, NY	245	3,727
Grand Gulf	II	Port Gibson, MS	447	2,895
Haddam Neck	I	Haddam Neck, CT	338	4,160
Harris	II	New Hill, NC	1,225	3,846
Hatch	II	Baxley, GA	933	3,724
Humboldt Bay	IV	Eureka, CA	1,502	1,502
Indian Point	I	Buchanan, NY	304	4,027
Kewaunee	III	Carlton, WI	608	2,971
Lacrosse	III	Genoa, WI	782	2,678

* Strongly based on the Gibbs report, simply using travel planning software to determine driving distances between destinations and assuming the rail distances would be comparable.

LaSalle County	III	Ottawa, IL	381	2,665
Limerick	I	Limerick Township, PA	486	3,882
Maine Yankee	I	Wiscasset, ME	621	4,535
McGuire	II	Charlotte, NC	987	3,642
Millstone	I	Waterford, CT	378	4,184
Monticello	III	Montecello, MN	1,027	2,668
North Anna	II	Louisa County, VA	1,291	3,787
Oconee	II	Seneca, SC	803	3,571
Oyster Creek	I	Ocean County, NJ	476	4,085
Palisades	III	South Haven, MI	380	2,906
Palo Verde	IV	Wintersburg, AZ	661	661
Peach Bottom	I	Peach Bottom Township, PA	624	3,830
Perry 1	III	North Perry, OH	562	3,343
Pilgrim 1	I	Plymouth, MA	447	4,361
Point Beach	III	Carlton, WI	608	2,971
Prairie Island	III	Red Wing, MN	914	2,614
Quad Cities	III	Cordova, IL	528	2,512
Rancho Seco	IV	Clay Station, CA	1,073	1,073
River Bend	IV	St. Francisville, LA	2,992	2,992
Robinson	II	Hartsville, SC	1,061	3,816
Salem/Hope Creek	I	Lower Alloways Township, NJ	565	3,988
San Onofre	IV	San Onofre Park, CA	628	628
Seabrook	I	Seabrook, NH	451	4,365
Sequoyah	II	Soddy-Daisy, TN	594	3,249
South Texas	IV	Bay City, TX	2,528	2,528
St. Lucie	II	Fort Pierce, FL	1,505	4,295
Summer Unit 1	II	Jenkinsville, SC	1,027	3,681
Surry	II	Surry County, VA	1,440	3,938
Susquehanna	I	Luzerne County, PA	391	3,785
Three Mile Island	I	Harrisburg, PA	534	3,772
Trojan	IV	Rainier, OR	1,648	1,648
Turkey Point	II	Homestead, FL	1,730	4,519
Vermont Yankee	I	Brattleboro, VT	340	4,241
Vogtle	II	Burke County, GA	890	3,679
Waterford	IV	Hahnville, LA	3,190	3,190
Watts Bar	II	Spring City, TN	589	3,244
Wolf Creek	IV	Burlington, KS	2,095	2,095
Yankee Rowe	I	Rowe, MA	219	4,134
Zion	III	Zion, IL	377	2,800

3.5 Centralized Interim Storage

Many propose interim storage as a short-term fix to reduce the accumulating nuclear waste at reactor sites. Some estimate that the government will soon owe an additional $500M for each year it fails to collect the spent fuel (3.17). When presented with extreme costs like this, a cheap facility capable of operating for 40 to 100 years seems like a fiscally responsible alternative, even if vast political challenges stand in its way. The Private Fuel Storage facility in Utah received Nuclear Regulatory Commission approval, but state opposition led to changes in land zoning, preventing possible operation of the interim storage complex if built. In the absence of a functioning facility on which to gauge expenses, the published estimates shown in Table 3.13 hold little value, particularly given the lack of explicit details in their determination. One must assume that the higher quotes include the cost of repackaging the fuel and purchasing single-purpose dry storage casks.

Table 3.13. Documented Interim Storage Cost Estimates.

Cost ($/kgHM)	Source	Notes
50-70	(3.14)	2 years at an unknown facility type and size
94-116	(3.4)	40 years at a 40,000 MT capacity facility, 2007$
121	(3.12)	15,000 MT capacity facility, 1991$
176*	(3.5)	50 years at a 40,000 MT facility, 1998$
194	(3.12)	50 years at a 8,000 MT Swedish facility, 1991$
230†	(3.4)‡	54 years at a 5,000 MT Japanese facility, 1998$
251	(3.12)	40 years at an unknown facility type and size
300	(3.18)	50 years at an unknown facility type and size
350	(3.18)	100 years at an unknown facility type and size

This module attempts to recreate the cost of a large interim storage facility relying heavily on the Private Fuel Storage 1997 license application to the Nuclear Regulatory Commission (3.19). To allow for flexibility in policy analysis, this report provides an estimate for the levelized interim storage costs associated with a reference design facility similar to the proposed 40,000 MT Private Fuel Storage site as well as a smaller regional facility with a capacity on the order of 10,000 MT and a very large 80,000 MT complex.

---

* Assumed only capital cost and annual operation and maintenance for 40,000 MT and 50 years

† Numbers presented in Japanese Yen and converted using 1998 exchange rate of 1 ¥ = 0.00768 USD

‡ Referenced "MITI, Toward Implementation of Interim Storage for Recycled Fuel Resources, op. cit." Unconfirmed.

Figure 3.7 gives an overview of the reference facility layout in Skull Valley, Utah where a 720 acre exclusion and buffer region surrounds a 100 acre restricted access zone (3.20). The reference design calls for 500 modular concrete storage pads each supporting 8 dry storage casks of roughly 10 MTIHM in spent nuclear fuel. The license application outlines the capital expenses as shown in Table 3.14.

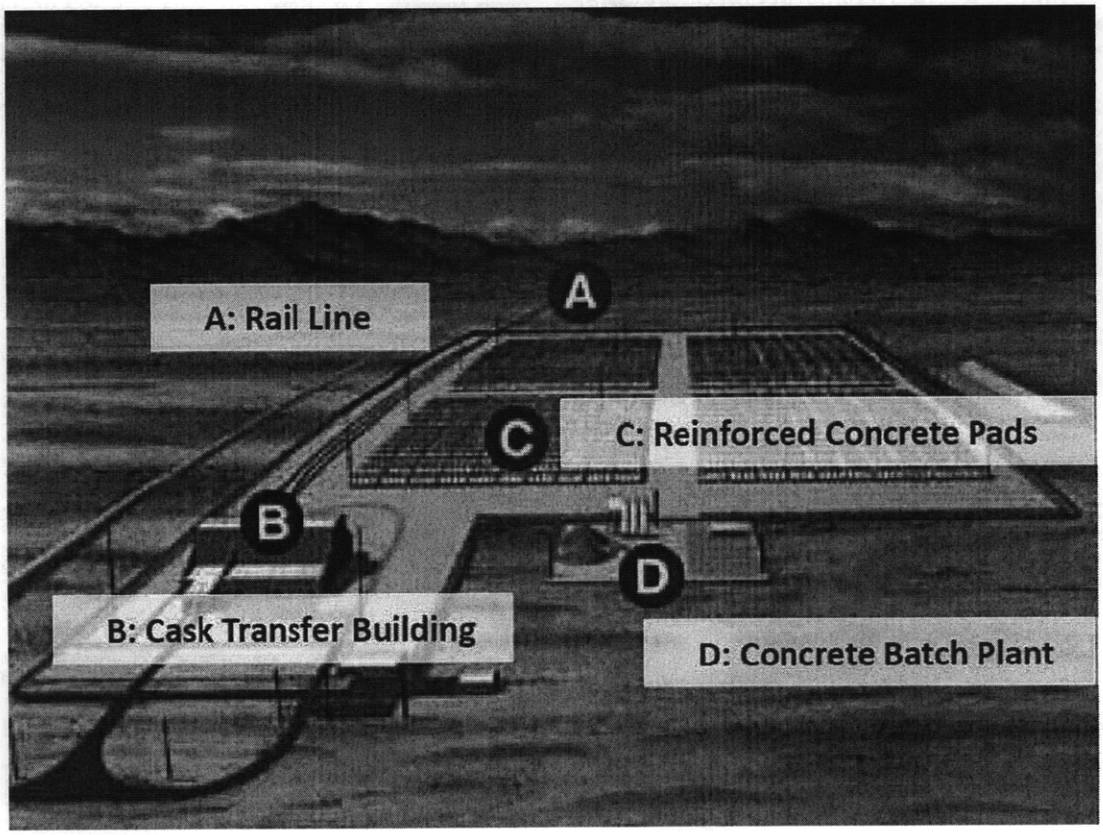


Figure 3.7. Skull Valley, Utah Reference 40,000 MT Interim Storage Facility (3.21).

Table 3.14. Private Fuel Storage Facility Phase Expenditures (3.19).

Cost* (M$)	Phase Description
10	NRC licensing proceeding, design and preparation of bid specifications
100	Facility construction
68†	Cask decommissioning
1.6	Site decommissioning

* The report was written in 1997, but suggested many of the costs were expressed as estimated 2000$
† Quoted as $17,000 per cask

The overnight cost of the site, heavy machinery, and licenses totals $110M, which we assume gets paid in equal annual payments during the construction of the project. This module ignores the cask decommissioning expense, which the repository conditioning module will address, and ignores the site decommissioning, which is negligible and occurs far in the future. Firstly, the site decommissioning should only represent a small fraction of the total site cost because no cask ever gets opened and no radioactive material should escape to the environment. Secondly, even a very large expense that occurs a hundred years in the future has little impact on the economic evaluation of a project. Note that at a discount rate of just 5%, one would need to set aside less than 1% of the expected future expenditure to cover the cost of decommissioning. The level of uncertainty in the estimations overshadows this approximation.

The general construction includes the cost of the site preparation, construction of access routes, and development of several buildings and procurement of necessary heavy equipment. The major buildings include the massive, reinforced-concrete, Canister Transfer Building approximately 60 meters wide, 80 meters long, and 27 meters tall. In general, the initial capital costs presumably do not scale with the site capacity. Therefore, a major assumption is that the different sized facilities generally have the same maximum loading rate and that four regional interim storage sites receive the nation's waste four times faster than a single national storage facility, but at roughly four times the initial capital cost. In practice, any interim storage strategy owned by a single organization might find cost savings in their second, third, and fourth facilities, which the low initial capital estimates in the interim storage summary may reasonably reflect.

The Skull Valley concept houses the Security and Health Physics Building at the entrance to the restricted area in a 23 meter wide, 37 meter long, and 5.5 meter tall structure with sufficient laboratory space to meet radiation protection needs as well as enough security and surveillance equipment for defense requirements. The Administration Building and Operations and Maintenance Building sit outside the restricted area. The general offices, meeting area, and emergency response center reside in the 24 meter wide, 46 meter long, and 5 meter tall Administration Building, while the 24 meter wide, 61 meter long, and 8 meter high Operations and Maintenance Building functions more like a large warehouse and repair shop storing spare parts and equipment for the entire complex.

The annual operating budget should consist of three components: (a) base rate to cover costs of site utilities as well as management, health physics, and security, (b) handling rate that considers the additional staff required during the initial loading and final unloading of the facility, and (c) monitoring

rate that reflects the number of engineers and environmental scientists required as a function of the amount of spent fuel stored on site.

One must include significant overhead costs when budgeting annual expenditures on salaried personnel. The amount of overhead depends heavily on the employee function and can account for as much as one-half of the budgeted salary to cover insurance, taxes, tools, utilities, facility maintenance, and so on. As an approximation and without strong justification, Table 3.15 displays the assumed annual employee budgets, including overhead, for the primary job functions mentioned previously.

Table 3.15. Annual Budget Requirements in 2008 Dollars by Employee Function.

Employee Function	Budget Per Employee (x$1,000)
Manager	250
Health Physics	200
Security	200
Handler	150
Engineer	200
Environment	150

Presumably, a single manager should adequately supervise the normal operation of the interim storage facility, three health physics personnel should sufficiently monitor worker dose, and the reactor site standard of $10M in annual security costs, corresponding to 50 guards, should protect the site. While 50 guards may seem like excessive force to safeguard sealed spent fuel assemblies, most reactor sites split their security into several teams. This allows for regular rotation of the tasks and provides sufficient time off from work and abundant training time. This estimated base cost for operation and maintenance totals $10.85M (2008$) annually.

The periods of loading and unloading should require one more manager and 30 handlers yielding an additional $4.75M (2008$) annually. We reach the conclusion that 30 handlers must run the daily operations because we assume 15 handlers could conservatively unload a single cask from a railcar and transfer it to the concrete storage pads in a day. Then the required work schedule defines the number of handlers given a 10 year loading period for the 40,000 MT reference facility, proposed by the Idaho National Laboratory report, and several other assumptions as follows:

Handler Capacity = 15 handlers/cask/day

Maximum Facility Storage = 40,000 MTIHM

Cask Capacity = 10 MT/cask

Loading and Unloading Duration = 10 years

Capacity Factor = 80%

Conversion = 50 workweeks/year * 5 workdays/workweek = 250 workdays/year

$$Handlers = 15\left[\frac{handlers}{cask/day}\right] \times \frac{40{,}000[MT] \Big/ 10\left[\frac{MT}{cask}\right]}{10[yr] \times 0.80 \times 250\left[\frac{workdays}{yr}\right]}$$

$$Handlers = 30[handlers]$$

Finally, the amount of spent fuel on site determines the budget for the monitoring rate. If we assume every 200 dry storage casks require a single engineer for maintenance and two environmental scientists, then after the first year of loading, each proposed interim storage facility houses up to 400 casks, needing 2 engineers and 4 environmental scientists, or $1.3M (2008$) additional annual spending. Because the monitoring rate really reflects the amount of fuel in storage, we choose a convention to express the monitoring rate in terms of dollars per kilogram per year, as shown in the summary of interim storage expenses listed in Table 3.16. Notice that using this convention makes the values expressed in Table 3.16 general for any sized interim storage facility.

Table 3.16. Interim Storage Cost Summary.

Variable	Nominal	Low*	High
Initial Capital	$110M	$99M	$121M
Base O&M	$8.67M/year	$7.80M/year	$9.54M/year
Loading O&M	$3.79M/year	$3.41M/year	$4.17M/year
Monitoring O&M	$0.260/kgHM/year	$0.234/kgHM/year	$0.286/kgHM/year

* For lack of a better choice, the high and low estimate simply represents a 10% deviation from the nominal.

To get an estimate of these expenses expressed as dollars per kilogram of initial heavy metal in the spent nuclear fuel, Table 3.17 illustrates the significant economies of scale determined from simple calculations using the expenditures for the various sized repositories with a storage lifetime of 50 years. To account for the loading and unloading time, the 10,000 MT facility must remain open 52.5 years, the 20,000 MT facility must remain open 55 years, the 40,000 MT facility must remain open 60 years, and the 80,000 MT facility must remain open 70 years. In each case, the individual spent fuel assemblies remain stored on concrete pads for 50 years.

Table 3.17. Fifty Year Interim Storage Costs in $/kgHM for Various Capacities*.

Capacity	Nominal	Low	High
10,000 MTIHM	$77/kg	$70/kg	$85/kg
20,000 MTIHM	$47/kg	$42/kg	$51/kg
40,000 MTIHM	$31/kg	$28/kg	$35/kg
80,000 MTIHM	$24/kg	$21/kg	$26/kg

* These prices represent undiscounted costs.

3.6 Conditioning and Repackaging for Repository Disposal

The repository conditioning module acts in the same way as the "transportation conditioning" module or the "loading" component of the interim storage module because it simply accounts for the annual expenditures associated with receiving and handling the transportation casks, repackaging the fuel into a disposal cask, and transferring the spent fuel into the repository. The repository module covers the capital cost associated with the very expensive conditioning and repackaging facilities because presumably only one facility will coexist with a repository location. At this point one must distinguish between the costs of conditioning for a Yucca Mountain type repository versus a deep borehole concept because the disposal casks must serve very different functions. Figure 3.8 illustrates the stages of the conditioning and repackaging process for a borehole disposal system, but the general steps appear similar for disposal in a Yucca Mountain repository.

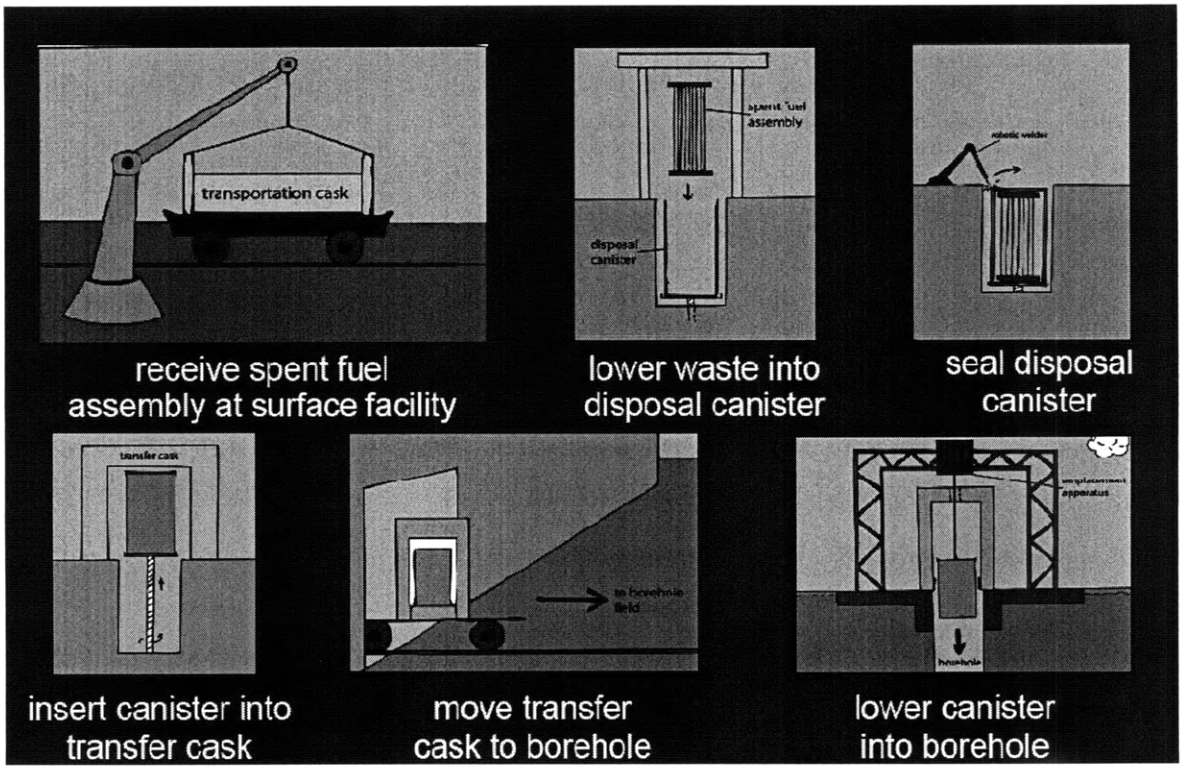


Figure 3.8. Conditioning and Repackaging Stages for Deep Borehole Disposal (3.22).

The first components of the repackaging expenses consist of cask costs. This module not only considers the cost of single-purpose disposal casks, but also conservatively assume the cost of decommissioning for each of the old multipurpose casks used to transport the spent fuel to the repository site. The simplest assumption for the cask decommissioning cost results from taking the cask decommissioning

budget of the Private Fuel Storage interim storage facility and assuming the same expense applies to these multipurpose casks. Recall, the decommissioning budget totaled $68M for casks that held 40,000 MTIHM and if we assume the same cost for disposing of the inner canisters, the most straightforward decommissioning estimate totals a modest $3.40 per kilogram of initial heavy metal. The assumptions involved in reusing the casks after interim storage and delivery of spent fuel to the repository are mentioned in Section 4 and detailed in the reference documents. Still, a thorough reevaluation of cask reusability is recommended for future investigations.

Several studies completed thorough design evaluations and economic analyses for borehole waste canisters, which this report will borrow and integrate into the repackaging module for disposal (3.23; 3.24). One borehole strategy considered the use of small canisters, with dimensions listed in Table 3.18, each capable of only holding one PWR assembly. Calculations of a single-assembly canister cost totaled $3,630 or $7.26 per kilogram of initial heavy metal versus the estimates of a larger, four-assembly canister totaling $12,000 or $6.00 per kilogram (3.24)*. Considering the significant expenses associated with conditioning and packaging of spent nuclear fuel, only the drilling costs could make the economic argument for going to larger capacity disposal canisters.

Table 3.18. Borehole Waste Canister Parameters (3.25).

Parameter	Single-Assembly Canister
Steel Grade	C95
Outer Diameter	339.7 mm
Inner Diameter	315.3 mm
Wall Thickness	12.19 mm
Length	5.0 m
Collapse Pressure	161 bar
Body Yield Strength	8,216 kN
Tube Mass	101.20 kg/m
Tube Cost†	$2.57/kg
Packing Material	Silicon Carbide
Packing Mass	118.00 kg/m
Packing Cost	$3.95/kg

* We assume costs in Guerin report are expressed in 2008$, which must be adjusted to 2000$ in the cost summary.
† Estimated based on assumptions presented in Guerin report.

The Forrest and Rogers report extensively detailed the operations of surface facilities and estimated the total borehole emplacement costs for a 3,000 MT per year acceptance rate as $4.61B. Table 3.19 summarizes these conditioning, repackaging, and borehole disposal costs, which we should compare to the estimates in the transportation and dry storage conditioning and packaging module. There we found a range of prices from $91 to $209 per kilogram, which one might expect because that price range reflects extremely poor economies of scale, very high overhead costs for utilities to contract vendors to visit sites, and large capital equipment purchasing and leasing. In the case of borehole conditioning and repackaging, the centralized facility reduces expenses and overhead and the ultimate repository disposal module covers the heavy machinery costs.

Table 3.19. Borehole Conditioning and Repackaging Cost Summary.

Variable	Nominal	Low*	High
Decommissioning	$3.40/kgHM	$3.06/kgHM	$3.74/kgHM
Borehole Canister	$7.26/kgHM	$6.53/kgHM	$7.99/kgHM
Handling	$55/kgHM	$49/kgHM	$61/kgHM
Total	$66/kgHM	$59/kgHM	$73/kgHM

In the case of a Yucca Mountain type of repository, the 2001 Office of Civilian Radioactive Waste Management report, which assumed an 83,800 MT total disposal requirement, already provided the cost estimates in 2000$. The spent fuel disposal canisters, shown in Figure 3.9, must meet far greater standards than the borehole canister design because in Yucca Mountain, the canisters provide a required engineering barrier to radioactivity release. These engineered containment systems have an inner canister for structural integrity made of stainless steel, which is sealed inside an outer canister made of a highly corrosion resistant nickel alloy, all of which gets protected from water by a drip shield. The report estimates $13.29B as the total cost associated with the waste packages and protective drip shields, $4.69B for the operation of surface facilities, and $4.94B for the operation of subsurface facilities. Table 3.20 outlines the summary of the Yucca Mountain repackaging costs for a roughly 3,000 MT per year waste acceptance rate, which drastically exceed the borehole packaging costs due to the high engineering performance requirements of the Yucca Mountain canisters and the additional handling expenses required to ensure long-term retrievability from the mined geologic repository.

* For lack of a better choice, the high and low estimate simply represents a 10% deviation from the nominal.

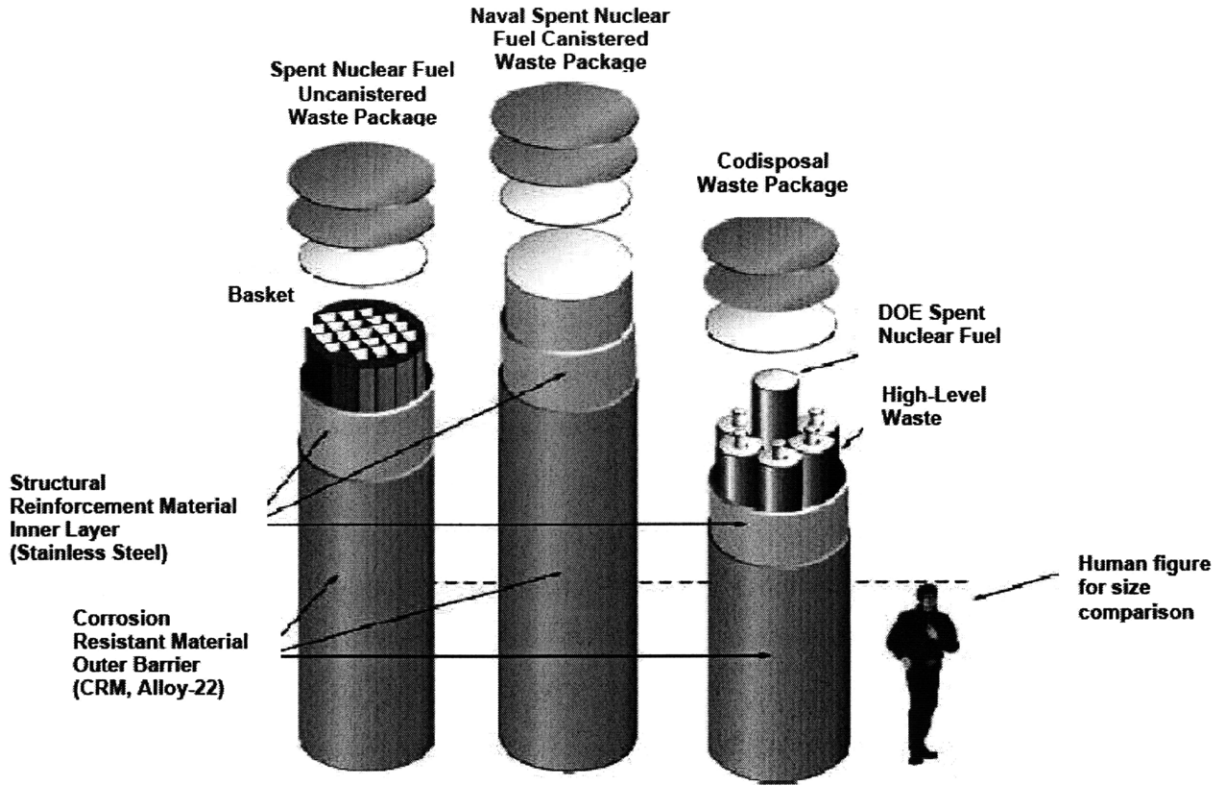


Figure 3.9. Disposal Canisters for Yucca Mountain Project (13).

Table 3.20. Yucca Mountain Type Repository Conditioning and Repackaging Cost Summary.

Variable	Nominal	Low*	High
Repository Canister	$159/kgHM	$143/kgHM	$175/kgHM
Surface Handling	$56/kgHM	$50/kgHM	$62/kgHM
Subsurface Handling	$59/kgHM	$53/kgHM	$65/kgHM
Total	$274/kgHM	$246/kgHM	$302/kgHM

* For lack of a better choice, the high and low estimate simply represents a 10% deviation from the nominal.

3.7 Ultimate Repository Disposal

The repository disposal module includes any required capital expenses associated with the repository handling facilities, the repository site construction or borehole drilling, and the monitoring costs. Clearly, the borehole disposal concept will greatly differ in cost from the Yucca Mountain repository type, because of the differing technology and modularity of the borehole design.

The full Yucca Mountain Project site includes several unique features, shown in Figure 3.10, including an advanced ventilation system and engineered barriers (3.26). Figure 3.11 shows the layout of the surface facilities, which primarily consists of a waste handling building and its associated heavy machinery and emplacement equipment. The strategy of primarily handling dry waste with optional wet processing for off-normal spent fuel provides the greatest flexibility and cheaply incorporates lessons learned in the La Hague experiences while the robust design of these buildings allows the structure to maintain integrity during most conceivable natural phenomena (3.27). There exist many subtleties to the operation of the surface facilities that play important roles in the stocks and flows of waste on a micro level. However, these distinctions do not make any significant difference in the overall waste management system dynamics because we concern ourselves with behaviors that have time steps on the order of a year, therefore mitigating the distinction between a waste handling operation with six days of capacity stored onsite versus six months.

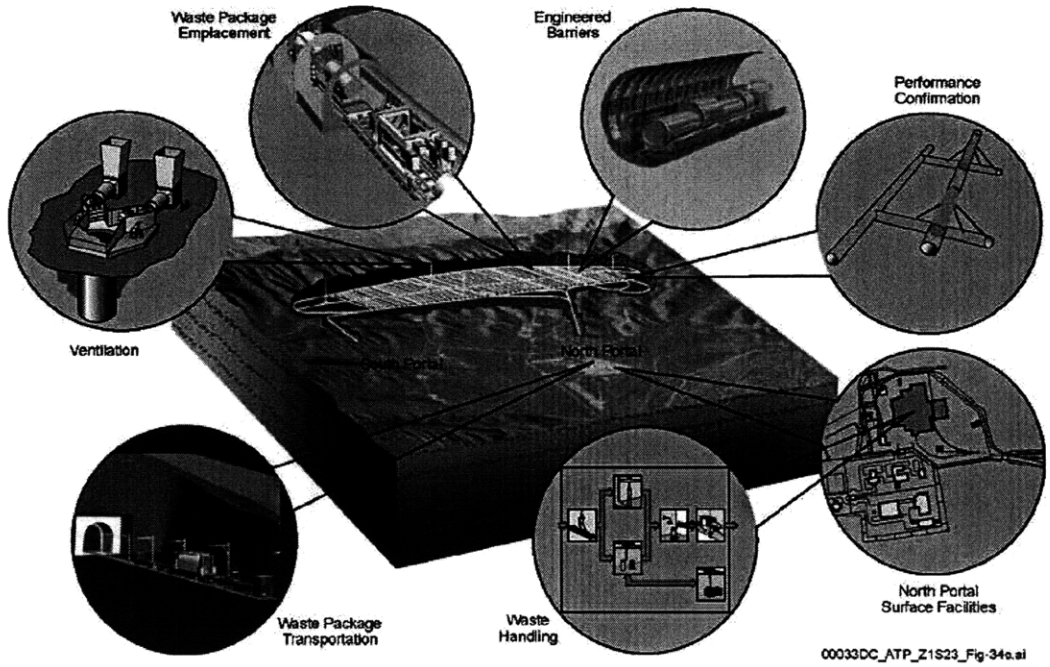


Figure 3.10. Yucca Mountain Project Proposed Monitored Geologic Repository Facilities (3.28).

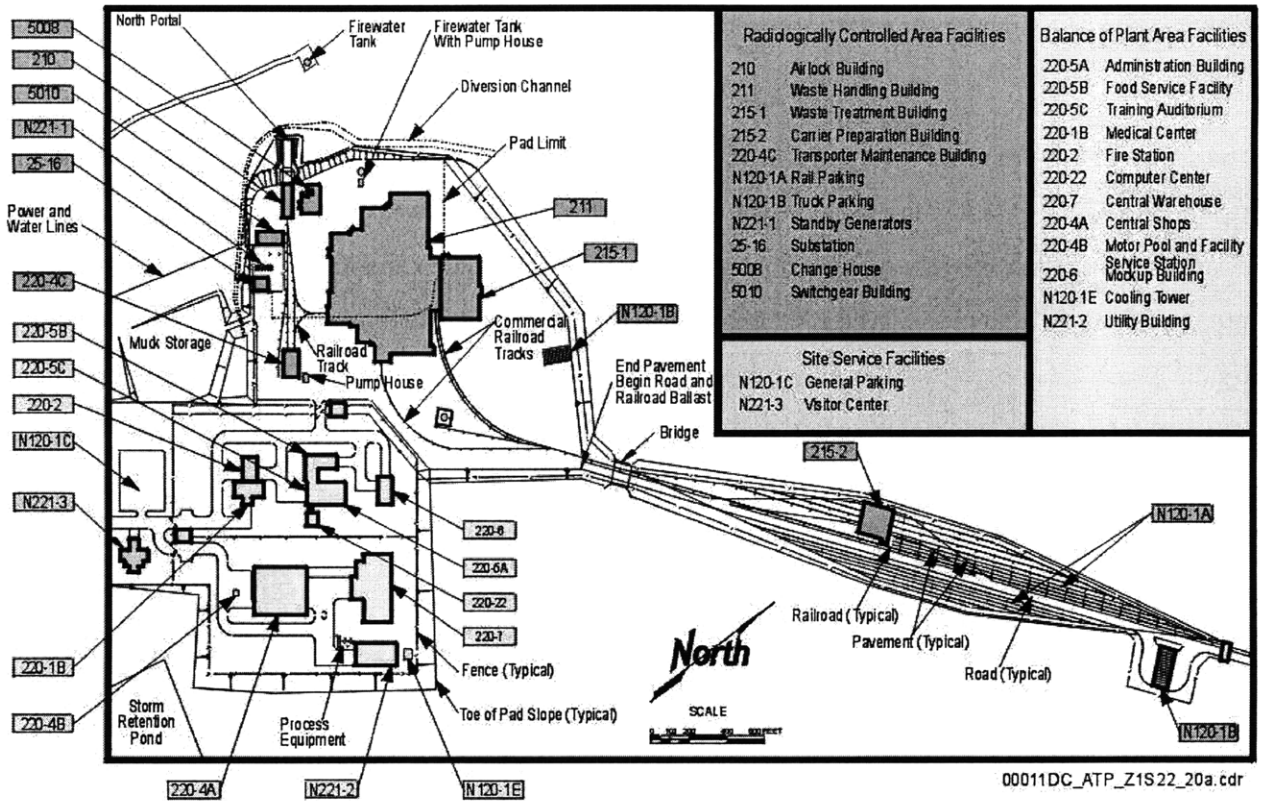


Figure 3.11. Yucca Mountain Project Surface Facilities Layout (3.26).

The Forrest and Rogers report itemizes these components for a proposed borehole surface facility totaling $1.20B and representing a cost of roughly $580M less than the Yucca Mountain Project equivalent facilities (3.22; 3.13). The deeply discounted construction and equipment expenses for the borehole case results from the reduction or elimination of several components associated with receiving multiple waste forms and those items supporting more complex subsurface operation of the Yucca Mountain design. This thesis rejects the premise that a large borehole disposal facility would not need to also handle defense related and high level waste forms in addition to the spent nuclear fuel requirements and conservatively assumes similar complexities of the required support systems on the surface, so this report proposes identical nominal expenditures for the surface facilities of both disposal concepts. The lower estimate for the borehole surface facility reflects the reduced cost proposed in the Forrest and Rogers report, while the higher estimate reflects a simple 10% additional margin. In both cases, we assume the regulatory and licensing costs for the surface facilities total $310M, a number consistent with the Yucca Mountain estimates. While there exists little capacity dependence on the repository licensing costs, and therefore one might expect to incur four separate licensing expenses on the order of $310M should the Department of Energy pursue a four repository strategy, these

expenditures likely diminish and a learning curve might reduce the licensing costs of the second, third, and fourth repositories by as much as 10% each.

The vast expense discrepancy reflects the fact that the subsurface facilities of the two different disposal technologies differ greatly. In the case of Yucca Mountain, much effort to characterize and estimate system costs still leaves the country with great uncertainty about the expenditure because of the project's first of a kind nature. The 2001 subsurface project design estimate totals $4.04B and includes multiple access ramps, a complex network of ventilation shafts, and a highly engineered set of drifts for waste emplacement and monitoring operations (3.29; 3.13). Figures 3.12 and 3.13 show the layout and vertical placement of the waste containers and the ventilation system. Figure 3.14 illustrates the engineering design of a single emplacement drift with several waste package types.

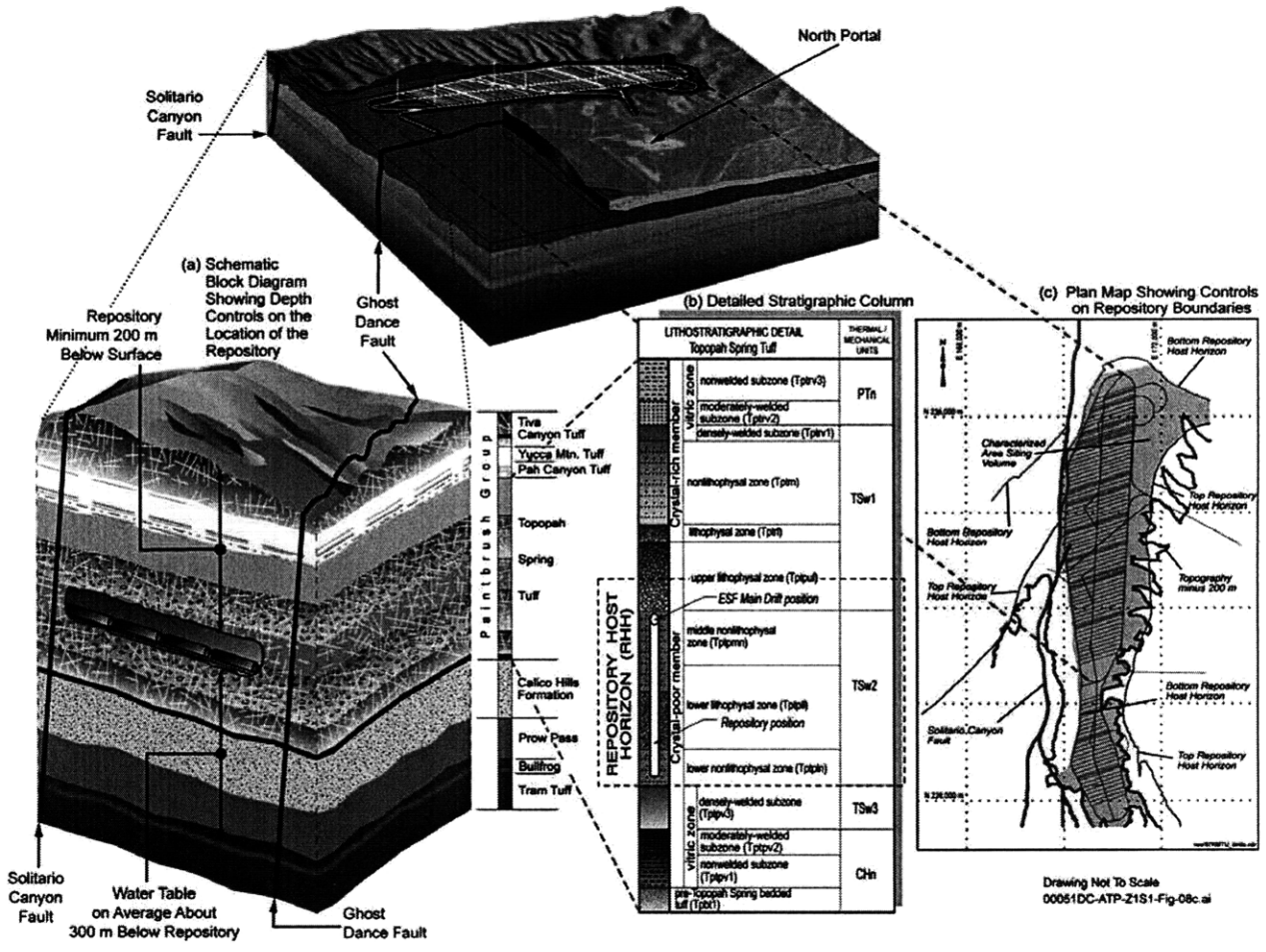


Figure 3.12. Yucca Mountain Project Subsurface Facilities (3.26).

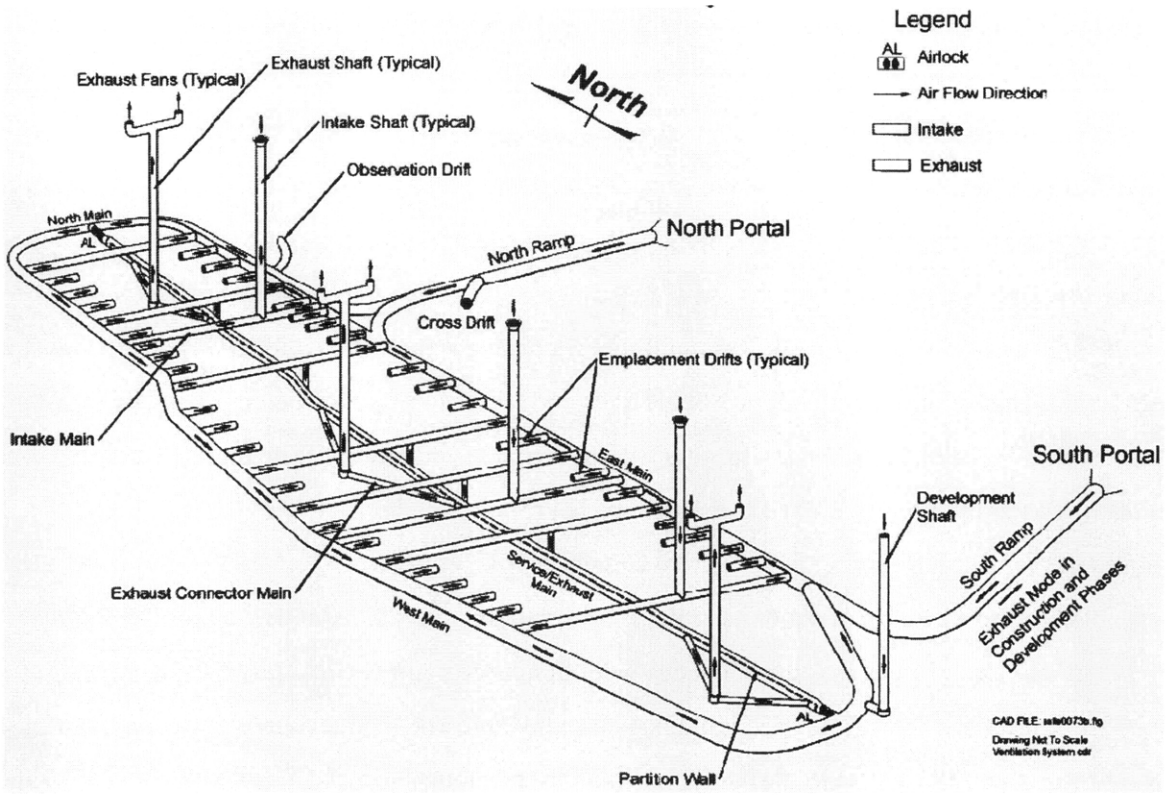


Figure 3.13. Yucca Mountain Project Subsurface Ventilation System (3.29).

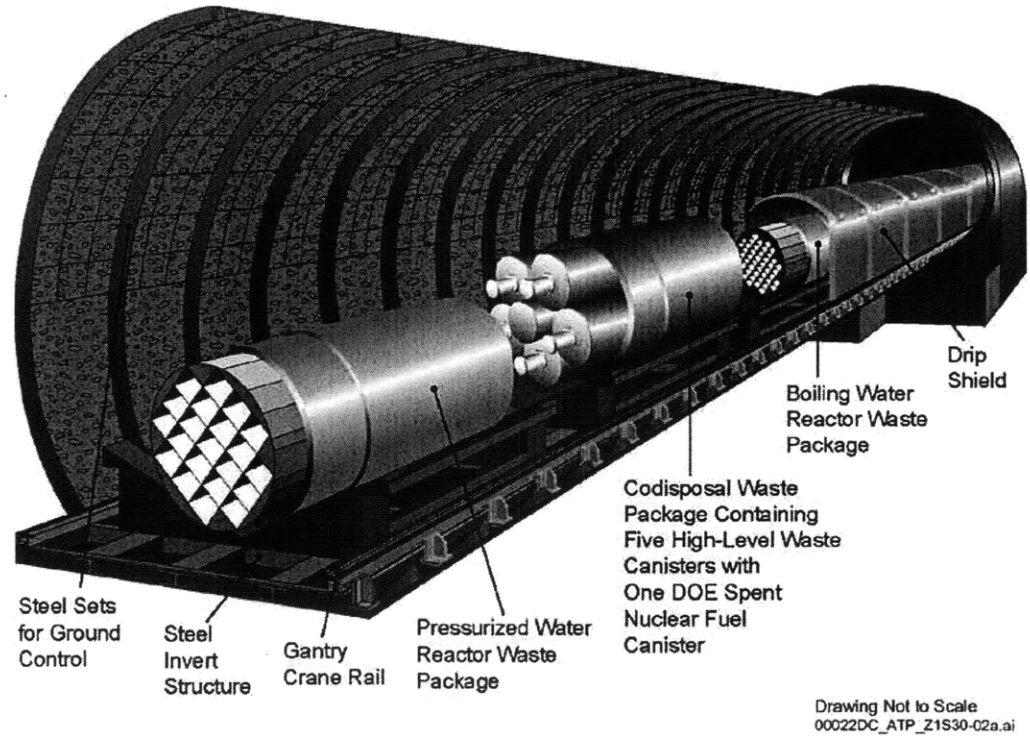


Figure 3.14. Yucca Mountain Emplacement Drift Showing Several Waste Package Types (3.26).

The deep borehole concept utilizes existing technology from the oil exploration and drilling industry and while extremely complex drilling orientation and branching methods currently exist, the proposed waste management strategy will employ the simple vertical design shown in Figure 3.15. Previous studies reviewed the engineering requirements of such a design and suggest that the telescoping nature of the drilling and exponential cost as a function of depth, as shown in Figure 3.16, lead to an optimal depth of 5 kilometers. This finding conservatively assumed only 3 kilometers of active repository depth with a 2 kilometer seal and backfill region and assumed the use of single assembly canisters. The MIT report on geothermal energy estimated an appropriately designed 5 kilometer hole would cost roughly $7M in 2004$. Given the use of 5 meter canisters each holding a single assembly of roughly half a metric tonne of initial heavy metal, the subsurface construction cost required to house the 83,800 MT capacity proposed in the Yucca Mountain Total System Life Cycle Cost report would total $1.79B in 2000$. Of course, the borehole strategy's major advantage comes in the modularity of cost so the $6.4M, in 2000$, per hole expenditure is applied in a more continuous fashion versus a large single capital expenditure experienced in a Yucca Mountain style repository. The higher subsurface estimate of $3.11B corresponds to a drilling cost of $10.0M per hole, while the lower subsurface estimate of $0.85B reflects the more optimistic plan proposed in the Guerin and De Roo reports that suggest cost savings from going to slightly shallower, 4 kilometer holes at $7.3M each, and widening the boreholes to handle 4 assemblies per canister.

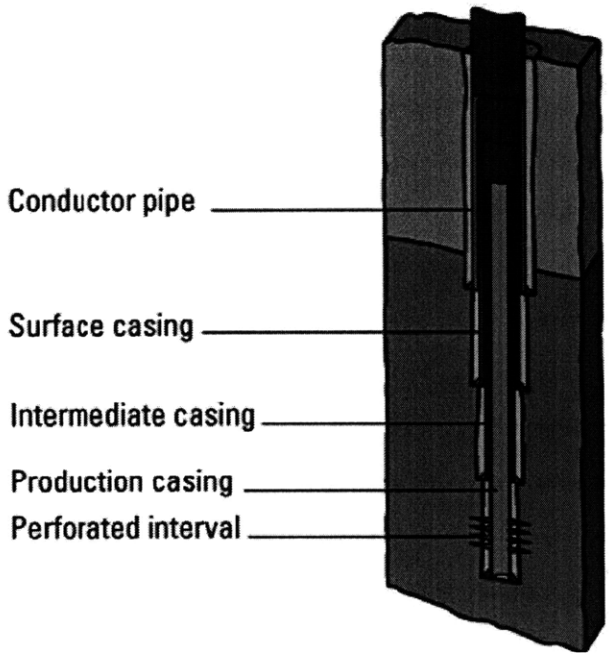
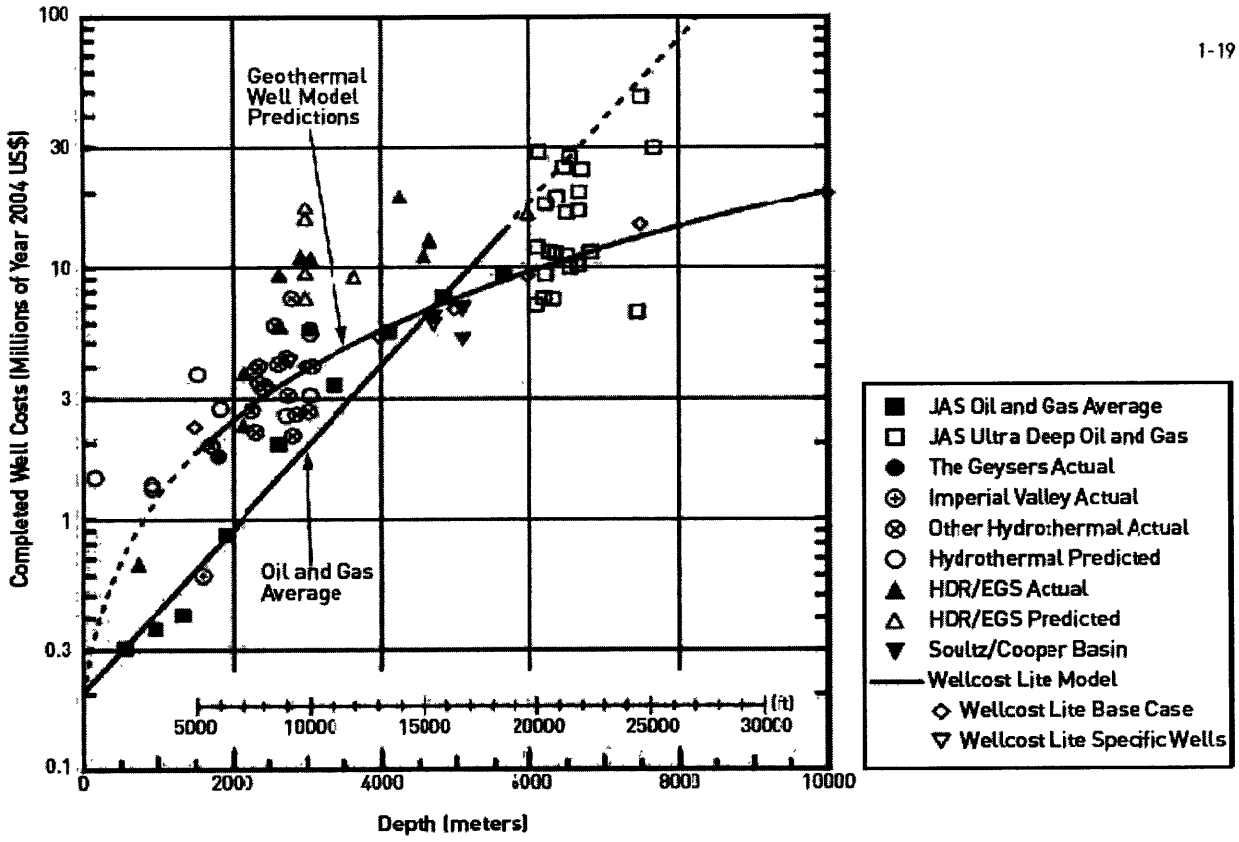


Figure 3.15. Typical Casing Illustrating the Telescoping Nature of Vertical Borehole Drilling (3.30).



