Using Case-Based Reasoning to Learn About Ecological Engineering

Tania R. Lanphere

Bioresource Engineering
McGill University
Montreal, Quebec

April 2009

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Abstract

Ecological engineering, the practice of designing, creating or manipulating, and monitoring ecosystems, is applied for a variety of purposes benefiting both human society and the natural environment, often integratively. While there are basic principles that help practitioners in the development and implementation process, at this time there is no comprehensive theory that guides the design of ecosystems. In order for such theory to be developed, extensive knowledge about the interactions between ecosystem constitution and comportment, and ways to analyze and integrate this knowledge, are needed. Consequently, the ability to qualitatively and quantitatively evaluate large datasets in a multivariate fashion is required. Thus, the objective of this project was to investigate the use of case-based reasoning as a method of gathering and analyzing large sets of ecological data not only for prediction but for engineering purposes, a previously untested application.

To maximize the number of cases to be analyzed without limiting the inputs to only known systems described in the literature, a virtual ecosystem and simulation platform was created. Simulation outputs and values for applied measures were compiled into a case base for use with a case-based reasoner to attempt to predict the results of several additional randomly created virtual ecosystems. Actual results were compared to the predicted results. The accuracy of the predictions made by the case-based reasoner varied, but they were more than 75% accurate 83.3% of the time. An initial attempt was made to apply this approach to "engineering" ecosystems for specified performance levels within the virtual ecosystem framework. While the targeted values of persistence were not obtained, the "engineered" virtual ecosystems were more persistent overall than the

randomly created systems, with an average ratio of 0.40527 surviving species to initial species versus an average persistence of 0.20750 for the random systems. This is indicative of the potential of this novel approach for data analysis in ecological engineering.

Résumé

Le génie des écosystèmes, soit la pratique de concevoir, créer ou manipuler, et faire la suivi des écosystèmes, s'applique, souvent de manière intégrée, au bénéfice de la société humaine et de l'environnement naturel. Quoiqu'il y ait des principes de base pouvant servir à de tels ingénieurs dans le développement et mise en œuvre de tels écosystèmes, présentement il n'existe aucune théorie compréhensive pouvant guider la conception d'écosystèmes artificiels. Afin qu'une telle théorie soit énoncée, il nous faut acquérir des connaissances approfondies quant aux interactions existant entre les constituants et le comportement de l'écosystème, et quant à comment procéder dans l'analyse et l'intégration de ces connaissances. Il devient donc nécessaire de pouvoir faire l'évaluation qualitative et quantitative de grands ensembles de données par des méthodes d'analyse multivariable. L'objectif de ce projet fut donc d'étudier l'utilisation d'une méthodologie de raisonnement par cas pour recueillir et faire l'analyse de grands ensembles de données écologiques, autant pour servir à des prédictions qu'à des fins d'ingénierie, une application préalablement inévalué.

Afin de maximiser le nombre de cas pouvant être analysés sans limiter les données d'entrée à celles décrites dans les ouvrages scientifiques, un écosystème virtuel et une plateforme de simulation furent conçus. Les données de sortie des simulations et les valeurs pour les mesures mises en œuvre furent compilés dans une base de cas conçue pour servir d'intrant à un raisonneur par cas qui servirait à prédire les résultats de plusieurs écosystèmes virtuels supplémentaires, chacun créé de façon aléatoire. Ces résultats furent comparés aux valeurs prédites. L'exactitude des prédictions du raisonneur par cas varia, mais, 83.3% du temps, dépassa 75%. Un essai préliminaire fut entrepris

pour mettre en œuvre cette démarche d'ingénierie d'écosystème pour des niveaux de performance précis dans le cadre d'un écosystème virtuel. Quoique les niveaux de persistance visés ne furent pas atteints, les écosystèmes virtuels "façonnés" furent, dans l'ensemble, plus persistants que ceux bâtis de façon aléatoire, avec un rapport moyen des espèces ayant survécu aux espèces initiales de 0.40527, comparé à 0.20750 pour les écosystèmes aléatoires. Cela met en évidence le potentiel de cette nouvelle démarche pour l'analyse de données en génie des écosystèmes.

Acknowledgements

First, of course, I wish to express my appreciation to my advisors, Robert Kok and Grant Clark, for their guidance throughout this project and the writing of this thesis. I would also like to thank my fellow graduate students, both in my research group and the department in general. Of particular help were Dr. Jennifer Karsten, Yung-Chien Sun, Marc Abbyad, Michael Muffles, Christina LaFlamme, and Dr. Heidi Webber. The staff members of the Bioresource Engineering department were also very important to my experience here, so many thanks to Susan Gregus, Abida Subhan, and Trish Singleton. Thank you all for making this a good place to work and study.

I traveled to many conferences during the course of my research and met many people, several of whom helped me develop ideas for this research. Thank you to the various members of the American Ecological Engineering Society, the Institute of Biological Engineering, the Canadian Society for Bioengineering, and the American Society of Agricultural and Biological Engineers who listened and encouraged me along my path. In particular, I want to thank the attendees of the Ecological Network Analysis Conference in Athens, Georgia in March 2005.

I would also like to thank my friends and family. To Janielle Guzinski and Élyann and Roxane Périard-Fournier: thank you for keeping me from becoming a hermit and only venturing out to go to school and the bookstore. Janielle, I also appreciate your editing help; I think I owe you ten dinners for even sending you Chapter 3. To my parents, Starr and Lois Lanphere: I cannot thank you enough for your unwavering love and support through my whole life and especially my time here. You let me believe I could do anything and never stopped me when it took me far away. Finally, I do not know what I would do without Dr. Gwen Gross: sister, friend, inspiration, shopping buddy, translator of proffesorese, shoulder to cry on, ear to bend, and all around hero.

This thesis is dedicated to the memory of Hortense Marie Lanphere (1909 – 2004), who gave me my first ecology lesson. She was a woman and scientist ahead of her time, and I will forever be grateful that she helped teach me how to see the world.

Table of Contents

List of Figures

List of Tables

List of Terms

artificial ecosystem – an ecosystem that has been at least partially constructed by human hands
ascendency – a measure combining the vigor and organization of an ecosystem; can be used as an index of ecosystem development (i.e. more developed systems will have higher ascendency)
backward knowledge – descriptive and observational knowledge of the set of causes that may result in a given effect
backward reasoning – prescription of how to achieve a given effect
case base – a set of known situations and their outcomes
case-based reasoner – a computational tool used to perform case-based reasoning
case-based reasoning (CBR) – an approach to prediction and problem solving based on previous knowledge
closed ecosystem – ecosystem that cannot receive material or energy from outside its defined boundaries
comportment – the way the in which the state of the system changes over time
composition – the types and numbers of components in a system
connectance – the ratio of actual species interactions to possible species interactions in an ecosystem; a measure of structural complexity
constitution – the combined composition and structure of a system
consumer – individual in an ecosystem that obtains energy by taking it from another individual in the system
forcing functions – factors that cause changes in an ecosystem such as temperature, energy entering the system, rainfall, disturbances, etc.
forward knowledge – descriptive and observational knowledge of causal relationships
forward reasoning – prediction of the effect of a given cause
fractal dimension – a measure of path tortuosity; geometric dimension of paths between 1 for a straight line and 2 for a path that covers an entire plane
guided ecosystem – an ecosystem that is monitored and adjusted on a periodic basis, either after a larger scale manipulation or as the whole of an ecosystem engineering effort
natural ecosystem – an ecosystem that has developed independently of intentional human influences
open ecosystem – ecosystem that can receive material and / or energy from outside its defined boundaries
persistence – the ratio of surviving species to initial species in an ecosystem; a measure of ecosystem stability
physical ecosystem – an ecosystem that exists on the same physical plane as humanity and can be observed through physical means
Pred(n) – prediction at level n, the percentage of predictions that are within n percent of the observed value; a measure for prediction accuracy
producer – individual in an ecosystem that directly absorbs energy from an outside source
species richness – the number of species in an ecosystem; a measure of compositional complexity

state – the total set of descriptors for all of the elements in a system at a given time (Note: when dealing with ecosystems, *state* may refer to a small area of the state space rather than a single point)

state space – the multi-dimensional space containing all the possible states of a given system

structure – the relationships and connections between the elements in a system

validation – the process of comparing model output to an exemplar system

verification – the process of examining the computational model to ensure that it functions as intended

virtual ecosystem – an ecosystem that exists only in cyberspace; usually constructed for a specific purpose though theoretically possible to develop spontaneously

wild ecosystem – an ecosystem that is allowed to develop without further manipulation; generally defined to start after manipulation through an ecological engineering effort

1. Introduction

Ecological engineering involves designing and creating or manipulating ecosystems such that they may provide specific services to humanity as well as themselves and the surrounding environment (Mitsch and Jørgensen 2004). In practice, the ecological engineer uses the self-organizing and regenerative properties of ecosystems to solve problems, generally those with some sort of environmental implication, in a sustainable fashion. A well engineered ecosystem will, because of its inherent functions as a living system, continue to provide services for many years with little or no human intervention. While some systems are designed for periodic adjustment and others, particularly systems restored for conservation purposes, are intended to be left to their natural process of succession, sustainability is a key feature of engineered ecosystems. Ecological engineering has been applied to waste management, pollution control, habitat restoration, sustainable harvest enhancement, and many other purposes in a variety of ecosystems, both natural and artificially created.

An example of ecological engineering that has made the news is the sinking of derelict airplanes to create habitat for fish and strata for sessile marine life. Another application is known as a "living machine", a contained facility in which a sequence of retention tanks, upon which float mats of vegetation, are used to treat wastewater. Such machines, combinations of living and mechanical parts, have seen great success in a variety of industrial and community applications (Todd 2005). One interesting area of ecological engineering is the creation of life support systems, or even full ecosystems, for human habitats in space (Morowitz 2005, Nitta 2005, Wang et al 2004, Blüm et al 2003, Allen and Nelson 1999, Salisbury et al 1997). While no ecological life support system using a

large, complex ecosystem has come into common use yet, a few well-known byproducts of the search have made their way into mainstream society. Miniature ecosystems in glass terrariums have been used as household decorations for years, and ecological engineering research has contributed a new version of that decoration: small aquatic ecosystems enclosed in a glass globe.

As awareness of environmental damage and limited resources grows, it is likely that ecological engineering will be favored increasingly over traditional 'hard' engineering solutions. Unfortunately, the state of the field does not yet allow projects to yield consistent results. Currently, projects are based mostly upon prior experience or experimentation. Some types of ecosystems have been worked with many times, and projects in them are more likely to be successful. Projects in entirely new types of systems, however, are hampered by a lack of knowledge. There are a few basic ecological principles that can help improve the design of systems in ecological engineering projects, but there is not yet any comprehensive theory to guide all projects (Mitsch and Jørgensen 2004).

Ultimately, comprehensive theory for ecological engineering could allow a practitioner to design an optimized system for a specific goal and given conditions with minimal to no experimental work. This would even be possible when the parameters of the job have never been seen before, such as would occur on an extraterrestrial planet; the species available have never been used in combination before; or even if the species have never been worked with before and the practitioner only has knowledge of the values of particular attributes of those species. Obviously, if that sort of ability is ever possible it

is far in the future, but the knowledge found during development of theory that would allow practitioners to predict the behavior of an engineered ecosystem under different conditions and allow them to choose species for a given goal, again without extensive experimentation, would be a start toward that goal and improve the success of projects in the present.

In order to develop such theory, more knowledge is required about the patterns and connections between the constitution of an ecosystem and its comportment, where *constitution* refers to the composition and structure of the ecosystem and *comportment* refers to the series of changes in values of its attributes through time due to internal interactions between elements of the system as well as response to forcing functions, such as weather, and disturbances. Ecosystems are complex systems, made up of many intricately connected parts. Small changes to the system, both from internal and external sources, can have a large impact. The results from such changes are not necessarily easily tracked though the system; nor are the causes of any changes always easy to find. For ecological engineering, there needs to be more knowledge and greater understanding of the effect of local interactions on the global dynamics of the system. Gathering such knowledge in natural systems is a difficult task given the long temporal scale of many ecological processes and the multi-scalar nature of ecosystems in general.

Much relevant knowledge already exists in the literature but has not been analyzed for ecological engineering applications. The body of existing knowledge also has not been organized and analyzed as whole, which may be one way to find the general patterns that apply to all ecosystems and form comprehensive theory for engineering ecosystems. In

the meantime, information garnered from traditional ecosystem models and virtual ecosystems can be used to form a rough theoretical basis for ecological engineering. Computational methods can also provide ways to study and examine data sets larger than those used in the past, which may be the key to developing ecological engineering theory (Mitsch and Jørgensen 2004).

1.1 Knowledge and reasoning for engineering theory

One way to map the formation of engineering theory is through a progression of required knowledge and the reasoning it enables. The first type of knowledge required is *forward knowledge*, the descriptive and observational knowledge of causal relationships that is often the pervue of scientific research. For ecological engineering, this knowledge will mostly come from the various disciplines of ecology, many studies of which also include methods for *forward reasoning*, predicting the effect of a given cause. Thus, a sufficient amount of forward knowledge of the type

IF {*constitution, forcing functions, disturbances}*

THEN {*comportment*}

allows for forward reasoning of the form

IF {*modified constitution, forcing functions, disturbances*}

THEN {*modified comportment*}.

When an adequate number of predictive rules are considered together, it is possible that patterns and relationships for ecosystems in general can be found from forward reasoning.

Backward knowledge, descriptive and observational knowledge of the set of causes that

may result in a given effect, is a necessary type of knowledge for engineering efforts but is more difficult to obtain and analyze because any single effect or outcome may have a wide variety of causes or initial conditions. The difficulty of that analysis transfers to performing *backward reasoning*, prescribing a set of adjustments or an initial system state to achieve a desired outcome. As with forward knowledge and reasoning, though, a sufficient quantity of backward knowledge of the type

IF {*sets of ecosystem constitutions, forcing functions, disturbances*}

THEN {*desired comportment*}

allows for backward reasoning in the form

IF {*required modifications*}

THEN {*desired comportment*}.

The synthesis of a number of prescriptive rules from backward reasoning may lead to general rules, and possibly even theory, for engineering ecosystems.

1.2 Research objectives and methodology

The objective of this research was to test a tool for compiling and analyzing data about ecosystems and ecological engineering efforts for its feasibility as a method of organizing forward and backward knowledge and performing forward and backward reasoning. This tool, a computational approach from the field of artificial intelligence called case-based reasoning, was used to explore the connections between ecosystem constitution and comportment as they relate to ecological engineering. A rudimentary body of knowledge, containing initial ecosystem parameters and the resulting comportments of those ecosystems when run in simulation with given forcing functions, was created with a virtual ecosystem model and simulation program and used with the

case-based reasoner. The data from a number of simulations were compiled into a case base, several measures were applied, a case-based reasoner was used to try and predict the comportments resulting from the simulation of another set of ecosystems, and the accuracy of the predictions was evaluated. The case-based reasoner was also used to try and engineer ecosystems with a higher degree of survival among the species present in the system than expected in the randomly created systems.

1.3 Conceptual framework

In order to clearly express the research and findings discussed in this thesis, there are a few terms that must be defined. The simplest term to be used here is *ecosystem*. While this term may seem simple, there are a number of issues that are important to discuss regarding ecosystems that are not as obvious. An ecosystem is most often considered to be a collection of interconnected living organisms and their habitat, generally at the landscape-scale, such as a desert or a forest, though there are many ecosystems at the microscopic scale as well. Because of the microorganisms living on and inside them, organisms themselves can also be considered ecosystems, but that is neither the common usage of the term nor is it the one being used here. The planet itself can also be considered an ecosystem, one in which many landscape-level ecosystems are connected together into one larger ecosystem, also known as the biosphere. The fact that most ecosystems on the planet are connected to other ecosystems creates difficulty in defining where the boundaries of a particular ecosystem lie. Ecosystems often gradually change from one to another with no distinctive line that can be said to be the boundary between the two. Of course, this is not as much of a problem when the ecosystem in question has been artificially created. For ecological engineering, most projects are performed on

landscape-level ecosystems with the boundaries defined by the needs of the particular project, such as land ownership and artificial boundaries like roads.

There are a number of terms that can be used to classify ecosystems, generally arranged upon sets of orthogonal axes with paired terms on each end of an axis (Molenaar 1998). *Open* and *closed* would be the one such pair of terms important to this project. This refers to whether components are able to move in or out of the ecosystem. Most ecosystems are open to energy, entropy, and materials (in the form of water, soil, migrating animals, etc.). Some systems, like remote islands and "sky islands", are closed to migrating individuals though still open to energy, entropy, and various materials. In geological or evolutionary time frames, the openness of systems often changes, so this distinction is mostly of use for the shorter term view seen in human life times. Ecosystems in space habitats would likely be closed to all materials but still open to energy and entropy. Space station ecosystems would also be *artificial*, meaning that they would be entirely planned and constructed by human hands, as opposed to *natural* systems that have originated from ecological processes without interference from humans. As human influence has reached most parts of this planet, there are very few ecosystems on Earth that are wholly natural, if human are considered to be outside of the natural system. The inclusion or exclusion of humans in the natural realm often changes with the situation, the biases of the observers, and the needs of a given project or issue. Thus, most ecosystems exist somewhere on the continuum between the two ends of the axis, and the degree of artificiality can be very crucial to planning an ecological engineering project. A set of terms that is usually more important after an ecological engineering has been implemented is *wild* and *guided*. Wild systems are allowed to

develop without interference – after the initial engineering in the case of engineered systems, while guided systems are monitored and adjusted to meet certain goals. For example, a forest may be thinned and subjected to controlled burning in order to keep down the fire danger.

Another pair of words, of particular importance to this project, is *physical* and *virtual*. It is generally considered that all ecosystems are physical, that human beings can see and touch them. However, as computing technology increases, there are a growing number of systems that exist only in computers and computer networks. There is much debate about whether computer constructs can be considered living systems. Many of these systems, though, show complex behaviors and interactions similar to physical life. They can self-replicate, and unexpected and un-programmed dynamics have been known to emerge, much the same as unexpected dynamics can emerge in physical systems. Such phenomena mean that virtual ecosystems can be treated as if they were alive, even if they are not "truly" alive. Indeed virtual ecosystems are a very useful platform for studying the processes and theoretical aspects of ecological engineering because they exist in a habitat where it is easy to measure and record all aspects of the system. They are also convenient because experiments can be performed in much less time than is often required for experiments in physical systems.

While there are other terms that can be used for classifying ecosystems, those are the pairs that are most applicable to the current project. There are also a few other terms that are applicable to all ecosystems and necessary for discussing them and their dynamics. As mentioned above, ecosystems can be described in terms of their *constitution* and

comportment (Clark 2000). The constitution of a system is its *composition*, the types of components in the system (i.e. the species, substrate, etc.), and *structure*, the set of relationships between the components. Individual components have attributes such as age and mass, and the total set of those attributes at a given time is the *state* of the system at that moment. The state of the system changes in response to internal dynamics and external influences like temperature and rainfall, known as *forcing functions*, moving through the multi-dimensional space of all possible states, the *state space*. The path traced by the system during these changes is the *comportment* of the system. It is important to note that when dealing with living systems, the term "state" has a certain degree of flexibility; it is sometimes used to refer to a small area of a state space rather than a single point.

1.4 Summary

Ecological engineering relies upon the inherent properties of ecosystems for the design and manipulation of systems in such a way as to provide services to both humanity and the environment. It has application in many industries and ecosystems, even in space. Current projects, however, are based upon experience and experimentation, not comprehensive guiding theory. In order to develop comprehensive theory, or even facilitate current projects, large quantities of knowledge about the relationships between ecosystem constitution and comportment needs to be organized and analyzed holistically. It is the purpose of this project to examine case-based reasoning as a method that could be used for that organization and analysis.

2. Literature Review

There are many examples of ecological engineering projects with a variety of purposes and techniques. As stated before, these projects are based upon experience and experiments that are mostly only applicable to that type of ecosystem. There are, however, some basic principles and methods which are considered to apply to all ecosystems. Presented here are several examples of ecological engineering projects, as well as a number of related principles. The research approaches and methods from ecology that are most applicable to ecological engineering are discussed. Also presented will be examples of the computational techniques relevant to this project: models used with ecological engineering projects, virtual ecosystems, and case-based reasoners. Due to the interdisciplinary nature of the project and the volume of knowledge across these areas of study, it is not feasible to cover all subjects in great depth herein, and thusly a brief survey of the most relevant literature is offered below.

2.1 Examples of ecological engineering

Wetlands are probably the most frequently used ecosystem type in ecological engineering projects because they can fulfill a variety of functions. One such function of wetlands is their ability to remove contaminants from water, so wetlands are often constructed or restored to treat wastewater or storm water before it enters the watershed (Bruland et al. 2003, Carleton et al. 2001, Verhoeven and Meuleman 1999). Aside from general pollution control, such treatment wetlands are being used or studied to find sustainable methods for purifying water, especially in small communities or areas that do not have access to centralized wastewater treatment (Kavanagh and Keller 2007, Mbuligwe 2005). The contaminant removal function of wetlands has also been used as part of the process in

post-mining restoration (Kalin 2001). That function also makes constructed wetlands suitable for use as a biological filter to allow the recirculation of water in sustainable shrimp aquaculture (Tilley et al. 2002).

Constructed and restored wetlands have benefits other than water treatment. They are important in preserving the biodiversity of both animals and vegetation (Finlayson et al. 2006). They create habitat, both seasonal and resting, for rare or endangered species like migratory water birds (Shuwen et al. 2001). Because wetlands can have both fresh water and salt water components, they are also home to a wide variety of vegetation adapted to periodic water logging or flooding, as well as fully aquatic conditions (Finlayson et al. 2006). There is also now interest in the possibility that properly restored or designed coastal wetlands could provide a buffer from rising sea levels and flooding due to weather events (Morris 2007). In a somewhat less ecological motivation, healthy wetlands are often considered attractive to live near, thus raising property values, and provide green space in urban settings (Nassauer 2004).

Of course, wetlands are not the only type of ecosystem associated with ecological engineering; reefs are also common targets of projects. Reefs provide a variety of ecological services, including food production, waste treatment, and disturbance regulation (Costanza 1997). There has been a high occurrence of reef deterioration in the past few decades due to both natural disturbance and anthropogenic disturbances like over-fishing and pollution, and thus artificial reefs and reef restoration projects have become common (Abelson 2006). Artificial reefs have been constructed for a number of purposes: helping in the restoration of natural reefs by providing a source for species and

individual recruitment, promoting biodiversity, and increasing habitat for commercial fish species (Perkol-Finkel and Benayahu 2007, Santos and Monteiro 2007, Abelson 2006, Powers et al. 2003). Ecological engineering is also applied when natural reefs are restored to historical or healthier appearance and function (Nestlerode 2007, Abelson 2006, Rodney and Paynter 2006, Shafir et al. 2006).

While wetlands and reefs are probably the two most common types of ecosystems involved in ecological engineering, projects and experiments have been done in a wide variety of ecosystems. For example, damaged ecosystems are left after many mining projects, and ecological engineering has been used in the restoration of these ecosystems after limestone, sand, peat, coal, topsoil, and other types of mining (Andres and Mateos 2006, Turner et al. 2006, Rochefort et al. 2003, Zhang et al. 2001, Bell 2001, Hart et al. 1999). The application of ecological engineering has also been studied for the management of water quality in seas, lakes, and rivers (Stigebrandt and Gustafsson 2007, Jørgensen 2006, Marques et al. 2003). Ecological engineering is considered to have an important role in the management and regeneration of forests, particularly tropical and harvested forests (Fulé 2002, Kozlowski 2002, Lugo 2002). Agro-ecological engineering is a category of ecological engineering that is committed to improving crop systems, pest control, erosion control, and water use in a way that also protects the environment (Hengsdijk and Van Ittersum 2003, Li et al. 1998, Zhang et al. 1998).

An interesting category of ecological engineering projects are artificially created, closed ecosystems. The study of energetically open but materially closed ecosystems is also known as *biospherics* (Allen and Nelson 1999). Some of these projects are intended to

study how our own biosphere (Earth) functions, while others are experiments regarding the creation of ecosystems or habitats in space vehicles or on other planets (Burk 1995). Clair Folsome is credited with one of the first major contributions to this field when, in 1968, he created tiny aquatic ecosystems (complete with sea water, sand, microbes, and algae) in sealed flasks that turned out to be viable systems, some for many years (Jones 1996, Nelson et al. 1993). Others since have created similar contained environments for various reasons, including a decorative version that is commercially available (Folsome and Hanson 1986, Maguire 1980, Ecosphere Associates Inc., Tucson, AZ, USA).

Biosphere 2, a 1.27 hectare facility near Oracle, AZ, USA, is probably the best known large, closed environment. Inside the steel and glass structure is a rainforest, a savannah, an ocean, a marsh, and a desert along with an area intended for intensive agriculture and a human habitat. The project was intended to explore the ability of such an environment to support biodiversity and human in space (Allen and Nelson 1999). Other closed ecosystems include the Closed Ecology Experiment Facility (CEEF, also known as the Mini-Earth and "Biosphere-J") in Rokkasho, Japan, the German Closed Equilibrated Biological Aquatic System (C.E.B.A.S.) project and its precursor AQUARACK, the Bios-3 facility in Siberia, and various Closed Ecological Life Support Systems (CELSS) projects developed by NASA as well as the European and Chinese space agencies (Morowitz 2005, Nitta 2005, Wang et al. 2004, Blüm et al. 2003, Salisbury et al. 1997).

2.2 Basic ecological engineering theory

Many of the projects discussed above are a combination of application and experiment and concentrate more on the success of specific techniques than general principles of

ecological engineering. There have been a few people, however, who have composed lists of the ecological and design principles that apply to ecological engineering. Mitsch and Jørgensen (2004) outlined nineteen principles derived primarily from ecology and ecosystem theory that were intended as a checklist of things to consider for ecological engineering projects and have gained credibility through use (Jørgensen 2006). Among these principles are reminders that ecosystems are self-designing, have interconnected and interrelated components, are connected to a variety of other systems and networks (both biological and physical), have vulnerable edges, and have their structure and function determined by forcing functions. Also included in Mitsch and Jørgensen's (2004) principles is advice to design for pulsing systems, remember that ecosystem processes have characteristic temporal and spatial scales that need to be considered in design and management plans, champion biodiversity, and couple ecosystems (e.g. an agricultural system to a natural system) whenever possible. One very important principle outlined is that ecological engineering requires a holistic approach because ecosystems are more than the sum of their parts and have emergent properties.

Kangas (2004) considered similar ecosystem features and functions but condensed them into three, more general, principles to guide projects: energy signature, self-organization, and pre-adaptation. The energy signature of an ecosystem is the set of forcing functions that influence the system comportment, including both positive and negative inputs. According to Kangas (2004), the ecological engineer must ensure that the ecosystem is designed to "match" the energy signature, i.e. the ecosystem can be supported by the energy available. Self-organization, as with Mitsch and Jørgensen's (2004) concept of self-design, is related to the fact that much of the work involved in ecological engineering

projects comes from the ecosystems instead of the human operators. This concept, of using the ecosystem itself to do some or all of the work, is one of the oldest principles elucidated and emphasized in ecological engineering (Odum 1994, 1988). Kangas' (2004) final principle, pre-adaptation, is a reminder to the practitioner that there are probably many species available that are already adapted to the conditions of a given project, and the use of those species will facilitate the self-organizational ability of a system.

The Chinese have a long participation in the development of ecological engineering, both on their own and in collaboration with western scientists (Yan et al. 1993). Influenced by ancient traditions and Chinese philosophy, as well as modern science, the Chinese principles of ecological engineering are summarized with four words: 'holism', 'harmony', 'regeneration', and 'cycling'. Again, 'holism' relates to the emergent properties of ecosystems. 'Harmony' emphasizes that good relationships and balance between parts of the system, a system's structure and function, and man and nature are necessary for successful ecological engineering. For example, the use of symbiotic relationships between system components should be used whenever possible and competitive relationships avoided. 'Regeneration' and 'cycling' both refer to ways to minimize resource use and waste production. Yan et al. (1993) proposed that the regenerative properties of ecosystems, used with careful application of technology and human intervention, can lead to greater sustainability in ecological engineering projects.

Sometimes, practitioners of the various fields that fall under the umbrella of ecological engineering will use their experience on certain projects to share ideas to improve their

fields. Weinstein et al. (2001) proposed a number of principles specific to restoration ecology, some of which are applicable to ecological engineering in general. A few of the principles are similar to those discussed above, such as reminders that ecosystems exist in a greater landscape and are self-organizing. Others, however, are more specific to the human element of ecological engineering. For example, one of the principles is regarding the importance of having realistic and clearly stated goals for a project upon which all the stakeholders have agreed. Advice regarding how to design for monitoring, and how involved monitoring should be, is also included in this set of principles.

2.3 Ecology and ecological engineering

Researchers and practitioners agree that ecological engineering is rooted in ecology and can also be a source for new theories in ecology (Kangas 2004, Mitsch and Jørgensen 2004). There are many disciplines of ecology, though, with concentrations on different hierarchical levels, spatial scales, and time scales and using a wide variety of approaches; while the nature of ecological engineering requires a fairly holistic view with information from all the hierarchal levels (Krebs 2006, Kangas 2004, Mitsch and Jørgensen 2004, Ghilarov 2001, Müller 1997). Current ecological theory has also been criticized as unevenly developed, fractured, and possibly inadequate for current applications in management (Krebs 2006, Wallington et al. 2005, Müller 1997). However, there have been some directions suggested for places to start in connecting ecological theory to ecological engineering as well as establishing connections between the disciplines.

Mitsch and Jørgensen (2004) proposed modeling and whole-ecosystem experimentation as two approaches from ecology that can be most helpful to ecological engineering design

questions. Modelling in particular allows researchers and practitioners to integrate information from a number of different disciplines, scales, and hierarchical levels, and there have been methods proposed for using modelling approaches to develop resource management plans and advance ecological engineering (Gattie et al. 2007, Grant 1998, Patten 1994). Exotic species control is another discipline that is closely related to ecological engineering, and the lessons and the approaches learned there may be easily applicable to ecological engineering (Kangas 2004). The invasion of exotic species is an excellent example of the self-organizational capacity of ecosystems, and studying exotics can provide insight into basic ecosystem structure and function. Other areas of study that are pertinent to ecological engineering include island biogeography, complexity-stability relationships, ecological economics, hierarchy theory, microcosm and mesocosm studies, ecosystem succession, and evolution (Brown et al. 2004, Kangas 2004, Mitsch and Jørgensen 2004, Odum and Odum 2003).

Systems ecology, an ecological discipline that emphasizes studying ecosystems holistically, may provide an approach and methods that will allow researchers to discover and analyze causal relationships in ecosystems that are needed for ecological engineering theory (Grant 1998). The modelling approaches advocated for ecological engineering often come from systems ecology (Gattie et al. 2007, Mitsch and Jørgensen 2004). Ecological network analysis, a methodology used by systems ecologists for obtaining information about ecosystems by examining the transfer of energy or materials through the structure of the systems' food webs, is also becoming a commonly used tool for ecological engineering and has been suggested as an integral element in advancing the field of ecological engineering (Dame and Christian 2008, Gattie et al. 2007, Gattie et al.

2006, Fath 2004).

Studying ecosystems through energy and material transfer can also be considered the domain of ecosystem ecology, though the divisions between the disciplines are fluid and researchers in the area may designate their work as belonging to either discipline. In the last few decades, ecologists have been applying principles from energetics and thermodynamics to examine ecosystems holistically, connect ecological observations to ecological theory, and investigate principles for ecological engineering (Brown et al. 2004, Jørgensen and Fath 2004, Odum and Odum 2003, Odum 2002, Svirezhev 2000, Patten 1995, Schneider and Kay 1994, Odum 1988, Gallucci 1973). This approach has had some detractors and engendered debate, both about its usefulness and how to improve its application, but is a generally accepted method of evaluating ecosystem services, function, and health (Jørgensen and Fath 2004, Odum 1995, Månsson and McGlade 1993, Patten 1993).

2.4 Ecosystem models and virtual ecosystems

The computational methods used in this project have already been proven useful in a number of fields. Models, in particular, have been used for many years in ecology to study a wide variety of the features and interactions of ecosystems and, as mentioned above, are considered one of the main tools for advancing the field of ecological engineering. Indeed there are so many models in existence that presented here are only a few recent examples of ecosystem models which are related to ecological engineering, as well as some examples of virtual ecosystems.

One use of models in ecological engineering is during the planning stage of projects. Before the commencement of closed system experiments in the Closed Ecology Experiment Facility in Japan (see section 2.1 above), a simulation model of the facility was created to explore different operational schedules (Abe et al. 2005). To improve the design of artificial reefs, Lan and Hsui (2006) proposed the DARCs (deployment of artificial reef communities) model. In DARCs, habitat complexity and budgetary constraints are taken into account in order to suggest the best, in terms of highest resulting species diversity and biomass, configuration of the reef to be deployed. A qualitative ecosystem model was also used to design an ecological engineering approach to treating the eutrophication of an aquatic system in China (Li and Guo 2000).

The use of models in planning is very common for restoration projects. For example, restoration of oyster populations has been suggested as a means of reducing phytoplankton biomass in Chesapeake Bay, and an ecosystem model was used to compare currently accepted and alternate restoration strategies (Fulford et al. 2007). Another model was used to determine which of two habitats for aquatic species would result in larger population size and greater productivity when restored, thus establishing restoration priorities (McCay and Rowe 2003). FIRESUM, an ecological process model originally used to study ecosystem change and succession under various fire regimes, was adapted to predict the future changes of a number of experimental restoration treatments in ponderosa pine forests (Covington et al. 2001). Models have also been used to study factors that affect restoration planning, such as soil properties and disturbance regimes (Baker et al. 2007, Laughlin et al. 2007).

Some of the ecosystem properties found important in the design principles discussed above (section 2.2) have also been studied with ecosystem models. Green and Sadedin (2005) review the way that various types of ecological models – cellular automata, individual-based models, and evolutionary computation – have been used to study ecosystem properties like self-organization and emergence. One such study was a cellular automaton used to simulate both the spatial and the temporal self-organization of complex landscapes (Bolliger 2005). Using an individual-based model, Parrott (2005) also studied complex spatio-temporal dynamics.

Resource management is a type of ecological engineering where ecosystems are not created but undergo periodic to constant manipulation. Ecosystem models are a very common tool for management projects, as they allow practitioners to simulate the results of various management scenarios and then choose the best one. One type of model used in resource management is the harvest optimization model. Such models have been used to increase or regulate the harvest of timber, game animals such as moose or deer, and oysters (Wam et al. 2005, Müller et al. 2004, Coen and Luckenbach 2000, Jensen 1996). Most management plans, however, are departing from straight harvest optimization and integrating general biodiversity. One such model, HARVEST, was used to project the impact of various timber harvest schedules (generated with a harvest optimization model) on the overall landscape (Gustafson et al. 2005). ELFSim is a similar model used to evaluate fish harvesting and management options in the Great Barrier Reef (Little et al. 2007). Other models include the Woody Weed Planner – a tool for managing the increasing density of woody shrubs in eastern Australia, FORECAST – a management model that includes the effects of disturbance by fires, and EDYS – a generic ecosystem

model intended for evaluating the large-scale effects of land use and land management (Noble and Walker 2006, McIntire et al. 2005, Childress et al. 2002).

Although virtual ecosystems are a fairly recent concept, coinciding with the increased use of computers for modeling and the concept of "cyber-space", the use of models of purely theoretical ecosystems for studying ecology was certainly not uncommon before virtual ecosystems. In fact, there is perhaps no difference between virtual ecosystems and theoretical models other than the intent of the researchers. For example, Marín and Delgado (2001) created a virtual ecosystem with which to test management scenarios for the krill fishery in the South Shetland Islands area. They considered their model a virtual ecosystem instead of a standard management model because they created a spatially explicit environment, using a cellular automaton initiated with physical data, but otherwise allowed to run without data input, to create possible changes in krill resources. Then they coupled that environment to models of other factors affecting the fishery for simulation.

Many virtual ecosystems are intended as computational laboratories to explore ecosystem features – of both the entire system and of individual parts of the system – without having to validate the model to a specific physical system. Instead, virtual ecosystems are created with general ecological processes and configured as needed for a given research project. One such virtual ecosystem was created to study phenotypic plasticity – the way species sometimes change behavior and morphology in response to the densities of other species – a feature often left out of other models (Peacor et al. 2007). Other virtual ecosystems have been used to explore trophic-level effects of fishing in marine

ecosystems and to learn about the effect of environmental changes on plant-insect interactions and insect movement (Gascuel 2005, Hanan et al. 2002). In a precursor to the current project, Parrot and Kok (2002, 2001) created a virtual ecosystem modeled after an artificial, closed ecosystem – such as would be found on a space station – to study the engineering and control of ecosystems.

2.5 Case-based reasoners

Case-based reasoning (CBR) is a type of decision support system that uses knowledge of previous instances to propose answers for new situations and is most useful in fields where there is not strong theory or where most decisions are based on past experience anyway (Juell and Paulson 2003). From the beginning case-based reasoners were used in diverse fields like recipe creation and meal planning, heart failure diagnosis, mediation and labor negotiation, robot navigation, and warfare decision making (Dutta et al. 1997). The use of CBR systems continues to be prevalent in the medical field. As well as various diagnostic reasoning systems, CBR is applied in many different areas of patient care (Schmidt et al. 2001). Just a few examples include reasoning systems used for helping doctors make decisions regarding patients' needs for occupational therapy, how to conduct long term and ongoing treatment, and what antibiotics are best to use as treatment for bacterial infections before lab results are obtained (Taylor et al. 2007, Rossille et al. 2005, Schmidt and Gierl 2005, Gierl et al. 2003). CBR has also been applied in medical education (Schmidt et al. 2001, Frize and Frasson 2000).

Many fields in business and industry have found ways to use CBR. Systems have been created to help in managing supply chains, projecting project costs, and making

managerial decisions (Kwon et al. 2007, Raphael et al. 2007, Sun et al. 2003). There are also CBR systems for analyzing failure and faults in both the textiles industry and metallic mechanical components (Dlodlo 2007, Jacobo 2007). CBR can also be used to help in the design process. The design of the fixtures in manufacturing, conceptual ship design, and architecture have also benefited from CBR (Kang et al. 2007, Delatte and Butler 2003, Dutta et al. 1997).

Case-based reasoning has been found useful in a number of applications in the environmental sciences as well. Planning and management, in particular, have benefited from the use of CBR systems, which have been applied in water resource management, planning and land use management for conservation districts, environmental problem solving, minimizing environmental impact in chemical process design, rangeland management, and pest control (Chen et al. 2007, Bock at al. 2005, Kaster et al. 2005, King et al. 1999, Bosch et al. 1997, Hastings et al. 1996). In often related endeavors, CBR has been used to improve the monitoring of cultivation systems, the classification of the condition of environmental systems, and facilitating species and habitat, as well as soil, mapping (Li and Yeh 2004, Núñez et al. 2004, Remm 2004, Shi et al. 2004). The predictive capabilities of CBR systems have also found use in the environmental sector. They have been used to predict possible crop injury by the application of herbicides, the performance of a constructed wetland in filtering contaminants out of water, possible risks of pesticide use, and short term air quality (Zhou et al. 2005, Lee et al. 2006, Van den Brink et al. 2002, Kalapanidas and Avouris 2001).

2.6 Summary

Reviewed above are a number of examples of ecological engineering projects. The basic principles discussed may or may not have been considered in the planning of these projects, but it remains that the designs of the ecosystems for them were based on experience and / or experimentation and not any sort of comprehensive theory that can guide any ecological engineering project. There are a number of ecological principles and studies that can contribute to forming theory for engineering ecosystems. It is the goal of this project to use some of the computational techniques reviewed above, in particular the predictive capabilities of case-based reasoning, to find ways to organize and analyze knowledge for the process of building ecological engineering theory.

3. Methodology - Virtual Ecosystem

Ecosystem models have long been acknowledged as essential tools for studying ecosystems and planning ecological engineering projects. Virtual ecosystems are similar to computational ecosystem models; they are conceptualized, designed, and implemented in much the same way. There are, however, important differences. A traditional ecosystem model used in simulation is intended to emulate or predict the comportment of an exemplar system. Such systems may be hypothetical and never actually created, but they are generally conceptualized with known agents (species or individuals thereof) interacting in known habitats or habitats that could conceivably be created my human hands. A virtual ecosystem has no exemplar system and simulation results are the output of the system itself, i.e. the computational model and the system are the same thing. The system is considered to exist as part of cyberspace – in the memory of one or more computers or in a communication network like the internet. Some programmers design a virtual "landscape" for the entities in the system to interact in, while other systems do not include such a feature and cyberspace itself is the habitat of the virtual ecosystem. The agents in the ecosystem may be designed to be similar to entities (plants, animals, etc.) from the physical world or again be purely data constructs. Similarly, the forcing functions of the system may or may not be based upon forcing functions known from physical reality.

Leaving debate about what makes life and reality to the philosophers, what makes a virtual ecosystem be an *ecosystem*, and thus usable as a research tool for questions of ecology and ecological engineering, is that at its most basic, a virtual ecosystem is still a collection of organisms interacting in a habitat. The organisms compete for resources,

some of which may be other organisms in the system, and interact in a variety of ways that affect the various populations and perhaps the habitat itself much the same as in a physical ecosystem. The resulting local dynamics and global comportment of virtual ecosystems, then, can be used to make inferences or highlight areas of study for physical ecology and ecosystems. Using theoretical or randomly created ecosystems for ecological research has been used for decades to investigate the complexity and stability of ecosystems (May 1972, Pimm and Lawton 1977). Virtual ecosystems are an updated version of that practice, using the increased computational power and simulation experience of today, as well as techniques from fields such as artificial intelligence.

As with traditional models, virtual ecosystems are very useful for predicting or studying the long term comportment of a system in a short term experiment because simulations usually run faster than real time. Furthermore, virtual ecosystems are ideal for studying system interactions in general because they are easily configurable and there is no need to validate the result of those interactions with an exemplar system, making it possible to study the processes and interactions themselves, the effect of changes to them, and the boundaries of those processes and interactions that are not necessarily found in "realistic" models.

As mentioned above, the techniques for conceptualizing and creating virtual ecosystems are similar to the techniques used for creating traditional models. Thus, there are a variety of different approaches, paired differential equations and cellular automata among others, that could have been taken. The technique chosen for the virtual ecosystem used in this project was that of object-based modeling. More specifically individual-based

modeling was chosen as a technique suitable for creating the interactions in the virtual ecosystem.

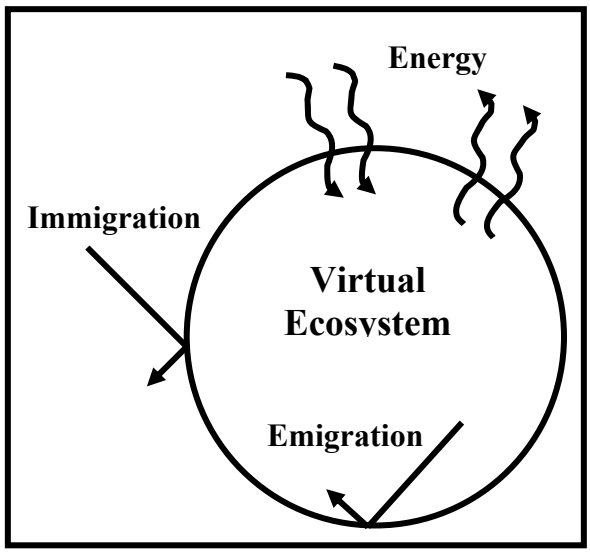


Figure 3.1 Materially Closed, Energetically Open Virtual Ecosystem

3.1 Conceptual model

The virtual ecosystem used for this study is materially closed and energetically open (Figure 3.1). It could be, perhaps, considered similar to a small island isolated far out in the ocean. Immigration and emigration are impossible, so population changes are determined only by the birth and death rates that arise from species activities and interactions. Energy enters the system in the form of radiation (sunlight), moves through the system during some system processes, and is lost to "space" through others. Almost all processes in the virtual ecosystem are in the form of energy gain, loss, or exchange. Energy is the only resource in the system; thus the rate at which energy enters the system is the primary forcing function that drives system comportment (see section 3.1.1). In all these exchanges, energy is expressed in generic "energy units" (EU).

The style of the virtual ecosystem is object based; a collection of objects interact according to certain processes and parameter values. The objects in the system are instances of various species and generally referred to as individuals. Species are divided into two types: producers, plant-like species that gain energy by absorbing radiation, and consumers, animal-like species whose individuals gain energy by 'eating' individuals of certain, preferred food species. Components have a few attributes, such as age and size,

Variable	Description	Units
AGE	Current age	s
ENERGY	Energy content	EU
MAXAGE	Maximum age	s

Table 3.1 Individual Level Attributes

Variable	Description	Units
ENERMIN	Minimum energy level (death)	EU
ENERBIR	Energy level at birth	EU
ENERREP	Reproduction threshold	EU
METAB	Specific base metabolic rate	$EU \cdot EU^{-1} \cdot s^{-1}$
MINMAXAGE	Low end of max age range	s
MAXMAXAGE	Upper end of max age range	s
AFFECT2	Health affectedness	-
ENERQUAN	Amount of energy that can be absorbed at a given time (producer only)	EU
AFFECT1	Hunting ability (consumers only)	-
FOOD	Consumer food preferences	-
INTER	Strength of species to species health interaction	-

Table 3.2 Species Level Attributes

Variable	Description	Units
ENERTOT	Total energy	EU
ENERMAX	Maximum total energy	EU
POP	Total population	indv
MAXPOP	Maximum total population	indv
NTOT	Total species	spp
MAXSPEC	Maximum number of species	spp
N1	Number of producer species	spp
N2	Number of consumer species	spp
DBLETIMES	Minimum time in which the system size can double	s
Δt	Time step size for the simulation	s
t	Time	s

Table 3.3 System Level Attributes

which are defined at the individual level (see Table 3.1), but the majority of an object's attributes and attribute values are inherited from the species level. This includes attributes such as the minimum size for each species, size at which they reproduce, and which species consumers prefer as food (see Table 3.2 for a complete list). The species level attribute values that an individual inherits determine how it is affected by the various processes in the system, which in turn affects the comportment of the species and the system as whole (see Table 3.3 for a list of system level attributes).

The virtual ecosystem is not spatially explicit. As such, there is no habitat differentiation, and the system is not given a specific size except in terms of the maximum total energy it can contain. Thus, objects in the system are not placed on a landscape as such. As a result, there is no way for species and individuals to develop niches that would influence competition for resources. In this way, the virtual ecosystem does not resemble many physical ecosystems. Rather it is more like a glass globe where all the objects in the globe can easily interact with each other and light penetrates equally to all areas (Figure 3.2). Whether an individual of a producer species receives energy from radiation, and how much energy it absorbs, is determined mostly by the attribute values of the individual's species and, to a small degree, by chance. Each individual within a species has the same likelihood of receiving energy, but each species receives a limited amount of energy to be

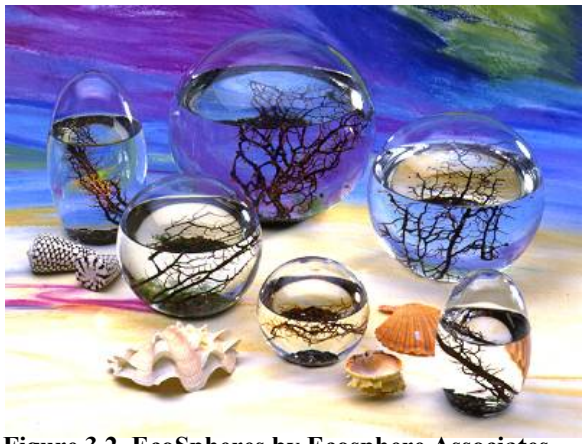


Figure 3.2 EcoSpheres by Ecosphere Associates Inc., Tucson, AZ, USA. (Photo: http://www.eco-sphere.com/

distributed among the individuals in the species. The likelihood of an individual of a consumer species eating an individual of one of its preferred food species is determined by the species attribute values of the consumer and the abundance of the food species. The consumer individual does not need to find a particular individual of the food species through a hunting or seeking method. Instead, once it has been determined that the consumer is going to eat a member of the food species, each individual in that species has the same chance of being eaten. The specific victim is then chosen randomly.