1. JAS = Joint Association Survey on Drilling Costs.
2. Well costs updated to US$ (yr. 2004) using index made from 3-year moving average for each depth interval listed in JAS (1976-2004) for onshore, completed US oil and gas wells. A 17% inflation rate was assumed for years pre-1976.
3. Ultra deep well data points for depths greater than 6 km are either individual wells or averages from a small number of wells listed in JAS (1994-2000).
4. "Other Hydrothermal Actual" data include some non-US wells (Source: Mansure 2004).

Figure 3.16. Completed Oil, Gas, and Geothermal Well Costs as a Function of Depth (3.31).

The Forrest and Rogers report estimated the monitoring expenses associated with borehole waste disposal at just $1.55B, a significantly lower cost than estimates for the analogous expenditures in the Yucca Mountain Project because of the elimination of the ventilation requirements and the drastic reduction of the retrievability. Nominally, we assume the borehole waste system has the same closure and decommissioning costs as the Yucca Mountain Project, but the lower estimate of $0.22B reflects an estimated closure cost identical to the closure of the surface facilities plus the barebones cost of just the seal* with assumed free backfill.

One should note that the simulations performed in this report assume sufficient aging of the spent nuclear fuel for both repository concepts, such that a significant use of additional surface storage at the

* Hole Seal Volume ≈ π*(0.28m)^2*1000m = 246m^3, Bentonite Density = 0.593MT/m^3, Bentonite Cost = $340/MT

disposal locations is not required. This assumption is justified by the fact that much of the current accumulation of spent fuel has aged for decades, so no shortage of disposal-ready spent fuel should be expected, and because this might only be an issue in cases of an early repository opening with a high acceptance rate. Table 3.21 summarizes the costs for the borehole repository concept and Table 3.22 summarizes these costs for a mined geologic repository similar to the Yucca Mountain Project.

Table 3.21. Borehole Repository Cost Summary.

Variable	Nominal	Low	High
Surface Licensing	$0.31B	$0.28B	$0.34B
Surface Capital	$1.78B	$1.20B	$1.96B
Surface Monitoring	$0.71B	$0.64B	$0.78B
Surface Closure	$0.21B	$0.19B	$0.23B
Subsurface Licensing	$0.19B	$0.17B	$0.21B
Subsurface Capital	$1.79B	$0.85B	$3.11B
Subsurface Monitoring	$1.55B	$1.40B	$1.71B
Subsurface Closure	$0.47B	$0.22B	$0.47B*

Table 3.22. Yucca Mountain Repository Cost Summary.

Variable	Nominal	Low	High
Surface Licensing	$0.31B	$0.28B	$0.34B
Surface Capital	$1.78B	$1.20B	$1.96B
Surface Monitoring	$0.71B	$0.64B	$0.78B
Surface Closure	$0.21B	$0.19B	$0.23B
Subsurface Licensing	$0.19B	$0.17B	$0.21B
Subsurface Capital	$1.20B	$1.20B†	$3.60B‡
Subsurface Monitoring	$2.18B	$1.96B	$2.40B
Subsurface Closure	$0.47B	$0.42B	$0.52B

* The $0.47B equivalent closure cost for the borehole repository is considered extremely conservative.
† The $1.20B subsurface capital cost is considered extremely optimistic.
‡ The new OCRWM TSLCC report expected in July 2008 is rumored have subsurface construction costs that are estimated at multiples of the 2001 estimates. We have assumed a multiple of 3, purely by speculation.

Summary

Table 3.24 presents a summary of the individual operation levelized waste management expenses considered in tallying the total system life cycle cost. As a rule of thumb, the term "operation levelized cost" refers to the cost that the manager of a particular function might quote, while "system averaged cost" takes into account the fact that not every spent fuel assembly goes through all of the possible modules at the hypothesized lengths of time. For instance, the $113/kgIHM value for shutdown spent fuel pool storage represents the average cost per kilogram of initial heavy metal to store 1,000 metric tonnes in a spent fuel pool of a shutdown reactor for 10 years. Assuming only 10 percent of the waste generated in the U.S. went through this process reduces the portion of the system averaged cost associated with shutdown spent fuel storage to just $11 per kilogram.

Table 3.23. Summary of Mined Repository Operation Levelized Waste Management Costs.

Cost Category	Operation Levelized Cost
Shutdown Spent Fuel Pool Storage*	$113/kgIHM
Initial Conditioning for Transp. and Dry Storage	$150/kgIHM
Online Reactor Onsite Dry Storage†	$12/kgIHM
Shutdown Reactor Onsite Dry Storage‡	$57/kgIHM
Inter-region Transportation	$7 - $14/kgIHM
Intra-region Transportation	$20 - $26/kgIHM
Centralized Interim Storage§	$31/kgIHM
Final Conditioning and Packaging (MGR)	$274/kgIHM
Repository, Monitoring, and Closure (MGR)**	$118/kgIHM

Table 3.24. Summary of Operation Levelized Repository Costs for Borehole Disposal.

Cost Category	Operation Levelized Cost
Final Conditioning and Packaging (DBH)	$66/kgIHM
Repository, Monitoring, and Closure (DBH)	$101/kgIHM

* 1,000 MT for 10 years.
† 1,000 MT for 10 years.
‡ 1,000 MT for 10 years.
§ 40,000 MT for 50 years.
** Assumes 93,200 MT, ignores indirect financing charges, but includes $30M/yr base O&M cost for 80 years.

Chapter 3 References

3.1. **Saling, James H and Fentiman, Audeen W.** *Radioactive Waste Management.* 2nd Edition. New York, NY : Taylor & Francis, 2002. ISBN 1-56032-842-8.

3.2. **Office of Civilian Radioactive Waste Management.** Unit 1 - Lesson 3 Reading (OCRWM). [Online] Department of Energy. [Cited: July 4, 2008.] http://www.ocrwm.doe.gov/curriculum/unit1/lesson3reading.shtml.

3.3. *The Impact of Dry Spent-Fuel Storage on Decommissioning.* **Bowser, R. C., Taylor, M. and Miller, K. R.** San Francisco, CA : American Nuclear Society, 1993. Transactions of the American Nuclear Society. Vol. 69, pp. 69-71. ISSN0003-018X.

3.4. **Idaho National Laboratory.** *Advanced Fuel Cycle Cost Basis.* Idaho Falls, ID : US Department of Energy, 2008. INL/EXT-07-12107.

3.5. *The Problems of Used Nuclear Fuel: Lessons for Interim Solutions from A Comparative Cost Analysis.* **Macfarlane, Allison.** s.l. : Energy Policy, 2001, Vol. 29, pp. 1379-1389.

3.6. **Nuclear Regulatory Commission.** Typical Dry Cask Storage System. *US Nuclear Regulatory Commission.* [Online] [Cited: May 16, 2008.] http://www.nrc.gov/waste/spent-fuel-storage/diagram-typical-dry-cask-system.html.

3.7. **Fairlie, Ian.** *Dry Storage of Spent Nuclear Fuel: The Safer Alternative to Reprocessing.* London, England : Greenpeace International, 2000.

3.8. *Used Nuclear Fuel-- What will we do with it?* **Hanson, Alan S.** Cambridge, MA : Massachusetts Institute of Technology, 2006. Nuclear Science and Engineering Seminar Series.

3.9. *Reducing the Hazards from Stored Spent Power-Reactor Fuel in the United States.* **Alvarez, Robert, et al.** 1, 2003, Science & Global Security, Vol. 11, pp. 1-51.

3.10. **Bunn, Matthew, et al.** *Interim Storage of Spent Nuclear Fuel.* Cambridge, MA : Harvard University Press, 2001.

3.11. *Interim Storage of Spent Fuel in the United States.* **Macfarlane, Allison.** 2001, Annual Review of Energy and the Environment, Vol. 26, pp. 201-235.

3.12. **Nuclear Energy Agency.** *The Economics of the Nuclear Fuel Cycle.* Paris, France : Organization for Economic Cooperation and Development, 1994. ISBN 92-64-14154-5.

3.13. **Office of Civilian Radioactive Waste Management.** *Analysis of the Total System Life Cycle Cost of the Civilian Radioactive Waste Management Program.* Washington, DC : U.S. Department of Energy, 2001. DOE/RW-0533.

3.14. **Nuclear Energy Agency.** *Accelerator-driven Systems and Fast Reactors in Advanced Nuclear Fuel Cycles.* Paris, France : Organization for Economic Cooperation and Development, 2002. ISBN 92-64-18482-1.

3.15. **Office of Civilian Radioactive Waste Management.** Transportation of Spent Nuclear Fuel - Fact Sheet. [Online] [Cited: July 7, 2008.] http://www.ocrwm.doe.gov/factsheets/doeymp0500.shtml.

3.16. *Borehole Disposal of Spent Nuclear Fuel Transportation Module.* **Gibbs, Jonathan S.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2008. 22.812 Managing Nuclear Technology.

3.17. *Recent License Application and Status of Yucca Mountain Project.* **Sproat, Edward F.** Cambridge, MA : Massachusetts Institute of Technology, 2008. Nuclear Science and Engineering Seminar Series.

3.18. **Nuclear Energy Agency.** *Advanced Nuclear Fuel Cycles and Radioactive Waste Management.* Paris, France : Organization for Economic Cooperation and Development, 2006. ISBN 92-64-02485-9.

3.19. **Private Fuel Storage, LLC.** *Application for a License to Construct and Operate a Spent Fuel Storage Facility.* s.l. : Nuclear Regulatory Commission, 1997. Docket Number 72-22.

3.20. **Private Fuel Storage, LLC.** *Final Environmental Impact Statement for the Construction and Operation of an Independent Spent Fuel Storage Installation on the Reservation of the Skull Valley Band of Goshute Indians and the Related Transportation Facility in Tooele County, Utah.* s.l. : Nuclear Regulatory Commission, 2001. NUREG 1714 Volume 1.

3.21. **Private Fuel Storage, LLC.** The Private Fuel Storage Facility. [Online] [Cited: May 16, 2008.] http://privatefuelstorage.com/project/facility.html.

3.22. *Surface Facilities Cost Module.* **Forrest, Eric and Rogers, Bradley.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2008. 22.812 Managing Nuclear Technology.

3.23. **Hoag, Ian.** *Canister Design for Deep Borehole Disposal of Nuclear Waste.* Cambridge, MA : MIT Nuclear Science and Engineering, 2006.

3.24. *Canisters: Cost Assessment.* **Guerin, Laurent.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2008. 22.812 Managing Nuclear Technology.

3.25. **TPS Technitube.** TPS TECHNITUBE RÖHRENWERKE GmbH - OCTG - Tubing And Casing (API Casing). [Online] [Cited: July 10, 2008.] http://www.tps-technitube.de/Products/API_Casing/.

3.26. **Office of Civilian Radioactive Waste Management.** *Yucca Mountain Science and Engineering Report.* Washington, DC : U.S. Department of Energy, 2002. DOE/RW-0539-1.

3.27. *Advances in Yucca Mountain Design.* **Harrington, P G, et al.** Tucson, AZ : Office of Repository Development, 2003. Waste Management Conference.

3.28. *Surface Facilities Design.* **Gardiner, James T.** Las Vegas, NV : Office of Civilian Radioactive Waste Management, 2003. Nuclear Waste Technical Review Board Panel on the Waste Management System.

3.29. **Office of Civilian Radioactive Waste Management.** *Reference Design Description for a Geologic Repository.* Washington, DC : U.S. Department of Energy, 2000. TDR-MGR-SE-000008 Rev 03.

3.30. *Borehole Economic Module.* **De Roo, Guillaume.** Cambridge, MA : MIT Nuclear Science and Engineering Department, 2008. 22.812 Managing Nuclear Technology.

3.31. **Massachusetts Institute of Technology.** *The Future of Geothermal Energy.* Idaho Falls, ID : U.S. Department of Energy, 2006. ISBN 0-615-13438-6.

Chapter 4:
Policy Analysis

Introduction

This chapter illustrates the impact of various policy decisions through SNuFManager simulations in an effort to gain an understanding of the effects of those choices in an inexpensive computational experiment without the need to make costly mistakes in real life. The results from this chapter will show the magnitude of the economic impacts from delaying the opening of a repository as well as changing the unloading algorithm that determines which fuel takes priority in moving through the waste management system given the expected resources and available moving capacity each year. The simulated results will also illustrate the impact on diffusing the expenditures by going to a modular waste management strategy using deep borehole disposal and the potential benefits to increasing the waste removal rates from the fleet of nuclear reactor sites by operating multiple repositories and opening one or more interim storage sites. As a reminder, all costs are expressed in constant 2000$ expenditures in an effort to stay consistent with the OCRWM's 2001 Total System Life Cycle Cost report for the Yucca Mountain Project.

4.1 Base Case

While the absolute values of the results from the economic simulator may fall within a realistic range of expected waste management total system life cycle costs, SNuFManager primarily aids policy analysis by reasonably estimating the sensitivity of incremental system changes and therefore an analyst should always index the results to a reference run of the program. The thesis at hand follows this recommendation and measures the magnitude of the changes in results with respect to the so-called "base case" that utilizes the simple unloading algorithm, which takes fuel from the oldest reactors first, and opens a national mined geologic repository, like Yucca Mountain, in 2020. Such a timetable appears optimistic given the historical record of delays and false starts in dealing with the waste management problems and the recent comments of Edward Sproat, the Director of the Office of Civilian Radioactive Waste Management, that Yucca Mountain could begin significant acceptance of spent fuel as soon as 2020, "but only if adequate funding is provided (4.1)."

Table 4.1 summarizes the input parameters in the base case simulation. Notice the simulation period runs from 1998 to 2100, while later figures only present the results from 2000 to 2100. This small introductory period provides stabilization because SNuFManager requires that all reactor shutdowns occur during the simulation and therefore the base case artificially assigns a shutdown year of 1998 to all reactors that went offline prior to the time period of interest. Also note that the Department of Energy generated the spent fuel database used to define the initial conditions on nuclear waste in 2002 (4.2). In order to start the simulation in 1998, the input utilizes the same global assumptions on capacity factor, burnup, and thermodynamic efficiency outlined later in this section to work backwards and estimate the conditions at the beginning of the simulation time. In doing so, the input allows the use of reasonable starting conditions and ensures the matching of conditions in 2002.

The base case avoids the use of interim storage, but notice the input summary only lists a capacity instead of an array of expansion dates. While SNuFManager has the ability to expand interim storage sites as needed, this effect will not greatly alter the cash flows because of the relatively modest initial capital of interim storage in comparison with repository costs.

Table 4.1. Base Case Input Summary.

Time Parameters	Value	Spent Fuel Pool Parameters	Value
Simulation Period	1998 - 2100	Minimum Cooling	5 years
Interim Storage Parameters	Value	Repository Parameters	Value
Number of Sites (Loc.)	0* (N/A)	Number of Sites (Loc.)	1 (IV)
Capacities†	N/A	Number of Expansions‡	1
Opening Dates	N/A	Capacities	93,200 MT
Acceptance Rates	N/A	Expansion Dates	2020
Initial Capital	N/A	Acceptance Rates	3,000 MT/year
O&M Base	N/A	Total Capital	$3.48B
O&M Loading	N/A	O&M Base	$44.5M/year
O&M Monitoring	N/A	Closure	$0.68B
Other Economic Parameters	Value	Other Economic Parameters Continued	Value
Initial Conditioning	$150/kgIHM	Shutdown SFP	$11.3M/year
Repos. Cond. & Pack.	$274/kgIHM	Initial Onsite DS	$6.0M
Inter-region Transport	$7 - $14/kg	Online DSC Monitoring	$0.64M/year
Intra-region Transport	$20 - $26/kg	Offline DSC Monitoring	$5.1M/year

The system behaves normally and Figure 4.1 illustrates the expected filling of spent fuel pools, increased use of onsite dry storage, and eventual transfer to the geologic repository. Notice that the simulation shows a little less than 45,000 MT of spent nuclear fuel in the year 2000, while Figure 1.4 from the Keystone report estimates slightly more than 40,000 MT. The Keystone report goes on to project an accumulation of 110,000 to 130,000 MT by 2050, however those estimates include many license renewals not assumed in the base case of this thesis and so the projection of nearly 95,000 MT of spent fuel by 2050 appears within reason (4.3). This thesis also assumes all reactors operate from 1998 onward with 3 batch cores and cycle lengths of 1.5 years. In addition, SNuFManager globally assigns a 50 GWD/MT burnup, 85% capacity factor, and 33% thermodynamic efficiency while the Keystone report neglects to completely state the assumptions necessary in making their projections. One can speculate

* Although the scenario has no interim storage, the current version of SNuFManager requires one be specified, so for the purposes of the simulation an interim storage site of negligible size is modeled.
† SNuFManager allows interim storage expansions, but this policy will not be analyzed in detail.
‡ The initial opening counts as the first expansion.

that they might include more detailed assumptions specific to each reactor or they may use global assumptions that naturally lead to higher waste accumulation. Whatever the reason, the discrepancy appears to be within the uncertainty of the evaluations and falls far short of causing concern over the reasonableness of the results.

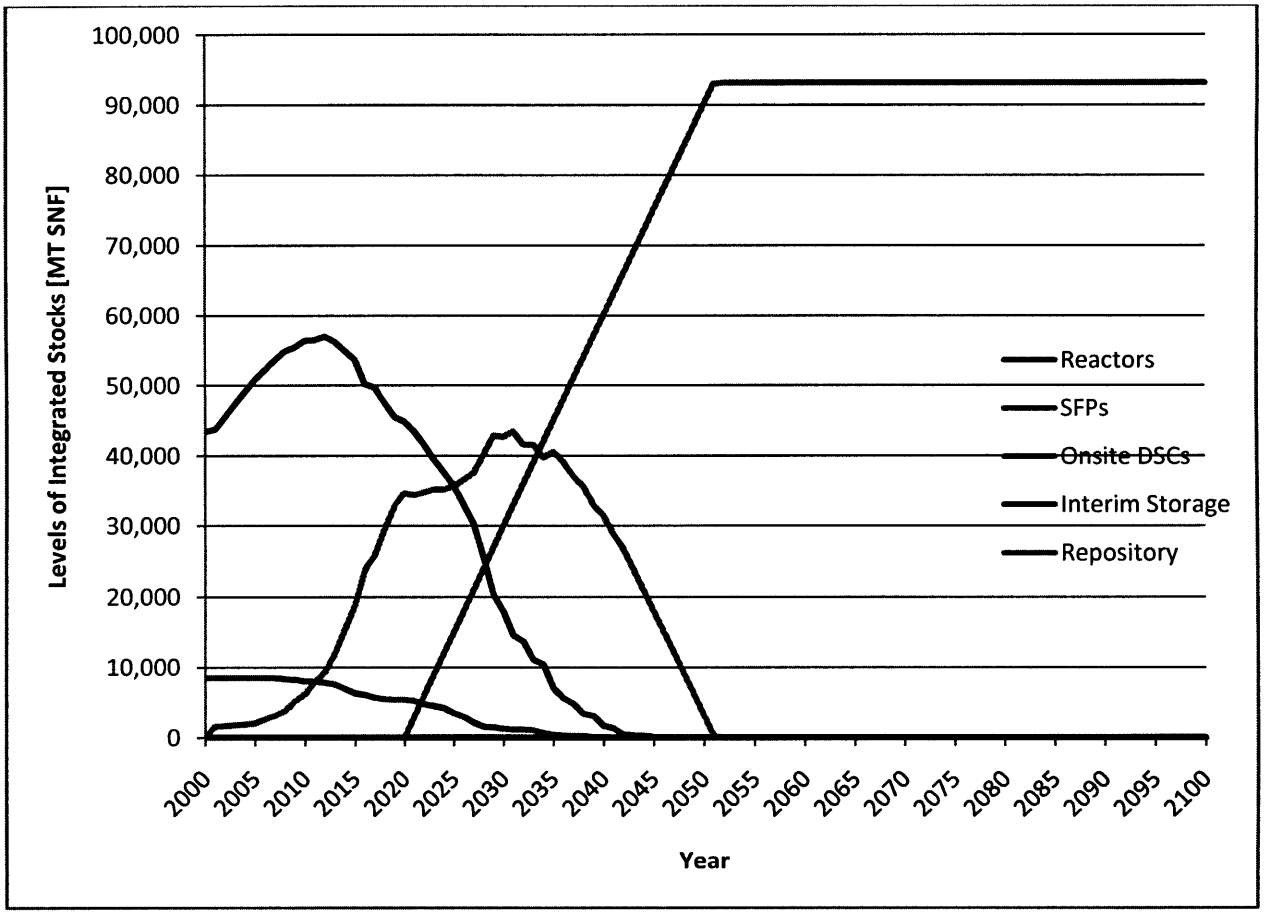


Figure 4.1. Base Case Levels of Integrated Stocks of Nuclear Fuel.

SNuFManager estimates a total system life cycle cost of roughly $60B from 2000 to 2100 for the base case waste management strategy. Figure 4.2 shows the expected annual expenditures in constant 2000 dollars and identifies the costs associated with shutdown spent fuel pools, initial conditioning of waste packages for dry storage and transportation, onsite dry storage, transportation, interim storage, final conditioning for geologic disposal, and those associated with the repository. Table 4.2 summarizes these costs and shows the majority of the waste management expense in the base case goes to repository conditioning, initial conditioning, and the construction and operation of the repository itself.

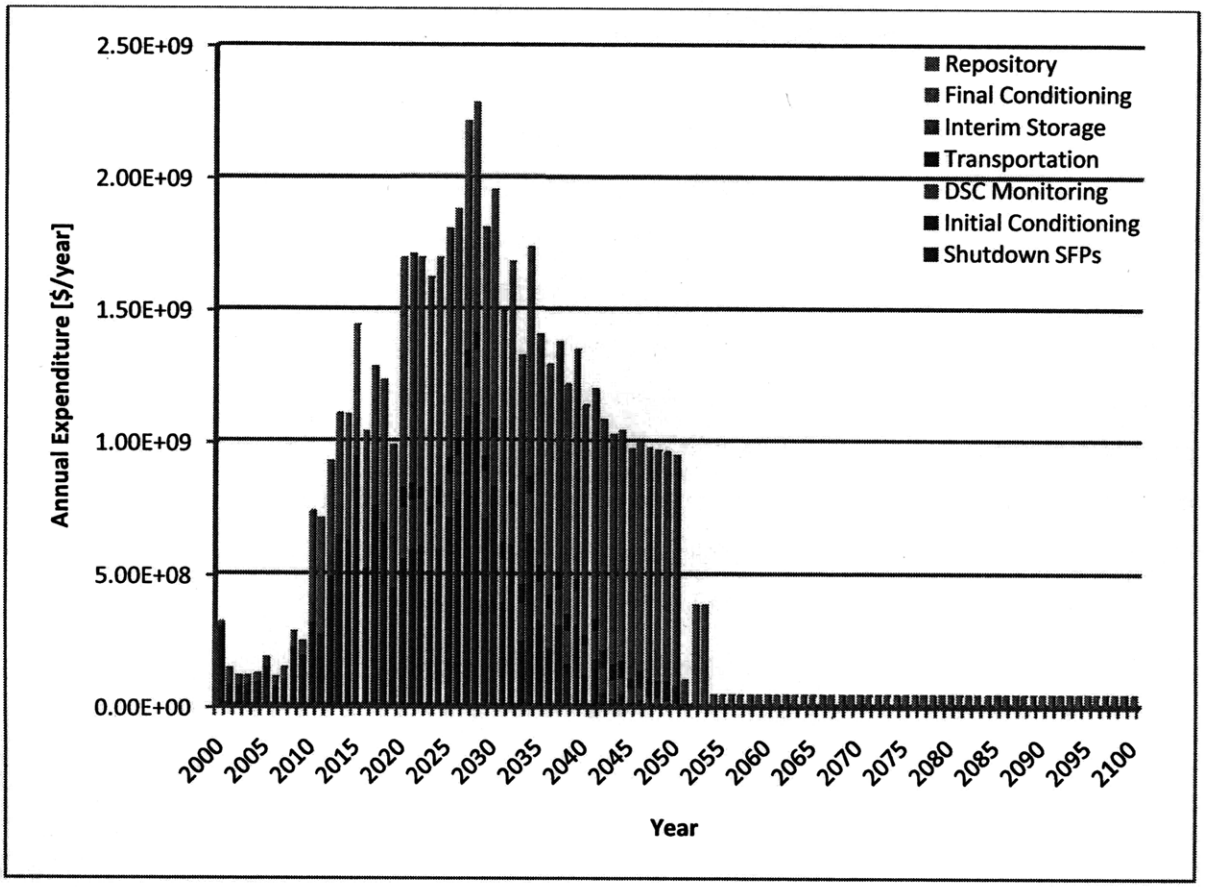


Figure 4.2. SNuFManager Estimated Annual Cash Flow for the Base Case Strategy.

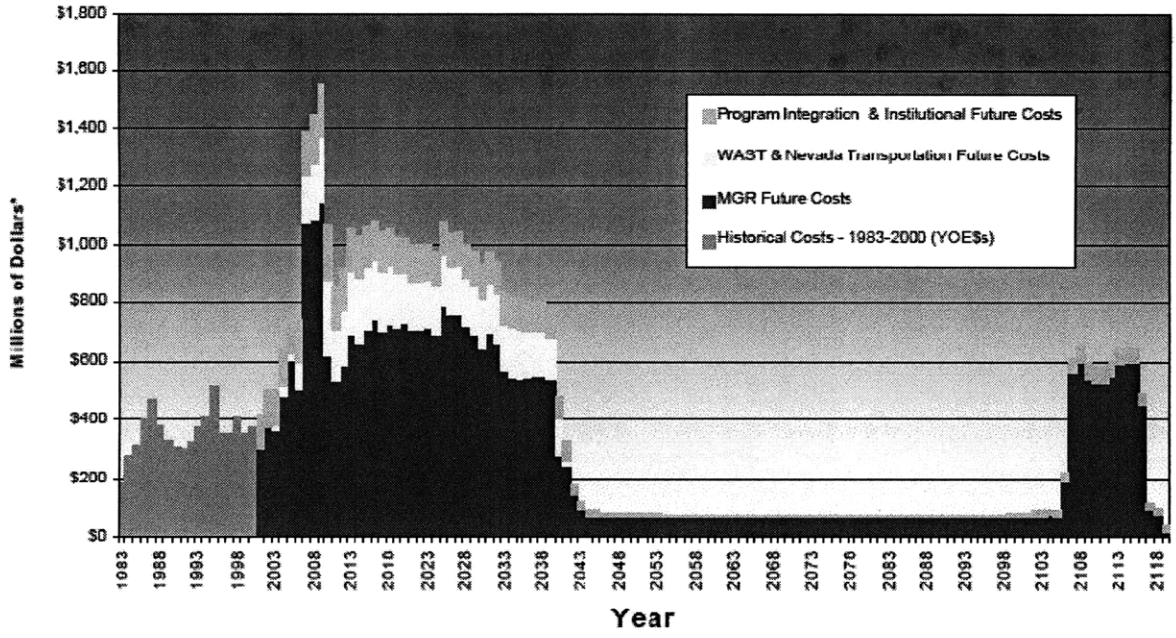
Table 4.2. Base Case Total System Life Cycle Cost Summary.

Cost Category	Total	System Averaged Cost*	Percentage
Shutdown SFPs	$5.0B	$54/kgIHM	8.4%
Initial Conditioning	$14.0B	$150/kgIHM	23.4%
Onsite Dry Storage	$5.6B	$60/kgIHM	9.3%
Transportation	$2.0B	$22/kgIHM	3.3%
Interim Storage	$0.0B	$0/kgIHM	0.0%
Final Conditioning	$25.5B	$274/kgIHM	42.7%
Repository	$7.8B	$83/kgIHM	13.0%
Total	$59.9B	$642/kgIHM	100.0%

* Based on the SNuFManager estimate of 93,200 MTIHM total spent fuel accumulation.

One should immediately notice the system averaged waste management cost of $642 per kilogram of initial heavy metal in the spent nuclear fuel significantly exceeds the roughly $400 per kilogram allowance provided by the 1 mill per electrical kilowatt nuclear waste fee. Assessments of nuclear waste fee adequacy extend beyond the scope of this thesis, but the divergence merits attention. The disparity may not actually lead to waste management deficits because when the Department of Energy conducts its adequacy report on the nuclear waste fund fee, it considers interest on its fund. Therefore, by blocking spending of the collection, Congress has actually provided a significant cash flow from interest able to contribute to future waste management expenditures (4.4). More importantly, the waste fee generally does not apply toward the first three expense categories shown in all the cost summary tables, which sum to $264 per kilogram, with the possible exception of some fraction of the initial conditioning that can be considered "transportation preparation." However, these costs are important in considering the economics of the entire back-end of the fuel cycle and in creating intelligent policy initiatives that avoid the litigations the government currently faces by not taking possession of the accumulating spent nuclear fuel.

Figure 4.3 shows the Office of Civilian Radioactive Waste Management annual cost profile of the total system life cycle cost of the Yucca Mountain project, and clearly differs in distribution. The major differences include a large spike in expenditures at the opening of the repository, a long delay before considering closing costs, and a greater amount spent on closing the repository. Even with these differences, the similarities make meaningful interpretation of the results promising. In both cost profiles, the project incurs the bulk of the expenditures in the first half of the twenty-first century and the total cost matches reasonably well, $60B compared with $58B. In fact, the unpublished, but more recent Department of Energy cost estimate for the total system lifecycle cost reaches over $90B in current dollars, but both DOE reports include historical costs, some of which are not considered in this thesis (4.1). While the closure costs admittedly occur much later than SNuFManager shows, this thesis keeps the convention of applying these costs immediately after the repository fills for simplicity of programming and to ensure they always show up in the twenty-first century time horizon.



* Note: Historical Costs from 1983-2000 are shown as year of expenditure dollars (YOE$s)
Costs from 2001-2119 are constant 2000 dollars (2000$s)

Figure 4.3. Official Yucca Mountain Project Annual Total System Life Cycle Cost Profile (4.5).

4.2 Repository Opening Time

In analyzing the effect of delaying the transfer of ownership for the spent nuclear fuel, one must carefully minimize the number of other variables allowed to change. In this case, only the date of the opening of the geologic repository changes in 5 year increments starting at 2020 with the base case and going up to 2060. Figure 4.4 shows the changing profile of the level of spent fuel stored in onsite dry storage. Since the cost increases with the integral of the curves shown in Figure 4.4, delaying the opening of a geologic repository or some other means of removing the spent fuel from the many U.S. reactor sites adds an increasing burden on the utilities, which in turn sue the government, likely at a premium above the real cost of monitoring. Notice also that all reactors shutdown by 2040 in the base case and so by that time all reactor sites effectively operate as highly inefficient and highly expensive interim storage facilities.

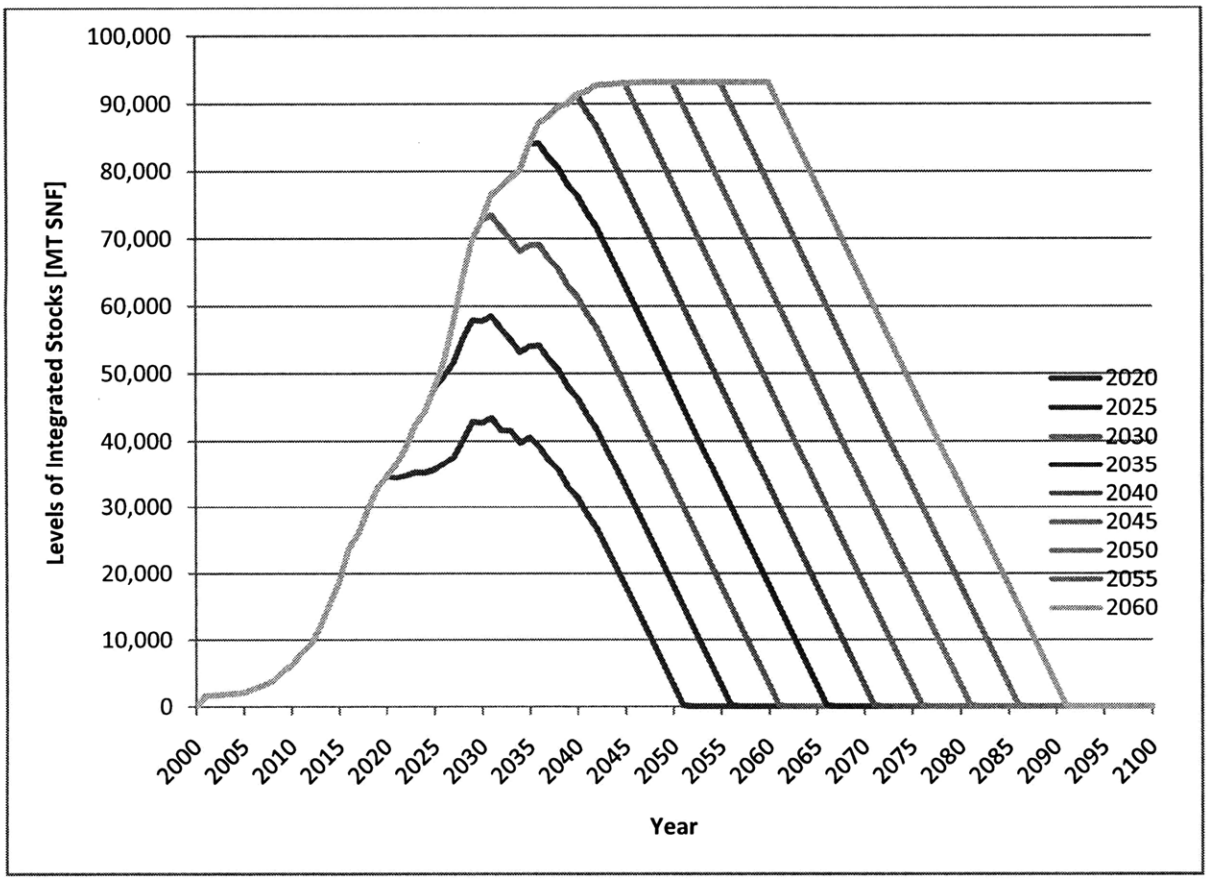


Figure 4.4. Level of Spent Nuclear Fuel in Onsite Dry Storage Given Repository Delays.

Because the incremental area under each successive curve in Figure 4.4 remains approximately constant, the total system life cycle cost increases linearly with the repository opening date at over $330M per year of delay coming exclusively from the cost of onsite storage, often after reactors shut down. While not exactly matching official Department of Energy estimates, Director Sproat stated the cost of delay totaled $500M per year, but this estimate was probably expressed in current-year dollars and considers litigation expenses that likely include significant premiums above the actual cost of onsite dry storage (4.6). By simply increasing all dry storage costs in the inputs by 20% to account for this litigation premium and then adjusting the figure appropriately to account for inflation, one can match Director Sproat's estimate of the cost of delay for the Yucca Mountain Project.

Figure 4.5 illustrates the estimated total system life cycle cost for the repository opening times up to 2060 and clearly shows its linear relationship with delay time. As a reminder, the index increases almost entirely because of added onsite dry storage costs that nearly quadruple when the repository is delayed 40 years from the base case.

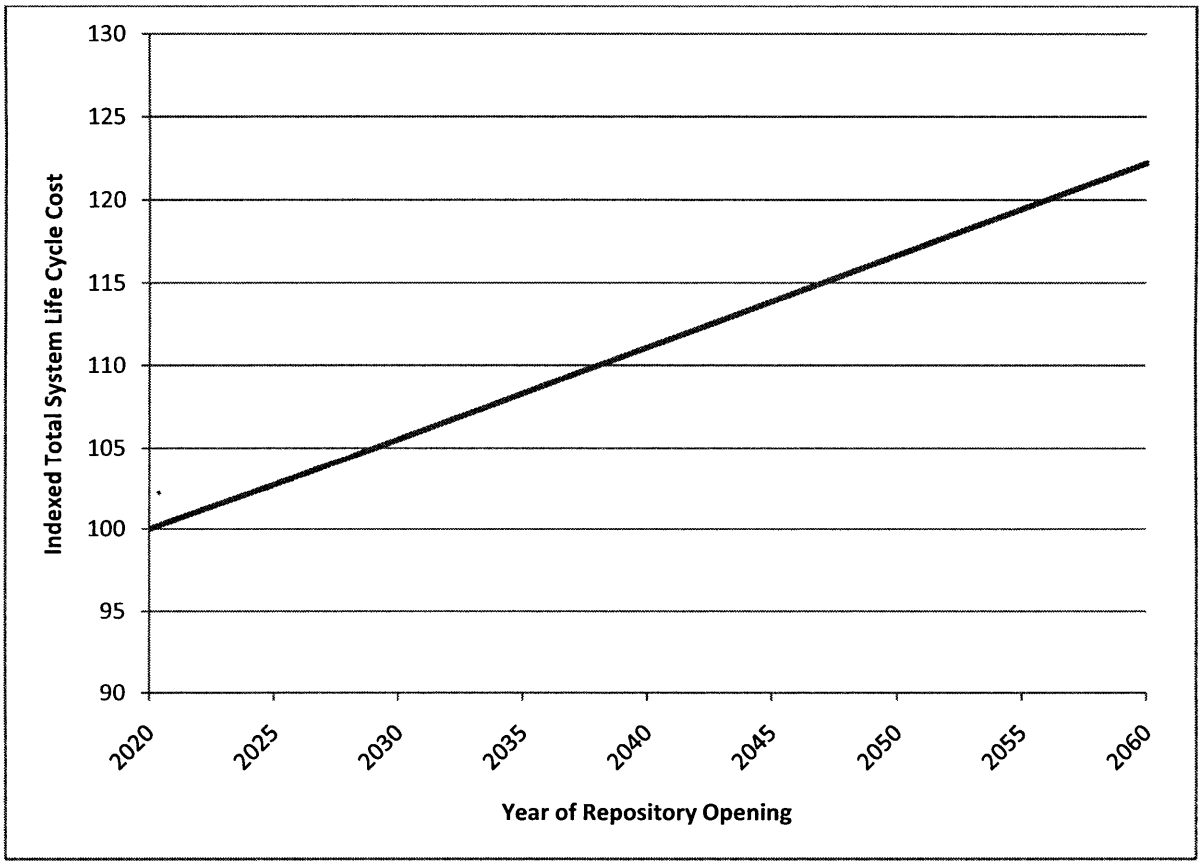


Figure 4.5. Total System Life Cycle Cost Index Versus Repository Opening Year.

Figures 4.6 and 4.7 illustrate the simulated cash flows for a repository opening date of 2040 and 2060, respectively. The delays spread the annual waste management expenditures over more years with a somewhat reduced maximum annual cash flow, but the magnitude of the costs of onsite dry storage monitoring grow alarmingly large and account for the drastic increase in the total system life cycle costs shown previously.

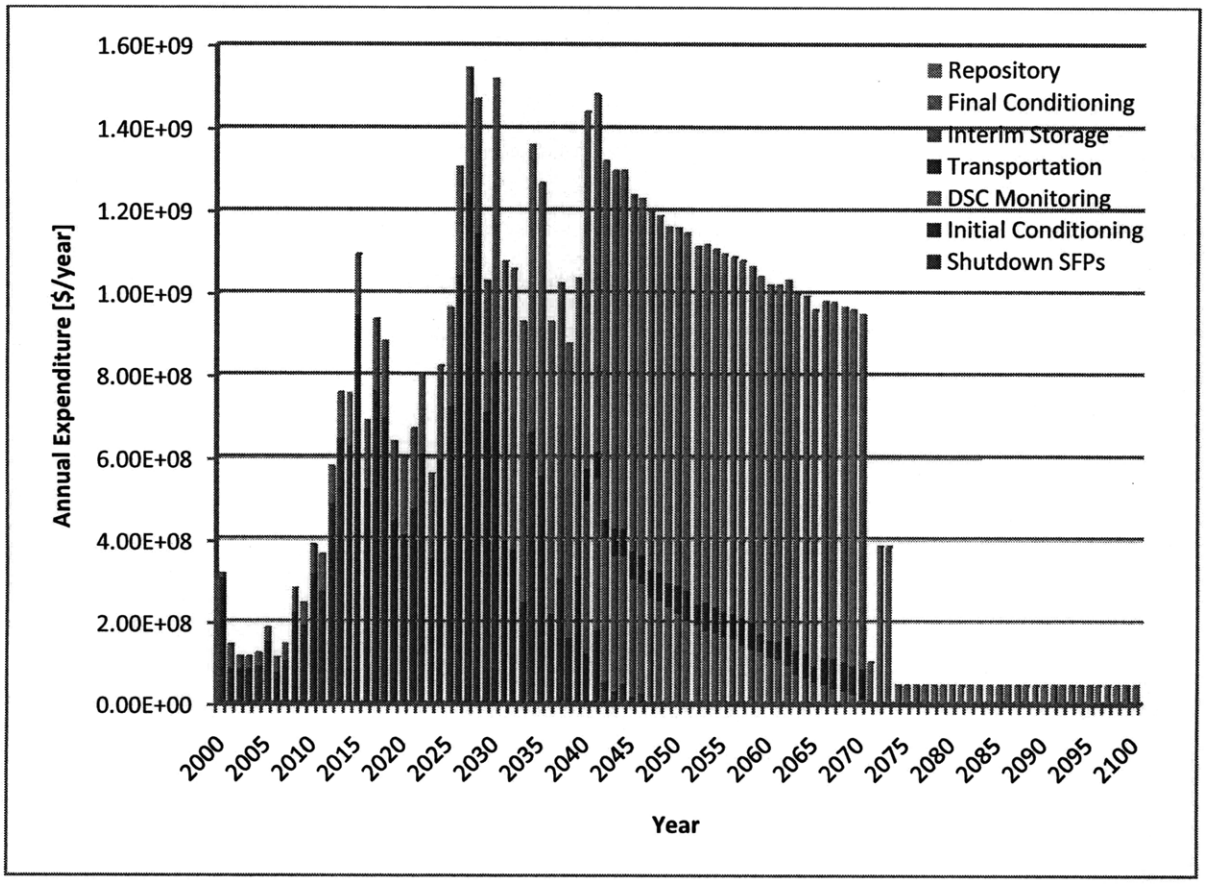


Figure 4.6. Estimated Annual Cash Flow for a 2040 Repository Opening.

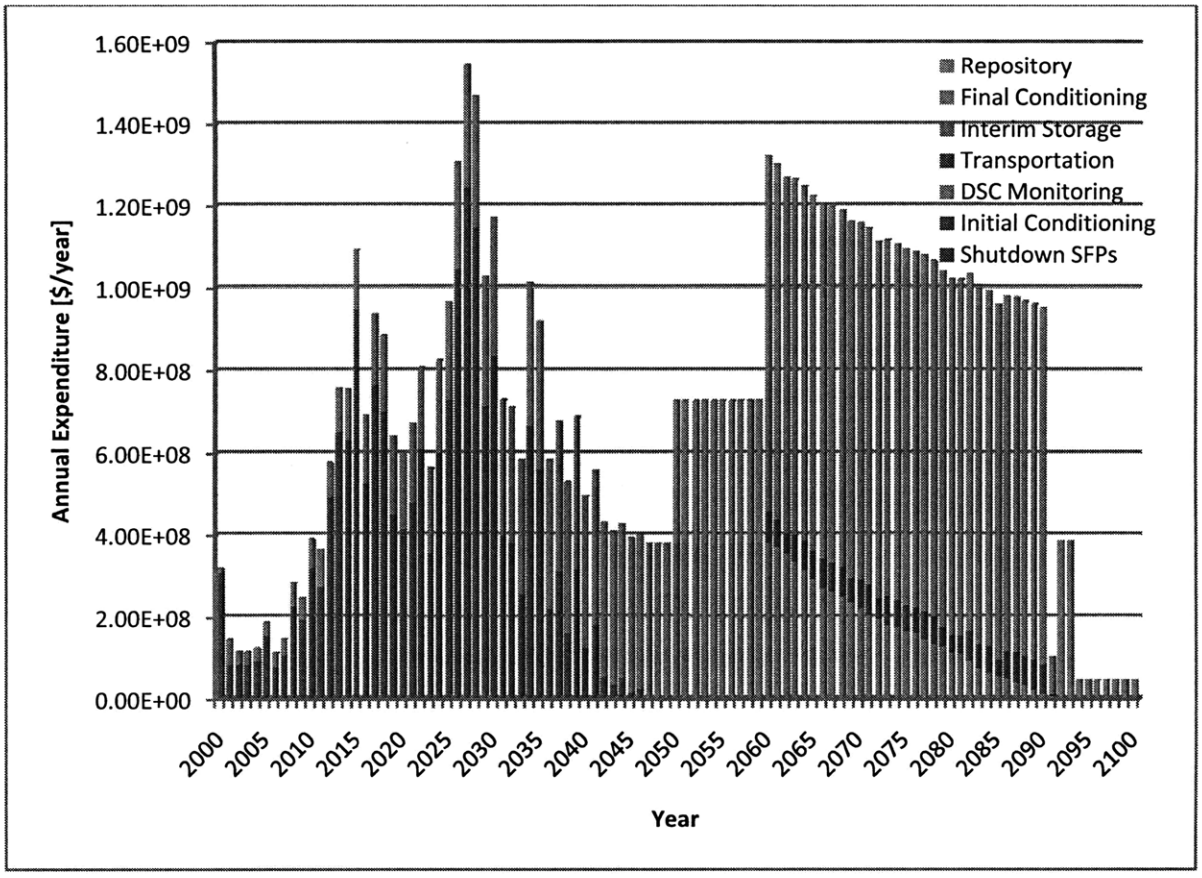


Figure 4.7. Estimated Annual Cash Flow for a 2060 Repository Opening.

4.3 Repository Modularity and Acceptance Rates

To analyze the economic benefits of a modular waste management strategy using deep boreholes, we must again compare the new strategy with one that minimizes the changes to the base case, as seen in Table 4.3. To this aim, an equivalent complex consisting of 311 boreholes, built in 11 expansions of 30 holes each (only 11 holes added in final expansion), replaces the Yucca Mountain repository. The first expansion includes the cost of licensing the surface and subsurface facilities as well as building the surface facilities. Major changes include a significant reduction in the base cost of $34.8M per year versus $44.5M and a drastically cheaper final conditioning and packaging cost of $66 per kilogram versus $274 per kilogram. In each case, the mined geologic repository is significantly more expensive because long-term retrievability must be maintained and the more robust casks must provide a radioactivity barrier for tens of thousands or hundreds of thousands of years.

Table 4.3. Deep Borehole Base Case Input Summary.

Time Parameters	Value	Spent Fuel Pool Parameters	Value
Simulation Period	1998 - 2100	Minimum Cooling	5 years
Interim Storage Parameters	Value	Repository Parameters	Value
Number of Sites (Loc.)	0 (N/A)	Number of Sites (Loc.)	1 (IV)
Capacities	N/A	Number of Expansions	11
Opening Dates	N/A	Capacities	9,000; ... ; 93,300 MT
Acceptance Rates	N/A	Expansion Dates	2020, 2023, ... , 2050
Initial Capital	N/A	Acceptance Rates	3,000 MT/year
O&M Base	N/A	Total Capital	$4.25B*
O&M Loading	N/A	O&M Base	$34.8M/year
O&M Monitoring	N/A	Closure	$0.68B
Other Economic Parameters	Value	Other Economic Parameters Continued	Value
Initial Conditioning	$150/kgIHM	Shutdown SFP	$11.3M/year
Repos. Cond. & Pack.	$66/kgIHM	Initial Onsite DS	$6.0M
Inter-region Transport	$7 - $14/kg	Online DSC Monitoring	$0.64M/year
Intra-region Transport	$20 - $26/kg	Offline DSC Monitoring	$5.1M/year

* $0.31B (Surf. Lic.) + $1.78B (Surf. Cap.) + $0.19B (Subsurf. Lic.) + 6.4[M$/hole]*ceil(93,200[MT]/300[MT/hole])

Figure 4.8 shows the new annual waste management expenditures for the deep borehole strategy. In this case, the reductions in the final conditioning and the repository charges clearly show themselves leading to only a $40.5B total system life cycle cost, $19.4B or 32% less than the Yucca Mountain type of geologic repository strategy. Table 4.4 outlines a summary of the costs and shows that at 34.5%, the initial conditioning replaces the final conditioning as the greatest cost contributor, followed by the final repository that accounts for 19.2% of the total system life cycle cost.

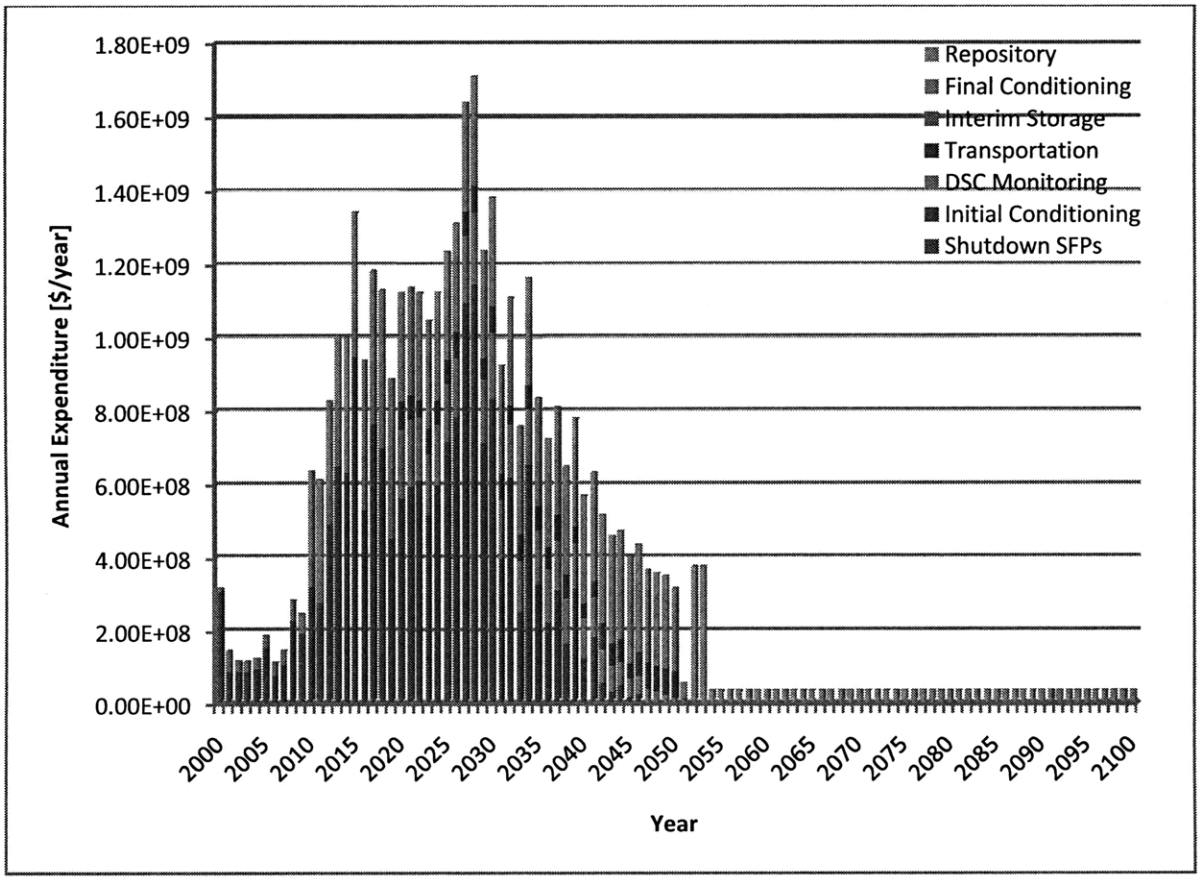


Figure 4.8. Estimated Annual Cash Flow for the Deep Borehole Base Case Strategy.

Table 4.4. Deep Borehole Base Case Total System Life Cycle Cost Summary.

Cost Category	Total	System Averaged Cost	Percentage
Shutdown SFPs	$5.0B	$54/kgIHM	12.4%
Initial Conditioning	$14.0B	$150/kgIHM	34.5%
Onsite Dry Storage	$5.6B	$60/kgIHM	13.7%
Transportation	$2.0B	$22/kgIHM	5.0%
Interim Storage	$0.0B	$0/kgIHM	0.0%
Final Conditioning	$6.2B	$66/kgIHM	15.2%
Repository	$7.8B	$83/kgIHM	19.2%
Total	$40.5B	$434/kgIHM	100.0%

By performing a net present value analysis of the total system expenses discounted at an annual rate of 5%, the borehole strategy appears to only be 21% better than the mined geologic repository, and by discounting at 10%, the borehole strategy appears to only be 16% better. What at first seems to be a paradox, in fact simply brings attention to the fact that the largest expense of the mined geologic repository approach is the final conditioning and packaging expenses. Because these expenses occur in the far future, the time value of those cash flows is less significant.

The benefits of modularity and the effects of discounting are best illustrated by focusing only on the repository capital costs to show the behavior that was originally expected for the total system life cycle cost. Let us take the capital cost of the mined geologic repository, $3.48B, and distribute it evenly over a 10 year period, as we have in our SNuFManager simulations. Let us also take the total capital cost of the deep borehole facility, $4.25B, and distribute it appropriately to reasonably reflect the expected annual expenditures. Figure 4.9 shows these undiscounted repository capital expense cash flows side by side. While the total borehole construction costs sum to more than the mined geologic repository, much of the borehole expense is incurred much later when a net present value analysis will significantly discount those cash flows.

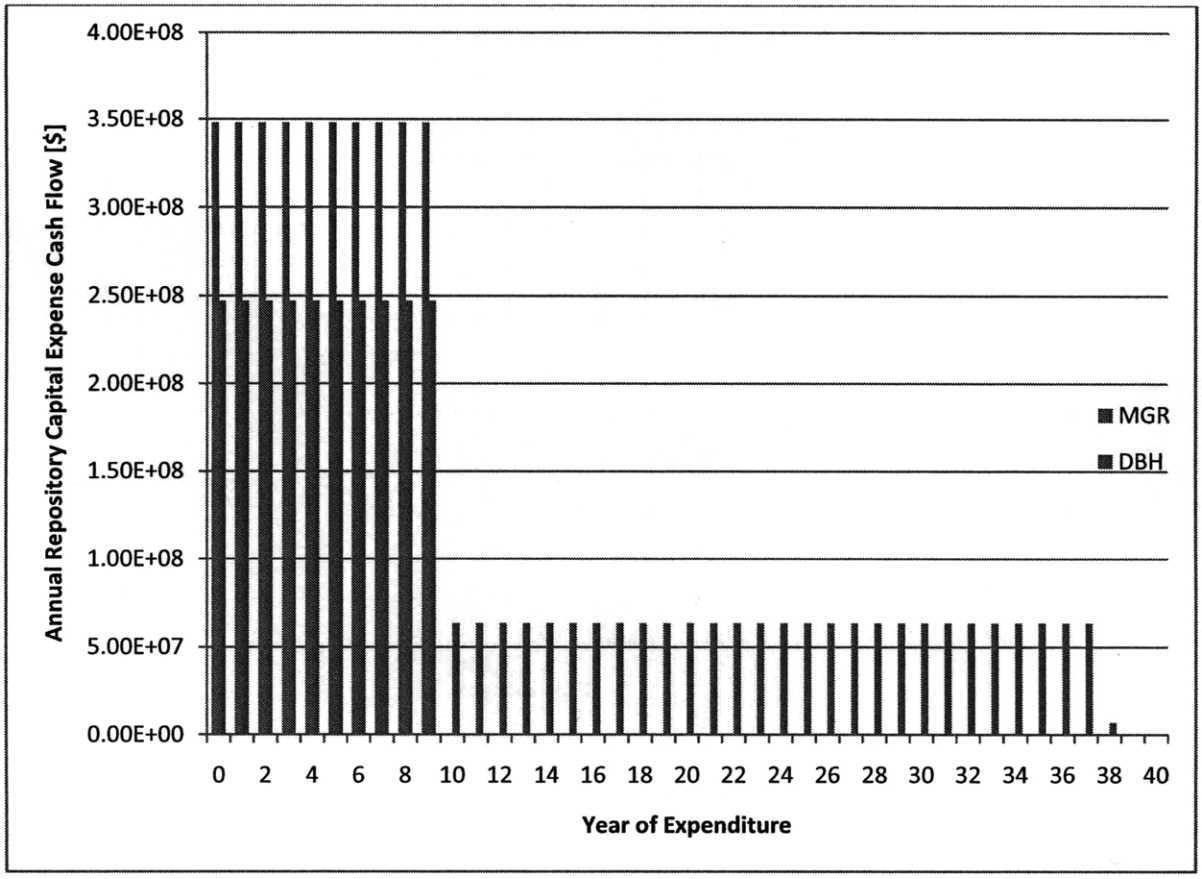


Figure 4.9. Undiscounted Annual Repository Construction Cash Flows.

Figure 4.10 illustrates the effect of discounting and shows that the net present value of the two different repository costs are equal if a discount rate of about 3% is assumed. Unfortunately, much of the SNuFManager analysis was completed before the publication of the most recent Department of Energy estimates for the Yucca Mountain Project total system life cycle costs, so the comparison of the strategies with the revised cost figures would be a beneficial contribution in the future. However, we can quickly demonstrate a component of the change simply by assuming a construction cost of $10B and showing net present value analysis even more dramatically illustrates the benefits of going to the deep borehole technology. Figure 4.11 shows how the net present value as a fraction of the undiscounted repository cost, decreases more sharply for the deep borehole case. Again, this added benefit is because a significant fraction of the borehole expense occurs much later in time.

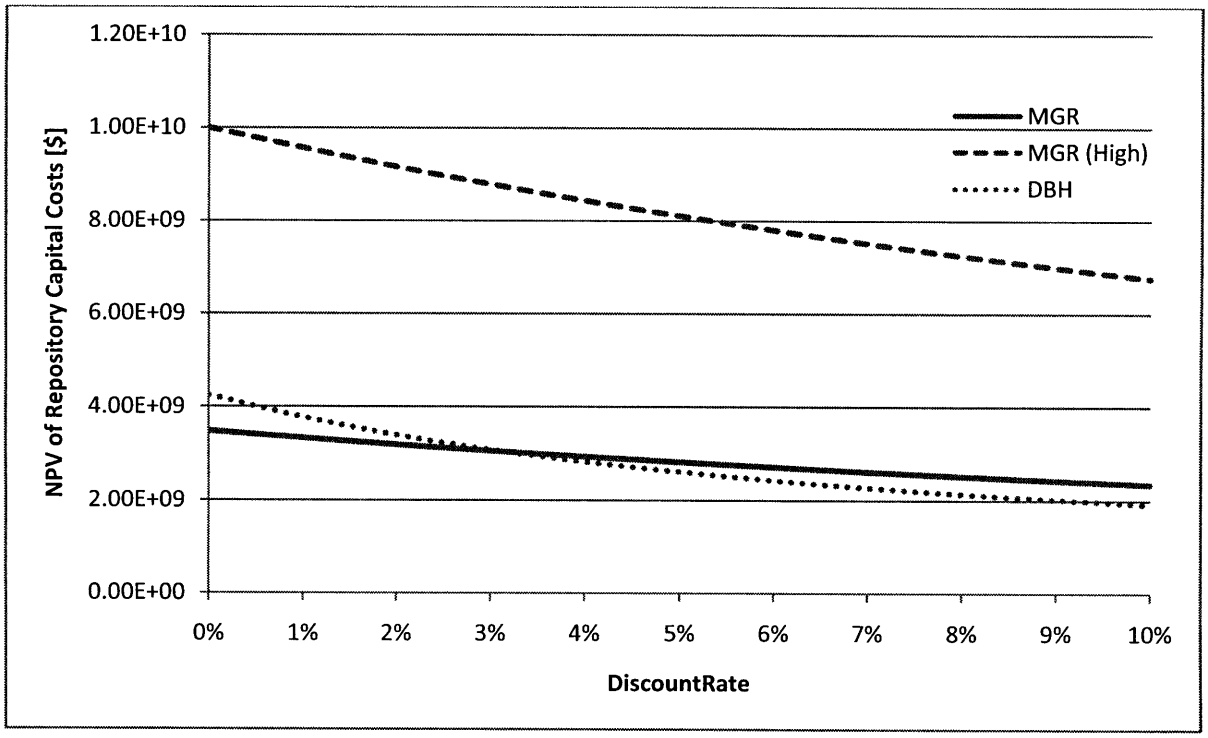


Figure 4.10. Net Present Value of Repository Construction Costs for Various Discount Rates.

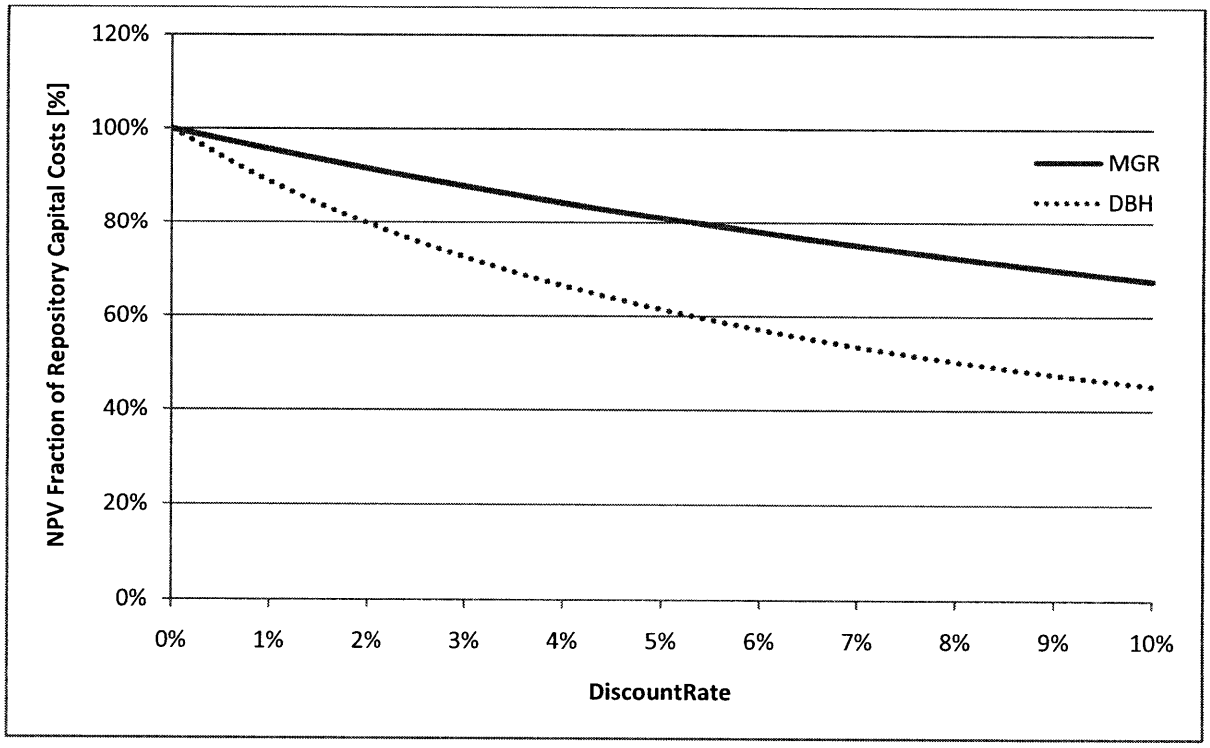


Figure 4.11. Relative Change in Net Present Value of Repository Construction Cost.

The government could look at modularity in a more macro level and realize that they can effectively increase the rate of spent fuel acceptance by a factor of four by opening up four regional borehole complexes. This strategy should greatly reduce any spending on shutdown fuel monitoring and in fact it succeeds in reducing the quantity of fuel stored in onsite dry storage casks and spent fuel pools of shutdown reactors as shown in the levels of stocks displayed in Figure 4.12. Notice how the 12,000 metric tonne per year total repository acceptance rate dramatically halts and reverses the accumulation of spent fuel in onsite dry storage and quickly removes the waste in spent fuel pools. In fact, this aggressive waste collection strategy disposes of 99% of the spent nuclear fuel by 2040 and officially finishes safely storing the waste by 2047. Policy makers must decide if the reduction in onsite spent fuel management offsets the added repository expense and possible repository heat load limits that could be reached with such a high spent fuel acceptance rate.

Figure 4.13 displays the annual spending for this aggressive strategy and reveals it calls for heavy expenditures surpassing $2.5B per year. Table 4.5 summarizes the total system life cycle cost of the aggressive strategy that totals $53.4B or $573 per kilogram. Notice, the shutdown spent fuel pool costs remain constant because the simple unloading algorithm forces shutdown spent fuel pools to empty as soon as their fuel cools for the required 5 years. The strategy does reduce the cost of onsite dry storage by more than half to just $2.3B from the $5.6B estimated in the base case and in the deep borehole base case and because the strategy places the repositories across the country, the regional borehole repository plan also reduces the transportation by more than half to just $0.88B from $2.0B. That having been said, the huge capital expense associated with the licensing of surface and subsurface facilities as well as the construction of independent surface facilities for each disposal complex greatly increases the total repository costs to $25.1B from just $11.7B in the single deep borehole base case.

Of course, this four repository base case represents a fairly unlikely scenario because the government would most likely stagger the construction of the repositories and take advantage of a learning curve in their construction and licensing. By opening one borehole repository every 10 years in order of most needed to least needed, determined by regional spent fuel accumulation, and assuming an incremental construction cost savings of 10% on each new repository, the total system life cycle cost actually increases slightly to $55.1B because the repository delays add significant onsite storage costs. This more realistic approach of staggering the opening of the four repositories is summarized in Table 4.6.

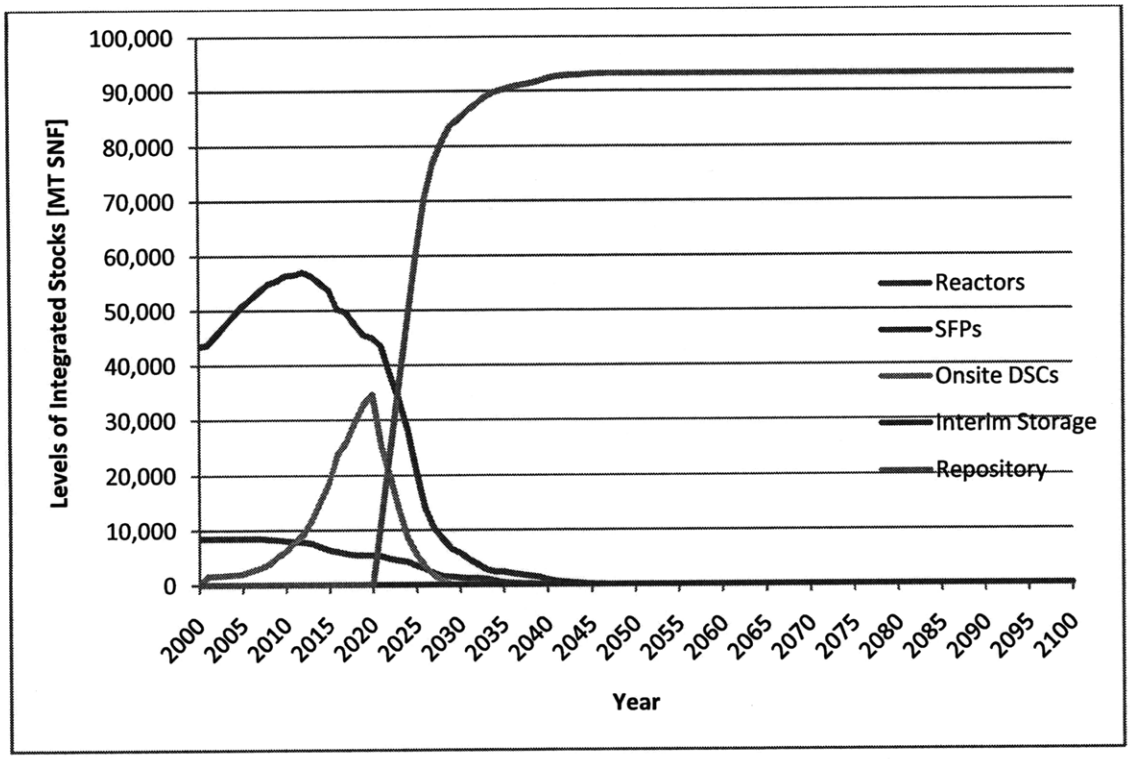


Figure 4.12. Four Repository Strategy Levels of Integrated Stocks of Nuclear Fuel.

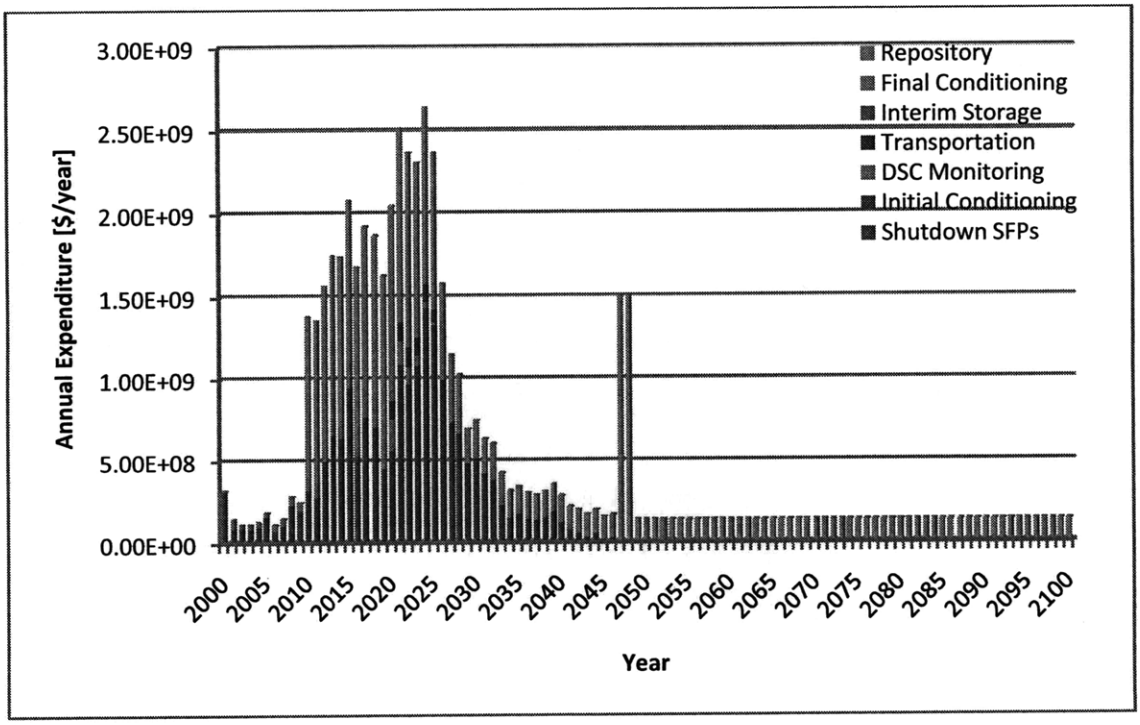


Figure 4.13. Estimated Annual Cash Flow for the Four Deep Borehole Strategy.

Table 4.5. Four Deep Borehole Repositories Total System Life Cycle Cost Summary.

Cost Category	Total	System Averaged Cost	Percentage
Shutdown SFPs	$5.0B	$54/kgIHM	9.4%
Initial Conditioning	$14.0B	$150/kgIHM	26.2%
Onsite Dry Storage	$2.3B	$24/kgIHM	4.3%
Transportation	$0.88B	$9/kgIHM	1.7%
Interim Storage	$0.0B	$0/kgIHM	0.0%
Final Conditioning	$6.2B	$66/kgIHM	11.5%
Repository	$25.1B	$269/kgIHM	47.0%
Total	$53.4B	$573/kgIHM	100.0%

Table 4.6. Four Staggered Deep Borehole Repositories Total System Life Cycle Cost Summary.

Cost Category	Total	System Averaged Cost	Percentage
Shutdown SFPs	$5.0B	$54/kgIHM	9.1%
Initial Conditioning	$14.0B	$150/kgIHM	25.4%
Onsite Dry Storage	$7.6B	$82/kgIHM	13.8%
Transportation	$0.88B	$9/kgIHM	1.6%
Interim Storage	$0.0B	$0/kgIHM	0.0%
Final Conditioning	$6.2B	$66/kgIHM	11.2%
Repository	$21.5B	$231/kgIHM	39.0%
Total	$55.1B	$591/kgIHM	100.0%

While the increased repository acceptance rates of spent fuel greatly improved the flow of waste through the system, the multi-repository approach added too great an expense compared with the single borehole complex for which the onsite spent fuel storage discounts could not outweigh the added construction costs. Section 4.5 will return to this idea of increasing the transfer rate of spent fuel from the reactor sites using interim storage facilities to see if a more cost effective methodology makes the increased acceptance plan work.

After exploring the one and four borehole repository strategies, one must again reinvestigate the effect of delaying the repository opening date. For this we simply modify the borehole base case runs to begin receiving fuel from 2020 to 2060 in 10 year increments to compare with the results from section 4.2 by

indexing the total system life cycle costs to the base case with a Yucca Mountain type of repository opening in 2020. As one might expect, Figure 4.14 shows the total waste management expense increases with increasing delay in the repository opening with a few subtleties worth noting. As one might expect, the total waste management expense for the deep borehole strategy has a much lower magnitude, but increases with the same slope as the mined geologic repository strategy. This confirms that the two strategies provide spent fuel transfer to the repositories at the same rate. Also interesting to note, the slope of the regional deep borehole strategy is only $225M per year compared with the base case, which is over $330M per year. Indeed, this occurs because the much faster acceptance rate of four regional repositories mitigates the consequences of delays better than a single national repository could. In other words, the higher spent fuel acquisition rate of regional disposal sites leads to lower onsite dry storage costs for a given repository starting date.

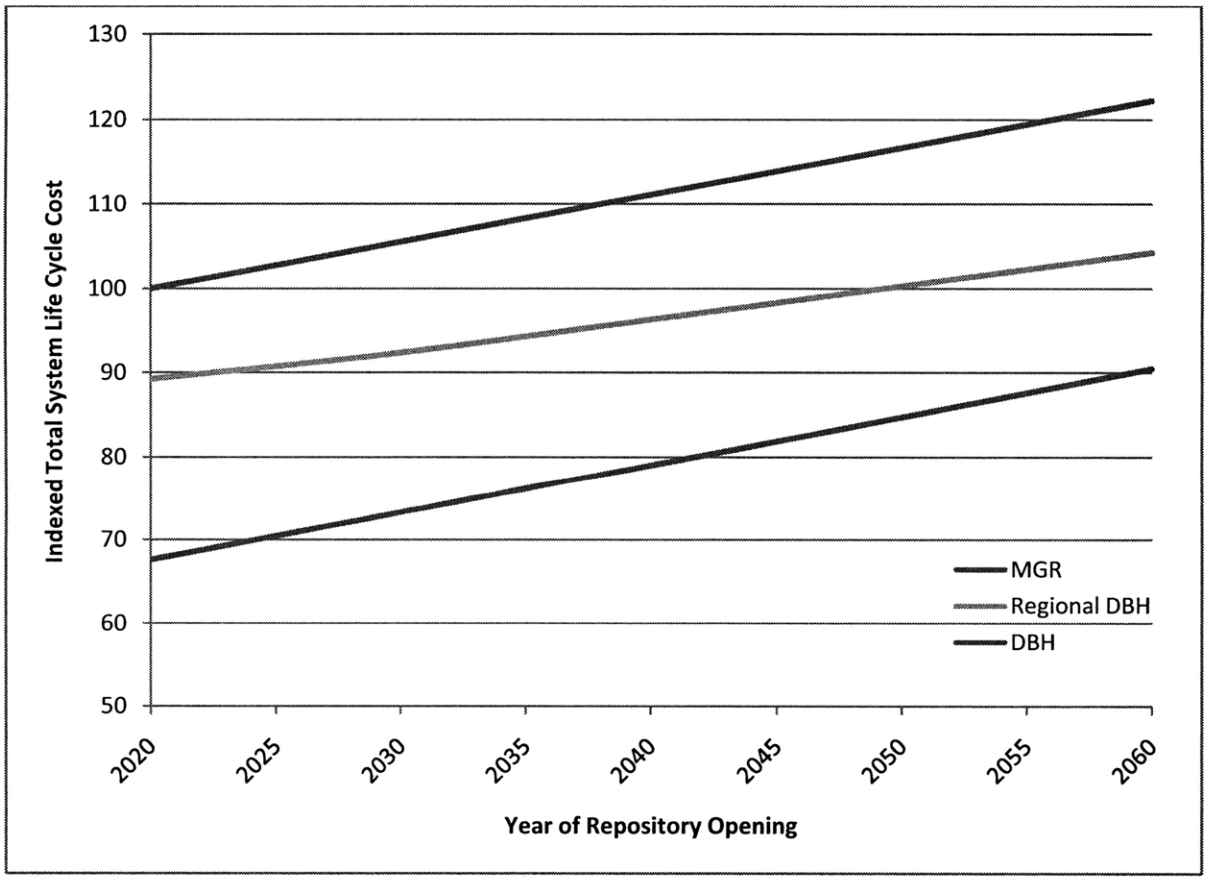


Figure 4.14. Total System Life Cycle Cost Index for Delays in Deep Borehole Strategies.

4.4 Unloading Algorithms

Thus far, SNuFManager relied exclusively on the simple unloading algorithm* that searched for the reactors shutting down soonest and offloaded their spent fuel first. This strategy has elegance in its simplicity, but one must determine the effects of more complicated algorithms intended to improve the economic efficiency of the total waste management system. Of course, the system's complexity makes finding an analytic solution to the total system life cycle cost unrealistic and so a thorough analysis should consider many unloading algorithms and deterministically identify the optimized strategy. This thesis proposes only one alternative methodology for illustrative purposes and leaves further optimization for future investigations when the preferred waste management strategy is more clearly defined.

The alternative algorithm, referred to as the Taylor algorithm and denoted with a "(T)" in the figures, begins in the same way as the simple unloading scheme by updating the reactor loading and discharging rates specified in the input for the given power level of each reactor, followed by the spent fuel pool cooling rates and required spent fuel pool discharge rates determined by analyzing the age of the fuel in the pools and each pool's remaining capacity. Like the simple flow calculator, the Taylor algorithm checks the required spent fuel pool discharge rate after each set of flow assignments to determine the necessity of future steps and avoid unnecessary computation.

The next stage† loads the available repository capacity, first with waste cool enough for handling from shutdown reactor spent fuel pools in the order of the oldest to most recent shutdown dates, followed by dry storage casks located at shutdown reactors, fuel in online spent fuel pools, fuel in dry storage casks at online reactors, and finally from available interim storage discharge capacity, where the status of shutdown only applies to reactor sites in which no reactors onsite continue to operate. The next stage follows the same pattern in loading the available interim storage capacity and the final stage sends the remaining required spent fuel pool discharge into dry storage casks onsite.

---

* SNuFManager calls the simple algorithm, "calculateSimpleFlows" and the Taylor algorithm, "calculateFlows." A future revision of the code should begin by allowing the unloading strategy to be defined in the input file, but because the current application of the code hardly changes this setting it is hardwired and the preferred strategy must be selected by commenting out the calling of the undesired subroutine.

† Originally, the Taylor algorithm started with another stage that unloaded spent fuel from reactors scheduled to shut down soon in an amount inversely proportional to the remaining life of the reactors, but because this added complexity and had only a minor effect on the total system life cycle costs, it was commented out and left in the code for possible changes in the future.

Figure 4.15 shows the effects of the complicated unloading patterns that lead to a moderately worse total system life cycle cost for repository opening dates before 2040 and a nominally better life cycle cost for opening dates after 2040 for both the Yucca Mountain base case and the deep borehole base case. The improvements for a repository opening date after 2040 likely appear due to the larger accumulation of spent fuel at many sites with a wide range of capacity constraints that lend themselves to improvements through sophisticated waste management patterns, while for an opening date before 2040, the reduced constraints make the complex Taylor algorithm more of a burden than a benefit. The Taylor algorithm led to no noticeable changes in the four repository deep borehole strategy. These results suggest that advanced unloading schemes may reduce waste management costs by approximately 1%, especially if the repository opens after 2040, but this pales in comparison to the savings realized by switching from a highly capital intensive waste disposal strategy to a modular deep borehole approach.

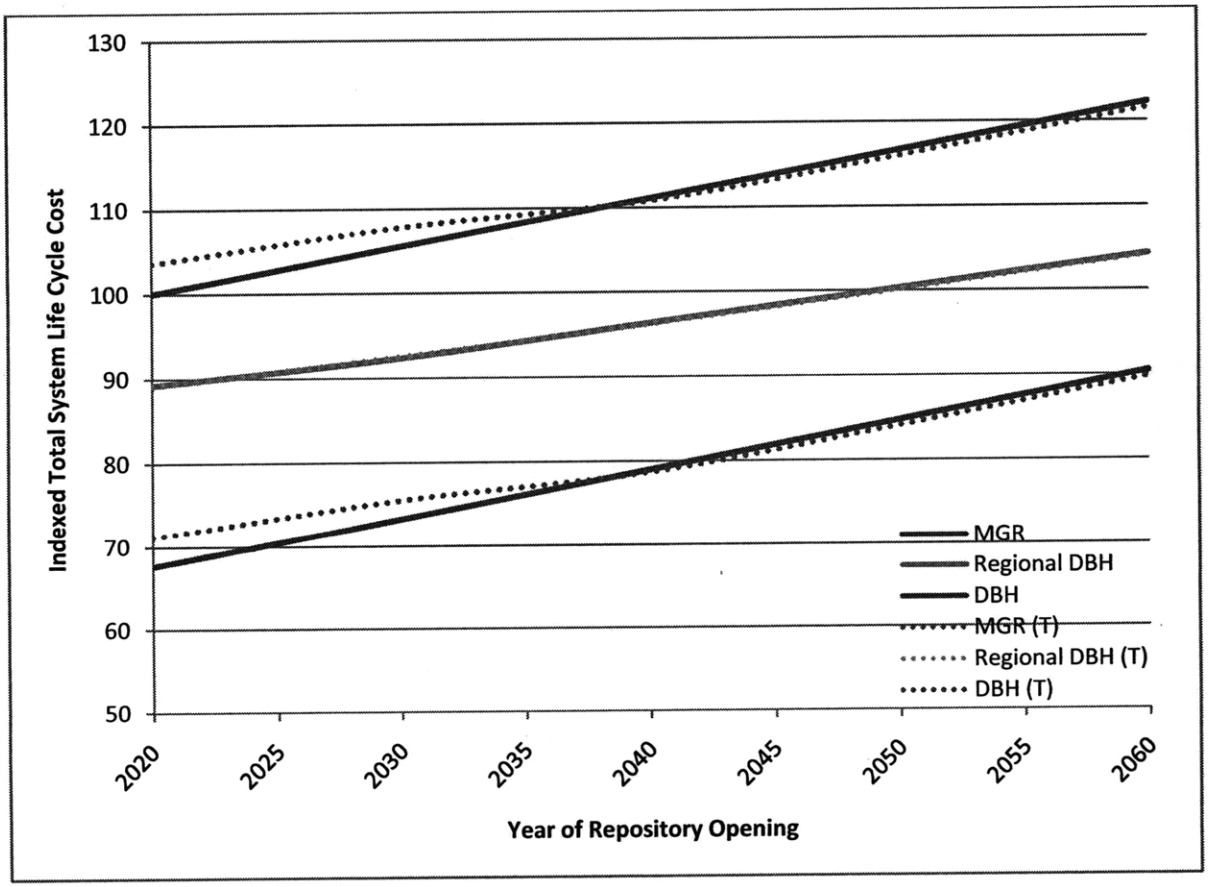


Figure 4.15. Total System Life Cycle Cost Index for Delays Using Taylor Algorithm.

4.5 Interim Storage Strategies

The use of interim storage provides the ability to increase the transfer rate of spent fuel from reactors cheaply avoiding the costs associated with onsite waste storage in spent fuel pools and dry storage casks while providing many decades of retrievability. This retrievability provides the ultimate flexibility in case the preferred disposal method changes or in case the nation decides to reprocess spent fuel to hedge the risk of uranium shortages or to reduce the concentration of long-lived isotopes in the waste that dominates the radioactive burden hundreds of thousands of years after reactor discharge.

This thesis makes the opening in 2020 of the Private Fuel Facility's interim storage design, also known as the Skull Valley concept and discussed in section 3.5, the default option for policy analysis. As a reminder, this facility resides in Utah, region IV, and has a capacity of 40,000 metric tonnes of spent fuel and a nominal loading rate of 4,000 metric tonnes per year. Table 4.7 outlines the updated parameters to the base case in order to analyze the effect of adding the centralized interim storage.

Table 4.7. Base Case with Interim Storage Input Summary.

Time Parameters	Value	Spent Fuel Pool Parameters	Value
Simulation Period	1998 - 2100	Minimum Cooling	5 years
Interim Storage Parameters	Value	Repository Parameters	Value
Number of Sites (Loc.)	1 (IV)	Number of Sites (Loc.)	1 (IV)
Capacities	40,000 MT	Number of Expansions	1
Opening Dates	2020	Capacities	93,200 MT
Acceptance Rates	4,000 MT/year	Expansion Dates	2020
Initial Capital	$110M	Acceptance Rates	3,000 MT/year
O&M Base	$8.67M/year	Total Capital	$3.48B
O&M Loading	$3.79M/year	O&M Base	$44.5M/year
O&M Monitoring	$0.26/kgIHM/year	Closure	$0.68B
Other Economic Parameters	Value	Other Economic Parameters Continued	Value
Initial Conditioning	$150/kgIHM	Shutdown SFP	$11.3M/year
Repos. Cond. & Pack.	$274/kgIHM	Initial Onsite DS	$6.0M
Inter-region Transport	$7 - $14/kg	Online DSC Monitoring	$0.64M/year
Intra-region Transport	$20 - $26/kg	Offline DSC Monitoring	$5.1M/year

Initiating both an interim storage facility and a Yucca Mountain repository in 2020 seems like an inefficient strategy because, as shown in Figure 4.16, spent fuel sent to interim storage only stays there about 20 years before moving on to the repository. Even so, the interim storage facility immediately reverses the use of onsite dry storage and, as long as the cost savings from reduced onsite storage outweighs the added cost of interim storage and more transportation, the strategy of adding this centralized temporary surface monitoring option makes economic sense.

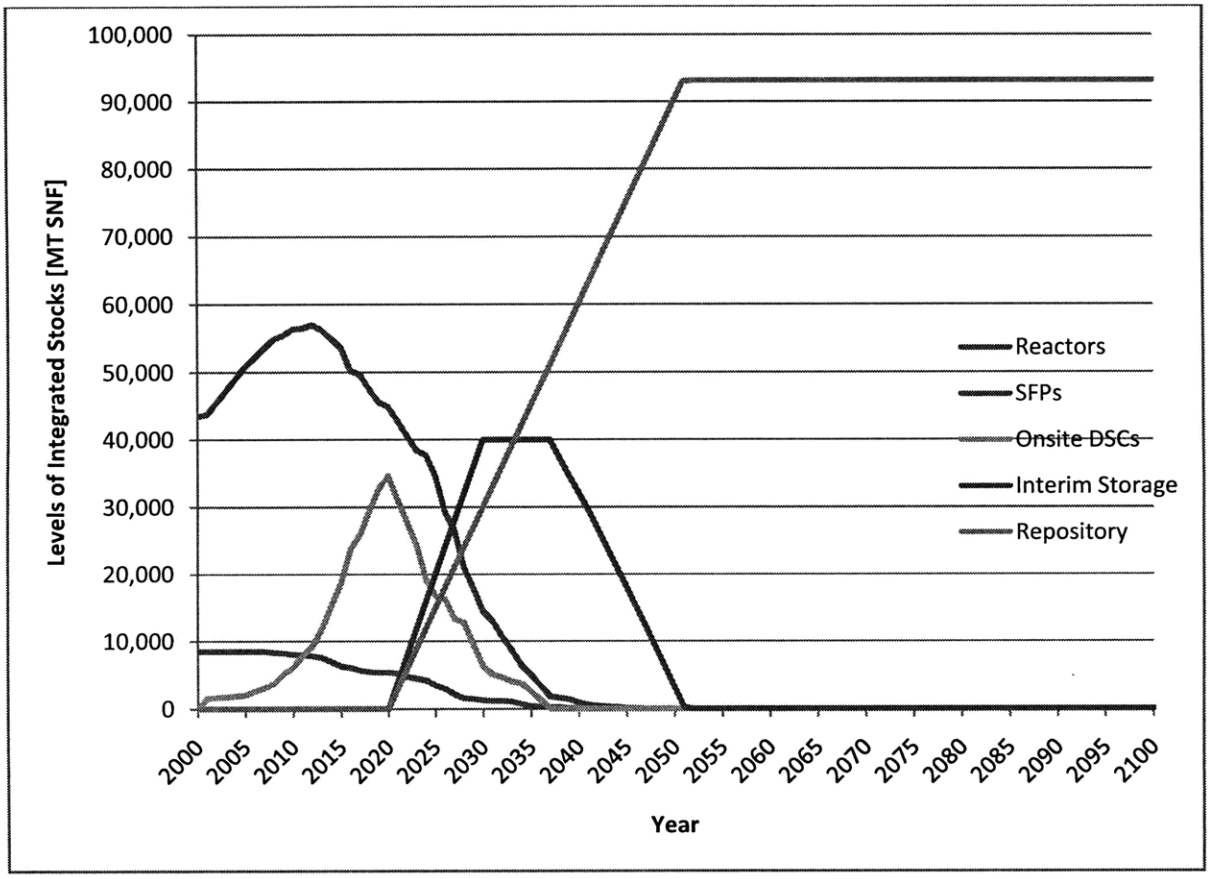


Figure 4.16. Base Case Levels of Integrated Stocks of Nuclear Fuel.

In fact, the expected annual cash flows for this default interim storage strategy, shown in Figure 4.17, appear quite similar to those for the original base case, except the figure clearly illustrates the benefits of reduced onsite dry storage after 2020. Upon close inspection, one sees the increased transportation costs, but since the transportation accounts for such a small portion of the total system life cycle cost, increasing that portion of the spending has little effect on the overall budget.

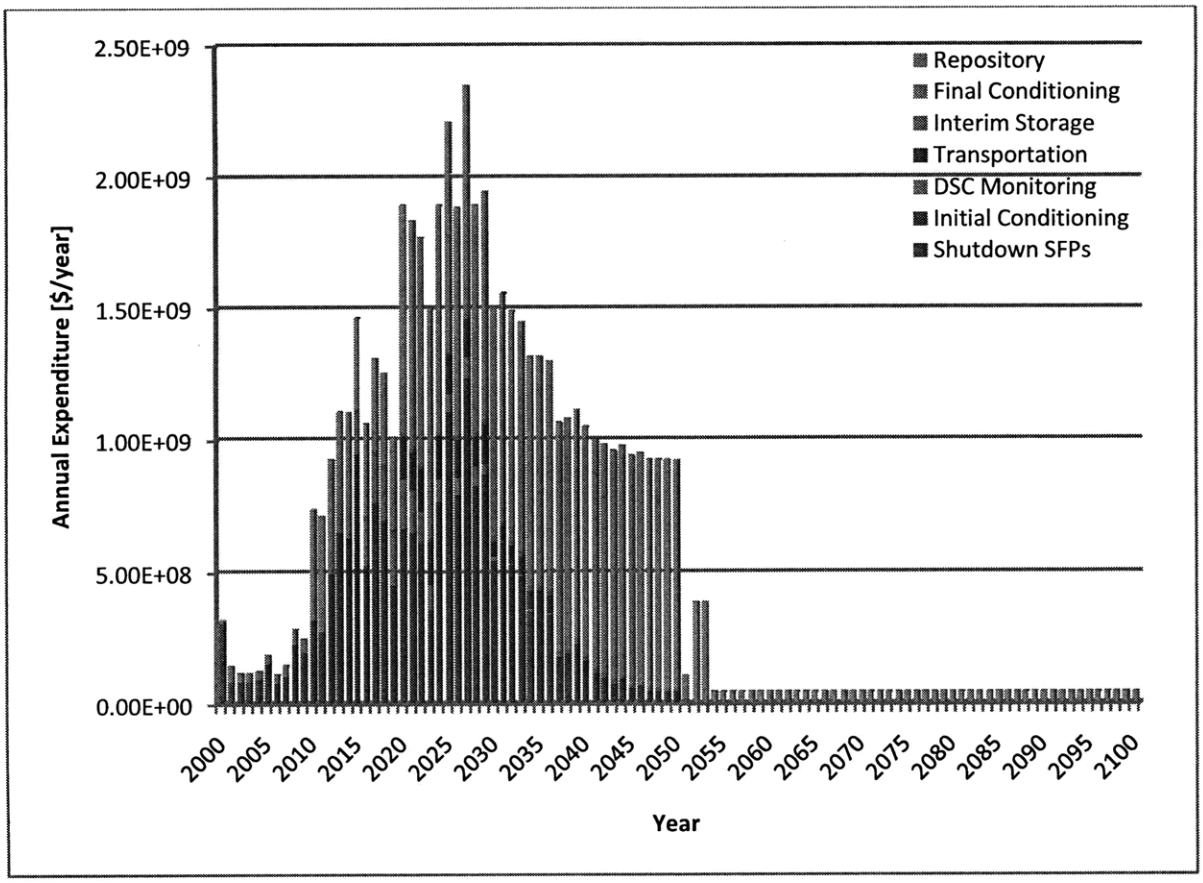


Figure 4.17. Estimated Annual Cash Flow for the Base Case with Default Interim Storage.

Upon closer inspection of the integrated costs, Table 4.8 shows the onsite dry storage totals just $2.8B, half of the onsite dry storage expenses estimated for the original base case. On the other hand, the transportation cost increased $600M and the interim storage added $680M, while all the other costs remained the same. Clearly, even though the policy uses interim storage inefficiently, it still benefits the bottom line to do so. One might imagine these benefits should grow substantially as the opening date of the geologic repository increases because the use of interim storage costs so little in the grand scheme of things. To test how much the use of interim storage benefits the waste management strategy given delays in the opening of a repository, we should reevaluate each of the repository strategies adding in the default interim storage option.

Table 4.8. Base Case with Default Interim Storage Total System Life Cycle Cost Summary.

Cost Category	Total	System Averaged Cost	Percentage
Shutdown SFPs	$5.0B	$54/kgIHM	8.6%
Initial Conditioning	$14.0B	$150/kgIHM	24.0%
Onsite Dry Storage	$2.8B	$30/kgIHM	4.8%
Transportation	$2.6B	$28/kgIHM	4.4%
Interim Storage	$0.68B	$7/kgIHM*	1.2%
Final Conditioning	$25.5B	$274/kgIHM	43.8%
Repository	$7.8B	$83/kgIHM	13.3%
Total	$58.3B	$626/kgIHM	100.0%

As expected, for greater delays in the opening of a geologic repository, the use of interim storage offers significant relief to the waste management burden at reactor sites and translates to very real life cycle cost savings. Figures 4.18 and 4.19 show the expected annual expenditures when the default interim storage facility opens in 2020 with a Yucca Mountain type of repository opening in 2040 and 2060, respectively. Again, delaying the opening of the repository helps to flatten the cost profile by avoiding further large expenditures on top of the high initial conditioning costs that occur mostly from 2010 to 2040.

The expected savings from opening a 40,000 metric tonne interim storage facility in 2020 over no interim storage grows to nearly $5B when the repository opens in 2040 and over $8B if the repository opens as late as 2060. While no one currently recommends the delay of a repository, the government's history of setbacks in dealing with the waste management solution makes consideration of further delays a prudent action. That having been said, one might actually recommend the delay of a repository if the use of interim storage provides a cheap storage alternative in the short term and allows cheaper, safer, more politically acceptable, and/or more sustainable waste solutions in the future. For instance, if the industry expects the significant use of reprocessing in fifty years, pushing the development of a Yucca Mountain repository now seems a waste of money when a centralized interim storage facility could function just as well over periods as long as a hundred years.

* As a reminder, just like the shutdown spent fuel pool and onsite dry storage levelized cost averages the expenditures over the total accumulation of spent fuel, the interim storage cost does the same even though only a fraction of the fuel goes through interim storage.