As mentioned above, the amount of energy that is available to be distributed among the members of a species is limited. The amount of energy that a producer species receives from radiation is relative to its presence in the system, so species that have already have a greater portion of the system total energy will also receive more energy from radiation. However, the balance of energy between species can change due the differing abilities of the various species to keep the energy and utilize it to reproduce, as determined by the species' attribute values. Thus, while there is no competition between members of the same species, there is competition for resources between the producer species just as is physical systems.

The representation of time is a problem somewhat unique to virtual ecosystems. In an ecosystem model, time is obviously measured in seconds, minutes, hours, days, months, and years because the system being emulated will be described in those units. A virtual ecosystem, however, can be conceptualized and simulated in generic time units if the designer so wishes. Also, although the variation of certain system functions with time is the norm in physical systems and models thereof, daily and yearly patterns are not strictly

necessary in virtual ecosystems. Nevertheless, the virtual ecosystem used in this study was intended to be as much like typically extant ecosystems in general as practical, and its functions are therefore expressed in standard time units. The passing of days and years is also tracked, so that daily and yearly variations can be observed when desired.

Reproduction in the virtual ecosystem is parthenogenetic. There are also no limitations on reproduction related to time of year or age of individuals. When an individual crosses the reproduction threshold for its species (a level of energy above which they reproduce), it creates a new instance of the species. The new individual inherits all the species' attribute values, as well as nearly all of the attribute values of the parent individual. Thus, other than the maximum possible life span for the individual, there is no "genetic" difference between the child and the parent, or any other member of the species. With no chance for genetic mutation, there is no evolution in this virtual ecosystem. Furthermore, individuals lack the ability to adapt in any way to the specific conditions of the system.

It is also important to note that there is no age-based differences between individuals. Although common in physical ecosystems, an individual's minimum energy for survival in the virtual ecosystem does not change as the individual ages. This means that the base "size" of every individual of a species is the same even though some are newborn and thus "smaller" than individuals that have been in the system for a while. How an individual interacts with other individuals and the habitat of the ecosystem also does not change with its age. Every member of a producer species has the same ability to gain energy regardless of how long it has been in the system. A "baby" consumer is just as skilled at "hunting" as its parent and has the same probability of eating a member of one

of its prey species.

The main type of interaction between components in the virtual ecosystem is predation by the consumer species. Predation is not arranged in a strict hierarchy of trophic levels, but rather in a food web. Consumers can be carnivorous, herbivorous, or omnivorous; but there are no detritivores in the system. Each consumer species has preferred food species, and individuals will eat an individual from one of those species when conditions are favorable. The rate of victim capture is in part determined by the relative abundances of the food species. There is no partial predation; any individual that is chosen as prey dies, even when it is from a producer species. When a consumer individual does not absorb all of a food individual's energy, the remaining energy is considered to be lost from the system. Also, although it is technically possible through the appropriate setting of values in the food web, cannibalism does not occur in the virtual ecosystem at present.

Besides predation, the only other direct inter-species interaction in the system is a health interaction based on relative abundances of the various species in the system. A species can have a positive, negative, or neutral effect on any species in the system, itself included. Positive and negative interactions have an impact on the amount of energy in excess of the birth size of the "baby" that a parent individual loses during reproduction. This type of interaction can mimic the effects of symbiotic relationships such as parasitism, mutualism, and commensalism. It can also be set up to add to the effect of crowding and intraspecific competition. However, the health interaction is fairly weak and has the most effect in systems that are not otherwise stable.

3.1.1 Forcing functions

The comportment of a system is usually driven by the forcing functions to which it is exposed. As mentioned above, energy is the main currency in the virtual ecosystem. Accordingly, the radiation of energy into the system is the principal forcing function. Energy comes from an outside source and is distributed among the producers in the system. The amount of energy that enters the system during a given time increment is controlled by several factors. The size of the time increment itself will obviously have an effect. For example, given two systems with identical initial states and patterns of radiation intensity but different size time increments, the power (energy per unit time) at a specific year, day, and time will be the same, but the amounts of energy will vary according to the size of the time increments. Thus, depending on the size of the time increment, the same system initial state might evolve differently in the long term.

Time of year and time of day both affect the quantity of energy entering the system during a given time step. As in physical systems, power input into the virtual ecosystem is simulated with a period of little or no energy entering the system during part of the daily cycle (night), as well as reduced power during part of the yearly cycle (winter). However, the radiation forcing function can be configured to fulfill a variety of different schemes. For example, it can be set so that a constant amount of energy enters the system in each time increment. In such a case, it would be like an ecosystem in a growth chamber with artificial lighting that can be left on all the time.

If the function is not set to deliver a constant amount of energy, radiation is an interactive forcing function; meaning that not only does it affect the system, but it responds to the

characteristics of the system in turn. When there is only a small portion of the virtual ecosystem occupied by producers, then only a small portion of the total available energy for that time increment will actually enter the system. Each producer can only absorb a certain amount of energy in a given amount of time, so any energy in excess to the total that can be absorbed never enters the system at all and, although available, is lost. This is equivalent to sunlight being 'lost' because it falls on bare ground. As the total energy of the producers in the system gets larger, so does the percentage of the total available energy that actually gets absorbed into the system. As the system nears maximum capacity, however, the proportion of the total available energy that can be absorbed drops again. Although the producers in the system could technically absorb more energy, the system has become crowded which limits the amount of energy that reaches each individual and thus how much enters the system as a whole. This is equivalent to the "island" being fully occupied.

An attenuation curve determines how the power for a given day, time, and year is adjusted in accordance with the proportion of the system occupied by producers, and thus the amount of energy that ultimately enters the system during a time increment. The equation for the attenuation curve is

$$F1 = X * (1.0 - e^{(-\alpha/X)} / e^{-\alpha}) \qquad (3.1)$$

where F1 is the attenuation factor returned for a given percentage of the upper system bound occupied by producers (X), when normalized to yield a maximum of 1.0. The shape of the curve is determined by α, the value of which can be input into the system. When the value of $\alpha = 1$, the resulting curve is symmetrical, with the highest input of energy (least attenuation) at $X = 0.500$. However, such a curve is not reasonable when

thinking in terms of system crowding and shading. After experimentation with different values, $\alpha = 4$ was chosen as resulting in a "reasonable" curve. With this curve, the maximum input occurs at approximately $X = 0.673$ (Figure 3.3).

Temperature is the second forcing function that acts upon the virtual ecosystem. As with radiation, the temperature of the virtual ecosystem at a given moment can depend on the time of day and year. However, temperature depends only on the parameters of the forcing function itself; the condition of the system has no effect on it. This is not generally true in physical systems, but factors like shading and evaporation, which can have an impact on temperature, are not taken into account in the virtual ecosystem. Although its effect is fairly minor compared to that of radiation, temperature can be very important to the survival of producer species, whose metabolic rates slow down as the temperature decreases. This mimics the semi to complete hibernative state of plants

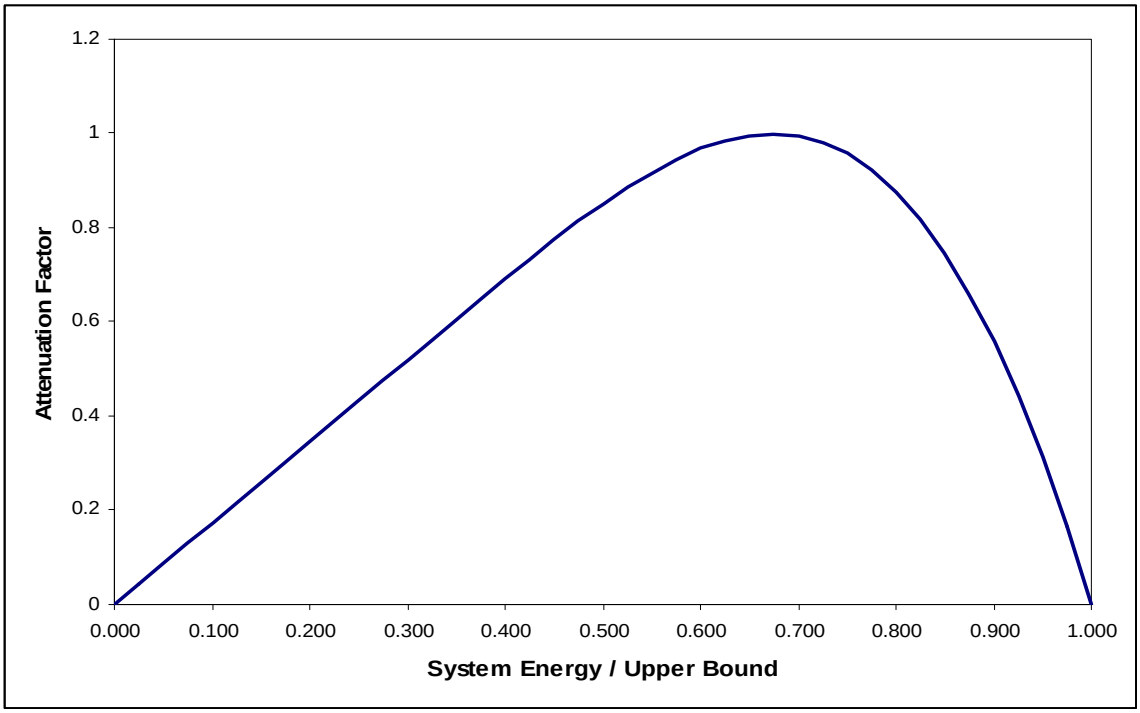


Figure 3.3 Attenuation Factor Curve for $\alpha = 4$

during the winter in physical systems. Note, however, that there are no hibernative consumer species and all consumer species keep same metabolic rate regardless of temperature. In general, cold temperatures occur during the same parts of the day and year when there is less or no energy coming into the system. Again, however, the temperature forcing function can also be set to a constant to mimic an ecosystem in an artificial environment.

A very useful feature of the type of virtual ecosystem being used in this project is that the forcing functions can easily be changed. The same parameter values for the radiation and temperature forcing functions above can be used for many simulations, or they can be changed for every simulation. The object-based style of the virtual ecosystem also makes it relatively simple to include additional forcing functions. For example, rainfall could be added and the resultant water transport in the system observed. The movement of other resources in the system could also be added. Another type of forcing function that could be included to act upon the virtual ecosystem is disturbances. Major disasters such as forest fires cause significant changes in systems and can easily be enacted in simulations of the virtual ecosystem. Less dramatic but equally important disturbances such as disease or invasive species are also possible to simulate. The study of system comportment after disturbances is an important concept for ecological engineering, and the virtual ecosystem facilitates and supports the simulation of that.

3.2 Representational model

Individuals in an object-based system like the virtual ecosystem of this study interact according to a set of rule-based expressions that, combined with the attribute values of the

Assumptions

- the virtual ecosystem consists of a number of interacting individuals
- time is represented in standard units
- the virtual ecosystem is closed to materials – there is no immigration or emigration
- the individuals in the system are incidents of various species, all individuals of which share certain attribute values
- individuals will have a maximum age and will die when they reach it
- there are no age-based differences in attribute values for individuals within a species
- individuals will have a minimum energy below which they die – the value of which depends on the species of the individual
- individuals will have an energy level above which they are able to reproduce – the value of which depends on the species of the individual
- reproduction is parthenogenetic
- individuals will lose energy when they reproduce – both the size of the new individual plus an amount determined by their health
- new individuals will be created with an energy level determined by their species
- there is no mutation – new individuals have the same species-level attribute values as their parent individuals
- the abundance of each species will affect the health of the other species in the system
- energy will enter from outside the system and be used by the individuals of some species (producers) in the system
- what portion of the entering energy an individual producer will receive is determined mostly by the attributes of its species and partially by chance
- individuals of producer species can absorb only finite amounts of energy at a given time
- individuals of some species (consumers) will gain energy by taking it from other individuals, eating
- consumers will eat approximately once a day
- individuals of a consumer species will not eat other individuals of the same species
- individuals of all consumer species have equal access to all other individuals in the system – predation success depends only on the ability of the species to capture its prey
- there are no niches within the system
- there is no partial predation
- energy entering the system and the temperature in the system will act as forcing functions for the virtual ecosystem
- both forcing function will vary on a daily and yearly basis
- the abundance of producers in the system will affect the amount of energy absorbed by the system
- the metabolic rates of individuals of producer species will be affected by the temperature in the virtual ecosystem
- individuals of consumer species will not be directly affected by the forcing functions

Table 3.4 Assumptions for the virtual ecosystem

individuals and the system, define the internal transitions of objects and the relationships between them. The expressions are executed in sequence and repeated for a number of time increments to obtain the global dynamics of the system. The sequence of execution can have an impact on the comportment. For example, if individuals metabolize an allotted amount of energy for each increment of time before they gain energy from radiation or feeding in that increment, more individuals are likely to die of starvation than if they were to gain energy first. As with most models, the magnitude of the time increment also has an impact on the system comportment. Smaller time increments lead to greater "accuracy" but increased computation time when the model is implemented in simulation.

The overall sequence of events for each increment of time used in execution of the virtual ecosystem starts with obtaining values from the forcing functions: whether or not there is energy from radiation entering the system, how much if any energy enters during the time increment, and the temperature. Any special instructions, such as disturbances not related to radiation or temperature, would also be obtained at this time. After the forcing function values for the time increment are obtained, all the individuals in the system age. For this, the value of the time increment is simply added to each individual's age (Equation 3.2).

$$AGE_t = AGE_{t-1} + \Delta t \tag{3.2}$$

If that age is then greater than the individual's maximum age, the individual dies from old age. Consequently, the population for that species, as well as the overall population of the system, is reduced by one (Equation 3.3).

$$\text{If } AGE_t > \text{MAXAGE then death, POP=POP} - 1 \tag{3.3}$$

Having aging and death by old age before death by starvation or predation allows for a clearer observation of the different types of death in the system and how each affects the comportment of the system.

The next process in the sequence is the metabolism of energy by all individuals in the system. Metabolism is performed after obtaining the forcing function values because the temperature (TEMPERAT) during any given time increment partially determines the metabolic rate of the producers during that increment (Equation 3.4).

$$ENERGY_t = ENERGY_{t-1} - ENERGY_{t-1} * METAB * \Delta t * e^{((TEMPERAT-20)/20)} \quad (3.4)$$

The metabolic energy consumption of consumers, however, is not affected by the temperature and is calculated from the individual's current energy (Equation 3.5).

$$ENERGY_t = ENERGY_{t-1} - ENERGY_{t-1} * METAB * \Delta t \quad (3.5)$$

Metabolic rate is a species level attribute, so all members of a species will lose proportionally equal amounts of their individual energy during a given time increment. Any individual that falls below the minimum energy for its species dies of starvation, and the populations are reduced as before (Equation 3.6).

$$\text{If } ENERGY_t < ENERMIN \text{ then death, } POP=POP - 1 \quad (3.6)$$

The sequence continues with the allocation of energy to producers and feeding by the consumers. If energy has entered the system during the current time increment, it is then divided among the species in proportion to their energy contents relative to the total energy of all producer species in the system. The energy allocated to each species is then broken into energy quanta, the size of which are a species level attribute, and distributed among the individuals of the species, thus increasing their individual energy levels

(Equation 3.7).

$$ENERGY_t = ENERGY_{t-1} + ENERQUAN \quad (3.7)$$

Once all energy has been distributed to the producers (or discarded) for that time increment, the consumer species begin feeding. Each consumer individual is given a chance to "hunt" if it obtains a random probability value higher than the probability that individuals of its species will not eat at all, based on the availability of the species' various preferred food species and its hunting ability (Equation 3.8).

$$\text{If Random \# > Probability of Not Feeding then feeds} \quad (3.8)$$

If the individual is able to feed during a given time increment, its prey species is determined by which of its preferred food species it has the highest chance of "finding" based on the relative energy levels of these food species. The consumer individual then eats a random individual of the prey species and gains up to a specified fraction of its previous energy, set to 10 percent in the current work (Equations 3.9 and 3.10).

$$ENERGY_t = ENERGY_{t-1} + ENERGY_{prey} \quad (3.9)$$

or

$$ENERGY_t = ENERGY_{t-1} + 0.10 * ENERGY_{t-1} \quad (3.10)$$

If the prey individual contains energy in excess of that ten percent, such energy is lost to the system. Finally, the population of the chosen prey species and the total system population are both reduced by one.

The final process of the sequence is reproduction. Individuals that have an energy level greater than the threshold to reproduce for their species will "give birth" at this point to a new individual of their species (Equation 3.11).

If ENERGY > ENERREP then reproduces, POP = POP + 1 (3.11)

The new individual will be created with the birth energy of that species, and the parent individual will lose up to two and half times that value, depending on the overall health of the species (Equation 3.12)

$$ENERGY_t = ENERGY_{t-1} - ENERBIR * (2.0-0.5 * HEALTH) \quad (3.12)$$

the value for HEALTH, between -1 and +1, being determined by the values in the species interaction matrix (XINTER) and the relative abundances of the species in the system.

3.3 Computational Model

In order to elicit the comportment of a virtual ecosystem, the representational model must be encoded in a computational model (i.e. computer program) and run in simulation. Although the model in this case was object-based, the program for a virtual ecosystem did not necessarily need to be written in an object-oriented programming language. Any language can be used, in accordance with the requirements of the virtual ecosystem and the experience of the modelers. For the virtual ecosystem used in this study, the computational model was written in FORTRAN 90/95. That language was chosen for its faster execution speed when dealing with large matrices compared to some other programming languages like C++. FORTRAN is also easily portable between different platforms and operating systems.

3.3.1 Basic program structure

The computer program used for this project encompasses both the computational model of the virtual ecosystem (i.e. the virtual ecosystem itself) and the platform that controls how simulations run (Figure 3.4). Forcing functions are included as subroutines that can

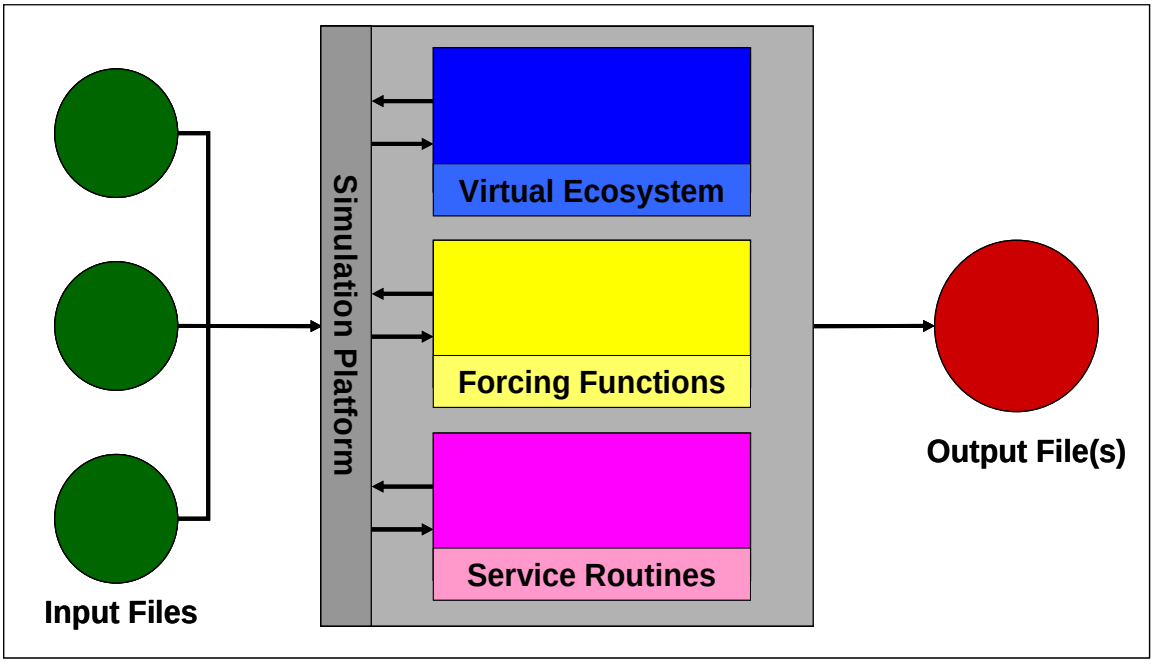


Figure 3.4 Virtual Ecosystem and Simulation Program Structure

be removed or added as needed. Services routines are interspersed throughout the program, and random number generators are included as subroutines. For each simulation, input and output is in the form of text files that can be read both by the program and by a human operator. One input file contains the definition of the constitution of each virtual ecosystem: the numerical values for the number of species, species attributes, and the interactions between them. The other input files contain parameters for the forcing functions and simulation control parameters such as the starting time, the time step magnitude, and the maximum allowed duration of the simulation. Output files can be specified to contain as much data as desired, with the main output file containing all numerical data together with textual descriptors. Secondary output files generally contain only numerical data in order to facilitate further analysis and may be direct output from the program or obtained by filtering the main output file. This approach is used for the flexibility it provides in creating virtual ecosystems, configuring simulations, and extracting the desired data. See Appendix A for

the program source code. See Appendix A for source code, Appendix B for sample input files, and Appendix C for sample output files.

3.3.2 Initialization

When a simulation of a virtual ecosystem is performed, the first step is to acquire the input data and initialize the simulation and ecosystem, starting with the simulation. In this stage, the simulation parameters are read from the appropriate input file. These parameters include the names of any output files, start day and time for the simulation, the maximum number of days the simulation can run, the length of the time increment to be used, the upper bound on total system energy, and the minimum amount of time in which the system is allowed to double in size (corresponding to the maximum growth rate possible). The maximum number of species allowed in the system and the maximum total population allowed are hard coded in the program itself. Counters for time and cycle number, as well as various matrices needed for simulation, are also initialized during this stage.

Once the simulation itself has been initialized, the virtual ecosystem needs to be initialized. Again, parameter values are read from the appropriate input file. After all the values for species and individual level attributes are acquired, establishing the structure and species composition of the system, the state of the system needs to be initialized. Using the initial population sizes given in the input file, individuals of each species are created with uniformly distributed random values, generated with pseudo-random number generator subroutines, for the individuals' energy, age, and maximum possible age. An individual's initial energy is calculated using the birth and reproduction threshold energy

values for its species, and its maximum age is determined using its species' upper and lower boundaries for maximum age. The same random number is used for initial age and energy so that there is a correlation between those values for each individual. A different random number is used for the maximum age calculation, however, to ensure that there is no correlation between the initial age of an individual and its maximum age.

At this point, the last few counters are set up and the starting day and time are adjusted so that the first time step uses the desired starting day and time and not a time step later. The total energy content of the species is calculated and compared to the total energy content of the individuals in the system to ensure that they match. Whether or not the total is below the maximum allowed energy in the system is also checked. The final step during initialization is to initialize the forcing functions. The total energy of the producer species is calculated and used in the initialization call to the weather subroutine. This is because, as mentioned in section 3.1.1, that radiation is an interactive control, and the amount of energy that enters the system at a given time is, in part, determined by the total energy of the producers in the system at that time. The main weather subroutine calls the radiation and temperature subroutines in turn, each of which sets up any counters required as well as initial matrices that contain yearly, daily, or hourly values for temperature and radiation.

3.3.3 Iteration

After a system is initialized, a sequence of events is iterated a number of times in order to elicit the system's comportment. (The sequence for the virtual ecosystem used in this study is described in detail in section 3.2.) In short: the forcing function subroutine is

called, all individuals age and then possibly die of old age, all individuals metabolize some of their stored energy and then possibly die of starvation, energy from radiation is parceled out among the producer individuals, consumer individuals get the chance to feed, and individuals over the reproduction threshold for their species reproduce. In the computational model, this process is interspersed with routines from the simulation framework that clean up matrices, check to make sure that the system has not gone outside any of the established boundaries, and do any preparatory calculations that are needed for the steps of the process. Writing to the output file(s) is also performed throughout iteration.

The sequence is repeated each time step, although some processes may be skipped, depending on the conditions of that time step. For example, if there is no radiation entering the system (i.e. it is "night"), the process by which energy is handed out to producer individuals will be skipped. Iteration continues until the simulation has reached the maximum time allowed as set in the input file, the system violates any of the system boundaries (maximum population, upper bound on energy, etc.), or all individuals in the system have died. Then a few final system statistics are printed to the main output file and all the output files are closed.

3.3.4 Forcing functions

As mentioned above, the main program calls the forcing function subroutine both during initialization and at the beginning of each time increment, and the main forcing function subroutine calls the specific forcing function subroutines in turn. The main subroutine also keeps track of errors from the secondary subroutines and reports them to the main

program. As mentioned previously (section 3.1.1), the forcing functions for this virtual ecosystem are radiation and temperature. They are included as subroutines in the program so that they can be easily changed if so desired. Early testing of the program was done with simplified routines for both radiation and temperature, but routines based on real physical weather patterns were chosen for the final version of the virtual ecosystem program.

3.3.4.1 Radiation subroutine

The model for the radiation subroutine was based on physical data from four cities in Canada (Sun and Kok 2007). On-surface, daily overall solar (DOR) energy values for a number of years were analyzed and compared to theoretical models for outside the atmosphere. Statistics for daily, weekly, and seasonal trends were examined and modeled with three principal sinusoids, a beat sinusoid, and a number of polynomials for the residual noise data in the signal. The averages and standard deviations of the principle and beat sinusoids are the first eighteen coefficients for the model, with each city studied having different values. More coefficients describe the polynomials as well as other features of the model. When tested, output from the model is statistically similar to the actual data when using the parameter values for the example cities.

As well as the coefficients for describing the polynomials, the computational model for radiation uses input values for the latitude of the target location and the descriptors for the associated polynomials. One value required for the model to work properly is the minimum DOR allowed. In essence, this sets how ‘sunny’ the virtual ecosystem being simulated will be; a simulation with a high value for the minimum DOR will likely have

more radiation overall than a simulation with a low value for the minimum DOR. The model creates one year's worth of DORs at a time. A vector of hourly attenuation factors (the degree to which solar radiation outside the atmosphere is reduced before reaching the ground) is also created on a yearly basis. A day's worth of radiation intensity values, at ten minute intervals, is generated using the hourly attenuation factors that correspond to that day (run through a routine to calculate attenuation values with ten minute intervals) and the DOR value for the day.

When the radiation subroutine is called, it tests for the present year, day, and time of the simulation. If it is a new year, the next year of DORs is created; if not, that step is skipped. If it is a new day, another day's worth of radiation intensity values is created. Then, the present simulation time is compared to the times for when there are radiation intensity values. If the time matches, that radiation intensity is returned to the main forcing function routine. An interpolated value is returned if there is not an exact time match. In the main forcing function subroutine, the radiation intensity is converted into an amount of incoming energy for the time step that will be returned to the main routine. The attenuation factor for the "size" of the system, the maximum energy allowed to enter the system in a given time step, and the size of the time step are all used in this calculation.

3.3.4.2 Temperature subroutine

As with the radiation subroutine, the model for the temperature subroutine was based on physical data (Parrott et al. 1996). Hourly temperature values for three Canadian cities were analyzed for daily, weekly, and monthly average temperature values. The

underlying cyclical characteristics of these data were described with sinusoids as in the radiation model. Again, the averages and standard deviations of the sinusoids, as well as the averages and standards deviations of the polynomial descriptors, are coefficients used when implementing the model. Synthetic data created with this model produced temperature values statistically similar to the physical temperature data of the example cities.

Parameters for the computational model of the temperature subroutine include the coefficients mentioned above as well as two variables that limit the amplitude of the daily temperature variation. Changing the values for these two variables produces different diurnal temperature shifts, resulting in different climates with otherwise identical parameters. For example, a desert and a temperate forest may have the same daily average temperature on a given day of the year, but the desert will be hotter during the day and colder at night than the forest.

During initialization, the model creates three years worth of daily average temperature (DAT) values. Three years are required because temperature is more continuous than radiation and some calculations for the first and last day of the year require data from other years. Similarly, the calculation for the temperature at a given time requires information from more than one day. The temperature subroutine uses a technique, located in a tertiary subroutine, to fit a curve called a "spline" through the average, minimum, and maximum values for three days and limited by the amplitude boundaries mentioned above. Another tertiary subroutine is used to interpolate a temperature value for a given time on the curve from the spline. Like the radiation subroutine, the

temperature subroutine tests for the year, day, and time every time it is called by the main forcing function subroutine. If it a new year, a year's worth of new DATs are created and the vector of three years of DATs is adjusted to put the three years in the correct positions of previous, current, and next (the one just created). Then a new spline is created fitting through the average, minimum, and maximum of the new three day spread (two days of which have already been used but are now in new positions). A new spline is also created when it is a new day though not a new year. These steps are skipped if it is neither a new year nor a new day, but every call finishes by returning a temperature value for the time of the call. Most of the values returned are interpolated, though the few used to calculate the spline are not.

Although the models for both the radiation and temperature subroutines were based on physical locations, simulations are not necessarily performed with the parameters for those locations. As the constitutions of ecosystems simulated with the virtual ecosystem program are not emulations of physical systems, it is not necessary for the forcing functions to emulate the weather of particular geographic locations. Also, the ability to represent a large variety of forcing function situations was desirable for this program. Thus, parameters for the forcing functions are set either to represent a specific situation (e.g. constant radiation level in a growth chamber) or created to be merely 'reasonable' for any of the many available climates possible on Earth.

3.4 Program verification

In order to ensure that the virtual ecosystem program operates both as intended and reasonably for ecosystems in general, extensive verification was performed. During this

procedure, simple ecosystems were run in simulation for a short time and detailed information was written to the output file so that the results and intermediate steps of every process could be examined. Such examination made it possible to find errors in the program code, and comportment that did not conform to general ecosystem comportment (e.g. a large predator eating several times a day). The process of verification also created familiarity with the virtual ecosystem program and a basic idea of how to initialize systems.

3.4.1 Isolating processes

The first stage of verification involved the isolation of various processes and calculations for both the simulation framework and the computational model of the virtual ecosystem. Parameter and attribute values recorded to the output file were compared to the values from the input files, and any errors in data reading / writing were corrected. One of the identical random number generators used during simulation was run separately and the results plotted to determine whether the distribution of values was uniform, as desired. The process for creating individuals was tested in much the same way; the distribution of values for a single attribute of all the individuals in each species were plotted and found to be appropriate. Mathematical processes of the simulation framework were also checked with a calculator.

A number of the ecosystem processes were isolated and studied, including metabolism of energy by all individuals. To study metabolism, a small population of individuals was created and run in simulation for a few time steps. The change in energy values from pre- to post- metabolism was compared to the value for the change when calculated with a

hand calculator. The process of producers gaining energy from the radiation forcing function was tested in much the same way. A small population of a single producer species was run in simulation and the total gain in energy by species compared to that of the total energy that was added to the system by the forcing function. The energies of individuals before and after energy distribution were compared as well to see if the energy quanta were distributed uniformly. Consumer feeding was checked by comparing feeding probabilities to those calculated with a hand calculator.

During this process, there were a number of times when the program was not found to be functioning as desired. For example, the input values for consumer hunting ability (AFFECT1) were determined to be too high when the rate of feeding was examined and it was discovered that individuals were eating more often than intended. When re-tested after the values were reduced by an order of magnitude, the process worked as desired. Another example of undesired program function found during verification was found when the reproduction of individuals was isolated and studied. The reproduction process was verified by recording births, as well as parent energy before and after birth, during a simulation. At first, parent individuals were not losing the correct amount of energy when they gave birth, and thus being able to reproduce too often. That was found to be a missing bracket in the equation used to calculate parental energy loss and easily corrected.

The forcing functions were also verified, both separately from the main program and with it. The temperature routine was broken into pieces and each section tested as the routine was re-built. First the method for creating daily average temperatures (DATs) was examined. It was used to create fifteen years worth of DATs, which were then put

through the same analysis process as the original data off which the temperature model was based. During the testing of DAT creation, another error was found. The section for creating the 'noise' in the signal did not work properly, so a new scheme for generating correlated numbers had to be found. Once that fix was performed, the temperature subroutine was rebuilt and the spline section tested. There were no other instances of undesired program function during the verification of the temperature subroutine. The radiation subroutine was examined in much the same way. The sections for creating daily overall radiation values, hourly attenuation factor values for a year, and daily radiation intensity values were all tested in turn. The test of the latter two sections required the section(s) before, so the radiation subroutine was rebuilt as each section was determined to work as foreseen.

The forcing function subroutines were then added to the main program individually and their output recorded and examined. During this part of the verification process, another fix had to be performed. The temperature subroutine performed as desired, but it was found that the radiation subroutine resulted in more energy entering the system than desired. After examining the changes in units throughout the radiation subroutine, it was discovered that the solar constant, the amount of incoming solar radiation for a given area, was required in the conversion of radiation intensity to amount of energy entering the system. Once a value for the solar constant was obtained and used in the conversion of radiation intensity to energy entering the system, the radiation subroutine performed as desired. At this point, all errors had been found and corrected, and the forcing functions worked as desired (Sun and Kok 2007, Parrott et al. 1996).

3.4.2 Exploration of simulation output

After the individual processes had been tested and verified, a number of simulations were run with various ecosystem constitutions. The purpose of these simulations was to further verify the operation of the computational model / simulation program as well as to explore the basics of setting up and running ecosystems in simulation. The output files were examined in their raw form but were also filtered to extract various types of data which were then graphed (Figure 3.5). Examining the output graphically resulted in the identification of at least one other programming problem. When the deaths by old age of one species were plotted, it was discovered that there were extremely fast decreases in the population of the species (Figure 3.5a). While cohorts dying at approximately the same time of year is a common phenomenon in physical ecosystems, it was badly exaggerated in this case, with a large portion of the population dying on the same day. The sudden reduction in the population in one species would then affect the other species in the system, sometimes resulting in a catastrophic event. This problem, which was discovered through the examination of the graph, was determined to be due to inappropriate choice of seeds for the random number generators used to determine new individuals. The seeds being used up to that point were sequential, which causes certain harmonics to occur when used with the random number generation subroutines in this program. Once the seeds were adjusted to widely different values, the problem no longer occurred (Figure 3.5b).

Graphs of output data were also used to examine particular sections of the program, both during the isolation of processes discussed above and during a step by step build-up of an ecosystem with the same forcing function parameters being used for each simulation.

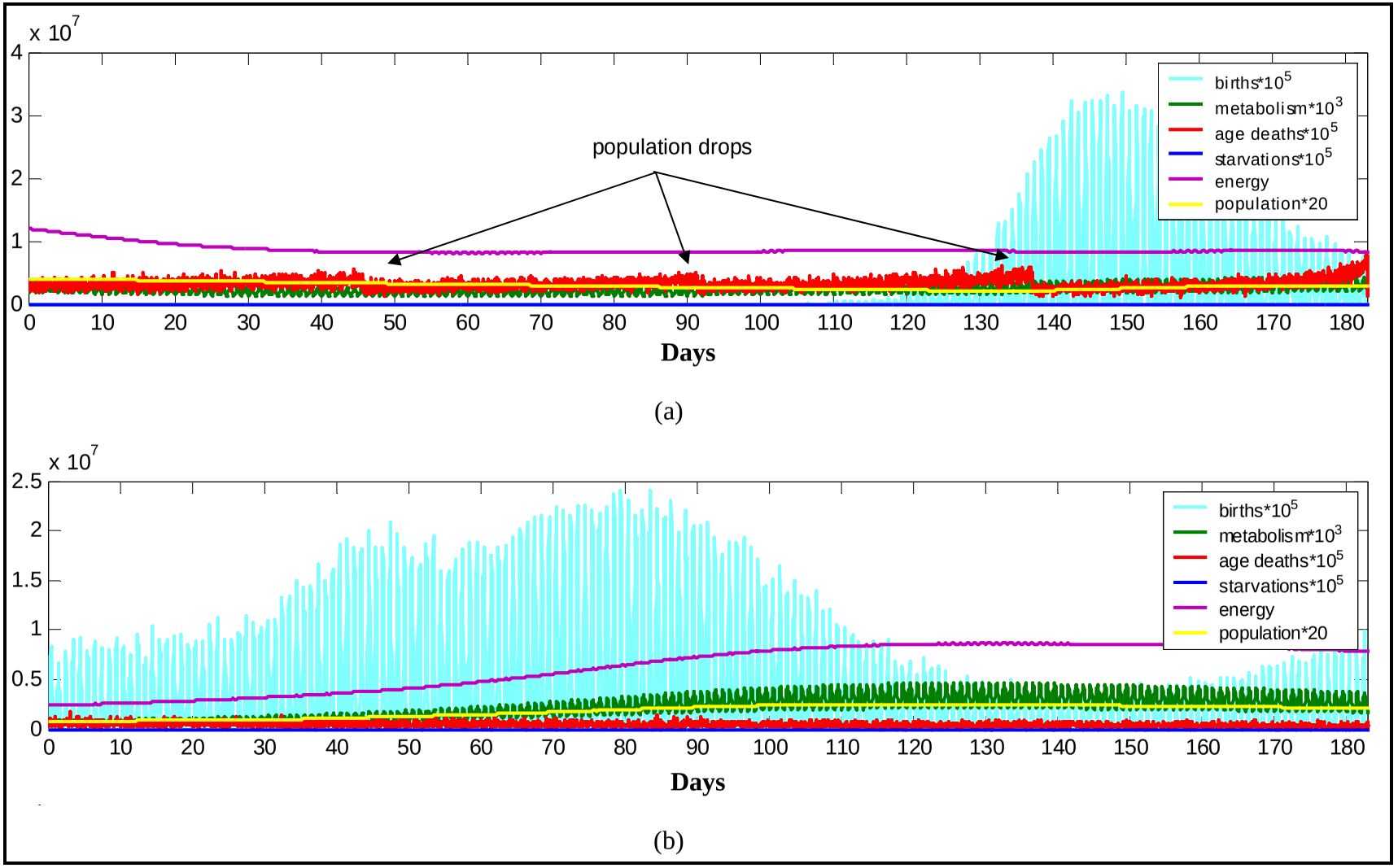


Figure 3.5 Species Variables vs. Time

The step by step approach was used to see if the output in each situation would be reasonable for that situation. The first version of the system had a single producer species that had no health interactions with itself. It was run in several simulations with different values for the species' baseline metabolism. The final populations of the simulations were then plotted versus the corresponding base metabolic rate values (Figure 3.6). As expected, there was a point at which the baseline metabolism of the species became so high that all the individuals died of either old age or starvation before they could reproduce, resulting in extinction for the species and the system overall. In the next step, the species was given various values for health interaction with itself, and then simulated in the same manner for each health interaction value. Though the point where the species' metabolism became critical was approximately the same for all interaction values, the graph made it easy to see that, as expected, negative interaction values made the population drop faster and positive interaction values bolstered the population for a time (Figure 3.7).

The system build-up was continued with the addition of a second, but identical, producer species. The initial population from the previous simulations was split between the two species and the system was then run in simulation with a variety of base metabolic rates. As anticipated, the populations of the two species stayed approximately the same throughout the simulations, indicating that the system was behaving appropriately. As before, the species went extinct when their metabolism reached a certain base rate (Figure 3.8). Then another set of simulations were performed with several different health interaction matrices and varying base metabolic rates. Though the two species still reached extinction with the same metabolic rates, their populations were no longer the

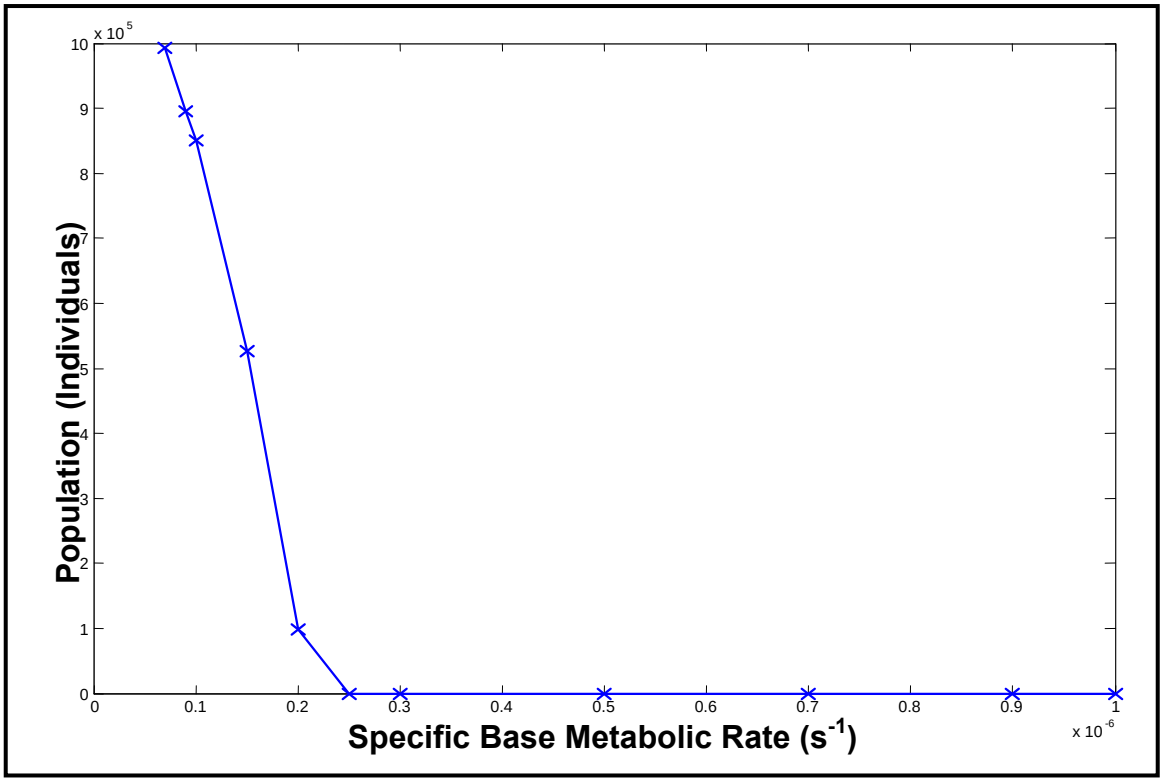


Figure 3.6 Average Population of Final Year vs. Specific Base Metabolic Rate

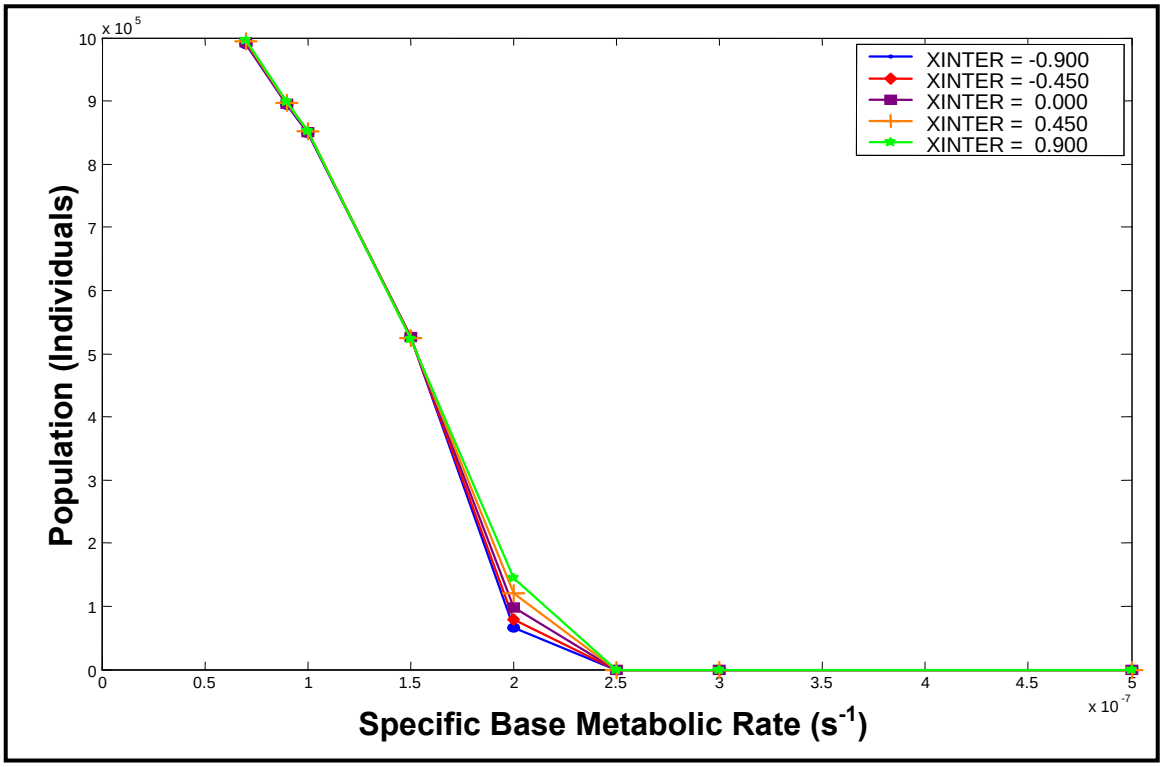


Figure 3.7 Average Population of Final Year vs. Specific Base Metabolic Rate for five interaction values