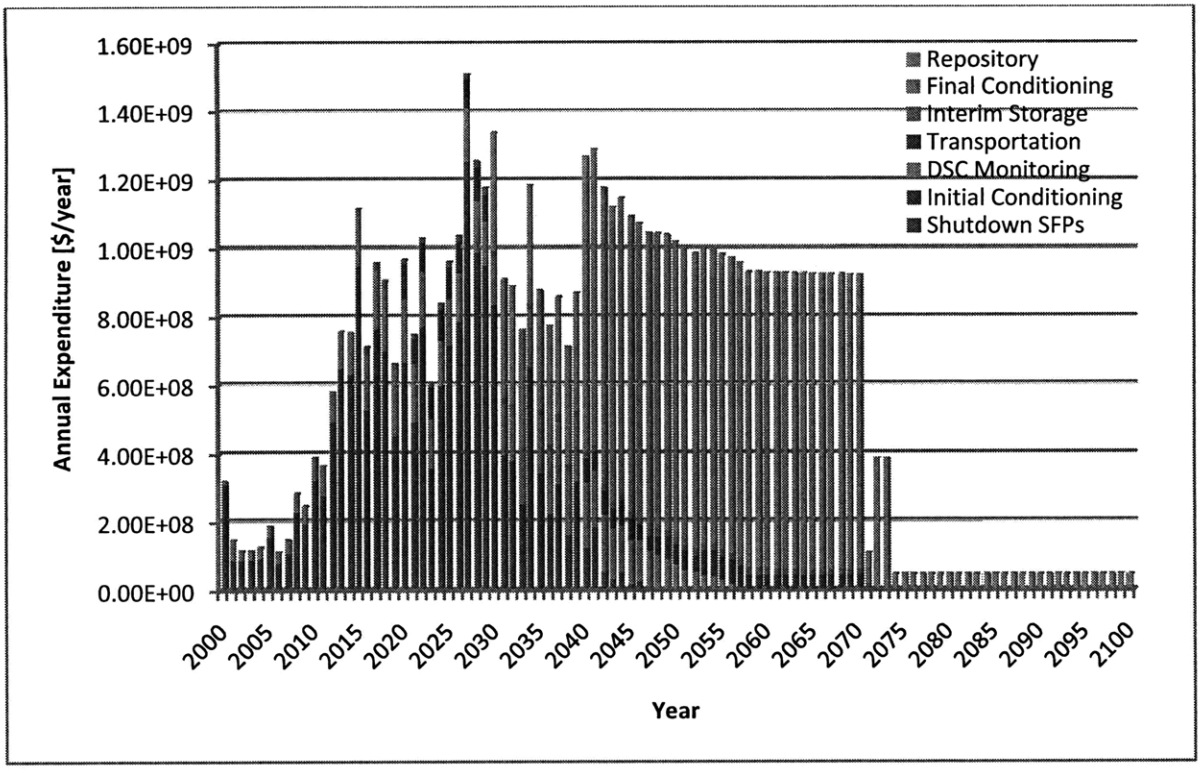


Figure 4.18. Estimated Annual Cash Flow for a 2040 Repository and 2020 Interim Storage.

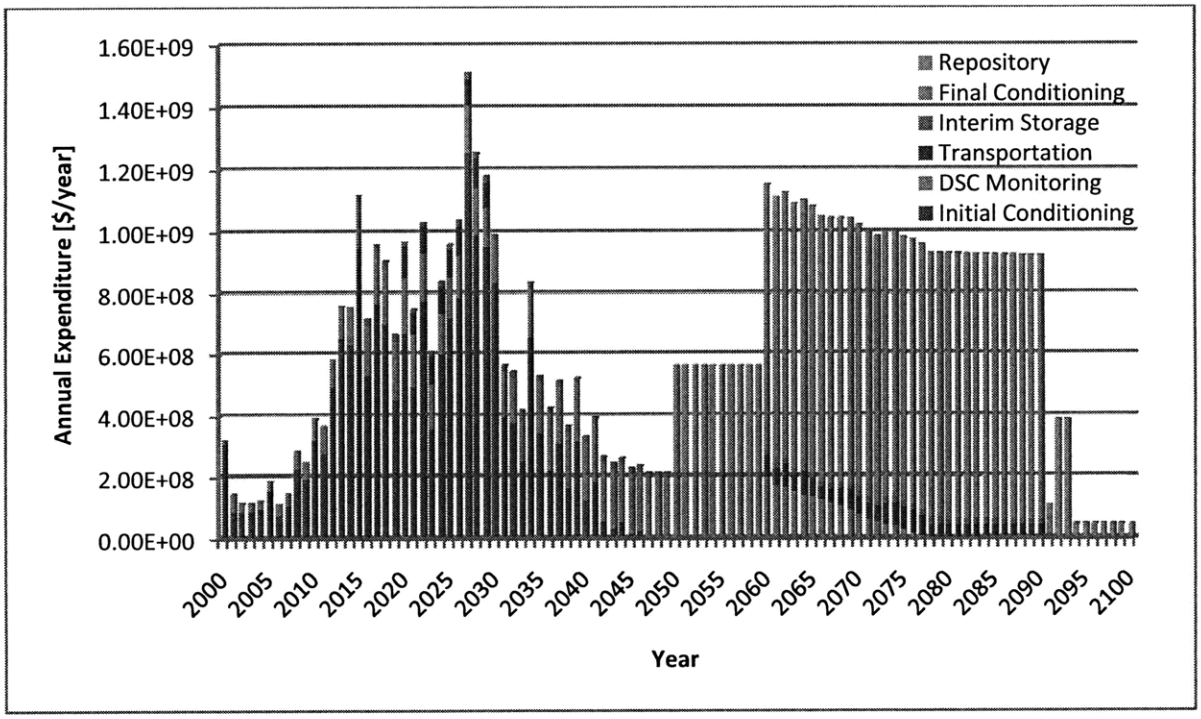


Figure 4.19. Estimated Annual Cash Flow for a 2060 Repository and 2020 Interim Storage.

As shown in Figure 4.20, the total system lifecycle cost decreases with increasing interim storage capacity. Because the slope of the lines also decreases for increasing interim storage capacity, adding interim storage effectively moderates the added costs associated with delaying the opening of a repository. For a repository opening date of 2020, the benefits of interim storage saturate by the time the capacity increases to 40,000 metric tonnes, but as the repository opens later and later, the increased interim storage space pays dividends.

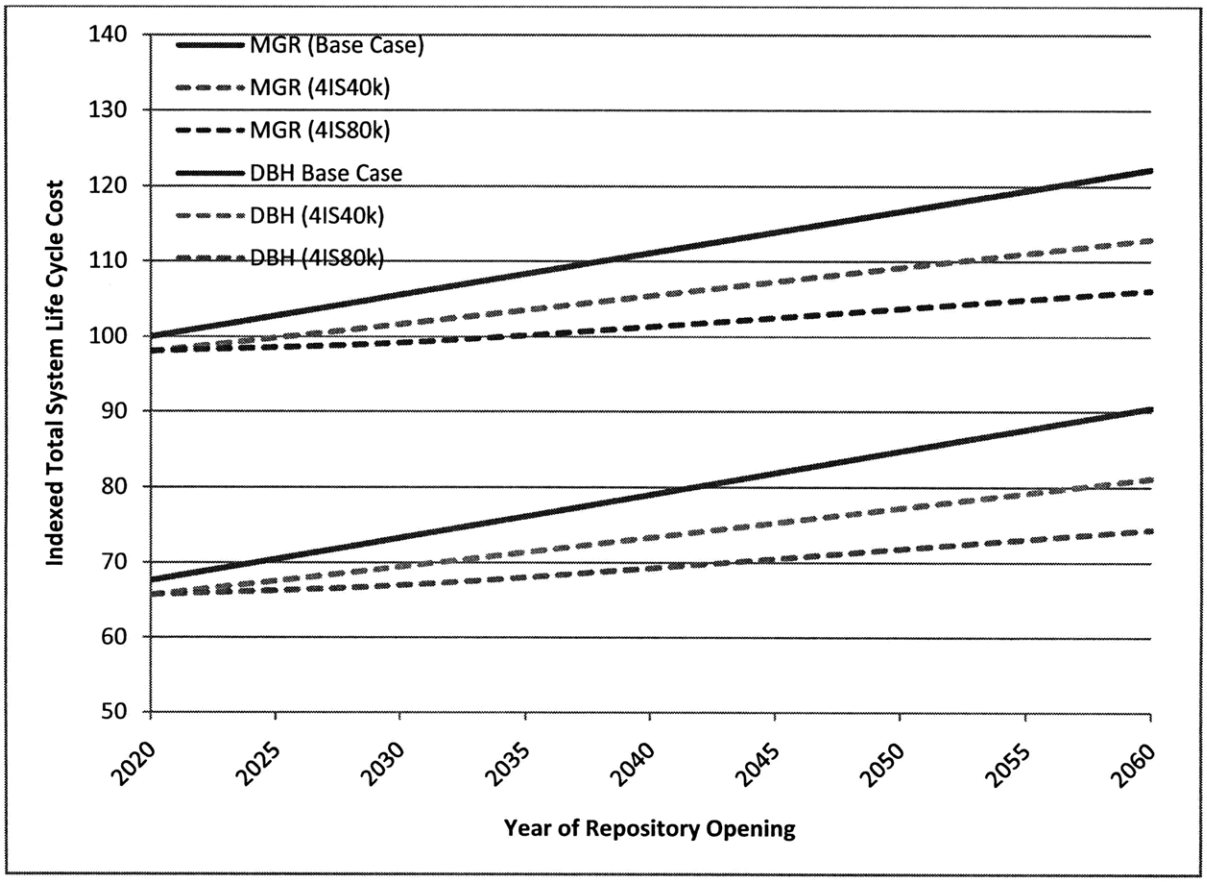


Figure 4.20. Total System Life Cycle Cost Index for Various National Interim Storage Capacities.

Again, we test the benefits of increasing the transfer rate of spent fuel from reactor sites by simulating four regional interim storage facilities each with a maximum loading rate of 4,000 metric tonnes per year. The simulations, shown in Figure 4.21, only test regional facilities of 10,000 and 20,000 metric tonne capacities, because extending beyond 20,000 metric tonnes exceeds the source of spent fuel generated in two of the regions and therefore only makes sense if considering additional reactor license renewals.

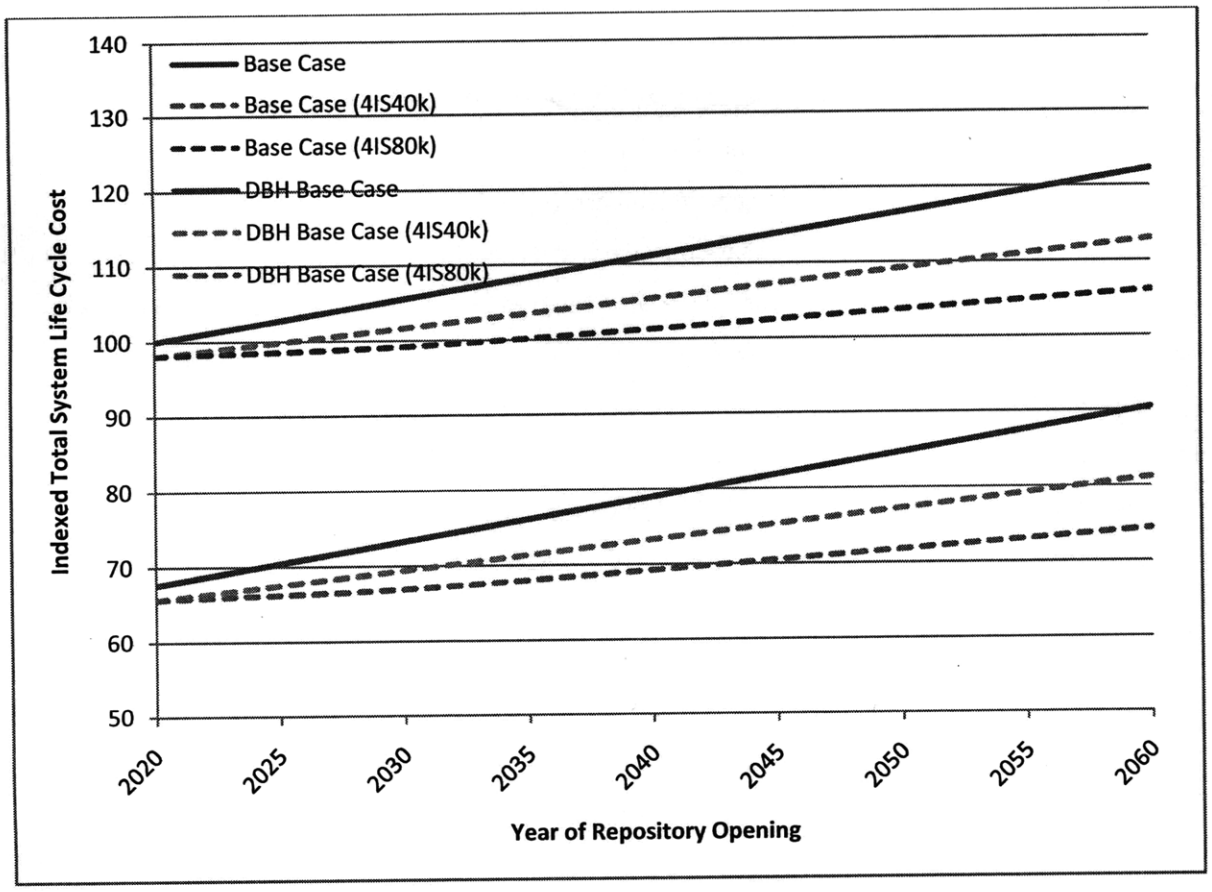


Figure 4.21. Total System Life Cycle Cost Index for Various Regional Interim Storage Capacities.

Figure 4.21 shows no benefits of an increased acceptance rate from four regional facilities, but this lesson only applies if the interim storage facilities open in 2020. One should expect the value of four interim storage sites, and the same goes for four repositories, increases significantly relative to a single national facility for later repository start dates because at that point the accumulation of spent fuel in onsite dry storage and in shutdown spent fuel pools significantly increases the penalty for slow removal of the waste.

Summary

This chapter first illustrated the components of the total system life cycle cost for the base case scenario and showed that the initial conditioning, final conditioning, and repository itself make up the majority of the expenditures. The total waste management expense depends on the timing of the repository opening in a highly linear and correlated fashion such that a repository delay of 20 years could likely add more than 10% to the life cycle cost. Switching from a Yucca Mountain type of repository to a borehole technology cut the expected costs 33% by reducing the engineering requirements on the disposal packaging and dramatically discounting the final conditioning expenses. The net present value of the expenditures dedicated to the repository itself is lower for the mined geologic repository and a discount rate of less than 3%. When discounting above 3%, the modularity of the deep borehole costs lead to a lower net present value. A regional borehole strategy in which each complex receives spent fuel from its region's reactors provides a discount to the base case, but still costs much more than a national borehole strategy and therefore one should only pursue this approach if politically required by those who oppose the burden of the entire nation's waste going to a single state. In theory, the disparity sets the indifference price for the government such that the attractiveness of the mined geologic repository strategy opening in 2020 is equivalent to the attractiveness of a regional borehole strategy with a payment of roughly 10% of the total system life cycle cost to those who might oppose the plan. SNuFManager showed some sensitivity to the unloading algorithms and so a wise policy maker should look to advanced transfer priority schemes after settling on a disposal technology and interim storage use. Finally, the use of interim storage facilities greatly alleviates the increasing burdens on the reactor sites that must resort to expensive onsite storage only to end up suing the government to recompense their costs. The expected savings from opening a 40,000 metric tonne interim storage facility in 2020 over a strategy with no interim storage amounts to nearly $5B when the repository opens in 2040.

Chapter 4 References

4.1. **Posner, Michael.** Energy Department revises cost estimate for Yucca Mountain project. *GovernmentExecutive.com.* [Online] July 15, 2008. [Cited: July 25, 2008.] http://www.govexec.com/dailyfed/0708/071508cdpm2.htm.

4.2. **U.S. Department of Energy.** *Nuclear Fuel Data.* [Database] Washington, DC : s.n., 2002. RW-859.

4.3. **Keystone.** *Nuclear Power Joint Fact-Finding.* Keystone, CO : The Keystone Center, 2007.

4.4. **Office of Civlian Radioactive Waste Management.** *Nuclear Waste Fund Fee Adequacy: An Assessment.* Washington, DC : U.S. Department of Energy, 2001. DOE/RW-0534.

4.5. **Office of Civilian Radioactive Waste Management.** *Analysis of the Total System Life Cycle Cost of the Civilian Radioactive Waste Management Program.* Washington, DC : U.S. Department of Energy, 2001. DOE/RW-0533.

4.6. *Yucca Mountain Licensing Status.* **Sproat, Edward.** Cambridge, MA : Massachusetts Institute of Technology, 2008. Nuclear Science and Engineering Seminar Series.

Chapter 5:
Sensitivity Analysis

Introduction

This chapter is structured to give the reader insight into the range of uncertainty involved in the estimations along with a likely upper and lower bound on the total system life cycle costs. Clearly, there is a sound basis for comparisons of identical systems operating under slightly different conditions. That said, there are larger uncertainties when it comes to comparisons of completely different technologies, such as the life cycle cost of the base case versus a regional borehole strategy with regional interim storage facilities, and this chapter attempts to provide the reader with further intuition as to these unknowns. One should not confuse the uncertainty analysis provided with a rigorous propagation of uncertainties, but rather regard it as a reasonable estimation of the range of expected results given the poor degree of knowledge of many factors, including proprietary cost data. This chapter qualitatively discusses the sensitivity of the SNuFManager tool to plausible deviations in globally assumed variables , such as the assumed burnup levels and capacity factors, and quantitatively reviews the results under an optimistic scenario of reactor operating license extensions to the existing fleet. This thesis avoids hypothesizing about possible development of the so-called "nuclear renaissance," but one should note that timing of interim storage facility use fits particularly well when considering additional nuclear reactors because the ultimate disposal site should receive the current accumulation of spent fuel by the time the new reactors accumulate enough to need transportation offsite. Therefore, the interim storage facilities could easily unload the old assemblies and simultaneously load new spent fuel. In addition, the use of borehole disposal easily meets a flexible waste storage requirement as opposed to the more rigid Yucca Mountain approach.

5.1 Uncertainty Analysis

The uncertainty analysis presented gives the most rudimentary estimate of a likely range of plausible values of the life cycle costs given the poor transparency associated with many of the proprietary expenditures. A more realistic uncertainty analysis would require sophisticated methods, such as Monte Carlo analysis. Chapter 3 provided no significant discussion about the meaning of the range of values presented. In many cases, the high and low values represented the extremes found in the open literature, while in other cases the literature provided inadequate quantity or quality data and so these values represented some nominal deviation from the expected value. Other work presenting similar analyses assumed a simple triangular distribution between the low, mid, and high values. This may offer a reasonable balance of simplicity and complexity, but intuition suggests that the high and low values more likely reflect a logical confidence interval and that a standard distribution, such as the normal distribution, should describe the spread of the data. This analysis is outside the scope of the current thesis, and the uncertainty analysis presented here, as noted, is more rudamentary. Figure 5.1 shows the upper and lower bounds on the base cases.

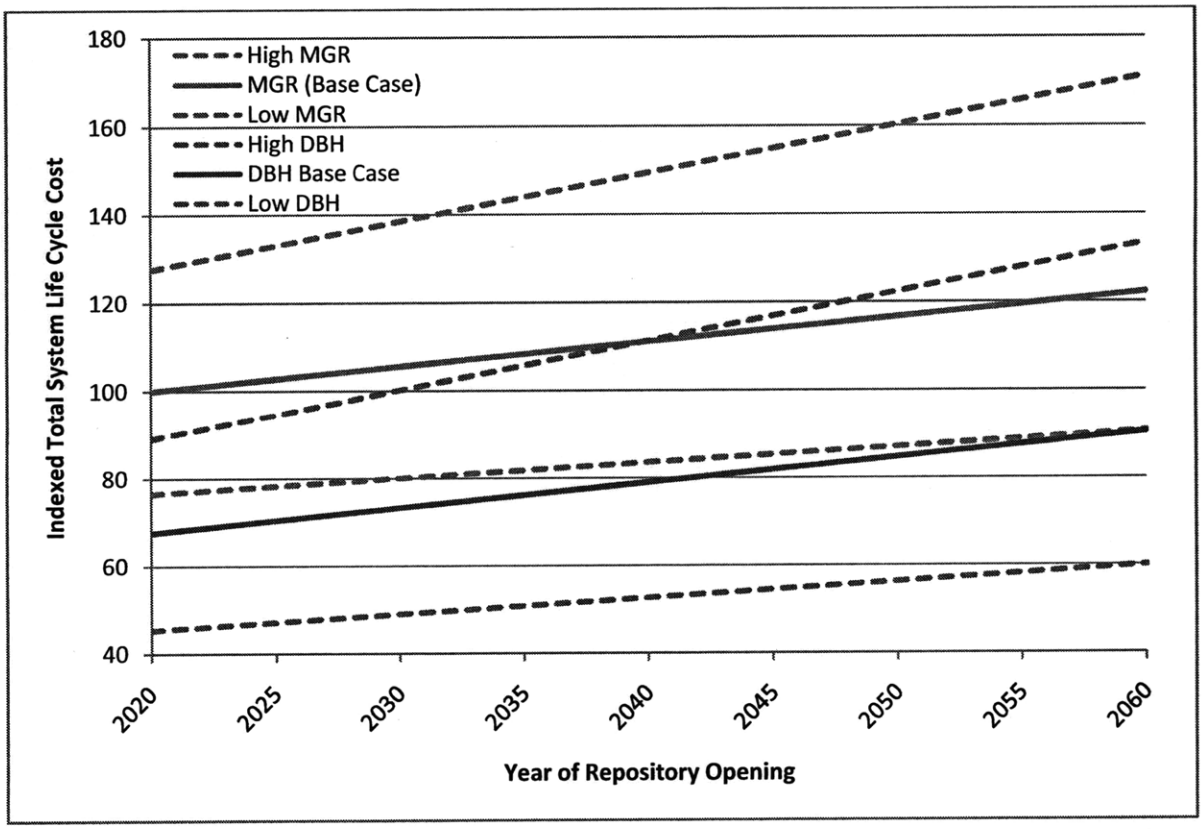


Figure 5.1. Uncertainty Bands of Base Case Indexed Total System Life Cycle Cost.

Figure 5.1 shows that it takes extremely high costs on the deep borehole case and extremely low costs on the Yucca Mountain repository type in order to change the ranking of economic efficiency between the base case disposal strategies. Few plausible scenarios exist in which the deep borehole strategy incurs high costs while the mined geologic repository incurs only low costs. In fact, more realistic conditions should lead to higher or lower costs for both disposal concepts. Table 5.1 reviews the upper and lower estimates used in the SNuFManager evaluations based on the economic module summaries presented in the third chapter of this thesis. For details on the rationale of each variable's range of expected values, refer to the relevant section in Chapter 3. This section presents the total system life cycle results using the extreme cases summarized below.

Table 5.1. Low and High Uncertainty Economic Input Values.

Parameter	Low Value	High Value
Shutdown Spent Fuel Pool Operation	$5.3M/year	$12.6M/year
Initial Conditioning and Packaging	$91/kgIHM	$209/kgIHM
Initial Onsite Dry Storage Capital	$4.2M	$8.8M
Online Dry Storage Cask Monitoring	$0.50M/year	$0.79M/year
Offline Dry Storage Cask Monitoring	$3.4M/year	$9.5M/year
Inter-region Rail Transportation	$4 - $11/kgIHM	$7 - $16/kgIHM
Intra-region Rail Transportation	$15 - $21/kgIHM	$22 - $30/kgIHM
Interim Storage Initial Capital	$99M	$121M
Interim Storage Base Operation & Maintenance	$7.80M/year	$9.54M/year
Interim Storage Loading O&M	$3.41M/year	$4.17M/year
Interim Storage Monitoring O&M	$0.234/kgIHM/year	$0.286/kgIHM/year
Final Conditioning and Packaging (Yucca)	$246/kgIHM	$302/kgIHM
Repository Total Capital (Yucca)	$2.85B	$6.11B
Repository Base O&M (Yucca)	$40.1M/year	$49.0M/year
Repository Closure (Yucca)	$0.61B	$0.75B
Final Conditioning and Packaging (DBH)	$59/kgIHM	$73/kgIHM
Repository Total Capital (DBH)	$2.50B	$5.62B
Repository Base O&M (DBH)	$31.3M/year	$38.3M/year
Repository Closure (DBH)	$0.41B	$0.70B

Tables 5.2 and 5.3 summarize the uncertainty among the individual components of the total system life cycle cost for both the mined geologic repository and deep borehole disposal strategies.

Table 5.2. Uncertainty in Life Cycle Cost of Mined Geologic Repository Opening in 2020.

Cost Category	Low	Nominal	High
Shutdown SFPs	$2.4B	$5.0B	$5.6B
Initial Conditioning	$8.5B	$14.0B	$19.5B
Onsite Dry Storage	$3.8B	$5.6B	$10.1B
Transportation	$1.6B	$2.0B	$2.3B
Interim Storage	$0.0B	$0.0B	$0.0B
Final Conditioning	$22.9B	$25.5B	$28.1B
Repository	$6.7B	$7.8B	$10.8B
Total	$45.8B	$59.9B	$76.4B

Table 5.3. Uncertainty in Life Cycle Cost of Deep Borehole Repository Opening in 2020.

Cost Category	Low	Nominal	High
Shutdown SFPs	$2.4B	$5.0B	$5.6B
Initial Conditioning	$8.5B	$14.0B	$19.5B
Onsite Dry Storage	$3.8B	$5.6B	$10.1B
Transportation	$1.6B	$2.0B	$2.3B
Interim Storage	$0.0B	$0.0B	$0.0B
Final Conditioning	$5.5B	$6.2B	$6.8B
Repository	$5.4B	$7.8B	$9.1B
Total	$27.1B	$40.5B	$53.3B

5.2 License Extensions

From a waste management perspective, extending the reactor operating licenses has its pros and cons. On one hand, the extensions mean twenty additional years during which spent fuel pools and onsite dry storage will remain relatively cheap before reactors shutdown, while on the other hand, the longer reactor lifetimes mean significant increases in the total spent fuel accumulation. Figure 5.2 illustrates this tradeoff and shows the significant reduction of onsite dry storage use compared to the substantial increase in total spent fuel generation, where the dashed lines represent the base case with twenty year license extensions awarded to all reactors scheduled to shut down in 2010 or later. Since spent fuel conditioning and packaging dominates the total system life cycle cost for waste management, one expects the effects of increasing the production of nuclear waste will outweigh the benefits of reducing onsite dry storage use.

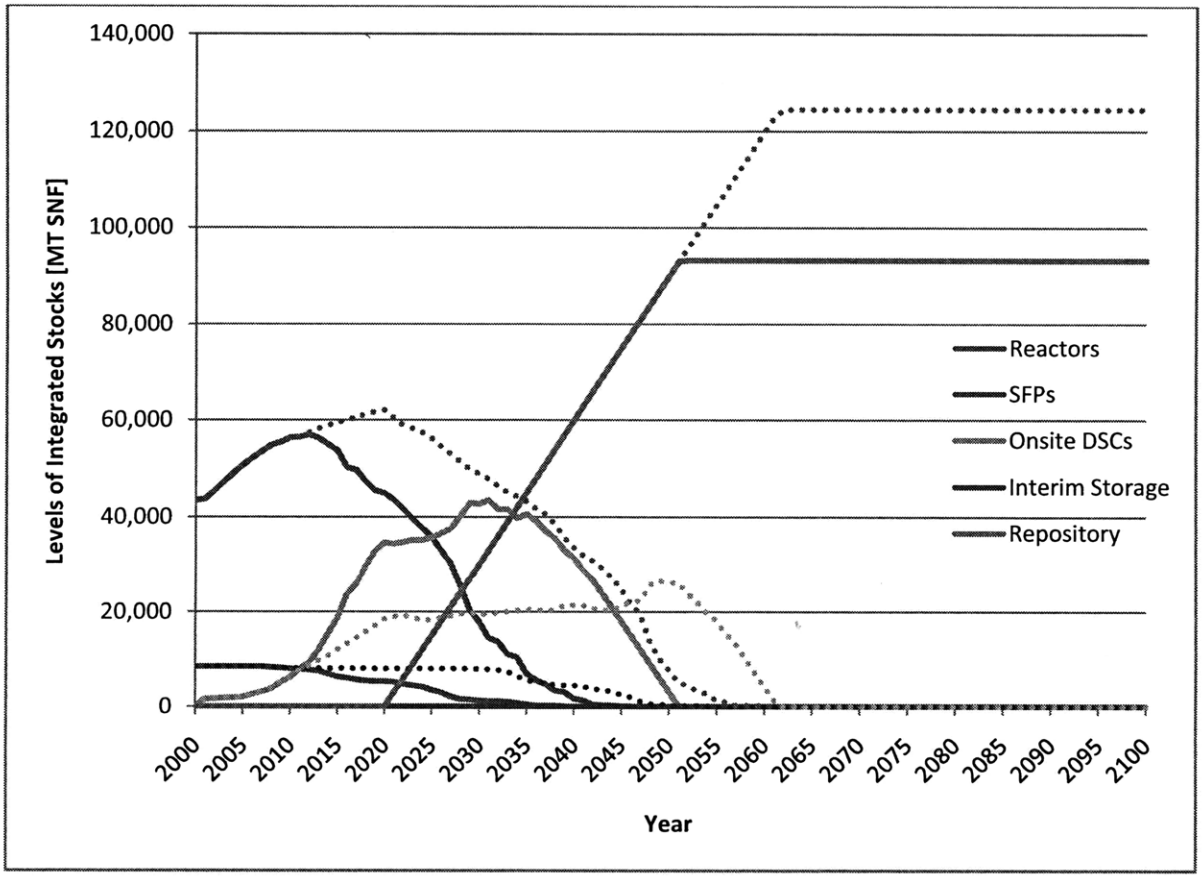


Figure 5.2. Base Case Levels of Integrated Stocks of Nuclear Fuel with License Extensions.

As expected, the roughly 33% more spent nuclear fuel in this scenario increases the waste management total system life cycle cost, but by less than 20% for disposal in a mined geologic repository and by less than 15% for a deep borehole disposal concept, as shown in Figure 5.3. In evaluating the license extensions, the summarized mountain disposal life cycle cost in Table 5.4 reflects a highly optimistic assumption that the Yucca Mountain subsurface construction costs remain unchanged through the massive expansion. The higher repository line item considers the added loading costs, but assumes no benefit from economies of scale for the surface operations. In fairness, the deep borehole cost in Table 5.5 also reflects no economies-of-scale benefit, but includes the additional costs of more boreholes.

Table 5.4. Base Case Total System Life Cycle Cost Summary with License Extensions.

Cost Category	Total	Levelized Cost	Percentage
Shutdown SFPs	$5.0B	$40/kgIHM	7.1%
Initial Conditioning	$18.7B	$150/kgIHM	26.2%
Onsite Dry Storage	$2.9B	$24/kgIHM	4.1%
Transportation	$2.7B	$21/kgIHM	3.7%
Interim Storage	$0.0B	$0/kgIHM	0.0%
Final Conditioning	$34.1B	$274/kgIHM	47.9%
Repository	$7.8B	$62/kgIHM	10.9%
Total	$71.1B	$572/kgIHM	100.0%

Table 5.5. Deep Borehole Total System Life Cycle Cost Summary with License Extensions.

Cost Category	Total	Levelized Cost	Percentage
Shutdown SFPs	$5.0B	$40/kgIHM	10.9%
Initial Conditioning	$18.7B	$150/kgIHM	40.7%
Onsite Dry Storage	$2.9B	$24/kgIHM	6.4%
Transportation	$2.7B	$21/kgIHM	5.8%
Interim Storage	$0.0B	$0/kgIHM	0.0%
Final Conditioning	$8.2B	$66/kgIHM	17.9%
Repository	$8.4B	$68/kgIHM	18.3%
Total	$45.9B	$369/kgIHM	100.0%

The magnitude of the effect of license extensions, shown in Figure 5.3, remains nearly unchanged for various repository opening dates. The slight artificial narrowing of the gap for a repository opening date after 2050 simply reflects the fact that some of the waste management costs, like the repository closure costs, occur after 2100 in the SNuFManager simulator and therefore were not tallied because the relevant time horizon for all simulations presented in this thesis concluded at that time.

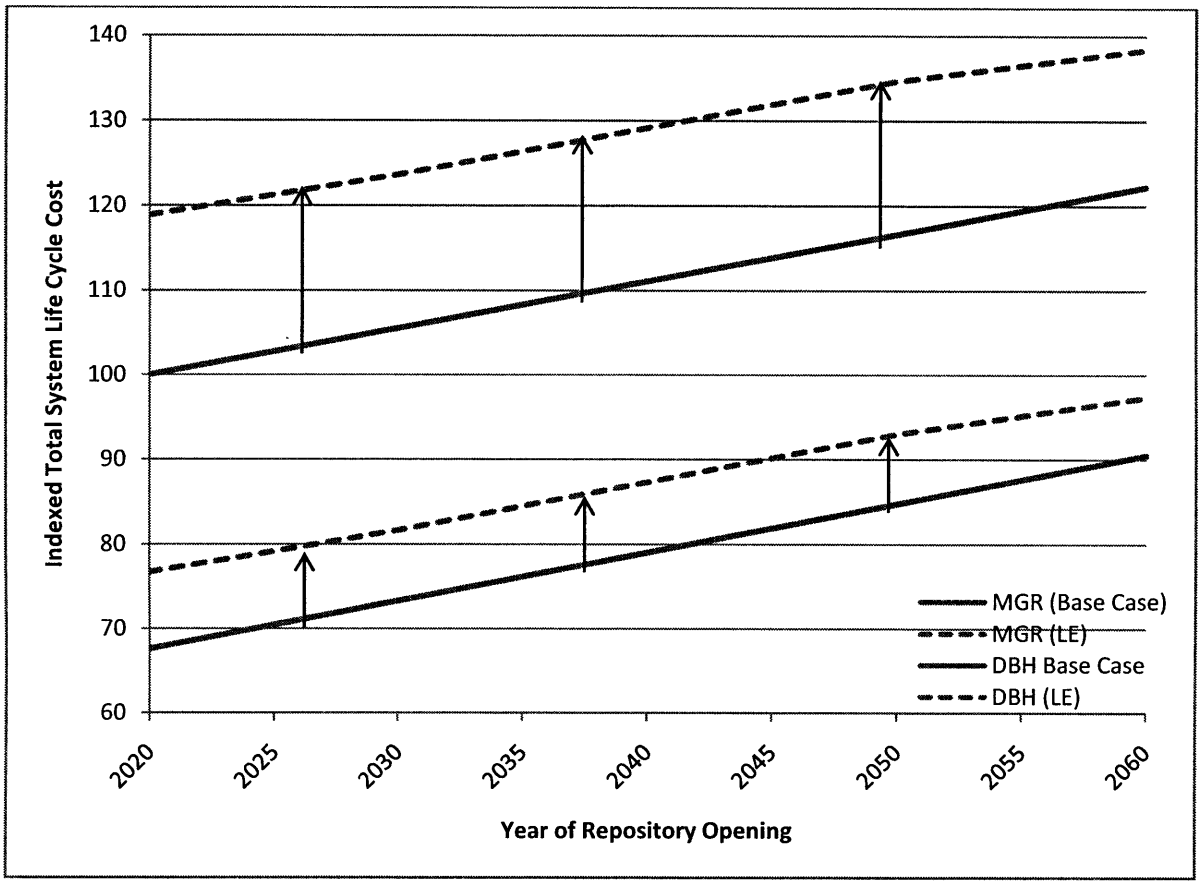


Figure 5.3. Indexed Total System Life Cycle Cost for the Base Cases with License Extensions.

5.3 Increasing Burnup and Improving Capacity Factor

Trends in reactor discharge burnup continue to rise. While the historical average for the accumulated spent nuclear fuel remains down at around 35 GWD per metric tonne of initial heavy metal, current values tend to hover around 50 GWD and future goals are to achieve roughly 65 GWD discharge burnup. If one ignores the possible additional costs of monitoring and transporting higher burnup fuel, this trend greatly benefits the Office of Civilian Radioactive Waste Management because a larger nuclear waste fee will be collected for a given volume of waste.

In order to completely capture this effect, one must rewrite the SNuFManager code to track assemblies and provide a detailed list of parameters to characterize the spent fuel. This requires significantly more data management and might include defining the assemblies as objects, for which using a language like C++ may make more sense at this level of detail. Because global assumptions exist in the current version of SNuFManager, variables like the capacity of spent fuel pools, which are generally defined in terms of assemblies, are converted and approximated in units of metric tonnes when writing the input deck. This means that in order to properly examine the impact of changing burnup, one must recalculate all the variables for each reactor that depend on these assumed values and come up with plausible changes in cycle length or power to account for the changing burnup. The effort required to do all this exceeds the expected benefit from the exact results, so we restrict ourselves to a qualitative discussion of the effects of changing burnup. Table 5.6 presents a summary of these variables with the assumed values for changing burnup.

Table 5.6. Burnup Dependent Variables.

Variable	35 GWD/MT	50 GWD/MT*	65 GWD/MT
Repository Acceptance	4,286 MT/year	3,000 MT/year	2,308 MT/year
Initial Conditioning	$105/kgIHM	$150/kgIHM	$195/kgIHM
Inter-region Transport	$5 - $10/kgIHM	$7 - $14/kgIHM	$9 - $18/kgIHM
Intra-region Transport	$14 - $18/kgIHM	$20 - $26/kgIHM	$26 - $34/kgIHM
Final Conditioning (Yucca)	$192/kgIHM	$274/kgIHM	$356/kgIHM
Final Conditioning (DBH)	$46/kgIHM	$66/kgIHM	$86/kgIHM

* The base case assumes a 50 GWD/MT burnup.

As a reminder, Table 5.6 assumes the repository is not limited by the heat generation and that a facility able to handle 300 casks of spent fuel a year would handle the same number of casks no matter what the composition of the spent fuel inside those assemblies. Therefore, assemblies with burnups of only 35 GWD have a lower loading of initial heavy metal and so a higher value in metric tonnes per year leads to the equivalent repository acceptance in units of assemblies. At very high burnups and for handling times that are relatively soon after reactor discharge, this assumption breaks down primarily because surface operations need additional safety measures and long-term temperature margins may not be maintained.

Increasing burnup effectively produces the same amount of electricity while slowing the generation of spent fuel assemblies. In the end, fewer assemblies reduces spending on conditioning and packaging that makes up a majority of the waste management life cycle cost. While competing forces, such as increased transportation costs for higher radioactivity waste, offset the benefits associated with reduced conditioning and packaging, these portions contribute minimally and the net effect results in fewer expenditures for higher burnup waste.

Changes in the capacity factor relate to the changes in the burnup, which the SNuFManager input deck assumes for all reactors. With a fixed cycle length, increasing the capacity factor requires either increasing the initial enrichment and discharge burnup or increasing the fraction of the core that gets discharged and refueled each cycle. Following the latter option leads to a straightforward increase in waste accumulation directly proportional to the increase in capacity factor and therefore a significant increase in the waste management life cycle cost. Although the life cycle cost increases, the system averaged waste management cost should not increase because the characteristics of the spent fuel remain unchanged.

Summary

This chapter reviewed the two base cases of the total system life cycle costs and showed that for the most realistic range of economic parameters, the Yucca Mountain repository type costs more than the deep borehole disposal concept. Under unlikely conditions, a high borehole cost and a low Yucca Mountain cost could reverse the technology preference by economic standards. However, few scenarios exist in which the life cycle cost estimates would err in opposite directions. Globally consistent trends, such as a worldwide increase in commodity prices that drive all disposal costs up or a revolutionary initial conditioning technology that drives all disposal costs down, represent more likely circumstances, in which case the economic preference of the deep borehole concept will likely persist.

The license extensions represent an added waste management burden and a source to drive up the life cycle cost, but do so in a disproportionate manner such that the levelized waste management cost drops considerably. The estimated cost of the Yucca Mountain waste management system grows from $59.9B to $71.1B in 2000$ because of the roughly one-third more spent fuel requiring disposal, but the levelized cost falls from $642 to $572 per kilogram of initial heavy metal. Similarly, the deep borehole life cycle cost estimate grows from $40.5B to $45.9B, while the levelized cost drops from $434 to $369 per kilogram.

The results presented in this thesis assumed burnup values of 50 GWD per metric tonne and 85% capacity factors. In general, the trends of increasing burnup and capacity factors work in favor of the waste management system by reducing the volume of waste while increasing the nuclear waste fee collected to pay for disposal.

Chapter 6: Conclusions

6.1 Results Summary

The SNuFManager tool developed in the present work provides policy makers the opportunity to quickly and cheaply simulate the costs and benefits of waste management decisions under various scenarios. Doing so with a sophisticated computer model avoids the expense and high risks associated with making poor decisions in real life. SNuFManager simulates the waste management system with two distinct solvers. The first solver utilizes the principles of System Dynamics to deterministically walk through each time step and recalculate the physical stocks and flows based on the conditions of the system and an unloading algorithm programmed in the simulator. By default, SNuFManager makes use of the simple unloading algorithm that prioritizes flows from reactors with the earliest shutdown dates. Once the program resolves the logistics, SNuFManager evaluates the recorded stocks and flows and applies the economic data provided in the user defined input file to account for annual expenditures broken down by function.

SNuFManager allocates memory dynamically and relies on user defined input in an effort to avoid making the program too rigid and hardwired, so that a diverse set of users can benefit from its generality. Therefore, those who have a well defined problem and comprehensively understand their costs simply need to take the sample inputs provided in the appendix and modify the variables as necessary. Chapter 3 provides cost estimates for each of the steps in the waste management process, which are used in the provided sample input files. Many of the expenditures consist of proprietary components that this thesis infers from publications in the open literature or estimates based on best judgment. Tables 6.1 and 6.2 summarize the levelized waste management costs for each category that SNuFManager distinguishes. For the mined geologic repository strategy, the highest operation levelized costs include the final conditioning, initial conditioning, and repository costs. Notice that shutdown reactor spent fuel storage accounts for the fourth most expensive cost category for the Yucca Mountain strategy, but this ranking is deceptive because only a small fraction of the total waste accumulated stays in shutdown reactor wet storage for any significant period of time, as illustrated by the base case system averaged cost column. Considering this caveat, the same items are also the largest contributors to the

deep borehole disposal strategy, but in a slightly different ranking of initial conditioning, repository, and final conditioning.

Table 6.1. Mined Geologic Repository Waste Management Cost Summary.

Cost Category	Operation Levelized Cost*	Base Case System Averaged Cost
Shutdown Spent Fuel Pool Storage†	$113/kgIHM	$54/kgIHM
Initial Conditioning for Transp. and Dry Storage	$150/kgIHM	$150/kgIHM
Online Reactor Onsite Dry Storage‡	$12/kgIHM	$60/kgIHM
Shutdown Reactor Onsite Dry Storage§	$57/kgIHM	
Inter-region Transportation	$7 - $14/kgIHM	$22/kgIHM
Intra-region Transportation	$20 - $26/kgIHM	
Centralized Interim Storage**	$31/kgIHM	$0/kgIHM
Final Conditioning and Packaging (MGR)	$274/kgIHM	$274/kgIHM
Repository, Monitoring, and Closure (MGR)††	$118/kgIHM	$83/kgIHM‡‡

Table 6.2 Deep Borehole Repository Cost Summary.

Cost Category	Operation Levelized Cost	Base Case System Averaged Cost
Final Conditioning and Packaging (DBH)	$66/kgIHM	$66/kgIHM
Repository, Monitoring, and Closure (DBH)	$101/kgIHM	$83/kgIHM

The modularity of deep boreholes provides a great economic advantage over a Yucca Mountain system because of the dispersion of costs over a much longer period of time and therefore a reduction in finance charges. However, Tables 6.1 and 6.2 illustrate the greatest cost savings in going to a deep borehole disposal technology comes from the drastically reduced engineering requirements for the borehole disposal canister that must only maintain integrity for a relatively short period of time.

---

* Reminder, costs are undiscounted and expressed in 2000$ to remain consistent with the DOE 2001 TSLCC report.
† 1,000 MT for 10 years.
‡ 1,000 MT for 10 years.
§ 1,000 MT for 10 years.
** 40,000 MT for 50 years.
†† Assumes 93,200 MT, ignores indirect financing charges, but includes $30M/yr base O&M cost for 80 years.
‡‡ Less than the operation levelized cost because we very conservatively assumed no increase in subsurface facility costs with a higher capacity of 93,200 MT.

No matter what the disposal technology, the single repository waste management total system life cycle cost increases proportionately with any delay of the repository opening and quantifying these costs provides policy makers better information for their decision analysis. The SNuFManager results presented in this thesis suggest the utilities incur over a $330M cost of delay for each year the repository remains unopened beyond 2020, shown in Figure 6.1. Interestingly enough, the delay cost of a hypothetical regional repository system where four borehole disposal complexes accept the spent fuel from nearby reactors follows lesser slope indicating the delay cost is mitigated by the higher fuel acceptance rates. This work did not explore in detail the economic benefits of a four repository system because the uncertainties of the cost of each repository made confidence in the nominal results of the regional strategy only strong enough to say that the life cycle cost appears comparable to that of the Yucca Mountain strategy.

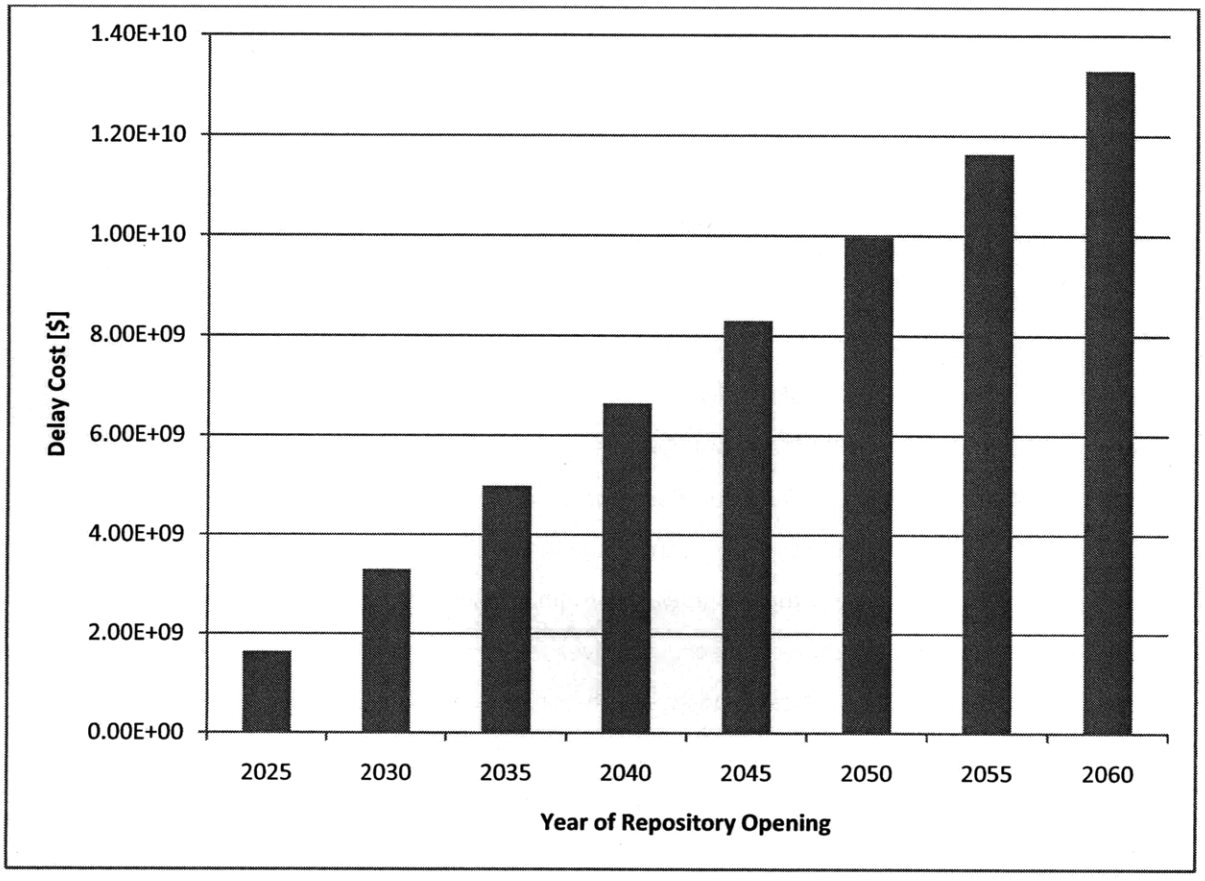


Figure 6.1. Estimated Delay Cost For Single Repository Waste Management System.

Technically and politically, a four borehole strategy contains real advantages over a single repository. For instance, the strategy significantly reduces the transportation burden. Although that cost generally remains below 10% of the life cycle cost, and the scientific community agrees on the excellent safety of spent nuclear fuel transportation, making the case that the government considers system changes in the name of public safety may constitute a necessary political bargaining chip in selling a disposal strategy to the public. Agreeing on four disposal sites also reduces the burden on Nevada, which could feel unfairly singled out as the site of early above ground nuclear weapons testing and now the proposed site for the entire country's nuclear waste. The original Nuclear Waste Policy Act included the need for multiple repositories for this reason of distributional risk equity, and again this idea may represent another item worth compromising over, because the strategy appears capable of coming in at a budget less than the Yucca Mountain Project.

With all the alternatives for political leaders to choose from and with the high cost of delaying the implementation of a waste management strategy, interim storage provides an interesting chance for the government to step up and claim responsibility for the country's nuclear waste and for policy makers to continue to analyze the benefits of various disposal alternatives. This report stops short of recommending interim storage without preconditions. In fact, given the government's history with delays in the waste management situation, it seems likely that should an interim storage facility get constructed without a clear strategy to move the fuel some fifty to one hundred years later, the temporary interim storage facility could easily become a permanent "interim" storage facility. This outcome of permanently passing on the waste management burden for centuries fails to meet generational responsibility objectives and therefore a conscientious policy maker would not endorse such a strategy. However, a wise policy maker should see the flexibility that interim storage provides and consider signing onto a strategy that includes fifty to one hundred years of interim storage followed by some agreed upon disposal technology because it gives the country a relatively inexpensive grace period with which to continue investigating disposal alternatives without making irreversible investments. Figure 6.2 illustrates the value of interim storage by showing the cost savings on what the government would spend given various delays in the opening of Yucca Mountain.

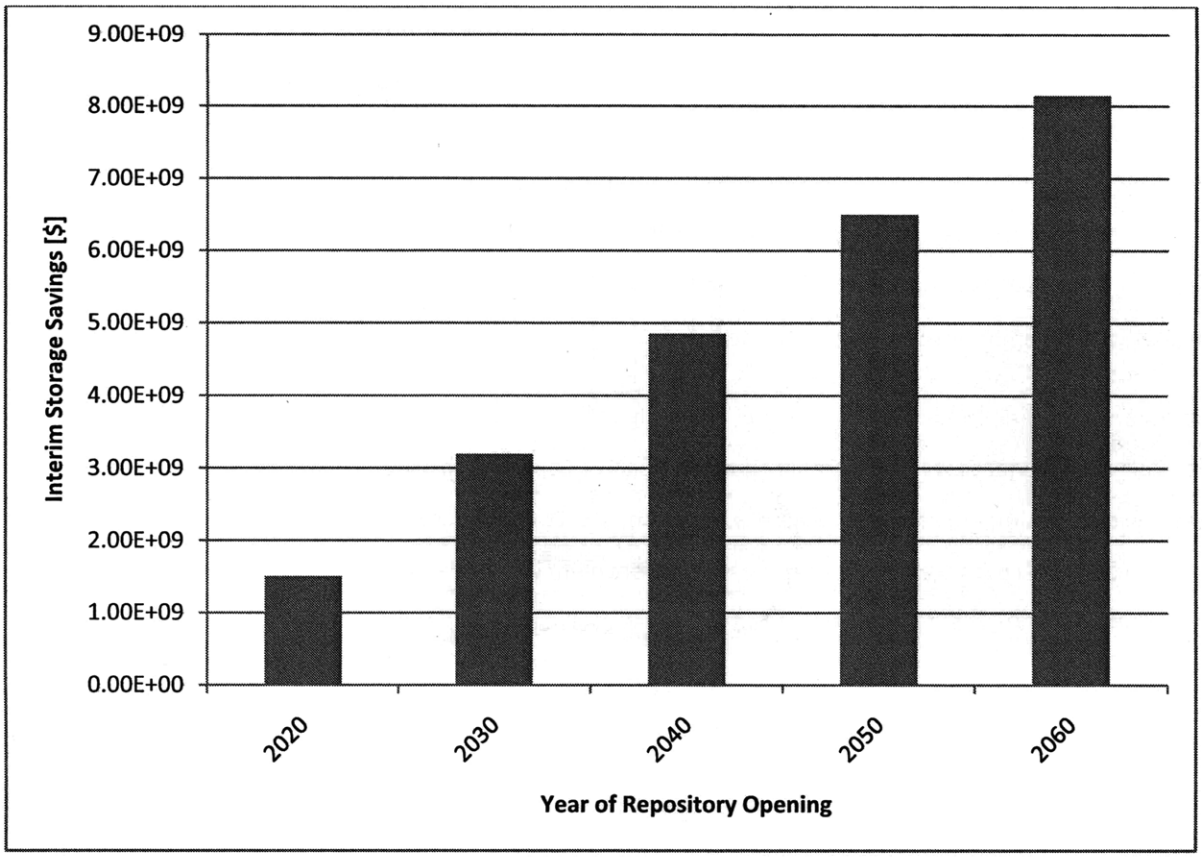


Figure 6.2. Interim Storage Life Cycle Cost Savings Given Delays in Repository Opening.

After numerous simulations of the waste management system and considering the economic and political challenges involved in dealing with the spent fuel, the recommended approach involves

(a) opening a large interim storage facility as soon as possible
(b) endorsing Yucca Mountain Project as the waste management solution, but continue slow concept development for decades while interim storage provides a cheap grace period
(c) analyze and propose a deep borehole disposal solution in parallel to the Yucca Mountain research quickly so Congress could reasonably change direction to see economic benefits of deep boreholes without making too large an investment in Yucca Mountain

This strategy meets all the demands of technical viability, cost effectiveness, and maintains a flexibility to change direction for many decades before significant irreversible investment in the ultimate disposal technology takes place. For instance, the U.S. could decide to greatly reduce, but not eliminate, its need for nuclear waste storage by closing the fuel cycle. Then the government might prefer to use a limited

number of boreholes to dispose of the reprocessing waste and the isotopes fast reactors cannot easily transmute. As the recent instability in uranium prices showed, keeping reprocessing on the table seems a prudent policy direction because prolonged inflation in fuel prices can eventually make sustainable recycling of spent fuel economically efficient and open the way for breeder reactor deployment. Developing an expensive permanent disposal facility now reduces the government's flexibility in responding to changing market conditions and seems financially risky, given the growing likelihood that the U.S. might begin reprocessing this century.

A brief investigation into changing the unloading algorithm led to relatively insignificant changes in the life cycle cost. Benefits were only realized when delays in opening the repository led to a far greater waste accumulation at reactor sites across the country that could actually benefit from a more sophisticated fuel management scheme. Until the problem gets too challenging, the added complications of such a strategy will not result in life cycle cost savings.

As previously mentioned, the SNuFManager tool benefits policy makers by estimating the impact of changing policies and scenarios on the magnitude and direction of changes in the life cycle cost. For this reason, this thesis reports the majority of the life cycle costs as an index relative to the base case assuming a mined geologic repository opening in 2020. This helps the reader to interpret the results, but fails to identify the uncertainty associated with the economic estimates. Figure 6.3 reminds us of the simplistic uncertainty analysis comparing all the most and least expensive costs presented in the third chapter with the nominal results for the original base case and the deep borehole base case. The results strongly suggest the borehole disposal technique saves significant money, over 30% in the median estimates, and Yucca Mountain only gains economic preference in unlikely scenarios in which the nominal values significantly underestimated the costs for the borehole disposal and significantly overestimated the costs for the Yucca Mountain strategy.

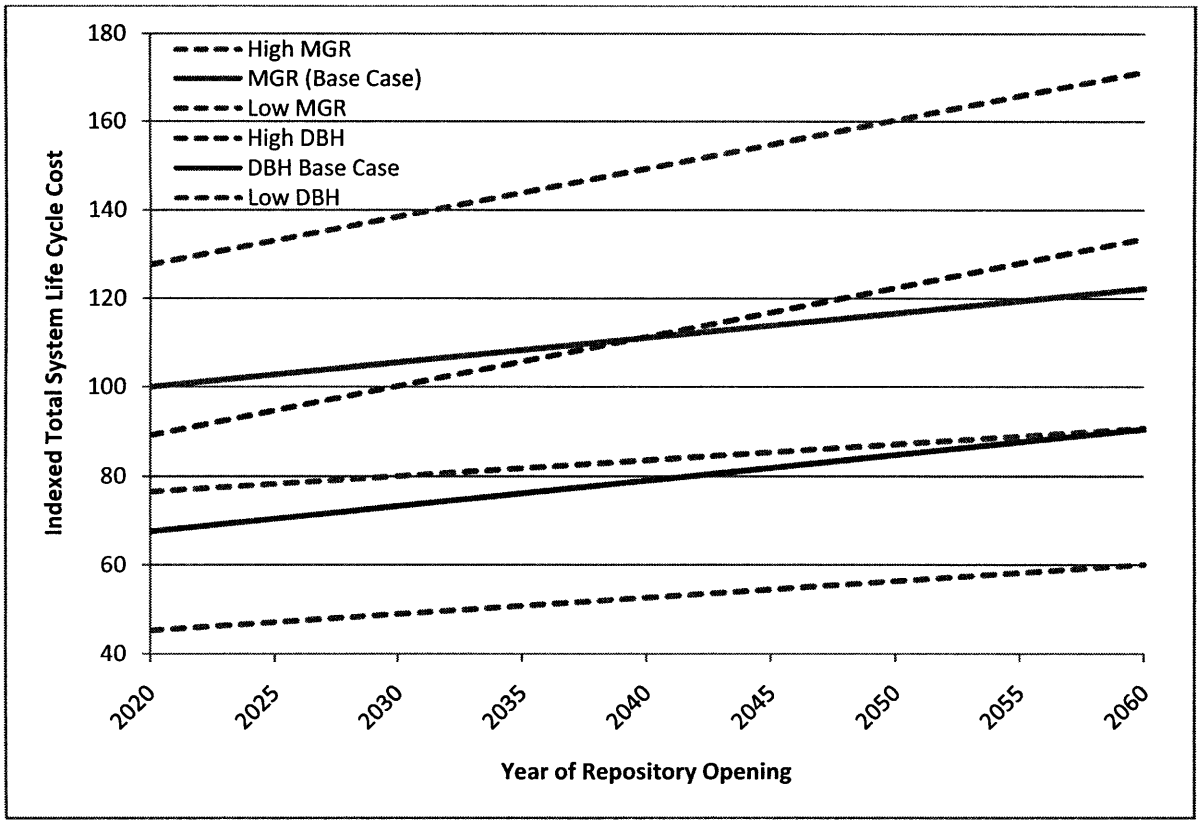


Figure 6.3. Uncertainty Bands of Base Case Indexed Total System Life Cycle Cost.

6.2 Future Work

Three areas may benefit from further work. Updating and reviewing the economic and spent fuel data input will improve the value of the results. In addition, several programming developments as well as strategy and scenario considerations might bring further significance to this project.

The current input data accurately characterize the spent fuel loading and capacity as well as the reactor discharge rates. However, the input files utilized by this thesis ignored any spent fuel already stored in dry storage by 2002 when the Department of Energy compiled the spent fuel database and ignored any spent fuel stored in offsite facilities, such as the General Electric wet storage facility in Morris, Illinois.

The economic input poses more challenges because of the proprietary nature of many aspects of the fuel cycle. Unfortunately, the least developed portions of the waste management expenses also represent the most significant contributors to the life cycle cost. In particular, both the initial and final conditioning and packaging modules need considerable review. Unfortunately, the published data on these stages remains quite limited because vendors generally perform these services and like to keep their costs proprietary. The work needs some narrowing of the uncertainties associated with borehole drilling because the documented price range varied greatly and the need for many boreholes means narrowing the uncertainty on a single hole gets compounded many times for the total disposal system.

Of course all of the economic input benefits from additional review and improvement, but most of the remaining modules contribute less to the life cycle cost and therefore more sophisticated or accurate characterization of the costs will add somewhat less to the analysis. For instance, some might consider the transportation cost overly conservative because of the use of the high-end impact limiters on each transportation cask. However, the transportation cost generally totals less than 5% of the life cycle cost so significant improvement in accuracy offers minimal benefit.

The SNuFManager program itself could use more development for improved generality as well as functionality to handle more complicated scenarios and conditions. As discussed earlier, the suggestion of tracking each assembly requires complete overhaul of the code, but for the users who need a highly accurate characterization of spent fuel conditions, this may afford that detail. This requires the consideration of assemblies as objects and the fact that there are assemblies of many different types complicates the problem. By reducing the problem to mass flows, the current version of SNuFManager avoids these obstacles.

For the most part, the code considers general circumstances in solving the physics of the spent fuel stocks and flows. However, the economic solver currently requires annual time steps and so further programming must continue to generalize the economics. Furthermore, some areas of minor significance could use further relaxation of assumptions. For instance, the code should allow the declaration of individual reactor operation variables like power*, cycle length, burnup, capacity factor, and efficiency in the input deck instead of assuming global values for many of these variables and preprocessing the data before generating the input deck. Likewise, although the transportation component constitutes a minor fraction of the life cycle cost, one might benefit from an update of SNuFManager in which the input explicitly states the required distances of travel to both the nearest interim storage facility and the nearest repository so the code can calculate the exact transportation charges instead of assuming all the transportation from region X to region Y costs the same amount. Some other minor updates could significantly improve the usability of the code. Right now no warning messages exist and the user guide provided in the appendix simply reviews the basics because the appendix also contains the full and commented Fortran90 version of the code. Other conditions must be met, such as the fact that the code insists on at least as many interim storage sites as repositories and that this value either equal one or the number of defined regions. Currently, this problem can be circumvented by defining a negligibly small interim storage site with a negligible cost. The interim storage cannot presently exist before or after the simulation time horizon, nor can any spent fuel remain outside of the repository when the simulation ends. In essence, the current version of the code requires the complete solution of the waste management problem before analyzing and evaluating the economics.

More advanced functionality might contribute to gaining additional insight from the use of the code. Creating a stochastic version of the code capable of more precisely analyzing the uncertainty of the inputs could lead to the more decisive ranking of policy alternatives. A Monte Carlo analysis requires the augmentation of the nominal input deck with some measurement of uncertainty and some suggestion of the distribution profile of each variable. This clearly adds more complication than the average reader of this thesis requires, but fully informed policy makers need these results to completely analyze their decision.

Finally, some suggest the inclusion of a set of market rules to allow for reactor site bidding on waste management capacity at each time step. Similarly, a significant piece of additional work would attempt

* Actual power levels are assumed explicitly for each reactor and not globally assumed like the other variables.

to value and compare the value of the flexibility provided by different scenarios using an options framework. This is likely to further differentiate the mined repository from the borehole strategy. One could base an entire new thesis on any of the last few upgrades, but successful implementation of these features would drastically expand the functionality of SNuFManager beyond its current stage of simply providing a proof of principle on the strategic and economic viability of waste management alternatives.

Appendix A:
Vensim Model of Simplified Single Reactor Waste Management System

Introduction

Vensim is a System Dynamics software tool that allows a user to create dynamic stock and flow models in a basic programming language. The following is an extremely rudimentary version of the earliest SNuFManager concept that was served as a proof-of-principle that the System Dynamics approach could be used in a modified fashion on the combinatorial complexity of a real waste management system. The visual representation of the modeling process can be seen in the second chapter.

Vensim Model

```
RequiredAAASFPRemovalRate=
        IF THEN ELSE(Time>=RxAAAShutdownTime,Cold Used Fuel In Rx AAA SFP/TIME STEP,IF THEN
ELSE\
                ((RxAAASFPCapacity-RxAAASFPLoading
        )<=Fuel In Rx AAA, Fuel In Rx AAA/TIME STEP, 0))
        ~       MT/Year
        ~               |

Region1BoreholeFillRateFromIS=
        Region1MaxBoreholeFillRateFromDSC-RxAAADSC2BoreholeRate
        ~       MT/Year
        ~               |

Used Fuel In All SFPs=
        RxAAASFPLoading
        ~       MT
        ~               |

Fuel In All Reactors=
        Fuel In Rx AAA
        ~       MT
        ~               |

RxAAADSC2ISRate=
        MIN(Region1MaxISFillRateFromDSC, Used Fuel In Rx AAA DSC/TIME STEP)
        ~       MT/Year
        ~               |

RxAAADSCAccumulationRate=
        MAX( RequiredAAASFPRemovalRate-RxAAASFP2BoreholeRate-RxAAASFP2ISRate, 0)
        ~       MT/Year
        ~               |

Used Fuel In All IS=
        Used Fuel In Region 1 IS
        ~       MT
        ~               |

Region1MaxISFillRate=
        100
        ~       MT/Year
        ~               |

RxAAASFP2ISRate=
        MIN(Region1ISFillRate, AAAMaxSFP2ISRate)
        ~       MT/Year
```

```
        ~               |

Region1IS2BoreholeRate=
        MIN(Region1BoreholeFillRateFromIS, Used Fuel In Region 1 IS/TIME STEP)
        ~       MT/Year
        ~               |

Region1MaxISFillRateFromDSC=
        Region1ISFillRate-RxAAASFP2ISRate
        ~       MT/Year
        ~               |

Region1ISFillRate=
        MIN((Region1ISCapacity-Used Fuel In Region 1 IS)/TIME STEP, Region1MaxISFillRate)
        ~       MT/Year
        ~               |

Used Fuel In All Boreholes=
        Used Fuel In Region 1 Boreholes
        ~       MT
        ~               |

Used Fuel In Region 1 IS= INTEG (
        RxAAADSC2ISRate+RxAAASFP2ISRate-Region1IS2BoreholeRate,
                0)
        ~       MT
        ~               |

Region1MaxBoreholeFillRateFromDSC=
        Region1BoreholeFillRate-RxAAASFP2BoreholeRate
        ~       MT/Year
        ~               |

RxAAADSC2BoreholeRate=
        MIN(Region1MaxBoreholeFillRateFromDSC, Used Fuel In Rx AAA DSC/TIME STEP)
        ~       MT/Year
        ~               |

Used Fuel In All DSCs=
        Used Fuel In Rx AAA DSC
        ~       MT
        ~               |

Region1ISStartTime=
        35
        ~       Year
        ~               |
```

```
RxAAASFP2BoreholeRate=
        MIN(Region1BoreholeFillRate, RequiredAAASFPRemovalRate)
        ~       MT/Year
        ~               |

Region1ISCapacity=
        500*PULSE(Region1ISStartTime, 1e+006)
        ~       MT
        ~               |

Used Fuel In Region 1 Boreholes= INTEG (
        Region1IS2BoreholeRate+RxAAADSC2BoreholeRate+RxAAASFP2BoreholeRate,
                0)
        ~       MT
        ~               |

Used Fuel In Rx AAA DSC= INTEG (
        RxAAADSCAccumulationRate-RxAAADSC2BoreholeRate-RxAAADSC2ISRate,
                0)
        ~       MT
        ~               |

AAAMaxSFP2ISRate=
        RequiredAAASFPRemovalRate-RxAAASFP2BoreholeRate
        ~       MT/Year
        ~               |

Cold Used Fuel In Rx AAA SFP= INTEG (
        RxAAASNFCoolingRate-RxAAADSCAccumulationRate-RxAAASFP2BoreholeRate-
RxAAASFP2ISRate,
                100)
        ~       MT
        ~               |

Fuel In Region 1 Reactors=
        Fuel In Rx AAA
        ~       MT
        ~               |

Fuel In Rx AAA= INTEG (
        RxAAALoadRate-RxAAADischargeRate,
                100)
        ~       MT
        ~               |

Hot Used Fuel In Rx AAA SFP= INTEG (
        RxAAADischargeRate-RxAAASNFCoolingRate,
                RxAAADischargeRate*SFPCoolingTime)
```

```
        ~       MT
        ~               |

Region1BoreholeCapacity=
        1000*PULSE(Region1BoreholeStartTime,
1e+006)+1000*PULSE(Region1BoreholeExpansion1Time\
                , 1e+006)+1000*PULSE(Region1BoreholeExpansion2Time, 1e+006)
        ~       MT
        ~               |

Region1BoreholeExpansion1Time=
        65
        ~       Year
        ~               |

Region1BoreholeExpansion2Time=
        90
        ~       Year
        ~               |

Region1BoreholeFillRate=
        MIN((Region1BoreholeCapacity-Used Fuel In Region 1 Boreholes)/TIME STEP,
Region1MaxBoreholeFillRate\
                )
        ~       MT/Year
        ~               |

Region1BoreholeStartTime=
        40
        ~       Year
        ~               |

Region1MaxBoreholeFillRate=
        50
        ~       MT/Year
        ~               |

RxAAADischargeRate=
        IF THEN ELSE(Time<=RxAAAShutdownTime, 25, Fuel In Rx AAA/TIME STEP)
        ~       MT/Year
        ~               |

RxAAALoadRate=
        IF THEN ELSE(Time<=RxAAAShutdownTime, 25, 0)
        ~       MT/Year
        ~       Reactor stops loading once time reaches "shutdown time"
        |
```

```
RxAAASFPCapacity=
        500
        ~       MT
        ~               |

RxAAASFPLoading=
        Cold Used Fuel In Rx AAA SFP+Hot Used Fuel In Rx AAA SFP
        ~       MT
        ~               |

RxAAAShutdownTime=
        50
        ~       Year
        ~               |

RxAAASNFCoolingRate=
        DELAY MATERIAL(RxAAADischargeRate, SFPCoolingTime, RxAAADischargeRate, 0)
        ~       MT/Year
        ~               |

SFPCoolingTime=
        10
        ~       Year
        ~               |

********************************************************
        .Control
********************************************************~
                Simulation Control Parameters
        |

FINAL TIME  = 100
        ~       Year
        ~       The final time for the simulation.
        |

INITIAL TIME  = 0
        ~       Year
        ~       The initial time for the simulation.
        |

SAVEPER  =
    TIME STEP
        ~       Year [0,?]
        ~       The frequency with which output is stored.
        |

TIME STEP  = 1
```

~ Year [0,?]
~ The time step for the simulation.
|

\\\---/// Sketch information - do not modify anything except names
V300 Do not put anything below this section - it will be ignored
*Main
$192-192-192,0,Times New Roman|12||0-0-0|0-0-0|0-0-255|-1--1--1|-1--1--1|96,96,100,0
10,1,Fuel In All Reactors,217,373,34,19,8,3,0,0,0,0,0,0
10,2,Used Fuel In All SFPs,415,376,53,19,8,3,0,0,0,0,0,0
10,3,Used Fuel In All DSCs,637,376,53,19,8,3,0,0,0,0,0,0
10,4,Used Fuel In All IS,874,377,53,19,8,3,0,0,0,0,0,0
10,5,Used Fuel In All Boreholes,1098,377,53,19,8,3,0,0,0,0,0,0
10,6,SFPCoolingTime,419,493,54,11,8,3,0,0,0,0,0,0
10,7,Fuel In Rx AAA,217,411,39,19,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,8,7,1,0,0,0,0,0,64,0,-1--1--1,,1|(217,392)|
10,9,RxAAASFPLoading,415,414,76,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,10,9,2,0,0,0,0,0,64,0,-1--1--1,,1|(415,407)|
10,11,Used Fuel In Rx AAA DSC,637,414,57,19,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,12,11,3,0,0,0,0,0,64,0,-1--1--1,,1|(637,395)|
10,13,Used Fuel In Region 1 IS,874,415,46,19,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,14,13,4,0,0,0,0,0,64,0,-1--1--1,,1|(874,396)|
10,15,Used Fuel In Region 1 Boreholes,1098,415,70,19,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,16,15,5,0,0,0,0,0,64,0,-1--1--1,,1|(1098,396)|
\\\---/// Sketch information - do not modify anything except names
V300 Do not put anything below this section - it will be ignored
*Region1Summary
$192-192-192,0,Times New Roman|12||0-0-0|0-0-0|0-0-255|-1--1--1|-1--1--1|96,96,100,0
10,1,Fuel In Region 1 Reactors,360,-49,53,19,8,3,0,0,0,0,0,0
10,2,Used Fuel In Region 1 IS,585,285,40,20,3,3,0,0,0,0,0,0
10,3,Used Fuel In Region 1 Boreholes,583,533,40,20,3,3,0,0,0,0,0,0
1,4,6,3,4,0,0,22,0,0,0,-1--1--1,,1|(580,464)|
1,5,6,2,100,0,0,22,0,0,0,-1--1--1,,1|(580,354)|
11,6,636,580,409,8,6,33,3,0,0,4,0,0,0
10,7,Region1IS2BoreholeRate,670,409,82,9,40,3,0,0,-1,0,0,0
10,8,Region1BoreholeStartTime,166,667,86,9,8,3,0,0,0,0,0,0
10,9,Region1BoreholeCapacity,376,669,84,9,8,3,0,0,0,0,0,0
10,10,Region1BoreholeFillRate,584,671,80,9,8,3,0,0,0,0,0,0
10,11,Region1MaxBoreholeFillRate,582,727,94,9,8,3,0,0,0,0,0,0
1,12,8,9,0,0,0,0,0,64,0,-1--1--1,,1|(265,667)|
1,13,9,10,0,0,0,0,0,64,0,-1--1--1,,1|(475,669)|
1,14,11,10,0,0,0,0,0,64,0,-1--1--1,,1|(582,705)|
1,15,3,10,0,0,0,0,0,64,0,-1--1--1,,1|(583,600)|
10,16,TIME STEP,441,712,50,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,17,16,10,0,0,0,0,0,64,0,-1--1--1,,1|(508,692)|
10,18,Region1BoreholeExpansion1Time,174,703,108,9,8,3,0,0,0,0,0,0
1,19,18,9,0,0,0,0,0,64,0,-1--1--1,,1|(267,687)|

10,20,Region1BoreholeExpansion2Time,206,742,108,9,8,3,0,0,0,0,0,0
1,21,20,9,0,0,0,0,0,64,0,-1--1--1,,1|(284,708)|
10,22,Fuel In Rx AAA,361,10,39,19,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,23,22,1,0,0,0,0,0,64,0,-1--1--1,,1|(360,-13)|
10,24,RxAAASFPLoading,592,6,76,9,8,2,1,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
10,25,RxAAASFP2ISRate,585,324,77,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,26,25,2,0,0,0,0,0,64,0,-1--1--1,,1|(585,317)|
10,27,RxAAASFP2BoreholeRate,399,560,98,9,8,2,1,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
10,28,Region1MaxBoreholeFillRateFromDSC,855,671,128,9,8,3,0,0,0,0,0,0
10,29,RxAAADSCAccumulationRate,264,575,110,9,8,2,1,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
10,30,RxAAADSC2ISRate,585,324,79,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,31,30,2,0,0,0,0,0,64,0,-1--1--1,,1|(585,317)|
10,32,RxAAADSC2BoreholeRate,583,572,100,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,33,32,3,0,0,0,0,0,64,0,-1--1--1,,1|(583,565)|
10,34,RxAAASFP2BoreholeRate,583,572,98,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,35,34,3,0,0,0,0,0,64,0,-1--1--1,,1|(583,565)|
10,36,Region1ISStartTime,125,198,65,9,8,3,0,0,0,0,0,0
10,37,Region1ISCapacity,349,197,63,16,8,131,0,0,0,0,0,0
10,38,Region1ISFillRate,584,200,57,11,8,3,0,0,0,0,0,0
10,39,Region1MaxISFillRateFromDSC,895,201,106,9,8,3,0,0,0,0,0,0
1,40,36,37,0,0,0,0,0,64,0,-1--1--1,,1|(231,197)|
1,41,37,38,0,0,0,0,0,64,0,-1--1--1,,1|(462,197)|
1,42,38,39,0,0,0,0,0,64,0,-1--1--1,,1|(708,200)|
1,43,10,28,0,0,0,0,0,64,0,-1--1--1,,1|(688,671)|
10,44,Region1MaxISFillRate,581,129,73,9,8,3,0,0,0,0,0,0
1,45,44,38,0,0,0,0,0,64,0,-1--1--1,,1|(581,156)|
10,46,TIME STEP,486,160,50,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,47,46,38,0,0,0,0,0,64,0,-1--1--1,,1|(528,177)|
1,48,2,38,0,0,0,0,0,64,0,-1--1--1,,1|(584,244)|
10,49,RxAAASFP2BoreholeRate,829,628,98,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,50,49,28,0,0,0,0,0,64,0,-1--1--1,,1|(837,643)|
10,51,Region1BoreholeFillRateFromIS,1148,672,104,9,8,3,0,0,0,0,0,0
1,52,28,51,0,0,0,0,0,64,0,-1--1--1,,1|(1006,671)|
10,53,RxAAADSC2BoreholeRate,1100,631,100,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,54,53,51,0,0,0,0,0,64,0,-1--1--1,,1|(1118,646)|
1,55,51,7,1,0,0,0,0,64,0,-1--1--1,,1|(1019,506)|
10,56,TIME STEP,751,449,50,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,57,56,7,0,0,0,0,0,64,0,-1--1--1,,1|(714,431)|
1,58,2,7,1,0,0,0,0,64,0,-1--1--1,,1|(693,340)|
10,59,RxAAASFP2ISRate,846,164,77,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,60,59,39,0,0,0,0,0,64,0,-1--1--1,,1|(864,178)|
\\\---/// Sketch information - do not modify anything except names
V300 Do not put anything below this section - it will be ignored
*Reg1RxAAA
$192-192-192,0,Times New Roman|12||0-0-0|0-0-0|0-0-255|-1--1--1|-1--1--1|96,96,100,0
10,1,Fuel In Rx AAA,345,335,40,20,3,3,0,0,0,0,0,0
10,2,Hot Used Fuel In Rx AAA SFP,674,335,40,20,3,3,0,0,0,0,0,0
10,3,Cold Used Fuel In Rx AAA SFP,962,336,40,20,3,3,0,0,0,0,0,0

12,4,48,57,336,10,8,0,3,0,0,-1,0,0,0
1,5,7,1,4,0,0,22,0,0,0,-1--1--1,,1|(248,336)|
1,6,7,4,100,0,0,22,0,0,0,-1--1--1,,1|(123,336)|
11,7,48,186,336,6,8,34,3,0,0,1,0,0,0
10,8,RxAAALoadRate,186,355,57,11,40,3,0,0,-1,0,0,0
1,9,11,2,4,0,0,22,0,0,0,-1--1--1,,1|(574,335)|
1,10,11,1,100,0,0,22,0,0,0,-1--1--1,,1|(444,335)|
11,11,748,509,335,6,8,34,3,0,0,1,0,0,0
10,12,RxAAADischargeRate,509,352,73,9,40,3,0,0,-1,0,0,0
1,13,15,3,4,0,0,22,0,0,0,-1--1--1,,1|(873,332)|
1,14,15,2,100,0,0,22,0,0,0,-1--1--1,,1|(763,332)|
11,15,236,818,332,6,8,34,3,0,0,1,0,0,0
10,16,RxAAASNFCoolingRate,818,349,82,9,40,3,0,0,-1,0,0,0
1,17,18,3,100,0,0,22,0,0,0,-1--1--1,,1|(1058,335)|
11,18,716,1121,335,6,8,34,3,0,0,1,0,0,0
10,19,RxAAADSCAccumulationRate,1121,352,100,9,40,3,0,0,-1,0,0,0
10,20,RxAAAShutdownTime,187,271,74,9,8,3,0,0,0,0,0,0
1,21,20,7,0,0,0,0,0,64,0,-1--1--1,,1|(186,298)|
1,22,20,11,1,0,0,0,0,64,0,-1--1--1,,1|(397,229)|
10,23,Time,88,391,26,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,24,23,8,0,0,0,0,0,64,0,-1--1--1,,1|(128,376)|
1,25,1,11,1,0,0,0,0,64,0,-1--1--1,,1|(415,305)|
1,26,11,15,1,0,0,0,0,64,0,-1--1--1,,1|(658,242)|
10,27,Time,433,389,26,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
10,28,TIME STEP,582,388,50,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,29,27,12,0,0,0,0,0,64,0,-1--1--1,,1|(466,372)|
1,30,28,12,0,0,0,0,0,64,0,-1--1--1,,1|(549,372)|
10,31,SFPCoolingTime,857,279,65,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,32,31,15,0,0,0,0,0,64,0,-1--1--1,,1|(840,301)|
10,33,RxAAASFPLoading,818,418,67,9,8,3,0,0,0,0,0
1,34,2,33,1,0,0,0,0,64,0,-1--1--1,,1|(704,385)|
1,35,3,33,1,0,0,0,0,64,0,-1--1--1,,1|(923,396)|
10,36,Used Fuel In Region 1 IS,960,694,42,19,8,3,0,0,-1,0,0,0
10,37,Region1IS2BoreholeRate,724,691,91,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,38,37,36,0,0,0,0,0,0,0,-1--1--1,,1|(859,692)|
1,39,31,2,0,0,0,0,0,64,1,-1--1--1,,1|(777,302)|
10,40,Used Fuel In Region 1 Boreholes,951,86,66,19,8,3,0,0,-1,0,0,0
10,41,Region1IS2BoreholeRate,694,85,91,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,42,41,40,0,0,0,0,0,0,0,-1--1--1,,1|(828,85)|
1,43,44,3,100,0,0,22,0,0,0,-1--1--1,,1|(962,266)|
11,44,588,962,210,8,6,33,3,0,0,4,0,0,0
10,45,RxAAASFP2BoreholeRate,1059,210,89,9,40,3,0,0,-1,0,0,0
1,46,48,36,4,0,0,22,0,0,0,-1--1--1,,1|(962,598)|
1,47,48,3,100,0,0,22,0,0,0,-1--1--1,,1|(962,432)|
11,48,476,962,515,8,6,33,3,0,0,4,0,0,0
10,49,RxAAASFP2ISRate,1037,515,67,9,40,3,0,0,-1,0,0,0
10,50,RxAAASFPCapacity,550,477,69,9,8,3,0,0,0,0,0,0
10,51,RequiredAAASFPRemovalRate,818,480,103,9,8,3,0,0,0,0,0,0

1,53,50,51,0,0,0,0,0,64,0,-1--1--1,,1|(660,477)|
1,54,33,51,0,0,0,0,0,64,0,-1--1--1,,1|(818,442)|
10,55,TIME STEP,640,509,50,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,56,55,51,0,0,0,0,0,64,0,-1--1--1,,1|(719,496)|
10,57,AAAMaxSFP2ISRate,817,547,73,9,8,3,0,0,0,0,0,0
1,58,51,57,0,0,0,0,0,64,0,-1--1--1,,1|(817,506)|
1,59,51,45,1,0,0,0,0,64,0,-1--1--1,,1|(1028,375)|
1,60,51,19,1,0,0,0,0,64,0,-1--1--1,,1|(1013,447)|
1,61,57,48,1,0,0,0,0,64,0,-1--1--1,,1|(898,546)|
1,62,45,57,1,0,0,0,0,64,0,-1--1--1,,1|(1126,508)|
10,63,Used Fuel In Rx AAA DSC,1316,335,40,20,3,3,0,0,0,0,0,0
1,64,18,63,4,0,0,22,0,0,0,-1--1--1,,1|(1201,335)|
1,65,67,36,4,0,0,22,0,0,0,-1--1--1,,1|(1156,687)|
1,66,67,63,100,0,0,22,0,0,0,-1--1--1,,1|(1316,517)|
11,67,428,1316,687,6,8,34,3,0,0,1,0,0,0
10,68,RxAAADSC2ISRate,1316,704,69,9,40,3,0,0,-1,0,0,0
1,69,44,40,4,0,0,22,0,0,0,-1--1--1,,1|(962,154)|
1,70,72,40,4,0,0,22,0,0,0,-1--1--1,,1|(1162,85)|
1,71,72,63,100,0,0,22,0,0,0,-1--1--1,,1|(1313,204)|
11,72,380,1313,85,6,8,34,3,0,0,3,0,0,0
10,73,RxAAADSC2BoreholeRate,1313,68,91,9,40,3,0,0,-1,0,0,0
10,74,Region1BoreholeFillRate,825,179,89,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,75,74,44,1,0,0,0,0,64,0,-1--1--1,,1|(875,204)|
10,76,Region1ISFillRate,1196,570,68,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,77,76,49,1,0,0,0,0,64,0,-1--1--1,,1|(1096,562)|
1,78,45,18,1,0,0,0,0,64,0,-1--1--1,,1|(1100,262)|
1,79,49,19,1,0,0,0,0,64,0,-1--1--1,,1|(1116,445)|
10,80,Region1MaxBoreholeFillRateFromDSC,1170,120,137,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,81,80,72,0,0,0,0,0,64,0,-1--1--1,,1|(1249,100)|
1,82,63,72,1,0,0,0,0,64,0,-1--1--1,,1|(1411,182)|
1,83,63,67,1,0,0,0,0,64,0,-1--1--1,,1|(1412,494)|
10,84,TIME STEP,1258,23,50,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,85,84,73,0,0,0,0,0,64,0,-1--1--1,,1|(1281,42)|
10,86,Region1MaxISFillRateFromDSC,1175,822,116,9,8,2,1,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
10,87,Region1MaxISFillRateFromDSC,1173,637,116,9,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,88,87,67,0,0,0,0,0,64,0,-1--1--1,,1|(1247,663)|
10,89,TIME STEP,1381,764,50,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,90,89,68,0,0,0,0,0,64,0,-1--1--1,,1|(1352,737)|
1,91,3,51,1,0,0,0,0,64,0,-1--1--1,,1|(935,420)|
10,92,Time,728,524,26,11,8,2,0,3,-1,0,0,0,128-128-128,0-0-0,|12||128-128-128
1,93,92,51,0,0,0,0,0,64,0,-1--1--1,,1|(768,504)|
1,94,20,51,1,0,0,0,0,64,0,-1--1--1,,1|(621,596)|
1,95,1,51,1,0,0,0,0,64,0,-1--1--1,,1|(566,436)|

Appendix B: SNuFManager Fortran Model of Dynamically Allocated Waste Management System

Introduction

The SNuFManager tool is a dynamically allocated Fortran90 code that reads a user defined input and provides policy makers the opportunity to quickly and cheaply simulate the costs and benefits of waste management decisions under various scenarios. Doing so with a sophisticated computer model avoids the expense and high risks associated with making poor decisions in real life. SNuFManager simulates the waste management system with two distinct solvers. The first solver utilizes the principles of System Dynamics to deterministically walk through each time step and recalculate the physical stocks and flows of the waste masses based on the conditions of the system and an unloading algorithm programmed in the simulator. By default, SNuFManager makes use of the simple unloading algorithm that prioritizes flows from reactors with the earliest shutdown dates. Once the program resolves the physics of the waste management stocks and flows, SNuFManager evaluates the recorded stocks and flows and applies the economic data provided in the user defined input file to account for annual expenditures broken down by function. Before publically distributing SNuFManager, substantial effort must go into including detailed warning messages and making the program user friendly because the current revision requires significant experience to quickly debug errors in the input. A basic user's guide to creating input files can be found in Appendix C.

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! NAME: Taylor Allen Moulton (tmoulton@mit.edu)
! DATE: 2008-06-04
! PROJECT: Repository Thesis
! PROFESSOR: Richard Lester (rklester@mit.edu)
!
! PROBLEM:
! develop a program that can track the likely dynamics of waste accurately
! across the entire US fleet of nuclear reactors
! ASSUMPTION: only valid combinations of repositories and interim storage sites -
! (a) 1 repository, 1 interim storage site
! (b) 1 repository, [nRegions] interim storage sites
! (c) [nRegions] repositories, [nRegions] interim storage sites
!
! FUTURE IMPROVEMENT SUGGESTIONS:
! (a) deal with more general repository and interim storage scenarios
!     ... for example, having 1 IS to cover reg1&3, 1 IS to cover reg2, 0 IS to cover reg4
! (b) expand the input to specify the "start" and "end" for each reactor
!     ... in doing so, we can specify reactors that will come online in the future
! (c) create flexibility in repository capacity as a function of time
!     ... THIS WILL BE A REQUIREMENT BEFORE THE PROGRAM IS COMPLETE
!
! UPDATE: 2008-06-04 (Borehole08)
! built menu subroutine to read user defined input file
! note: should successfully allocate memory dynamically
! ASSUMPTION: only one spent fuel pool and dry storage stock per reactor site
!
! UPDATE: 2008-06-04 (Borehole09)
! added interim storage sites to input file and added initial stock variable
!
! UPDATE: 2008-06-05 (Repository10)
! added spent fuel cooling time to input, added initializeStock subroutine
! ASSUMPTION: all reactors are starting in dynamic equilibrium
! INVALID if simulation initializes within a couple years of starting a new reactor
! added initializeFlows subroutine
! ASSUMPTION: all repositories and interim storage facilities must start with
! no fuel and no fuel loading rate
! note: if you want to assume an interim storage facility exists, you may be able
! to model it as a reactor with zero loading/unloading and a shutdown time whenever
!
! UPDATE: 2008-06-05 (Repository11)
```

```
! built simulate subroutine structure
! finished calculateStocks based on previous flows
!
! UPDATE: 2008-06-06 (Repository12)
! added totalSiteReactorDischargeRate as an array that can be tracked
! started calculateFlows subroutine...
! completed reactorLoadingRate, reactorDischargeRate, and spentFuelPoolCoolingRate
!
! UPDATE: 2008-06-06 (Repository13)
! continued development of calculateFlows...
! added calculation of requiredSpentFuelPoolDischargeRate
! note: this is not properly initialized yet... it currently is
!       initialized to zero instead of generally
!
! UPDATE: 2008-06-07 (Repository14)
! continued development of calculateFlows...
! added calculation of requiredRepositoryFillRate
! added spentFuelPoolToRepositoryRate from soon to shutdown reactors
!
! UPDATE: 2008-06-07 (Repository15)
! continued development of calculateFlows...
! forgot to add condition of only 1 repository for Repository14
! it is included in this minor update
!
! UPDATE: 2008-06-07 (Repository16)
! removed "simulate" subroutine since it was simply calculateStocks
! followed by calculateFlows
! continued development of calculateFlows...
! note: calculateFlows is getting HUGE
! added spentFuelPoolToRepositoryRate from shutdown reactors algorithm
!
! UPDATE: 2008-06-07 (Repository17)
! added missing step of updating requiredSpentFuelPoolDischargeRate
! continued development of calculateFlows...
! added dryStorageToRepositoryRate from shutdown reactors algorithm
!
! UPDATE: 2008-06-07 (Repository18)
! fixed bug where I was using fuelInSpentFuelPool to determine rate
! instead of using coldFuelInSpentFuelPool
! continued development of calculateFlows...
! added spentFuelPoolToRepositoryRate from online reactors algorithm
```

```
!
! UPDATE: 2008-06-08 (Repository19)
! changed index of logic step (4) parts (c) and (d) to search down to
! passedTimeSteps of zero (not one), because a zero passed time step
! represents a reactor that just shut down
! continued development of calculateFlows...
! added dryStorageToRepositoryRate from online reactors algorithm
!
! UPDATE: 2008-06-08 (Repository20)
! continued development of calculateFlows...
! added interimStorageToRepositoryRate
!
! UPDATE: 2008-06-08 (Repository21)
! continued development of calculateFlows...
! added spentFuelPoolToInterimStorageRate from soon to shutdown reactors
!
! UPDATE: 2008-06-08 (Repository22)
! continued development of calculateFlows...
! added spentFuelPoolToInterimStorageRate from shutdown reactors algorithm
!
! UPDATE: 2008-06-09 (Repository23)
! continued development of calculateFlows...
! added dryStorageToInterimStorageRate from shutdown reactors algorithm
!
! UPDATE: 2008-06-09 (Repository24)
! continued development of calculateFlows...
! added spentFuelPoolToInterimStorageRate from online reactors algorithm
!
! UPDATE: 2008-06-09 (Repository25)
! continued development of calculateFlows...
! added dryStorageToInterimStorageRate from online reactors algorithm
!
! UPDATE: 2008-06-09 (Repository26)
! continued development of calculateFlows...
! added spentFuelPoolToDryStorageRate from remaining
! requiredSpentFuelPoolDischargeRate algorithm
!
! UPDATE: 2008-06-10 (Repository27)
! debug fuel in reactor... working well
! debug hot fuel in spent fuel pool... working well
! debug cold fuel in spent fuel pool... small issue that I will ignore
```

```
!       for now because it is relatively minor in magnitude, but there
!       is a slight numerical error when the spent fuel pool becomes
!       completely depleted... it will attempt to fill the required
!       repository and IS fill rates and lead to a small negative value
!       for the stock and subsequently a small negative flow during the
!       next time step to correct for this.
! debug fuel in repository... same issue about negative stocks and flows
!       I will need to fix this and I propose the addition of variables
!       like "available SFP discharge rate" that tally what changes I am
!       proposing and ensure that I do not go too high...
!       another bug found in this debug session... the integration of
!       fuelInRepository is not correct... rewrote calculateStocks to fix
! debug fuel in interim storage... same issue about negative values
! debug fuel in dry storage casks... working well
!
! UPDATE: 2008-06-10 (Repository28)
! rewrite calculateFlows with condensed do loops
! completed rewrite of logic step (4) ... parts (a) and (b)
! completed rewrite of logic step (4) ... part (c)
! completed rewrite of logic step (4) ... part (d)
! completed rewrite of logic step (4) ... part (e)
! completed rewrite of logic step (4) ... part (f)
!
! UPDATE: 2008-06-10 (Repository29)
! major debugging
!
! UPDATE: 2008-06-12 (Repository30)
! fix negative stock and flow issue... this took a couple days
!
! UPDATE: 2008-06-12 (Repository31)
! correct flow into interim storage... it was going one time step too soon
!
! UPDATE: 2008-06-12 (Repository32)
! add flexible repository space
! note: we are assuming the repository is only expanding
! added sumStockInSpentFuelPools
! added sumStockInDryStorage
! added sumStockInInterimStorage
! added sumStockInRepositories
!
! UPDATE: 2008-06-12 (Repository33)
```

```
! add printResults subroutine
! fixed if statement error in calculateFlows
! fixed index error in the integration of fuel in interim storage
! fixed index of interimStorageFillRate in calculateFlows
! fixed second error in integration of interim storage
!
! UPDATE: 2008-06-24 (Repository34)
! add calculateSimpleFlows subroutine
! corrected index error in requiredSpentFuelDischarge calculation
!
! UPDATE: 2008-06-27 (Repository35)
! add economic parameters to input deck
!
! UPDATE: 2008-07-30 (SNuFManager36)
! edit repository cost structure
!
! consider flexible interim storage space
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

program SNuFManager36
use problemParameters
implicit none

    integer :: countTime

    ! the following variables are defined in the menu subroutine
    integer :: totalReactorSites
    integer :: totalReactors

    ! the following variables are defined in countFuelInReactor subroutine

    !double precision :: totalFlowIntoRepositories
    double precision :: totalFlowIntoInterimStorage

    integer :: countOriginRegion
    integer :: countDestinationRegion
    integer :: countInterimStorageSite
    integer :: interimStorageShutdownTimeStep
    integer :: countRepository
    integer :: countRepositoryExpansion
    integer :: repositoryShutdownTimeStep
```

```
countTime = 1

call menu(countTime, totalReactorSites, totalReactors)
!debug
write(*,*) totalReactorSites, 'totalReactorSites', totalReactors, 'totalReactors'

! first perform fuel management calculations

! fuel management part a: initialize stocks and flows
call initializeStocks(countTime)
call initializeFlows(countTime)

! fuel management part b: simulate future stocks and flows
do countTime = 2, nTimeSteps
  call calculateStocks(countTime)
  !call calculateFlows(countTime)
  call calculateSimpleFlows(countTime)

  ! sum stocks
  call sumStockInReactors(countTime)
  call sumStockInSpentFuelPools(countTime)
  call sumStockInDryStorage(countTime)
  call sumStockInInterimStorage(countTime)
  call sumStockInRepositories(countTime)

  ! sum flows
  call sumFlowIntoDryStorage(countTime)
  call sumFlowIntoInterimStorage(countTime)
  call sumFlowIntoRepositories(countTime)

enddo ! countTime = 1, nTimeSteps

! now perform economic calculations
! economics part a: most of annual economic expenses

do countTime = 1, nTimeSteps

  wetStorageExpense(countTime) = 0.0
  caskConditioningExpense(countTime) = 0.0
  dryStorageExpense(countTime) = 0.0
```

```
      transportationExpense(countTime) = 0.0
      interimStorageExpense(countTime) = 0.0
      repositoryConditioningExpense(countTime) = 0.0
      repositoryExpense(countTime) = 0.0

      call sumShutdownSFPWithFuel(countTime)
      !write(*,*) initialTime+(countTime-1)*timeStep, countShutdownSFPWithFuel(countTime)
      wetStorageExpense(countTime) =
countShutdownSFPWithFuel(countTime)*timeStep*shutdownAnnualWetStorageCost

      call sumFlowFromSFP(countTime)
      caskConditioningExpense(countTime) =
totalSFPDischargeRate(countTime)*timeStep*caskConditioningCost*1000
      ! timeStep is very important variable to include here!!! expected time step is 1 year so it would not
matter, but if alternative time step is used, the value must adjust the rates...
      ! factor of 1000 is needed because conditioning cost is in $/kg but discharge rate is in MT/year

      call sumDSC(countTime)
      dryStorageExpense(countTime) = countNewDSC(countTime)*initialOnSiteDryStorageCost +
countOnlineDSC(countTime)*timeStep*onlineAnnualDryStorageCost +
countShutdownDSC(countTime)*timeStep*shutdownAnnualDryStorageCost
      !write(*,*) dryStorageExpense(countTime)

      do countOriginRegion = 1, nRegions
         do countDestinationRegion = 1, nRegions
            call sumTransportationFlow(countOriginRegion,countDestinationRegion,countTime)
            transportationExpense(countTime) = transportationExpense(countTime) +
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime)*timeStep*transportationCost(cou
ntOriginRegion,countDestinationRegion)*1000
         enddo ! countDestinationRegion = 1, nRegions
      enddo ! countOriginRegion = 1, nRegions

      call sumInterimStorage(countTime)
      interimStorageExpense(countTime) = countUsedIS(countTime)*timeStep*interimStorageAnnualBaseCost +
countLoadingIS(countTime)*timeStep*interimStorageAnnualLoadingCost +
totalStockInInterimStorage(countTime)*timeStep*interimStorageProrateCost*1000