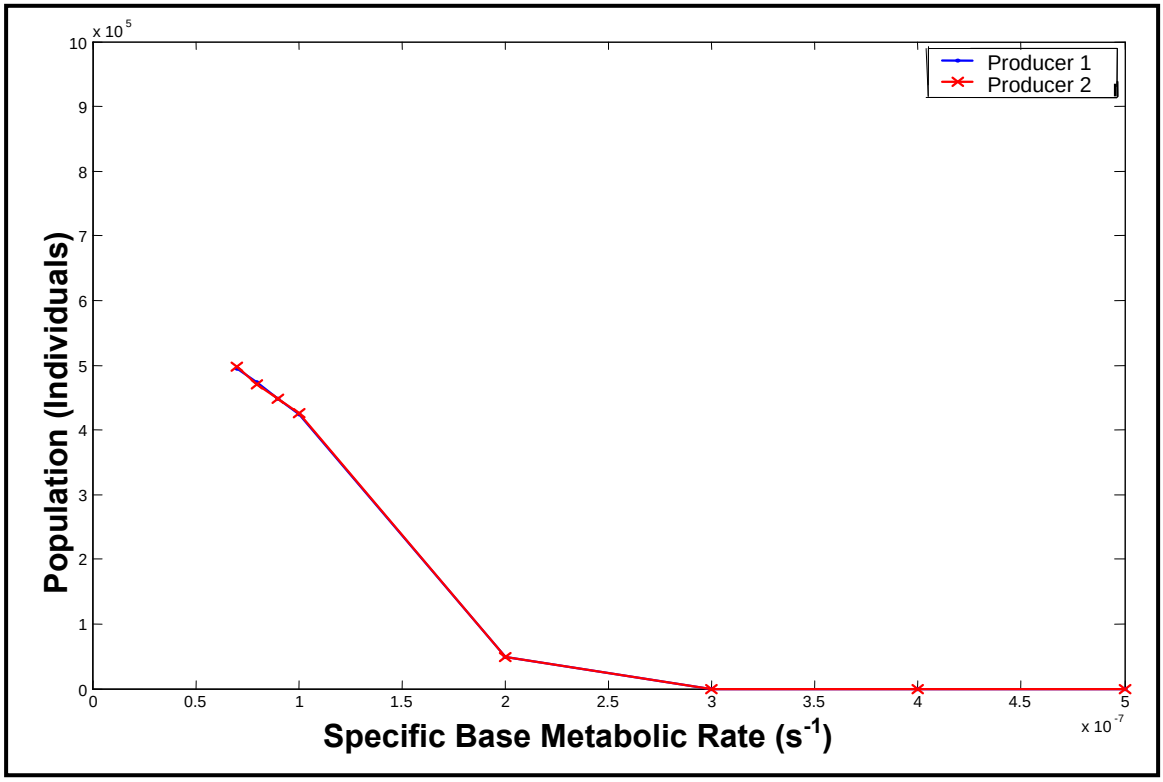


Figure 3.8 Average Population of Final Year vs. Specific Base Metabolic Rate

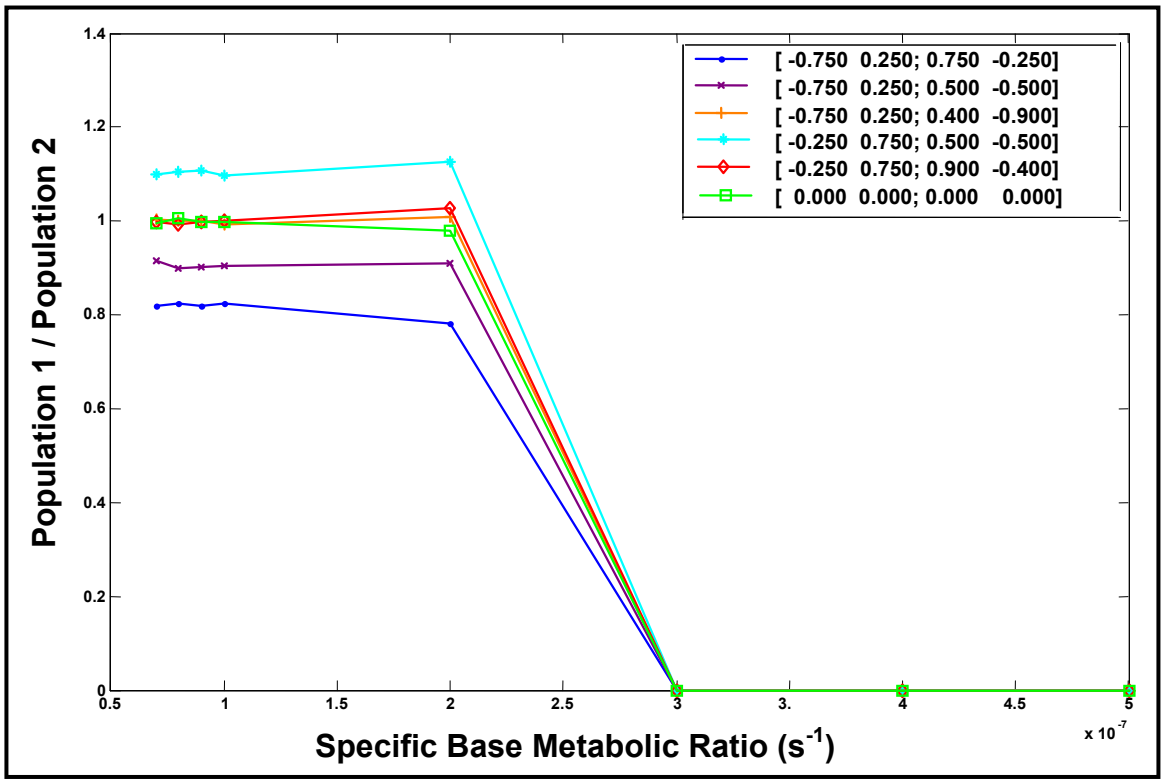


Figure3.9 Ratio of Average Populations of Final Year vs. Specific Base Metabolic Rate for different interaction matrices

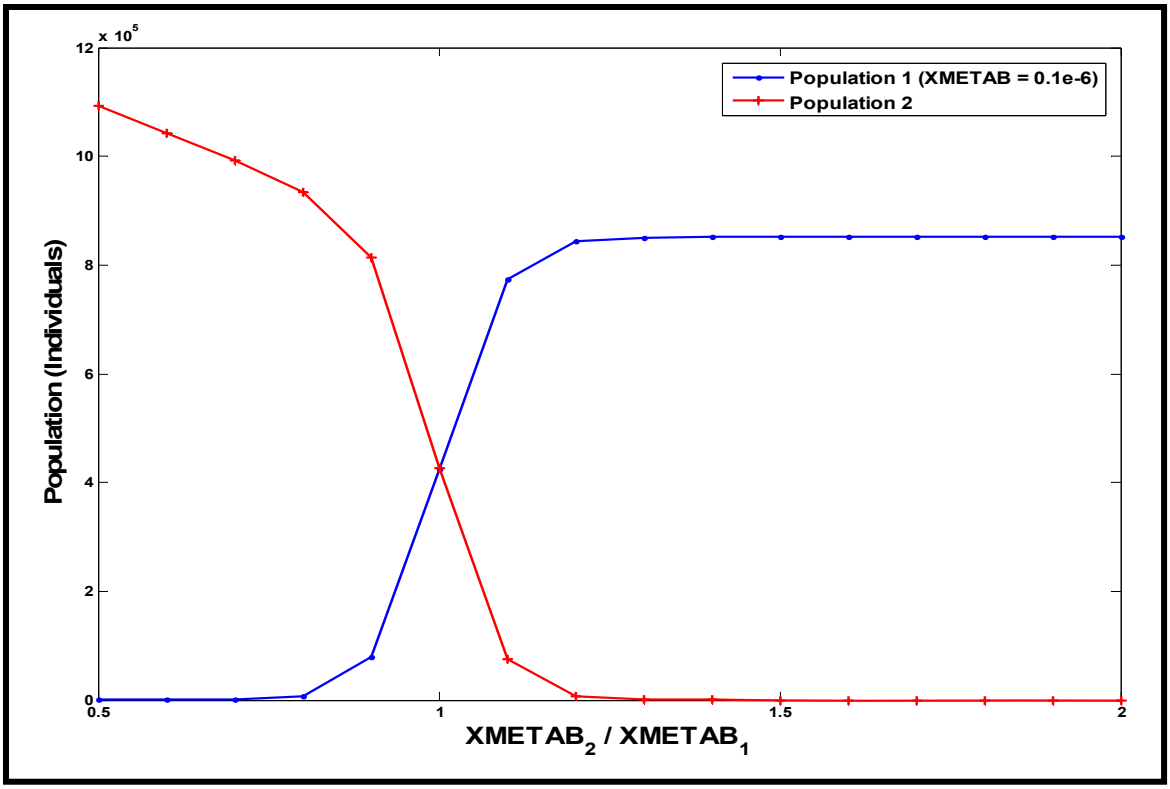


Figure 3.10 Average Population of Final Year vs. Ratio of Specific Base Metabolic Rates

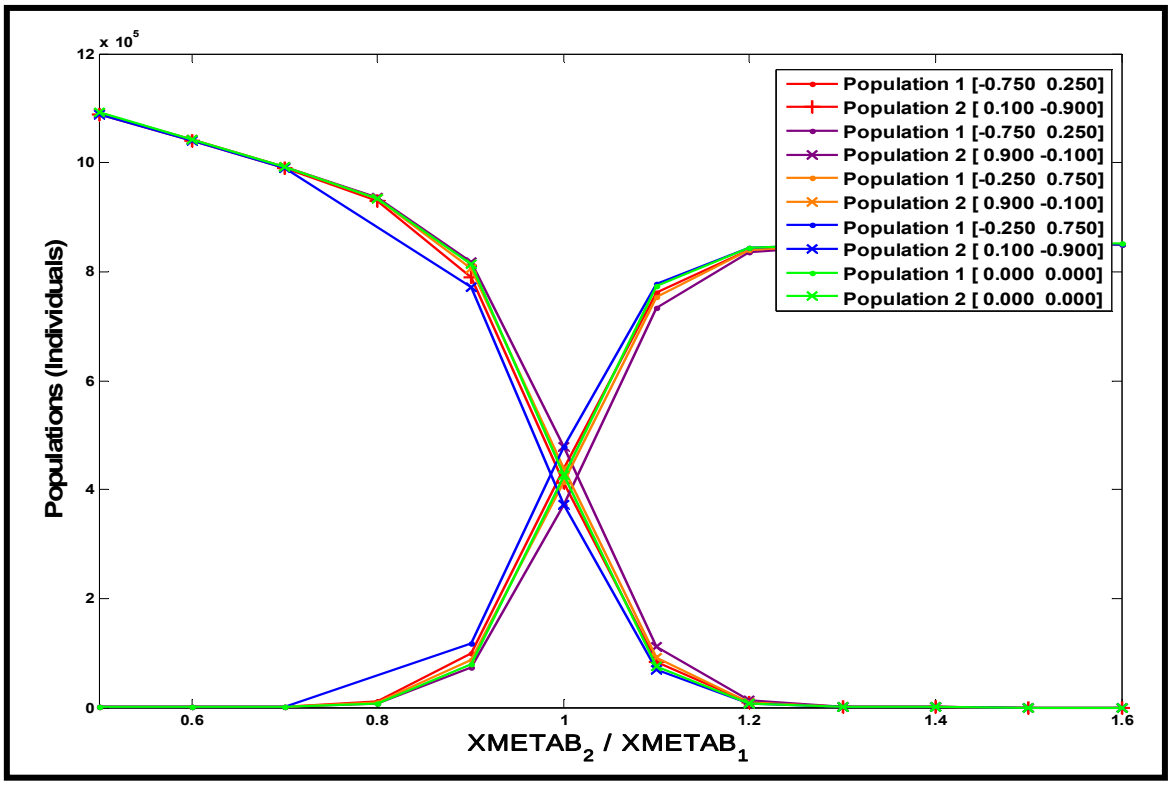


Figure 3.11 Average Population of Final Year vs. Ratio of Specific Base Metabolic Rates for different interaction matrices

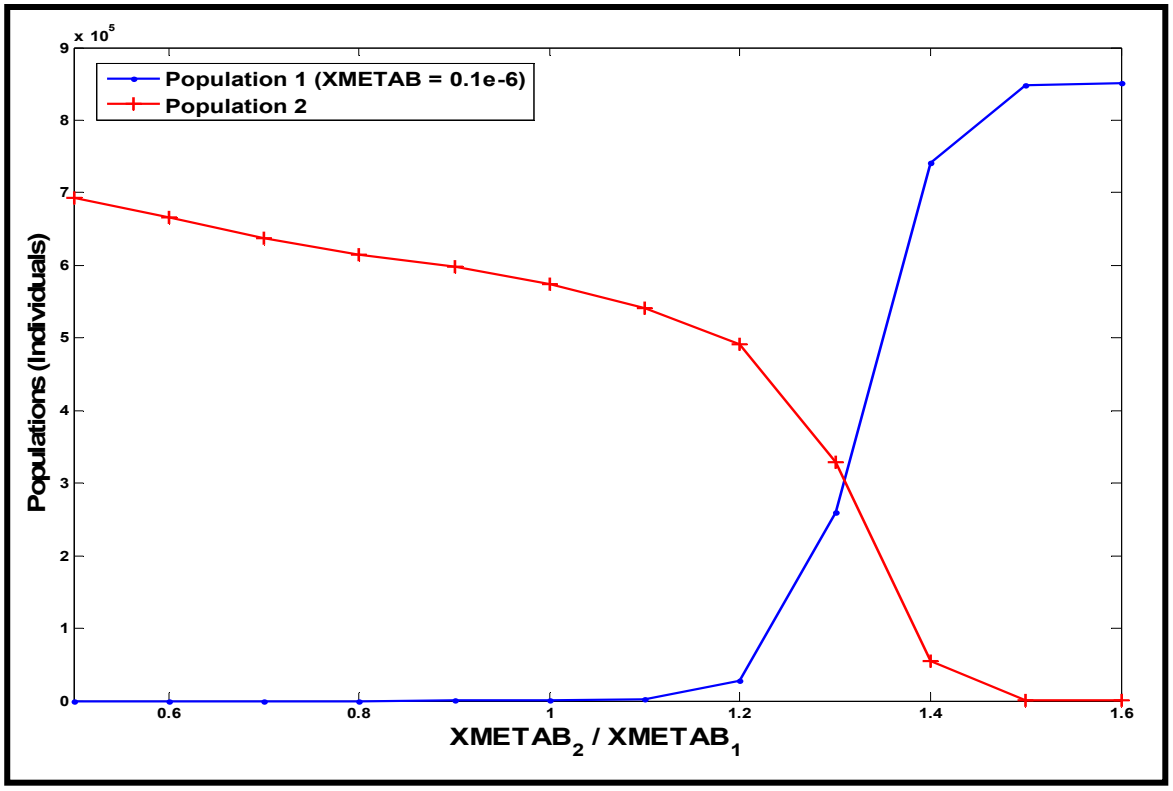


Figure 3.12 Average Population of Final Year vs. Ratio of Specific Base Metabolic Rates

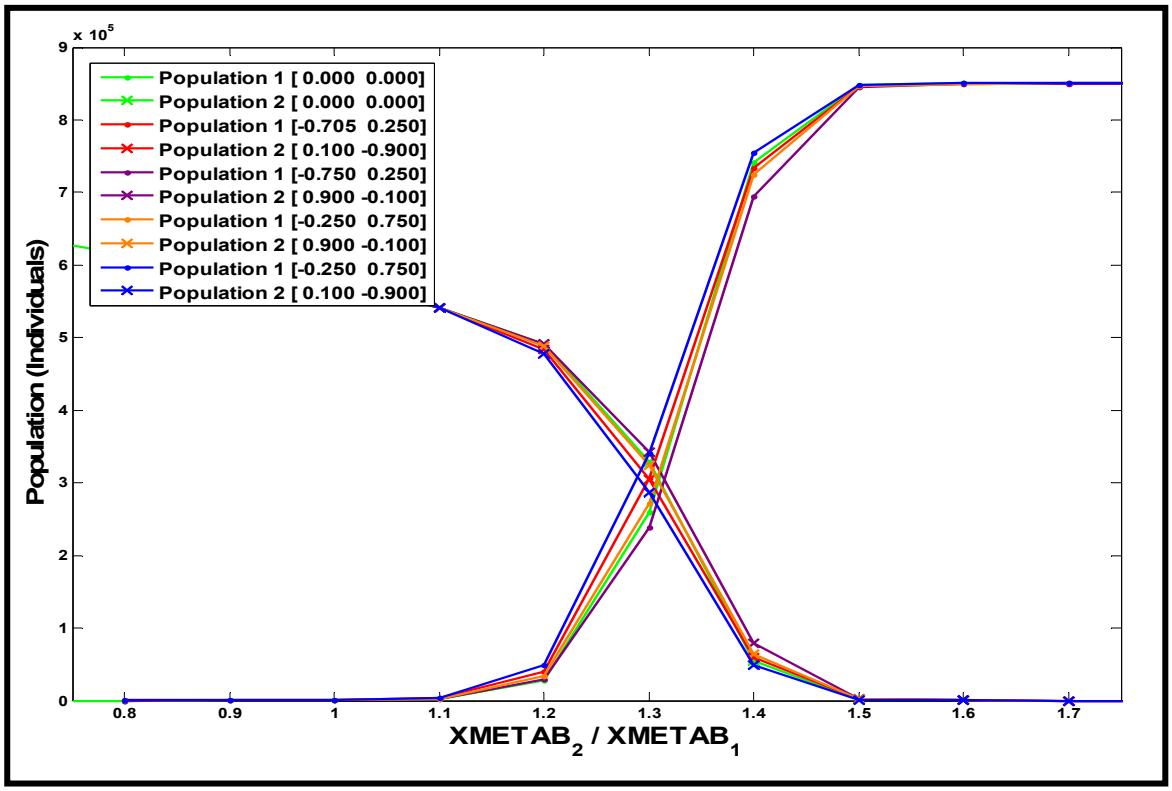


Figure 3.13 Average Population of Final Year vs. Ratio of Specific Base Metabolic Rates for different interaction matrices

same for some of the interaction matrices (Figure 3.9).

In the next phase of the system build-up, the two producer species were no longer completely identical. To start, the metabolic rate of one species was set to a constant value while the other was varied. It was expected, when run in simulation, that either one species or the other would dominate the system. Indeed, the results showed only a narrow range of base metabolic rates for the second species where both species survived, centered around the point where the ratio between base metabolic rates of the two species is 1:1 (Figure 3.10). Again, another set of simulation were then run with various health interaction matrices. From the graphs, it is possible to see that although the effect of the health interaction is fairly slight, it is certainly present (Figure 3.11). The two species were then made to be significantly different and simulated again both without and with the variety of health interaction matrices and metabolic rates for the second species. The results were very similar to the previous set, but the range where both species survived no longer centered around the 1:1 ratio of base metabolic rates (Figure 3.12, Figure 3.13).

As a final exploration of the two-producer system, the second species' base metabolic rate was fixed and a single interaction matrix chosen. Then a set of simulations was run with different combinations of initial populations for the two species. The results were plotted with one species' population versus the population of the other. This type of graph shows if and how a system cycles over time. When the cycles for all the simulations in this set were plotted together, it was possible to see whether the different versions of the system reach the same final point and the differences in the path they take to it. Although it was possible that most of the simulations would reach the same final point but that some

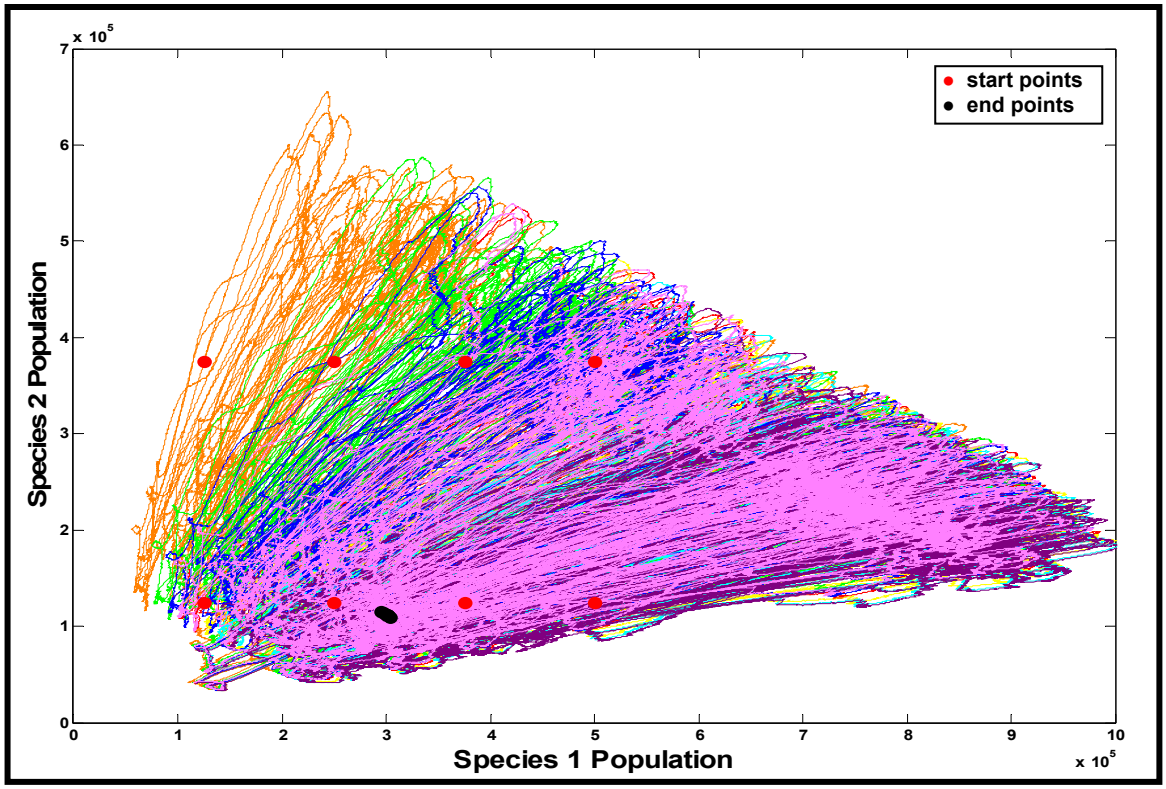


Figure 3.14 Producer Species 2 Population vs. Producer Species 1 Population

others might go toward different points, all of the systems simulated ended at approximately the same point, meaning that all the initial states were in a range of the state space in which the ecosystem could be considered stable (Figure 3.14).

The last step in building-up a system for verification was to add a consumer species. The first set of simulations for this version of the system had no health interaction between the species. Several matrices for the consumer species' food preferences were created, and then simulations were performed with a variety of values for the consumer's hunting ability (AFFECT1) for each food matrix. The final populations of each simulation were plotted against the corresponding AFFECT1 value. For some food matrices, there were no simulations where all three species survived (Figure 3.15). Others, however, had a narrow range of AFFECT1 values in which all three species could survive. To one side

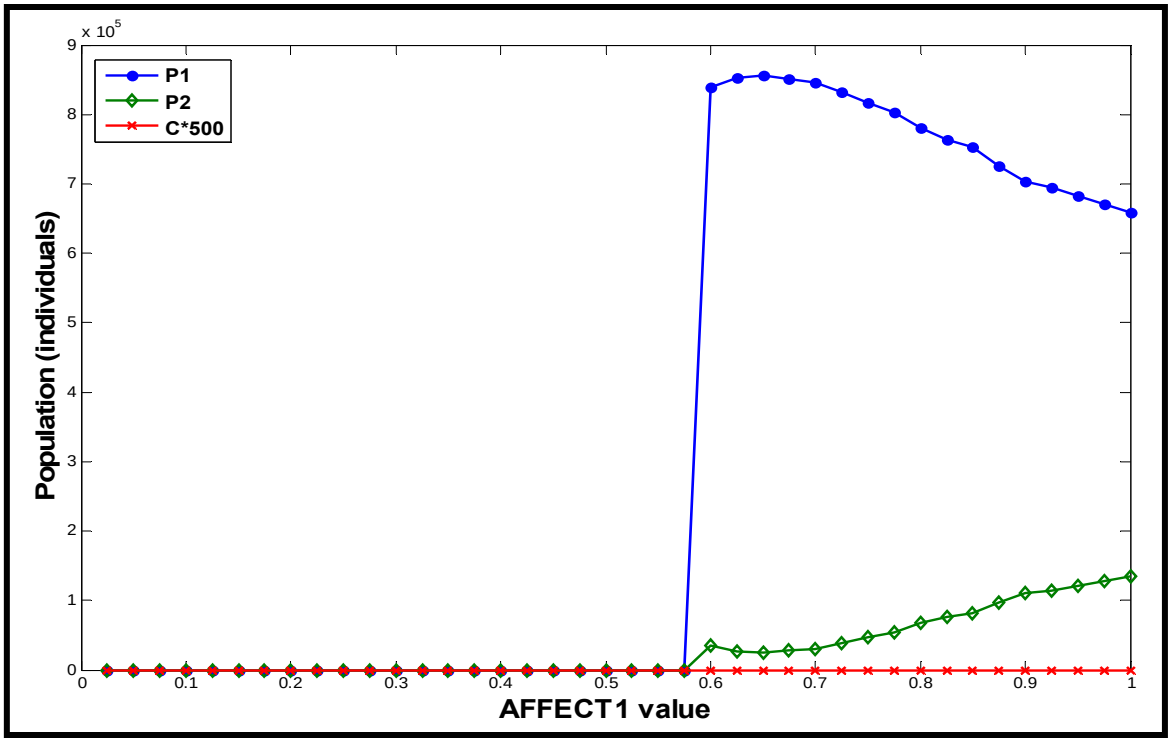


Figure 3.15 Average Population (of Producer 1, Producer 2, & Consumer) of Final Year vs. Consumer Hunting Ability (AFFECT1) when the consumer has a 75% preference for Producer 2

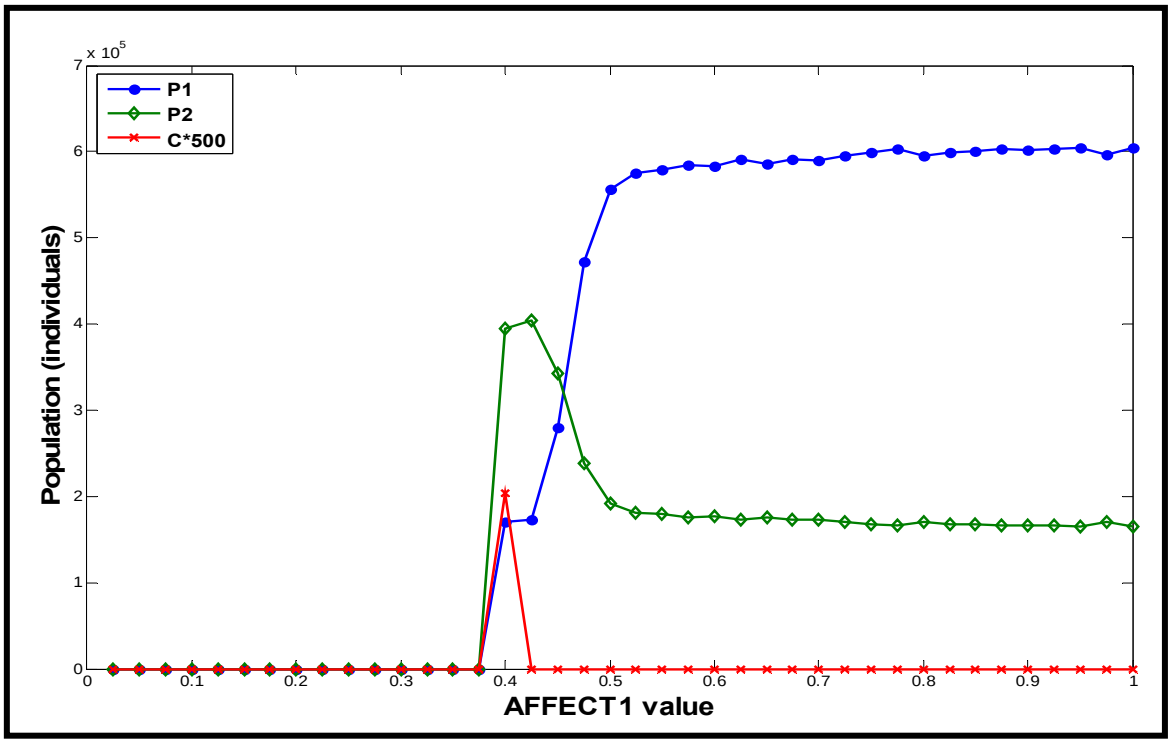


Figure 3.16 Average Population (of Producer 1, Producer 2, & Consumer) of Final Year vs. Consumer Hunting Ability (AFFECT1) when the consumer has a 90% preference for Producer 1

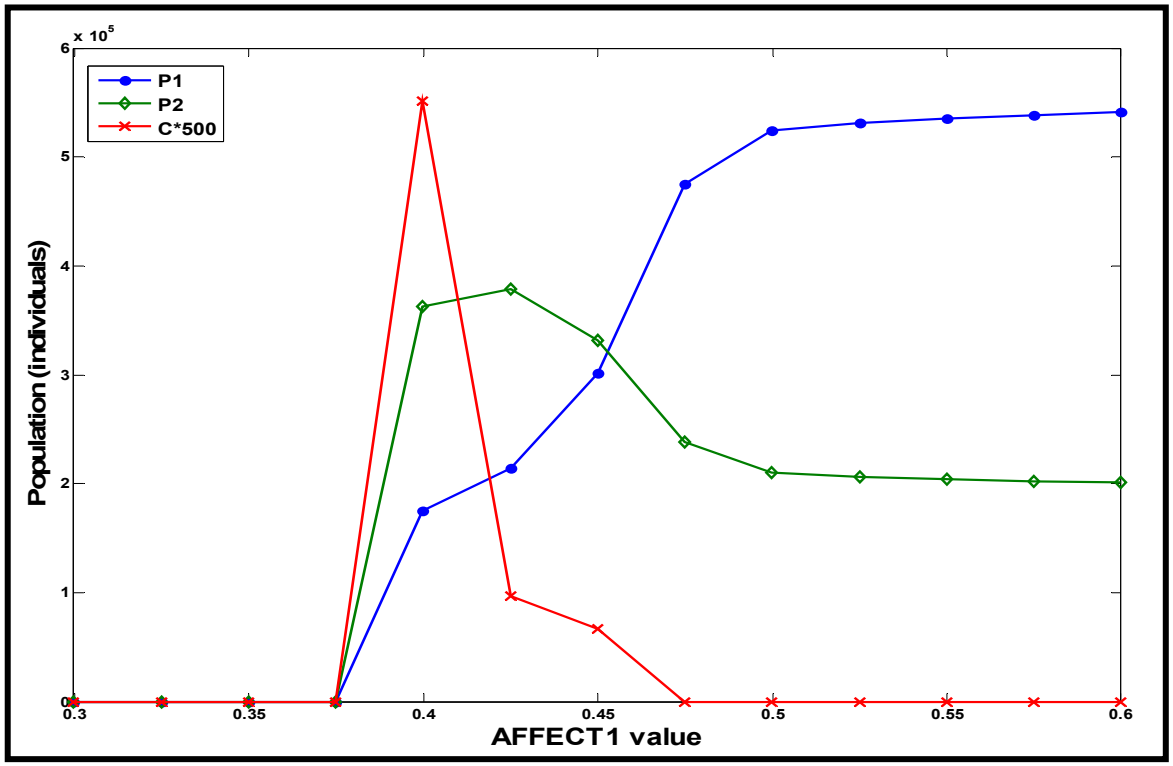


Figure 3.17 Average Population (of Producer 1, Producer 2, & Consumer) of Final Year vs. Consumer Hunting Ability (AFFECT1) with a given interaction matrix

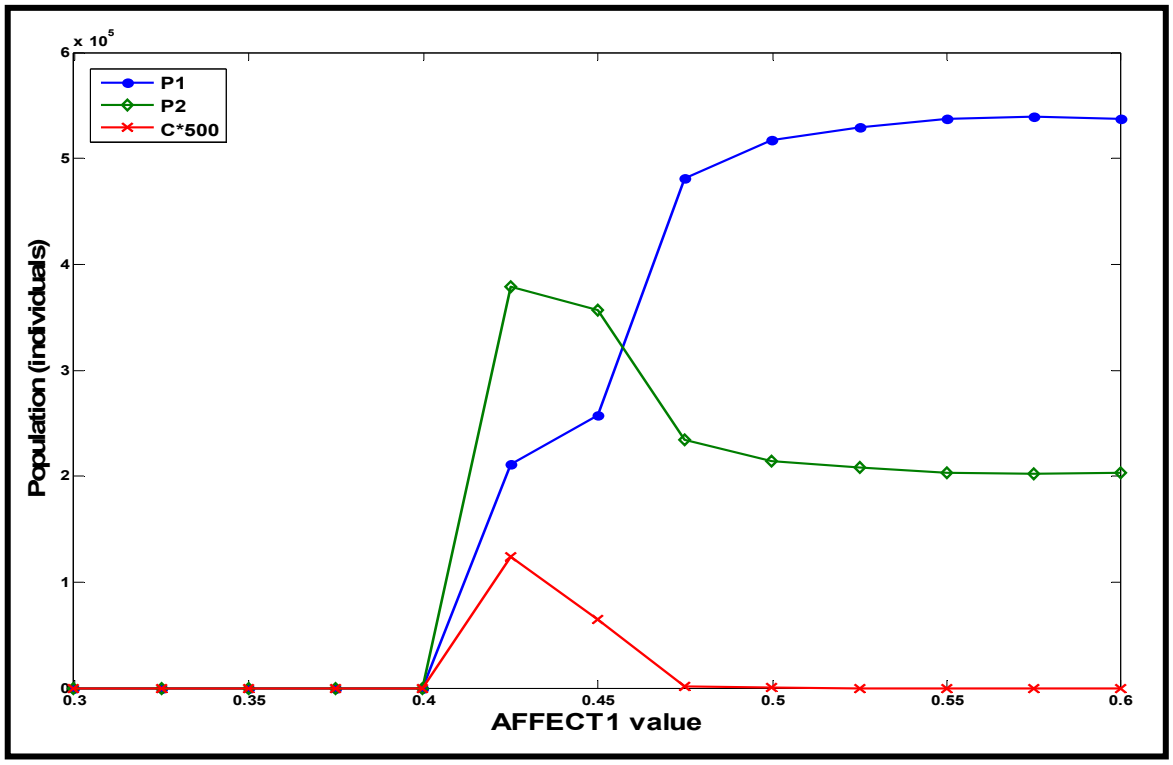


Figure 3.18 Average Population (of Producer 1, Producer 2, and Consumer) of Final Year vs. Consumer Hunting Ability with a different interaction matrix than Figure 3.17

of that range, the consumer was a poor hunter and starved to death; and to the other side, it was too efficient, ate its entire food source, and then starved to death (Figure 3.16).

Following the same method as above, the next phase was to add health interaction to the system. A single matrix was chosen for the consumer's food preferences, and a range of values for its hunting ability was used for each health interaction matrix. The output was plotted as before for each health matrix, and the effect of the various matrices is quite clear when the graphs are compared (Figure 3.17, Figure 3.18). The final phase for the three-species system was to test the field of initial conditions, as described above with only producers. The results were also similar, with all systems moving towards one area on the graph, as seen in Figure 3.14.

3.5 Validation and virtual ecosystems

When working with a traditional ecosystem model of an extant system, validation is often carried out after verification is complete. In that process, the output from the model is compared to the comportment of the exemplar system to see if they are statistically similar. If they are, then the model is considered to be a good representation of the exemplar system. If not, then the model may be adjusted and readjusted until its output and the system's comportment are similar. It should be noted that when a model is created to emulate a system that does not exist yet (e.g. a model to explore a proposed ecosystem restoration plan), then validation can only be performed if and when the system has been created. A virtual ecosystem, however, is an entity in its own right, and there is not and never will be an exemplar system with which to compare it. Thus, in such a case, the output of a simulation is the comportment of a system in reaction to

certain forcing functions, rather than an emulation or projection of another system that is being modeled. Hence, validation is not applicable for virtual ecosystems and can only be approximated with the verification process.

3.6 Comparing the virtual ecosystem to ecological theory

The exploration of the computational model and simulation platform through graphs and the slow build-up of a system imparted confidence that the program worked as intended. It was decided, however, that a comparison of the output from the virtual ecosystem to standard ecological theory regarding the comportment of ecosystems would be another way to approximate validation and build confidence in the idea that this virtual ecosystem does represent ecosystems in general. As the virtual ecosystem tracks only energy

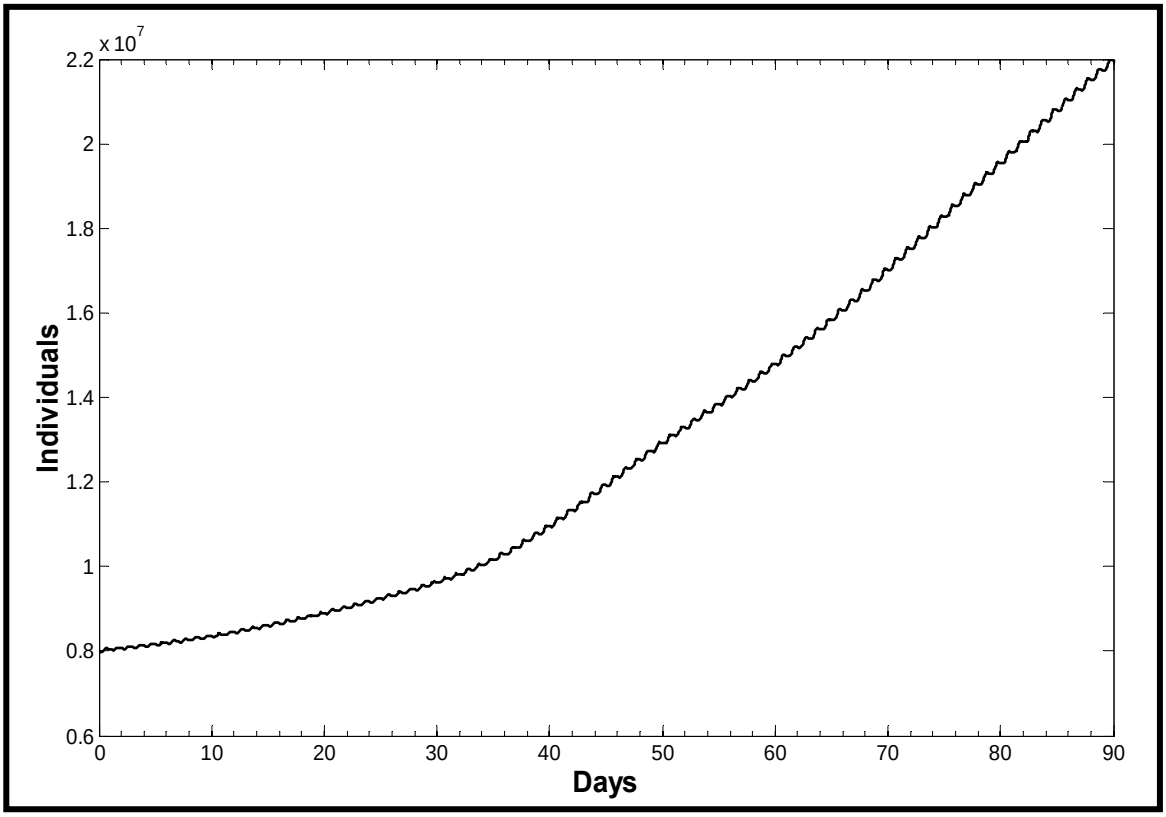


Figure 3.19 Exponential Growth: Individuals vs. Time

exchanges and populations, theory from population ecology is the most relevant to this situation.

The most basic tenet of population biology is that a population will exhibit exponential growth in an environment with unlimited resources (Gotelli 2001). While there is no way to simulate unlimited resources in the virtual ecosystem (the simulation platform requires system boundaries), a population started far below the system's maximum energy limit does grow in a roughly exponential fashion (Figure 3.19). The population growth is not exactly exponential because of the interactive nature of radiation in the virtual ecosystem. Of course, very few systems with unlimited resources exist. Most systems have a *carrying capacity*, an approximate maximum population of a given species that a system can sustain due to competition for resources such as food and space. In the virtual ecosystems created here, the maximum energy allowed for all producer species combined is set at a specific value. However, the carrying capacity for each producer species – and thus the consumer species – within a system is not predetermined and therefore unknown until derived by simulation.

The graph of a population under resource constraint will generally be in the style of one of two possible curves. If the initial population of the species is below the carrying capacity, the population curve will start with exponential growth, see Figure 3.20a from Gotelli (2001). Then the growth rate will slow down until the curve levels out at approximately the carrying capacity, creating an 'S' shaped logistic growth curve (Figure 3.20a). When the initial population is greater than the carrying capacity, the population will decline, rapidly at first then slower until it too levels out at the carrying capacity

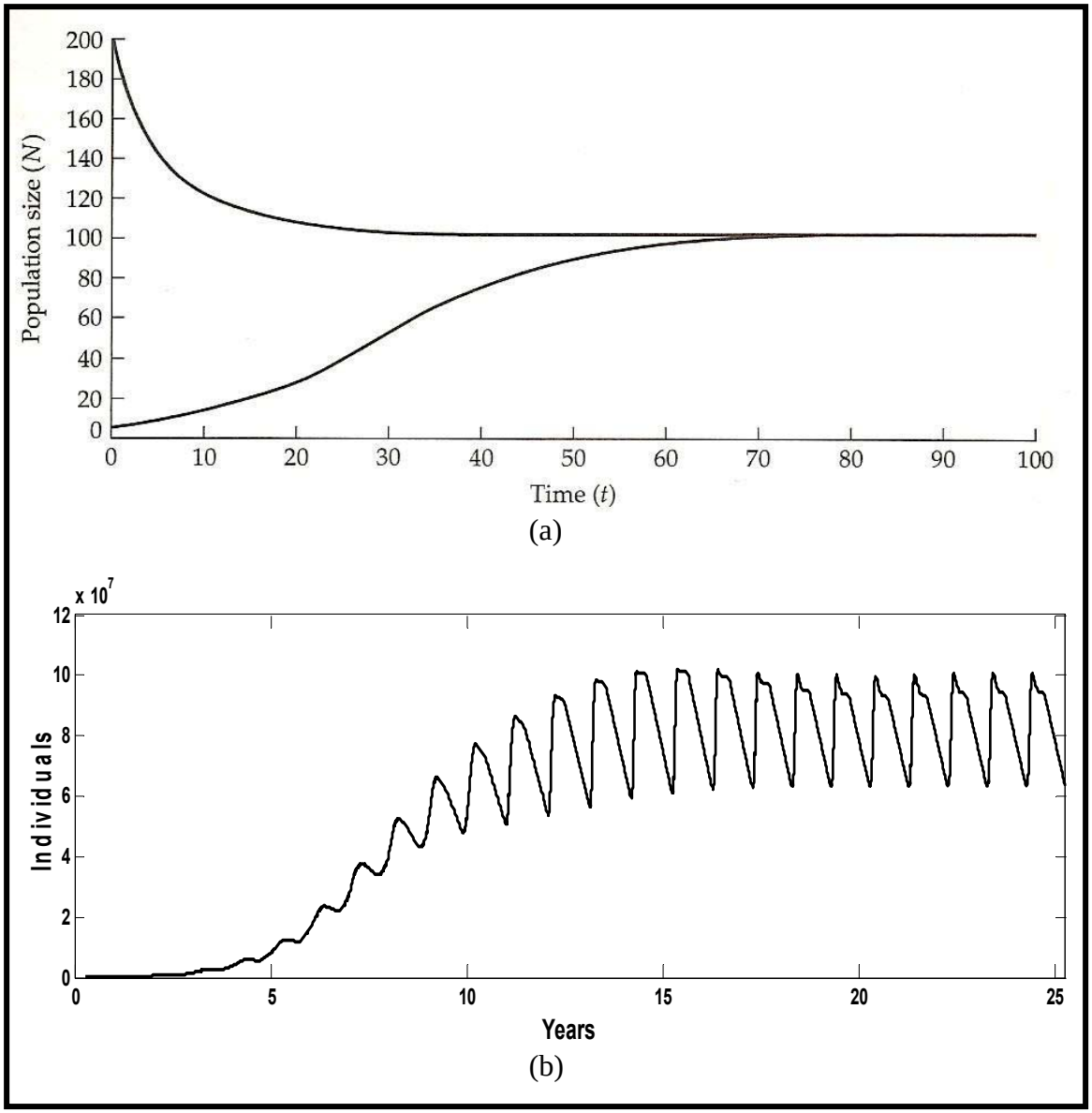


Figure 3.20 Logistic Growth Curves, Population vs. Time: (a) Example curves (Gotelli 2001), (b) Output curve from virtual ecosystem.

(Figure 3.20a). These graph standards were developed with continuous growth population models and thus produce a smoother curve than is seen in an object-based model with non-constant forcing function (i.e. the roughness is due to annual weather patterns). However, one of the two curve types can still be seen in the overall trend of the population curve (Figure 3.20b).

The comportment of the virtual ecosystems was also compared to the dynamics of previously established predator-prey models. In the Lotka-Volterra predation model, the dynamics of the predator and prey populations follow a cycle depending on the density of the two populations, see Figure 3.21 from Gotelli (2001). At one point of the cycle (Figure 3.21, upper right quadrant), both populations are abundant. There is enough food for the predator population to increase. However, the predator population becomes too large and causes the prey population to decrease. As the prey population falls, the predator population no longer has the food abundance required to increase and begins to decrease as well. With much of the predatory pressure lifted, the prey population begins to increase. They become abundant again, allowing the predator population to start to increase again until the system reaches the beginning of the cycle again. In the virtual ecosystem, however, oscillating populations only occur in the presence of forcing functions with oscillations. Thus, the virtual ecosystem does not conform to the Lotka-Volterra density dependent predation model. There is evidence, however, that the Lotka-Volterra model is not really applicable in physical systems either.

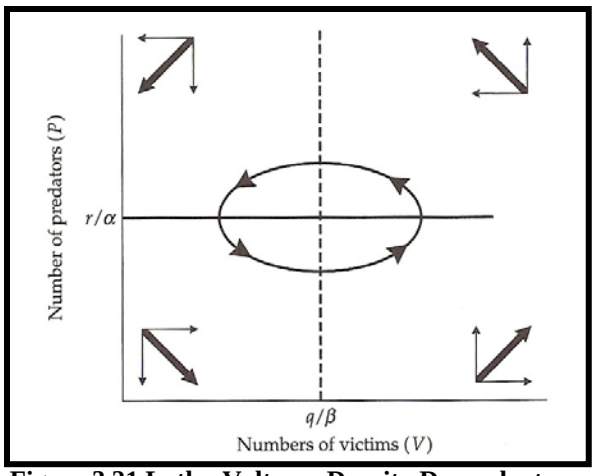


Figure 3.21 Lotka-Volterra Density Dependent Predation Cycle (Gotelli 2001)

A classic example of predator-prey dynamics, the Canada lynx and the snowshoe hare, was once thought to be cyclical purely because of density dependent interactions between the two species (Trostel et al 1987). However, newer studies have determined that, although the lynx is almost entirely dependent on the hare for food, the hare population is

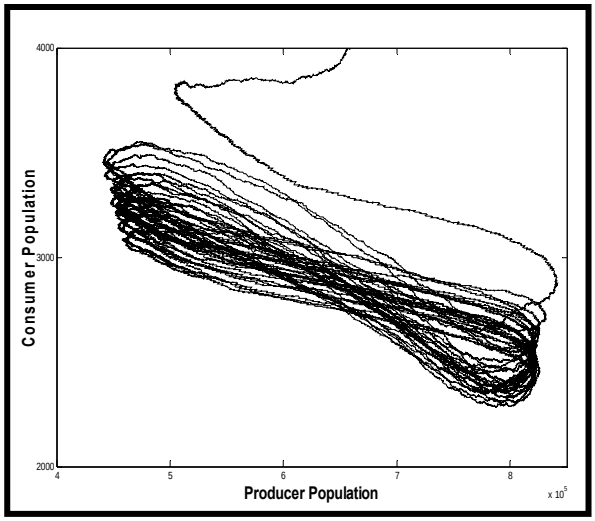


Figure 3.22 Cyclical compartment of virtual ecosystem: Consumer Population vs. Producer Population

not solely dependent upon the predation of lynx but rather a complex set of pressures including the abundance of its own food species (Stenseth et al 1997). As snowshoe hares are herbivores, those food species are primary producers, the populations of which are in part dependent on forcing functions like radiation and temperature, making the cyclical dynamics of the lynx and hare also at least in part dependent upon forcing functions. Similarly, when a consumer population from a virtual ecosystem is plotted versus the population of its food species, in the case of having a single food species, the resulting graph displays the same type of cyclical dynamic when in the presence of cyclical forcing functions (Figure 3.22).