      ! sumFlowIntoRepositories already exists and can be used to determine repository conditioning costs
      repositoryConditioningExpense(countTime) =
totalFlowIntoRepositories(countTime)*timeStep*repositoryConditioningPackaging*1000
```

```
      call sumRepository(countTime)
      repositoryExpense(countTime) = countUsedRepository(countTime)*timeStep*repositoryAnnualBaseCost +
countLoadingRepository(countTime)*timeStep*repositoryAnnualLoadingCost

    enddo ! countTime = 1, nTimeSteps

    ! economics part b: initial capital cost for interim storage and repository

    do countInterimStorageSite = 1, nInterimStorageSites
      do countTime = nint((interimStorageStartTime(countInterimStorageSite) -
interimStorageConstructionTime(countInterimStorageSite) - initialTime)/timeStep + 1),
nint((interimStorageStartTime(countInterimStorageSite) - initialTime)/timeStep)
        interimStorageExpense(countTime) = interimStorageExpense(countTime) +
amortizedISCapital(countInterimStorageSite)
      enddo ! countTime = ...
      ! find the time the IS site shuts down
      interimStorageShutdownTimeStep = nTimeSteps+1
      do countTime = 2, nTimeSteps
        if (fuelInInterimStorage(countInterimStorageSite,countTime).lt.0.01 .and.
fuelInInterimStorage(countInterimStorageSite,countTime-1).gt.0.01) interimStorageShutdownTimeStep =
countTime
      enddo ! countTime = 1, nTimeSteps
      ! note, this do loop will crash if end of DD amortization is after simulation period is supposed to
end...
      do countTime = interimStorageShutdownTimeStep, min(nTimeSteps,nint(interimStorageShutdownTimeStep +
(interimStorageDDTime(countInterimStorageSite)/timeStep) - 1))
        interimStorageExpense(countTime) = interimStorageExpense(countTime) +
amortizedISDD(countInterimStorageSite)
      enddo ! countTime = ...
    enddo ! countInterimStorageSite = 1, nInterimStorageSites

    do countRepository = 1, nRepositories
      do countRepositoryExpansion = 1, nRepositoryExpansions(countRepository)
        do countTime = nint((repositoryExpansionTimes(countRepository,countRepositoryExpansion) -
repositoryConstructionTime(countRepository,countRepositoryExpansion) - initialTime)/timeStep)+1,
nint((repositoryExpansionTimes(countRepository,countRepositoryExpansion) - initialTime)/timeStep)
          repositoryExpense(countTime) = repositoryExpense(countTime) +
amortizedRepositoryCapital(countRepository,countRepositoryExpansion)
        enddo ! countTime = ...
      enddo ! countRepositoryExpansion = 1, nRepositoryExpansions(countRepository)
      repositoryShutdownTimeStep = nTimeSteps+1
```

```
      do countTime = nTimeSteps, 2, -1
         if (totalStockInReactors(countTime).lt.0.01 .and. totalStockInSpentFuelPools(countTime).lt.0.01
.and. totalStockInDryStorage(countTime).lt.0.01 .and. totalStockInInterimStorage(countTime).lt.0.01)
repositoryShutdownTimeStep = countTime
      enddo ! countTime = nTimeSteps, 2, -1
      do countTime = repositoryShutdownTimeStep, min(nTimeSteps,nint(repositoryShutdownTimeStep +
(repositoryClosureTime(countRepository)/timeStep) - 1))
         repositoryExpense(countTime) = repositoryExpense(countTime) +
amortizedRepositoryClosure(countRepository)
      enddo ! countTime = ...
    enddo ! countRepository = 1, nRepositories


   call printResults

endprogram SNuFManager36
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: calculateFlows
! DESCRIPTION: calculates the new flows based on current conditions
! note: this is where the meat of the decision and logic functions occur
! note: the basic flows are assumed to follow a logical progression shown
!       below in steps 1 - 6
!       changing this order will be somewhat difficult, but it is doable
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine calculateFlows(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime

   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor
   integer :: countRepository
   integer :: countInterimStorageSites

   integer :: countRemainingTimeSteps
   integer :: remainingTimeSteps

   integer :: countPassedTimeSteps
   integer :: passedTimeSteps

   integer :: countAvailableDischarges
   integer :: availableDischarges

   integer :: countFuelInDryStorage

   double precision :: maxRequiredRepositoryFillRate ! used to skip steps in logic and save time
   double precision :: incrementalSpentFuelPoolToRepositoryRate

   double precision :: maxFuelInDryStorage

   integer :: maxAvailableInterimStorageLoadings
   integer :: countAvailableInterimStorageLoadings

   double precision :: incrementalInterimStorageToRepositoryRate
```

```
   double precision :: maxInterimStorageFillRate

   double precision :: incrementalSpentFuelPoolToInterimStorageRate

   ! basic logic to flow pattern:
   ! (1) deal with flow into reactors
   ! (2) deal with flows out of reactors
   ! (3) deal with flows from hot spent fuel to cold spent fuel
   ! ... now pause and work from the repository backward... whatever is leftover will end up in dry storage
casks
   ! (4) fill availabile repository space (or repositoryMaxLoadingRate) with cold spent fuel & DSCs
   !    (a) look for linked reactors that are shutting down in 10 years or less
   !    (b) take spent fuel from linked online reactors closest to shutdown to furthest from shutdown in
the amount of SFPloading/(1+time-shutdownTime)
   !    (c) take fuel from linked shutdown reactor SFP of longest shutdown reactor to most recent
   !    (d) take fuel from linked shutdown reactor DSC of longest shutdown reactor to most recent
   !    (e) take fuel from linked online reactor SFP of lowest available space... measured as
reactorDischargeRate/(SFPcapacity - SFPloading)... in the amount of 4*reactorDischargeRate... factor of 4
is ARBITRARY
   !    (f) take fuel from linked online reactor DSC of lowest fuelInDSC
   !    (g) take fuel from linked IS of lowest available space... measured as maxFillRate/(IScapacity-
ISloading)
   !    * note: "linked" refers to the fact that if we run a 4 repository, 4 interim storage strategy,
repository 1 can only be filled from IS 1 and from reactors in region 1
   ! (5) fill available interim storage space (or ISmaxLoadRate) with cold spent fuel & DSCs
   !    same logic as repository fill strategy, except last part (no IS to IS loading)
   ! (6) fill DSCs
   !    (a) if all reactors on site have been shutdown for at least SFPCoolingTime, because then all fuel
should have been discharged and ready to go
   !    (b) if SFP has available space <= reactorDischargeRate/(SFPcapacity - SFPloading) in amount of
reactorDischargeRate
   !  otherwise, allow spent fuel to accumulate in online SFP
   ! end of basic logic to flow pattern

   ! logic steps (1), (2), and (3)...
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)

         totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime) = 0.0
         passedTimeSteps = countTime
         do countReactor = 1, nReactors(countRegion,countReactorSite)
```

```
            passedTimeSteps = min(passedTimeSteps,countTime-
nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-initialTime)/timeStep+1))
            if
(initialTime+countTime*timeStep.gt.reactorShutdownTime(countRegion,countReactorSite,countReactor)) then !
if reactor is shutdown
               reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime) = 0.0
               reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime) =
fuelInReactor(countRegion,countReactorSite,countReactor,countTime)/timeStep
            else ! NOTE: in order to make code more general such that it can add reactors after
initialTime, must add additional logic to previous if statement to search if reactor is shutdown or if time
is before reactorStartTime and correct following line for first reactorLoadingRate
               reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime) =
reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime-1)
               reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime) =
reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime-1)
            endif !
reactorShutdownTime(countRegion,countReactorSite,countReactor).lt.initialTime+(countTime-1)*timeStep
            totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime) =
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime) +
reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime)

         enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)

         ! find the reactor site's spent fuel pool cooling rate
         if (countTime*timeStep.gt.spentFuelPoolCoolingTime) then ! if we have left the initial dynamic
equilibrium phase
            spentFuelPoolCoolingRate(countRegion,countReactorSite,countTime) =
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime-
nint(spentFuelPoolCoolingTime/timeStep))
         else
            spentFuelPoolCoolingRate(countRegion,countReactorSite,countTime) =
spentFuelPoolCoolingRate(countRegion,countReactorSite,1)
         endif ! spentFuelPoolCoolingTime.lt.countTime*timeStep

         ! find the required spent fuel pool discharge rate for each site
         !if (passedTimeSteps.gt.-1) then ! if all reactors on site are shutdown...
         if (passedTimeSteps.gt.0) then ! if all reactors on site are shutdown...
            requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/timeStep
```

```
         elseif (spentFuelPoolCapacity(countRegion,countReactorSite)-
fuelInSpentFuelPool(countRegion,countReactorSite,countTime).lt.totalSiteReactorDischargeRate(countRegion,co
untReactorSite,countTime)*timeStep) then
            requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime)
         else
            requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) = 0.0
         endif !
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime).gt.(spentFuelPoolCapacity(countRegion
,countReactorSite)-fuelInSpentFuelPool(countRegion,countReactorSite,countTime))/timeStep
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   ! logic step (4)... determine the requiredRepositoryFillRate
   do countRepository = 1, nRepositories
      ! if the repository is not yet open
      if (countTime-1.lt.(repositoryExpansionTimes(countRepository,1)-initialTime)/timeStep) then
      ! may be more appropriate to say "if countTime.lt...", but it depends on how you interpret repository
opening time
         requiredRepositoryFillRate(countRepository) = 0.0
      else
         requiredRepositoryFillRate(countRepository) =
min(repositoryMaxFillRate(countRepository,countTime),(repositoryCapacity(countRepository,countTime)-
fuelInRepository(countRepository,countTime))/timeStep)
      endif ! countTime.lt.(repositoryExpansionTimes(countRepository)-initialTime)/timeStep
   enddo ! countRepository = 1, nRepositories

!   ! logic step (4) ... parts (a) and (b) ... take fuel from spent fuel pools of reactors soon to shutdown
!   do countRemainingTimeSteps = 1, ceiling(10/timeStep) ! 1 time step to 10 years
!      do countRegion = 1, nRegions
!         do countReactorSite = 1, nReactorSites(countRegion)
!            remainingTimeSteps = 0
!            do countReactor = 1, nReactors(countRegion,countReactorSite)
!               remainingTimeSteps =
max(remainingTimeSteps,nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-
initialTime)/timeStep+1)-countTime)
!            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
!            if (remainingTimeSteps.eq.countRemainingTimeSteps) then
!               if (nRepositories.eq.nRegions) then ! note how this would be true even if we forced just
one region
!                  if (requiredRepositoryFillRate(countRegion).gt.0.0) then
```

```
!                     spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(countRegion),coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
/(countRemainingTimeSteps*timeStep))
!                     ! noticed the "min" statement to make sure we are not sending more than the
repository can handle
!                     requiredRepositoryFillRate(countRegion) = requiredRepositoryFillRate(countRegion) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime)
!                     requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
!                     ! notice that we must also update the required spentFuelPoolDischargeRate
!                   endif ! requiredRepositoryFillRate(countRegion).gt.0.0
!                 elseif (nRepositories.eq.1 .and. requiredRepositoryFillRate(1).gt.0.0) then ! this is only
going to be called if the number of repositories is one and the number of regions is greater than one
!                   spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(1),coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/(countRem
ainingTimeSteps*timeStep))
!                   requiredRepositoryFillRate(1) = requiredRepositoryFillRate(1) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime)
!                   requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
!                 endif ! nRepositories.eq.nRegions .and. requiredRepositoryFillRate(countRegion).gt.0.0
!               endif ! remainingTimeSteps.eq.countRemainingTimeSteps
!             enddo ! countReactorSite = 1, nReactorSites
!           enddo ! countRegion = 1, nRegions
!         enddo ! countReamainingTimeSteps = 1, ceiling(10/timeStep)
!
!         ! check to see if we should jump to logic step (5) because repository fill rate is already maxed out?
!         maxRequiredRepositoryFillRate = 0.0
!         do countRepository = 1, nRepositories
!           maxRequiredRepositoryFillRate =
max(maxRequiredRepositoryFillRate,requiredRepositoryFillRate(countRepository))
!         enddo ! countRepository = 1, nRepositories
!         if (maxRequiredRepositoryFillRate.eq.0.0) goto 5
!
        ! logic step (4) ... part (c) ... take fuel from spent fuel pools of shutdown reactors
        do countPassedTimeSteps = countTime-1, 0, -1
        ! this is a "brute force" way to find the oldest shutdown reactor
          do countRegion = 1, nRegions
            do countReactorSite = 1, nReactorSites(countRegion)
```

```
            passedTimeSteps = countTime
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               passedTimeSteps = min(passedTimeSteps,countTime-
nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-initialTime)/timeStep+1))
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (passedTimeSteps.eq.countPassedTimeSteps) then
               if (nRepositories.eq.nRegions) then
                  if (requiredRepositoryFillRate(countRegion).gt.0.0) then
                     spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(countRegion),coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
/timeStep)
                     requiredRepositoryFillRate(countRegion) = max(requiredRepositoryFillRate(countRegion)
- spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
                     requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
                  endif ! requiredRepositoryFillRate(countRegion).gt.0.0
               elseif (nRepositories.eq.1 .and. requiredRepositoryFillRate(1).gt.0.0) then
                  spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(1),coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/timeStep)
                  requiredRepositoryFillRate(1) = max(requiredRepositoryFillRate(1) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
                  requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
               endif ! nRepositories.eq.nRegions .and. requiredRepositoryFillRate(countRegion).gt.0.0
            endif ! passedTimeSteps.eq.countPassedTimeSteps
         enddo ! countReactorSite = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
   enddo ! countPassedTimeSteps = countTime-1, 0, -1

   ! check to see if we should jump to logic step (5) because repository fill rate is already maxed out?
   maxRequiredRepositoryFillRate = 0.0
   do countRepository = 1, nRepositories
      maxRequiredRepositoryFillRate =
max(maxRequiredRepositoryFillRate,requiredRepositoryFillRate(countRepository))
   enddo ! countRepository = 1, nRepositories
   if (maxRequiredRepositoryFillRate.eq.0.0) goto 5

   ! logic step (4) ... part (d) ... takes fuel from dry storage casks of shutdown reactors
   do countPassedTimeSteps = countTime-1, 0, -1
```

```
   ! this is a "brute force" way to find the oldest shutdown reactor
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            passedTimeSteps = countTime
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               passedTimeSteps = min(passedTimeSteps,countTime-
nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-initialTime)/timeStep+1))
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (passedTimeSteps.eq.countPassedTimeSteps) then
               if (nRepositories.eq.nRegions) then
                  if (requiredRepositoryFillRate(countRegion).gt.0.0) then
                     dryStorageToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(countRegion),fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/t
imeStep)
                     requiredRepositoryFillRate(countRegion) = requiredRepositoryFillRate(countRegion) -
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
                  endif ! requiredRepositoryFillRate(countRegion).gt.0.0
               elseif (nRepositories.eq.1 .and. requiredRepositoryFillRate(1).gt.0.0) then
                  dryStorageToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(1),fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/timeStep)
                  requiredRepositoryFillRate(1) = requiredRepositoryFillRate(1) -
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
               endif ! nRepositories.eq.nRegions .and. requiredRepositoryFillRate(countRegion).gt.0.0
            endif ! passedTimeSteps.eq.countPassedTimeSteps
         enddo ! countReactorSite = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
   enddo ! countPassedTimeSteps = countTime-1, 0, -1

   ! check to see if we should jump to logic step (5) because repository fill rate is already maxed out?
   maxRequiredRepositoryFillRate = 0.0
   do countRepository = 1, nRepositories
      maxRequiredRepositoryFillRate =
max(maxRequiredRepositoryFillRate,requiredRepositoryFillRate(countRepository))
   enddo ! countRepository = 1, nRepositories
   if (maxRequiredRepositoryFillRate.eq.0.0) goto 5

   ! logic step (4) ... part (e) ... takes fuel from spent fuel pool of online reactors
   do countAvailableDischarges = 0, ceiling(maxSpentFuelPoolDischargeCapacity)
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            remainingTimeSteps = 0
```

```
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               remainingTimeSteps =
max(remainingTimeSteps,nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-
initialTime)/timeStep+1)-countTime)
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            availableDischarges = nint((spentFuelPoolCapacity(countRegion,countReactorSite)-
fuelInSpentFuelPool(countRegion,countReactorSite,countTime))/(totalSiteReactorDischargeRate(countRegion,cou
ntReactorSite,countTime)*timeStep))
            if (availableDischarges.eq.countAvailableDischarges .and. remainingTimeSteps.gt.0) then
            ! the "and" is required in the if statement because we want to make sure we do not alter the
flows from a reactor that shutdown
               if (nRepositories.eq.nRegions) then
                  if (requiredRepositoryFillRate(countRegion).gt.0.0) then
                     spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) +
4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime)
                     spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),requiredRepositoryFillRate(countR
egion))
                     spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),coldFuelInSpentFuelPool(countRegi
on,countReactorSite,countTime)/timeStep)
                     ! note: the factor of 4 is arbitrary, but is supposed to reflect the fact that it is
unlikely we would want to go through the trouble of transporting just one batch worth of fuel
                     ! note: the two minimum statements are required !!!
                     incrementalSpentFuelPoolToRepositoryRate =
min(4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime),requiredRepositoryFillRate(coun
tRegion))
                     incrementalSpentFuelPoolToRepositoryRate =
min(incrementalSpentFuelPoolToRepositoryRate,coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime
)/timeStep)
                     requiredRepositoryFillRate(countRegion) = requiredRepositoryFillRate(countRegion) -
incrementalSpentFuelPoolToRepositoryRate
                     requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
incrementalSpentFuelPoolToRepositoryRate,0.0)
                  endif ! requiredRepositoryFillRate(countRegion).gt.0.0
               elseif (nRepositories.eq.1 .and. requiredRepositoryFillRate(1).gt.0.0) then
                  spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) +
4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime)
```

```
                  spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),requiredRepositoryFillRate(1))
                  spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),coldFuelInSpentFuelPool(countRegi
on,countReactorSite,countTime)/timeStep)
                  incrementalSpentFuelPoolToRepositoryRate =
min(4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime),requiredRepositoryFillRate(1))
                  incrementalSpentFuelPoolToRepositoryRate =
min(incrementalSpentFuelPoolToRepositoryRate,coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime
)/timeStep)
                  requiredRepositoryFillRate(1) = requiredRepositoryFillRate(1) -
incrementalSpentFuelPoolToRepositoryRate
                  requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
incrementalSpentFuelPoolToRepositoryRate,0.0)
               endif ! nRepositories.eq.nRegions .and. requiredRepositoryFillRate(countRegion).gt.0.0
            endif ! availableDischarges.eq.countAvailableDischarges .and. remainingTimeSteps.gt.0
         enddo ! countReactorSite = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
   enddo ! countAvailableDischarges = 0, ceiling(maxSpentFuelPoolDischargeCapacity)

   ! check to see if we should jump to logic step (5) because repository fill rate is already maxed out?
   maxRequiredRepositoryFillRate = 0.0
   do countRepository = 1, nRepositories
      maxRequiredRepositoryFillRate =
max(maxRequiredRepositoryFillRate,requiredRepositoryFillRate(countRepository))
   enddo ! countRepository = 1, nRepositories
   if (maxRequiredRepositoryFillRate.eq.0.0) goto 5

   ! logic step (4) ... part (f) ... takes fuel from dry storage casks of online reactors
   ! first search for maxFuelInDryStorage
   maxFuelInDryStorage = 0.0
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         maxFuelInDryStorage =
max(maxFuelInDryStorage,fuelInDryStorageCasks(countRegion,countReactorSite,countTime))
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   countFuelInDryStorage = 10 ! if maximum fuel in dry storage on any given site is less than 10/2, program
ignores this step
```

```
   do while (countFuelInDryStorage.lt.floor(2*maxFuelInDryStorage)) ! notice the factor of 2 to ensure all
is checked
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            remainingTimeSteps = 0
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               remainingTimeSteps =
max(remainingTimeSteps,nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-
initialTime)/timeStep+1)-countTime)
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (fuelInDryStorageCasks(countRegion,countReactorSite,countTime).lt.countFuelInDryStorage
.and. remainingTimeSteps.gt.0) then
               if (nRepositories.eq.nRegions) then
                  if (requiredRepositoryFillRate(countRegion).gt.0.0) then
                     dryStorageToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(countRegion),fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/t
imeStep)
                     requiredRepositoryFillRate(countRegion) = requiredRepositoryFillRate(countRegion) -
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
                  endif ! requiredRepositoryFillRate(countRegion).gt.0.0
               elseif (nRepositories.eq.1 .and. requiredRepositoryFillRate(1).gt.0.0) then
                  dryStorageToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(1),fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/timeStep)
                  requiredRepositoryFillRate(1) = requiredRepositoryFillRate(1) -
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
               endif ! nRepositories.eq.nRegions .and. requiredRepositoryFillRate(countRegion).gt.0.0
            endif ! fuelInDryStorageCasks(countRegion,countReactorSite,countTime).lt.countFuelInDryStorage
.and. remainingTimeSteps.gt.0
         enddo ! countReactorSite = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
      countFuelInDryStorage = countFuelInDryStorage*2
   enddo ! while (countFuelInDryStorage.lt.maxFuelInDryStorage)

   ! check to see if we should jump to logic step (5) because repository fill rate is already maxed out?
   maxRequiredRepositoryFillRate = 0.0
   do countRepository = 1, nRepositories
      maxRequiredRepositoryFillRate =
max(maxRequiredRepositoryFillRate,requiredRepositoryFillRate(countRepository))
   enddo ! countRepository = 1, nRepositories
   if (maxRequiredRepositoryFillRate.eq.0.0) goto 5
```

```
   ! logic step (4) ... part (g) ... takes fuel from interim storage of lowest available space
   ! first search for maxInterimStorageCapacity
   maxAvailableInterimStorageLoadings = 0
   do countInterimStorageSites = 1, nInterimStorageSites
      maxAvailableInterimStorageLoadings =
max(maxAvailableInterimStorageLoadings,ceiling(interimStorageCapacity(countInterimStorageSites)/(interimSto
rageMaxFillRate(countInterimStorageSites)*timeStep)))
   enddo ! countInterimStorageSites = 1, nInterimStorageSites

   do countAvailableInterimStorageLoadings = 0, maxAvailableInterimStorageLoadings
      do countInterimStorageSites = 1, nInterimStorageSites
         if (nint((interimStorageCapacity(countInterimStorageSites)-
fuelInInterimStorage(countInterimStorageSites,countTime))/(interimStorageMaxFillRate(countInterimStorageSit
es)*timeStep)).eq.countAvailableInterimStorageLoadings) then
            if (nRepositories.eq.nInterimStorageSites) then ! therefore number of interim storage sites
also equals the number of regions
               if (requiredRepositoryFillRate(countInterimStorageSites).gt.0.0) then
                  interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
interimStorageToRepositoryRate(countInterimStorageSites,countTime) +
interimStorageMaxFillRate(countInterimStorageSites)
                  interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),requiredRepositoryFillRate(countInte
rimStorageSites))
                  interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),fuelInInterimStorage(countInterimSto
rageSites,countTime)/timeStep)
                  incrementalInterimStorageToRepositoryRate =
min(interimStorageMaxFillRate(countInterimStorageSites),requiredRepositoryFillRate(countInterimStorageSites
))
                  incrementalInterimStorageToRepositoryRate =
min(interimStorageMaxFillRate(countInterimStorageSites),fuelInInterimStorage(countInterimStorageSites,count
Time)/timeStep)
                  requiredRepositoryFillRate(countInterimStorageSites) =
requiredRepositoryFillRate(countInterimStorageSites) - incrementalInterimStorageToRepositoryRate
               endif ! requiredRepositoryFillRate(countInterimStorageSites).gt.0.0
            elseif (nRepositories.eq.1 .and. nInterimStorageSites.eq.nRegions .and.
requiredRepositoryFillRate(1).gt.0.0) then
               interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
interimStorageToRepositoryRate(countInterimStorageSites,countTime) +
interimStorageMaxFillRate(countInterimStorageSites)
```

```
                interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),requiredRepositoryFillRate(1))
                interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),fuelInInterimStorage(countInterimSto
rageSites,countTime)/timeStep)
                incrementalInterimStorageToRepositoryRate =
min(interimStorageMaxFillRate(countInterimStorageSites),requiredRepositoryFillRate(1))
                incrementalInterimStorageToRepositoryRate =
min(interimStorageMaxFillRate(countInterimStorageSites),fuelInInterimStorage(countInterimStorageSites,count
Time)/timeStep)
                requiredRepositoryFillRate(1) = requiredRepositoryFillRate(1) -
incrementalInterimStorageToRepositoryRate
             endif ! nRepositories.eq.nInterimStorageSites
          endif ! nint((interimStorageCapacity(countInterimStorageSites)-
fuelInInterimStorage(countInterimStorageSites,countTime))/(interimStorageMaxFillRate(countInterimStorageSit
es)*timeStep)).eq.countAvailableInterimStorageLoadings
       enddo ! countInterimStorageSites = 1, nInterimStorageSites
    enddo ! countAvailableInterimStorageLoadings = 0, maxAvailableInterimStorageLoadings

5   continue ! just an jump ahead point to get to logic step 5 if repository fill rate is already maxed out

    ! logic step (5)... determine the interimStorageFillRate
    do countInterimStorageSites = 1, nInterimStorageSites
       ! if the interim storage is not yet open
       if (countTime-1.lt.(interimStorageStartTime(countInterimStorageSites)-initialTime)/timeStep) then
          interimStorageFillRate(countInterimStorageSites) = 0.0
       else !if (nRepositories.eq.nInterimStorageSites) then
          interimStorageFillRate(countInterimStorageSites) =
min(interimStorageMaxFillRate(countInterimStorageSites) +
interimStorageToRepositoryRate(countInterimStorageSites,countTime),interimStorageToRepositoryRate(countInte
rimStorageSites,countTime) + (interimStorageCapacity(countInterimStorageSites)-
fuelInInterimStorage(countInterimStorageSites,countTime))/timeStep)
       endif ! countTime.lt.(repositoryExpansionTimes(countRepository)-initialTime)/timeStep
    enddo ! countInterimStorageSites = 1, nInterimStorageSites

!   ! logic step (5) ... parts (a) and (b) ... take fuel from spent fuel pools of reactors soon to shutdown
!   do countRemainingTimeSteps = 1, ceiling(10/timeStep) ! 1 time step to 10 years
!      do countRegion = 1, nRegions
!         do countReactorSite = 1, nReactorSites(countRegion)
!            remainingTimeSteps = 0
!            do countReactor = 1, nReactors(countRegion,countReactorSite)
```

```
!                remainingTimeSteps =
max(remainingTimeSteps,nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-
initialTime)/timeStep+1)-countTime)
!             enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
!             if (remainingTimeSteps.eq.countRemainingTimeSteps) then
!                if (nInterimStorageSites.eq.nRegions) then
!                   if (interimStorageFillRate(countRegion).gt.0.0) then
!                      spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(countRegion),coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/(co
untRemainingTimeSteps*timeStep))
!                      interimStorageFillRate(countRegion) = interimStorageFillRate(countRegion) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime)
!                      requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
!                   endif ! interimStorageFillRate(countRegion).gt.0.0
!                elseif (nInterimStorageSites.eq.1 .and. interimStorageFillRate(1).gt.0.0) then
!                   spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(1),coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/(countRemaini
ngTimeSteps*timeStep))
!                   interimStorageFillRate(1) = interimStorageFillRate(1) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime)
!                   requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
!                endif ! nInterimStorageSites.eq.nRegions .and. interimStorageFillRate(countRegion).gt.0.0
!             endif ! remainingTimeSteps.eq.countRemainingTimeSteps
!          enddo ! countReactorSite = 1, nReactorSites(countRegion)
!       enddo ! countRegion = 1, nRegions
!    enddo ! countRemainingTimeSteps = 1, ceiling(10/timeStep)
!
!    ! check to see if we should jump to logic step (6) because interim storage fill rate is already maxed
out?
!    maxInterimStorageFillRate = 0.0
!    do countInterimStorageSites = 1, nInterimStorageSites
!       maxInterimStorageFillRate =
max(maxInterimStorageFillRate,interimStorageFillRate(countInterimStorageSites))
!    enddo ! countInterimStorageSites = 1, nInterimStorageSites
!    if (maxInterimStorageFillRate.eq.0.0) goto 6

   ! logic step (5) ... part (c) ... take fuel from spent fuel pools of shutdown reactors
```

```
   do countPassedTimeSteps = countTime-1, 0, -1
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            passedTimeSteps = countTime
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               passedTimeSteps = min(passedTimeSteps,countTime-
nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-initialTime)/timeStep+1))
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (passedTimeSteps.eq.countPassedTimeSteps) then
               if (nInterimStorageSites.eq.nRegions) then
                  if (interimStorageFillRate(countRegion).gt.0.0) then
                     spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(countRegion),(coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/ti
meStep)-spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                     !spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
max(spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
                     interimStorageFillRate(countRegion) = interimStorageFillRate(countRegion) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime)
                     requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
                  endif ! interimStorageFillRate(countRegion).gt.0.0
               elseif (nInterimStorageSites.eq.1 .and. interimStorageFillRate(1).gt.0.0) then
                  spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(1),(coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/timeStep)-
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                  !spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
max(spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
                  interimStorageFillRate(1) = interimStorageFillRate(1) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime)
                  requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
               endif ! nInterimStorageSites.eq.nRegions .and. interimStorageFillRate(countRegion).gt.0.0
            endif ! passedTimeSteps.eq.countPassedTimeSteps
         enddo ! countReactorSites = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
   enddo ! countPassedTimeSteps = countTime-1, 0, -1

   ! check to see if we should jump to logic step (6) because interim storage fill rate is already maxed
out?
```

```
   maxInterimStorageFillRate = 0.0
   do countInterimStorageSites = 1, nInterimStorageSites
      maxInterimStorageFillRate =
max(maxInterimStorageFillRate,interimStorageFillRate(countInterimStorageSites))
   enddo ! countInterimStorageSites = 1, nInterimStorageSites
   if (maxInterimStorageFillRate.eq.0.0) goto 6

   ! logic step (5) ... part (d) ... takes fuel from dry storage casks of shutdown reactors
   do countPassedTimeSteps = countTime-1, 0, -1
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            passedTimeSteps = countTime
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               passedTimeSteps = min(passedTimeSteps,countTime-
nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-initialTime)/timeStep+1))
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (passedTimeSteps.eq.countPassedTimeSteps) then
               if (nInterimStorageSites.eq.nRegions) then
                  if (interimStorageFillRate(countRegion).gt.0.0) then
                     dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(countRegion),(fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/time
Step)-dryStorageToRepositoryRate(countRegion,countReactorSite,countTime))
                     interimStorageFillRate(countRegion) = interimStorageFillRate(countRegion) -
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
                  endif ! interimStorageFillRate(countRegion).gt.0.0
               elseif (nInterimStorageSites.eq.1 .and. interimStorageFillRate(1).gt.0.0) then
                  dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(1),(fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/timeStep)-
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime))
                  interimStorageFillRate(1) = interimStorageFillRate(1) -
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
               endif ! nInterimStorageSites.eq.nRegions .and. interimStorageFillRate(countRegion).gt.0.0
            endif ! passedTimeSteps.eq.countPassedTimeSteps
         enddo ! countReactorSites = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
   enddo ! countPassedTimeSteps = countTime-1, 0, -1

   ! check to see if we should jump to logic step (6) because interim storage fill rate is already maxed
out?
   maxInterimStorageFillRate = 0.0
   do countInterimStorageSites = 1, nInterimStorageSites
```

```
      maxInterimStorageFillRate =
max(maxInterimStorageFillRate,interimStorageFillRate(countInterimStorageSites))
   enddo ! countInterimStorageSites = 1, nInterimStorageSites
   if (maxInterimStorageFillRate.eq.0.0) goto 6

   ! logic step (5) ... part (e) ... takes fuel from spent fuel pool of online reactors
   do countAvailableDischarges = 0, ceiling(maxSpentFuelPoolDischargeCapacity)
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            remainingTimeSteps = 0
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               remainingTimeSteps =
max(remainingTimeSteps,nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-
initialTime)/timeStep+1)-countTime)
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            availableDischarges = nint((spentFuelPoolCapacity(countRegion,countReactorSite)-
fuelInSpentFuelPool(countRegion,countReactorSite,countTime))/(totalSiteReactorDischargeRate(countRegion,cou
ntReactorSite,countTime)*timeStep))
            if (availableDischarges.eq.countAvailableDischarges .and. remainingTimeSteps.gt.0) then
               if (nInterimStorageSites.eq.nRegions) then
                  if (interimStorageFillRate(countRegion).gt.0.0) then
                     spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) +
4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime)
                     spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),interimStorageFillRate(countR
egion))
                     spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),(coldFuelInSpentFuelPool(coun
tRegion,countReactorSite,countTime)/timeStep)-
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                     incrementalSpentFuelPoolToInterimStorageRate =
min(4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime),interimStorageFillRate(countReg
ion))
                     incrementalSpentFuelPoolToInterimStorageRate =
min(incrementalSpentFuelPoolToInterimStorageRate,(coldFuelInSpentFuelPool(countRegion,countReactorSite,coun
tTime)/timeStep)-spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                     interimStorageFillRate(countRegion) = interimStorageFillRate(countRegion) -
incrementalSpentFuelPoolToInterimStorageRate
```

```
                    requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
incrementalSpentFuelPoolToRepositoryRate,0.0)
                  endif ! interimStorageFillRate(countRegion).gt.0.0
                elseif (nInterimStorageSites.eq.1 .and. interimStorageFillRate(1).gt.0.0) then
                  spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) +
4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime)
                  spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),interimStorageFillRate(1))
                  spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),(coldFuelInSpentFuelPool(coun
tRegion,countReactorSite,countTime)/timeStep)-
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                  incrementalSpentFuelPoolToInterimStorageRate =
min(4*totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime),interimStorageFillRate(1))
                  incrementalSpentFuelPoolToInterimStorageRate =
min(incrementalSpentFuelPoolToInterimStorageRate,(coldFuelInSpentFuelPool(countRegion,countReactorSite,coun
tTime)/timeStep)-spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                  interimStorageFillRate(1) = interimStorageFillRate(1) -
incrementalSpentFuelPoolToInterimStorageRate
                  requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
incrementalSpentFuelPoolToRepositoryRate,0.0)
                endif ! nInterimStorageSites.eq.nRegions .and. interimStorageFillRate(countRegion).gt.0.0
              endif ! availableDischarges.eq.countAvailableDischarges .and. remainingTimeSteps.gt.0
            enddo ! countReactorSites = 1, nReactorSites(countRegion)
          enddo ! countRegion = 1, nRegions
        enddo ! countAvailableDischarges = 0, ceiling(maxSpentFuelPoolDischargeCapacity)

        ! check to see if we should jump to logic step (6) because interim storage fill rate is already maxed
out?
        maxInterimStorageFillRate = 0.0
        do countInterimStorageSites = 1, nInterimStorageSites
          maxInterimStorageFillRate =
max(maxInterimStorageFillRate,interimStorageFillRate(countInterimStorageSites))
        enddo ! countInterimStorageSites = 1, nInterimStorageSites
        if (maxInterimStorageFillRate.eq.0.0) goto 6

        ! logic step (5) ... part (f) ... takes fuel from dry storage casks of online reactors
        ! first search for maxFuelInDryStorage
```

```
   maxFuelInDryStorage = 0.0
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         maxFuelInDryStorage =
max(maxFuelInDryStorage,fuelInDryStorageCasks(countRegion,countReactorSite,countTime))
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   countFuelInDryStorage = 10 ! if maximum fuel in dry storage on any given site is less than 10/2, program
ignores this step
   do while (countFuelInDryStorage.lt.floor(2*maxFuelInDryStorage)) ! notice the factor of 2 to ensure all
is checked
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            remainingTimeSteps = 0
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               remainingTimeSteps =
max(remainingTimeSteps,nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-
initialTime)/timeStep+1)-countTime)
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (fuelInDryStorageCasks(countRegion,countReactorSite,countTime).lt.countFuelInDryStorage
.and. remainingTimeSteps.gt.0) then
               if (nInterimStorageSites.eq.nRegions) then
                  if (interimStorageFillRate(countRegion).gt.0.0) then
                     dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(countRegion),(fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/time
Step)-dryStorageToRepositoryRate(countRegion,countReactorSite,countTime))
                     interimStorageFillRate(countRegion) = interimStorageFillRate(countRegion) -
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
                  endif ! interimStorageFillRate(countRegion).gt.0.0
               elseif (nInterimStorageSites.eq.1 .and. interimStorageFillRate(1).gt.0.0) then
                  dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(1),(fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/timeStep)-
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime))
                  interimStorageFillRate(1) = interimStorageFillRate(1) -
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
               endif ! nInterimStorageSites.eq.nRegions
            endif ! fuelInDryStorageCasks(countRegion,countReactorSite,countTime).lt.countFuelInDryStorage
.and. remainingTimeSteps.gt.0
         enddo ! countReactorSites = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
```

```
   countFuelInDryStorage = countFuelInDryStorage*2
   enddo ! while (countFuelInDryStorage.lt.maxFuelInDryStorage)

6  continue ! just an jump ahead point to get to logic step 6 if interim storage fill rate is already maxed
out

   ! logic step (6) ... sends the reamaining requiredSpentFuelPoolDischargeRate to dry storage
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         spentFuelPoolToDryStorageRate(countRegion,countReactorSite,countTime) =
requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite)
         requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) = 0.0
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

endsubroutine calculateFlows
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: calculateSimpleFlows
! DESCRIPTION: calculates the new flows based on current conditions
! simplified program based entirely on shutdown time of reactor
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine calculateSimpleFlows(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime

   integer :: passedTimeSteps

   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor

   double precision :: maxShutdownTime
   double precision :: minShutdownTime

   integer :: countTempTime

   double precision :: minSiteShutdownTime

   integer :: countRepository
   integer :: countInterimStorageSites

   ! basic logic to flow pattern:
   ! (1) deal with flow into reactors
   ! (2) deal with flows out of reactors
   ! (3) deal with flows from hot spent fuel to cold spent fuel
   ! ... now pause and work from the repository backward... whatever is leftover will end up in dry storage
casks
   ! (4) fill availabile repository space (or repositoryMaxLoadingRate) with cold spent fuel & DSCs
   !    (a) look for linked reactors that are shutting down in 10 years or less
   !    (b) take spent fuel from linked online reactors closest to shutdown to furthest from shutdown in
the amount of SFPloading/(1+time-shutdownTime)
   !    (c) take fuel from linked shutdown reactor SFP of longest shutdown reactor to most recent
   !    (d) take fuel from linked shutdown reactor DSC of longest shutdown reactor to most recent
```

```
   !     (e) take fuel from linked online reactor SFP of lowest available space... measured as
reactorDischargeRate/(SFPcapacity - SFPloading)... in the amount of 4*reactorDischargeRate... factor of 4
is ARBITRARY
   !     (f) take fuel from linked online reactor DSC of lowest fuelInDSC
   !     (g) take fuel from linked IS of lowest available space... measured as maxFillRate/(IScapacity-
ISloading)
   !     * note: "linked" refers to the fact that if we run a 4 repository, 4 interim storage strategy,
repository 1 can only be filled from IS 1 and from reactors in region 1
   ! (5) fill available interim storage space (or ISmaxLoadRate) with cold spent fuel & DSCs
   !     same logic as repository fill strategy, except last part (no IS to IS loading)
   ! (6) fill DSCs
   !     (a) if all reactors on site have been shutdown for at least SFPCoolingTime, because then all fuel
should have been discharged and ready to go
   !     (b) if SFP has available space <= reactorDischargeRate/(SFPcapacity - SFPloading) in amount of
reactorDischargeRate
   !  otherwise, allow spent fuel to accumulate in online SFP
   ! end of basic logic to flow pattern

   ! logic steps (1), (2), and (3)...
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)

         totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime) = 0.0
         passedTimeSteps = countTime
         do countReactor = 1, nReactors(countRegion,countReactorSite)
            passedTimeSteps = min(passedTimeSteps,countTime-
nint((reactorShutdownTime(countRegion,countReactorSite,countReactor)-initialTime)/timeStep+1))
            if
(initialTime+countTime*timeStep.gt.reactorShutdownTime(countRegion,countReactorSite,countReactor)) then !
if reactor is shutdown
               reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime) = 0.0
               reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime) =
fuelInReactor(countRegion,countReactorSite,countReactor,countTime)/timeStep
            else ! NOTE: in order to make code more general such that it can add reactors after
initialTime, must add additional logic to previous if statement to search if reactor is shutdown or if time
is before reactorStartTime and correct following line for first reactorLoadingRate
               reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime) =
reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime-1)
               reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime) =
reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime-1)
```

```
            endif !
reactorShutdownTime(countRegion,countReactorSite,countReactor).lt.initialTime+(countTime-1)*timeStep
            totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime) =
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime) +
reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime)

         enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)

         ! find the reactor site's spent fuel pool cooling rate
         if (countTime*timeStep.gt.spentFuelPoolCoolingTime) then ! if we have left the initial dynamic
equilibrium phase
            spentFuelPoolCoolingRate(countRegion,countReactorSite,countTime) =
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime-
nint(spentFuelPoolCoolingTime/timeStep))
         else
            spentFuelPoolCoolingRate(countRegion,countReactorSite,countTime) =
spentFuelPoolCoolingRate(countRegion,countReactorSite,1)
         endif ! spentFuelPoolCoolingTime.lt.countTime*timeStep

         ! find the required spent fuel pool discharge rate for each site
         !if (passedTimeSteps.gt.-1) then ! if all reactors on site are shutdown...
         if (passedTimeSteps.gt.0) then ! if all reactors on site are shutdown...
            requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/timeStep
         elseif (spentFuelPoolCapacity(countRegion,countReactorSite)-
fuelInSpentFuelPool(countRegion,countReactorSite,countTime).lt.totalSiteReactorDischargeRate(countRegion,co
untReactorSite,countTime)*timeStep) then
            requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime)
         else
            requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) = 0.0
         endif !
totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime).gt.(spentFuelPoolCapacity(countRegion
,countReactorSite)-fuelInSpentFuelPool(countRegion,countReactorSite,countTime))/timeStep

      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   ! determine the requiredRepositoryFillRate
   do countRepository = 1, nRepositories
      ! if the repository is not yet open
```

```
      if (countTime-1.lt.(repositoryExpansionTimes(countRepository,1)-initialTime)/timeStep) then
      ! may be more appropriate to say "if countTime.lt...", but it depends on how you interpret repository
opening time
         requiredRepositoryFillRate(countRepository) = 0.0
      else
         requiredRepositoryFillRate(countRepository) =
min(repositoryMaxFillRate(countRepository,countTime),(repositoryCapacity(countRepository,countTime)-
fuelInRepository(countRepository,countTime))/timeStep)
      endif ! countTime.lt.(repositoryExpansionTimes(countRepository)-initialTime)/timeStep
   enddo ! countRepository = 1, nRepositories


   ! determine first and last shutdown date
   maxShutdownTime = 0.0
   minShutdownTime = finalTime
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         do countReactor = 1, nReactors(countRegion,countReactorSite)
            maxShutdownTime =
max(maxShutdownTime,reactorShutdownTime(countRegion,countReactorSite,countReactor))
            minShutdownTime =
min(minShutdownTime,reactorShutdownTime(countRegion,countReactorSite,countReactor))
         enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   ! send fuel from reactor sites to repository
   do countTempTime = floor(minShutdownTime), ceiling(maxShutdownTime)
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            minSiteShutdownTime = finalTime
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               minSiteShutdownTime =
min(minSiteShutdownTime,reactorShutdownTime(countRegion,countReactorSite,countReactor))
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (nint(minSiteShutdownTime).eq.countTempTime) then
               if (nRepositories.eq.nRegions) then
                  if (requiredRepositoryFillRate(countRegion).gt.0.0) then
                     ! send fuel from spent fuel pool to repository
```

```
                    spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) +
requiredRepositoryFillRate(countRegion)
                    spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),
coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/timeStep)
                    requiredRepositoryFillRate(countRegion) = max(requiredRepositoryFillRate(countRegion)
- spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
                    requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
                    ! send fuel from dry storage to repository
                    dryStorageToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(countRegion),fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/t
imeStep)
                    requiredRepositoryFillRate(countRegion) = requiredRepositoryFillRate(countRegion) -
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
                endif ! requiredRepositoryFillRate(countRegion).gt.0.0
              elseif (nRepositories.eq.1 .and. requiredRepositoryFillRate(1).gt.0.0) then
                ! send fuel from spent fuel pool to repository
                spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) + requiredRepositoryFillRate(1)
                spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) =
min(spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),
coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/timeStep)
                requiredRepositoryFillRate(1) = max(requiredRepositoryFillRate(1) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
                requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime),0.0)
                ! send fuel from dry storage to repository
                dryStorageToRepositoryRate(countRegion,countReactorSite,countTime) =
min(requiredRepositoryFillRate(1),fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/timeStep)
                requiredRepositoryFillRate(1) = requiredRepositoryFillRate(1) -
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
              endif ! nRepositories.eq.nRegions
            endif ! reactorShutdownTime(countRegion,countReactorSite,countReactor).le.countTempTime)
          enddo ! countReactorSite = 1, nReactorSites(countRegion)
        enddo ! countRegion = 1, nRegions
      enddo ! countTempTime = floor(minShutdownTime), ceiling(maxShutdownTime)
```

```
   ! send fuel from interim storage to repository
   do countInterimStorageSites = 1, nInterimStorageSites
      if (nRepositories.eq.nInterimStorageSites) then
         if (requiredRepositoryFillRate(countInterimStorageSites).gt.0.0) then
            interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
interimStorageToRepositoryRate(countInterimStorageSites,countTime) +
interimStorageMaxFillRate(countInterimStorageSites)
            interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),requiredRepositoryFillRate(countInte
rimStorageSites))
            interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),fuelInInterimStorage(countInterimSto
rageSites,countTime)/timeStep)
            requiredRepositoryFillRate(countInterimStorageSites) =
requiredRepositoryFillRate(countInterimStorageSites) -
interimStorageToRepositoryRate(countInterimStorageSites,countTime)
         endif ! requiredRepositoryFillRate(countInterimStorageSites).gt.0.0
      elseif (nRepositories.eq.1 .and. nInterimStorageSites.eq.nRegions .and.
requiredRepositoryFillRate(1).gt.0.0) then
         interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
interimStorageToRepositoryRate(countInterimStorageSites,countTime) +
interimStorageMaxFillRate(countInterimStorageSites)
         interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),requiredRepositoryFillRate(1))
         interimStorageToRepositoryRate(countInterimStorageSites,countTime) =
min(interimStorageToRepositoryRate(countInterimStorageSites,countTime),fuelInInterimStorage(countInterimSto
rageSites,countTime)/timeStep)
         requiredRepositoryFillRate(1) = requiredRepositoryFillRate(1) -
interimStorageToRepositoryRate(countInterimStorageSites,countTime)
      endif ! nRepositories.eq.nInterimStorageSites
   enddo ! countInterimStorageSites = 1, nInterimStorageSites

   ! determine the interim storage fill rate
   do countInterimStorageSites = 1, nInterimStorageSites
      ! if the interim storage is not yet open
      if (countTime-1.lt.(interimStorageStartTime(countInterimStorageSites)-initialTime)/timeStep) then
         interimStorageFillRate(countInterimStorageSites) = 0.0
      else !if (nRepositories.eq.nInterimStorageSites) then
         interimStorageFillRate(countInterimStorageSites) =
min(interimStorageMaxFillRate(countInterimStorageSites) +
interimStorageToRepositoryRate(countInterimStorageSites,countTime),interimStorageToRepositoryRate(countInte
```

```
rimStorageSites,countTime) + (interimStorageCapacity(countInterimStorageSites)-
fuelInInterimStorage(countInterimStorageSites,countTime))/timeStep)
      endif ! countTime.lt.(repositoryExpansionTimes(countRepository)-initialTime)/timeStep
   enddo ! countInterimStorageSites = 1, nInterimStorageSites

   ! send fuel from reactor sites to interim storage
   do countTempTime = floor(minShutdownTime), ceiling(maxShutdownTime)
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            minSiteShutdownTime = finalTime
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               minSiteShutdownTime =
min(minSiteShutdownTime,reactorShutdownTime(countRegion,countReactorSite,countReactor))
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            if (nint(minSiteShutdownTime).eq.countTempTime) then
               if (nInterimStorageSites.eq.nRegions) then
                  if (interimStorageFillRate(countRegion).gt.0.0) then
                     ! send fuel from spent fuel pool to interim storage
                     spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(countRegion),(coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/ti
meStep)-spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                     interimStorageFillRate(countRegion) = interimStorageFillRate(countRegion) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime)
                     requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
                     ! send fuel from dry storage to interim storage
                     dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(countRegion),(fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/time
Step)-dryStorageToRepositoryRate(countRegion,countReactorSite,countTime))
                     interimStorageFillRate(countRegion) = interimStorageFillRate(countRegion) -
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
                  endif ! interimStorageFillRate(countRegion).gt.0.0
               elseif (nInterimStorageSites.eq.1 .and. interimStorageFillRate(1).gt.0.0) then
                  ! send fuel from spent fuel pool to interim storage
                  spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(1),(coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)/timeStep)-
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime))
                  interimStorageFillRate(1) = interimStorageFillRate(1) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime)
```

```
                  requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) =
max(requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime),0.0)
                  ! send fuel from dry storage to interim storage
                  dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime) =
min(interimStorageFillRate(1),(fuelInDryStorageCasks(countRegion,countReactorSite,countTime)/timeStep)-
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime))
                  interimStorageFillRate(1) = interimStorageFillRate(1) -
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
               endif ! nRepositories.eq.nRegions
            endif ! reactorShutdownTime(countRegion,countReactorSite,countReactor).le.countTempTime)
         enddo ! countReactorSite = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions
   enddo ! countTempTime = floor(minShutdownTime), ceiling(maxShutdownTime)

   ! send fuel from spent fuel pool to dry storage casks
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         spentFuelPoolToDryStorageRate(countRegion,countReactorSite,countTime) =
requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite)
         requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) = 0.0
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

endsubroutine calculateSimpleFlows
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: calculateStocks
! DESCRIPTION: calculates the new stocks based on last time step's flows
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine calculateStocks(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime

   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor
   integer :: countRepository
   integer :: countInterimStorageSites
   double precision, allocatable :: incrementalRepositoryFillRate(:)
   double precision, allocatable :: incrementalInterimStorageFillRate(:)

   allocate (incrementalRepositoryFillRate(nRepositories))
   allocate (incrementalInterimStorageFillRate(nInterimStorageSites))

   !debug
   !write(*,*) 'countTime', countTime, 'realTime', initialTime+(countTime-1)*timeStep

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         do countReactor = 1, nReactors(countRegion,countReactorSite)
            fuelInReactor(countRegion,countReactorSite,countReactor,countTime) =
fuelInReactor(countRegion,countReactorSite,countReactor,countTime-1) +
(reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime-1) -
reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime-1))*timeStep
            !debug
            !write(*,*) countRegion, countReactorSite, countReactor,
fuelInReactor(countRegion,countReactorSite,countReactor,countTime)
         enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
         hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime) =
hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime-1) +
(totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime-1) -
spentFuelPoolCoolingRate(countRegion,countReactorSite,countTime-1))*timeStep
         !debug
```

```
         !write(*,*) countRegion, countReactorSite, countReactor,
hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
         coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime) =
coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime-1) +
(spentFuelPoolCoolingRate(countRegion,countReactorSite,countTime-1) -
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime-1) -
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime-1) -
spentFuelPoolToDryStorageRate(countRegion,countReactorSite,countTime-1))*timeStep
         fuelInSpentFuelPool(countRegion,countReactorSite,countTime) =
hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime) +
coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
         !debug
         !write(*,*) countRegion, countReactorSite, countReactor,
coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
         fuelInDryStorageCasks(countRegion,countReactorSite,countTime) =
fuelInDryStorageCasks(countRegion,countReactorSite,countTime-1) +
(spentFuelPoolToDryStorageRate(countRegion,countReactorSite,countTime-1) -
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime-1) -
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime-1))*timeStep
         !write(*,*) 'countRegion', countRegion, 'countReactorSite', countReactorSite,
'hotFuelInSpentFuelPool', hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
         !coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime) =
fuelInSpentFuelPool(countRegion,countReactorSite,countTime) -
hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   ! first initialize your stocks to the previous time step values and set incremental fill rates to zero
   do countRepository = 1, nRepositories
      fuelInRepository(countRepository,countTime) = fuelInRepository(countRepository,countTime-1)
      incrementalRepositoryFillRate(countRepository) = 0.0
   enddo ! countRepositories = 1, nRepositories
   do countInterimStorageSites = 1, nInterimStorageSites
      fuelInInterimStorage(countInterimStorageSites,countTime) =
fuelInInterimStorage(countInterimStorageSites,countTime-1)
      incrementalInterimStorageFillRate(countInterimStorageSites) = 0.0
   enddo ! countInterimStorageSites = 1, nInterimStorageSites

   ! now calculate the incremental fill rates of the repositories and interim storage sites by only the
spent fuel pools and dry storage casks
   do countRegion = 1, nRegions
```

```
      do countReactorSite = 1, nReactorSites(countRegion)
      if (nRepositories.eq.nRegions) then ! nInterimStorageSites also equals nRegions
         incrementalRepositoryFillRate(countRegion) = incrementalRepositoryFillRate(countRegion) +
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime-1) +
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime-1)
         incrementalInterimStorageFillRate(countRegion) = incrementalInterimStorageFillRate(countRegion) +
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime-1) +
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime-1)
      elseif (nRepositories.eq.1 .and. nInterimStorageSites.eq.nRegions) then
         incrementalRepositoryFillRate(1) = incrementalRepositoryFillRate(1) +
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime-1) +
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime-1)
         incrementalInterimStorageFillRate(countRegion) = incrementalInterimStorageFillRate(countRegion) +
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime-1) +
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime-1)
      elseif (nRepositories.eq.1 .and. nInterimStorageSites.eq.1) then
         incrementalRepositoryFillRate(1) = incrementalRepositoryFillRate(1) +
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime-1) +
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime-1)
         incrementalInterimStorageFillRate(1) = incrementalInterimStorageFillRate(1) +
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime-1) +
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime-1)
      endif ! nRepositoreis.eq.nRegions
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   ! add to the repository and interim storage the calculated incremental fill rates from spent fuel pools
and dry storage casks
   do countRepository = 1, nRepositories
      fuelInRepository(countRepository,countTime) = fuelInRepository(countRepository,countTime) +
incrementalRepositoryFillRate(countRepository)*timeStep
      incrementalRepositoryFillRate(countRepository) = 0.0
   enddo ! countRepositories = 1, nRepositories
   do countInterimStorageSites = 1, nInterimStorageSites
      fuelInInterimStorage(countInterimStorageSites,countTime) =
fuelInInterimStorage(countInterimStorageSites,countTime) +
incrementalInterimStorageFillRate(countInterimStorageSites)*timeStep
      incrementalInterimStorageFillRate(countInterimStorageSites) = 0.0
   enddo ! countInterimStorageSites = 1, nInterimStorageSites

   ! now determine the amount of fuel that goes into repository from interim storage
```

```
   do countInterimStorageSites = 1, nInterimStorageSites
      if (nRepositories.eq.nInterimStorageSites) then
         incrementalRepositoryFillRate(countInterimStorageSites) =
interimStorageToRepositoryRate(countInterimStorageSites,countTime-1)
      elseif (nRepositories.eq.1) then
         incrementalRepositoryFillRate(1) = incrementalRepositoryFillRate(1) +
interimStorageToRepositoryRate(countInterimStorageSites,countTime-1)
      endif ! nRepositories.eq.nInterimStorageSites
   enddo ! countInterimStorageSites = 1, nInterimStorageSites

   ! add to the repository the calculated incremental fill rate from interim storage
   do countRepository = 1, nRepositories
      fuelInRepository(countRepository,countTime) = fuelInRepository(countRepository,countTime) +
incrementalRepositoryFillRate(countRepository)*timeStep
   enddo ! countRepositories = 1, nRepositories

   ! subtract from the interim storages the incremental fill rate to the repository
   do countInterimStorageSites = 1, nInterimStorageSites
      fuelInInterimStorage(countInterimStorageSites,countTime) =
fuelInInterimStorage(countInterimStorageSites,countTime) -
interimStorageToRepositoryRate(countInterimStorageSites,countTime-1)*timeStep
   enddo ! countInterimStorageSites = 1, nInterimStorageSites

endsubroutine calculateStocks
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: initializeFlows
! DESCRIPTION: establishes starting values for all flows in system
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine initializeFlows(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: tempCountTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor
   integer :: countInterimStorageSites
   integer :: countRepository
   double precision :: totalRxDischargeRate

   ! flow into reactors (reactorLoadingRate) is already initialized in menu subroutine, no need to repeat
it

   ! ASSUMPTION: we assume dynamic equilibrium for all reactor cores at begining of simulation
   ! therefore, we will initialize all reactorDischargeRates to reactorLoadingRate
   do tempCountTime = 1, nTimeSteps
      do countRegion = 1, nRegions
         do countReactorSite = 1, nReactorSites(countRegion)
            totalRxDischargeRate = 0.0 ! initialize to zero
            do countReactor = 1, nReactors(countRegion,countReactorSite)
               reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime) =
reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime)
               ! note: spent fuel pool cooling rate is really based on the reactor discharge rate
"spentFuelCoolingTime" ago...
               ! however, we are continuing our assumption of initializing the system in dynamic
equilibrium through this stage of flows
               totalRxDischargeRate = totalRxDischargeRate +
reactorDischargeRate(countRegion,countReactorSite,countReactor,countTime)
            enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
            totalSiteReactorDischargeRate(countRegion,countReactorSite,countTime) = totalRxDischargeRate
            spentFuelPoolCoolingRate(countRegion,countReactorSite,countTime) = totalRxDischargeRate ! rate
at which fuel flows from "hotUsedFuel" to "coldUsedFuel"
```

```
            requiredSpentFuelPoolDischargeRate(countRegion,countReactorSite) = 0.0 ! MUST REDEFINE
PROPERLY!!!
            ! initialize all flows to repository as zero
            spentFuelPoolToRepositoryRate(countRegion,countReactorSite,tempCountTime) = 0.0
            spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,tempCountTime) = 0.0
            dryStorageToRepositoryRate(countRegion,countReactorSite,tempCountTime) = 0.0
            dryStorageToInterimStorageRate(countRegion,countReactorSite,tempCountTime) = 0.0
            ! flow into dry storage casks (spentFuelPoolToDryStorageRate) is already initialized in menu
subroutine, no need to repeat it
         enddo ! countReactorSite = 1, nReactorSites(countRegion)
      enddo ! countRegion = 1, nRegions

      do countInterimStorageSites = 1, nInterimStorageSites
         interimStorageToRepositoryRate(countInterimStorageSites,tempCountTime) = 0.0
         interimStorageFillRate(countInterimStorageSites) = 0.0
      enddo ! countInterimStorageSites = 1, nInterimStorageSites

      do countRepository = 1, nRepositories
         requiredRepositoryFillRate(countRepository) = 0.0 ! MUST REDEFINE PROPERLY!!!
      enddo ! countRepository = 1, nRepositories

      totalFlowIntoRepositories(tempCountTime) = 0.0

   enddo ! tempCountTime = 1, nTimeSteps

   call sumFlowIntoDryStorage(countTime)

endsubroutine initializeFlows
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: initializeStocks
! DESCRIPTION: establishes starting values for all stocks in system
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine initializeStocks(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor
   double precision :: totalSiteReactorLoadingRate

   ! fuel in reactors is already initialized in menu subroutine, no need to repeat it

   ! used fuel in spent fuel storage pool needs to be divided into hot and cold used fuel
   ! assume hot used fuel initializes to RxLoading&DischargeRate*SFPCoolingTime
   ! NOTE: this is a major assumption that the hot fuel starts in dynamic equilibrium !!!
   ! This assumption would not work if we began the simulation within a couple years of starting a new
reactor !!!
   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         totalSiteReactorLoadingRate = 0.0 ! initialize to zero
         do countReactor = 1, nReactors(countRegion,countReactorSite)
            totalSiteReactorLoadingRate = totalSiteReactorLoadingRate +
reactorLoadingRate(countRegion,countReactorSite,countReactor,countTime)
         enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
         hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime) =
totalSiteReactorLoadingRate*spentFuelPoolCoolingTime
         !write(*,*) 'countRegion', countRegion, 'countReactorSite', countReactorSite,
'hotFuelInSpentFuelPool', hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
         coldFuelInSpentFuelPool(countRegion,countReactorSite,countTime) =
fuelInSpentFuelPool(countRegion,countReactorSite,countTime) -
hotFuelInSpentFuelPool(countRegion,countReactorSite,countTime)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! countRegion = 1, nRegions

   call sumStockInReactors(countTime)
   call sumStockInSpentFuelPools(countTime)
```

```
    call sumStockInDryStorage(countTime)
    call sumStockInInterimStorage(countTime)
    call sumStockInRepositories(countTime)

    ! fuel in dry storage casks is already initialized in menu subroutine, no need to repeat it

    ! fuel in interim storage facilities is already initialized in menu subroutine, no need to repeat it

    ! fuel in repositories is already initialized in menu subroutine, no need to repeat it

endsubroutine initializeStocks
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: Menu
! DESCRIPTION: reads input file to collect problem variables
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine menu (countTime, totalReactorSites, totalReactors)
use problemParameters
implicit none

   character(25) :: inpFile                    ! name of the input file
   logical :: valid                            ! does the input file exist?

   integer, intent(in) :: countTime        !
   integer :: tempCountTime
   integer, intent(out) :: totalReactorSites ! total number of reactor sites in system
   integer, intent(out) :: totalReactors     ! total number of reactors in system


   integer :: countRepository                  ! counter for repositories
   integer :: repositoryNumber                 ! read from input file for verification purposes
   integer :: countRepositoryExpansions
   integer :: maxRepositoryExpansions          ! used to allocate array
   integer :: countInterimStorage              ! counter for interim storage sites
   integer :: countRegion                      ! counter for regions
   integer :: regionNumber                     ! read from input file for verification purposes
   integer :: countReactorSite                 ! counter for sites
   integer :: reactorSiteNumber                ! read from input file for verification purposes
   integer :: maxReactorSites                  ! maximum number of reactor sites in any single region
(expected to be under 30)
   integer :: maxReactors                      ! maximum number of reactors on any single site (likely 2 or
3)
   integer :: countReactor

   double precision, allocatable :: tempRepositoryCapacity(:,:)
   double precision, allocatable :: tempRepositoryMaxFillRate(:,:)

   integer :: t, i, j !debug variables... should be deleted

! ============================================================
! I N I T I A L I Z E   V A R I A B L E S   A N D   B E G I N
! ============================================================
```

```
totalReactorSites = 0
totalReactors = 0

write(*,*)
write(*,'(a)') 'Welcome to the Taylor Repository Thesis Program'
write(*,*)
write(*,'(a)', advance = 'no') 'What is the input file named? '
read(*,*) inpFile
write(*,*)
inquire(file= inpFile, exist=valid)
if(.not.valid) stop 'ERROR: Input file not found'
write(*,*) 'Thank you, input file is named : ', trim(inpFile)

open(11, file=inpFile,status='old')
read(11,*) initialTime, finalTime, timeStep

nTimeSteps = 1 + (finalTime - initialTime)/timeStep

allocate(totalStockInReactors(nTimeSteps))
allocate(totalStockInSpentFuelPools(nTimeSteps))
allocate(totalStockInDryStorage(nTimeSteps))
allocate(totalStockInInterimStorage(nTimeSteps))
allocate(totalStockInRepositories(nTimeSteps))

allocate(totalFlowIntoDryStorage(nTimeSteps))
allocate(totalFlowIntoInterimStorage(nTimeSteps))
allocate(totalFlowIntoRepositories(nTimeSteps))

allocate(countShutdownSFPWithFuel(nTimeSteps))
allocate(totalSFPDischargeRate(nTimeSteps))
allocate(countShutdownDSC(nTimeSteps))
allocate(countNewDSC(nTimeSteps))
allocate(countOnlineDSC(nTimeSteps))
allocate(countUsedIS(nTimeSteps))
allocate(countLoadingIS(nTimeSteps))
allocate(countUsedRepository(nTimeSteps))
allocate(countLoadingRepository(nTimeSteps))

allocate(wetStorageExpense(nTimeSteps))
allocate(caskConditioningExpense(nTimeSteps))
```

```
allocate(dryStorageExpense(nTimeSteps))
allocate(transportationExpense(nTimeSteps))
allocate(interimStorageExpense(nTimeSteps))
allocate(repositoryConditioningExpense(nTimeSteps))
allocate(repositoryExpense(nTimeSteps))


read(11,*) spentFuelPoolCoolingTime

read(11,*) nRepositories

if (nRepositories.eq.1) read(11,*) repositoryRegion

allocate (nRepositoryExpansions(nRepositories))

read(11,*) nRepositoryExpansions(1:nRepositories)
!write(*,*) 'nRepExp', nRepositoryExpansions(1:nRepositories)

write(*,*) nRepositories, 'nRepositories', nRepositoryExpansions(1:nRepositories), 'expansions'

maxRepositoryExpansions = 0
do countRepository = 1, nRepositories
   maxRepositoryExpansions = max(maxRepositoryExpansions,nRepositoryExpansions(countRepository))
enddo ! countRepository = 1, nRepositories

allocate (repositoryExpansionTimes(nRepositories,maxRepositoryExpansions))
allocate (fuelInRepository(nRepositories,nTimeSteps))
allocate (repositoryCapacity(nRepositories,nTimeSteps))
allocate (tempRepositoryCapacity(nRepositories,maxRepositoryExpansions))
allocate (repositoryMaxFillRate(nRepositories,nTimeSteps))
allocate (tempRepositoryMaxFillRate(nRepositories,maxRepositoryExpansions))
allocate (requiredRepositoryFillRate(nRepositories))

allocate (repositoryConstructionCost(nRepositories,maxRepositoryExpansions))
allocate (repositoryConstructionTime(nRepositories,maxRepositoryExpansions))
allocate (repositoryInterestRate(nRepositories,maxRepositoryExpansions))
allocate (repositoryClosureCost(nRepositories))
allocate (repositoryClosureTime(nRepositories))
allocate (repositoryClosureInterestRate(nRepositories))

allocate (repositoryLoadingRate(nRepositories,nTimeSteps))
```

```
   allocate (amortizedRepositoryCapital(nRepositories,maxRepositoryExpansions))
   allocate (amortizedRepositoryClosure(nRepositories))

   do countRepository = 1, nRepositories
      fuelInRepository(countRepository,countTime) = 0.0
      read(11,*) repositoryNumber,
repositoryExpansionTimes(countRepository,1:nRepositoryExpansions(countRepository)),
tempRepositoryCapacity(countRepository,1:nRepositoryExpansions(countRepository)),
tempRepositoryMaxFillRate(countRepository,1:nRepositoryExpansions(countRepository))
   enddo ! countRepository = 1, nRepositories

   ! first initialize the repository capacity and max fill rates to zero
   do countRepository = 1, nRepositories
      do tempCountTime = 1, nTimeSteps
         repositoryCapacity(countRepository,tempCountTime) = 0.0
         repositoryMaxFillRate(countRepository,tempCountTime) = 0.0
      enddo ! tempCountTime = 1, nTimeSteps
   enddo ! countRepository = 1, nRepositories

   ! then go through and at every time step after the repository expansion, overwrite the capacity and max
fill rates
   do countRepository = 1, nRepositories
      do countRepositoryExpansions = 1, nRepositoryExpansions(countRepository)
         do tempCountTime = 1, nTimeSteps
            if
(repositoryExpansionTimes(countRepository,countRepositoryExpansions).lt.initialTime+tempCountTime*timeStep)
then
               repositoryCapacity(countRepository,tempCountTime) =
tempRepositoryCapacity(countRepository,countRepositoryExpansions)
               repositoryMaxFillRate(countRepository,tempCountTime) =
tempRepositoryMaxFillRate(countRepository,countRepositoryExpansions)
            endif !
repositoryExpansionTimes(countRepository,tempCountTime).lt.initialTime+tempCountTime*timeStep
         enddo ! tempCountTime = 1, nTimeSteps
      enddo ! countRepositoryExpansions = 1, nRepositoryExpansions(countRepository)
   enddo ! countRepository = 1, nRepositories

   read(11,*) nInterimStorageSites

   if (nInterimStorageSites.eq.1) read(11,*) interimStorageRegion
```

```
   write(*,*) nInterimStorageSites, 'nInterimStorageSites'

   allocate (interimStorageStartTime(nInterimStorageSites))
   allocate (fuelInInterimStorage(nInterimStorageSites,nTimeSteps))
   allocate (interimStorageCapacity(nInterimStorageSites))
   allocate (interimStorageMaxFillRate(nInterimStorageSites))
   allocate (interimStorageToRepositoryRate(nInterimStorageSites,nTimeSteps))
   allocate (interimStorageFillRate(nInterimStorageSites))

   allocate (amortizedISCapital(nInterimStorageSites))
   allocate (amortizedISDD(nInterimStorageSites))

   allocate (interimStorageLoadingRate(nInterimStorageSites,nTimeSteps))
   allocate (interimStorageDischargeRate(nInterimStorageSites,nTimeSteps))

   do countInterimStorage = 1, nInterimStorageSites
      read(11,*) interimStorageStartTime(countInterimStorage),
fuelInInterimStorage(countInterimStorage,countTime), interimStorageCapacity(countInterimStorage),
interimStorageMaxFillRate(countInterimStorage)
   enddo ! countInterimStorage = 1, nInterimStorageSites

   read(11,*) nRegions
   write(*,*) nRegions, 'nRegions'

   allocate (transportationCost(nRegions,nRegions))
   allocate (interimStorageConstructionCost(nInterimStorageSites))
   allocate (interimStorageConstructionTime(nInterimStorageSites))
   allocate (interimStorageConstructionInterestRate(nInterimStorageSites))
   allocate (interimStorageDDCost(nInterimStorageSites))
   allocate (interimStorageDDTime(nInterimStorageSites))
   allocate (interimStorageDDInterestRate(nInterimStorageSites))

   allocate (totalTransportationRate(nRegions,nRegions,nTimeSteps))

   read(11,*) onlineAnnualWetStorageCost, shutdownAnnualWetStorageCost
   read(11,*) caskConditioningCost
   read(11,*) initialOnSiteDryStorageCost, onlineAnnualDryStorageCost, shutdownAnnualDryStorageCost
   do countRegion = 1, nRegions
      read(11,*) transportationCost(countRegion,1:nRegions)
   enddo ! countRegion = 1, nRegions
```

```
   do countInterimStorage = 1, nInterimStorageSites
      read(11,*) interimStorageConstructionCost(countInterimStorage),
interimStorageConstructionTime(countInterimStorage),
interimStorageConstructionInterestRate(countInterimStorage)
      amortizedISCapital(countInterimStorage) =
interimStorageConstructionCost(countInterimStorage)*(interimStorageConstructionInterestRate(countInterimSto
rage)*(1+interimStorageConstructionInterestRate(countInterimStorage))**interimStorageConstructionTime(count
InterimStorage))/(((1+interimStorageConstructionInterestRate(countInterimStorage))**interimStorageConstruct
ionTime(countInterimStorage))-1)
      read(11,*) interimStorageDDCost(countInterimStorage), interimStorageDDTime(countInterimStorage),
interimStorageDDInterestRate(countInterimStorage)
      amortizedISDD(countInterimStorage) =
interimStorageDDCost(countInterimStorage)*(interimStorageDDInterestRate(countInterimStorage)*(1+interimStor
ageDDInterestRate(countInterimStorage))**interimStorageDDTime(countInterimStorage))/(((1+interimStorageDDIn
terestRate(countInterimStorage))**interimStorageDDTime(countInterimStorage))-1)
   enddo ! countInterimStorage = 1, nInterimStorageSites
   read(11,*) interimStorageAnnualBaseCost, interimStorageAnnualLoadingCost, interimStorageProrateCost
   read(11,*) repositoryConditioningPackaging
   do countRepository = 1, nRepositories
      read(11,*) repositoryConstructionCost(countRepository,1:nRepositoryExpansions(countRepository)),
repositoryConstructionTime(countRepository,1:nRepositoryExpansions(countRepository)),
repositoryInterestRate(countRepository,1:nRepositoryExpansions(countRepository)),
repositoryClosureCost(countRepository), repositoryClosureTime(countRepository),
repositoryClosureInterestRate(countRepository)
      do countRepositoryExpansions = 1, nRepositoryExpansions(countRepository)
         amortizedRepositoryCapital(countRepository,countRepositoryExpansions) =
repositoryConstructionCost(countRepository,countRepositoryExpansions)*(repositoryInterestRate(countReposito
ry,countRepositoryExpansions)*(1+repositoryInterestRate(countRepository,countRepositoryExpansions))**reposi
toryConstructionTime(countRepository,countRepositoryExpansions))/(((1+repositoryInterestRate(countRepositor
y,countRepositoryExpansions))**repositoryConstructionTime(countRepository,countRepositoryExpansions))-1)
      enddo ! countRepositoryExpansions = 1, nRepositoryExpansions
      amortizedRepositoryClosure(countRepository) =
repositoryClosureCost(countRepository)*(repositoryClosureInterestRate(countRepository)*(1+repositoryClosure
InterestRate(countRepository))**repositoryClosureTime(countRepository))/(((1+repositoryClosureInterestRate(
countRepository))**repositoryClosureTime(countRepository))-1)
   enddo ! countRepository = 1, nRepositories
   read(11,*) repositoryAnnualBaseCost, repositoryAnnualLoadingCost

   allocate (nReactorSites(nRegions))

   read(11,*) nReactorSites(:)
```

```
write(*,*) nReactorSites(:)

! the following do loop is only used to allocate nReactors array
maxReactorSites = 1
do countRegion = 1, nRegions
   totalReactorSites = totalReactorSites + nReactorSites(countRegion)
   maxReactorSites = max(maxReactorSites,nReactorSites(countRegion))
enddo ! countRegion = 1, nRegions

allocate (nReactors(nRegions,maxReactorSites))
allocate (totalSiteReactorDischargeRate(nRegions,maxReactorSites,nTimeSteps))

maxReactors = 1
do countRegion = 1, nRegions
   read(11,*) nReactors(countRegion,1:nReactorSites(countRegion))
   !write(*,*) nReactors(countRegion,1:nReactorSites(countRegion))
   do countReactorSite = 1, nReactorSites(countRegion)
      maxReactors = max(maxReactors,nReactors(countRegion,countReactorSite))
   enddo ! countReactorSite = 1, nSites
enddo ! countRegion = 1, nRegions

allocate (reactorShutdownTime(nRegions,maxReactorSites,maxReactors))
allocate (fuelInReactor(nRegions,maxReactorSites,maxReactors,nTimeSteps))
allocate (reactorLoadingRate(nRegions,maxReactorSites,maxReactors,nTimeSteps))
allocate (reactorDischargeRate(nRegions,maxReactorSites,maxReactors,nTimeSteps))

allocate (fuelInSpentFuelPool(nRegions,maxReactorSites,nTimeSteps))
allocate (spentFuelPoolCapacity(nRegions,maxReactorSites))
allocate (hotFuelInSpentFuelPool(nRegions,maxReactorSites,nTimeSteps))
allocate (coldFuelInSpentFuelPool(nRegions,maxReactorSites,nTimeSteps))
allocate (spentFuelPoolCoolingRate(nRegions,maxReactorSites,nTimeSteps))
allocate (spentFuelPoolToRepositoryRate(nRegions,maxReactorSites,nTimeSteps))
allocate (spentFuelPoolToInterimStorageRate(nRegions,maxReactorSites,nTimeSteps))
allocate (requiredSpentFuelPoolDischargeRate(nRegions,maxReactorSites))

allocate (fuelInDryStorageCasks(nRegions,maxReactorSites,nTimeSteps))
allocate (spentFuelPoolToDryStorageRate(nRegions,maxReactorSites,nTimeSteps))
allocate (dryStorageToRepositoryRate(nRegions,maxReactorSites,nTimeSteps))
allocate (dryStorageToInterimStorageRate(nRegions,maxReactorSites,nTimeSteps))
```

```
    !debug
    ! must set all rates to zero to make sure my gunk is calculating properly
    do t = 1, nTimeSteps
     do i = 1, nRegions
      do j = 1, nReactorSites(i)
       spentFuelPoolToRepositoryRate(i,j,t) = 0.0
       dryStorageToRepositoryRate(i,j,t) = 0.0
       spentFuelPoolToInterimStorageRate(i,j,t) = 0.0
       dryStorageToInterimStorageRate(i,j,t) = 0.0
       spentFuelPoolToDryStorageRate(i,j,t) = 0.0
      enddo
     enddo
     do i = 1, nInterimStorageSites
       interimStorageToRepositoryRate(i,t) = 0.0
     enddo
    enddo

    maxSpentFuelPoolDischargeCapacity = 0.0
    do countRegion = 1, nRegions
         read(11,*) regionNumber
         if(regionNumber.ne.countRegion) then
             write(*,*) 'WARNING, regionNumber.ne.countRegion'
         endif
         do countReactorSite = 1, nReactorSites(countRegion)
             totalReactors = totalReactors + nReactors(countRegion,countReactorSite)
             read(11,*) reactorSiteNumber,
reactorShutdownTime(countRegion,countReactorSite,1:nReactors(countRegion,countReactorSite)),
fuelInReactor(countRegion,countReactorSite,1:nReactors(countRegion,countReactorSite),countTime),
reactorLoadingRate(countRegion,countReactorSite,1:nReactors(countRegion,countReactorSite),countTime),
fuelInSpentFuelPool(countRegion,countReactorSite,countTime),
spentFuelPoolCapacity(countRegion,countReactorSite),
fuelInDryStorageCasks(countRegion,countReactorSite,countTime),
spentFuelPoolToDryStorageRate(countRegion,countReactorSite,countTime)
             do countReactor = 1, nReactors(countRegion,countReactorSite)
                maxSpentFuelPoolDischargeCapacity = max(maxSpentFuelPoolDischargeCapacity,
spentFuelPoolCapacity(countRegion,countReactorSite)/(reactorLoadingRate(countRegion,countReactorSite,countR
eactor,countTime)*timeStep))
             enddo ! countReactor = 1, nReactors(countRegion,countReactorSite)
         enddo ! countReactorSite = 1, nReactorSites(countRegion)
    enddo ! countRegion = 1, nRegions
```

```
   !write(*,*) 'maxSpentFuelPoolDischargeCapacity', maxSpentFuelPoolDischargeCapacity
endsubroutine menu
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: printResults
! DESCRIPTION: prints summary of all the stocks and flows of importance
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine printResults
use problemParameters
implicit none

integer :: countTime

   write(*,*) '  S       T          O          C          K          S'
   write(*,*) '  Time    Rxs        SFPs       DSCs       IS         Rep'

   do countTime = 1, nTimeSteps
      write(*,50) initialTime+(countTime-1)*timeStep, totalStockInReactors(countTime),
totalStockInSpentFuelPools(countTime), totalStockInDryStorage(countTime),
totalStockInInterimStorage(countTime), totalStockInRepositories(countTime)
   enddo ! countTime = 1, nTimeSteps

   write(*,*) '  F       L             O             W             S             !'
   write(*,*) '  Time    Rxs           SFPs          inDSCs        IS            Rep'

   do countTime = 1, nTimeSteps
      write(*,51) initialTime+(countTime-1)*timeStep, totalFlowIntoDryStorage(countTime)
   enddo ! countTime = 1, nTimeSteps

   write(*,*) '  C       O          S          T          S          !          !          !'
   write(*,*) '  Time    SFPs       CaskCond   DSC        Tran       IS         RepCond    Rep'

   do countTime = 1, nTimeSteps
      write(*,52) initialTime+(countTime-1)*timeStep, wetStorageExpense(countTime),
caskConditioningExpense(countTime), dryStorageExpense(countTime), transportationExpense(countTime),
interimStorageExpense(countTime), repositoryConditioningExpense(countTime), repositoryExpense(countTime)
   enddo ! countTime = 1, nTimeSteps

50  format (f8.1, 5es11.3)
51  format (f8.1, 1es14.6)
52  format (f8.1, 7es11.3)

endsubroutine printResults
```

```
module problemParameters

    !integer :: totalReactorSites
    !integer :: totalReactors

    ! defined in menu subroutine
    double precision :: initialTime              ! year to start simulation (ie. 2007.5)
    double precision :: finalTime                ! year to end simulation (ie. 2007.5)
    double precision :: timeStep                 ! time step of simulation (ie. 0.5 years)
    integer :: nTimeSteps                        ! number of time steps for counting purposes

    double precision :: spentFuelPoolCoolingTime ! time fuel must remain in spent fuel pool before it can
be handled

    integer :: nRepositories                       ! number of repositories
    integer :: RepositoryRegion                    ! only used if nRepositories.eq.1
    integer, allocatable :: nRepositoryExpansions(:) ! number of expansions of repository (opening counts
as 1st expansion)
    double precision, allocatable :: repositoryExpansionTimes(:,:) ! date that repository can start
receiving fuel
    double precision, allocatable :: fuelInRepository(:,:) ! fuel in repositories at any given countTime in
MT
    double precision, allocatable :: repositoryCapacity(:,:) ! repository capacity in MT
    double precision, allocatable :: repositoryMaxFillRate(:,:) ! repository maximum fill rate in MT/year
    double precision, allocatable :: totalStockInRepositories(:)
    double precision, allocatable :: totalFlowIntoRepositories(:)

    integer :: nInterimStorageSites             ! number of interim storage
    integer :: interimStorageRegion             ! only used if nInterimStorageSites.eq.1
    double precision, allocatable :: interimStorageStartTime(:) ! date that interim storage can start
receiving fuel
    double precision, allocatable :: fuelInInterimStorage(:,:) ! fuel in interim storage at any given
countTime in MT
    double precision, allocatable :: interimStorageCapacity(:) ! interim storage capacity in MT
    double precision, allocatable :: interimStorageMaxFillRate(:) ! interim storage maximum fill rate in
MT/year
    double precision, allocatable :: totalStockInInterimStorage(:)
    double precision, allocatable :: totalFlowIntoInterimStorage(:)

    integer :: nRegions                          ! number of regions for reactors and repositories
```

```
! economics
double precision :: onlineAnnualWetStorageCost
double precision :: shutdownAnnualWetStorageCost
double precision :: caskConditioningCost
double precision :: initialOnSiteDryStorageCost
double precision :: onlineAnnualDryStorageCost
double precision :: shutdownAnnualDryStorageCost
double precision, allocatable :: transportationCost(:,:)
double precision, allocatable :: interimStorageConstructionCost(:)
double precision, allocatable :: interimStorageConstructionTime(:)
double precision, allocatable :: interimStorageConstructionInterestRate(:)
double precision, allocatable :: amortizedISCapital(:)
double precision, allocatable :: interimStorageDDCost(:)
double precision, allocatable :: interimStorageDDTime(:)
double precision, allocatable :: interimStorageDDInterestRate(:)
double precision, allocatable :: amortizedISDD(:)
double precision :: interimStorageAnnualBaseCost
double precision :: interimStorageAnnualLoadingCost
double precision :: interimStorageProrateCost
double precision :: repositoryConditioningPackaging
double precision, allocatable :: repositoryConstructionCost(:,:)
double precision, allocatable :: repositoryConstructionTime(:,:)
double precision, allocatable :: repositoryInterestRate(:,:)
double precision, allocatable :: amortizedRepositoryCapital(:,:)
double precision, allocatable :: repositoryClosureCost(:)
double precision, allocatable :: repositoryClosureTime(:)
double precision, allocatable :: repositoryClosureInterestRate(:)
double precision, allocatable :: amortizedRepositoryClosure(:)
double precision :: repositoryAnnualBaseCost
double precision :: repositoryAnnualLoadingCost

double precision, allocatable :: wetStorageExpense(:)
double precision, allocatable :: caskConditioningExpense(:)
double precision, allocatable :: dryStorageExpense(:)
double precision, allocatable :: transportationExpense(:)
double precision, allocatable :: interimStorageExpense(:)
double precision, allocatable :: repositoryConditioningExpense(:)
double precision, allocatable :: repositoryExpense(:)

integer, allocatable :: nReactorSites(:)     ! number of reactor sites in regions 1 to nRegions
integer, allocatable :: nReactors(:,:)       ! number of reactors at each coordinate (region,site)
```

```
    double precision, allocatable :: reactorShutdownTime(:,:,:) ! time (year) for shutdown of each reactor
    double precision, allocatable :: fuelInReactor(:,:,:,:) ! fuel loaded in each reactor in MT (4th
dimension is time)
    double precision, allocatable :: reactorLoadingRate(:,:,:,:) ! each reactors fuel loading rate in
MT/year
    double precision, allocatable :: totalStockInReactors(:)

    double precision, allocatable :: fuelInSpentFuelPool(:,:,:) ! each spent fuel pool loading in MT
(region, site)
    double precision, allocatable :: spentFuelPoolCapacity(:,:) !
    double precision :: maxSpentFuelPoolDischargeCapacity !
    double precision, allocatable :: totalStockInSpentFuelPools(:)

    double precision, allocatable :: fuelInDryStorageCasks(:,:,:) ! each site's used fuel loading in
interim storage MT
    double precision, allocatable :: spentFuelPoolToDryStorageRate(:,:,:)
    double precision, allocatable :: totalStockInDryStorage(:)
    double precision, allocatable :: totalFlowIntoDryStorage(:)

    ! allocated in menu, defined in initializeStocks
    double precision, allocatable :: hotFuelInSpentFuelPool(:,:,:) ! distinguished from cold fuel because
we cannot do anything with hot fuel
    double precision, allocatable :: coldFuelInSpentFuelPool(:,:,:) !

    ! allocated in menu, defined in initializeFlows
    double precision, allocatable :: reactorDischargeRate(:,:,:,:)
    double precision, allocatable :: totalSiteReactorDischargeRate(:,:,:)
    double precision, allocatable :: spentFuelPoolCoolingRate(:,:,:)
    double precision, allocatable :: requiredSpentFuelPoolDischargeRate(:,:)
    double precision, allocatable :: requiredRepositoryFillRate(:)
    double precision, allocatable :: spentFuelPoolToRepositoryRate(:,:,:)
    double precision, allocatable :: spentFuelPoolToInterimStorageRate(:,:,:)
    double precision, allocatable :: dryStorageToRepositoryRate(:,:,:)
    double precision, allocatable :: dryStorageToInterimStorageRate(:,:,:)
    double precision, allocatable :: interimStorageToRepositoryRate(:,:)
    double precision, allocatable :: interimStorageFillRate(:)

    ! used for economics subroutines
    double precision, allocatable :: totalSFPDischargeRate(:)
    double precision, allocatable :: totalTransportationRate(:,:,:)
    integer, allocatable :: countShutdownSFPWithFuel(:)
```

```
    integer, allocatable :: countShutdownDSC(:)
    integer, allocatable :: countNewDSC(:)
    integer, allocatable :: countOnlineDSC(:)
    integer, allocatable :: countUsedIS(:)
    integer, allocatable :: countLoadingIS(:)
    double precision, allocatable :: interimStorageLoadingRate(:,:)
    double precision, allocatable :: interimStorageDischargeRate(:,:)
    integer, allocatable :: countUsedRepository(:)
    integer, allocatable :: countLoadingRepository(:)
    double precision, allocatable :: repositoryLoadingRate(:,:)


endmodule problemParameters
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumOnlineDSC
! DESCRIPTION: searches through all reactor sites and counts number that
! are online with fuel in DSCs.
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumDSC(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor
   integer :: countTempTime
   double precision :: siteShutdownTime
   double precision :: currentTime
   double precision :: pastFuelInReactorSiteDSC

   currentTime = initialTime + (countTime-1)*timeStep
   countShutdownDSC(countTime) = 0 ! initialize to zero reactors
   countNewDSC(countTime) = 0
   countOnlineDSC(countTime) = 0

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         siteShutdownTime = initialTime
         do countReactor = 1, nReactors(countRegion,countReactorSite)
            siteShutdownTime =
max(siteShutdownTime,reactorShutdownTime(countRegion,countReactorSite,countReactor))
         enddo ! countReactor = 1, nReactors(countReactorSite)
         if (currentTime.gt.siteShutdownTime .and.
fuelInDryStorageCasks(countRegion,countReactorSite,countTime).gt.0.01) then
            countShutdownDSC(countTime) = countShutdownDSC(countTime)+1
            ! see if this particular site is just starting DS to determine if we need to include initial
DSC cost
            pastFuelInReactorSiteDSC = 0.0
            do countTempTime = 1, countTime-1
               pastFuelInReactorSiteDSC = pastFuelInReactorSiteDSC +
fuelInDryStorageCasks(countRegion,countReactorSite,countTempTime)
            enddo ! countTempTime = 1, countTime-1
```

```
            if (pastFuelInReactorSiteDSC.lt.0.01) countNewDSC(countTime) = countNewDSC(countTime)+1
          elseif (currentTime.le.siteShutdownTime .and.
fuelInDryStorageCasks(countRegion,countReactorSite,countTime).gt.0.01) then
            countOnlineDSC(countTime) = countOnlineDSC(countTime)+1
          endif ! currentTime.gt.siteShutdownTime .and.
fuelInDryStorageCasks(countRegion,countReactorSite,countTime).gt.0.01
       enddo ! countReactorSite = 1, nReactorSites(countRegion)
    enddo ! region = 1, nRegions

endsubroutine sumDSC
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumFlowFromSFP
! DESCRIPTION: searches through all spent fuel pools and counts mass of
! spent fuel that is discharged for conditioning
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumFlowFromSFP(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite

   totalSFPDischargeRate(countTime) = 0.0 ! initialize to zero MT/year

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         totalSFPDischargeRate(countTime) = totalSFPDischargeRate(countTime) +
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) +
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) +
spentFuelPoolToDryStorageRate(countRegion,countReactorSite,countTime)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! region = 1, nRegions

endsubroutine sumFlowFromSFP
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumFlowIntoDryStorage
! DESCRIPTION: searches through all reactor sites and counts flows to DSC
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumFlowIntoDryStorage(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor

   totalFlowIntoDryStorage(countTime) = 0.0 ! initialize to zero MT

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         totalFlowIntoDryStorage(countTime) = totalFlowIntoDryStorage(countTime) +
spentFuelPoolToDryStorageRate(countRegion,countReactorSite,countTime)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! region = 1, nRegions

endsubroutine sumFlowIntoDryStorage
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumFlowIntoInterimStorage
! DESCRIPTION: searches through all flows into all interim storage sites and finds sum
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumFlowIntoInterimStorage(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime

   integer :: countRegion
   integer :: countReactorSite
   integer :: countInterimStorageSite

   totalFlowIntoInterimStorage = 0.0 ! initialize to zero MT/year
   do countInterimStorageSite = 1, nInterimStorageSites
      interimStorageLoadingRate(countInterimStorageSite,countTime) = 0.0
   enddo ! countInterimStorageSite = 1, nInterimStorageSites

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         totalFlowIntoInterimStorage = totalFlowIntoInterimStorage +
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) +
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
         ! the following is used for the economics later on...
         if (nInterimStorageSites.eq.nRegions) then
            interimStorageLoadingRate(countRegion,countTime) =
interimStorageLoadingRate(countRegion,countTime) +
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) +
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
         elseif (nInterimStorageSites.eq.1) then
            interimStorageLoadingRate(1,countTime) = interimStorageLoadingRate(1,countTime) +
spentFuelPoolToInterimStorageRate(countRegion,countReactorSite,countTime) +
dryStorageToInterimStorageRate(countRegion,countReactorSite,countTime)
         endif ! nInterimStorageSites.eq.nRegions
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! region = 1, nRegions


endsubroutine sumFlowIntoInterimStorage
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumFlowIntoRepositories
! DESCRIPTION: searches through all flows into all repositories and finds sum
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumFlowIntoRepositories(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime

   integer :: countRegion
   integer :: countReactorSite
   integer :: countInterimStorageSite
   integer :: countRepository

   totalFlowIntoRepositories(countTime) = 0.0 ! initialize to zero MT/year

   do countRepository = 1, nRepositories
      repositoryLoadingRate(countRepository,countTime) = 0.0
   enddo ! countRepository = 1, nRepositories

   do countInterimStorageSite = 1, nInterimStorageSites
      interimStorageDischargeRate(countInterimStorageSite,countTime) = 0.0
   enddo ! countInterimStorageSite = 1, nInterimStorageSites

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         totalFlowIntoRepositories(countTime) = totalFlowIntoRepositories(countTime) +
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) +
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
         if (nRepositories.eq.nRegions) then
            repositoryLoadingRate(countRegion,countTime) = repositoryLoadingRate(countRegion,countTime) +
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) +
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
         elseif (nRepositories.eq.1) then
            repositoryLoadingRate(1,countTime) = repositoryLoadingRate(1,countTime) +
spentFuelPoolToRepositoryRate(countRegion,countReactorSite,countTime) +
dryStorageToRepositoryRate(countRegion,countReactorSite,countTime)
         endif ! nRepositories.eq.nRegions
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
```

```
   enddo ! region = 1, nRegions

   do countInterimStorageSite = 1, nInterimStorageSites
      totalFlowIntoRepositories(countTime) = totalFlowIntoRepositories(countTime) +
interimStorageToRepositoryRate(countInterimStorageSite,countTime)
      ! the following is used for the economics later on...
      interimStorageDischargeRate(countInterimStorageSite,countTime) =
interimStorageDischargeRate(countInterimStorageSite,countTime) +
interimStorageToRepositoryRate(countInterimStorageSite,countTime)
      if (nRepositories.eq.nInterimStorageSites) then
         repositoryLoadingRate(countInterimStorageSite,countTime) =
repositoryLoadingRate(countInterimStorageSite,countTime) +
interimStorageToRepositoryRate(countInterimStorageSite,countTime)
      elseif (nRepositories.eq.1) then
         repositoryLoadingRate(1,countTime) = repositoryLoadingRate(1,countTime) +
interimStorageToRepositoryRate(countInterimStorageSite,countTime)
      endif ! nRepositories.eq.nInterimStorageSites
   enddo ! countInterimStorageSite = 1, nInterimStorageSites

endsubroutine sumFlowIntoRepositories
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumInterimStorage
! DESCRIPTION: searches through all interim storage sites and counts
! number that are in use, how many are in the process of loading or
! unloading, and how much fuel is stored at all IS sites.
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumInterimStorage(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countInterimStorageSite

   double precision :: siteShutdownTime
   double precision :: currentTime
   double precision :: pastFuelInReactorSiteDSC

   currentTime = initialTime + (countTime-1)*timeStep
   countUsedIS(countTime) = 0 ! initialize to zero interim storage sites
   countLoadingIS(countTime) = 0

   do countInterimStorageSite = 1, nInterimStorageSites
      if (currentTime.ge.interimStorageStartTime(countInterimStorageSite)) then
         countUsedIS(countTime) = countUsedIS(countTime)+1
         ! now check to see if the interim storage site is shut down and the count must be corrected
(reduced by 1)
         if(fuelInInterimStorage(countInterimStorageSite,countTime).lt.0.01 .and.
interimStorageLoadingRate(countInterimStorageSite,countTime).lt.0.01 .and.
interimStorageDischargeRate(countInterimStorageSite,countTime).lt.0.01) countUsedIS(countTime) =
countUsedIS(countTime)-1
         if(interimStorageLoadingRate(countInterimStorageSite,countTime).gt.0.01 .or.
interimStorageDischargeRate(countInterimStorageSite,countTime).gt.0.01) countLoadingIS(countTime) =
countLoadingIS(countTime)+1
      endif ! currentTime.ge.interimStorageStartTime(countInterimStorageSite)
      ! we have already calculated totalStockInInterimStorage(countTime)
   enddo ! countInterimStorageSite = 1, nInterimStorageSites

endsubroutine sumInterimStorage
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumRepository
! DESCRIPTION: searches through all repositories and counts number that
! are in use and how many are in the process of loading
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumRepository(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRepository

   double precision :: siteShutdownTime
   double precision :: currentTime
   double precision :: pastFuelInReactorSiteDSC

   currentTime = initialTime + (countTime-1)*timeStep
   countUsedRepository(countTime) = 0 ! initialize to zero interim storage sites
   countLoadingRepository(countTime) = 0

   do countRepository = 1, nRepositories
      if (currentTime.ge.repositoryExpansionTimes(countRepository,1)) then
         countUsedRepository(countTime) = countUsedRepository(countTime)+1
         if(repositoryLoadingRate(countRepository,countTime).gt.0.01) countLoadingRepository(countTime) =
countLoadingRepository(countTime)+1
      endif ! currentTime.ge.repositoryStartTime(countRepository)
   enddo ! countRepository = 1, nRepositories

endsubroutine sumRepository
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumShutdownSFPWithFuel
! DESCRIPTION: searches through all spent fuel pools and counts number of
! spent fuel pools that are both (a) shutdown and (b) contain fuel.
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumShutdownSFPWithFuel(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor
   double precision :: siteShutdownTime
   double precision :: currentTime

   currentTime = initialTime + (countTime-1)*timeStep
   countShutdownSFPWithFuel(countTime) = 0 ! initialize to zero reactors