3.7 Summary

The virtual ecosystem for this project is materially closed and energetically open with no spatial representation, much like an ecosystem in a well mixed jar. A number of objects interact in the system, each object being an individual of a species that is either a producer or a consumer. Species have a number of attributes, the values of which individuals of those species inherit. Individuals also have a number of attributes whose values are defined at the individual level. The individuals interact according to a set of rule based expressions that form a simple process of intra- and inter-specific interactions. Two forcing functions, temperature and radiant energy intensity, provide the main

impetus for the comportment of the system.

The object-based modeling method used here is very useful for illustrating the complex nature of ecosystems. From this set of fairly simple rules and equations, the entire comportment of the system arises. Birth and death rates result from the interactions within a system, not imposed upon it. Intricate behaviors and cycles emerge without guidance, a hallmark of complex systems. Object-based models are also well suited to situations where the overall system is not fully understood, making it possible to obtain system level results from better known local interactions. The computational model for the virtual ecosystem was written in FORTRAN 90/95. Input and output is handled via various text files. Initialization and iteration routines are contained in the main program, while forcing functions are separate subroutines.

A series of tests were performed in order to verify that the computation model works properly and results are consistent with ecosystems in general. Much of the verification was done with graphical analysis. Although the standard practice of validation does not usually apply to virtual ecosystems, the output from the virtual ecosystems was compared to theory from population ecology as an approximation of the process. Altogether, the virtual ecosystem program was found to be working as desired, comportment of systems simulated with it compared favorably to theoretical expectations of ecosystem comportment, and thus, the virtual ecosystem program and simulation program is an adequate representation of ecosystems for the purposes of this project.

4. Data Generation

The desired outcome of this project was to test case-based reasoning as a method for compiling and analyzing large data sets of diverse ecological data. Thus, a large number of simulations had to be performed to generate data on differing system constitutions and the resulting comportments. The constitutions of the simulated systems needed to be widely varied within the state space in order to represent ecosystems as generally as possible, as well as allow the exploration of the boundaries and parts of the state space that are not normally investigated with traditional models. Therefore, a method to randomly create a variety of systems was formulated. Then the systems thus created were run in simulation. The data from those simulations formed the core of the case base to be used to test case-based reasoning. Because of the nature of virtual ecosystems and the fact that this project is at such a preliminary stage of the research process, it was not required to make the individuals in or the composition of the systems "match" known species or systems.

4.1 Creating simulations

As seen previously, several different system constitutions were used with the virtual ecosystem simulation program during the course of verifying the program and testing the various quantitative measures. Most of these systems were created 'by hand' or modified in small ways with a computer program, e.g. incrementally increasing the value of one species attribute. However, creating each system individually for the large number of case base simulations would have taken far too long. Also the maximum degree of randomness was desired so that the systems simulated would represent a fair portion of the state space. Thus, an automated method was needed to create ecosystem constitutions

to be used with the virtual ecosystem program.

In order to have all simulations be meaningful, the system constitutions could not be entirely random. It would have been a waste of time and computing resources to simulate systems which were highly likely to fail completely, e.g. a system with only consumer species. A set of guidelines was needed to create reasonable systems. This requirement applied to both the creation of the species in each system as well as the structure between them. It was further decided that, although new species could be created for each constitution, it would facilitate comparison between systems and the analysis of the importance of ecosystem structure if there was a set pool of species to draw from when defining the structure of each system. Not all species would be present in every system constitution, but all systems would be inside the state space defined by the pool of species. This makes the virtual ecosystem project more like a physical ecological engineering project where the engineer has certain plants, animals, etc. to choose from.

4.1.1 Creating species

Twenty each of producer and consumer species were defined. Species creation was performed in Excel spreadsheets (Appendix D), one spreadsheet for the non-interactive attributes (minimum energy, birth energy, base metabolic rate, etc.) and one each for the two interactive matrices (consumer food preferences and health interaction). The formulas for each species level attribute were entered in the matrix of species number (numbers 1-20 for producers and numbers 21-40 for consumers) versus species attribute (minimum energy, energy at birth, etc.) in the main sheet and in species versus species matrices for the interaction matrices. Once all the formulas to calculate the attribute

values for the species had been entered, a copy of the Excel file was saved without formulas and only the calculated values. This file served as the source file of species attributes to be used in the creation of ecosystems for simulation.

As stated above, certain guidelines were required to make "reasonable" species. For example, it would be fruitless to create a species whose birth energy was smaller than the minimum energy required for an individual of that species to remain alive. Thus, the value for the minimum energy for a species would need to be set first and the value for its birth energy calculated based on the minimum energy value. It was not desired, however, to have all species have birth energies that were the exact same multiple of their minimum energies. In physical systems, some species have a greater chance of dying shortly after birth than others. It was desirable to mimic that phenomenon in the virtual ecosystems for this project; therefore, the birth energy values were decided to be in a range between 1.5 and 3 times the minimum energy values, which were randomly chosen in an even distribution between 10 and 150 energy units – values that were deemed reasonable after the simulations performed during testing and verification. The same reasoning was applied to the reproduction threshold of each species, and the values for that attribute were determined best between 5 and 10 times the value of the birth energy for each species.

In a slightly different manner, the absolute maximum age for each species was calculated using the already assigned value for the lower end of the maximum age range of the species. It was discovered during the verification process for the virtual ecosystem program that having all individuals of a species have the same maximum possible age

resulted in undesirable, sudden large population decreases. Though part of this problem was fixed by setting the values for the seeds used with the random number generators in a specific manner (see chapter 3), it was determined that the individuals within a species should have slightly different maximum ages. Thus, the concept of initializing individuals with different maximum possible ages was added to the original aging concept in the system. It was also necessary, therefore, that each species be created with a large enough range from which maximum ages for each individual would be calculated. Although the values for each of the two attributes – absolute maximum age and minimum maximum age – were randomly generated in an even distribution within two different ranges, the values for absolute maximum age were set to be at least 30 days larger than the values for the lower end of the maximum age range. The value for the minimum maximum age for a species was chosen to be between 60 and 1825 days, and the value for absolute maximum was assigned in the range between 120 days or 30 days over the minimum value, whichever was larger, and 3650 days.

Other species attribute values were determined randomly within a range but not based upon any other species' attribute value. The values for the base metabolic rate of producer species were generated in a range that was between 5.0e-8 and 2.5e-7 EU / EU·s. For consumers, the range for the same attribute was between 3.5e-7 and 6.5e-7 EU / EU·s. The food affectedness (hunting ability) values for consumer species were set between 0.250 and 0.750, where the hunting ability of a species was inversely related to the value of its food affectedness (i.e. a smaller value equals a more efficient hunter). The values for health affectedness (how sensitive a species was to the abundance of the various species in a system) were set between 5 and 25 for all species. The health interaction

values for all species were determined between -1 and 1. Each species was assigned a health interaction value for every other species created, including itself. Because of the way the evenly distributed random number generator in Excel functions, no zero values were obtained for health interactions.

The two remaining attributes at the species level were not determined in the same manner as the attributes above. The size of the energy quanta received by individuals of producer species during the absorption of energy was set at the species level and could technically vary a great deal between species. However, the effect of this particular attribute had not been explored during program testing and verification, and it was decided to use the same value for all species. During testing and verification, the only value for energy quanta size used was 0.5 energy quanta, so that value was assigned for all producer species.

As the species being created were to be used in a number of different ecosystems of differing compositions, the food preferences of the consumer species could not be given specific values. Though food preferences – what food sources a species was willing to consume – were a species level attribute when considered for a single species, the actual values for those preferences were set at the system level. Those values form part of the system structure, which was very dependent on the system composition. In a system, the total of the values of a single consumer species' preferences must equal 1. It was unlikely that a given system created for simulation would include all of the species that a consumer species was willing to eat. Therefore, if the values for food preferences were set before the system composition was created, the sum of a species' food preferences in a system were also unlikely to be 1.

Accordingly, the food preferences were set during species creation without the values that would be used in simulation. Only preferred food sources were determined for each consumer. Evenly distributed random numbers between 0 and 1 were generated for each possible consumer relationship. If the value of the random number was above 0.5, the relationship would receive a 1, meaning that that consumer species was willing to eat that food source. Random number values below 0.5 received a 0 score, indicating that the consumer species was not willing to eat that food source. Of the twenty consumer species, six were created to be herbivorous, six more were designated carnivorous, and the remaining eight were generated to be omnivorous. As stipulated previously, none were allowed to be cannibalistic.

4.1.2 Creating systems

The ecosystem constitutions used in the simulations for this project were created directly as input files for the simulation program using another FORTRAN program, the "system creator program" (Appendix E). The tables of species attribute values and interactions created in Excel were converted into text files which could be read by the system creator program. The system creator program then randomly chose which species would be in each system, generated the structure of each system, and created the input files containing the system constitution and control parameters for the simulation. As with the creation of species, the system constitutions were created following a set of guidelines that would make it more likely that these systems would at least partially survive.

After reading in the attribute values for the species, as well as the consumer food preferences and the health interaction values for all species, the first task of the system

creator program was to determine how many species were in the system. As the maximum number of species allowed in the simulation program was set at 30, that value was also used for the upper limit of the number of species. The lower limit was set at two so that each system would initially have at least one each of producer and consumer species. A uniformly distributed random number generator was used to determine where the total number of species falls in that range.

The value for the total number of species was then used to calculate the values for number of producers and number of consumers. As mentioned previously, it was required that each virtual ecosystem be initialized with at least one species of each type. Thus, if the total number of species for a particular system was only two, then there was automatically one species of each type. Otherwise, the number of producer species was randomly calculated from the total number of species, with the remainder being the number of consumer species.

Two guidelines further affect the calculation of the number of producer species. As there were only twenty species of each type in the pool of species, the upper boundary on the number of producer species was twenty in a system where the total number of species was greater than twenty. In systems where the total number of species was twenty or less, the upper boundary on the number of producer species was one less than the total number of species. In both cases, the lower boundary on the number of producer species in the system was half the total number. This was decided in order to increase the likelihood of the consumer species having adequate food resources in the system.

Which species were to be present in a given system was determined randomly. Thus, a random number generator was used to calculate values between one and twenty to choose producer species, and values between twenty-one and forty were calculated for consumer species. The values calculated correspond to species numbers from the pool of species. Subroutines were included to make sure that there were no repeated species in a system. While the result of repeating a species would merely be that that species would have a larger initial presence in the system, it was desired that all the species of a given type (i.e. producer, the types of consumers – herbivore, omnivore, carnivore) have the same initial presence in the system (see below for initial population values by type). That way, it would be possible to see the dominance of different species, within each type, emerge from equal initial presences in the system. Once the particular species were chosen for a system, the vectors of species attributes values (e.g. minimum energy, birth energy, etc.) were built using the values for each species as recorded in the pool of species.

The next step in creating a system was to build the matrix of food preference values for the consumer species. For each consumer in the system, the program searched through the species present in the system to see if any of them were food sources for the given consumer. Then preference values were determined for each of the consumers' food sources in the system. The total of the preference values for each consumer species needed to equal 1. A baseline value of 0.005 was given to each food species, and the remainder of the total preference value was then randomly distributed among the food species for the consumer. The process was repeated for each consumer, after which the health interaction matrix for a system was built using the values already determined during the creation of the pool of species.

At that point, the system was fully created. However, in order to set up the system for simulation, initial population sizes for each species and the seeds for the random number generators used during the simulation to set the initial state of individuals as they are created were also needed. Initial populations were set according to type of species. Producers all began with a population of 10,000. Consumer species initial population values were set by what type of food preferences they had: herbivores at 1000, omnivores at 100, and carnivores at 10. This was representative of the observed phenomenon of decreasing energy being available for increasing levels in a trophic pyramid, commonly known as the 10% rule of thumb. Values for the random number seeds used for the system during a simulation were obtained between -1 and -1,000,000 (the random number generator subroutines used in the main program require negative integer values as seeds) in the system creator program from a uniformly distributed random number generator.

The system creator program also wrote the simulation parameters file. Most of the values for simulation parameters were constant for the simulations being done for the project. Only another set of random number seeds needed to be calculated, and the file name for simulation output needed to be determined. Once those two tasks were completed, the simulation creator program wrote the input files: one containing all the species attribute and interaction values as well as the initial state of the system, the other the simulation parameter values. It is important to note that the input files of forcing function parameters for each simulation were not created by the system creator program. In order to highlight the relationship between ecosystem constitution and comportment, it was decided to use the same forcing function parameters for all the simulations.

4.2 Summary

To build the case base of knowledge desired for this project, a large number of reasonable systems were needed to be run in simulation with the virtual ecosystem program. Random generation was chosen because it was desired that the constitutions of the systems cover as much of the state space as possible and more “realistic” representation was not required at this stage in the research process. A pool of species was created using an Excel spreadsheet and a set of guidelines to make species that had a chance of surviving. A system creator program was used to generate ecosystems with reasonable compositions.

5. Analyzing the Constitution and Comportment of Virtual Ecosystems

Ecosystems are complex systems, and it is very difficult to describe complex systems without losing information. Even with relatively simple systems, like those being used in this project, there is too much information to meaningfully encode in a knowledge base and utilize. Thus, each ecosystem must be analyzed with quantitative measures so as to produce a set of values that represent the constitution of the system. In this way, the systems can be compared to each other much more easily. The same is true regarding the simulation output (comportment) for each ecosystem. Numerous variables are tracked throughout simulation, and it is very difficult to directly compare the large vectors and matrices of results. Hence, more quantitative measures are necessary to generate a value or values that characterize the overall comportment of the virtual ecosystems.

There are many aspects of an ecosystem that can be directly measured or analyzed, the number and distribution of species being commonly examined system features. Long term studies concerned with succession may concentrate on the changes in the system composition (Gent and Morgan 2007, Ward and Jennings 1990). DNA mapping of various species is also becoming common, and as it is supposed that species genetics can affect an entire community, so the genetic compositions of systems are also analyzed (Whitham et al. 2006). Population levels, birth and death rates, and immigration and emigration rates of one or more species in a given system are also frequently observed. Vegetation density, mass, canopy cover, photosynthesis rates, and other plant based indicators are also used to indicate ecosystem organization or health (Miller III and Dunton 2007, Aoki and Mizushima 2001).

In many studies, the movement of various materials such as nitrogen or water through the system is tracked and analyzed to quantify the transport efficiency of the system (Holdo et al. 2007,Oguntunde et al. 2007). The level of pollutants in the soil or water of ecosystems is also studied (Friedli et al. 2007). More general system aspects that are often analyzed are the stability of an ecosystem's comportment through time and the complexity of a system's constitution (Chen and Cohen 2001,Rozdilsky and Stone 2001, Pimm 1984, Parrott and Kok 2000).

Some of these methods are only applicable to physical systems or are otherwise unsuited for the virtual ecosystems used in this project. For example, as the virtual ecosystem for this project has no spatial or mass reckoning, measures based on vegetative cover or mass would be impossible to use. The methods for studying the transport efficiency could be adapted by tracking energy transfer in the system, and then analyzed in much the same way as is generally done with materials. This project is intended to represent and examine the connection between the constitution and comportment of ecosystems, however, so ecosystem functions like energy transport are not desired aspects to be examined. The long term goal of this project is concerned with developing theory for engineering persistent systems, so more general aspects are needed. Stability and complexity seem to be the most relevant; stability because of its relationship to system survival and complexity because it can be used to examine and quantify ecosystems holistically. Thus these two ecosystem characteristics have been chosen as appropriate for the analysis of the systems in the project.

5.1 Measuring complexity

A complex system is one that is made up of many components which interact to give rise to structural and dynamical patterns that are not easily inferred from the system description (Parrott and Kok 2000). The degree to which systems are complex varies both between and within types of systems. The degree of complexity may be indicative of other features, such as the stability or other behavior of the system. For example, a social system (also known as a social network or social ecosystem) with a very complex set of relationships between different units in the system will have different dynamics than one in which all units in the social system are equally connected to each other. In ecosystems, complexity can be applied to both system constitution and comportment.

5.1.1 Constitutional complexity

An agro-ecosystem with only a few species would be considered to have a fairly low degree of constitutional complexity, while a rainforest containing thousands of intricately interacting species would be considered to have a high degree of constitutional complexity. The number of species is not the only factor that affects the constitutional complexity of an ecosystem; the types of species and the structure of the relationships between them also have an effect. For example, a system with five producers would probably be less complex than one with three producers and two consumers. Interspecific relationships like predation, competition, and the health interactions (mutualism, parasitism, etc.) all increase the complexity of a system. Also, the organization of the structure affects the constitutional complexity. For example, a system with clearly defined trophic levels (Figure 5.1a) is likely to be less complex than a system where some species in the food web occupy more than one trophic level (Figure 5.1b).

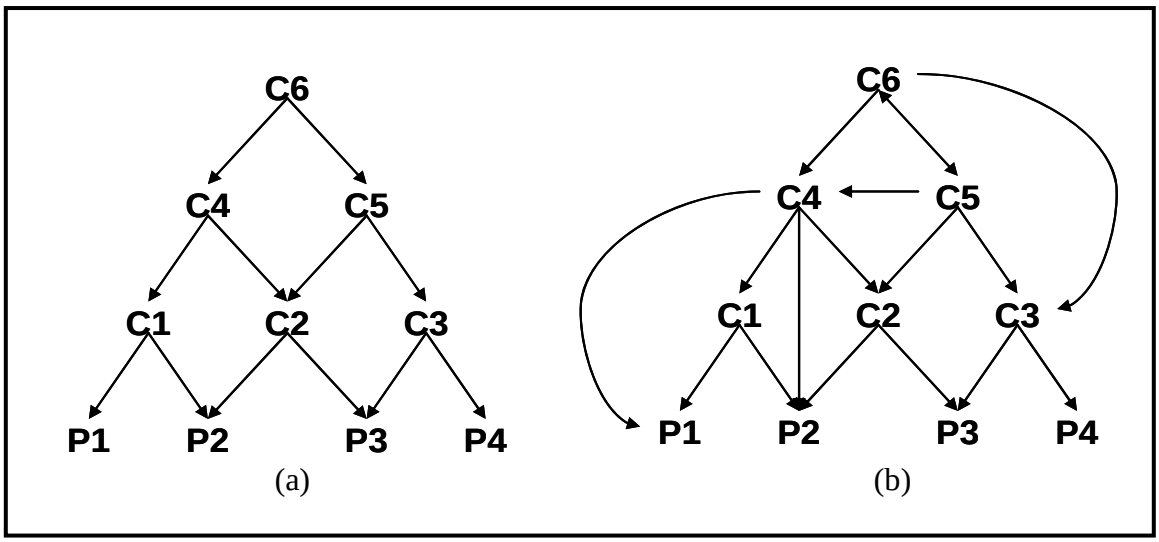


Figure 5.1 Food Webs of systems with (a) clearly defined trophic levels and (b) species occupying more than one trophic level.

To simplify the measurement of the constitutional complexity of the ecosystems, this aspect has been further split into compositional and structural complexities, where compositional complexity is related to the numbers and types of species, and structural complexity refers to the interactions between species. Although not used in this case, composite measures do exist. For example, the ***U* index** is a measure of ecosystem maturity based on material flows within a system (Perez-Espana and Arreguin-Sanchez 2001). In this index, the magnitudes of the flows are used to calculate a value of system complexity based on system components, trophic interactions, and detritus recycling.

5.1.1.1 Compositional complexity

There are two main possibilities for quantifying the compositional complexity of an ecosystem: *species richness* and *species diversity*. Species richness is simply a count of how many species exist in the system. Species diversity not only includes the number of species but also the types of species, their distribution in the system, and the species

evenness (how numerically equal the populations are) as well. One measure of species diversity, the **Shannon index**, is larger when the evenness of species is high or when there is a large number of unique species in the system. **Compositional pattern diversity**, a measure of the relative arrangement of subunits in a system, also includes the variation in species richness among communities and the evenness of the species (Scheiner 1992).

High species evenness does not, however, necessarily mean that a system is more complex. In fact, it is possible that high species evenness may imply a less complex system. Systems with many trophic levels are likely to have differing populations at the various levels (e.g. high level consumers like carnivores rarely have large populations while scavengers may). Thus, species evenness is better not considered when determining the compositional complexity of the ecosystems for this project. Furthermore, initial species richness is recorded directly in the input files (NTOT) and the presence of all species is listed in the output files throughout simulation; thus, it is a simple and reasonable measure to use. For this project, the total species richness was measured as well as a breakdown by producer and consumer species richness. Both initial and final values were recorded for comparison purposes.

5.1.1.2 Structural complexity

The structural complexity of ecosystems is an important consideration in ecological theory. There is considerable debate regarding whether complexity is related to the stability and persistence of systems, and many different ways of measuring structural complexity have been devised. A method that comes directly from the realm of

complexity theory is *q-analysis* (Casti 1994). In this technique, species interactions are represented by an incidence matrix of predator relations. Even species that do not have direct interactions can be related to each other through intermediate species, the number of intermediaries determining the strength of the interaction, the *q* dimension. The entire system is analyzed at a particular *q* dimension to find how well connected the system is at that level. However, this multi-dimensional view of structural complexity is not really desirable for the purposes of this project.

The field of ecological network analysis provides a measure, *ascendency*, which combines the vigor and organization of the system into a single variable (Ulanowicz 2000). Ascendency is based on flows of currency between compartments in the ecological network. Currency can be materials like water, nitrogen, and contaminants that move through the system. Anything else that can be exchanged between different parts of a system can also be used as currency in an ecological network. In this case, the currency would be energy and the compartments the various species in a given system. As ascendency is based on currency exchange, it requires that at least one time step of simulation be performed and is not an indication of the system structure independent of the conditions of simulation, but rather how the structure of the system reacts to such conditions. Thus, it is not appropriate for measuring structural complexity.

One common tool to measure the complexity of an ecosystem is *connectance*, the ratio of actual species interactions to the total number of possible species interactions (Chen and Cohen 2001, Rozdilsky and Stone 2001). Vasconcellos et al. (1997) combined connectance with how feeding interactions were distributed between the trophic levels of

the food web. Either of these methods is suitable to use with the virtual ecosystem. However, since trophic levels are not being specifically assigned to consumer species, the first version of the measure was chosen.

The only data required to calculate connectance is the number of consumer species, the total number of species, and the food preference matrix. These are all present in the simulation input files. As mentioned in chapter 3, there is also a set of health interactions between the species. These could be included in measuring connectance. However, the effects of health interactions are fairly weak. Also, due to the system generation technique used in the project (see Chapter 6), there are no zero values in the health interaction matrix. This would mean that, of the possible links between species contributed by the health interaction, all links would be used for all systems. The result would be a 1:1 ratio for all systems, making the links contributed by the health interaction meaningless for system comparison purposes. Thus, the health interaction has not been included in the calculation of connectance, and the number of possible species interactions is based on the food web only.

The number of possible interactions for each system is the number of consumers in the system multiplied by one less than the total number of species. Because cannibalism is not being allowed in the virtual ecosystem program, the link between each consumer and itself is not considered possible. Predator relationships are also considered to be directional, so there are two possible links between two consumer species. The number of actual interactions in the system is the number of non-zero values in the food

preference matrix. Connectance is then the second value divided by the first (Equation 5.1),

$$C = L / N_2*(NTOT - 1) \quad (5.1)$$

where C is connectance, L is the number of actual interactions (links), and N_2 and NTOT are respectively the number of consumer species and total number of species in the system. It is important to note that systems with more species may seem to require more intricate structures for the same connectance value. This is because adding even one species to a system may add many possible links. For example, adding one consumer to a system that previously had six species adds at least five more possible links. To investigate the measure, it was applied to two sets of systems, the systems within each set having identical species compositions but different structures (Figure 5.2). It was decided the results were satisfactory and the measure usable.

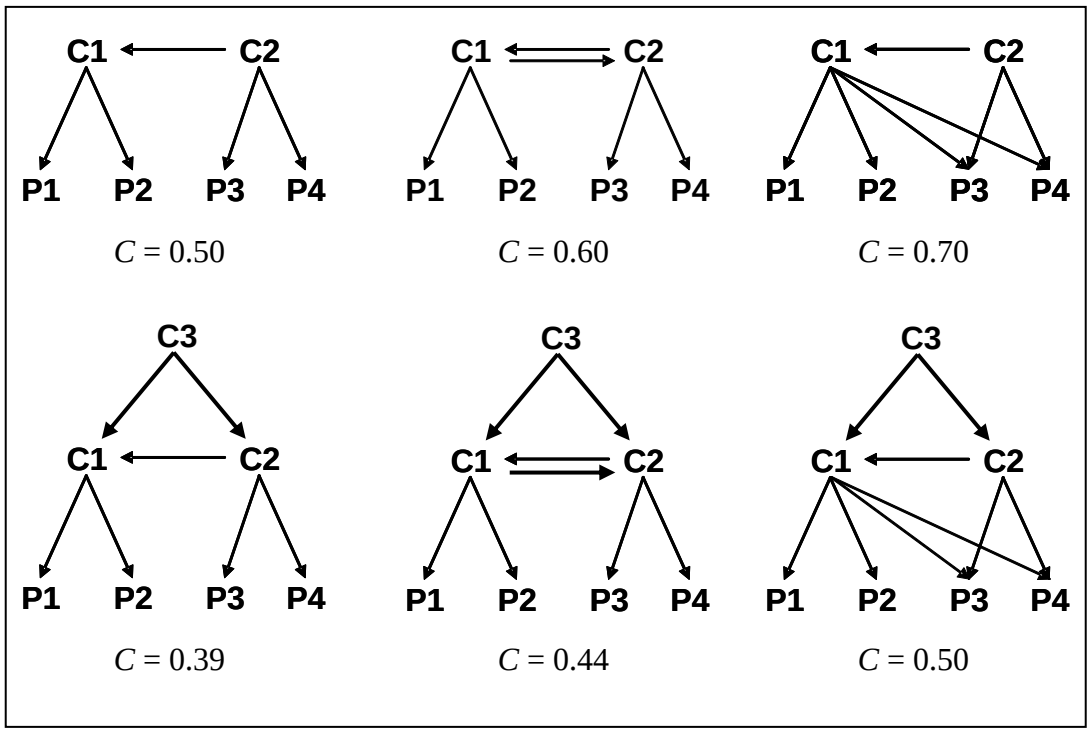


Figure 5.2 Connectance Values for ecosystems with different compositions and structures

5.1.2 Comportmental Complexity

The comportmental complexity of a system indicates how complex the behavior of the system is and can be anywhere in the range from “not very complex” to “quite complex”. In a system in which the populations remain relatively steady over time, the comportmental complexity is fairly low. A system with regular cycles would be somewhat more complex, and a system that is deterministic but has no regular pattern at all would be more complex yet. All of the factors that contribute to constitutional complexity, as well as the interactions that take place during the comportment (either by simulation or, in physical systems, the passage of time), affect the comportmental complexity of a system. The sampling frequency used when looking at the system comportment also has an impact. Populations sampled on a daily basis are likely to have more variation than populations sampled on a yearly basis, given that the yearly sample is taken at approximately the same time of year and in the same part of the life cycle of the species.

The comportmental complexity of systems is not often analyzed, but there are a few examples. Scheiner (1992) applied pattern diversity analysis to temporal subunit arrangement as well as compositional arrangement. The complexity of species abundances patterns have been measured by calculating the fractal dimension of the species abundances over time (Mancinelli et al. 2007). Fractal properties, such as fractal dimension, are also often used to analyze the behavior of individual animals and to classify habitats such as coast lines (Tanner et al. 2006, McDonald and St. Clair 2004, Nams and Bourgeois 2004). However, while temporal pattern diversity could be used with the virtual ecosystem simulations, the fractal analysis methods all deal with a single

species over time, not the comportment of an entire system.

Despite being otherwise suitable, temporal pattern diversity was not used because the high percentage of species that were expected to go extinct during simulations would affect pattern diversity too much. Instead, a measure was needed that could be used on the comportment of the system regardless of how many species had gone extinct. Because of the difficulty involved in trying to measure the complexity of multiple populations and variables at once, it was desirable to find a way to indicate the overall state of the system at a given time. Although it was decided not to be suitable to measure structural complexity, ascendency did provide a way to quantify the state of the system at a given time. Thus, the ascendency was calculated at every time step during simulation of each virtual ecosystem and recorded in the output files.

In order to use ascendency with the virtual ecosystem, the constitution of the system had to also be conceptualized as an ecological network. Ecological networks consist of a number of compartments with paths of currency exchange connecting the compartments. These compartments can be organized in a number of ways, such as by trophic levels or ecological functional groups such as detritivores, herbivore, etc. In this case, the network is structured with each species having its own compartment. There are two non-species compartments in the network, one for the source of energy that enters the system and the other as the sink where energy is lost from the system. The source compartment connects only to the producer species, but all species lose energy to the sink compartment. Otherwise, the currency exchange pathways follow the transfer of energy from species to species during feeding by the consumer species (Figure 5.3).

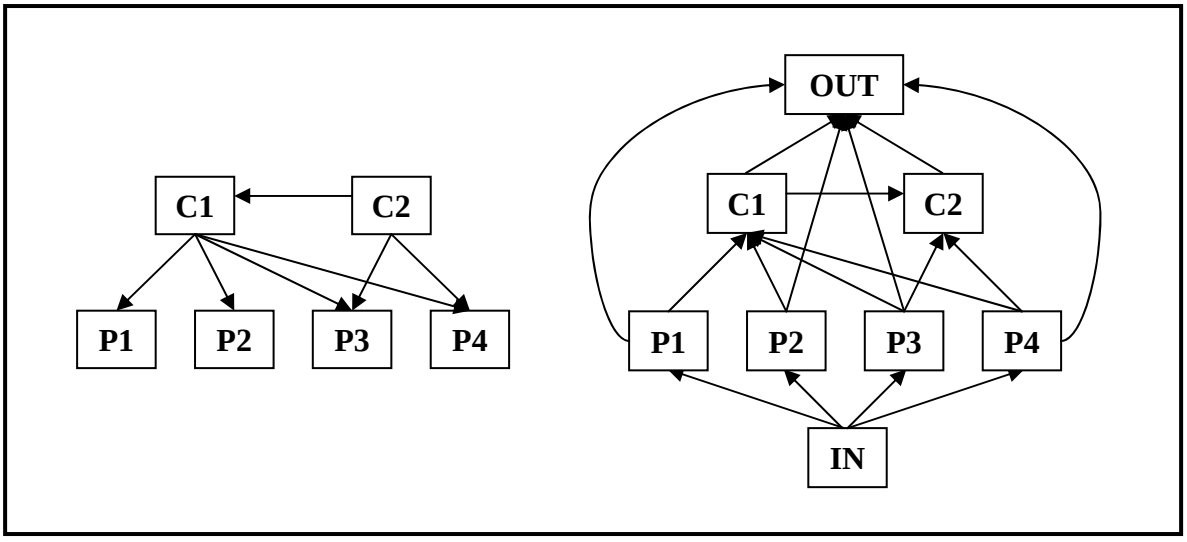


Figure 5.3 Ecosystem Food Web and Corresponding Network for calculating ascendency

During simulation, the flow of energy between compartments was tracked for each time step and the ascendency calculated at the end of the time steps. As mentioned in the previous section, ascendency is a combination of the vigor and the organization of the system (Ulanowicz 2000). Vigor, in this instance, is the throughput of currency, the total amount of energy that has moved through the system during the time step. Organization is the average mutual constraint of the system, which is in turn the sum of the pairwise mutual constraints of each pair of compartments multiplied with the probability of that particular pair occurring. Thus,

$$\text{Ascendency} = \text{Vigor} \times \text{Organization}$$

or

$$A = T \times \sum_{i,j} \left(\frac{T_{ij}}{T} \right) \log \left(\frac{T_{ij} T}{\sum_k T_{kj} \sum_q T_{iq}} \right) \tag{5.2}$$

where T is the total system throughput and T_{ij} the flow between compartments i and j.

It was found that, because the currency in the network was energy, and energy entered the system with a diurnal pattern, the ascendency also followed a diurnal pattern. This resulted in a great deal of variation in the ascendency vector (Figure 5.4a), reflecting that, in this case, hourly measurement was too fine a granularity for examining comportment. Daily overall ascendency and daily average ascendency were also both considered for use, having courser granularity. However, a seven day moving average was found to produce a smoother, more easily "read", line and was used instead. The resulting line represents the ascendency of the system through time with the high frequency noise removed but still reflecting the effect of the annual fluctuation of energy input (Figure5.4b). This is a two dimensional representational of the multi-dimensional path of a given system's comportment.

With a single line representing the whole system, it was then possible to use the fractal dimension of the line to indicate the comportmental complexity of that system. The fractal dimension was calculated using the freeware Fractal program by Vilis Nams (Nams 2006). The program uses the divider method, by which a pair of dividers is walked along the path in question to measure the length of the path. Larger and larger dividers are used, and the slope of the plot for log(path length) vs. log(divider size) is 1-*D*, where *D* is the fractal dimension of the path. A straight line would have a fractal dimension of 1 while a plane would have a fractal dimension of 2, thus *D* for all comportmental paths in this project fall in between 1 and 2. The FractalMean option in the Fractal program, which walks the path once from each end for each divider size, results in better accuracy, and was thus used for the systems in this project. To assure that the Fractal program fulfilled the needs of this project, it was tested with a variety of

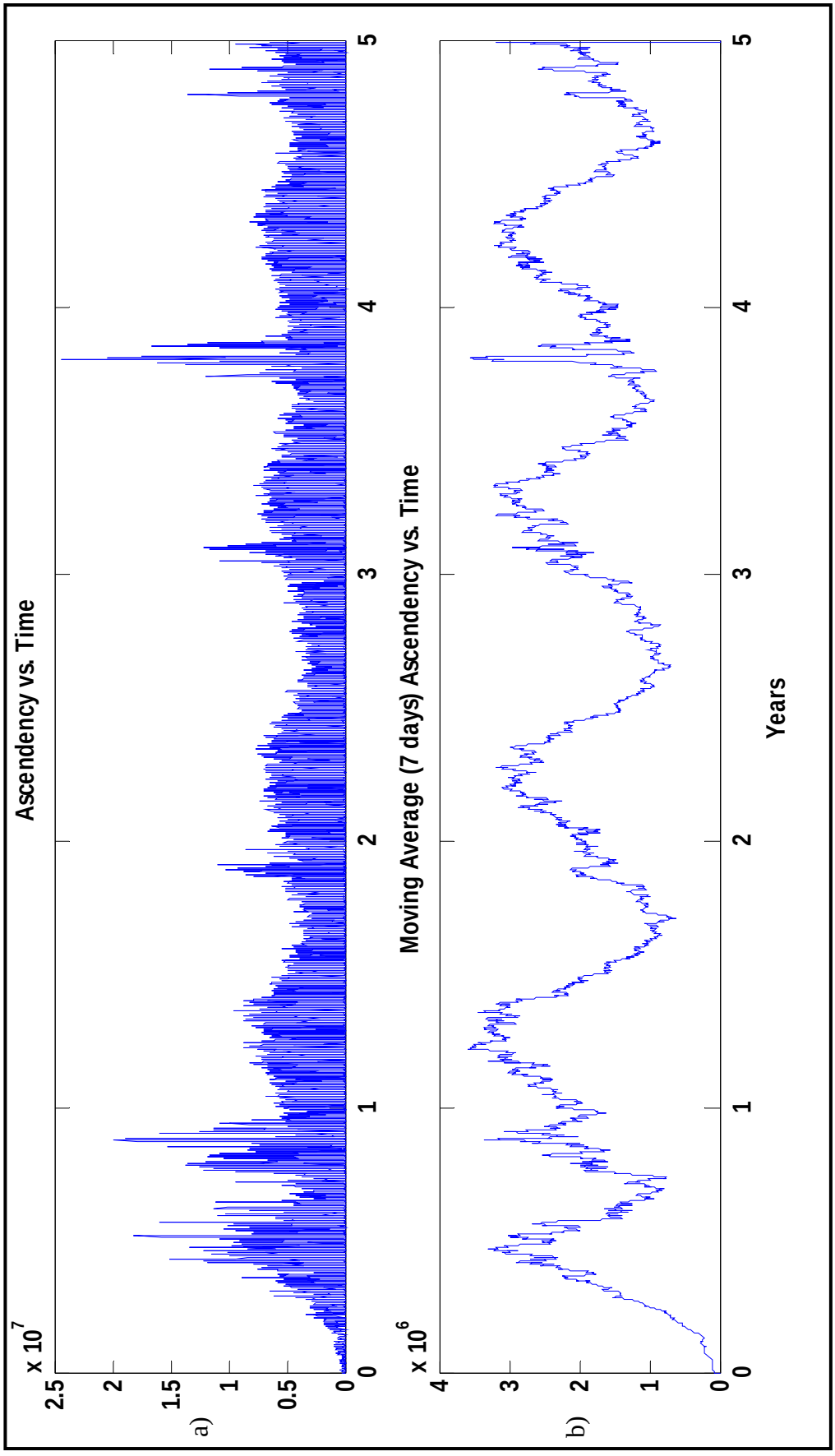


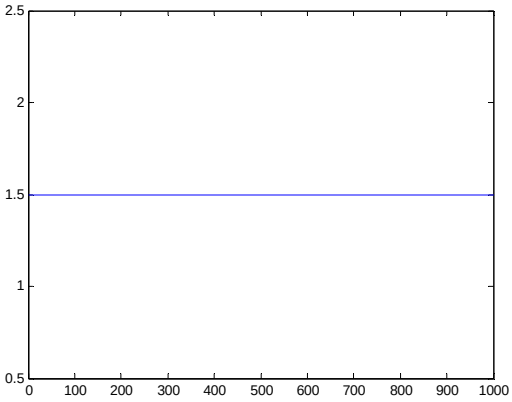
Figure 5.4 (a) Unfiltered Ascendency Vector and (b) Seven Day Moving Average Ascendency Vector

lines including straight lines, exponential curves, parabolas, and various spirals as well as sample ecosystem comportments (Figure 5.5). It was determined that the resulting values for fractal dimension would be satisfactory for quantifying comportmental complexity.

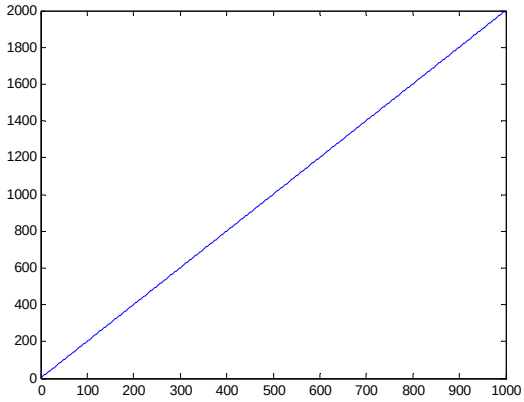
5.2 Measuring stability

In general, stability is a system's ability to remain unchanged. Ecological stability, however, is different from stability in many other complex systems. Because of the living nature of ecosystems, by definition there are going to be elements entering and leaving the system. Individuals die and are born or migrate between ecosystems, and there will be some population fluctuation due to those events as well as due to seasonal changes. The organisms themselves are constantly changing mass, energy content, and chemical composition. Thus, ecosystems can only conform to the general definition of stability within very short spans of time. However, an ecosystem that remains in the same, relatively small, area of its state space over time and is able to return to that area when perturbed is considered stable.

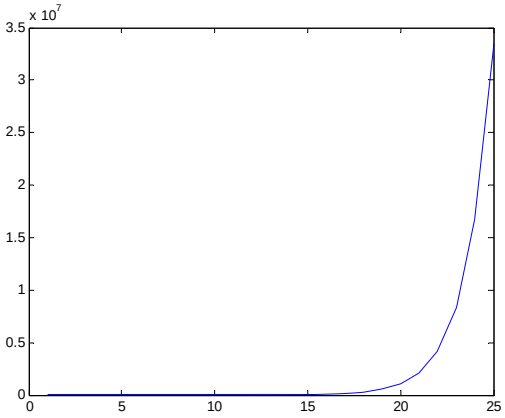
Because a stable ecosystem stays roughly in the same area of its state space, the comportment of such a system will either show little variation or may cycle, regularly or irregularly, around an attractor. Often such cycles occur in direct response to seasonal changes, but multi-year cycles are also common. Some systems also have multiple steady states (note that here the ecological use of the word "state" refers to the general constitution of the ecosystem in a stable area of its state space); instead of returning to the same region of the state space after every disturbance, such a system may switch back and forth between a number of steady states in response to different disturbances. The



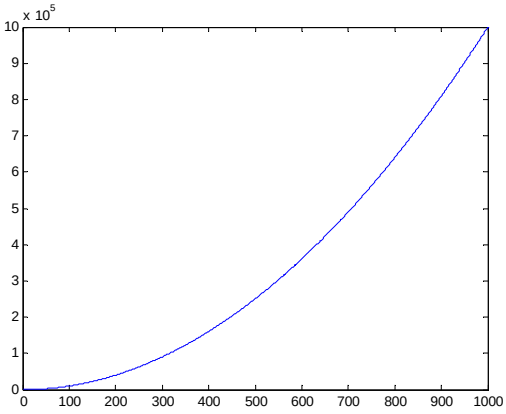
$D = 1.0000$



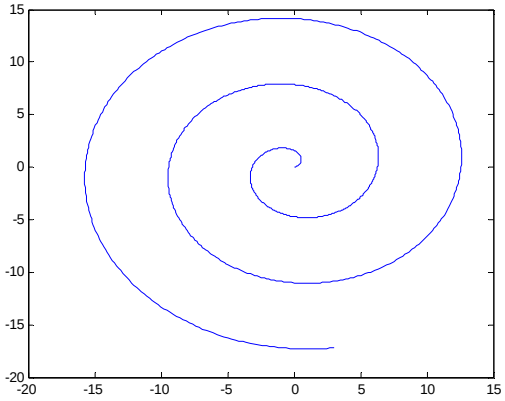
$D = 1.0000$



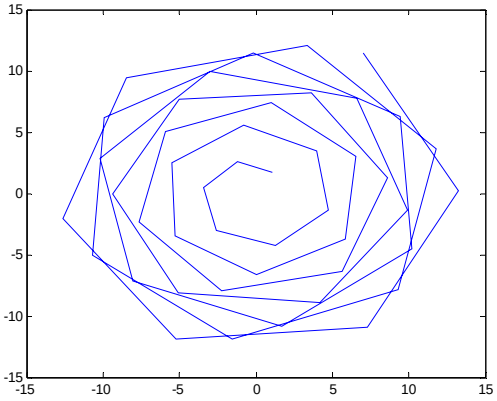
$D = 1.0013$



$D = 1.0001$



$D = 1.0387$



$D = 1.2134$

Figure 5.5 Test Lines Used with Fractal Program

stability of such systems can be difficult to identify. In general, however, ecosystems are assumed to have a single steady state, beyond which they become different systems. One traditional method for assessing the stability of ecosystems is based upon computing the eigenvalues of the community matrices. This approach has been used for systems that are modeled with a set of differential equations (Moore 1993, Pimm and Lawton 1977). In this method, systems where the eigenvalues are all negative are considered stable. As the ecosystems are here described in a quite different manner, this method is not suitable for use with the virtual ecosystem program of this project. The stability of ecosystems is also often assessed by means of their *resilience*, the speed at which a system returns to its steady state after a disturbance. Resilience is generally quantified by the time this takes, the *return time*, which can be calculated from the eigenvalues of the community matrix as per above (Kinzig et al. 1999). Again, this method is unsuitable for the virtual ecosystem program. Return time can also be obtained directly from a physical system or model (Stone et al. 1996).

The problem with return times and resilience, however, is that they are based on the assumption that the system in question has a known stable state to which to return. None of the ecosystems simulated in this project have a known stable state. Indeed, it was unknown whether the systems for this project would survive at all. Thus, it was not possible to consider steady states when assessing the stability of the virtual ecosystems. It might have been possible to calculate the stability of the comportment of each ecosystem in terms of whether the comportment itself followed a stable path (i.e. was constant or had a cyclical nature). Again, the expected high extinction rates made that not really desirable. It does not matter if a system has a stable comportment after a few years

if there are only two of the original thirty species left in the system.

Stability is being examined because the desired product of most ecological engineering projects is a sustainable system that will fulfill a specific function over time. Many ecological engineering projects rely on the ability of ecosystems to change with conditions to continue to provide the desired function, while others are intended to remain basically the same because the survival of certain species is part of the desired results for the project. For the first type of project, while it is expected that the composition of the system will change, it is still desirous to know how much change will occur under different circumstances and the ability of the system to continue providing the desired function. In the latter type of project, it is imperative to know how much of the initial system will survive. In either case, to truly engineer the system, the practitioner should know how much of the initial system can be expected to survive, how much the system can change, and, of course, whether the system will survive at all.

The virtual ecosystem is more like a project intended to preserve certain species. Indeed, because of its isolated nature, it depends entirely on the ability of at least part of the initial composition to persist. Accordingly, the purpose of the virtual ecosystem simulations was to test the ability of many different system compositions to survive. In this case, all that is needed to indicate the stability of the system is a calculation of how well the system survives, a comparison of the initial system constitution to the final system constitution. As the structure arises from the composition of the system, and is thus a secondary effect, structure does not need to be considered. Therefore, only the *persistence* of the system composition is significant in regards to system stability in this

case. In terms of measures, persistence is the ratio of the number of species surviving at the end of the simulation to the number of initial species, the value of which is between 0 and 1 inclusive. As species richness was already being measured at both the beginning and the end of each simulation, persistence was easily calculated by dividing the initial value by the final value. To allow for observations at different levels of the systems, persistence was calculated for producers and consumers, as well as the total of all species (denoted P_{N1}, P_{N2}, and P respectively).

To supplement persistence, the mean and standard deviation of the ascendency values measured at each time step of a given simulation were also calculated. As with the fractal dimension measure used for compositional complexity, the smoothed seven day moving average of the raw data was used to remove the excess noise from the data set. The mean ascendency of the system over the course of the simulation represents the overall health of the system, while the standard deviation will illustrate how much the health of the system varied. This variation will serve as another indication of the stability of the system.

5.3 Summary

Quantitative measures are required to compare systems to each other and to form hypotheses regarding system relationships and function. While many aspects of ecosystems can be assessed, complexity and stability were determined to be the most pertinent for the purpose of this project. The complexity of the system was broken down and measured as compositional, structural, and comportmental complexity. Species richness was chosen to measure compositional complexity, connectance was chosen for

structural complexity, and the fractal dimension of the system ascendency over time was chosen to quantify the comportmental complexity. System stability was assessed with the persistence of the system as well as the mean and standard deviation of the ascendency of the system over time.

6. Using Case-Based Reasoning to Predict and Engineer Ecosystems

Case-based reasoning (CBR) is modeled after human cognitive function. This function is the human ability to reason or problem solve by analogy, applying lessons learned from past experiences to new circumstances. One of the advantages of CBR is that it can be used to propose solutions or make fairly accurate predictions even in domains that are not completely understood (Kolodner 1993). It is also a comfortable method for human being to use and understand because of the similarity between CBR and human problem solving. In CBR, a set of already known situations and their outcomes, the case base, are used to propose or predict a solution or likely outcome for a new situation or problem. This is done by finding the closest matches to the new case among the known cases, then calculating the most likely solution from them. The more cases there are in the case base, the more accurate the proposed outcome or solution will usually be. The general process of performing reasoning when a new problem or situation is proposed to a reasoner is as follows (Kolodner 1993):

- Recall previous cases: the reasoner evaluates each case in the case base for similarity to the current situation
- Select best cases: the reasoner selects the cases that score best during the similarity matching
- Propose a solution: the reasoner uses the best cases to propose the desired solution or prediction

In many case-based reasoners, the new situation is added to the case base with the results from the performed reasoning. For some, the proposed solution is evaluated for prediction accuracy or whether the proposed solution results in a desired outcome. That

evaluation is used to adapt the case-based reasoner, improving the accuracy of the reasoner.

Ecological engineering involves many variables and large time frames, so it is difficult to extract domain level knowledge from the many projects and models. Case-based reasoning may provide a way to accumulate and evaluate data for ecological engineering knowledge at the domain level. Such knowledge, and the reasoning it enables, is necessary for the formulation of ecological engineering theory. Evaluating CBR for this possibility was one of the main purposes of this project, with the virtual ecosystem being used as a proof of concept tool. Thus, the constitutions and simulation results of the virtual ecosystems, in the form of final compositions and the values of the applied measures for each system, were used to form a case base for this project. Then a commercial case-based reasoner was used to explore the possibility of predicting the simulation results of virtual ecosystems. The case base and case-based reasoner were also used to attempt to roughly engineer more successful systems, given the same pool of species and forcing functions.

6.1 Compiling the case base

Of the case-based reasoners commercially available, many are intended for specific functions such as menu planning for the restaurant industry or diagnosis assistance for medical facilities. Others do not have the sort of predictive capability which was required for this project. The reasoner chosen for this project, *Induce It* by Inductive Solutions Inc (New York, NY, USA), has prediction capability and was also chosen for ease of use. *Induce It* is available as an add-on function to Excel, so it was possible to use the already

established Excel database (see Appendix F) of the simulations as the case base. The simulation numbers were used as case names and each case consisted of the initial and final numbers of species both by type and total, the presence (yes or no) of each species from the pool of species at the beginning and end of simulation, the connectance of the system structure, the persistence of the system composition by type and total, and the fractal dimension of the system comportment. The yes or no presence indicator for all forty species from the pool of species was the only change from the original database (used to record the simulation results and calculate the values for the applied measures – see Chapter 6). In the original database, the species were listed by number in a single data cell.

The first step in creating the case base from the original spreadsheet was to format the database, using *Induce It*, by defining which areas contained case names and which ones the case data. An area of parameter definitions was created in which the names of the data columns were recorded, as well as the type (numerical, character, etc.) of the data in each column. All of the columns in the case base were defined as "numerical" except for the species presence indicators for which "hierarchy" was used. The hierarchy that was installed for these columns was simple; the top level was merely 'presence' while the second level contained the 'y' and 'n' values that were used in the data area. "Hierarchy" was used instead of "character" because *Induce It* can only perform predictions for numerical and hierarchical data. In another area, the importance (weight) of each data column was indicated numerically. Because it was determined that there was no correlation between fractal dimension and connectance, fractal dimension and persistence, or persistence and connectance, those columns were given low weights while the species

presence indicator columns were given more weight. The weight values used were 5 for the species presence columns and 1 for all other columns.

Another area that had to be defined is the space, called the "inductive database", in which the similarity values for each data column of each case are recorded after being calculated by *Induce It*. The inductive database had the same dimensions as the case base. The values recorded in the inductive database indicate how similar each data column of each case is to the corresponding data column of the new case being proposed in the reference area. The reference area is another area and was defined with a dimension of one row by the number of data columns. It is important to note that when a new case is entered in the reference area, there does not need to be values in all of the data columns. Similarity values are calculated for any column in which data is present and does not have a zero weight. The values are then recorded in the inductive database. The overall similarity scores for each case are recorded in another area, appropriately called the "scores" region.

The inductive database can only be installed (i.e. calculated) when there are values in the reference region, and the case scores can not be calculated until the inductive database has been installed. There are four methods available for calculating case scores: linear weight, fuzzy logic, Euclidean, and cosine scores. The desired method is generally chosen during the process of tuning the case base. In tuning, each method is tried with a test case in the reference area. The cases in the case base are then ranked by similarity score and examined to see if the highest ranked cases are those that are desired as the top matches. The method that returns the closest matches to the ones desired is the method then used when the case-based reasoner is applied to later reference cases. For this project, linear weight scores returned the closest matches to those desired. The various

scoring methods were also tested with the two options for extrapolation method: least mean squares and polynomial. In that test, the results for linear weight scores with least mean squares extrapolation were the most reasonable. That combination of methods most consistently resulted in numerical values in the correct ranges for the data columns (e.g. between 0 and 1 for persistence) and more 'y' and 'n' responses for species presence instead of the undesired upper level of the hierarchy, 'present'. Thus, linear weight scores were the chosen scoring method for the rest of the project, and extrapolation was done with the least mean squares method. The case base can be found in Appendix G.

6.2 Using the case-based reasoner

6.2.1 Predicting simulation results

A set of cases to test the predictive ability of the case base and case-based reasoner was created and run in simulation using the same method as the original set of systems. The pre-simulation data – initial composition (numbers of species by type and total, presence of specific species) and structural complexity (connectance) – for each system were proposed to the case-based reasoner. Final system composition, stability (persistence), and comportmental complexity (fractal dimension) were then predicted with the case-based reasoner and compared to the system composition, stability, and comportmental complexity obtained through simulation.

The pre-simulation data for each case were entered into the reference area of the case-based reasoner, and the columns for which predictions were desired marked accordingly. Each time the case-based reasoner was used to make a prediction, the inductive database was installed anew. Then the scores were calculated, the cases in the case base ranked,

and the case-based reasoner used to extrapolate a possible outcome. The simulation results and the measures applied to the results were recorded and calculated as before, then compared to the predicted results.

6.2.1.1 Analyzing prediction accuracy

There are a number of practices frequently used to analyze how accurately a case-based reasoner makes predictions including percent error, Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), and Prediction at level n (Pred(n)) (Lee et al. 2006, Mendes 2002, Shepperd and Schofield 1997). The MRE of a prediction is the absolute value of the difference between the observed value (V_o) and the predicted value (V_p) divided by the observed value (Equation 6.1) (Mendes 2002).

$$MRE = \left| V_o - V_p \right| / V_o \tag{6.1}$$

Percent error is the same calculation turned into a percentage (Lee et al. 2006). The MMRE is the mean of the MREs for all the predictions, and the MdMRE is the median of the MREs for all the predictions (Mendes 2002). Prediction at level n is the percentage of predictions that are within n percent of the observed value. The most commonly used value for n is 25%, and it has been suggested that a good prediction system should meet that level of accuracy at least 75% of the time (Mendes 2002, Shepperd and Schofield 1997). One other method of prediction accuracy that has been used is to compare the predicted and observed values using a Pearson's correlation coefficient (Aqil et al. 2007).

While each of these methods was considered, the nature of the project made several of them undesirable. Because part of the project was to explore a large area of the state

space, including the boundary zones and areas where systems may not necessarily have any surviving species, values of zero for many types of data were expected to be obtained from the simulations. Observed values of zero are problematical for percent error and MRE, and thus MMRE and MdMRE, because the observed value is in the denominator of the equation. Using the statistical "trick" of using a very small value like 1E-11 in place of zero in the equation caused the percent error or MRE of a particular prediction to be very large even when the absolute error was relatively small (e.g. given an observed value of 0 and a predicted value of 0.1, the absolute error is only 0.1, but the MRE is 10,000,000,000).

As Pred(n) relies on a percentage as well, it too was undesirable in its standard form. However, the idea of calculating how often the predictions fell within a certain range of the observed value seemed to be a practical way of assessing the accuracy of the predictions made by the case-based reasoner. Thus a method was devised were predictions that fell within such a range, 'good' predictions, would score a 1, and predictions that fell outside the range, 'bad' predictions, would score 0. To set that range, the standard deviation was calculated for each column of out put numerical data (final numbers of producer, consumer, and total species; persistence of producer, consumer, and total species; and the complexity of the system comportment) in the case base. The twenty-five cases used for prediction were included in this calculation. Thus, a predicted value within one standard deviation of the observed value was given a score of 1, and predicted values outside that range were given a score of 0. Negative predictions were automatically given a score of 0 because negative values were not possible for any of the numerical values in the case base.

The accuracy of the presence indicators were calculated slightly differently since the data values for those columns were textual and had only two values in the cases. Direct matches, 'y' to 'y' and 'n' to 'n', were ranked with 1. Predicted values that were the opposite of the simulation results received a rank of 0. Although those were the only two values present in the cases, the predicted values also included 'present', the upper level of the hierarchy for the presence indicator columns in the case base. It was observed that the case-based reasoner returned this value for predictions when the top ranked cases were evenly split between 'y' and 'n' for the given column. Thus, instead of ranking predictions of 'present' as completely inaccurate, they were instead given a ranking of 0.5. The overall prediction accuracy of each case was calculated by taking the average of all the scores, and expressing it as a percent Then it was possible to calculate the Pred(25) of all the prediction cases and whether the case-based reasoner met that level of accuracy at least 75% of the time.

It was also possible to use the Pearson's correlation coefficient between the observed values and the values predicted by the case-based reasoner as a method of assessing prediction accuracy. The textual descriptors indicating the presence or absence of a given species were changed to numerical values; 'y' was changed to 1, and 'n' was changed to 0. As mentioned above, the case-based reasoner also returned 'present' in its predictions. Again, because of the circumstances that caused that descriptor to be predicted, the 'present' indicator was given a value of 0.5. With all the data being numerical, it was then possible to calculate the Pearson's correlation coefficient using the function native to Microsoft Excel.

6.2.2 Engineering systems

The case-based reasoner was also used to roughly engineer several systems. The initial values and results for the twenty-four simulations used for predicting simulation results were added to the case base, bringing the total number of cases to one hundred twenty-four. Then, various values for persistence were entered in the reference area of the case-based reasoner and used to extrapolate possible ecosystem compositions that would have those values for persistence when run in simulation. Most of the proposed persistence values were producer (P_{N1}) and consumer persistence (P_{N2}) values in three different combinations: 1 and 1, 1 and 0.5, and 0.5 and 0.5. Two different weight schemes were used as well. Ecosystem compositions were extrapolated with equal weight placed on the two persistence values and with more weight placed on the consumer persistence values. The second scheme was used because of the class imbalance, a common problem for machine learning and artificial intelligence applications in complex domains: so few of the cases in the case base had surviving consumers, which could result in those cases not being ranked highly enough to be considered in the extrapolation process (Charest et al. 2008, Japkowicz 2002). An overall persistence (P) of 1 was also proposed to the case-based reasoner.

For each combination of persistence values and weights, two sets of initial species were obtained. One set was based upon the columns for initial species presence in the top ranked cases, and the other set was extrapolated from the final species presence columns in the top ranked cases. While, in theory, it should have been possible to obtain accurate results using only the initial species presence columns, so few of the simulations in the case base had high persistences that it was reasoned that testing the final species presence

columns could be useful as well. It was expected that the systems based upon the final species columns would have better results when the desired persistence was 1, and the systems based upon the initial species would have better results when the desired persistence was 0.5. Although seven different combinations of persistence values and weights were proposed to the case-based reasoner with two sets of species obtained from each of those, only ten different systems resulted from extrapolation. In both the initial species and in the final species extrapolations, there were system compositions that resulted from multiple combinations of persistence values and weights.

Using the species compositions resulting from the case-based reasoner, ten sets of input files were created with a variation of the previous system creator program. The numbers of producer and consumer species, as well as the specific species numbers, were entered by hand. The program then found the species parameter values for those species from the pool of species. Food matrices and initial populations were determined in the same manner as the original one hundred runs and the twenty-four runs used for the prediction simulations. For comparison purposes, an eleventh system was also created without using the case-based reasoner. Instead, the system composition for this ecosystem was based upon the survival statistics of the species for the one hundred and twenty-four previous runs. Any producers that had a higher than 50% survival rate were used, and any consumers that had survived at all were used. Fortunately, the system as such had food sources for all of those consumers. The rest of the system constitution and the initial populations of the species were then determined in the same manner as the other ten 'engineered' systems.

6.2.2.1 Analyzing engineering success

The effectiveness of using the case-based reasoner to propose systems was tested in the same manner as the accuracy of using the case-based reasoner to predict simulation results. The persistence values obtained by simulation were compared to the proposed persistence values. Some simulation results were compared to more than one of the persistence value schemes, because the some of the schemes proposed to the case-based reasoner resulted in the same system constitutions. Thus, although there were only ten systems simulated (eleven if the system not obtained from the case-based reasoner is taken into consideration), accuracies were calculated for the fourteen persistence schemes proposed to the case-based reasoner, as well as the system based upon species survival statistics. Persistence values within one standard deviation for the given type of persistence were ranked 1 and those outside a standard deviation were ranked 0. Overall percent accuracy for each scheme was calculated from the average of the three values.

Although some systems were engineered using P_{N1} and P_{N2} and others using only P, accuracy analysis was based upon all three values so that the analysis would not be skewed by differing amounts of data being used for the various cases. Therefore, the values not used with the case-based reasoner had to be calculated. This was only possible because, in the cases where only P was used, a value of 1 was used each time, thus making the values of P_{N1} and P_{N2} automatically equal to 1 as well. For the cases where P_{N1} and P_{N2} were used, the value of P was calculated using P_{N1}, P_{N2}, N1, N2, and NTOT. The expected overall persistence of the system equals the sum of the proposed producer persistence multiplied by the number of producers obtained from the case-based reasoner and the proposed consumer persistence multiplied by the obtained number of consumers,

divided by the total number of species in the system, as obtained from the case-based reasoner (Equation 7.1).

$$P = (P_{N1} * \mathrm{N1} + P_{N2} * \mathrm{N2}) / \mathrm{NTOT} \quad (7.1)$$

For the single case engineered using species survival statistics, values of 1 were assumed for all types of persistence.

6.3 Summary

Case-based reasoning was used to further examine any patterns that might be found between system constitution and comportment in virtual ecosystems. System and simulation data was compiled into a case base. Then a commercial case-based reasoner was used to predict the simulation outcomes of a number of new systems and the accuracy of those predictions analyzed. The case-based reasoner was also used to roughly engineer a number of cases based on desired values of persistence. A number of schemes were used to propose possible system compositions. Simulation results were analyzed in the same manner as the original one hundred simulations, and the accuracy of the case-based reasoner in proposing systems was also analyzed.

7. Results and Discussion

One hundred ecosystems were created with the system creator program to be run in simulation to form the core of the case base for this project. The computers available to run the simulations were a SunFire 880 with eight processors and a SunBlade 1000 with a single processor, which made it possible to run nine simulations concurrently. Batch files were created so that when one simulation finished, another started, and all processors were kept running at all times. The Solaris operating system used on the two machines allowed the progress of the simulations to be monitored both through a process manager and by the ability to open a snapshot of an output file even while the simulation program was still writing data to it. The total time required to simulate the one hundred ecosystems was approximately four months.

7.1 Simulation results

Once the simulations were completed, results were compiled in a spreadsheet (Appendix F) and analyzed with the measures for complexity and stability decided upon earlier (see chapter 4). Of the one hundred runs, six had all species go extinct before the end of the allotted simulation time. Only two simulations had even one consumer species survive to the end. Eight simulations had all initial producer species still present at the end of the simulation time. The majority of the ecosystems completed the simulation time as producer-only systems, although three of these ecosystems resulted in simulations that ended early because the population in the system went over the maximum value allowed by the simulation program.

The populations in these systems grew over the maximum because the energy contents of

the individuals of one of the species in the systems (in two cases the only species left in the system) were small enough that the number of individuals could become larger than the maximum allowed population, even though the total energy of the species and system were still below the allowed maximum. The virtual ecosystem feature that slows population growth when the system energy becomes large did not affect the populations in these cases. This problem could have been avoided if, during the creation of the pool of species, the minimum energy level in every species was calculated to be larger than the maximum energy allowed in the system divided by the maximum population in the system.

Calculating the measures for these three systems presented a problem. They were not failed systems in that all the species went extinct, so the species present at the time that the simulations ended were considered to be the final compositions of the ecosystems, even though the simulations did not run to completion. Thus, the persistences of these systems were not considered zero. The fractal dimensions of the comportments for these three systems were calculated in the same way as for the rest of the systems run in simulation. However, the results may not reflect the true comportment of the three simulations because the ascendancy values after the simulations stopped were all zero, resulting in a straight line and a different fractal dimension after that point. Fortunately, these are the only three cases where there was such a problem, so the results should not have a significant impact on the overall statistics and patterns of the case base.

During the gathering of the results, it was possible to see a few basic patterns in the data. Certain producer species survived in most of the systems in which they were present,

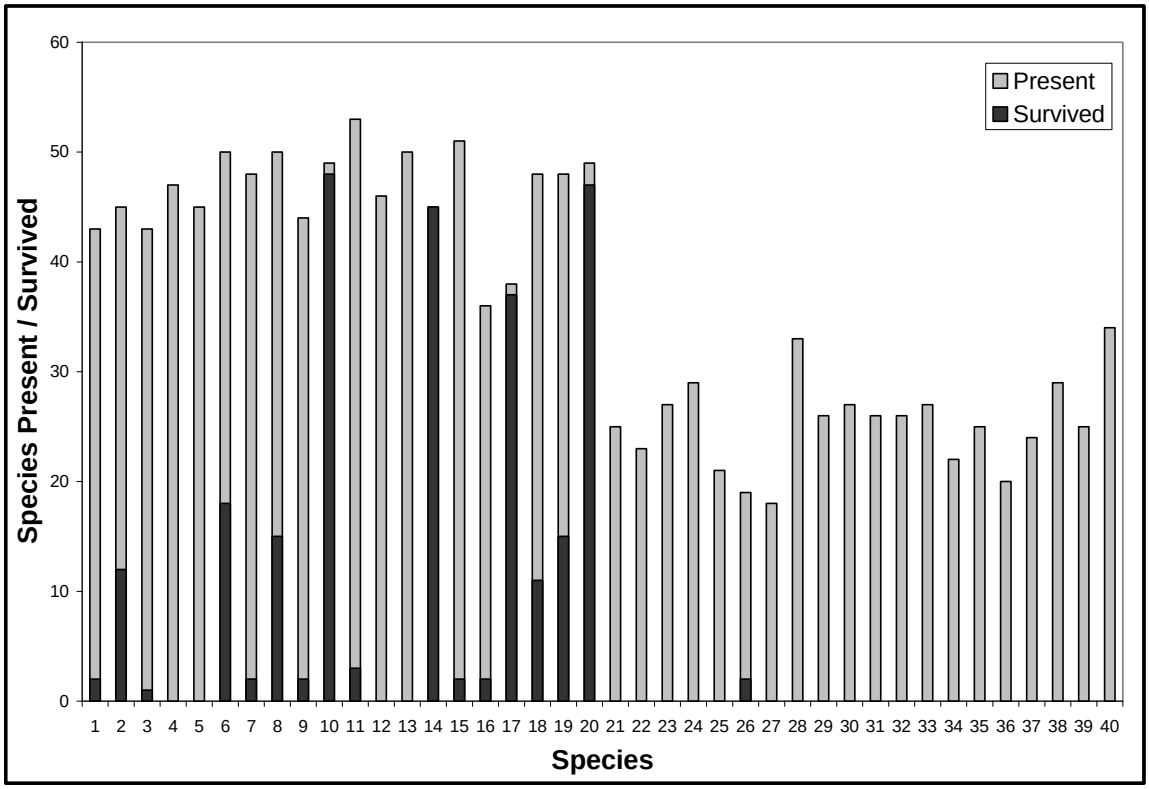


Figure 7.1 Species Survivorship for producers (1-20) and consumers (21-40).

although only one species survived in every system in which it was initially present (Figure 7.1). Some producer species only survived when those species mention above were not present, and some only survived when they were in combination with other species. There were also some producer species that did not survive in any simulation for which they were present in the ecosystem. The same consumer species survived in both of the ecosystems that had a consumer present at the end of the simulation, and two of the same producer species were present in both systems and survived until the end of the simulations as well.

There were, however, no obvious patterns in the measures applied to the simulation results (Table 7.1). The minimum species richness was 2 and the maximum 30, just as limited by the system creator program. The average species richness was 14. Overall

	NTOT	*C*	P_{N1}	P_{N2}	*P*	*D*	*A*
min:	2	0.00000	0.00000	0.00000	0.00000	1.03180	116.74
max:	30	0.83333	1.00000	0.12500	0.66667	1.15670	3900308.20
ave:	14	0.36110	0.34656	0.00236	0.20750	1.13976	1836287.96

Table 7.1 System Statistics for the First 100 Systems - minimum, maximum, and average values for species richness (*NTOT*), connectance (*C*), producer, consumer, and total persistence (P_{N1}, P_{N2}, and *P*), fractal dimension (*D*), and the average ascendency (*A*).

persistence for the systems varied between 0.00000 and 0.66667 with an average of 0.20750. The persistence for producer species had a minimum of 0.00000 and a maximum of 1.00000 with an average of 0.34656. For consumer species, the persistence had a minimum of 0.00000 and a maximum of 0.12500 with an average of 0.00236. Ecosystem connectances for the simulated systems varied between 0.00000 and 0.83333 with an average of 0.36110. The fractal dimension of the moving average of the system ascendancy during simulation had a narrow range of 1.03180 to 1.15670 with an average of 1.13976. The lower values for fractal dimension were mostly found for the systems where all species went extinct before the end of the simulation.

To see if there were any relationships between the constitution and comportment of each system, as well as between some of the comportmental phenomena, correlations were performed between the sets of values calculated for a number of the measures applied to both constitution and comportment. Correlations of intermediate strength were found between species richness and persistence, connectance and persistence, connectance and fractal dimension, average ascendency and connectance, and average ascendency and fractal dimension (Pearson's Correlation Coefficient, $\rho_{x,y}$ = -0.31872, -0.37027, -0.38847, -0.49974, and 0.41452 respectively). A relatively small correlation was found between fractal dimension and persistence ($\rho_{x,y}$ = 0.11039). Nearly no correlation was found between species richness and average ascendency ($\rho_{x,y}$ = 0.09686) while a stronger

correlation was found between average ascendency and persistence ($\rho_{x,y}$ = 0.54973).

7.2 Discussion of preliminary results for data production phase of the project

7.2.1 The virtual ecosystem program

In order to facilitate running simulations, the virtual ecosystem program used is a fairly simplified version of what an ecosystem would normally consist of. This creates conditions that actually only apply to a few physical systems. For example, because there is no landscape and the system is well-mixed, prey is available to consumers at all times. There are no places for prey to hide and no way for it to run away. The attribute values and interaction rules for consumers and prey are structured to recreate the effects of hunting ability and prey availability on how often consumers eat, but there is nothing included for escape ability of prey, from which could emerge some interesting dynamics between the various food species of even a single consumer, as well as changes in the survival of the consumer species. In many systems, such competition between prey species creates specialized niches where certain species find greater success at surviving. A system of niches can also develop from standard competition for resources.

Without a landscape, however, there is no way for species to develop such niches. Of course, there is no adaptation or inheritance in the virtual ecosystem either, so the individuals in the system are capable of neither creating niches – or other survival strategies – for themselves nor passing those niches down to the next generation. It is quite possible that systems would stabilize more readily if the individuals and species were able to develop survival strategies. For example, the producer species that could not survive when combined with other species in the simulations run with this virtual

ecosystem program might be able to establish themselves despite the “stronger” species in a virtual ecosytem program that included a variety of habitat types.

During verification of the virtual ecosystem program, systems were stabilized by manipulating the attribute values of the species. Such action is equal to genetically modifying a species instead of choosing a different one. At this time, there are various reasons that that strategy is not entirely viable for engineering physical ecosystems on Earth. For one, there are likely to be species available that are already adapted to the conditions of the ecosystem, and pre-adaptation is one of the basic principles Kangas (2004) proposed for ecological engineering (Chapter 2). Also, genetically modifying organisms is still in the early stages of development. However, when the time comes for humans to create new ecosystems in space habitats or on other planets, genetic manipulation may be an excellent method for stabilizing ecosystems in conditions not present on Earth.

7.2.2 System creation

The methods used for creating species and systems were chosen to make it possible to create a large number of systems in a reasonable amount of time. However, these methods may not necessarily have resulted in systems with the best chance for high persistence of species. The species were created somewhat randomly and not tested for individual viability, so it is reasonable that a number of the species created would not be able to survive under any simulation conditions. Two of the producer species and most of the consumer species did not survive in any of the systems in which they were present. While it is always more difficult for consumers to survive in the virtual ecosystem

program, making it difficult to know if it was the species "designs" of the consumers or system conditions which caused them to go extinct, those two producer species may represent species "designs" that just cannot survive in the virtual ecosystem. For the consumer species, it is also possible that their species "designs" would not necessarily cause them to fail if the prey interactions were structured differently. Food interaction matrices were randomly generated, which may have resulted in food preferences that were not "realistic" for a given species. However, it was not unexpected that many of the consumers would go extinct from the systems. Complexity-stability researchers as far back as May (1972) and Pimm and Lawton (1977) have noted that with theoretical systems, increased complexity or trophic levels make systems more likely to be unstable. There were also values that were held constant between the species which could affect the comportment of the systems (e.g. the size of the energy packets distributed to the individuals of producer species), but were treated so for simplicity of use at this very early stage research.

Generating systems randomly not only made it easier to create a large number of cases, it also made it more likely that the initial states of the systems would be better distributed in the state space, which was desired to maximize the knowledge gathered. However, even with only forty species and relatively few species and individual attributes, the state space is still vast, and most of it was not explored. There are 6.18479E+11 combinations of species possible under the current restraints in constructing the virtual ecosystems for this project. Even without changing the values for any other attributes, many more systems and simulations would be required to explore the state space fully. It is possible that there would be areas where the systems would show more persistence than seen in the systems

simulated to date.

7.2.3 Measures and correlations

Although the virtual ecosystem simulation program was mostly intended to provide a quick means of generating data to test case-based reasoning as a method for finding patterns and relationships in ecosystems that may be useful for forming ecological engineering theory, there were a few relationships that are suggested for investigation from the data in its preliminary form. The strongest correlation, however, between average ascendency and persistence, does not necessarily carry great import. While ascendency and persistence could be used one as an indicator for the other, which has possible uses for predictive efforts, there is no causation between the two and there is likely no way to use the correlation in the design portion of engineering pursuits.

The moderate correlations between species richness and persistence, connectance and persistence, average ascendency and connectance, and average ascendency and fractal dimension, on the other hand, are of interest for the continuing debate in ecology regarding the relationship between complexity and stability in ecosystems. As mentioned above, theoretical research often suggests that greater complexity leads to less stability (May 1972, Pimm and Lawton 1977). However, observation of natural systems show the exact opposite, and there have been numerous studies attempting to reconcile the contradiction (Christianou and Kokkoris 2008, Kondoh 2007, Neutel et al. 2007, Uchida and Drossel 2007, Chen and Cohen 2001, Pérez-España and Arreguín-Sánchez 2001, Rozdilsky and Stone 2001, Manne and Pimm 1996, Pimm 1984). Many of these studies have been aimed at finding the features in natural systems that allow stability and

complexity to coexist. The correlations found in this project between the constitutional complexity and stability measures were all negative, supporting the theoretical side of the debate. However, the object based approach used with the virtual ecosystem makes the difference between theoretical and empirical models more obvious than some of the approaches used before. As mentioned above, the set of systems simulated here is a very small fraction of the systems that could be created from the pool of species. It is possible that, as with the persistence of systems in general, there may be areas where the relationship between complexity and stability is different than the average for the entire state space and successful physical systems, with the aforementioned features developed through millennia of evolution, happen to fall into those areas. Again, more simulations to find areas in the state space of greater persistence would be required to test this hypothesis.

7.3 Results for the case-based reasoner

7.3.1 Accuracy of predictions

After the initial set of simulations was completed, the initial and final compositions of the systems, the species presences at the beginning and end of simulation, and the values obtained for the measures were compiled into the case base as previously described (see chapter 6). Then the initial composition and species presences of twenty-five systems created in the same way as the initial one hundred were proposed to the case-based reasoner to obtain predictions for final composition, final species presences, and the values of the measures. One simulation stopped in the middle with species still present in the system, so that system was removed from the analysis. The remaining predictions were then compared to observed values for the variables as obtained from simulating the

systems with the virtual ecosystem program. The average accuracy for the twenty-four sets of predictions was 79.7% accurate. The minimum accuracy was 14.9% and the maximum was 100%. Only four of the predictions were less than 75% accurate, meaning Pred(n) was 83.3% with n set at 25. The Pearson's correlation coefficient was calculated between the observed and predicted values for each case. The average correlation coefficient was 0.43083, the minimum was -0.82145, and the maximum was 0.99726.

7.3.2 Results for the "engineered" systems

None of the eleven systems had all species go extinct, although no system finished simulation with all species present either. Three simulations ended early when the total population of the system went over the amount allowed by the program, but as with the original data runs, those were not counted as failed systems. Only one system had any consumer species survive. There were, however, two systems that had all producer species survive: one of the systems whose compositions were generated with the case-based reasoner and the system created "by hand" based upon species survival rates.

The same set of measures as before was applied to the system constitutions and simulation results, although connectance and fractal dimension were not considered as important as persistence in this part of the experiment (Table 7.2). The overall

	NTOT	*C*	P_{N1}	P_{N2}	*P*	*D*	*A*
min:	4	0.33333	0.25000	0.00000	0.18182	1.05480	253313.33
max:	15	0.50000	1.00000	0.50000	0.75000	1.15300	2468320.93
ave:	8.5	0.42046	0.58898	0.04545	0.40527	1.11280	1512728.69

Table 7.2 System Statistics for "Engineered" Cases - minimum, maximum, and average values for species richness (*NTOT*), connectance (*C*), producer, consumer, and total persistence (P_{N1}, P_{N2}, and *P*), fractal dimension (*D*), and average ascendency (*A*).

persistence for the systems varied between 0.18182 and 0.75000 with an average of 0.40527. Producer persistence ranged between 0.25000 and 1.00000 with an average of 0.58898. The persistence of consumer species was 0.00000 in all cases except for one, which had a persistence of 0.50000, making the average consumer persistence 0.04545. The system connectances were between 0.33333 and 0.50000 with an average of 0.42046. Fractal dimension values ranged between 1.05480 and 1.15300 with an average of 1.11280.

As with the original one hundred runs, correlations were performed between the sets of values for the measures applied to the constitution and comportment of the engineered systems. Fairly strong correlations were found between species richness and persistence, connectance and fractal dimension, average ascendency and species richness, average ascendency and connectance, and average ascendency and fractal dimension (Pearson's Correlation Coefficient, $\rho_{x,y}$ = -0.56049, 0.78658, 0.50203, 0.87740, and 0.91804 respectively). Only small correlations were found between connectance and persistence, fractal dimension and persistence, and average ascendency and persistence ($\rho_{x,y}$ = -0.22647, -0.28404, -0.39131).

The average accuracy obtained for the simulation results versus the proposed values of persistence for all the engineered systems, both the fourteen "engineered" with the case-based reasoner and the one "engineered" based upon species survival statistics, was 26.7% accurate. Minimum and maximum accuracy values were 0% and 100%, respectively. The average accuracy of the fourteen case-based engineered schemes, without the hand engineered system, was 26.2%. The minimum and maximum values

remained the same. Thirteen of the fourteen schemes were less than 75% accurate; meaning that Pred(n), with n set at 25, for the case-based reasoners ability to be used to "engineer" a system for a specific persistence was 7.1%. The results of the simulation of the system based on species survival statistics were 33.3% accurate.

Average accuracies were also calculated for the various schemes used with the case-based reasoner to propose possible system compositions. The average accuracy of the simulation results for all systems based upon initial species columns in the case base was 14.3% while the average accuracy for systems based upon final species columns was 38.1%. Systems generated with equally weighted producer and consumer persistences were, on average, 33.3% accurate. For systems generated with more heavily weighted consumer persistences, the accuracy of the simulation results were, on average, 22.2%. In a full break down: systems generated with equally weighted persistences using the initial species columns were, on average, 22.2% accurate; systems generated with equally weighted persistences using the final species present columns were, on average, 44.4% accurate; systems generated with unequally weighted persistences using the initial species columns were, on average, 11.1% accurate; and systems generated with unequally weighted persistences using the final species present columns were, on

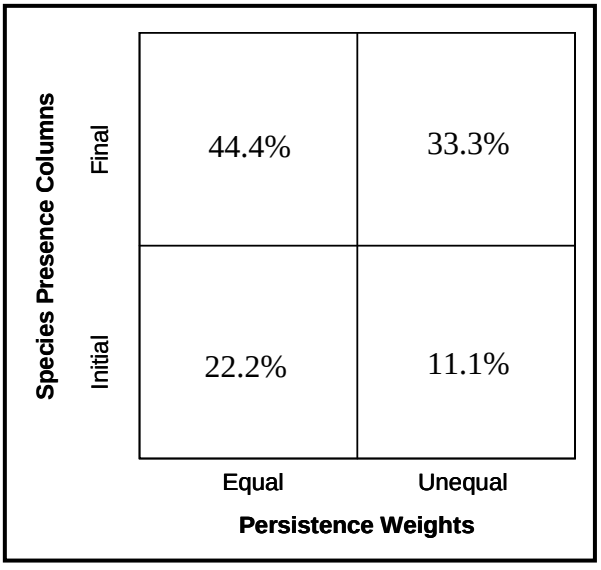


Figure 7.2 Average Accuracies of Various Schemes used to engineer systems with the case-based reasoner.

average, 33.3% accurate (Figure 7.2). Although it was expected that systems where the desired persistence was 1 would be more accurate when based upon final species columns and systems with a desired persistence of 0.5 would be more accurate when based upon the initial species columns, both persistences had better accuracy when used with final species columns, but systems with a desired persistence of 0.5 were more significantly improved.

7.4 Discussion of the case-based reasoner

7.4.1 Improving predictions

The ability of the case-based reasoner to predict simulation results from initial ecosystem constitution was within accepted levels, which conforms to the findings of other experiments showing that relatively small case bases can be sufficient for making predictions (Mendes 2002, Lee *et al.* 2006). Although there were already a few systems for which the simulation results were very close to the predicted results, the case-based reasoner could still be improved as a prediction tool by having more cases available. The more accurate predictions occurred when the top ranked cases had higher similarity scores to the reference case than happened for the other systems. With more cases in the case base, there would be more times when the top ranked case matches would have high similarity scores. Of course, prediction accuracy could also be improved if a case-based reasoner with a feedback function (a routine where predicted results are compared to actual results and the analysis used to tune the prediction function) were used. *Induce It* does not have a feedback function, but for the framework of this project, such a function was not really required.

When considering future use or improved functionality, it is important to note that the case based is only suitable for use with systems based on the pool of species created for this project, although different simulation programs might be used. The case base would have to be adapted if someone wanted to use it with an entirely different model or to add cases based on different species. It would be possible to add more species presence indicator columns, but the case base could quickly become difficult to manage with a larger number of columns. With more experience in using the case base and case-based reasoner, it could be possible to no longer use species presence columns. Instead a hierarchy with species "names" could be used, to which it would be easy to add more species. Of course, a different case base reasoner could also be used if so desired by the practitioner.

7.4.2 Discussion of the "engineered" cases

Using the case-based reasoner to "engineer" systems to have specific persistences did not result in actual persistence values very similar to the proposed persistence values. The value of Pred(n) for this use was far below the acceptable level. There were a couple of cases, however, that did come very close to meeting the targeted values for persistence, which implies that a case-based reasoner could be used to engineer systems. The knowledge requirements for engineering are greater than for prediction, so it is likely that if there were more cases in the case base, the actual results of the engineered cases would be closer to the targeted results.

It was also possible with the case-based reasoner to find patterns in which species would be likely to survive in a system that could not be found with basic statistical analysis. The

system “engineered” on the basis of species survival statistics was only slight more successful than the average for the systems “engineered” with the case-based reasoner; its overall persistence was 0.50000 and the average of the other systems was 0.39580, and it did not have the highest persistence of all the engineered systems. The accuracy of the simulation results for the system based upon species statistics, compared to its desired persistence, was only a little higher than the average accuracy of the other systems: 33.3% versus 26.2%. It was, however, higher than the average accuracy for systems “engineered” with a desired persistence of 1, which was 16.7%, but that was again probably because there were so few systems with high persistence in the case base. One other interesting point is highlighted by the system “engineered” on the basis of species survival statistics. It was one of two engineered systems that had all of the initial producer species survive the entire simulation. The two systems had different constitutions, with only two producer species (out of 5 and 4) and one consumer species (out of 2 and 4) in common between the two systems. This clearly demonstrates that the case-based reasoner able to find a workable system constitution, at least in terms of producer survival, not discernable through normal methods like survival statistics.

7.4.3 Comparing the “engineered” systems to the randomly generated systems

A number of interesting discussion points arose from comparing the engineered systems to the original set of randomly generated systems used to form the case base. The engineered systems were more successful than the initial set of simulations in that the average persistence values of the engineered systems were higher than the average persistence values of the initial set of simulations. The average overall persistence in the original systems was 0.20750 while the average overall persistence for the engineered

systems was 0.40527. The average persistence values for producers and consumers were also higher for the engineered systems: 0.58898 versus 0.34957 and 0.04545 versus 0.00236 respectively (Table 7.3). Thus, it seems that the case-based reasoner was able to find some patterns in what makes a persistent system under the constraints applied in this project.

	P_{N1}	P_{N2}	P
Random systems	0.34957	0.00236	0.20750
"Engineered" systems	0.58898	0.04545	0.40527

Table 7.3 Comparison of Average Persistence Values between randomly created and "engineered" systems

It is also interesting to note that, although there was no significant correlation found between connectance and persistence in either the randomly created systems or the engineered systems, the range of connectance values was much narrower for the more persistent, engineered systems (0.33333 to 0.50000 instead of 0.00000 to 0.83333), implying that some underlying pattern may have surfaced with the use of the case-based reasoner. Also, a stronger correlation was found between the connectance of the system structure and the fractal dimension of the system comportment in the engineered, and the correlation changed from being negative to positive. Unlike with connectance, the ranges and averages for fractal dimension do not differ much between the two sets of systems, so this correlation may be a coincidence, or it may point to a difficult to define pattern only found by the case-based reasoner. Further testing would be required to determine if the correlation is of any significance.

There are a number of other differences in the results of the correlations performed on the engineered cases (Table 7.4). The correlations between species richness and persistence, average ascendency and connectance, and average ascendency and fractal dimension all

	Random systems	"Engineered" systems
P vs. NTOT	-0.31872	-0.56049
P vs. C	-0.37027	-0.22547
P vs. D	0.11039	-0.28404
D vs. C	-0.38847	0.78658
A vs. NTOT	0.09686	0.50203
A vs. C	-0.49974	0.87740
A vs. D	0.41452	0.91804
A vs. P	0.54973	-0.39131

Table 7.4 Comparison of Correlation Values between randomly created and "engineered" systems

had greater magnitude with the engineered cases, and the correlation between average ascendency and connectance changed from negative to positive. The magnitude of the correlation was found between fractal dimension and persistence remained small with the engineered cases, but it changed from a positive to negative correlation. In the original set of runs, very little correlation was found between species richness and average ascendency; however, in the engineered cases, a relatively strong correlation was found. Conversely, the correlation between average ascendency and persistence changed from a strong correlation to a weak one, as well as changing from positive to negative.

The stronger, and still negative, correlation between species richness and persistence found in the engineered systems would seem to put this research even more firmly on the theoretical side of the complexity-stability debate mentioned above. However, the strong positive correlation between average ascendency and connection is consistent with the idea that systems with more complex structures may be more persistent once they have had a chance to organize in the conditions to which they are exposed. The changes occurring in both magnitude and direction between some of the other measures applied to the systems may indicate that the results from the virtual ecosystem do not lie entirely on one side or the other of the debate. Particularly in the cases where the system was based upon the final species presence columns, the engineered systems were defined based upon system compositions that had been allowed to develop and go through some self-organization, while the randomly generated systems involved a wide variety of systems

composed of not necessarily co-adapted species. It is possible that the changes in the correlation values do not actually reveal any sort of diversity-stability rule but are rather evidence of the self-organizational properties of the virtual ecosystem. Further, this implies that relationships between ecosystem constitution and comportment may vary in the different areas of the state space. There may not be universal rules that cover the small areas where persistent ecosystems, such as those we can observe in nature, can be found and the outlying and boundary areas. Such possibilities are why it is important to include such a large number of system constitutions, including those that are not "successful" in any knowledge gathering intended to further the development of comprehensive theory for engineering ecosystems.

7.5 Recommendations for future work

The work to this point does not, and was not intended to, lead directly to ecological engineering theory. However, the results of this project can be used for further exploration and also suggest some future projects. For example, it might be possible to find more patterns in the constitution – comportment relationship simply by enlarging the case base with more simulations created and run with the same method as used in this project. This would be particularly useful for further testing of the feasibility of using case-based reasoning to "engineer" ecosystems. The case base could also be enlarged with simulation results from other models. One such model could be used with the same pool of species but have adaptation and inheritance to allow for further study of emergence and self-organization in predator-prey interactions (see section 7.2.1). Further possibilities include other virtual ecosystems, models based on physical species, and models incorporating functional types instead of species. A case base could also be

created with data from physical systems, much of which already exists in the literature. The results of performing case-based reasoning with any of these case bases could be further analyzed with other computational techniques, such as artificial neural networks, for example, to find rules that would be usable for ecological engineering theory. All of these approaches would be further steps in learning how to organize and analyze the knowledge that is necessary to form comprehensive ecological engineering theory. Further investigation could also be performed into the possibilities regarding the complexity-stability relationship in ecosystems implied by the differences between the randomly created and the "engineered" virtual ecosystems. Results from such investigation could be important to both the state of ecology and ecological engineering.

7.6 Summary

One hundred system constitutions were created with the system creator program. The systems were then run in simulation to obtain their comportments in the presence of a specific set of forcing functions. The results of these simulations were collected and the decided upon measures applied. A few general patterns were discernable, though they were fairly simple phenomena. The database was then converted into a case base, which was used with a case-based reasoner to test the possibility of using case-based reasoning as a tool for not only predicting ecosystem comportment from initial system constitution, but "engineering" ecosystem constitutions for specific target results. The case-based reasoner performed well as a prediction tool and also showed the potential of case-based reasoning for organizing and analyzing the knowledge required to form comprehensive theory for designing ecosystems for ecological engineering projects.

8. General Summary and Conclusions

Ecological engineering, the act of designing and manipulating or creating ecosystems, has been applied for many reasons in a number of situations, involving various different types of ecosystems. None of these projects, however, have been carried out based on any sort of comprehensive ecological engineering theory. Instead, they were mostly based upon experience, experimentation, trial and error, and a few related principles. Forming theory that would allow practitioners to easily engineer ecosystems for any conditions using any species, even those that have not been used before, will require a large body of knowledge regarding the relationships between ecosystem constitution and comportment, as well as ways to organize and analyze that knowledge. This project was intended to test case-based reasoning as a method of gathering and analyzing or using a large quantity of ecological knowledge for use with ecological engineering.

8.1 Methodology

To produce data to be used with the case-based reasoner, a virtual ecosystem simulation program was created. A pool of species was generated and combined into a number of systems that were then run in simulation with the virtual ecosystem program. Various measures quantifying the constitution and comportment of the systems were applied so that the virtual ecosystems could be compared to each other, and the simulation results and values for the measures compiled into a case base. The case-based reasoner was used to predict the comportment of virtual ecosystems, and the accuracy of those predictions assessed. The feasibility of using case-based reasoning for more than prediction was also examined by attempting to "engineer" several virtual ecosystems for targeted levels of persistence using the case-based reasoner

8.1.1 The virtual ecosystem and simulation program

The representational model of the virtual ecosystem is object-based, allowing self-organization and emergence to occur just as in physical systems. Conceptually, the virtual ecosystem is a well mixed, materially closed, and energetically open system. In it, a number of individuals interact according to a set of rule-based expressions. These individuals are members of various species, which are either plant-like producers or animal-like consumers, and have various attributes. The values of these attribute describe either the parameters of the individual which are inherited from the species level (e.g. energy at birth or minimum energy required for life), or the current state of the individual (e.g. current energy or age). Two forcing functions, radiation of energy and temperature, act upon the virtual ecosystem.

The computational model of the virtual ecosystem, which is integrated with the simulation platform, was programmed in FORTRAN 90/95. Simple text files are used for both input (such as species attribute values and simulation parameters) and output (such as population levels, forcing function values, etc) throughout the simulation. The forcing functions are contained in subroutines so that they may be easily changed if desired. For verification, a number of tests were performed upon the program, and program results were analyzed, in order to ensure that the program works properly and conforms to what would be expected of general ecosystem function. The output from the program was also compared to population ecology theory to approximate the process of validation, which does not normally apply to virtual ecosystems, to ensure that the general functions of the virtual ecosystems were consistent with physical ecosystems.

8.1.2 Running simulations and using the case-based reasoner

A large number of simulations were needed to create enough data for the case base. To create reasonable systems, a pool of species was created from which systems were randomly assembled with a system creator program. Initially, one hundred different system constitutions were defined for use with the virtual ecosystem program. These one hundred systems were run in simulation, and then the initial conditions, final conditions, and the results of the various measures were compiled into a database. Some basic patterns and statistics were mined from the database before it was reformatted for use with the case-based reasoning program.

Another set of systems was then created, and the case-based reasoner was used to make predictions regarding the final conditions and results of the measures for those systems. The new systems were run in simulation and the actual results compared to the predictions. With the second set of systems added to the case base, the case-based reasoner was next used to attempt to "engineer" several systems to have specific persistence values. A number of different schemes were used to generate the constitutions of those systems, and then the systems were run in simulation. Finally, the actual persistence values were compared to the desired values and the accuracy of the "engineering" was analyzed for the different schemes.

8.2 Results

The predictions made using the case-based reasoner were more than 75% accurate 83% of the time, meaning that the case-based reasoner performed within acceptable levels as a prediction tool. The effort to "engineer" virtual ecosystems with the case-based reasoner

was only more than 75% accurate 7.1% of the time. However, there were cases where the engineered systems performed fairly close to the targeted levels. Comparing the engineered systems to the randomly created systems also showed that some patterns were found with the case-based reasoner that were not obvious through standard analysis.

8.3 Conclusions

The viability of case-based reasoning as a way to organize and analyze a large body of ecological data for investigating the relationships between ecosystem constitution and comportment, as would be needed for formulating ecological engineering theory, was investigated to fulfill the objectives of this research. The use of a case-based reasoner showed potential in finding patterns not obvious through normal methods, as well as predicting ecosystem comportment. Further testing by enlarging the current case base or creating different case bases will allow more patterns to be found and determine how case-based reasoning can fit into the effort to formulate ecological engineering theory, as well as be a helpful tool for current projects. The attempt to use the case-based reasoner to roughly "engineer" some virtual ecosystems was less successful, and while it showed potential, further research will be required before the capability of case-based reasoning as a method for finding engineering type knowledge can be determined.