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         siteShutdownTime = initialTime
         do countReactor = 1, nReactors(countRegion,countReactorSite)
            siteShutdownTime =
max(siteShutdownTime,reactorShutdownTime(countRegion,countReactorSite,countReactor))
         enddo ! countReactor = 1, nReactors(countReactorSite)
         if (currentTime.gt.siteShutdownTime .and.
fuelInSpentFuelPool(countRegion,countReactorSite,countTime).gt.0.01) countShutdownSFPWithFuel(countTime) =
countShutdownSFPWithFuel(countTime)+1
         ! notice the use of "gt.0.01" instead of simply "gt.0.0" because there may be some very slight
roundoff errors that leave site with negligible mass of spent fuel
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! region = 1, nRegions

endsubroutine sumShutdownSFPWithFuel
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumStockInDryStorage
! DESCRIPTION: searches through all dry storage casks and counts fuel
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumStockInDryStorage(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor

   totalStockInDryStorage(countTime) = 0.0 ! initialize to zero MT

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         totalStockInDryStorage(countTime) = totalStockInDryStorage(countTime) +
fuelInDryStorageCasks(countRegion,countReactorSite,countTime)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! region = 1, nRegions

endsubroutine sumStockInDryStorage
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumStockInInterimStorage
! DESCRIPTION: searches through all interim storage sites and counts fuel
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumStockInInterimStorage(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countInterimStorageSite

   totalStockInInterimStorage(countTime) = 0.0 ! initialize to zero MT

   do countInterimStorageSite = 1, nInterimStorageSites
      totalStockInInterimStorage(countTime) = totalStockInInterimStorage(countTime) +
fuelInInterimStorage(countInterimStorageSite,countTime)
   enddo ! countInterimStorageSite = 1, nInterimStorageSites

endsubroutine sumStockInInterimStorage
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumStockInReactors
! DESCRIPTION: searches through all reactors and counts fuel in core
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumStockInReactors(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor

   totalStockInReactors(countTime) = 0.0 ! initialize to zero MT

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         do countReactor = 1, nReactors(countRegion,countReactorSite)
            totalStockInReactors(countTime) = totalStockInReactors(countTime) +
fuelInReactor(countRegion,countReactorSite,countReactor,countTime)
         enddo ! countReactor = 1, nReactors(countRegion,countSite)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! region = 1, nRegions

endsubroutine sumStockInReactors
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumStockInRepositories
! DESCRIPTION: searches through all repositories and counts fuel
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumStockInRepositories(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRepositories

   totalStockInRepositories(countTime) = 0.0 ! initialize to zero MT

   do countRepositories = 1, nRepositories
      totalStockInRepositories(countTime) = totalStockInRepositories(countTime) +
fuelInRepository(countRepositories,countTime)
   enddo ! countRepositories = 1, nRepositories

endsubroutine sumStockInRepositories
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumStockInSpentFuelPools
! DESCRIPTION: searches through all spent fuel pools and counts fuel
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumStockInSpentFuelPools(countTime)
use problemParameters
implicit none

   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite
   integer :: countReactor

   totalStockInSpentFuelPools(countTime) = 0.0 ! initialize to zero MT

   do countRegion = 1, nRegions
      do countReactorSite = 1, nReactorSites(countRegion)
         totalStockInSpentFuelPools(countTime) = totalStockInSpentFuelPools(countTime) +
fuelInSpentFuelPool(countRegion,countReactorSite,countTime)
      enddo ! countReactorSite = 1, nReactorSites(countRegion)
   enddo ! region = 1, nRegions

endsubroutine sumStockInSpentFuelPools
```

```
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !
! PROGRAM: sumTransportationFlow
! DESCRIPTION: searches through all reactor sites and determines the flow
! from the sites to various offsite locations, then determines IS to Rep
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ !

subroutine sumTransportationFlow(countOriginRegion,countDestinationRegion,countTime)
use problemParameters
implicit none

   integer, intent(in) :: countOriginRegion
   integer, intent(in) :: countDestinationRegion
   integer, intent(in) :: countTime
   integer :: countRegion
   integer :: countReactorSite

   totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) = 0.0 ! initialize to zero
MT/year

   do countReactorSite = 1, nReactorSites(countOriginRegion)
      ! sum transportation to repository
      if(nRepositories.eq.nRegions .and. countDestinationRegion.eq.countOriginRegion) then
         totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) =
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) +
spentFuelPoolToRepositoryRate(countOriginRegion,countReactorSite,countTime) +
dryStorageToRepositoryRate(countOriginRegion,countReactorSite,countTime)
      elseif(countDestinationRegion.eq.repositoryRegion) then ! nRepositories.eq.1 .and. nRegions.ne.1
         totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) =
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) +
spentFuelPoolToRepositoryRate(countOriginRegion,countReactorSite,countTime) +
dryStorageToRepositoryRate(countOriginRegion,countReactorSite,countTime)
      endif ! nRepositories.eq.nRegions .and. countDestinationRegion.eq.countOriginRegion
      ! sum transportation to interim storage
      if(nInterimStorageSites.eq.nRegions .and. countDestinationRegion.eq.countOriginRegion) then
         totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) =
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) +
spentFuelPoolToInterimStorageRate(countOriginRegion,countReactorSite,countTime) +
dryStorageToInterimStorageRate(countOriginRegion,countReactorSite,countTime)
      elseif(countDestinationRegion.eq.interimStorageRegion) then
         totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) =
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) +
```

```
spentFuelPoolToInterimStorageRate(countOriginRegion,countReactorSite,countTime) +
dryStorageToInterimStorageRate(countOriginRegion,countReactorSite,countTime)
      endif ! nInterimStorageSites.eq.nRegions .and. countDestinationRegion.eq.countOriginRegion
   enddo ! countReactorSite = 1, nReactorSites(countOriginRegion)

   if(nRepositories.eq.nRegions .and. countDestinationRegion.eq.countOriginRegion) then ! this means nIS
must also equal nRegions
      totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) =
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) +
interimStorageToRepositoryRate(countOriginRegion,countTime)
   elseif(nInterimStorageSites.eq.nRegions .and. countDestinationRegion.eq.repositoryRegion) then ! will
only be called if nIS eq nRegions and nRep eq 1
      totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) =
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) +
interimStorageToRepositoryRate(countOriginRegion,countTime)
   elseif(nInterimStorageSites.eq.1 .and. nRepositories.eq.1 .and.
countOriginRegion.eq.interimStorageRegion .and. countDestinationRegion.eq.repositoryRegion) then
      totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) =
totalTransportationRate(countOriginRegion,countDestinationRegion,countTime) +
interimStorageToRepositoryRate(1,countTime)
   endif ! nInterimStorageSites.eq.nRegions

endsubroutine sumTransportationFlow
```

Appendix C:
Basic SNuFManager User's Manual for Creating Input Files

Introduction

The SNuFManager tool is a dynamically allocated Fortran90 code that reads a user defined input and provides policy makers the opportunity to quickly and cheaply simulate the costs and benefits of waste management decisions under various scenarios. Doing so with a sophisticated computer model avoids the expense and high risks associated with making poor decisions in real life. SNuFManager simulates the waste management system with two distinct solvers. The first solver utilizes the principles of System Dynamics to deterministically walk through each time step and recalculate the physical stocks and flows of the waste masses based on the conditions of the system and an unloading algorithm programmed in the simulator. By default, SNuFManager makes use of the simple unloading algorithm that prioritizes flows from reactors with the earliest shutdown dates. Once the program resolves the physics of the waste management stocks and flows, SNuFManager evaluates the recorded stocks and flows and applies the economic data provided in the user defined input file to account for annual expenditures broken down by function. Before publically distributing SNuFManager, substantial effort must go into including detailed warning messages and making the program user friendly because the current revision requires significant experience to quickly debug errors in the input.

Basic Structure

The input follows a hierarchical structure starting with the description of each repository and going all the way down to each reactor, as shown in the following figure. The input requires that each level be completely defined before moving on to any elements in a lower level. Therefore, all the repositories must be defined before any description of the interim storage, reactor sites, and reactors can be included.

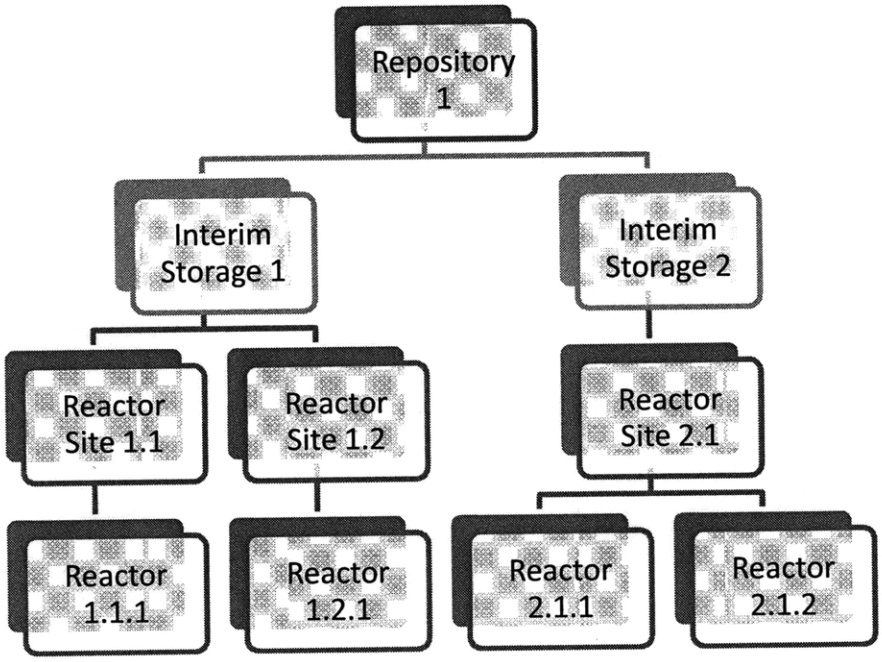


Figure 1. Hierarchy of SNuFManager Input Parameters.

It is important to point out a couple of issues right away that must be kept in mind when using this program. First, the program is hard wired to only allow the number of repositories to be equal to 1 or the number of regions. Second, the number of interim storage sites has the same requirement, but has the added constraint that the number of interim storage sites cannot be any lower than the number of repositories. Therefore, the current program cannot model a system that has four repositories and a single interim storage facility. Notice, the number of interim storage facilities cannot equal zero. Therefore, if you wish to simulate a system without interim storage, simply model the interim storage with negligible impact parameters, like a 1 MT capacity with costs at $1. These simulations will generally reach many billions of dollars, so do not be afraid of these engineering approximations. Third, for all practical purposes, the term "reactor site" should really be replaced with "spent fuel pool." The current

version of SNuFManager defines each "reactor site" with a single spent fuel pool. It is reasonable to assume all reactors on a site can share the capacity of all the spent fuel pools on the site because a moderate fraction of the nuclear facilities actually do follow this practice and because the errors introduced are negligible. In order to get around this problem, simply redefine each pool as a reactor site and define that pool's reactors appropriately. For instance, a site might have two pools, Pool 1 and Pool 2, and it might have three reactors, Reactor A, B, and C. If Reactors A and B discharge into Pool 1 and Reactor C discharges into Pool 2, simply redefine Pool 1 as its own "reactor site" containing just Reactors A and B and define Pool 2 as a separate "reactor site" containing just Reactor C. Doing this will open another problem, that the program will then no longer be able to appropriately determine the site's operational status. Right now, the SNuFManager looks at all reactors on site and says that if every one of them is shut down, the "shut down" costs apply to all remaining waste. Therefore, it is recommended that you simply follow the first approximation and combine all pool space at a reactor site and allow SNuFManager to define an artificial pool that all the reactors at a given site can share. The next revision of SNuFManager should address this issue, which is conceptually simple, but technically requires significant programming overhaul to implement the changes.

Time Constant Block

The very first two lines of the input define the time horizon and time constants as follows:

InitialTime[space]FinalTime[space]TimeStep

SpentFuelPoolCoolingTime

[blank line]

The space delimitates the input of a particular line and the lines are significant and cannot be broken apart and the blank line delimitates blocks of the input. In many cases, I provide my own comments on a particular line, which can be done after providing an additional space after the last significant input of a line. By convention, I add an exclamation point to remind myself that everything that follows is a comment. For example:

2000 2100 1.0 ! The simulation starts in 2000, ends in 2100, with a 1 year time step

5.0 ! The minimum time spent fuel must remain in wet storage is 5 years

The physical stock and flow solver is currently defined such that the time step can be any reasonable real number. For instance, 0.25 would represent a 3 month time step. However, SNuFManager still has an early version of the economic solver, which requires that the time step be 1 year because it was assumed that only annual cash flows would be needed.

Repository Blocks

There are two repository blocks and the first consists of either two or three lines, depending on the situation. As a reminder, the number of repositories must either equal 1 or the number of regions, which gets specified later. If the number of repositories equals the number of regions, no problem, but if the number of repositories only equals 1, we must specify in which region that repository resides so the correct transportation costs can be applied. The first block is as follows:

NumberOfRepositories

RepositoryLocation (this line only exists if number of repositories equals 1)

ArrayOfNumberOfExpansionsAtEachRepository

[blank line]

For example, the first block for a single repository system would look like this:

1 ! only one national repository

4 ! the repository is located in region IV, like Yucca Mountain in Nevada

1 ! the mined geologic repository is expected to only have one expansion, the initial opening

While the first block for a four borehole system might look like:

4 ! four regional repositories

5 5 5 5 ! each repository in this example has an initial opening and 4 additional expansions

The second repository block specifies the physical repository information and consists of a single line for each repository with the following information:

RepositoryNumber[space]ExpansionDateArray[space]CapacityArray[space]MaxLoadRateArray

Repeat as needed

[blank line]

So, the single mined geologic repository might look like this:

1 2020 100000.0 3000.0 ! A 100,000 MT capacity Yucca Mountain opens in 2020 receiving 3,000 MT/yr

And the four borehole repositories would be defined as:

1 2020 25000 3000 ! Let us assume repository 1 opens in 2020

2 2025 25000 3000 ! Let us assume repository 2 opens in 2025

3 2030 25000 3000 ! Let us assume repository 3 opens in 2030

4 2020 25000 3000 ! Let us assume repository 4 opens in 2020 like repository 1

Interim Storage Block

The interim storage block could be defined exactly the same way as the repository, except there are a few slight distinctions and the two-block system for the repository definition is combined into a single block for the interim storage as follows:

NumberOfInterimStorageSites

LocationOfSingleInterimStorage (only exists if first line is 1)

OpenDate[space]InitialLoad[space]Capacity[space]FillRate

Repeat previous line for each interim storage site

[blank line]

It is important to note, the program assumes that the maximum fill rate is the same as the maximum discharge rate. This subtle assumption will not likely make any difference because the repository maximum fill rate is likely less than the interim storage fill rate, so the maximum discharge will not be reached. A mined geologic repository and national interim storage facility might define this block as:

1 ! just one national interim storage site

4 ! in region IV, like the Private Fuel Storage facility in Utah

2020 0.0 40000.0 4000.0 ! open in 2020 w/o fuel initially on site. Capacity of 40,000 MT @ 4,000 MT/yr

Note, the initial loading variable was added for flexibility, but it is recommended that you always set this variable to zero. More generally, we can imagine an interim storage capacity that could expand by adding dry storage pads on site. The cost of interim storage expansion is so negligible that it is assumed there is no significant difference in simply determining the equivalent single opening of the facility.

What if we wanted to add interim storage to the four borehole strategy? Recall, the number of interim storage sites cannot be less than the number of repositories. Therefore, we must define 4 interim storage sites. Let us assume we really only wanted to have interim storage capacity in Region III because we felt that would help us bridge the gap while loading the repository. We might define this block as:

4 ! four regional interim storage facilities, but 3 are dummy facilities

2050 0.0 1.0 1.0 ! interim storage 1 is negligible

2050 0.0 1.0 1.0 ! interim storage 2 is negligible

2020 0.0 20000.0 4000.0 ! interim storage 3 is a 20,000 MT facility opening in 2020

2050 0.0 1.0 1.0 ! interim storage 4 is negligible

Defining the dummy interim storage facilities outside the time horizon will likely lead to an error when SNuFManager attempts to go in and allocate memory for the interim storage arrays. The same is true for all operations, so it is recommended that your simulation ending time be late enough such that all the waste can successfully transfer to the repository before the simulation ends.

Economics Block

This block contains all the economic variables necessary for determining the cash flows and ends by initializing the reactor information. Ideally, the economic block would exist on its own at the end of the input deck, but because the reactor parameters were generally assumed as given and largely unchanged, except for the license extension evaluation, they were placed at the bottom. The next revision of SNuFManager should reorder the input deck to follow a more logical block pattern of repository, interim storage, spent fuel pool, reactor, and finally the economics. In any event, the basic block structure is as follows:

NumberOfRegions

OnlineWetStorageCost[space]ShutDownWetStorageCost

InitialConditioningCost

OnsiteDryStorageInitialCapital[space]OnlineDryStorageCost[space]ShutDownDryStorageCost

TransportationCostMatrix

InterimStorageCapitalCost[space]ConstructionTime[space]InterestRate

InterimStorageShutdownCost[space]ClosureTime[space]InterestRate

InterimStorageBaseO&M[space]LoadingO&M[space]MonitoringO&M

FinalConditioning

Repository1ExpansionCostArray[ ]ConstrTimeArray[ ]IntRateArray[ ]ClosureCost[ ] Time[ ]IntRate

Repeat for each repository

RepositoryBaseO&M[space]LoadingO&M

ReactorSitesInEachRegionArray

ReactorsAtEachSiteInRegion1Array

Repeat for each region

The following illustrates what we might expect to see for a four borehole repository strategy:

4 ! four NRC regions represent the four regions modeled

0 10e6 ! online spent fuel pool costs are ignored, shutdown spent fuel pools cost $10M/yr to operate

150 ! conditioning for transportation and dry storage costs $150/kgIHM

5e6 1e6 5e6 ! initial dry storage costs $6M, $1M/yr online dry storage monitoring, $5M/yr shut down

5 0 0 0 ! transportation cost in $/kgIHM from region#row to region#column

0 5 0 0 ! notice non-diagonal matrix elements are zero because no shipping between those nodes

0 0 5 0 ! however, something must be entered in those spaces to fill the matrix elements

0 0 0 10

1.0 5.0 1e-9 ! dummy interim storage 1 costs $1, taking 5 years to build at 1e-9 interest rate

1.0 5.0 1e-9 ! dummy interim 1 storage shut down costs

1.0 5.0 1e-9 ! dummy interim storage 2

1.0 5.0 1e-9 ! dummy interim 2 storage shut down costs

1.0 5.0 1e-9 ! dummy interim storage 3

1.0 5.0 1e-9 ! dummy interim 3 storage shut down costs

1.0 5.0 1e-9 ! dummy interim storage 4

1.0 5.0 1e-9 ! dummy interim 4 storage shut down costs

66.0 ! final conditioning costs for borehole repository are $66/kgIHM

3e9 0.2e9 0.2e9 0.2e9 0.2e9 10 3 3 3 3 1e-9 1e-9 1e-9 1e-9 1e-9 500e6 2 1e-9 ! $3B initial, $200M expan.

3e9 0.2e9 0.2e9 0.2e9 0.2e9 10 3 3 3 3 1e-9 1e-9 1e-9 1e-9 1e-9 500e6 2 1e-9 ! $3B initial, $200M expan.

3e9 0.2e9 0.2e9 0.2e9 0.2e9 10 3 3 3 3 1e-9 1e-9 1e-9 1e-9 1e-9 500e6 2 1e-9 ! $3B initial, $200M expan.

3e9 0.2e9 0.2e9 0.2e9 0.2e9 10 3 3 3 3 1e-9 1e-9 1e-9 1e-9 1e-9 500e6 2 1e-9 ! $3B initial, $200M expan.

50e6 0.0 ! $50M/yr base O&M, $0/yr b/c we account for emplacement w/ final cond. and repackaging

20 18 19 17 ! 20 sites in region 1, 18 in region 2, 19 in region 3, 17 in region 4

1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 ! for illustrative purposes, number of reactors at each site region 1

1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 ! region 2

1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 ! region 3

1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 ! region 4

It is most important to pay attention to the units of the economic variables. Some are in $, some in $/year, some in $/kg, etc.

Reactor Blocks

There must be one block for each defined region. Within each reactor block, you must provide enough information to define the basic masses and their flows. This data is easily generated using Excel. The structure of the blocks and a simplified four region example are as follows:

RegionNumber

Site#[]CloseDateArray[]CoreMassArray[]DischargeArray[]MassInSFP[]CapacityOfSFP[]FuelInDSC[]DSCtoIS

Repeat for each reactor site in this region

[blank line]

1 ! first region, must be in numerical order

1 2020 100 25 300 800 0 0 ! off in 2020, 100 MT core, 25 MT/yr load, 300 MT in SFP w/ 800 MT capacity

2 ! second region

1 2020 2030 80 70 20 18 300 800 0 0 ! this site has two different reactors

2 2020 80 20 300 800 0 0 ! the second to last zero represents the initial spent fuel pool on site (MT)

3 ! third region

1 2030 100 25 300 800 0 0 ! the last zero represents the initial spent fuel pool to interim storage rate

4 ! fourth region

1 100 25 300 800 0 0

Appendix D:
Sample Input Files to SNuFManager Waste Management Program

Introduction

The Fortran90 code is presented in Appendix B and Appendix C is a basic user's guide to creating input files. This appendix simply gives many of the input files used to generate the results presented in the thesis and for those who wish to try models of their own, they are strongly encouraged to take an existing input file and modify it, rather than starting a new input file from scratch.

NOTE There may be instances in which lines of the input are longer than the width of the printable page and in order to present the full input files, these lines are continued on the next line. The actual input file cannot continue lines of input on a second line and the original files simply have lines that extend to nearly 200 characters in length. Most of the time, these long lines are simply the result of detailed comments

Basecase.inp

```
1998.0 2100.0 1.0 ! ınıtialTıme, fınalTıme, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be eıther 1 or equal to the number of regıons
4 ! ıf number of repositories.eq. 1, this tells whıch region reposıtory is located
1 ! number of expansions at each repository (openıng counts as an expansion)

1 2020.0 100000.0 3000.0 ! site 1, expansion date array, capacity array, max load rate array

1 ! number of ınterım storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositorıes
4 ! ıf number of ınterım storage sites.eq. 1, this tells which regıon ınterım storage ıs located
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacıty[MT], IS 1 max fıll
rate[MT/year]

4 ! number of regıons
0.0 11.3e6 ! annual wet storage cost per reactor site that ıs online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditıonıng and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! initial capıtal expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill ın unused transport costs wıth zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
1.0 5.0 1e-9 ! overnıght capıtal cost of ınterim storage 1, # yrs to amortıze, ınterest rate
1.0 5.0 1e-9 ! overnıght cost of decommıssioning and decontamination for IS 1, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSıte/yr], loadıng/unloadıng O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
274.0 ! reposıtory condıtioning and packaging cost [$/kg]
3.48e9 10.0 1e-9 680.0e6 2.0 1e-9 ! overnight repository 1 expansıon cost for expansion(1:nExp),
constr time (1:nExp), ıntrRate (1:nExp), overnight reposıtory closure cost, closure time,
ıntrRate
44.5e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/reposıtory/yr],
20 18 19 17 ! number of reactor sites in each regıon
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in regıon 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each sıte in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Regıon 1 reactor detaıls  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficıency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Clıffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrıck  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Poınt  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Lımerıck  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Mıllstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mıle Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrım  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
```

```
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6       450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1       114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6       309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040

4  ! Region 4 reactor details
```

```
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

Opening2040.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be either 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
1 ! number of expansions at each repository (opening counts as an expansion)

1 2040.0 100000.0 3000.0 ! site 1, expansion date array, capacity array, max load rate array

1 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
4 ! if number of interim storage sites.eq. 1, this tells which region interim storage is located
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max fill
rate[MT/year]

4 ! number of regions
0.0 11.3e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
1.0 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
274.0 ! repository conditioning and packaging cost [$/kg]
3.48e9 10.0 1e-9 680.0e6 2.0 1e-9 ! overnight repository 1 expansion cost for expansion(1:nExp),
constr time (1:nExp), intrRate (1:nExp), overnight repository closure cost, closure time,
intrRate
44.5e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
```

```
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040

4  ! Region 4 reactor details
```

```
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

dbhbasecase.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be either 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
11 ! number of expansions at each repository (opening counts as an expansion)

1 2020.0 2023 2026 2029 2032 2035 2038 2041 2044 2047 2050 9000.0 18000 27000 36000 45000 54000
63000 72000 81000 90000 99000 3000.0 3000 3000 3000 3000 3000 3000 3000 3000 3000 3000 ! site 1,
expansion date array, capacity array, max load rate array

1 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
4 ! if number of interim storage sites.eq. 1, this tells which region interim storage is located
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max fill
rate[MT/year]

4 ! number of regions
0.0 11.3e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
1.0 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
66.0 ! repository conditioning and packaging cost [$/kg]
2.47e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.07e9 10.0 3 3 3 3 3 3 3 3
3 3 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 680.0e6 2.0 1e-9 ! overnight
repository 1 expansion cost for expansion(1:nExp), constr time (1:nExp), intrRate (1:nExp),
overnight repository closure cost, closure time, intrRate
34.8e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
```

```
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040
```

```
4  ! Region 4 reactor details
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

Dbh2040.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be either 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
11 ! number of expansions at each repository (opening counts as an expansion)

1 2040.0 2043 2046 2049 2052 2055 2058 2061 2064 2067 2070 9000.0 18000 27000 36000 45000 54000
63000 72000 81000 90000 99000 3000.0 3000 3000 3000 3000 3000 3000 3000 3000 3000 3000 ! site 1,
expansion date array, capacity array, max load rate array

1 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
4 ! if number of interim storage sites.eq. 1, this tells which region interim storage is located
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max fill
rate[MT/year]

4 ! number of regions
0.0 11.3e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
1.0 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
66.0 ! repository conditioning and packaging cost [$/kg]
2.47e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.19e9 0.07e9 10.0 3 3 3 3 3 3 3 3
3 3 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 680.0e6 2.0 1e-9 ! overnight
repository 1 expansion cost for expansion(1:nExp), constr time (1:nExp), intrRate (1:nExp),
overnight repository closure cost, closure time, intrRate
34.8e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
```

```
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal Rıver  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robınson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor detaıls
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Bıg Rock Poınt  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davıs-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrıco Fermı 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairıe Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cıtıes  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zıon  786
400.0562364  1040  1040
```

```
4  ! Region 4 reactor details
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

Dbh4repositories.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

4 ! number of repositories... MUST be either 1 or equal to the number of regions
3 4 3 3 ! number of expansions at each repository (opening counts as an expansion)

1 2020.0 2023 2026 9000.0 18000 24000 3000.0 3000 3000 ! site 1, expansion date array, capacity
array, max load rate array
2 2020.0 2023 2026 2029 9000.0 18000 27000 29700 3000.0 3000 3000 3000
3 2020.0 2023 2026 9000.0 18000 19500 3000.0 3000 3000
4 2020.0 2023 2026 9000.0 18000 20400 3000.0 3000 3000

4 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max fill
rate[MT/year]
2050.0 0.0 1.0 1.0
2050.0 0.0 1.0 1.0
2050.0 0.0 1.0 1.0

4 ! number of regions
0.0 11.3e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
1.0 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
1.0 5.0 1e-9 ! overnight capital cost of interim storage 2, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 2, # yrs, interest
rate
1.0 5.0 1e-9 ! overnight capital cost of interim storage 3, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 3, # yrs, interest
rate
1.0 5.0 1e-9 ! overnight capital cost of interim storage 4, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 4, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
66.0 ! repository conditioning and packaging cost [$/kg]
2.47e9 0.19e9 0.13e9 10.0 3 3 1e-9 1e-9 1e-9 680.0e6 2.0 1e-9 ! overnight repository 1 expansion
cost for expansion(1:nExp), constr time (1:nExp), intrRate (1:nExp), overnight repository closure
cost, closure time, intrRate
2.47e9 0.19e9 0.19e9 0.06e9 10.0 3 3 3 1e-9 1e-9 1e-9 1e-9 680.0e6 2.0 1e-9
2.47e9 0.19e9 0.03e9 10.0 3 3 1e-9 1e-9 1e-9 680.0e6 2.0 1e-9
2.47e9 0.19e9 0.05e9 10.0 3 3 1e-9 1e-9 1e-9 680.0e6 2.0 1e-9
34.8e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
```

```
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
```

```
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palısades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040

4  ! Region 4 reactor details
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  Rıver Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

Is202040basecase.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool coolıng time[year]

1 ! number of reposıtories... MUST be eıther 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
1 ! number of expansions at each repository (opening counts as an expansion)

1 2020.0 100000.0 3000.0 ! site 1, expansion date array, capacity array, max load rate array

1 ! number of interim storage sites... MUST be eıther 1 or equal to the number of regions and
CANNOT BE less than # repositories
4 ! if number of interim storage sıtes.eq. 1, this tells which region interim storage is located
2020.0 0.0 40000.0 4000.0 ! IS 1 open date, IS 1 Initıal Fuel Load, IS 1 capacity[MT], IS 1 max
fill rate[MT/year]

4 ! number of regions
0.0 11.3e6 ! annual wet storage cost per reactor site that ıs online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! ınitial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost ın $/kg from region (row1) to region
(column1:nRegions)... must fill ın unused transport costs with zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
110.0e6 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.6e6 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
8.67e6 3.79e6 0.26 ! IS annual base O&M expenditure [$/ISSite/yr], loadıng/unloading O&M
[$/ISSite/yr], proRate O&M [$/kg/yr]
274.0 ! repository condıtioning and packaging cost [$/kg]
3.48e9 10.0 1e-9 680.0e6 2.0 1e-9 ! overnıght repository 1 expansion cost for expansion(1:nExp),
constr time (1:nExp), intrRate (1:nExp), overnight repository closure cost, closure tıme,
ıntrRate
44.5e6 0.0 ! repository annual base O&M expendıture [$/repository/yr], loadıng O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites ın each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in regıon 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fıtzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Poınt  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Lımerıck  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nıne Mıle Poınt
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrım  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
```

```
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040

4  ! Region 4 reactor details
```

```
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

Is4102020dbhbasecase.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be either 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
11 ! number of expansions at each repository (opening counts as an expansion)

1 2020.0 2023 2026 2029 2032 2035 2038 2041 2044 2047 2050 9000.0 18000 27000 36000 45000 54000
63000 72000 81000 90000 99000 3000.0 3000 3000 3000 3000 3000 3000 3000 3000 3000 3000 ! site 1,
expansion date array, capacity array, max load rate array

4 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
2020.0 0.0 10000.0 4000.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max
fill rate[MT/year]
2020.0 0.0 10000.0 4000.0
2020.0 0.0 10000.0 4000.0
2020.0 0.0 10000.0 4000.0

4 ! number of regions
0.0 11.3e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
110.0e6 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.6e6 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
110.0e6 5.0 1e-9 ! overnight capital cost of interim storage 2, # yrs to amortize, interest rate
1.6e6 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 2, # yrs, interest
rate
110.0e6 5.0 1e-9 ! overnight capital cost of interim storage 3, # yrs to amortize, interest rate
1.6e6 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 3, # yrs, interest
rate
110.0e6 5.0 1e-9 ! overnight capital cost of interim storage 4, # yrs to amortize, interest rate
1.6e6 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 4, # yrs, interest
rate
8.67e6 3.79e6 0.26 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M
[$/ISSite/yr], proRate O&M [$/kg/yr]
66.0 ! repository conditioning and packaging cost [$/kg]
2.46e9 0.18e9 0.18e9 0.18e9 0.18e9 0.18e9 0.18e9 0.18e9 0.18e9 0.18e9 0.18e9 10.0 3 3 3 3 3 3 3 3
3 3 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 680.0e6 2.0 1e-9 ! overnight
repository 1 expansion cost for expansion(1:nExp), constr time (1:nExp), intrRate (1:nExp),
overnight repository closure cost, closure time, intrRate
34.8e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
```

```
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
```

```
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040

4  ! Region 4 reactor details
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

lebasecase.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be either 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
1 ! number of expansions at each repository (opening counts as an expansion)

1 2020.0 200000.0 3000.0 ! site 1, expansion date array, capacity array, max load rate array

1 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
4 ! if number of interim storage sites.eq. 1, this tells which region interim storage is located
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max fill
rate[MT/year]

4 ! number of regions
0.0 11.3e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
150.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
6.0e6 0.64e6 5.1e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 26.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 10.0 0.00 24.0
0.00 0.00 7.00 20.0
0.00 0.00 0.00 14.0
1.0 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
274.0 ! repository conditioning and packaging cost [$/kg]
3.48e9 10.0 1e-9 680.0e6 2.0 1e-9 ! overnight repository 1 expansion cost for expansion(1:nExp),
constr time (1:nExp), intrRate (1:nExp), overnight repository closure cost, closure time,
intrRate
44.5e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2036.0  2047.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2034.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2046.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2033.0  2035.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
7  2044.0  2049.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2035.0  2045.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2046.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2032.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
15  2036.0  2040.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2046.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
```

```
17  2042.0  2044.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2034.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2032.0      43.2      9.6       450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1       114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2033.0  2034.0  2036.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2036.0  2034.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2044.0  2046.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2036.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2037.0  2041.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2046.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2041.0  2043.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2033.0  2033.0  2034.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2040.0  2041.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2041.0  2047.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2042.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2047.0  2049.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2055.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7       1.3       64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2046.0  2047.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2044.0  2046.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2046.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2034.0  2037.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2037.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2029.0  2031.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2034.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2045.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2033.0      43.2      9.6       309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2035.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2042.0  2043.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2030.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2031.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2046.0      104.5     23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2030.0  2033.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2033.0  2034.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2032.0  2032.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040

4  ! Region 4 reactor details
```

```
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2044.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2043.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2050.0  2053.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2034.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2041.0  2045.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2044.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2044.0  2045.0  2047.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2045.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2033.0  2033.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2047.0  2048.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2044.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2045.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

Lowdbhbasecase.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be either 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
11 ! number of expansions at each repository (opening counts as an expansion)

1 2020.0 2023 2026 2029 2032 2035 2038 2041 2044 2047 2050 9000.0 18000 27000 36000 45000 54000
63000 72000 81000 90000 99000 3000.0 3000 3000 3000 3000 3000 3000 3000 3000 3000 3000 ! site 1,
expansion date array, capacity array, max load rate array

1 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
4 ! if number of interim storage sites.eq. 1, this tells which region interim storage is located
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max fill
rate[MT/year]

4 ! number of regions
0.0 5.3e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
91.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
4.2e6 0.50e6 3.4e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
4.00 0.00 0.00 21.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 7.00 0.00 19.0
0.00 0.00 5.00 15.0
0.00 0.00 0.00 11.0
1.0 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
59.0 ! repository conditioning and packaging cost [$/kg]
1.73e9 82.0e6 82.0e6 82.0e6 82.0e6 82.0e6 82.0e6 82.0e6 82.0e6 82.0e6 29.0e6 10.0 3 3 3 3 3 3 3 3
3 3 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 1e-9 410.0e6 2.0 1e-9 ! overnight
repository 1 expansion cost for expansion(1:nExp), constr time (1:nExp), intrRate (1:nExp),
overnight repository closure cost, closure time, intrRate
31.3e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
```

```
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal Rıver  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuıre  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor detaıls
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Poınt  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Montıcello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairıe Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cıties  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zıon  786
400.0562364  1040  1040
```

```
4  ! Region 4 reactor details
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```

Highbasecase.inp

```
1998.0 2100.0 1.0 ! initialTime, finalTime, timeStep
5.0 ! spent fuel pool cooling time[year]

1 ! number of repositories... MUST be either 1 or equal to the number of regions
4 ! if number of repositories.eq. 1, this tells which region repository is located
1 ! number of expansions at each repository (opening counts as an expansion)

1 2020.0 100000.0 3000.0 ! site 1, expansion date array, capacity array, max load rate array

1 ! number of interim storage sites... MUST be either 1 or equal to the number of regions and
CANNOT BE less than # repositories
4 ! if number of interim storage sites.eq. 1, this tells which region interim storage is located
2050.0 0.0 1.0 1.0 ! IS 1 open date, IS 1 Initial Fuel Load, IS 1 capacity[MT], IS 1 max fill
rate[MT/year]

4 ! number of regions
0.0 12.6e6 ! annual wet storage cost per reactor site that is online and shutdown [$/RxSite/yr]
209.0 ! transportation and dry storage conditioning and packaging cost [$/kg]
8.8e6 0.79e6 9.5e6 ! initial capital expense for on-site DS, annual dry storage cost per reactor
site that is online and shutdown [$/RxSite/yr]
7.00 0.00 0.00 30.0 ! transportation cost in $/kg from region (row1) to region
(column1:nRegions)... must fill in unused transport costs with zero
0.00 10.0 0.00 27.0
0.00 0.00 8.00 22.0
0.00 0.00 0.00 16.0
1.0 5.0 1e-9 ! overnight capital cost of interim storage 1, # yrs to amortize, interest rate
1.0 5.0 1e-9 ! overnight cost of decommissioning and decontamination for IS 1, # yrs, interest
rate
1.0 1.0 0.01 ! IS annual base O&M expenditure [$/ISSite/yr], loading/unloading O&M [$/ISSite/yr],
proRate O&M [$/kg/yr]
302.0 ! repository conditioning and packaging cost [$/kg]
6.11e9 10.0 1e-9 750.0e6 2.0 1e-9 ! overnight repository 1 expansion cost for expansion(1:nExp),
constr time (1:nExp), intrRate (1:nExp), overnight repository closure cost, closure time,
intrRate
49.0e6 0.0 ! repository annual base O&M expenditure [$/repository/yr], loading O&M
[$/repository/yr],
20 18 19 17 ! number of reactor sites in each region
2 2 1 1 1 3 2 1 3 2 1 2 1 1 2 1 2 1 1 1 ! number of reactors at each site in region 1
3 2 2 1 2 1 2 1 2 2 3 2 2 1 2 2 2 2 1 ! number of reactors at each site in region 2
1 2 2 1 2 1 3 1 1 1 1 2 1 1 1 2 2 2 2
2 1 1 2 1 2 1 1 1 3 1 1 3 2 1 1 1

1  ! Region 1 reactor details  1998  3  batches  1.5  yrs/batch  50  GWD/MT  0.85  CF  0.33
efficiency      NAME  assCap  ass/core  pwr1  pwr2  pwr3
1  2016.0  2027.0    69.5  70.3    15.4  15.6    548.6  1216.1  0.0  0.0  !  Beaver Valley  1259
324  821  831
2  2034.0  2036.0    69.8  70.7    15.5  15.7    762.0  1479.2  0.0  0.0  !  Calvert Cliffs  1830
434  825  835
3  2014.0      68.8      15.3      422.6  525.1  0.0  0.0  !  Fitzpatrick  337  560  813
4  2007.0      102.1      22.7      356.4  514.5  0.0  0.0  !  Haddam Neck  153  232.148018  1207
5  2026.0      88.8      19.7      352.6  620.9  0.0  0.0  !  Hope Creek  1630  764  1049
6  1999.0  2013.0  2015.0  21.7  80.5  82.8  4.8  17.9  18.4  739.2  1233.3  0.0  0.0  !  Indian
Point  1404  435.4300254  257  951  979
7  2024.0  2029.0    96.0  96.0    21.3  21.3    653.4  1190.9  0.0  0.0  !  Limerick  2921  1528
1134  1134
8  1999.0      72.8      16.2      477.6  915.8  0.0  0.0  !  Maine Yankee  849  165.407867  860
9  1999.0  2015.0  2025.0  54.2  73.7  95.6  12.1  16.4  21.2  1029.1  1883.8  0.0  0.0  !
Millstone      641  871  1130
10  2009.0  2026.0    47.8  94.8    10.6  21.1    675.0  1206.4  0.0  0.0  !  Nine Mile Point
3679  1296  565  1120
11  2009.0      52.4      11.6      456.9  548.3  0.0  0.0  !  Oyster Creek  479  560  619
12  2033.0  2034.0    94.4  92.5    21.0  20.6    1087.2  1465.3  0.0  0.0  !  Peach Bottom  1733
1528  1116  1093
13  2012.0      55.3      12.3      364.8  564.9  0.0  0.0  !  Pilgrim  1585  580  653
14  2009.0      42.1      9.4      335.2  690.3  0.0  0.0  !  R.E. Ginna  912  121  498
15  2016.0  2020.0    71.0  71.0    15.8  15.8    706.4  1369.6  0.0  0.0  !  Salem  1460  386
839  839
16  2026.0      90.5      20.1      206.7  574.3  0.0  0.0  !  Seabrook  612  193  1070
```

```
17  2022.0  2024.0    93.5  94.0    20.8  20.9    810.2  1153.6  0.0  0.0  !  Susquehanna  1440
1528  1105  1111
18  2014.0      67.9      15.1      356.2  835.2  0.0  0.0  !  Three Mile Island  1092  177  802
19  2012.0      43.2      9.6      450.0  568.4  0.0  0.0  !  Vermont Yankee  682  368  510
20  1999.0      14.1      3.1      114.5  368.2  0.0  0.0  !  Yankee Rowe  548  32.11989975  167

2  ! Region 2 reactor details
1  2013.0  2014.0  2016.0  97.7  94.6  94.6  21.7  21.0  21.0  975.3  1542.2  0.0  0.0  !  Browns
Ferry  3717  2292  1155  1118  1118
2  2016.0  2014.0    73.8  68.6    16.4  15.2    729.6  1074.8  0.0  0.0  !  Brunswick  1719
1120  872  811
3  2024.0  2026.0    95.5  95.5    21.2  21.2    612.7  1305.7  0.0  0.0  !  Catawba  1057  386
1129  1129
4  2016.0      89.7      19.9      302.6  652.4  0.0  0.0  !  Crystal River  533  177  1060
5  2017.0  2021.0    71.0  92.1    15.8  20.5    758.7  2365.7  0.0  0.0  !  Farley  2814  314
839  1089
6  2026.0      76.2      16.9      191.4  1257.8  0.0  0.0  !  Harris  2059  157  900
7  2034.0  2038.0    72.4  74.7    16.1  16.6    929.7  1177.0  0.0  0.0  !  Hatch  1011  1277
856  883
8  2007.0      60.1      13.4      445.5  575.4  0.0  0.0  !  HB Robinson  200  157  710
9  2021.0  2023.0    93.1  93.1    20.7  20.7    764.8  1265.0  0.0  0.0  !  McGuire  694  386
1100  1100
10  2038.0  2040.0    78.3  77.6    17.4  17.2    735.1  1035.9  0.0  0.0  !  North Anna  327
314  925  917
11  2013.0  2013.0  2014.0  71.6  71.6  71.6  15.9  15.9  15.9  1415.1  1800.9  0.0  0.0  !
Oconee  723  531  846  846  846
12  2020.0  2021.0    91.4  97.7    20.3  21.7    614.5  1084.4  0.0  0.0  !  Sequoyah  616  386
1080  1155
13  2021.0  2027.0    97.7  95.2    21.7  21.2    699.2  1332.1  0.0  0.0  !  St. Lucie  1038
434  1155  1125
14  2022.0      81.7      18.2      281.2  822.5  0.0  0.0  !  Summer  900  157  966
15  2032.0  2033.0    68.5  69.0    15.2  15.3    799.0  1016.2  0.0  0.0  !  Surry  217  314
810  815
16  2032.0  2033.0    58.6  58.6    13.0  13.0    755.7  1216.2  0.0  0.0  !  Turkey Point  954
314  693  693
17  2027.0  2029.0    97.5  97.2    21.7  21.6    547.8  1696.9  0.0  0.0  !  Vogtle  1935  386
1152  1149
18  2035.0      95.2      21.2      51.7  772.1  0.0  0.0  !  Watts Bar  1289  193  1125

3  ! Region 3 reactor details
1  1999.0      5.7      1.3      64.4  120.3  0.0  0.0  !  Big Rock Point  441  49.0845987  67
2  2026.0  2027.0    98.2  97.6    21.8  21.7    454.6  1389.4  0.0  0.0  !  Braidwood  1499  386
1161  1154
3  2024.0  2026.0    98.4  95.7    21.9  21.3    583.9  1358.8  0.0  0.0  !  Byron  1198  386
1163  1131
4  2026.0      86.5      19.2      211.9  440.1  0.0  0.0  !  Clinton  1092  624  1022
5  2014.0  2017.0    64.6  70.6    14.4  15.7    848.8  1464.7  0.0  0.0  !  Cook  1415  386  764
834
6  2017.0      74.6      16.6      318.9  754.1  0.0  0.0  !  Davis-Besse  875  177  882
7  1999.0  2009.0  2011.0  16.7  71.9  71.9  3.7  16.0  16.0  1158.9  1418.2  0.0  0.0  !
Dresden  2096  1592.323372  197  850  850
8  2014.0      47.8      10.6      305.4  509.0  0.0  0.0  !  Duane Arnold  1240  368  565
9  2025.0      70.2      15.6      242.2  571.2  0.0  0.0  !  Enrico Fermi 2  2900  764  830
10  2013.0      43.2      9.6      309.2  455.2  0.0  0.0  !  Kewaunee  301  121  511
11  2015.0      82.8      18.4      0.0  38.1  0.0  0.0  !  La Crosse  0  35.16508563  979
12  2022.0  2023.0    94.0  94.0    20.9  20.9    577.4  1230.9  0.0  0.0  !  LaSalle  3953  1528
1111  1111
13  2010.0      48.9      10.9      390.8  524.7  0.0  0.0  !  Monticello  895  484  578
14  2011.0      61.8      13.7      378.2  507.9  0.0  0.0  !  Palisades  247  204  730
15  2026.0      104.5      23.2      285.5  648.3  0.0  0.0  !  Perry  1932  748  1235
16  2010.0  2013.0    43.3  43.8    9.6  9.7    577.6  708.8  0.0  0.0  !  Point Beach  149  242
512  518
17  2013.0  2014.0    44.2  44.2    9.8  9.8    594.1  764.2  0.0  0.0  !  Prairie Island  251
242  522  522
18  2012.0  2012.0    72.3  72.3    16.1  16.1    978.0  1250.3  0.0  0.0  !  Quad Cities  1438
1448  855  855
19  1999.0  1999.0    88.0  88.0    19.6  19.6    863.0  1365.2  0.0  0.0  !  Zion  786
400.0562364  1040  1040

4  ! Region 4 reactor details
```

```
1  2034.0  2018.0    70.7  72.6    15.7  16.1    781.0  1700.4  0.0  0.0  !  ANO  1956  354  836
858
2  2024.0      95.2      21.2      394.4  1230.7  0.0  0.0  !  Callaway  1524  193  1125
3  2023.0      93.7      20.8      311.4  481.8  0.0  0.0  !  Columbia  754  810.9947875  1107
4  2030.0  2033.0    97.3  97.3    21.6  21.6    367.7  1599.5  0.0  0.0  !  Comanche Peak  2100
386  1150  1150
5  2014.0      84.6      18.8      401.8  605.0  0.0  0.0  !  Cooper  829  548  1000
6  2021.0  2025.0    92.0  92.0    20.4  20.4    597.3  1195.4  0.0  0.0  !  Diablo Canyon  912
386  1087  1087
7  2033.0      40.4      9.0      268.9  379.1  0.0  0.0  !  Fort Calhoun  244  133  478
8  2024.0      40.6      9.0      524.1  620.5  0.0  0.0  !  Grand Gulf  1188  800  480
9  1999.0      5.3      1.2      24.2  28.9  0.0  0.0  !  Humboldt Bay  0  46.1541749  63
10  2024.0  2025.0  2027.0  105.2  105.2  105.5  23.4  23.4  23.4  877.0  2060.0  0.0  0.0  !
Palo Verede  3101  723  1243  1243  1247
11  2008.0      81.7      18.2      155.7  703.5  0.0  0.0  !  Rancho Seco  1080  185.7953483
966
12  2025.0      60.1      13.4      330.5  435.1  0.0  0.0  !  River Bend  532  624  710
13  1999.0  2013.0  2013.0  36.9  92.4  91.4  8.2  20.5  20.3  915.5  1313.9  0.0  0.0  !  San
Onofre  810  517.8579419  436  1092  1080
14  2027.0  2028.0    95.2  95.3    21.2  21.2    508.5  2002.2  0.0  0.0  !  South Texas  2684
386  1125  1126
15  1999.0      92.7      20.6      276.5  635.2  0.0  0.0  !  Trojan  628  210.6065283  1095
16  2024.0      91.0      20.2      315.4  875.8  0.0  0.0  !  Waterford  1144  217  1075
17  2025.0      98.6      21.9      339.7  1304.3  0.0  0.0  !  Wolf Creek  1717  193  1165
```