9. Contributions to Knowledge

The main impetus for this project was to find ways to organize and analyze large datasets regarding the relationships between ecosystem constitution and comportment for application as a tool in support of the formation of ecological engineering theory. The following contributions to knowledge occurred over the course of the project:

- Case-based reasoning can be utilized as a method to organize and analyze large bodies of knowledge. This tool is applicable to guiding individual projects but also as a way to help discover patterns and relationships that could contribute to ecological engineering theory.
    - The results of this project confirmed the predictive capability of case-based reasoning even when used with a case base of more varied data than is typical for a single project. *(Chapter 7)*
    - Preliminary exploration of the use of case-based reasoning to propose initial ecosystem constitutions for targeted performance goals, to "engineer" systems. *(Chapters 6 & 7)*
    - Differences seen in the comparison of the randomly generated systems and the "engineered" systems are indicative that case-based reasoning may provide a way to elucidate interactions between ecosystem constitution and comportment not necessarily discernible with traditional analysis methods. *(Chapter 7)*

- Verification and validation of a virtual ecosystem model incorporating a simulation program utilizing random system creation methodology in order to explore ecosystem

relationships in a large area of a state space, including outlying and boundary areas which would not typically be included in traditional approaches. *(Chapters 3 & 4)*

- A novel approach to quantifying the complexity of ecosystem comportment, the fractal dimension of the moving average of the ascendency over time, as a complement to methods commonly employed to measure the complexity of ecosystem constitution. *(Chapter 5)*

- Correlations between the randomly created systems and the "engineered" ecosystems were significantly different for certain pairs of measures (persistence and fractal dimension, average ascendency and species richness, average ascendency and connectance, average ascendency and fractal dimension) that can be related to the complexity-stability debate within the ecological research community. The methodology of including boundary and outlying areas of the state space in ecosystem study, such as utilized within this project, may prove beneficial within the context of this debate. *(Chapter 7)*

References

Abe, K., Y. Ishikawa, S. Kibe and K. Nitta. 2005. Simulation model for the closed plant experiment facility of CEEF. *Advances in Space Research* 35: 1597-1608.

Abelson, A. 2006. Artificial reefs vs. coral transplantation as restoration tools for mitigating coral reef deterioration: Benefits, concerns, and proposed guidelines. *Bulletin of Marine Science* 78: 151-159.

Allen, J. and M. Nelson. 1999. Biospherics and Biosphere 2, mission one (1991–1993). *Ecological Engineering* 13: 15-29.

Andrés, P. and E. Mateos. 2006. Soil mesofaunal responses to post-mining restoration treatments. *Applied Soil Ecology* 33: 67-78.

Aoki, I. and T. Mizushima. 2001. Biomass diversity and stability of food webs in aquatic ecosystems. *Ecological Research* 16: 65-71.

Aqil, M., I. Kita, A. Yano and S. Nishiyama. 2007. Analysis and prediction of flow from local source in a river basin using a Neuro-fuzzy modeling tool. *Journal of Environmental Management* 85: 215–223.

Baker, W.L., T.T. Veblen and R.L. Sherriff. 2007. Fire, fuels and restoration of ponderosa pine–Douglas fir forests in the Rocky Mountains, USA. *Journal of Biogeography* 34: 251-269.

Bell, L.C. 2001. Establishment of native ecosystems after mining-Australian experience across diverse biogeographic zones. *Ecological Engineering* 17: 179-186.

Blüm, V., M. Andriske, C.L. Paaen and D. Voeste. 2003. The "C.E.B.A.S. MINI-MODULE": a self-sustaining closed aquatic ecosystem for spaceflight experimentation. *Advances in Space Research* 31: 201-210.

Bock, M., G. Rossner, M. Wissen, K. Remm, T. Langanke, S. Lang, H. Klug, T. Blaschke and B. Vršča. 2005. Spatial indicators for nature conservation from European to local scale. *Ecological Indicators* 5: 322-338.

Bolliger, J. 2005. Simulating complex landscapes with a generic model: Sensitivity to qualitative and quantitative classifications. *Ecological Complexity* 2: 131-149.

Bosch, O.J.H., R.S. Gibson, K. Kellner and W.J. Allen. 1997. Using case-based reasoning methodology to maximize the use of knowledge to solve specific rangeland problems. *Journal of Arid Environments* 35: 549-557.

Brown, M.T., H.T. Odum, and S.E. Jørgensen. 2004. Energy hierarchy and transformity in the universe. *Ecological Modelling* 178: 17–28.

Bruland, G.L., M.F. Hanchey and C.J. Richardson. 2003. Effects of agriculture and wetland restoration on hydrology, soils, and water quality of a Carolina bay complex. *Wetlands Ecology and Management* 11: 141-156.

Burk, N.M. 1995. Replicating earth: Cracking the secrets to creating a closed ecosystem. *IEEE Potentials* 14: 32-33.

Carleton, J.N., T.J. Grizzard, A.N. Godrej and H.E. Post. 2001. Factors affecting the performance of stormwater treatment wetlands. *Water Research* 35: 1552-1562.

Casti, J.L. 1994. *Complexification: explaining a paradoxical world through the science of surprise*. HarperCollins Publishers. New York.

Charest, M., S. Delisle, O. Cervantes and Y. Shen. 2008. Bridging the gap between data mining and decision support: A case-based reasoning and ontology approach. *Intelligent Data Analysis* 12: 211–236.

Chen, X. and J.E. Cohen. 2001. Global stability, local stability and permanence in model food webs. *Journal of Theoretical Biology* 212: 223-235.

Chen, Y., K.W. Hipel and D.M. Kilgour. 2007. Multiple-criteria sorting using case-based distance models with an application in water resources management. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans* 37: 680 -691.

Childress, W.M., C.L. Coldren and T. McLendon. 2002. Applying a complex, general ecosystem model (EDYS) in large-scale land management. *Ecological Modelling* 153: 97-108.

Christianou, M. and G.D. Kokkoris. 2008. Complexity does not affect stability in feasible model communities. *Journal of Theoretical Biology* 253: 162– 169.

Clark, O.G. 1999. *Characterization of Cyborged Ecosystems*. PhD Thesis, McGill University – Department of Agricultural and Biosystems Engineering. Montreal.

Coen, L.D. and M.W. Luckenbach. 2000. Developing success criteria and goals for evaluating oyster reef restoration: Ecological function or resource exploitation? *Ecological Engineering* 15: 323-343.

Costanza, R., R. d'Arge, R.D. Groot, S. Farber, M. Grasso, B. Hannon, K. Limburg, S. Naeem, R.V. O'Neill, J. Paruelo, R.G. Raskin, P. Sutton and M.V.D. Belt. 1997. The value of the world's ecosystem services and natural capital. *Nature* 387: 253-260.

Covington, W.W., P.Z. Fulé, S.C. Hart and R.P. Weaver. 2001. Modeling ecological restoration effects on ponderosa pine forest structure. *Restoration Ecology* 9: 421-431.

Dame, J.K. and R.R. Christian. 2008. Evaluation of ecological network analysis: Validation of output. *Ecological Modelling* 210: 327–338.

Delatte, B. and A. Butler. 2003. An object-oriented model for conceptual ship design supporting case-based design. *Marine Technology* 40: 158-167.

Dlodlo, N., L. Hunter, C. Cele, R. Metelerkamp and A.F. Botha. 2007. A hybrid expert systems architecture for yarn fault diagnosis. *FIBRES & TEXTILES in Eastern Europe* 15: 43-49.

Dutta, S., B. Wierenga and A. Dalebout. 1997. Case-based reasoning systems: From automation to decision-aiding and stimulation. *IEEE Transactions on Knowledge and Data Engineering* 9: 911-922.

Fath, B.D. 2004. Network analysis applied to large-scale cyber-ecosystems. *Ecological Modelling* 171: 329–337.

Finlayson, C.M., J. Lowry, M.G. Bellio, S. Nou, R. Pidgeon, D. Walden, C. Humphrey and G. Fox. 2006. Biodiversity of the wetlands of the Kakadu Region, northern Australia. *Aquatic Sciences* 68: 374-399.

Folsome, C.E. and J.A. Hanson. 1986. The emergence of materially closed system ecology. In: N. Polunin, Editor, *Ecosystem Theory and Application, Environmental Monographs and Symposia*, pp. 269–288. Wiley, New York.

Friedli, H.R., L.F. Radke, N.J. Payne, D.J. McRae, T.J. Lynham and T.W. Blake. 2007. Mercury in vegetation and organic soil at an upland boreal forest site in Prince Albert National Park, Saskatchewan, Canada. *Journal of Geophysical Research (G, Biogeosciences)* 112: G01004, doi:10.1029/2005JG000061.

Frize, M. and C. Frasson. 2000. Decision-support and intelligent tutoring systems in medical education. *Clinical and Investigative Medicine* 23: 266-269.

Fulé, P.Z., W.W. Covington, H.B. Smith, J.D. Springer, T.A. Heinlein, K.D. Huisinga and M.M. Moore. 2002. Comparing ecological restoration alternatives: Grand Canyon, Arizona. *Forest Ecology and Management* 170: 19-41.

Fulford, R.S., D.L. Breitburg, R.I.E. Newell, W.M. Kemp and M. Luckenbach. 2007. Effects of oyster population restoration strategies on phytoplankton biomass in Chesapeake Bay: A flexible modeling approach. *Marine Ecology Progress Series* 336: 43-61.

Gallucci, V.F. 1973. On the principles of thermodynamics in ecology. *Annual Review of Ecology and Systematics* 4: 329-357.

Gascuel, D. 2005. The trophic-level based model: A theoretical approach of fishing effects on marine ecosystems. *Ecological Modelling* 189: 315-332.

Gattie, D.K., J.R. Schramski, S.A. Bata. 2006. Analysis of microdynamic environ flows in an ecological network. *Ecological Engineering* 28: 187–204.

Gattie, D.K., N.N. Kellam, H.J. Turk. 2007. Informing ecological engineering through ecological network analysis, ecological modelling, and concepts of systems and engineering ecology. *Ecological Modelling* 208: 25–40.

Gent, M.L. and J.W. Morgan. 2007. Changes in the stand structure (1975–2000) of coastal Banksia forest in the long absence of fire. *Austral Ecology* 32: 239-244.

Ghilarov, A.M. 2001. The changing place of theory in 20th century ecology: from universal laws to array of methodologies. *Oikos* 92: 357-362.

Gierl, L., D. Steffen, D. Ihracky and R. Schmidt. 2003. Methods, architecture, evaluation and usability of a case-based antibiotics advisor. *Computer Methods and Programs in Biomedicine* 72: 139-154.

Gotelli, N.J. 2001. *A Primer of Ecology*. Sinauer Associates, Inc. Sunderland, MA.

Grant, W.E. 1998. Ecology and natural resource management: reflections from a systems perspective. *Ecological Modelling* 108: 67–76.

Green, D.G. and S. Sadedin. 2005. Interactions matter—complexity in landscapes and ecosystems. *Ecological Complexity* 2: 117-130.

Gustafson, E.J., L.J. Roberts and L.A. Leefers. 2005. Linking linear programming and spatial simulation models to predict landscape effects of forest management alternatives. *Journal of Environmental Management* 81: 339-350.

Hanan, J., P. Prusinkiewicz, M. Zalucki and D. Skirvin. 2002. Simulation of insect movement with respect to plant architecture and morphogenesis. *Computers and Electronics in Agriculture* 35: 255-269.

Hart, P.B.S., A.W. West, J.A. Kings, H.M. Watts and J.C. Howe. 1999. Land restoration management after topsoil mining and implications for restoration policy guidelines in New Zealand. *Land Degradation and Development* 10: 435-453.

Hastings, J.D., L.K. Branting and J.A. Lockwood. 1996. A multiple-paradigm system for rangeland pest management. *Computers and Electronics in Agriculture* 16: 47-67.

Hengsdijk, H. and M.K. von Ittersum. 2003. Formalizing agro-ecological engineering for future-oriented land use studies. *European Journal of Agronomy* 19: 549-562.

Holdo, R.M., R.D. Holt, M.B. Coughenour and M.E. Ritchie. 2007. Plant productivity and soil nitrogen as a function of grazing, migration and fire in an African savanna. *Journal of Ecology* 95: 115–128.

Jacobo, V.H., A. Ortiz, Y. Cerrud and R. Schouwenaars. 2007. Hybrid expert system for the failure analysis of mechanical elements. *Engineering Failure Analysis* 14: 1435-1443.

Japkowicz, N. and S. Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent Data Analysis* 6: 429–449.

Jensen, A.L. 1996. Density-dependent matrix yield equation for optimal harvest of age-structured wildlife populations. *Ecological Modelling* 88: 125-132.

Jones, T.H. 1996. Biospherics, closed systems and life support. *Tree* 11: 448-450.

Jørgensen, S.E. 2006. Application of ecological engineering principles in lake management. *Lakes & Reservoirs: Research and Management* 11: 103-109.

Jørgensen, S.E. and B.D. Fath. 2004. Application of thermodynamic principles in ecology. *Ecological Complexity* 1: 267–280.

Juell, P. and P. Paulson. 2003. Using reinforcement learning for similarity assessment in case-based systems. *IEEE Intelligent Systems* 18: 60-67.

Kalapanidas, E. and N. Avouris. 2001. Short-term air quality prediction using a case-based classifier. *Environmental Modelling & Software* 16: 263-272.

Kalin, M. 2001. Biogeochemical and ecological considerations in designing wetland treatment systems in post-mining landscapes. *Waste Management* 21: 191-196.

Kang, Y.G., Z. Wang, R. Li and C. Jiang. 2007. A fixture design system for networked manufacturing. *International Journal of Computer Integrated Manufacturing* 20: 143-159.

Kangas, P.C. 2004. *Ecological Engineering: Principles and Practice*. CRC Press LLC. Boca Raton, FL.

Kaster, D.S., C.B. Medeiros and H.V. Rocha. 2005. Supporting modeling and problem solving from precedent experiences: the role of workflows and case-based reasoning. *Environmental Modelling & Software* 20: 689-704.

Kavanagh, L.J. and J. Keller. 2007. Engineered ecosystem for sustainable on-site wastewater treatment. *Water Research* 41: 1823-1831.

King, J.M.P., R. Bañares-Alcántara and Z.A. Manan. 1999. Minimizing environmental impact using CBR: an azeotropic distillation case study. *Environmental Modelling & Software* 14: 359-366.

Kinzig, A.P., S.A. Levin, J. Dushoff and S. Pacala. 1999. Limiting similarity, species packing, and system stability for hierarchical competition-colonization models. *The American Naturalist* 153: 371-383.

Kolodner, J. 1993. *Case-Based Reasoning*. Morgan Kaufmann Publishers, Inc. San Mateo.

Kondoh, M. 2008. Anti-predator defence and the complexity–stability relationship of food webs. *Proceedings of the Royal Society B* 274: 1617–1624.

Kozlowski, T.T. 2002. Physiological ecology of natural regeneration of harvested and disturbed forest stands: implications for forest management. *Forest Ecology and Management* 158: 195-221.

Krebs, C.J. 2006. Ecology after 100 years: Progress and pseudo-progress. *New Zealand Journal of Ecology* 30: 3-11.

Kwon, O., G.P. Im, and K.C. Lee. 2007. MACE-SCM: A multi-agent and case-based reasoning collaboration mechanism for supply chain management under supply and demand uncertainties. *Expert Systems with Applications* 33: 690-705.

Lan, C.-H. and C.-Y. Hsui. 2006. Insight from complexity: A new approach to designing the deployment of artificial reef communities. *Bulletin of Marine Science* 78: 21-28.

Laughlin, D.C., S.R. Abella, W.W. Covington and J.B. Grace. 2007. Species richness and soil properties in *Pinus ponderosa* forests: A structural equation modeling analysis. *Journal of Vegetation Science* 18: 231-242.

Lee, B.-H., M. Scholz, A. Horn and A.M. Furber. 2006. Constructed wetlands: prediction of performance with case-based reasoning (Part B). *Environmental Engineering Science* 23: 332-340.

Li, D. and Z. Guo. 2000. Some aspects of ecological modeling developments in China. *Ecological Modelling* 132: 3-10.

Li, X. and A.G. Yeh. 2004. Multitemporal SAR images for monitoring cultivation systems using case-based reasoning. *Remote Sensing of Environment* 90: 524-534.

Li, Z., Z. Shen, M. Yang, J. Zheng and J. Li. 1998. Computer-aided technology for regional pest management: Towards agricultural sustainability. *Ecological Engineering* 11: 37-43.

Little, L.R., A.E. Punta, B.D. Mapstone, F. Pantuse, A.D.M. Smith, C.R. Davies and A.D. McDonaldf. 2007. ELFSim—A model for evaluating management options for spatially structured reef fish populations: An illustration of the "larval subsidy" effect. *Ecological Modelling* 205: 381-396.

Lugo, A.E. 2002. Can we manage tropical landscapes? – an answer from the Caribbean perspective. *Landscape Ecology* 17: 601-615.

Maguire, B. 1980. Some patterns in post-closure ecosystem dynamics (failure). In: J.P. Giesy, Jr., Editor, *Microcosms in Ecological Research*, pp. 319-332. Technical Information Center, Oak Ridge, TN.

Mancinelli, G., L. Sabetta and A. Basset. 2007. Colonization of ephemeral detrital patches by vagile macroinvertabrates in a brackish lake: a body size-related process? *Oecologia* 151: 292-302.

Manne, L. and S.L. Pimm. 1996. Ecology: Engineered food webs. *Current Biology* 6: 29-31.

Månsson, B.Å. and J.M. McGlade. 1993. Ecology, thermodynamics and H. T. Odum's conjectures. *Oecologia* 93: 582-596.

Marín, V.H. and L.E. Delgado. 2001. A spatially explicit model of the Antarctic krill fishery off the South Shetland Islands. *Ecological Applications* 11: 1235-1248.

Marques, J.C., S.N. Nielsen, M.A. Pardal and S.E. Jørgensen. 2003. Impact of eutrophication and river management within a framework of ecosystem theories. *Ecological Modelling* 166: 147-168.

May, R.M. 1972. Will a large complex system be stable? *Nature* 238: 413-414.

Mbuligwe, S.E. 2005. Applicability of a septic tank/engineered wetland coupled system in the treatment and recycling of wastewater from a small community. *Environmental Management* 35: 99-108.

McCay, D.P.F. and J.J. Rowe. 2003. Habitat restoration as mitigation for lost production at multiple trophic levels. *Marine Ecology Progress Series* 264: 233-247.

McDonald, W.R. and C.C. St. Clair. 2004. The effects of artificial and natural barriers on the movement of small mammals in Banff National Park, Canada. *Oikos* 105: 397-407.

McIntire, E.J.B., R. Duchesneau and J.P.H. Kimmins. 2005. Seed and bud legacies interact with varying fire regimes to drive long-term dynamics of boreal forest communities. *Canadian Journal of Forest Research* 35: 2765-2773.

Mendes, E., I. Watson, C. Triggs, N. Mosley, and S. Counsell. 2002. A comparison of development effort estimation techniques for web hypermedia applications. *Proceedings of the Eighth IEEE Symposium on Software Metrics* (METRICS '02).

Miller III, H.L. and K.H. Dunton. 2007. Stable isotope (^{13}C) and O_2 micro-optode alternatives for measuring photosynthesis in seaweeds. *Marine Ecology Progress Series* 329: 85-97.

Mitsch, W.J. and S.E. Jørgensen. 2004. *Ecological Engineering and Ecological Restoration*. John Wiley & Sons, Inc. Hoboken, NJ.

Molenaar, R. 1998. *Design and Implementation of Biosystem Control and Tools for Biosystem Simulation*. PhD Thesis, McGill University - Department of Agricultural and Biosystems Engineering. Ste. Anne de Bellevue, QC.

Moore, J.C., P.C. de Ruiter and H.W. Hunt. 1993. Influence of productivity on the stability of real and model ecosystems. *Science* 261: 906-908.

Morowitz, H., J.P. Allen, M. Nelson and A. Alling. 2005. Closure as a scientific concept and its application to ecosystem ecology and the science of the biosphere. *Advances in Space Research* 36: 1305-1311.

Morris, J.T. 2007. Ecological engineering in intertidal saltmarshes. *Hydrobiologia* 577: 161-198.

Müller, D.B., H.-P. Bader and P. Baccini. 2004. Long-term coordination of timber production and consumption using a dynamic material and energy flow analysis. *Journal of Industrial Ecology* 8: 65-87.

Müller, F. 1997. State-of-the-art in ecosystem theory. *Ecological Modelling* 100: 135-161.

Nams, V.O. 2006. Improving accuracy and precision in estimating fractal dimension of animal movement paths. *Acta Biotheoretica* 54: 1-11.

Nams, V.O. and M. Bourgeois. 2004. Fractal analysis measures habitat use at different spatial scales: an example with American marten. *Canadian Journal of Zoology* 82: 1738-1747.

Nassauer, J.I. 2004. Monitoring the success of metropolitan wetland restorations: Cultural sustainability and ecological function. *Wetlands* 24: 756-765.

Nelson, M., T.L. Burgess, A. Alling, N. Alvarez-Romo, W.F. Dempster, R.L. Walford and J.P. Allen. 1993. Using a closed ecological system to study Earth's biosphere. *BioScience* 43: 225-236.

Nestlerode, J.A., M.W. Luckenbach and F.X. O'Beirn. 2007. Settlement and survival of the oyster *Crassostrea virginica* on created oyster reef habitats in Chesapeake Bay. *Restoration Ecology* 15: 273-283.

Neutel, A.M, J.A.P. Heesterbeek, J. van de Koppel, G. Hoenderboom, A. Vos, C. Kaldeway, F. Berendse and P.C. de Ruiter. 2007. Reconciling complexity with stability in naturally assembling food webs. *Nature* 449: 599-602.

Nitta, K. 2005. The Mini-Earth facility and present status of habitation experiment program. *Advances in Space Research* 35: 1531-1538.

Noble, J.C. and P. Walker. 2006. Integrated shrub management in semi-arid woodlands of eastern Australia: A systems-based decision support model. *Agricultural Systems* 88: 332-359.

Núñez, H., M. Sànchez-Marrè, U. Cortés, J. Comas, M. Martínez, I. Rodríguez-Roda and M. Poch. 2004. A comparative study on the use of similarity measures in case-based reasoning to improve the classification of environmental system situations. *Environmental Modelling & Software* 19: 809-819.

Odum, H.T. 1988. Self-Organization, transformity, and information. *Science* 242: 1132-1139.

Odum, H.T. 1994. Ecological engineering: The necessary use of ecological self-design. *Ecological Engineering* 3: 107-119.

Odum, H.T. 1995. Energy systems concepts and self-organization: A rebuttal. *Oecologia* 4: 518-522.

Odum, H.T. 2002. Explanations of ecological relationships with energy systems concepts. *Ecological Modelling* 158: 201-211.

Odum, H.T. and B. Odum. 2003. Concepts and methods of ecological engineering. *Ecological Engineering* 20: 339–361.

Oguntunde, P.G., N. van de Giesen and H.H.G. Savenije. 2007. Measurement and modelling of transpiration of a rain-fed citrus orchard under subhumid tropical conditions. *Agricultural Water Management* 87: 200-208.

Parrott, L. 2005. Quantifying the complexity of simulated spatiotemporal population dynamics. *Ecological Complexity* 2: 175-184.

Parrott, L. and R. Kok. 2000. Incorporating complexity in ecosystem modelling. *Complexity International* 7: Paper ID: lparro01.

Parrott, L. and R. Kok. 2001. A generic primary producer model for use in ecosystem simulation. *Ecological Modelling* 139: 75-99.

Parrott, L. and R. Kok. 2002. A generic, individual-based approach to modelling higher trophic levels in simulation of terrestrial ecosystems. *Ecological Modelling* 154: 151-178.

Parrott, L., R. Kok and R. Lacroix. 1996. Daily average temperatures: modeling and generation with a Fourier transform approach. *Transactions of the ASAE* 39: 1911-1922.

Patten, B.C. 1993. Toward a more holistic ecology, and science: The contribution of H. T. Odum. *Oecologia* 93: 597-602.

Patten, B.C. 1994. Ecological systems engineering: toward integrated management of natural and human complexity in the ecosphere. *Ecological Modelling* 75/76: 653-665.

Patten, B.C. 1995. Network integration of ecological extremal principles: exergy, emergy, power, ascendency, and indirect effects. *Ecological Modelling* 79: 75-84.

Peacor, S.D., S. Allesina, R.L. Riolo and T.S. Hunter. 2007. A new computational system, DOVE (Digital Organisms in a Virtual Ecosystem), to study phenotypic plasticity and its effects in food webs. *Ecological Modelling* 205: 13-28.

Pérez-España, H. and F. Arreguín-Sánchez. 2001. An inverse relationship between stability and maturity in models of aquatic ecosystems. *Ecological Modelling* 145: 189-196.

Perkol-Finkel, S. and Y. Benayahu. 2007. Differential recruitment of benthic communities on neighboring artificial and natural reefs. *Journal of Experimental Marine Biology and Ecology* 340: 25-39.

Pimm, S.L. 1984. The complexity and stability of ecosystems. *Nature* 207: 321-326.

Pimm, S.L. and J.H. Lawton. 1977. Number of trophic levels in ecological communities. *Nature* 268: 329-331.

Powers, S.P., J.H. Grabowski, C.H. Peterson and W.J. Lindberg. 2003. Estimating enhancement of fish production by offshore artificial reefs: uncertainty exhibited by divergent scenarios. *Marine Ecology Progress Series* 264: 265-277.

Raphael, B., B. Domer, S. Saitta and I.F.C. Smith. 2007. Incremental development of CBR strategies for computing project cost probabilities. *Advanced Engineering Informatics* 21: 311-321.

Remm, K. 2004. Case-based predictions for species and habitat mapping. *Ecological Modelling* 177: 259-281.

Rochefort, L., F. Quinty, S. Campeau, K. Johnson and T. Malterer. 2003. North American approach to the restoration of *Sphagnum* dominated peatlands. *Wetlands Ecology and Management* 11: 3-20.

Rodney, W.S. and K.T. Paynter. 2006. Comparisons of macrofaunal assemblages on restored and non-restored oyster reefs in mesohaline regions of Chesapeake Bay in Maryland. *Journal of Experimental Marine Biology and Ecology* 335: 39-51.

Rossille, D., J.-F. Laurent and A. Burguna. 2005. Modelling a decision-support system for oncology using rule-based and case-based reasoning methodologies. *International Journal of Medical Informatics* 74: 299-306.

Rozdilsky, I.D. and L. Stone. 2001. Complexity can enhance stability in competitive systems. *Ecology Letters* 4: 397-400.

Salisbury, F.B., J.I. Gitelson and G.M. Lisovsky. 1997. Bios-3: Siberian experiments in bioregenerative life support. *BioScience* 47: 575-585.

Santos, M.N. and C.C. Monteiro. 2007. A fourteen-year overview of the fish assemblages and yield of the two oldest Algarve artificial reefs (southern Portugal). *Hydrobiologia* 580: 225-231.

Scheiner, S.M. 1992. Measuring pattern diversity. *Ecology* 73: 1860-1867.

Schmidt, R. and L. Gierl. 2005. A prognostic model for temporal courses that combines temporal abstraction and case-based reasoning. *International Journal of Medical Informatics* 74: 307-315.

Schmidt, R., S. Montani, R. Bellazzi, L. Portinale and L. Gierl. 2001. Cased-based reasoning for medical knowledge-based systems. *International Journal of Medical Informatics* 64: 355-367.

Schneider, E.D. and J.J. Kay. 1994. Complexity and thermodynamics: Towards a new ecology. *Futures* 26: 626-647.

Shafir, S., J.V. Rijn and B. Rinkevich. 2006. Steps in the construction of underwater coral nursery, an essential component in reef restoration acts. *Marine Biology* 149: 679-687.

Shepperd, M. and C. Schofield. 1997. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering* 23: 736-743.

Shi, X., A.-X. Zhu, J.E. Burt, F. Qi and D. Simonson. 2004. A case-based reasoning approach to fuzzy soil mapping. *Soil Science Society of America Journal* 68: 885-894.

Shuwen, W., Q. Pei, L. Yang and L. Xi-Ping. 2001. Wetland creation for rare waterfowl conservation: A project designed according to the principles of ecological succession. *Ecological Engineering* 18: 115-120.

Stenseth, N.C., W. Falck, O.N. Bjørnstad and C.J. Krebs. 1997. Population regulation in snowshoe hare and Canadian lynx: Asymmetric food web configurations between hare and lynx. *Proceedings of the National Academy of Science* 94: 5147-5152.

Stigebrandt, A. and B.G. Gustafsson. 2007. Improvement of Baltic proper water quality using large-scale ecological engineering. *Ambio* 36: 280-286.

Stone, L., A. Gabric and T. Berman. 1996. Ecosystem resilience, stability, and productivity: seeking a relationship. *The American Naturalist* 148: 892-903.

Sun, B., L.D. Xu, X. Pei and H. Li. 2003. Scenario-based knowledge representation in case-based reasoning systems. *Expert Systems* 20: 92-99.

Sun, Y.C. and R. Kok. 2007. A solar radiation model with a Fourier transform approach. *Canadian Biosystems Engineering* 49: 7.17-7.24.

Svirezhev, Y.M. 2000. Thermodynamics and ecology. *Ecological Modelling* 132: 11–22.

Tanner, B.R., E. Perfect and J.T. Kelley. 2006. Fractal analysis of Maine's glaciated shoreline tests established coastal classification scheme. *Journal of Coastal Research* 22: 1300-1304.

Taylor, B., D. Robertson, N. Wiratunga, S. Crawd, D. Mitchell and E. Stewart. 2007. Using computer aided case based reasoning to support clinical reasoning in community occupational therapy. *Computer Methods and Programs in Biomedicine* 87: 170-179.

Tilley, D.R., H. Badrinarayanan, R. Rosati and J. Son. 2002. Constructed wetlands as recirculation filters in large-scale shrimp aquaculture. *Aquacultural Engineering* 26: 81-109.

Todd, N.J. 2005. *A Safe and Sustainable World: The Promise of Ecological Design*. Island Press. Washington, DC.

Trostel, K., A.R.E. Sinclair, C.J. Waiters and C.J. Krebs. 1987. Can predation cause the 10-year hare cycle? *Oecologia* 74: 185-192.

Turner, S.R., B. Pearce, D.P. Rokich, R.R. Dunn, D.J. Merritt, J.D. Majer and K.W. Dixon. 2006. Influence of polymer seed coatings, soil raking, and time of sowing on seedling performance in post-mining restoration. *Restoration Ecology* 14: 267-277.

Uchida, S. and B. Drossel. 2007. Relation between complexity and stability in food webs with adaptive behavior. *Journal of Theoretical Biology* 247: 713-722.

Ulanowicz, R.E. 2000. Toward the measurement of ecological integrity. In: D. Pimental, L. Westra, and R.F. Noss (Editor). *Ecological Integrity: Integrating Environment, Conservation, and Health*, pp. 99-113. Island Press, Washington, D.C.

Van den Brink, P.J., J. Roelsma, E.H.V. Nes, M. Scheffer and T.C.M. Brock. 2002. PERPEST model, a case-based reasoning approach to predict ecological risks of pesticides. *Environmental Toxicology and Chemistry* 21: 2500-2506.

Vasconcellos, M., S. Mackinson, K. Sloman and D. Pauly. 1997. The stability of trophic mass-balance models of marine ecosystems: a comparative analysis. *Ecological Modelling* 100: 125-134.

Verhoeven, J.T.A. and A.F.M. Meuleman. 1999. Wetlands for wastewater treatment: Opportunities and limitations. *Ecological Engineering* 12: 5-12.

Wallington, T.J., R.J. Hobbs, and S.A. Moore. 2005. Implications of current ecological thinking for biodiversity conservation: a review of the salient issues. *Ecology and Society* 10: 15.

Wam, H.K., O. Hofstad, E. Nævdal and P. Sankhayan. 2005. A bio-economic model for optimal harvest of timber and moose. *Forest Ecology and Management* 206: 207-219.

Wang, G.H., G.B. Li, C.X. Hu, Y.D. Liu, L.R. Song, G.H. Tong, X.M. Liu and E.T. Cheng. 2004. Performance of a simple closed aquatic ecosystem (CAES) in space. *Advances in Space Research* 34: 1455-1460.

Ward, Lena K. and R. D. Jennings. 1990. Succession of disturbed and undisturbed chalk grassland at Aston Rowant National Nature Reserve: Details of changes in species. *The Journal of Applied Ecology* 27: 913-923.

Weinstein, M.P., J.M. Teal, J.H. Balletto and K.A. Strait. 2001. Restoration principles emerging from one of the world's largest tidal marsh restoration projects. *Wetlands Ecology and Management* 9: 387-407.

Whitham, T.G., J.K. Bailey, J.A. Schweitzer, S.M. Shuster, R.K. Bangert, C.J. LeRoy, E.V. Lonsdorf, G.J. Allan, S.P. DiFazio, B.M. Potts, D.G. Fischer, C.A. Gehring, R.L. Lindroth, J.C. Marks, S.C. Hart, G.M. Wimp and S.C. Wooley. 2006. A framework for community and ecosystem genetics: from genes to ecosystems. *Nature Reviews Genetics* 7: 510-523.

Yan, J., Y. Zhang and X. Wu. 1993. Advances of ecological engineering in China. *Ecological Engineering* 2: 193-215.

Zhang, R., W. Ji and B. Lu. 1998. Emergence and development of agro-ecological engineering in China. *Ecological Engineering* 11: 17 -26.

Zhang, X.-X., B. Kosier and U.B. Priefer. 2001. Genetic diversity of indigenous *Rhizobium leguminosarum* bv. *viciae* isolates modulating two different host plants during soil restoration with alfalfa. *Molecular Ecology* 10: 2297-2305.

Zhou, J., C.G. Messersmith and J.D. Harrington. 2005. HIDES: A computer-based herbicide injury diagnostic expert system. *Weed Technology* 19: 486-491.

Appendix A
Source code of the virtual ecosystem model and simulation program. See Chapter 3, Section 3.3 for more information about the program.

```
!LATEST UPDATE BY TRL: November 8, 2006

!#########################################################
#########################################################
!####                  START OF MAIN PROGRAM                  ####
!#########################################################
#########################################################

PROGRAM ECOSYS


!#########################################################
#########################################################
!####                  START OF DECLARATION PHASE             ####
!#########################################################
#########################################################

!declare implicit rules
IMPLICIT NONE

!declare character arrays
CHARACTER (LEN=15):: FILENAME                !name of output file
CHARACTER (LEN=29):: OUTFILE                 !name of output file with folder
destination
CHARACTER (LEN=14):: TEMFILE                  !name of temperature forcing
function input file
CHARACTER (LEN=14):: RADFILE                 !name of radiation forcing function
input file
CHARACTER (LEN=14):: MODFILE                  !name of model input file
CHARACTER (LEN=14):: SIMFILE                  !name of simulation input file
CHARACTER (LEN=11):: EATFILE                 !name of who eats who output file
CHARACTER (LEN=11):: ASCFILE                 !name of ascendency output file
character (len=9):: rfile
CHARACTER (LEN=4):: SIM                       !simulation number (text form)
CHARACTER (LEN=5):: FOLDER                   !folder output file will be written to
CHARACTER (LEN=10):: TEXTLINE                !dummy input line variable

!declare integer parameters
INTEGER*4, PARAMETER:: MAXPOP=100000000      !maximum total population allowed
INTEGER*4, PARAMETER:: MAXSPEC=30            !maximum total numer of species
allowed
```

```
!declare 2-D integer arrays
INTEGER*4, DIMENSION (MAXSPEC,2):: IWHERE          !species location beginning
and end in population INDEX matrix

!declare 1-D integer arrays
INTEGER*4, DIMENSION (MAXPOP):: INDEX              !index matrix to individuals in
the WHAT and IWHAT matrices
INTEGER*4, DIMENSION (MAXSPEC):: IPOPS             !population sizes
INTEGER*4, DIMENSION (MAXPOP):: IWHAT              !integer part of the
population matrix: species number
INTEGER*4, DIMENSION (MAXSPEC):: IWHO              !flags for which species are
actually present
INTEGER*4, DIMENSION (MAXSPEC):: MINMAXAGE         !low end of maximum possible
age for each species - days
INTEGER*4, DIMENSION (MAXSPEC):: MAXMAXAGE         !absolute maximum possible
age for each species - days

!declare simple integer variables
INTEGER*4:: IARGC
INTEGER*4:: I, J, K, L, M                          !dummy variables
INTEGER*4:: ICYCLE                                 !daily cycle counter
INTEGER*4:: IDAY                                   !the day during the year (in
days - 1 <= DAY <= 365 )
INTEGER*4:: IECOCYCLE                              !overall interation cycle
counter
INTEGER*4:: IPOP1                                  !total # of presently living
individuals (within IPOP2; for these IWHAT(I) > 0)
INTEGER*4:: IPOP2                                  !length of the matrix
INTEGER*4:: IRAD                                   !flag for radiation presence
(0=nighttime-no radiation at all; 1=daytime)
INTEGER*4:: ISEED1, ISEED2, ISEED3                 !random number seeds for
the simulation
INTEGER*4:: ISEED4, ISEED5                         !random number seed for the
model - used in initialization only
INTEGER*4:: ISTARTDAY                              !start day of the simulation
(in days - 1 <= STARTDAY <= 365 )
INTEGER*4:: ITEMP1, ITEMP2, ITEMP3, ITEMP4         !temporary variables
INTEGER*4:: ITEMP5, ITEMP6, ITEMP7                 !temporary variables (added
for use with FLOWS)
INTEGER*4:: IYEAR                                  !year of the simulation
INTEGER*4:: MAXDAYS                                !maximum total number of
days possible in the simulation, after the start day (days)
INTEGER*4:: MAXECOCYLES                            !maximum total number of
iteration cycles allowed for the simulation
```

```
INTEGER*4:: N1                                  !number of producer species
INTEGER*4:: N2                                  !number of consumer species
INTEGER*4:: NCYCLES                             !number of ecocycles per day
INTEGER*4:: NTOT                                !total number of species in
system

!declare 2-D floating-point arrays
REAL*4, DIMENSION (MAXSPEC,MAXSPEC):: FEEDPROB  !species-species feed
probability matrix
REAL*4, DIMENSION (MAXSPEC,MAXSPEC):: FOOD      !species-species food matrix
REAL*4, DIMENSION (MAXSPEC,MAXSPEC):: XINTER    !species-species interaction
matrix - relates to species health - how species i is affected by j
REAL*4, DIMENSION (MAXPOP,3):: WHAT             !real part of the population
matrix: 1=age, 2=energy content, 3=maximum age for individual

!declare 1-D floating-point arrays
REAL*4, DIMENSION (MAXSPEC):: AFFECT1           !food affectedness of
consumer species
REAL*4, DIMENSION (MAXSPEC):: AFFECT2           !health affectedness of
species - used together with the INTER matrix
REAL*4, DIMENSION (MAXSPEC):: ENER              !total energy in a species
(energy units)
REAL*4, DIMENSION (MAXSPEC):: ENERALLO          !energy allocated to species
during DELTIME (energy units)
REAL*4, DIMENSION (MAXSPEC):: ENERBIR           !energy level at birth for
each species (energy units)
REAL*4, DIMENSION (MAXSPEC):: ENERMIN           !minimum allowable energy
level for each species (energy units)
REAL*4, DIMENSION (MAXSPEC):: ENERQUAN          !size of energy "quantum" for
each species (energy units)
REAL*4, DIMENSION (MAXSPEC):: ENERREL           !relative energy level of a
species in the system (no units)
REAL*4, DIMENSION (MAXSPEC):: ENERREP           !energy threshold at which
species can reproduce (energy units)
REAL*4, DIMENSION (MAXSPEC):: FEEDNOTPROB       !probability of members of a
consumer species not feeding at all
REAL*4, DIMENSION (MAXSPEC):: HEALTH            !species health status
REAL*4, DIMENSION (MAXSPEC):: XMINMAXAGE        !low end of maximum possible
age for species (seconds)
REAL*4, DIMENSION (MAXSPEC):: XMAXMAXAGE        !absolute maximum possible
age for species (seconds)
REAL*4, DIMENSION (MAXSPEC):: XMETAB            !specific base metabolic rate
for species (energy_units/total_energy_units.time)
```

```
!declare simple floating-point variables
REAL*4:: ALPHA                                  !variable used in the
calculation of the attenuation factor for energy input (no units)
REAL*4:: DELTIME                                !the time increment used for
the simulation (s) (should be integer fraction of 86400)
REAL*4:: DBLETIME                               !minimum doubling time for
the system (year)
REAL*4:: DBLETIMES                              !minimum doubling time for
the system (seconds)
REAL*4:: ENERIN                                 !amount of energy coming into
the system during time increment
REAL*4:: ENERMAX                                !maximum total energy for
system (energy units)
REAL*4:: ENERTOT                                !total energy present in the
system (all species, energy units)
REAL*4:: ENERTOTP                               !total energy present in
producer species (energy units)
REAL*4:: POWRMAX                                !maximum possible power into
the system (energy units/second)
REAL*4:: RELRATE                                !relative rate for inherent
metabolism (dependent on temperature)
REAL*4:: STARTTIME                              !start time during the day of
the simulation (in seconds, <86400, should be integer multiple of DELTIME)
REAL*4:: SUM1, SUM2                             !temporary variables
REAL*4:: TEMP1, TEMP2, TEMP3, TEMP4             !temporary variables
REAL*4:: TEMPERAT                               !temperature during the time
increment (deg. C)
REAL*4:: TIME                                   !the time during the day at
the end of the present time increment (s)
REAL*4:: TTIME                                  !total time since the
beginning of the simulation (s)
REAL*4:: TOTALTIME                              !time since the very beginning
of the year in which the simulation started (s)
REAL*4:: RTIME, S, E, C                         !run time, start, end, and call
variables for timing the program

!declare real functions
REAL*4:: RANDOM1, RANDOM2, RANDOM3              !random number generator
functions for simulation
REAL*4:: RANDOM4, RANDOM5                       !random number generator
functions for model - used only during initialization

!declare 1-D integer, allocatable arrays
INTEGER*4, DIMENSION (:), ALLOCATABLE:: AGED    !deaths due to old age by
species
```

```
INTEGER*4, DIMENSION (:), ALLOCATABLE:: STARVED    !deaths due to starvation by
species
INTEGER*4, DIMENSION (:), ALLOCATABLE:: BIRTHS     !births by species

!declare 2-D integer, allocatable arrays
INTEGER*4, DIMENSION (:,:), ALLOCATABLE:: EAT      !numbers of who eats who

!declare 1-D real, allocatable arrays
REAL*4, DIMENSION (:), ALLOCATABLE:: SPMETAB       !energy metabolized by all
members of a species

!declare 2-D real array ascendency calc
REAL*4, DIMENSION (MAXSPEC+2,MAXSPEC+2):: FLOWS !energy flow matrix for the
ecosystem network

!declare flouting point-variable for ascendancy calculation
REAL*4:: A                                         !ascendency of ecosystem at
a given time step


!temporary variable for verif
REAL*4:: NRG

!////////////////////////////////////////////////////////////////////////////
!#####                    END OF DECLARATION PHASE                    #####
!////////////////////////////////////////////////////////////////////////////

CALL CPU_TIME(C)
S=C

!M=IARGC()
!CALL GETARG (1,FOLDER)
!CALL GETARG (M,SIM)

print *, 'sim?'

read (*,707) SIM
707 format (a4)

write (SIMFILE,700) SIM
700 format ('ecosim',a4,'.inp')

write (MODFILE,701) SIM
701 format ('ecomod',a4,'.inp')

write (TEMFILE,702) SIM
```

```
702 format ('ecotem',a4,'.inp')

write (RADFILE,703) SIM
703 format ('ecorad',a4,'.inp')

WRITE (EATFILE,704) FOLDER,SIM
!704 FORMAT ('/export/',A5,'/eat',A4,'.dat')
704 format ('eat',a4,'.dat')

WRITE (ASCFILE,706) FOLDER,SIM
!706 FORMAT ('/export/',A5,'/asc',A4,'.dat')
706 format ('asc',a4,'.dat')

!#########################################################
#########################################################
!####                START OF INITIALIZATION PHASE                ####
!#########################################################
#########################################################

!#########################################################
#########################################################
!####                INTIALIZATION PHASE PART 1                   ####
!####          DATA INPUT AND OUTPUT INTO "PAPER" FILE            ####
!#########################################################
#########################################################

!####                INTIALIZATION PHASE PART 1.1                 ####
!####             SIMULATION DATA INPUT AND OUTPUT                ####

!open input file to read simulation information and parameter values
OPEN (UNIT=1, FILE=SIMFILE)
READ (1,1001) TEXTLINE
1001 FORMAT (10A1)

!read the output file name
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,4001) FILENAME
4001 FORMAT (A15)

WRITE (OUTFILE,705) FOLDER,FILENAME
705 FORMAT ('/export/',A5,'/',A15)

!open the output "paper" file: FILENAME if in same folder OUTFILE if in different folder
OPEN (UNIT=2, FILE=OUTFILE)
```

```
!open (unit=9,file='weather_sub.out')
!open (unit=10,file='weather.out')
!open (unit=11,file=rfile)
!open (unit=12,file='theo.out')
!open (unit=13,file='dris.out')
!open (unit=14,file='dors.out')
!OPEN (UNIT=15,FILE='atten.out')
!OPEN (UNIT=16,FILE='theodors2.out')

OPEN (UNIT=3, FILE=EATFILE)

!write the headings etc. into the output file

WRITE(2,2000)
2000 FORMAT('******************************** START OF OUTPUT FROM
INITIALIZATION PHASE ********************************',/)

WRITE(2,3007) FILENAME
3007 FORMAT('THE OUTPUT FILE NAME FOR THIS EXPERIMENT IS (THIS FILE):
',A25,/)

!fill all the major matrices to make sure memory is available etc. - part 1 - arrays of length
MAXPOP
DO I=1,MAXPOP
  INDEX(I)=0
  IWHAT(I)=0
  DO J=1,3
    WHAT(I,J)=0.0
  END DO
END DO

!fill all the major matrices to make sure memory is available etc. - part 1 - arrays of length
MAXSPEC
DO I=1,MAXSPEC
  IPOPS(I)=0
  IWHO(I)=0
  MINMAXAGE(I)=0
  MAXMAXAGE(I)=0
  AFFECT1(I)=0.0
  AFFECT2(I)=0.0
  ENER(I)=0.0
  ENERALLO(I)=0.0
  ENERBIR(I)=0.0
  ENERMIN(I)=0.0
  ENERQUAN(I)=0.0
  ENERREP(I)=0.0
```

```
  FEEDNOTPROB(I)=0.0
  HEALTH(I)=0.0
  XMINMAXAGE(I)=0.0
  XMAXMAXAGE(I)=0.0
  XMETAB(I)=0.0
  DO J=1,MAXSPEC
    FEEDPROB(I,J)=0.0
    FOOD(I,J)=0.0
    XINTER(I,J)=0.0
  END DO
END DO

WRITE(2,3008)
3008 FORMAT('ALL MATRICES SUCCESSFULLY FILLED AT INITIALIZATION',/)

WRITE(2,2001)
2001 FORMAT('********** OUTPUT FROM INITIALIZATION PHASE PART 1 - DATA
INPUT AND OUTPUT INTO "PAPER" FILE **********')

WRITE(2,2002)
2002 FORMAT(/,'PARAMETER VALUES FOR THE SIMULATION:')

!parameter values from the program file itself
WRITE(2,3001) MAXPOP
3001 FORMAT('MAXIMUM POPULATION SIZE ALLOWED FOR THIS SIMULATION
RUN: ',I10)
WRITE(2,3002) MAXSPEC
3002 FORMAT('MAXIMUM NUMBER OF SPECIES ALLOWED FOR THIS SIMULATION
RUN: ',I5)

!read random number seed from simulation parameter file
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1002) ISEED1, ISEED2, ISEED3
1002 FORMAT (3I10)
WRITE(2,2003) ISEED1, ISEED2, ISEED3
2003 FORMAT('SEEDS FOR RANDOM NUMBER GENERATOR FOR THE SIMULATION: ',
3I10)

!read start day for the simulation
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1003) ISTARTDAY
1003 FORMAT (I10)
WRITE(2,2004) ISTARTDAY
2004 FORMAT('START DAY FOR THE SIMULATION: ', I5)
```

```
!read start time for the simulation
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1004) STARTTIME
1004 FORMAT (F10.1)
WRITE(2,2005) STARTTIME
2005 FORMAT('START TIME FOR THE SIMULATION: ', F10.1, ' SECONDS')

!read maximum days allowed for simulation
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1005) MAXDAYS
1005 FORMAT (I10)
WRITE(2,2006) MAXDAYS
2006 FORMAT('MAXIMUM NO. DAYS FOR THE SIMULATION: ', I5, ' DAYS')

!read time increment
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1006) DELTIME
1006 FORMAT (F10.1)
WRITE(2,2007) DELTIME
2007 FORMAT('TIME INCREMENT FOR THE SIMULATION: ', F10.1, ' SECONDS')

!read upper bound on total system energy
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1007) ENERMAX
1007 FORMAT (E10.2)
WRITE(2,2008) ENERMAX
2008 FORMAT('UPPER BOUND ON TOTAL SYSTEM ENERGY: ', E10.2, ' ENERGY
UNITS')

!read minimum doubling time for the system
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1008) DBLETIME
1008 FORMAT (F10.2)
WRITE(2,2009) DBLETIME
2009 FORMAT('MINIMUM DOUBLING TIME FOR THE SYSTEM: ', F10.2, ' YEAR')

!read the variable "alpha" for the attenuation factor calculation
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1009) ALPHA
```

```
1009 FORMAT (F10.1)
WRITE(2,2010) ALPHA
2010 FORMAT('ALPHA VALUE FOR ATTENUATION FACTOR CALCULATION: ', F10.1)

!close the input file
CLOSE (UNIT=1)

!////              END OF INTIALIZATION PHASE PART 1.1 - SIMULATION DATA INPUT
AND OUTPUT          ////

!#####                INTIALIZATION PHASE PART 1.2 - MODEL DATA INPUT AND
OUTPUT            #####
!open input file to read model parameter values
OPEN (UNIT=1, FILE=MODFILE)
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
WRITE(2,2011)
2011 FORMAT(/, 'PARAMETER VALUES FOR THE MODEL:')

!Part 1 of the model parameter values - random number seed

!read random number seed from model parameter file
READ (1,1001) TEXTLINE
READ (1,1010) ISEED4, ISEED5
1010 FORMAT (2I10)
WRITE(2,2012) ISEED4, ISEED5
2012 FORMAT('SEEDS FOR RANDOM NUMBER GENERATOR FOR THE MODEL: ', 2I10)

!Part 2 of the model parameter values - ecosystem composition

READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE

!read number of producer species (N1) and number of consumer species (N2)
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1011) N1, N2
1011 FORMAT (2I10)
WRITE(2,2013) N1
2013 FORMAT('NUMBER OF PRODUCER SPECIES: ',I5)
WRITE(2,2014) N2
2014 FORMAT('NUMBER OF CONSUMER SPECIES: ',I5)

!calculate total number of species in ecosystem and validate: not too large
NTOT=N1+N2
IF (NTOT > MAXSPEC) THEN
```

```
 PRINT *
 PRINT *
 PRINT *, "Total number of species larger than allowed maximum"
 PRINT *
 PRINT *
 STOP
ENDIF
WRITE(2,2015) NTOT
2015 FORMAT('TOTAL NUMBER OF SPECIES IN SYSTEM: ', I5, '   ***** NOTE:
CALCULATED')

!read the minimum energy levels to continue to exist - energy units
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1012) (ENERMIN(I),I=1,NTOT)
1012 FORMAT (10F10.1)
WRITE(2,2016)
2016 FORMAT('THE ENERGY MINIMA FOR THE SPECIES ARE:')
WRITE(2,2017) (ENERMIN(I),I=1,NTOT)
2017 FORMAT (10F10.1)

!read energy levels at birth - energy units
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1013) (ENERBIR(I),I=1,NTOT)
1013 FORMAT(10F10.1)
WRITE(2,2018)
2018 FORMAT('THE BIRTH ENERGIES FOR THE SPECIES ARE:')
WRITE(2,2019) (ENERBIR(I),I=1,NTOT)
2019 FORMAT (10F10.1)

!read energy threshold for reproduction - energy units
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1014) (ENERREP(I),I=1,NTOT)
1014 FORMAT(10F10.1)
WRITE(2,2020)
2020 FORMAT('THE REPRODUCTION THRESHOLD ENERGIES FOR THE SPECIES
ARE:')
WRITE(2,2021) (ENERREP(I),I=1,NTOT)
2021 FORMAT(10F10.1)

!read values of the energy quanta for the producers - energy units
```

```
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1015) (ENERQUAN(I),I=1,N1)
1015 FORMAT(10F10.2)
WRITE(2,2022)
2022 FORMAT('THE ENERGY QUANTUM MAGNITUDES FOR THE PRODUCER SPECIES
ARE:')
WRITE(2,2023)(ENERQUAN(I),I=1,N1)
2023 FORMAT(10F10.2)

!read specific base metabolic rate of each species - energy unit/(total energy units x
second)
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1016) (XMETAB(I),I=1,NTOT)
1016 FORMAT(10E10.3)
WRITE(2,2024)
2024 FORMAT('THE SPECIFIC METABOLIC RATES FOR THE SPECIES ARE:')
WRITE(2,2025) (XMETAB(I),I=1,NTOT)
2025 FORMAT(10E10.3)

!read low end of maximum age of each species in days
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1017) (MINMAXAGE(I),I=1,NTOT)
1017 FORMAT(10I10)
WRITE(2,2026)
2026 FORMAT('THE LOW END OF MAXIMUM AGES FOR THE SPECIES (IN DAYS)
ARE:')
WRITE(2,2027) (MINMAXAGE(I),I=1,NTOT)
2027 FORMAT(10I10)

!read absoute maximum age of each species in days
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,6001) (MAXMAXAGE(I),I=1,NTOT)
6001 FORMAT(10I10)
WRITE(2,6002)
6002 FORMAT('THE ABSOLUTE MAXIMUM AGES FOR THE SPECIES (IN DAYS) ARE:')
WRITE(2, 6003) (MAXMAXAGE(I),I=1,NTOT)
6003 FORMAT(10I10)
```

```
!calculate low end maximum age of species in seconds
DO I=1,NTOT
  XMINMAXAGE(I)=FLOAT(MINMAXAGE(I))*86400.0
END DO
WRITE(2,2028)
2028 FORMAT('THE LOW END OF MAXIMUM AGES FOR THE SPECIES (IN SECONDS)
ARE:   ****** NOTE: CALCULATED')
WRITE(2,2029) (XMINMAXAGE(I),I=1,NTOT)
2029 FORMAT(15E15.5)

!calculate absolute maximum age of species in seconds
DO I=1,NTOT
  XMAXMAXAGE(I)=FLOAT(MAXMAXAGE(I))*86400.0
END DO
WRITE(2,6004)
6004 FORMAT('THE ABSOLUTE MAXIMUM AGES FOR THE SPECIES (IN SECONDS)
ARE:   ****** NOTE: CALCULATED')
WRITE(2,6005) (XMAXMAXAGE(I),I=1,NTOT)
6005 FORMAT(15E15.5)

!read food affectedness of consumer species
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1018) (AFFECT1(N1+I),I=1,N2)
1018 FORMAT (10F10.3)
WRITE(2,2030)
2030 FORMAT('THE FOOD AFFECTEDNESS VALUES FOR THE CONSUMER SPECIES
ARE:')
WRITE (2,2031) (AFFECT1(N1+I),I=1,N2)
2031 FORMAT (10F10.3)

!read health affectedness of all species - used together with the INTER matrix
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1019) (AFFECT2(I),I=1,NTOT)
1019 FORMAT(10F10.1)
WRITE(2,2032)
2032 FORMAT('THE HEALTH AFFECTEDNESS VALUES FOR THE SPECIES ARE:')
WRITE (2,2033) (AFFECT2(I),I=1,NTOT)
2033 FORMAT(10F10.1)

!Part 3 of the model parameter values - ecosystem structure

READ (1,1001) TEXTLINE
```

```
READ (1,1001) TEXTLINE

!read food matrix -  preference values of each consumer for other species - all values
between 0.0 and +1.0
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
DO I=N1+1,NTOT
  READ (1,1001) TEXTLINE
  READ (1,1020) (FOOD(I,J),J=1,NTOT)
END DO
1020 FORMAT(10F8.3)
WRITE(2,2034)
2034 FORMAT('THE FOOD PREFERENCE VALUES FOR THE CONSUMERS ARE:')
DO I=N1+1,NTOT
  WRITE (2,2035) (FOOD(I,J),J=1,NTOT)
END DO
2035 FORMAT(10F10.3)

!read interaction matrix - values between -1.0 and +1.0 - how species I is affected by
species J
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
DO I=1,NTOT
  READ (1,1001) TEXTLINE
  READ (1,1021) (XINTER(I,J),J=1,NTOT)
END DO
1021 FORMAT(10F8.3)
WRITE(2,2036)
2036 FORMAT('HOW SPECIES I (ROW) IS AFFECTED BY SPECIES J (COLUMN) - THE
INTERACTION VALUES ARE:')
DO I=1,NTOT
  WRITE (2,2037) (XINTER(I,J),J=1,NTOT)
END DO
2037 FORMAT(10F10.3)

!Part 4 of the model parameter values - initial state of system

READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE

!read the initial population sizes
```

```
READ (1,1001) TEXTLINE
READ (1,1001) TEXTLINE
READ (1,1022) (IPOPS(I),I=1,NTOT)
1022 FORMAT (10I10)
WRITE(2,2038)
2038 FORMAT('THE INITIAL POPULATION SIZES ARE:')
WRITE(2,2039) (IPOPS(I),I=1,NTOT)
2039 FORMAT (10I10)

!close the input file
CLOSE (UNIT=1)

!////              END OF INTIALIZATION PHASE PART 1.2 - MODEL DATA INPUT AND
OUTPUT            ////

WRITE(2,2042)
2042 FORMAT(/,'********************** END OF OUTPUT FROM INITIALIZATION
PHASE PART 1 ****************************')

!allocate death, birth, metabolization, and eat vectors/matrix
ALLOCATE (AGED(1:NTOT))
ALLOCATE (STARVED(1:NTOT))
ALLOCATE (BIRTHS(1:NTOT))
ALLOCATE (SPMETAB(1:NTOT))
ALLOCATE (EAT(1:N2,1:NTOT))

!fill death, birth, and metabolization vectors
DO I=1,NTOT
  AGED(I)=0
  STARVED(I)=0
  BIRTHS(I)=0
  SPMETAB(I)=0
END DO

!fill eat matrix
DO I=1,N2
  DO J=1,NTOT
    EAT(I,J)=0
  END DO
END DO

!fill flow matrix
ITEMP5=NTOT+2
DO I=1,ITEMP5
  DO J=1,ITEMP5
    FLOWS(I,J)=0.0
```

```
  END DO
END DO


!////////////////////////////////////////////////////////////////////////////
!####                  END OF INTIALIZATION PHASE PART 1                 ####
!####              DATA INPUT AND OUTPUT INTO "PAPER" FILE               ####
!////////////////////////////////////////////////////////////////////////////


!#########################################################
#########################################################
!####                    INTIALIZATION PHASE PART 2                    ####
!####   RANDOM INITIALIZATION PART OF THE SYSTEM COMPOSITION   ####
!#########################################################
#########################################################

WRITE(2,2043)
2043 FORMAT(//, '**** OUTPUT FROM INITIALIZATION PHASE PART 2 - RANDOM
INITIALIZATION PART OF THE SYSTEM COMPOSITION ****',/)

!initialize random number generator for random initialization of system composition - use
ISEED4 & ISEED5 from file ECOMOD.INP
TEMP1=RANDOM4(ISEED4)
TEMP2=RANDOM5(ISEED5)
WRITE(2,2044) TEMP1, TEMP2
2044 FORMAT('VALUES OBTAINED FROM INITIALIZATION CALLS TO RANDOM
NUMBER GENERATORS FOR MODEL (ISEED4,ISEED5) ARE: ', 2F10.6)

!check the population sizes, set the population presence Y/N indicators, and calculate the
total initial population size
IPOP1=0
DO I=1,NTOT
 IF (IPOPS(I) < 0) THEN
   PRINT *
   PRINT *
   PRINT *, "A species initial population was negative, program was stopped"
   PRINT *
   PRINT *
   WRITE(2,2045)
   CLOSE (UNIT=2)
   STOP
 END IF
 IF (IPOPS(I)==0) THEN
  IWHO(I)=0
 ELSE
```

```
   IWHO(I)=1                                    !switch on population presence indicator
   IPOP1=IPOP1+IPOPS(I)
   IF (IPOP1 > MAXPOP) THEN
    PRINT *
    PRINT *
    PRINT *, "Total initial population larger than allowed maximum, program was stopped"
    PRINT *
    PRINT *
    WRITE(2,2046)
    CLOSE (UNIT=2)
    STOP
   ENDIF
  ENDIF
END DO
2045 FORMAT(//, 'AN INITIAL POPULATION WAS NEGATIVE, PROGRAM WAS
STOPPED')
2046 FORMAT(//, 'TOTAL INITIAL POPULATION LARGER THAN ALLOWED
MAXIMUM, PROGRAM WAS STOPPED')
WRITE(2,2047) IPOP1
2047 FORMAT('THE TOTAL INITIAL POPULATION SIZE IS (IPOP1): ', I7)

!fill the location (IWHERE) and index (INDEX) matrices and initialize the age and energy
level of individuals for each species
K=0                                          ! dummy counter variable
DO I=1,NTOT
 IF (IWHO(I)==1) THEN
   ITEMP1=IPOPS(I)
   IWHERE (I,1)=K+1
   IWHERE (I,2)=K+ITEMP1
   DO J=1,ITEMP1
    K=K+1
    INDEX(K)=K                               !set the index value to the individual's
location in WHAT and IWHAT
    IWHAT(K)=I                               !set the species number in the population
matrix
    TEMP1=RANDOM4(1)                         !get random number, uniformly
distributed
     WHAT(K,3)=XMINMAXAGE(I)+TEMP1*(XMAXMAXAGE(I)-XMINMAXAGE(I))!set
maximum age of individual (distributed between XMINMAXAGE and XMAXMAXAGE)
    TEMP1=RANDOM5(1)                                    !get another random
number, uniformly distributed
     WHAT(K,1)=TEMP1*WHAT(K,3)                  !set initial age for individual
(uniformly distributed)
     WHAT(K,2)=ENERBIR(I)+TEMP1*(ENERREP(I)-ENERBIR(I))   !set initial energy for
individual at creation
   END DO
```

```
  ELSE
    IWHERE(I,1)=0
    IWHERE(I,2)=0
  END IF
END DO

!write details of the populations at initialization time
WRITE(2,2048)
2048 FORMAT(/,'---------- STATS OF THE POPULATIONS AT
INITIALIZATION TIME ----------')
WRITE(2,2049)
2049 FORMAT('SPECIES PRESENT(0/1) POPULATION    START        STOP')
DO I=1,NTOT
  WRITE(2,2050) I, IWHO(I), IPOPS(I), IWHERE(I,1), IWHERE(I,2)
END DO
2050 FORMAT(I5,I9,I13,I13,I13)

!write details of the individuals at initialization time
!WRITE(2,2051)
!2051 FORMAT(/,'---------- STATS OF THE INDIVIDUALS AT
INITIALIZATION TIME ----------')
!DO I=1,NTOT
!   WRITE(2,2052) I, IWHO(I), IPOPS(I), IWHERE(I,1), IWHERE(I,2)
!   WRITE(2,2053)
!   IF (IWHO(I)==1) THEN                          !check if the species exists
!     DO J=IWHERE(I,1),IWHERE(I,2)                    !cycle through the individuals of the
species
!       ITEMP1=INDEX(J)                           !find the individual's location in the
population matrix
!       WRITE(2,2054) I, J, ITEMP1, IWHAT(ITEMP1), WHAT(ITEMP1,1),
WHAT(ITEMP1,2), WHAT(ITEMP1,3)
!     END DO
!   END IF
!END DO
!2052 FORMAT('SPECIES, PRESENT, POPULATION, START, END: ',2I3,3I7)
!2053 FORMAT('SPECIES  COUNTER    INDEX  SPECIES#       AGE       ENERGY
MAXAGE')
!2054 FORMAT(I5,I11,I9,I8,E17.5,F10.1,E17.5)

WRITE(2,2055)
2055 FORMAT(/,'********** END OF OUTPUT FROM INITIALIZATION
PHASE PART 2 **********')
```

```
!//////////////////////////////////////////////////////////////////////////
!####                  END OF INTIALIZATION PHASE PART 2                 ####
!####  RANDOM INITIALIZATION PART OF THE SYSTEM COMPOSITION   ####
!//////////////////////////////////////////////////////////////////////////

!#########################################################
#########################################################
!####                  INTIALIZATION PHASE PART 3                 ####
!####               SETTING UP, SETTING COUNTERS, ETC.            ####
!#########################################################
#########################################################

WRITE(2,2056)
2056 FORMAT(//, '****** OUTPUT FROM INITIALIZATION PHASE PART 3 -
SETTING UP, SETTING COUNTERS, ETC. ******',/)

!set the initial length of the contents of the WHAT and IWHAT matrices (IPOP2) to the
total living population
IPOP2=IPOP1
WRITE(2,2057) IPOP2
2057 FORMAT('THE LENGTH OF THE WHAT MATRIX CONTENTS (IPOP2) WAS SET
AT (IPOP1 = TOTAL LIVING POPULATION): ', I7)

!calculate the number of ecocycles per day (NCYCLES) and then re-calculate DELTIME
NCYCLES=INT(86400.0/DELTIME+0.001)
WRITE(2,2058) NCYCLES
2058 FORMAT('THE NUMBER OF ECOCYCLES PER DAY IS: ',I5)
DELTIME=86400.0/FLOAT(NCYCLES)
WRITE(2,2059) DELTIME
2059 FORMAT('THE NEW VALUE OF DELTIME WAS SET AT (IN SECONDS): ', F10.2)

!calculate the maximum total number of cycles for the simulation
MAXECOCYLES=INT(FLOAT(MAXDAYS)/DELTIME*86400)
WRITE(2,2060) MAXECOCYLES
2060 FORMAT('THE MAXUMUM POSSIBLE NUMBER OF ECOCYCLES IS: ', I7)

!calculate the initial values of the daily ecocycle counter, starting time, and starting day
ICYCLE=INT(STARTTIME/DELTIME+0.001)
!correcting for "illegal" start time specification during first delta-time of the day - roll
back one
IF (ICYCLE==0) THEN
  ISTARTDAY=ISTARTDAY-1
  ICYCLE=NCYCLES
  WRITE(2,2061)
END IF
```

```
2061 FORMAT('NOTE: START DATA CORRECTED FOR "ILLEGAL" START TIME
SPECIFICATION DURING FIRST DELTA-TIME OF THE DAY')
STARTTIME=FLOAT(ICYCLE)*DELTIME
WRITE(2,2062) ISTARTDAY
2062 FORMAT('THE STARTING DAY WAS SET AT (STARTDAY, BEFORE FIRST
ITERATION INCREMENTATION): ',I5)
WRITE(2,2063) STARTTIME
2063 FORMAT('THE STARTING TIME WAS SET AT (STARTTIME, BEFORE FIRST
ITERATION INCREMENTATION, IN SECONDS): ',F10.0)
WRITE(2,2064) ICYCLE
2064 FORMAT('THE STARTING DAY CYCLE NUMBER IS (ICYCLE, BEFORE FIRST
ITERATION INCREMENTATION): ',I5)

!initialize iteration counters and time variables
IECOCYCLE=0                              !is the current ecocycle number
IDAY=ISTARTDAY                           !is the current day number
IYEAR=1                                  !is the current year number
WRITE(2,2065) IECOCYCLE
2065 FORMAT('THE STARTING ECOCYCLE NUMBER IS (IECOCYCLE, BEFORE FIRST
ITERATION INCREMENTATION): ',I5)
WRITE(2,2066) IDAY
2066 FORMAT('THE STARTING DAY NUMBER IS (IDAY, BEFORE FIRST ITERATION
INCREMENTATION): ',I5)
WRITE(2,2067) IYEAR
2067 FORMAT('THE STARTING YEAR NUMBER IS (IYEAR, BEFORE FIRST ITERATION
INCREMENTATION): ',I5)

!set the time variables to their initial values
TIME=STARTTIME                           !present time of the day (s)
TTIME=0.0                                !set initial value of total time since start of
simulation
TOTALTIME=FLOAT(IDAY-1)*86400.0+TIME     !calculate value of time since
very beginning of year of simulation start
WRITE(2,2068) TIME
2068 FORMAT('INITIAL TIME-OF-DAY VALUE AT END OF TIME INCREMENT (TIME,
BEFORE FIRST INCREMENTATION): ', F10.0)
WRITE(2,2069) TTIME
2069 FORMAT('TOTAL SIMULATION TIME ELAPSED (TTIME, BEFORE FIRST
INCREMENTATION): ', F10.0)
WRITE(2,2070) TOTALTIME
2070 FORMAT('TOTAL TIME ELAPSED SINCE START OF FIRST YEAR OF
SIMULATION (TOTALTIME, BEFORE FIRST INCREMENTATION): ', F10.0)

!calculate the total initial energy content of the system by going straight through the
WHAT matrix
ENERTOT=0.0
```

```
DO I=1,IPOP1
  ENERTOT=ENERTOT+WHAT(I,2)
END DO
WRITE(2,2071) ENERTOT
2071 FORMAT('TOTAL INITIAL ENERGY IN THE SYSTEM FROM SUMMING VALUES
IN THE "WHAT" MATRIX: ', E15.5)

!calculate the total energy content of each species at very start of simulation
DO I=1,NTOT
   ENER(I)=0.0                                !set initial value for species total energy
   IF (IWHO(I)==1) THEN                       !check if the species exists
     DO J=IWHERE(I,1),IWHERE(I,2)             !cycle through the individuals of the
species
       ITEMP1=INDEX(J)                        !find the individual's location in the
population matrix
       ENER(I)=ENER(I)+WHAT(ITEMP1,2)         !sum the energy for the species
     END DO
   END IF
END DO
WRITE(2,2072)
2072 FORMAT('ENERGY CONTENTS OF THE SPECIES AT VERY START OF
SIMULATION:',/,'---------------------------------------------')
WRITE(2,2073) (I,ENER(I),I=1,NTOT)
2073 FORMAT(I5,E15.5)

!calculate the total energy content of the system by summing the species' energies
TEMP1=0.0
DO I=1,NTOT
  TEMP1=TEMP1+ENER(I)
END DO
WRITE(2,2074) TEMP1
2074 FORMAT('TOTAL ENERGY IN SYSTEM FROM SUMMING SPECIES ENERGIES: ',
E15.5)
WRITE(2,2075) ENERTOT
2075 FORMAT('THE VALUE OBTAINED FROM SUMMING INDIVIDUAL ENERGIES
WAS: ', E15.5)

!calculate the minimum double time in seconds
DBLETIMES=DBLETIME*3.1536E7
WRITE(2,2076) DBLETIMES
2076 FORMAT('MINIMUM DOUBLING TIME FOR THE SYSTEM: ', E12.5, ' SECONDS')

!calculate maximum possible power into the system
POWRMAX=ENERMAX/DBLETIMES
WRITE(2,2077) POWRMAX
```

```
2077 FORMAT('MAXIMUM POSSIBLE POWER INTO THE SYSTEM: ', E12.5, ' ENERGY
UNITS/SECOND')

!initialize random number generators for the simulation - use ISEED1, ISEED2, & ISEED3
from file ECOSIM.DAT
TEMP1=RANDOM1(ISEED1)
TEMP2=RANDOM2(ISEED2)
TEMP3=RANDOM3(ISEED3)
WRITE(2,2078) TEMP1, TEMP2, TEMP3
2078 FORMAT('VALUES OBTAINED FROM INIT CALLS TO RANDOM NUMBER
GENERATORS FOR SIMULATION (ISEED1,ISEED2,ISEED3) ARE: ',3F10.6)

!calculate total energy in producer species - for weather generator
!energy in each species calculated above
ENERTOTP=0.0                                    !set initial value of total energy for
producer species
DO I=1,N1
  IF (IWHO(I)==1) THEN
    ENERTOTP=ENERTOTP+ENER(I)                   !increment total energy of all
producers
  END IF
END DO

!print energy content of producers to output file
WRITE(2,5001) ENERTOTP
5001 FORMAT ('TOTAL ENERGY IN PRODUCER SPECIES:',E15.5)

!initialize the weather generator (1=initialize, 2=routine use)
CALL
WEATHER(1,IYEAR,IDAY,TIME,DELTIME,ENERMAX,POWRMAX,ALPHA,ENERTOTP,TEM
PERAT,IRAD,ENERIN,TEMFILE,RADFILE)
!write (10,10001) TEMPERAT, ENERIN
!10001 format (f10.5,E15.5)


WRITE(2,2079)
2079 FORMAT(/,'********************** END OF OUTPUT FROM INITIALIZATION
PHASE PART 3 ***************************')

!////////////////////////////////////////////////////////////////////////////
!#####          END OF INTIALIZATION PHASE PART 3 - SETTING UP          #####
!////////////////////////////////////////////////////////////////////////////

WRITE(2,2080)
2080 FORMAT(/,'************************** END OF OUTPUT FROM
INITIALIZATION PHASE *********************************')
```

```
!////////////////////////////////////////////////////////////////////////////
!####                    END OF INITIALIZATION PHASE                    ####
!////////////////////////////////////////////////////////////////////////////

!#########################################################
#########################################################
!####                    START OF ITERATION PHASE                    ####
!#########################################################
#########################################################

WRITE(2,2081)
2081 FORMAT(////,'******************************** START OF OUTPUT FROM
ITERATION PHASE ********************************',/)

OPEN (UNIT=4, FILE=ASCFILE)
!OPEN (UNIT=5, FILE='flows.out')

!increment the ecocycle counter
9000 IECOCYCLE=IECOCYCLE+1

!check if maximum number of iteration cycles has been exceeded
IF (IECOCYCLE > MAXECOCYLES) THEN
  GOTO 9001
END IF

!increment the daily ecocycle counter, check, and calculate the time at the end of the
increment
!NOTE: all things are calculated at the end of the time increment, except weather which is
done at the centre
ICYCLE=ICYCLE+1
TIME=TIME+DELTIME
IF (ICYCLE>NCYCLES) THEN
  ICYCLE=1
  TIME=DELTIME
  IDAY=IDAY+1
  IF (IDAY>365) THEN
    IDAY=1
    IYEAR=IYEAR+1
  END IF
END IF

!calculate the total time (at end of time increment) since the start of the simulation
TTIME=FLOAT(IECOCYCLE)*DELTIME
!calculate time since start of year of simulation
TOTALTIME=FLOAT((IYEAR-1)*365+(IDAY-1))*86400.0+TIME
```

```
!output the ecocycle number and the time values onto screen
!PRINT *, IECOCYCLE, ICYCLE, IYEAR, IDAY, TIME, TTIME, TOTALTIME

!kill off everyone but producers after 1 ecocycle
!IF (IECOCYCLE==1) THEN
!   DO I=N1+1,NTOT                                  !for all consumer species
!     IF (IWHO(I)==1) THEN                          !if the species still exists
!       DO J=IWHERE(I,1),IWHERE(I,2)                        !for all individuals of the
species
!         ITEMP1=INDEX(J)
!         WHAT(ITEMP1,1)=WHAT(ITEMP1,3)           !age of individual equals max age for
individual
!       END DO
!     END IF
!   END DO
!END IF

!print the ecocycle etc. into the output file
WRITE(2,2082) IECOCYCLE
2082 FORMAT(/, '*************** START OF OUTPUT FROM ECOCYCLE',I8,'
*************')
WRITE(2,2083) IECOCYCLE, ICYCLE, IYEAR, IDAY
2083 FORMAT('IECOCYCLE, ICYCLE, IYEAR, IDAY: ',4I10)
WRITE(2,2084) TIME, TTIME, TOTALTIME
2084 FORMAT('AT END OF THIS ECOCYCLE: TIME, TTIME, TOTALTIME WILL BE:
',3F12.1)
WRITE(2,2085) ENERTOT
2085 FORMAT('TOTAL ENERGY AT START OF THIS ECOCYLE: ',E15.5)
WRITE(2,2086)
2086 FORMAT('SPECIES, POPULATION, AND SPECIES ENERGY AT START OF THIS
ECOCYCLE:')
WRITE(2,2087) (I,IPOPS(I),ENER(I),I=1,NTOT)
2087 FORMAT(I5,I10,E15.5)

!calculate energy content of producers
!energy in each species calculated at the end of previous ecocycle (or initialization for
ecocycle 1)
ENERTOTP=0.0                                    !set initial value of total energy for
producer species
DO I=1,N1
  IF (IWHO(I)==1) THEN
    ENERTOTP=ENERTOTP+ENER(I)                       !increment total energy of all
producers
  END IF
END DO
```

```
!print energy content of producers to output file
WRITE(2,5002) ENERTOTP
5002 FORMAT ('TOTAL ENERGY IN PRODUCER SPECIES AT START OF THIS
ECOCYCLE IS:',E15.5)

!call the weather generator (1=initialize, 2=routine use)
CALL
WEATHER(2,IYEAR,IDAY,TIME,DELTIME,ENERMAX,POWRMAX,ALPHA,ENERTOTP,TE
MPERAT,IRAD,ENERIN,TEMFILE,RADFILE)

!write (9,90001) TEMPERAT, ENERIN
!90001 FORMAT (F10.5,E15.5)
IF (ENERTOTP+ENERIN > ENERMAX) THEN
  ENERIN = ENERMAX - ENERTOTP
END IF
!write (10,10001) TEMPERAT, ENERIN

!print the temperature, day/night radiation flag, and the input energy (during delta-t) into
the output file
WRITE(2,2088) TEMPERAT, IRAD, ENERIN
2088 FORMAT('FROM WEATHER: TEMPERAT, IRAD, ENERIN: ',F7.1,I4,E15.5)

!kill off consumers after 1 ecocycle
!IF (IECOCYCLE==1) THEN
!   DO I=N1+1,NTOT                          !for all consumer species
!     IF (IWHO(I)==1) THEN                  !if the species still exists
!       DO J=IWHERE(I,1),IWHERE(I,2)              !for all individuals of the
species
!         ITEMP1=INDEX(J)
!         WHAT(ITEMP1,1)=WHAT(ITEMP1,3)     !age of individual equals max age for
individual
!       END DO
!     END IF
!   END DO
!END IF

!'zero' flow matrix
ITEMP5=NTOT+2
DO I=1,ITEMP5
  DO J=1,ITEMP5
    FLOWS(I,J)=0.0
  END DO
END DO

!increment ages - by individual
```

```
DO I=1,IPOP1
  WHAT(I,1)=WHAT(I,1)+DELTIME
END DO

!see who dies of old age - by species
DO I=1,NTOT
 AGED(I)=0                                              !set death counter to 0
END DO

DO I=1,NTOT
 IF (IWHO(I)==1) THEN
  DO J=IWHERE(I,1),IWHERE(I,2)
   ITEMP1=INDEX(J)
   IF (WHAT(ITEMP1,1) > WHAT(ITEMP1,3)) THEN
    IWHAT(ITEMP1)=0                    !this individual dies of old age
    IPOP1=IPOP1-1                      !the total population is reduced by one
    IPOPS(I)=IPOPS(I)-1                !the species population is reduced by one
    AGED(I)=AGED(I)+1                  !advance death counter
    ITEMP5=I+1                         !calculate the species position in FLOWS matrix
    ITEMP6=NTOT+2                      !calculate position of OUT compartment in
FLOWS
    FLOWS(ITEMP5,ITEMP6)=FLOWS(ITEMP5,ITEMP6)+WHAT(ITEMP1,2)  !add energy
of dead indv to flow for ItoOUT
    IF (IPOPS(I) <= 0) THEN
     IWHO(I)=0                         !if no members of the species are left
    END IF
   END IF
  END DO
 END IF
END DO

!print populations into the output file, after death from old age
WRITE(2,2089)
2089 FORMAT('POPULATIONS AFTER DEATH FROM OLD AGE:')
WRITE(2,2090) (I,IPOPS(I),I=1,NTOT)
2090 FORMAT(I5,I10)

!print death counter to file
WRITE (2,7000)
7000 FORMAT ('DEATHS DUE TO OLD AGE:')
WRITE(2,5003) (I,AGED(I),I=1,NTOT)
5003 FORMAT (I5,I7)

!check if there are any living individuals remaining at all after old age death
IF (IPOP1==0) THEN
 WRITE(2,3009)                         !if all species are dead, interrupt ecocycle and quit
```

```
  GO TO 9001
END IF
3009 FORMAT(//,'************** ECOCYCLE INTERRRUPTED SINCE NO-ONE
REMAINS AFTER OLD AGE DYING; PROGRAM STOPPED',//)

!inherent metabolism consuming energy reserves in all individuals, and death due to
starvation
RELRATE=EXP((TEMPERAT-20)/20)
WRITE(2,2091) RELRATE
2091 FORMAT('RELATIVE RATE FOR INHERENT METABOLISM (PRODUCERS ONLY,
DEPENDENT ON TEMPERATURE): ',F10.5)

DO I=1,NTOT
  SPMETAB(I)=0                                          !set energy metabolized by
each species (this time step) to 0
END DO

DO I=1,NTOT
  STARVED(I)=0                                          !set death counter to 0
END DO

DO I=1,NTOT
  IF (IWHO(I)==1) THEN
    DO J=IWHERE(I,1),IWHERE(I,2)
      ITEMP1=INDEX(J)
      IF (IWHAT(ITEMP1)/=0) THEN
        TEMP1=XMETAB(I)*WHAT(ITEMP1,2)*DELTIME          !energy loss due to
metabolism
        IF (I<=N1) THEN                          !for producer species only
          TEMP1=TEMP1*RELRATE                       !adjust metabolic rate of producers for
temperature effect
        END IF
        WHAT(ITEMP1 ,2)=WHAT(ITEMP1,2)-TEMP1
        SPMETAB(I)=SPMETAB(I)+TEMP1                     !add metabolized energy to
species' metabolized energy
        IF (WHAT(ITEMP1,2)<ENERMIN(I)) THEN           !check if individual dies of hunger
          IWHAT(ITEMP1)=0
          IPOP1=IPOP1-1
          IPOPS(I)=IPOPS(I)-1
          STARVED(I)=STARVED(I)+1                     !advance death counter for the
species
          ITEMP5=I+1                             !calculate the species position in FLOWS matrix
          ITEMP6=NTOT+2                          !calculate position of OUT compartment in
FLOWS
          FLOWS(ITEMP5,ITEMP6)=FLOWS(ITEMP5,ITEMP6)+WHAT(ITEMP1,2)  !add
energy of dead indv to flow for ItoOUT in FLOWS
```

```
        IF (IPOPS(I)<=0) THEN
          IWHO(I)=0
        END IF
       END IF
     END IF
   END DO
  END IF
END DO

!print death counter and total metabolized energy per species to file
WRITE(2,5004)
5004 FORMAT ('DEATHS DUE TO STARVATION:')
WRITE(2,7001) (I,STARVED(I),I=1,NTOT)
7001 FORMAT (I5,I7)
WRITE(2,5005)
5005 FORMAT ('TOTAL METABOLIZED ENERGY PER SPECIES:')
WRITE(2,7002) (I,SPMETAB(I),I=1,NTOT)
7002 FORMAT (I5,E15.5)

!add metabolized energy to FLOWS for each species to OUT
DO I=1,NTOT
  ITEMP5=I+1                !species I's position in FLOWS matrix
  ITEMP6=NTOT+2             !position of OUT compartment in FLOWS matrix
  FLOWS(ITEMP5,ITEMP6)=FLOWS(ITEMP5,ITEMP6)+SPMETAB(I)
END DO

!check if there are any living individuals remaining after metabolism death
IF (IPOP1==0) THEN
  WRITE(2,3010)                          !if all species are dead, interrupt ecocycle and
quit
  GO TO 9001
END IF
3010 FORMAT(//,'************** ECOCYCLE INTERRRUPTED SINCE NO-ONE
REMAINS AFTER METABOLISM; PROGRAM STOPPED',//)

!calculate the total energy content of each species for the feed probability calculations
etc.
DO I=1,NTOT
  ENER(I)=0.0                        !set initial value for species total energy
  IF (IWHO(I)==1) THEN                  !check if the species exists
    DO J=IWHERE(I,1),IWHERE(I,2)           !cycle through the individuals of the
species
      ITEMP1=INDEX(J)                   !find the individual's location in the population
matrix
      IF (IWHAT(ITEMP1)/=0) THEN            !check to see if individual is dead; if so,
do not use in calculation
```

```
      ENER(I)=ENER(I)+WHAT(ITEMP1,2)              !sum the energy for the species
     END IF
    END DO
  END IF
END DO

!calculate the total energy content of the system by summing the species' energies - for
printout purposes only
ENERTOT=0.0
DO I=1,NTOT
  ENERTOT=ENERTOT+ENER(I)
END DO

!print total energy into the output file
WRITE(2,2094) ENERTOT
2094 FORMAT('TOTAL ENERGY IN SYSTEM AFTER DEATH FROM OLD AGE AND FROM
INHERENT METABOLISM: ', E15.5)

!calculate the relative energy levels of each species for the feed probability calculations
etc.
DO I=1,NTOT
  IF (IWHO(I)==1) THEN                          !check if the species exists
    ENERREL(I)=ENER(I)/ENERTOT
  ELSE
    ENERREL(I)=0.0
  END IF
END DO

!print populations, energies, and relative energies into the output file, after inherent
metabolism and death from starvation
WRITE(2,2092)
2092 FORMAT('POPULATIONS, ENERGIES, AND RELATIVE ENERGIES AFTER
INHERENT METABOLISM AND DEATH FROM STARVATION:')
WRITE(2,2093) (I,IPOPS(I),ENER(I),ENERREL(I),I=1,NTOT)
2093 FORMAT(I5,I10,E15.5,F10.5)

!calculate the feed-not and feed-probability matrices for the consumer species
!based on the food preference matrix and the relative energies of the food species
DO I=N1+1,NTOT
  SUM1=0.0
  SUM2=0.0
  DO J=1,NTOT
    SUM1=SUM1+FOOD(I,J)*ENERREL(J)
    SUM2=SUM2+FOOD(I,J)*ENERREL(J)*ENERREL(J)
  END DO
  IF (SUM1<=0.0) THEN                !if there is no food at all for this species
```

```
   FEEDNOTPROB(I)=1.0                !the probability of not eating is 1.0
   DO J=1,NTOT
     FEEDPROB(I,J)=0.0               !and the feed probabilities are all zero
   END DO
  ELSE                               !else, if there IS food for this species
   TEMP1=1.0-TANH(SUM1/AFFECT1(I))    !TEMP1 = probability of this species not eating
at all for a DELTIME of 86400
   FEEDNOTPROB(I)=TEMP1+(86400.0-DELTIME)/86400.0*(1-TEMP1)  !this is the
probability of not eating at all during DELTIME used
   DO J=1,NTOT
     FEEDPROB(I,J)=(1.0-
FEEDNOTPROB(I))*(ENERREL(J)*ENERREL(J)*FOOD(I,J))/SUM2  !this is the probability
of it feeding on this species
   END DO
  END IF
END DO

!print the feed-not probabilities into the output file
WRITE(2,2095)
2095 FORMAT('THE FEEDNOT PROBABILITIES ARE (CONSUMER SPECIES ONLY):')
WRITE(2,2096) (FEEDNOTPROB(I),I=N1+1,NTOT)
2096 FORMAT(10F10.5)

!print the feed probabilities (matrix) into the output file
WRITE(2,2097)
2097 FORMAT('THE FEED PROBABILITIES ARE:')
DO I=N1+1,NTOT
  WRITE(2,2098) (FEEDPROB(I,J),J=1,NTOT)
END DO
2098 FORMAT(10F10.5)

!calculate the HEALTH vector, based on the species interaction
DO I=1,NTOT
  IF (IWHO(I)==1) THEN
    SUM1=0.0
    DO J=1,NTOT
      SUM1=SUM1+ENERREL(J)*XINTER(I,J)
    END DO
    HEALTH(I)=TANH(SUM1/AFFECT2(I))
  ELSE
    HEALTH(I)=0
  END IF
END DO

!print the health vector into the output file
WRITE(2,2099)
```

```
2099 FORMAT('THE HEALTH VECTOR IS:')
WRITE(2,2100) (HEALTH(I),I=1,NTOT)
2100 FORMAT(10F10.5)

!energy allocation to producer species - from radiation
IF (IRAD==1) THEN                                    !go ahead if there is radiation
  ENERTOTP=0.0                                       !set initial value of total energy for
producer species
  DO I=1,N1
    IF (IWHO(I)==1) THEN
      ENERTOTP=ENERTOTP+ENER(I)                      !increment total energy of all
producers
    END IF
  END DO
!print the total energy in the producer species to the output file
!  WRITE(2,2101) ENERTOTP
  DO I=1,N1
    IF (IWHO(I)==1) THEN
      ENERALLO(I)=ENER(I)/ENERTOTP*ENERIN            !allocate total energy to
each producer species
    ELSE
      ENERALLO(I)=0.0
    END IF
  END DO
!print the energy allocations to the producer species into the output file
!  WRITE(2,2102)
!  WRITE(2,2103) (I,ENERALLO(I),I=1,N1)

!add energy allocated to each producer species to FLOWS matrix
DO I=1,N1
  ITEMP5=I+1                    !species I's position in FLOWS matrix
  FLOWS(1,ITEMP5)=ENERALLO(I)       !was ABS(ENERALLO(I))
END DO

!energy allocation to individuals in each producer species
  DO I=1,N1                                          !process for producers only
!print energy allocation information into the output file
!    WRITE(2,2104) I,I,IWHO(I)
    IF (IWHO(I)==1) THEN                             !check if species exists
      ITEMP1=INT((ENERALLO(I)/ENERQUAN(I))+0.5)      !how many quanta to be
allocated to this species
!      WRITE(2,2105) ENERQUAN(I),ITEMP1
      ITEMP2=IWHERE(I,1)
      ITEMP3=IWHERE(I,2)-IWHERE(I,1)
!      WRITE(2,2106) ITEMP2, ITEMP3
      DO J=1,ITEMP1                                  !hand out the quanta to individuals
```

```
      K=0
      DO WHILE (K==0)
       L=INT(ITEMP2+ITEMP3*RANDOM1(1)+0.5)
       ITEMP4=INDEX(L)
       IF (IWHAT(ITEMP4)/=0) THEN
        WHAT(ITEMP4,2)=WHAT(ITEMP4,2)+ENERQUAN(I)
!        WRITE(2,2107) J, L, ITEMP4
        K=1
       END IF
      END DO
    END DO
   END IF
  END DO
ELSE
!  WRITE(2,2108)
END IF
!2101 FORMAT('THE TOTAL ENERGY IN THE PRODUCER SPECIES IS: ', E15.5)
!2102 FORMAT('THE ENERGY ALLOCATIONS TO THE PRODUCER SPECIES, BY
SPECIES, ARE:')
!2103 FORMAT(I5,E15.5)
!2104 FORMAT('ENERGY ALLOCATION INFORMATION FOR PRODUCER SPECIES:
',I4,';   IWHO(',I4,') IS: ',I2,';    INFORMATION FOLLOWING:')
!2105 FORMAT('    ENERGY QUANTUM SIZE: ',F10.3,'; NUMBER OF QUANTA
ALLOCATED: ', I7)
!2106 FORMAT('    START OF THIS SPECIES IN THE MATRIX IWHERE: ', I8,' VALUE
OF ITEMP3 IS: ',I8)
!2107 FORMAT('    QUANTUM # ',I8,'     GIVEN TO SPECIES MEMBER # ',I8,' (IN
IWHERE)      AT POSITION # ',I8,' (IN WHAT AND IWHAT)')
!2108 FORMAT('THERE WAS NO ENERGY ALLOCATION TO PRODUCER SPECIES SINCE
THE RADIATION LEVEL WAS ZERO')

!zero eat matrix
DO I=1,N2
  DO J=1,NTOT
    EAT(I,J)=0
  END DO
END DO

!feeding by consumer species
!WRITE(2,2109)
!2109 FORMAT('INFORMATION ABOUT FEEDING BY CONSUMER SPECIES:')
DO I=N1+1,NTOT
!  WRITE(2,2110) I, I, IWHO(I), I, IPOPS(I)
  IF (IWHO(I)==1) THEN                                    !continue only if there are
individuals for this species
!  WRITE(2,2111) I, IWHERE(I,1), IWHERE(I,2)
```

```
   DO J=IWHERE(I,1),IWHERE(I,2)                              !cycle through individuals of
this species
     ITEMP1=INDEX(J)                                        !find location in IWHAT and WHAT
matrices
!      WRITE(2,2112) J, ITEMP1, IWHAT(ITEMP1)
     IF (IWHAT(ITEMP1)/=0) THEN                             !if not dead, maybe look for
food
       TEMP2=RANDOM2(1)                                     !obtain random probability
between 0 and 1
       TEMP3=FEEDNOTPROB(I)                                 !probability it won't feed at all
       IF (TEMP2>TEMP3) THEN                                !if probability high enough, it
will attempt to feed
        DO K=1,NTOT                                         !choose a prey species
          TEMP3=TEMP3+FEEDPROB(I,K)
          IF (TEMP2<=TEMP3) EXIT
        END DO                                              !K is now the target species
!         WRITE(2,2113) FEEDNOTPROB(I), TEMP2, K
        IF (IWHO(K)==1) THEN                                !if no members of target species
left, then not eat
          ITEMP2=IWHERE(K,1)                                !first member of target range
          ITEMP3=IWHERE(K,2)-ITEMP2                         !rest of target range
          L=0                                               !set flag that no food has yet been found
          DO WHILE (L==0)                                   !continue to hunt while the flag L==1
            M=INT(ITEMP2+ITEMP3*RANDOM2(1)+0.5)             !randomly pick a
target index value inside the range
            ITEMP4=INDEX(M)                                 !this is the target organism
!             WRITE(2,2114) IWHERE(K,1),IWHERE(K,2),M,ITEMP4
            IF (IWHAT(ITEMP4)/=0) THEN                      !check to see if target
organism really exists
              IF (I /= J .OR. ITEMP4 /= ITEMP1) THEN        !prevent organism from
eating itself
                TEMP4=0.1*WHAT(ITEMP1,2)                    !find 10% of the energy
content of the feeding individual
                IF (WHAT(ITEMP4,2)>TEMP4) THEN              !if food individual has
more energy content than 10% of feeder
!                 WRITE(2,2115) WHAT(ITEMP1,2),TEMP4,WHAT(ITEMP1,2)+TEMP4
                  WHAT(ITEMP1,2)=WHAT(ITEMP1,2)+TEMP4       !then the feeding
individual only gets 10% of its energy
                  ITEMP5=K+1                                !position of food species in FLOWS
matrix
                  ITEMP6=I+1                                !position of consumer in FLOWS matrix
                  ITEMP7=NTOT+2                             !position of OUT in FLOWS matrix
                  FLOWS(ITEMP5,ITEMP6)=FLOWS(ITEMP5,ITEMP6)+TEMP4     !add energy
consumed to flow from food spp to consumer spp
                  FLOWS(ITEMP5,ITEMP7)=FLOWS(ITEMP5,ITEMP7)+(WHAT(ITEMP4,2)-
TEMP4)  !add energy 'lost' to flow from food spp to out
```

```
              ELSE
!               WRITE(2,2116)
WHAT(ITEMP1,2),WHAT(ITEMP4,2),WHAT(ITEMP1,2)+WHAT(ITEMP4,2)
               WHAT(ITEMP1,2)=WHAT(ITEMP1,2)+WHAT(ITEMP4,2)       !else it gets the
entire energy content of the food individual
               ITEMP5=K+1                                  !position of food species in FLOWS
matrix
               ITEMP6=I+1                                  !position of consumer in FLOWS matrix
               FLOWS(ITEMP5,ITEMP6)=FLOWS(ITEMP5,ITEMP6)+WHAT(ITEMP4,2)
!add energy of indv to flow from food spp to consumer spp
              END IF
              L=1                                      !set the flag that food has been found and
to stop eating
              IWHAT(ITEMP4)=0                           !remove the individual just eaten
from the system
              IPOPS(K)=IPOPS(K)-1                       !decrement the food species
population by one
              EAT(I-N1,K)=EAT(I-N1,K)+1                  !increase number of k's eaten by
i's in eat matrix by 1
              IF (IPOPS(K)==0)  THEN                    !check to see if the population still
has any members
               IWHO(K)=0                                !if no members left, set population
indicator flag to zero
              END IF
              IPOP1=IPOP1-1                             !decrement the entire population by
one
!              WRITE(2,2117)IPOPS(K),IWHO(K),IPOP1
             ELSE
!              WRITE(2,2118)
              !NOTE: this situation will only occur for a cannibalistic species, when a
consumer just tried to eat himself; this is not
              !allowed, but it is now possible there are no other members of this species left;
in that case, the feeding attempt should
              !be abandoned
              IF (IPOPS(I)==1) EXIT                     !exit from the feeding loop for this
individual consumer
             END IF
            ELSE
!             WRITE(2,2119) IWHAT(ITEMP4)
            END IF
          END DO
         ELSE
!          WRITE(2,2120)
         END IF
       ELSE
!        WRITE(2,2121) TEMP2, TEMP3
```

```
      END IF
     ELSE
!      WRITE(2,2122)
     END IF
   END DO
  ELSE
!   WRITE(2,2123)
  END IF
END DO
!2110 FORMAT('  FEEDING INFORMATION FOR CONSUMER SPECIES: ',I4,';
IWHO(',I4,') IS: ',I2,';   POPULATION(',I4,') IS: ',I7)
!2111 FORMAT('   LOCATION OF CONSUMER SPECIES ',I4,' IN INDEX MATRIX -
START: ',I7,';   END: ',I7)
!2112 FORMAT('      LOCATION OF INDIVIDUAL IN INDEX MATRIX IS: ',I7,';
LOCATION IN IWHAT MATRIX: ',I7,';   VALUE OF IWHAT:',I5)
!2113 FORMAT('       FEED_NOT PROBABILITY WAS: ',F7.4,';  RANDOM PROBABILITY
WAS: ', F7.4,'; THE CHOSEN TARGET SPECIES WAS: ',I5)
!2114 FORMAT('       START AND END OF FEED RANGE IN WHAT MATRICES: ',2I7,';
NUMBER PICKED WAS: ',I7,';   IN INDEX:',I7)
!2115 FORMAT('       INITIAL ENERGY CONTENT OF CONSUMER: ',E12.5,'   IT GETS
10% OF ITS ENERGY: ',E12.5,'   FINAL ENERGY: ',E12.5)
!2116 FORMAT('       INITIAL ENERGY CONTENT OF CONSUMER: ',E12.5,'   IT GETS
FROM FEEDING: ',E12.5,'   FINAL ENERGY: ',E12.5)
!2117 FORMAT('       TARGET POPULATION NOW: ',I7,';   IWHO VALUE OF TARGET:
',I2,';   TOTAL LIVING POPULATION NOW: ',I7)
!2118 FORMAT('       OOPS! THIS IS A CANNIBALISTIC SPECIES AND THIS GUY
JUST TRIED TO EAT HIMSELF - NOT ALLOWED - TRY AGAIN')
!2119 FORMAT('       OOPS! THE IWHAT VALUE FOR THIS GUY IS NOW',I2,'; THAT
MEANS IT''S DEAD - TRY FOR ANOTHER ONE OF THIS SPECIES')
!2120 FORMAT('       NO FEEDING BY THIS GUY BECAUSE THERE IS NO POPULATION
LEFT FOR THE TARGET SPECIES')
!2121 FORMAT('       RANDOM PROBABILITY WAS: ',F7.4,';  LESS THAN FEED_NOT
PROBABILITY: ',F7.4,'  THUS, NO FEEDING BY THIS GUY')
!2122 FORMAT('       THIS INDIVIDUAL IS DEAD - NO FEEDING BEHAVIOR')
!2123 FORMAT('   NO FEEDING BY THIS SPECIES SINCE THERE IS NO
POPULATION')

!write eat matrix to output file
DO I=1,N2
  WRITE (3,30) (EAT(I,J),J=1,NTOT)
  30 FORMAT (30I7)
END DO

!matrix cleanup - Part 1 - filter out the dead ones
!WRITE(2,2124)
```

```
!2124 FORMAT('INFORMATION ABOUT MATRIX CLEANUP PART 1 - FILTERING OUT
THE DEAD ONES:')
ITEMP1=IPOP2-IPOP1                    !ITEMP1 is now the number of dead ones to filter
out every ecocycle
ITEMP2=0                              !ITEMP2 is the number of dead ones found so far
!WRITE(2,2125) IPOP2, IPOP1,ITEMP1
!2125 FORMAT('  LIVING POPULATION AT START OF ECOCYCLE: ',I7,';    AT END:
',I7,'    DEAD ONES TO FILTER OUT: ',I7)
IF (ITEMP1 /= 0) THEN                 !only do this if there are dead individuals present
  K=IPOP2                             !set K - pointer at bottom of WAHT and IWHAT matrices
- it will move up gradually
  DO I=1,IPOP2                        !maximally, search through the entire matrix
   IF (IWHAT(I)==0) THEN              !if dead, try to exchange with the one on the
bottom, after checking if OK
     ITEMP2=ITEMP2+1                  !increment the number of dead ones found
!     WRITE(2,2126) I, ITEMP2
     IF (I>=K) THEN                   !check to see if it no longer makes sense to find more -
if YES, exit from loop
!       WRITE(2,2127) I,K
       EXIT
     END IF
     DO WHILE (IWHAT(K)==0)           !bottom one is dead - pop up one place
       K=K-1                          !move pointer up one place
       ITEMP2=ITEMP2+1                !increment the number of dead ones found
!       WRITE(2,2128) ITEMP2
     END DO
     IF (I>=K) THEN                   !check to see if it no longer makes sense to find more -
if YES, exit from loop
!       WRITE(2,2127) I,K
       EXIT
     END IF
     IWHAT(I)=IWHAT(K)                !move the next live one (from the bottom) into the
position of the dead one
     WHAT(I,1)=WHAT(K,1)
     WHAT(I,2)=WHAT(K,2)
     WHAT(I,3)=WHAT(K,3)
     K=K-1                            !move pointer up one place
!     WRITE(2,2129) K+1,I,K
!     WRITE(2,2130) IWHAT(I),WHAT(I,1),WHAT(I,2),WHAT(I,3)
     IF (ITEMP2>=ITEMP1) THEN         !all the dead ones were found - quit the loop
!       WRITE(2,2131) ITEMP2
       EXIT
     END IF
   END IF
  END DO
!  WRITE(2,2132)
```

```
ELSE
!  WRITE(2,2133)
END IF
!2126 FORMAT('  DEAD ONE FOUND AT LOCATION',I7,' IN THE IWHAT MATRIX;
TOTAL DEAD ONES FOUND SO FAR: ',I7)
!2127 FORMAT('  IN FINDING DEAD ONES I IS NOW PAST THE K POINTER, DON''T
BOTHER FINDING THE REST;  I AND K ARE: ',2I7)
!2128 FORMAT('  WHOA! FOUND ANOTHER DEAD ONE WHILE TRYING TO EXCHANGE
POSITIONS;    TOTAL DEAD FOUND SO FAR: ',I7)
!2129 FORMAT('  INDIVIDUAL MOVED UP FROM BOTTOM OF MATRIX
(POSITION',I7,') TO POSITION',I7,'    K POINTER NOW AT: ',I7)
!2130 FORMAT('    INDIVIDUAL MOVED WAS SPECIES: ',I4,';    AGE: ',E12.5,';
ENERGY: ',E12.5,';    MAXAGE:',E12.5)
!2131 FORMAT('ALL ',I7,' DEAD ONES WERE FOUND AND DEALT WITH - EXITING
FROM LOOP')
!2132 FORMAT('END OF MATRIX CLEANUP PART 1 - FILTERING OUT THE DEAD
ONES')
!2133 FORMAT('  NO DEAD ONES TO FILTER OUT - CARRY ON')

!check to see if the populations still make sense etc.
!re-calculate energy in each species
DO I=1,NTOT
  ENER(I)=0.0                                  !set initial value for species total energy
  IF (IWHO(I)==1) THEN                            !check if the species exists
    DO J=IWHERE(I,1),IWHERE(I,2)                     !cycle through the individuals of the
species
      ITEMP1=INDEX(J)                             !find the individual's location in the population
matrix
      ENER(I)=ENER(I)+WHAT(ITEMP1,2)                 !sum the energy for the species
    END DO
  END IF
END DO
WRITE(2,2134)
2134 FORMAT('CHECKING UP ON THE POPULATIONS TO SEE IF THEY STILL MAKE
SENSE:')
WRITE(2,2135)
2135 FORMAT('SPECIES#   PRESENCE   POPULATION   TOTAL SPECIES ENERGY')
WRITE(2,2136)
2136 FORMAT('--------  --------  ----------  --------------------')
WRITE(2,2137) (I,IWHO(I),IPOPS(I),ENER(I),I=1,NTOT)
2137 FORMAT(I6,I11,I13,E23.5)
ITEMP1=0.0
DO I=1,NTOT
  ITEMP1=ITEMP1+IPOPS(I)
END DO
WRITE(2,2138)ITEMP1,IPOP1,K
```

```
2138 FORMAT('TOTAL # INDIVIDUALS FROM SUMMING: ',I10,'    IPOP1: ',I10,'
INDEX K AT: ',I10)

!reproduction - all species - process individuals in sequence in the WHAT matrix
!WRITE(2,2139)
!2139 FORMAT('REPORT FROM REPRODUCTION SECTION:')

DO I=1,NTOT
  BIRTHS(I)=0                                          !set birth counter to 0
END DO

DO I=1,IPOP1                                  !cycle through the entire population, living and
dead
   J=IWHAT(I)                                  !J is now the species number
   IF (WHAT(I,2)>ENERREP(J)) THEN                      !this one is reproducing
     IPOP1=IPOP1+1                             !update total living population
     BIRTHS(J)=BIRTHS(J)+1                             !increase species birth counter
     IF (IPOP1 > MAXPOP) THEN                  !test the size of the new total population
       WRITE(2,3003)
       WRITE(2,3004)
       WRITE(2,3005) IPOP1, MAXPOP
       WRITE(2,3006)
       STOP
     END IF
     IPOPS(J)=IPOPS(J)+1                        !update species population
     WHAT(I,2)=WHAT(I,2)-(2.0-0.5*HEALTH(J))*ENERBIR(J) !"mother" individual loses
energy in accordance with HEALTH of species
     ITEMP5=J+1                                !postion of reproducing spp in FLOWS matrix
     ITEMP6=NTOT+2                              !position of OUT in FLOWS matrix
     FLOWS(ITEMP5,ITEMP6)=FLOWS(ITEMP5,ITEMP6)+(2.0-
0.5*HEALTH(J))*ENERBIR(J)-ENERBIR(J)  !add nrg mother loses minus energy of baby to
flow
     IWHAT(IPOP1)=J                             !baby inherits species
     WHAT(IPOP1,1)=0.0                          !baby born at age zero
     WHAT(IPOP1,2)=ENERBIR(J)                   !allocate newborn energy at birth
     TEMP1=RANDOM3(1)                                    !get random number
     WHAT(IPOP1,3)=XMINMAXAGE(J)+TEMP1*(XMAXMAXAGE(J)-XMINMAXAGE(J))
          !allocate maximum age between XMAXMAXAGE and XMINMAXAGE
!     WRITE(2,2141)IPOP1,IPOP1,IPOPS(J)
   END IF
END DO
!2140 FORMAT('  LOCATION IN WHAT:',I7,';    SPECIES:',I5,';    ENERGY OF THIS
INDIVIDUAL NOW: ',E15.5)
!2141 FORMAT('    NEW INDIVIDUAL CREATED AT LOCATION:',I7,'    TOTAL
POPULATION NOW:',I7,'    SPECIES POPULATION NOW:',I7)
```

```
3003 FORMAT(///,'***************************** PROBLEM REPORT
*****************************')
3004 FORMAT('DURING REPRODUCTION PHASE, TOTAL POPULATION BECOMES
LARGER THAN MAXIMUM ALLOWED')
3005 FORMAT('TOTAL POPULATION IS: ',I15,';    MAXIMUM ALLOWED IS: ',I15)
3006 FORMAT('PROGRAM STOPPED BEFORE FATAL CRASH OCCURRED')

!print number of births to file
WRITE(2,5006)
5006 FORMAT ('BIRTHS THIS CYCLE:')
WRITE(2,7003) (I,BIRTHS(I),I=1,NTOT)
7003 FORMAT (I5,I7)

!set total matrix length again to total living population
IPOP2=IPOP1
!WRITE(2,2142) IPOP1,IPOP2
!2142 FORMAT('TOTAL LIVING POPULATION (IPOP1) NOW: ',I10,';    MATRIX LENGTH
(IPOP2) NOW SET AT: ',I10)

!matrix cleanup - Part 2 - rebuild INDEX and WHERE matrices
K=0                             !set the index counter to zero
!WRITE(2,2143)
!2143 FORMAT('INFORMATION FROM INDEX AND WHERE MATRIX REBUILDING
SECTION')
DO I=1,NTOT                         !do this for each species
 IF (IWHO(I)==0) THEN               !if there are no members of this species present
  IWHERE(I,1)=0                     !set the locations in the INDEX matrix to zero
  IWHERE(I,2)=0
!   WRITE(2,2144)I
!   WRITE(2,2145)I,IWHO(I),IWHERE(I,1),IWHERE(I,2)
 ELSE                               !else, set the locations
  IWHERE(I,1)=K+1
  IWHERE(I,2)=K+IPOPS(I)
!   WRITE(2,2146)I
!   WRITE(2,2147)I,IWHO(I),IWHERE(I,1),IWHERE(I,2)
  L=0                               !set the hit counter for the species to zero
  DO J=1,IPOP1                      !go find IPOPS(I) members of this species
   IF (IWHAT(J)==I) THEN            !found one
    K=K+1                           !increment the index counter
    INDEX(K)=J                      !record the occurrence in INDEX
    L=L+1                           !increment the hit counter for the species
   END IF
   IF (L>=IPOPS(I)) EXIT            !test to see if all have been found; if YES, exit the loop
for the species
  END DO
!   WRITE(2,2148)I,IPOPS(I),L
```

```
  END IF
END DO
!2144 FORMAT('FOR SPECIES (',I5,'): NO POPULATION LEFT ---')
!2145 FORMAT('  SPECIES= ',I5,'    IWHO= ',I2,'    IWHERE(I,1)= ',I7,'    IWHERE(I,2)=
',I7)
!2146 FORMAT('FOR SPECIES (',I5,'): THERE STILL IS A LIVING POPULATION')
!2147 FORMAT('  SPECIES= ',I5,'    IWHO= ',I2,'    IWHERE(I,1)= ',I7,'    IWHERE(I,2)=
',I7)
!2148 FORMAT('  SPECIES= ',I5,'    SPECIES POPULATION= ',I10,'    INDIVIDUALS
FOUND= ',I10)

!total population check
ITEMP1=0
DO I=1,NTOT
  ITEMP1=ITEMP1+IPOPS(I)
END DO
WRITE(2,2149)IPOP1,ITEMP1
2149 FORMAT('POPULATION CHECK: TOTAL LIVING POPULATION (IPOP1)= ',I10,';
SUM OF SPECIES POPULATIONS= ',I10)
IF (IPOP1 /= ITEMP1) THEN
  WRITE(2,2150)
  STOP
END IF
2150 FORMAT(///,' ************************* POPULATION CHECK ERROR!!!!! ----
PROGRAM STOPPED',///)

!calculate the total energy content of each species at the end of this ecocyle
WRITE(2,2151) IECOCYCLE
2151 FORMAT('FINAL REPORT AT END OF ECOCYCLE',I8,':',/,'    I   IWHO   IPOPS
ENERGY')
WRITE(2,2152)
2152 FORMAT('-------------------------------------------------------------------')
DO I=1,NTOT
  ENER(I)=0.0                                !set initial value for species total energy
  IF (IWHO(I)==1) THEN                              !check if the species exists
    DO J=IWHERE(I,1),IWHERE(I,2)                    !cycle through the individuals of the
species
      ITEMP1=INDEX(J)                              !find the individual's location in the population
matrix
      ENER(I)=ENER(I)+WHAT(ITEMP1,2)                !sum the energy for the species
    END DO
  END IF
END DO
WRITE(2,2153)(I,IWHO(I),IPOPS(I),ENER(I),I=1,NTOT)
2153 FORMAT(I5,I7,I10,E15.5)
```

```
!calculate the total energy content of the system at the end of this ecocyle
ENERTOT=0.0
DO I=1,NTOT
  ENERTOT=ENERTOT+ENER(I)
END DO
WRITE(2,2154) IECOCYCLE, ENERTOT
2154 FORMAT('THE TOTAL ENERGY CONTENT OF THE SYSTEM AT THE END OF
ECOCYCLE',I8,' IS: ',E15.5)

!print flows to file
!DO I=1,NTOT+2
!  WRITE (5,501) (FLOWS(I,J),J=1,NTOT+2)
!  501 FORMAT (8F15.5)
!END DO

!WRITE (5,502) TEXTLINE
!502 FORMAT (A1)

!calculate ascendency for this time step and write to file
CALL ASCEND(FLOWS,A)
!print *, A
WRITE(4,401) A
401 FORMAT (F15.5)
WRITE(2,2161) A
2161 FORMAT ('ASCENDENCY FOR ECOCYCLE:',F15.5)

WRITE(2,2155)IECOCYCLE
2155 FORMAT('************************* END OF OUTPUT FROM ECOCYCLE',I8,'
**********************')

!return to the top for the next iteration cycle
GOTO 9000

9001 CONTINUE

WRITE(2,2156)
2156 FORMAT(/,'***************************** END OF OUTPUT FROM ITERATION
PHASE *********************************')

!////////////////////////////////////////////////////////////////////////////////
!#####                          END OF ITERATION PHASE                        #####
!////////////////////////////////////////////////////////////////////////////////

CALL CPU_TIME(C)
E=C
RTIME=E-S
```

```
WRITE(2,2160) RTIME
2160 FORMAT ('TIME TO RUN SIMULATION:',F10.2,'SECONDS')

!#########################################################
#########################################################
!####                    START OF ENDING PHASE                    ####
!#########################################################
#########################################################

!write ending message and close the output file
WRITE(2,2157)
2157 FORMAT(////,
'*******************************************************************************
****')
WRITE(2,2158)
2158
FORMAT('*******************************************************************
*************')
WRITE(2,2159)
2159 FORMAT('********************************** END OF OUTPUT
***********************************')
WRITE(2,2158)
WRITE(2,2158)

CLOSE (UNIT=2)
CLOSE (UNIT=3)
CLOSE (UNIT=4)
!CLOSE (UNIT=5)
!close (unit=9)
!close (unit=10)
!close (unit=11)
!close (unit=12)
!close (unit=13)
!close (unit=14)
!CLOSE (UNIT=15)
!CLOSE (UNIT=16)

END PROGRAM ECOSYS

!///////////////////////////////////////////////////////////////////////////////
!####                    END OF ENDING PHASE                    ####
!///////////////////////////////////////////////////////////////////////////////

!///////////////////////////////////////////////////////////////////////////////
!####                    END OF MAIN PROGRAM                    ####
!///////////////////////////////////////////////////////////////////////////////
```

```
!#########################################################
#########################################################
!####                    SUBROUTINES                  ####
!#########################################################
#########################################################

!*************************************************************************
!*************************************************************************

!####                START OF SUBROUTINE WEATHER              ####
SUBROUTINE
WEATHER(ICASE,IYEAR,IDAY,TIMEOFDAY,DELTIME,ENERMAX,POWRMAX,ALPHA,EN
ERTOTP,T,IRAD,ENERGY,TEMFILE,RADFILE)

!declarations
IMPLICIT NONE
INTEGER*4,INTENT(IN):: ICASE                    !weather case being dealt
with; 1=initialization, 2=routine operation
INTEGER*4,INTENT(IN):: IDAY                     !day for which data is needed
(days)
INTEGER*4,INTENT(OUT):: IRAD                    !radiation flag (0=nighttime-
no radiation at all; 1=daytime)
INTEGER*4,INTENT(IN):: IYEAR                    !year for which data is
needed (years)
INTEGER*4:: IERROR1                             !error code from temperature
subroutine
INTEGER*4:: IERROR2                             !error code from radiation
subroutine
REAL*4,INTENT(IN):: ALPHA                       !variable used in the
calculation of the attenuation factor for energy input (no units)
REAL*4,INTENT(IN):: DELTIME                     !time increment (seconds)
REAL*4:: DELX                                   !delta used in x for solving
REAL*4,INTENT(OUT):: ENERGY                     !energy into system during
delta-t
REAL*4,INTENT(IN):: ENERMAX                     !maximum total energy for
system (energy units)
REAL*4,INTENT(IN):: ENERTOTP                    !total energy present in the
producer species only (energy units)
REAL*4:: F1                                     !first function value
REAL*4:: F2                                     !second function value
REAL*4,PARAMETER:: PI=3.1415926536              !value of pi
REAL*4,INTENT(IN):: POWRMAX                     !maximum total power input
for the system (energy units/second)
```

```
REAL*4:: R                                    !the relative radiation
intensity (0.0 <= R <= 1.0)
REAL*4,INTENT(OUT):: T                        !the temperature (deg. C)
REAL*4:: TEMP1                                !temporary variable to hold
random value
REAL*4,INTENT(IN):: TIMEOFDAY                 !the time of day for which
data is needed (seconds)
REAL*4:: X                                    !independent variable for
attenuation curve (no units)
REAL*4:: XNORM                                !normalization value for the
attenuation factor
CHARACTER (LEN=14):: TEMFILE                  !input file name for
temperature subroutine
CHARACTER (LEN=14):: RADFILE                  !input file name for radiation
subroutine

SELECT CASE (ICASE)

  CASE(1)
    WRITE(2,101)
101 FORMAT(/,'****** MESSAGES FROM SUBROUTINE WEATHER DURING
INITIALIZATION ******')
!iniitialize temperature subroutine
   CALL TEMPERAT(0,IYEAR,IDAY,TIMEOFDAY,T,IERROR1,TEMFILE)
         IF (IERROR1 /= 0) THEN
           WRITE (2,102)
           102 FORMAT (/,'ERROR DURING TEMPERATURE SUBROUTINE
INITIALIZATION')
           STOP
         ELSE
           WRITE (2,103)
           103 FORMAT (/,'TEMPERATURE SUBROUTINE INITIALIZED')
         END IF
!initialize radiation subroutine
   CALL RADIAT(0,IYEAR,IDAY,TIMEOFDAY,IRAD,R,IERROR2,RADFILE)
         IF (IERROR2 /= 0) THEN
           WRITE (2,104)
           104 FORMAT (/,'ERROR DURING RADIATION SUBROUTINE
INITIALIZATION')
           STOP
         ELSE
           WRITE (2,105)
           105 FORMAT (/,'RADIATION SUBROUTINE INITIALIZED')
         END IF

   DELX=0.01
```

```
   X=0.5
   F1=EXP(-ALPHA/X)*(1+ALPHA/X)-EXP(-ALPHA)
1   X=X+DELX
   F2=EXP(-ALPHA/X)*(1+ALPHA/X)-EXP(-ALPHA)
   IF (ABS(F2-F1)<0.000001) THEN
    GO TO 2
   ENDIF
   IF (ABS(F2) < ABS(F1)) THEN
    F1=F2
    GO TO 1
   ELSE
    DELX=-DELX/2.0
    F1=F2
    GO TO 1
   END IF

2   WRITE(2,106) ALPHA
106 FORMAT('THE VALUE OF ALPHA FOR THE ATTENUATION FACTOR CURVE IS:
',F10.5)
   WRITE(2,107) X
107 FORMAT('THE CORRESPONDING VALUE OF X FOUND FOR THE ATTENUATION
FACTOR CURVE IS: ',F10.5)
   XNORM=X*(1.0-EXP(-ALPHA/X)/EXP(-ALPHA))
   WRITE(2,108) XNORM
108 FORMAT('THE NORMALIZATION VALUE FOR THE ATTENUATION FACTOR
CURVE IS: ',F10.5)
   WRITE(2,109)
109 FORMAT('****** END OF MESSAGES FROM SUBROUTINE WEATHER DURING
INITIALIZATION ******')
   RETURN

  CASE(2)
   !check inputs
   IF (IYEAR <= 0) THEN
    PRINT *
    PRINT *
    PRINT *, "MESSAGE FROM SUBROUTINE WEATHER"
    PRINT *, "VALUE OF IYEAR OUTSIDE ALLOWABLE LIMITS"
    PRINT *, "VALUE OF IYEAR WAS: ",IYEAR
    PRINT *, "PROGRAM WAS STOPPED"
    STOP
   ENDIF
   IF ((IDAY < 1) .OR. (IDAY > 365)) THEN
    PRINT *
    PRINT *
    PRINT *, "MESSAGE FROM SUBROUTINE WEATHER"
```

```
     PRINT *, "VALUE OF IDAY OUTSIDE ALLOWABLE LIMITS"
     PRINT *, "VALUE OF IDAY WAS: ",IDAY
     PRINT *, "PROGRAM WAS STOPPED"
     STOP
    ENDIF
    IF ((TIMEOFDAY < 0.0) .OR. (TIMEOFDAY > 86400.0)) THEN
     PRINT *
     PRINT *
     PRINT *, "MESSAGE FROM SUBROUTINE WEATHER"
     PRINT *, "TIMEOFDAY VALUE OUTSIDE ALLOWABLE LIMITS"
     PRINT *, "TIMEOFDAY VALUE WAS: ",TIMEOFDAY
     PRINT *, "PROGRAM WAS STOPPED"
     STOP
    ENDIF

!call temperature subroutine
    CALL TEMPERAT(1,IYEAR,IDAY,TIMEOFDAY,T,IERROR1,TEMFILE)
          IF (IERROR1 /= 0) THEN
           PRINT *
           PRINT *
           PRINT *, "MESSAGE FROM SUBROUTINE WEATHER"
           PRINT *, "ERROR FROM SUBROUTINE TEMPERAT"
           PRINT *, "ERROR CODE WAS: ", IERROR1
           PRINT *, "PROGRAM WAS STOPPED"
           STOP
          END IF

!call radiation subroutine
    CALL RADIAT(1,IYEAR,IDAY,TIMEOFDAY,IRAD,R,IERROR2,RADFILE)
          IF (IERROR2 /= 0) THEN
           PRINT *
           PRINT *
           PRINT *, "MESSAGE FROM SUBROUTINE WEATHER"
           PRINT *, "ERROR FROM SUBROUTINE RADIAT"
           PRINT *, "ERROR CODE WAS: ", IERROR2
           PRINT *, "PROGRAM WAS STOPPED"
           STOP
          END IF
          !write (11,1101) R
          !1101 format (E15.5)
    IF(IRAD==0) THEN
     ENERGY=0.0
     RETURN
    ELSEIF (ENERTOTP==0) THEN
     ENERGY=0.0
     RETURN
```

```
   ELSE
     X=ENERTOTP/ENERMAX
     F1=X*(1.0-EXP(-ALPHA/X)/EXP(-ALPHA))/XNORM
     ENERGY=POWRMAX*DELTIME*(R/1200)*F1
   END IF

!write (9,901) T, ENERGY
!901 FORMAT (f10.5,E15.5)

         RETURN

END SELECT

END SUBROUTINE WEATHER
!#####                    END OF SUBROUTINE WEATHER                    #####

!*********************************************************************************

!######                    START OF SUBTROUTINE TEMPERAT                    ####
SUBROUTINE TEMPERAT(ICASE,IYEAR,IDAY,TIME,RESULT1,IERROR,TEMFILE)
!Subroutine to obtain a temperature in deg. C (RESULT1) for a given year (IYEAR), day
(IDAY), and
!time in seconds (TIME).  Also included as an input is a case switch (ICASE) to determine
whether
!it is an initialization call or normal operation.  Outputs are the temperature value and an
error
!code (IERROR).

IMPLICIT NONE
!declare dimensioned REAL*4 variables
REAL*4, DIMENSION(1095):: DATS                    !vector of three years of
DATs
REAL*4, DIMENSION(365):: DATVECT                  !vector of a single year of
DATs - from DATGEN
REAL*4, DIMENSION(365):: DATTIME                  !vector of days in fractions of
a year (DATGEN)
REAL*4, DIMENSION(180):: MAGAVPL                  !magnitude averages
polynomial (DATGEN)
REAL*4, DIMENSION(180):: MAGSDPL                  !magnitude standard deviation
polynomial (DATGEN)
REAL*4, DIMENSION(13):: XA                        !time array for SPLINE and
SPLINT
REAL*4, DIMENSION(13):: YA                        !temperature array for
SPLINE and SPLINT
REAL*4, DIMENSION(13):: Y2                         !spline array
```

```
!declare REAL*4 variables
REAL*4:: TIME                                   !time of day in seconds
REAL*4:: RESULT1                                !temperature in deg. C - value
returned to main
REAL*4:: A0AV, A1AV, A2AV, A3AV, A4AV           !ave. param. values for DAT
frequency sinusoids
REAL*4:: A0SD, A1SD, A2SD, A3SD, A4SD           !stdev. param values for DAT
frequency sinusoids
REAL*4:: B0AV, B1AV, B2AV                       !ave. param. values for DAT
beat sinusoid
REAL*4:: B0SD, B1SD, B2SD                       !stdev. param. values for DAT
beat sinusoid
REAL*4:: D0AV, D1AV, D2AV, D3AV, D4AV, D5AV     !ave. param. values for DAT
noise
REAL*4:: D0SD, D1SD, D2SD, D3SD, D4SD, D5SD     !stdev. param. values for DAT
noise
REAL*4:: RHO                                    !correlation coefficient
between A0 and A1
REAL*4:: AMPUP, AMPDOWN                         !amplitude parameters for
SPLINE
REAL*4:: OLDVALUE                               !the temp. at 'midnight' for
IDAY-1 - for SPLINE
REAL*4:: RESULT2                                !temperature in deg. C -
returned from SPLINT
REAL*4:: FREQ                                   !working variable for
frequency calculations (DATGEN)
REAL*4:: TEMP                                   !temporary variable

!declare INTEGER*4 variables
INTEGER*4:: ICASE                               !0=initialization, 1=normal
operation
INTEGER*4:: IYEAR                               !year input from the call
INTEGER*4:: IDAY                                !day input from call
INTEGER*4:: IERROR                              !error code: 0=no error
detected, 1=ICASE out of range, 2=IYEAR out of range, 3=IDAY out of range, 4=TIME out
of range, 5=error from DATGEN, 6=error from SPLINE, 7=error from SPLINT
INTEGER*4:: IYEARC                              !'current' year from last call
(normal operation)
INTEGER*4:: IDAYC                               !'current' day from last call
(normal operation)
INTEGER*4:: IERROR1                             !error code from DATGEN:
0=none detected
INTEGER*4:: IERROR2                             !error code from SPLINE:
0=none detected
INTEGER*4:: IERROR3                             !error code from SPLINT:
0=none detected
```

```
INTEGER*4:: ISEED                                   !random number seed (for
DATGEN)
INTEGER*4:: ISTART                                  !working variable for use with
DATS
INTEGER*4:: I, J                                    !working variables

!declare CHARACTER variables
CHARACTER (LEN=14):: TEMFILE                        !symbolic name of the file
that contains input data
CHARACTER (LEN=10):: TEXTLINE                       !dummy input line

!declare functions
REAL*4:: RANDTEM                                    !random number generator

!common blocks
!weather parameters and variables used by DATGEN
COMMON /BLOCK01/
A0AV,A0SD,A1AV,A1SD,A2AV,A2SD,A3AV,A3SD,A4AV,A4SD,B0AV,B0SD,B1AV,B1SD,B2
AV,B2SD,D0AV,D1AV,D2AV,D3AV,D4AV,D5AV,D0SD,D1SD,D2SD,D3SD,D4SD,D5SD,RHO

COMMON /BLOCK02/    DATTIME, MAGAVPL, MAGSDPL

!parameters and variables used by SPLINE and SPLINT
COMMON /BLOCK03/    AMPUP,AMPDOWN

COMMON /BLOCK04/    XA,YA,Y2                        !XA used by TEMPERAT,
SPLINE, SPLINT; YA,Y2 used by SPLINE, SPLINT


!set initial value of IERROR to 0
IERROR = 0


!check ICASE value
IF (ICASE /= 0 .AND. ICASE /= 1) THEN
  IERROR = 1
  RETURN
END IF

!determine initialization or normal operation
IF (ICASE == 1) THEN
  GOTO 2000
END IF

!initialization routine
!initialization part 1 - read in data
```

```
OPEN (UNIT=1, FILE=TEMFILE)

100 FORMAT (1A10)
101 FORMAT (I10)
102 FORMAT (5F10.5)
103 FORMAT (3F10.5)
104 FORMAT (E15.5)
105 FORMAT (F10.5)
106 FORMAT (2F10.3)

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,101) ISEED

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,102) A0AV, A1AV, A2AV, A3AV, A4AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,102) A0SD, A1SD, A2SD, A3SD, A4SD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,103) B0AV, B1AV, B2AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,103) B0SD, B1SD, B2SD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,104) D0AV
READ (1,104) D1AV
READ (1,104) D2AV
READ (1,104) D3AV
READ (1,104) D4AV
READ (1,104) D5AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,104) D0SD
READ (1,104) D1SD
READ (1,104) D2SD
READ (1,104) D3SD
```

```
READ (1,104) D4SD
READ (1,104) D5SD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,105) RHO

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,106) AMPUP, AMPDOWN

CLOSE (UNIT=1)

!WRITE (2,101) ISEED
!WRITE (2,102) A0AV, A1AV, A2AV, A3AV, A4AV
!WRITE (2,102) A0SD, A1SD, A2SD, A3SD, A4SD
!WRITE (2,103) B0AV, B1AV, B2AV
!WRITE (2,103) B0SD, B1SD, B2SD
!WRITE (2,104) D0AV
!WRITE (2,104) D1AV
!WRITE (2,104) D2AV
!WRITE (2,104) D3AV
!WRITE (2,104) D4AV
!WRITE (2,104) D5AV
!WRITE (2,104) D0SD
!WRITE (2,104) D1SD
!WRITE (2,104) D2SD
!WRITE (2,104) D3SD
!WRITE (2,104) D4SD
!WRITE (2,104) D5SD
!WRITE (2,105) RHO
!WRITE (2,106) AMPUP, AMPDOWN

!initialization part 2 - initialization of variables for DATGEN
!initialize random number generator
TEMP = RANDTEM(ISEED)

!fill DATTIME vector
DO I=1,365
  DATTIME(I) = FLOAT(I)/365
END DO

!calculate polynomials for noise magnitude: average and standard deviation
DO I=1,180
  FREQ = I + 2
```

```
  MAGAVPL(I) = D0AV + D1AV*FREQ + D2AV*FREQ**2 + D3AV*FREQ**3 +
D4AV*FREQ**4 + D5AV*FREQ**5
  MAGSDPL(I) = D0SD + D1SD*FREQ + D2SD*FREQ**2 + D3SD*FREQ**3 +
D4SD*FREQ**4 + D5SD*FREQ**5
END DO

!initialization part 3 - initialize variables for SPLINE
!set XA vector - time since midnight yesterday
XA(1) = 18000.0                                          !yesterday 05:00
XA(2) = 28800.0                                          !yesterday 08:00
XA(3) = 50400.0                                          !yesterday 14:00
XA(4) = 79200.0                                          !yesterday 22:00
XA(5) = 86400.0                                          !yesterdy 24:00
XA(6) = 104400.0                                         !today 05:00
XA(7) = 115200.0                                         !today 08:00
XA(8) = 136800.0                                         !today 14:00
XA(9)  = 165600.0                                        !today 22:00
XA(10) = 190800.0                                        !tomorrow 05:00
XA(11) = 201600.0                                        !tomorrow 08:00
XA(12) = 223200.0                                        !tomorrow 14:00
XA(13) = 252000.0                                        !tomorrow 22:00

!initialization part 4 - set initial values for main variables
IYEARC = IYEAR - 1
IDAYC = IDAY - 1

!fill up DAT vector - DATS
ISTART = 0

DO I=1,3
  CALL DATGEN(DATVECT,IERROR1)
  IF (IERROR1 /= 0) THEN
    IERROR = 5
          RETURN
  END IF
  DO J=1,365
    DATS(ISTART+J) = DATVECT(J)
  END DO
  ISTART = ISTART + 365
END DO

!calculate spline for the current day
OLDVALUE = (DATS(IDAYC+365) + DATS(IDAYC+364))/2

CALL SPLINE(DATS,IDAYC,OLDVALUE,IERROR2)
```

```
IF (IERROR2 /= 0) THEN
  IERROR = 6
  RETURN
END IF

!trial call of SPLINT
CALL SPLINT(TIME,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
END IF

IF (IDAYC == 0) THEN
  IDAYC = 365
END IF

RETURN

!normal operation
2000 CONTINUE

!test range of IYEAR
IF (IYEAR<0 .OR. IYEAR>10000) THEN
  IERROR=2
  RETURN
END IF

!test range of IDAY
IF (IDAY<1 .OR. IDAY>365) THEN
  IERROR=3
  RETURN
END IF

!test range of TIME
IF (TIME<0.0 .OR. TIME>86400.0) THEN
  IERROR=4
  RETURN
END IF

!compare IYEAR to IYEARC
IF (IYEAR==IYEARC) THEN
  GOTO 6000                                    !same year
ELSEIF (IYEAR==IYEARC+1) THEN
  GOTO 4000                                    !next year
END IF
```

```
!for completely different year
!fill the DAT vector, DATS
ISTART = 0

DO I=1,3
  CALL DATGEN(DATVECT,IERROR1)
  IF (IERROR1 /= 0) THEN
    IERROR = 5
          RETURN
  END IF
  DO J=1,365
    DATS(ISTART+J) = DATVECT(J)
  END DO
  ISTART = ISTART + 365
END DO

!calculate spline for IDAY-1
OLDVALUE=(DATS((IDAY-1)+365) + DATS((IDAY-1)+364))/2

CALL SPLINE(DATS,IDAY-1,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
  IERROR = 6
  RETURN
END IF

!call splint to obtain value for OLDVALUE
CALL SPLINT(172800.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

OLDVALUE = RESULT2

!calculate spline for IDAY
CALL SPLINE(DATS,IDAY,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
  IERROR = 6
  RETURN
END IF

!call SPLINT for TIME
CALL SPLINT(TIME+86400.0,RESULT2,IERROR3)
```

```
IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

!set value for RESULT1
RESULT1 = RESULT2

!set IYEARC and IDAYC
IYEARC = IYEAR
IDAYC = IDAY

RETURN

!next year
4000 CONTINUE

!generate a new set of DATs
CALL DATGEN(DATVECT,IERROR1)

IF (IERROR1 /= 0) THEN
  IERROR = 5
  RETURN
END IF

DO I=1,365
  DATS(I) = DATS(I+365)
  DATS(I+365) = DATS(I+730)
  DATS(I+730) = DATVECT(I)
END DO

!check if it's the next day
IF (IDAY==1 .AND. IDAYC==365) THEN
  GOTO 5000                                                    !next day
END IF

!totally different day
!calculate spline for IDAY-1
OLDVALUE=(DATS((IDAY-1)+365) + DATS((IDAY-1)+364))/2

CALL SPLINE(DATS,IDAY-1,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
  IERROR = 6
  RETURN
```

```
END IF

!call SPLINT to obtain value for OLDVALUE
CALL SPLINT(172800.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

OLDVALUE = RESULT2

!calculate spline for IDAY
CALL SPLINE(DATS,IDAY,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
  IERROR = 6
  RETURN
END IF

!call SPLINT for TIME
CALL SPLINT(TIME+86400.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

!set value for RESULT1
RESULT1 = RESULT2

!set IYEARC and IDAYC
IYEARC = IYEAR
IDAYC = IDAY

RETURN

!next day
5000 CONTINUE

!call  SPLINT to obtain value for OLDVALUE
CALL SPLINT(172800.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
```

```
END IF

OLDVALUE = RESULT2

!calculate spline for IDAY
CALL SPLINE(DATS,IDAY,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
  IERROR = 6
  RETURN
END IF

!call SPLINT for TIME
CALL SPLINT(TIME+86400.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

!set value for RESULT1
RESULT1 = RESULT2

!set IYEARC and IDAYC
IYEARC = IYEAR
IDAYC = IDAY

RETURN

!same year
6000 CONTINUE

!compare IDAY to IDAYC
IF (IDAY == IDAYC) THEN
  GOTO 8000                                          !same day
ELSEIF (IDAY == IDAYC+1) THEN
  GOTO 7000                                          !next day
END IF

!totally different day
!calculate spline for day before IDAY
OLDVALUE=(DATS((IDAY-1)+365) + DATS((IDAY-1)+364))/2

CALL SPLINE(DATS,IDAY-1,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
```

```
  IERROR = 6
  RETURN
END IF

!call SPLINT to obtain value for OLDVALUE
CALL SPLINT(172800.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

OLDVALUE = RESULT2

!calculate spline for IDAY
CALL SPLINE(DATS,IDAY,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
  IERROR = 6
  RETURN
END IF

!call SPLINT for TIME
CALL SPLINT(TIME+86400.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

!set value for RESULT1
RESULT1 = RESULT2

!set IYEARC and IDAYC
IYEARC = IYEAR
IDAYC = IDAY

RETURN

!next day
7000 CONTINUE

!call  SPLINT to obtain value for OLDVALUE
CALL SPLINT(172800.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
```

```
 IERROR = 7
 RETURN
END IF

OLDVALUE = RESULT2

!calculate spline for IDAY
CALL SPLINE(DATS,IDAY,OLDVALUE,IERROR2)

IF (IERROR2 /= 0) THEN
 IERROR = 6
 RETURN
END IF

!call SPLINT for TIME
CALL SPLINT(TIME+86400.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
 IERROR = 7
 RETURN
END IF

!set value for RESULT1
RESULT1 = RESULT2

!set IYEARC and IDAYC
IYEARC = IYEAR
IDAYC = IDAY

RETURN

!same day
8000 CONTINUE

!call SPLINT for TIME
CALL SPLINT(TIME+86400.0,RESULT2,IERROR3)

IF (IERROR3 /= 0) THEN
 IERROR = 7
 RETURN
END IF

!set value for RESULT1
RESULT1 = RESULT2

!set IYEARC and IDAYC
```

```
IYEARC = IYEAR
IDAYC = IDAY

RETURN

END SUBROUTINE TEMPERAT
!#####                    END OF SUBTROUTINE TEMPERAT                    #####

!#####                    START OF SUBTROUTINE DATGEN                    #####
SUBROUTINE DATGEN(DATVECT,IERROR1)
!suberoutine to generate a vector of 365 Daily Average Temperatures

!REAL*4 variables - input parameters
REAL*4:: A0AV, A1AV, A2AV, A3AV, A4AV          !ave. param. values for DAT frequency
sinusoids
REAL*4:: A0SD, A1SD, A2SD, A3SD, A4SD          !stdev. param values for DAT
frequency sinusoids
REAL*4:: B0AV, B1AV, B2AV                      !ave. param. values for DAT beat sinusoid
REAL*4:: B0SD, B1SD, B2SD                      !stdev. param. values for DAT beat sinusoid
REAL*4:: D0AV, D1AV, D2AV, D3AV, D4AV, D5AV    !ave. param. values for DAT noise
REAL*4:: D0SD, D1SD, D2SD, D3SD, D4SD, D5SD    !stdev. param. values for DAT
noise
REAL*4:: RHO                                   !correlation coefficient between A0 and A1

!dimension REAL*4 variables
REAL*4, DIMENSION(365):: DATVECT               !vector of 365 DATs - 1 year
REAL*4, DIMENSION(365):: DATTIME               !vector of days in fractions of a year
REAL*4, DIMENSION(180):: MAGAVPL               !magnitude averages polynomial
REAL*4, DIMENSION(180):: MAGSDPL               !magnitude standard deviation
polynomial
REAL*4, DIMENSION(180):: MAG                   !magnitude of the noise term (uses
MAGAVPL and MAGSDPL)
REAL*4, DIMENSION(180):: ANGLE                 !phase angle of the noise term

!REAL*4 variables - non input
REAL*4:: A0,A1,A2,A3,A4
REAL*4:: B0,B1,B2
REAL*4:: FREQ0, FREQ1, FREQ2                   !freq=0, freq=1, and freq=2 parts of DAT
signal
REAL*4:: BEAT                                  !the beat sinusoid part of the DAT signal
REAL*4:: NOISE                                 !noise portion of DAT signal (uses MAG and
ANGLE, DATTIME and FREQ)
REAL*4:: RANDTEM, MAGSDEVTEM, ANGLETEM         !functions
REAL*4:: FREQ                                  !working variable for frequency calculations
REAL*4:: TEMP1, TEMP2                          !working variables
```

```
REAL*4, PARAMETER:: PI = 3.14159265358979

!INTEGER*4 variables
INTEGER*4:: IERROR1                          !error code 0=none detected
INTEGER*4:: I,J

!common blicks
COMMON /BLOCK01/
A0AV,A0SD,A1AV,A1SD,A2AV,A2SD,A3AV,A3SD,A4AV,A4SD,B0AV,B0SD,B1AV,B1SD,B2
AV,B2SD, &

D0AV,D1AV,D2AV,D3AV,D4AV,D5AV,D0SD,D1SD,D2SD,D3SD,D4SD,D5SD,RHO

COMMON /BLOCK02/    DATTIME, MAGAVPL, MAGSDPL

IERROR1=0

DO J=1,180
  MAG(J) = ABS(MAGAVPL(J) + MAGSDEVTEM(1,2.5)*MAGSDPL(J))
  ANGLE(J) = (RANDTEM(1)-0.5) * 2*PI
END DO

B0 = B0AV + MAGSDEVTEM(1,2.5)*B0SD
B1 = B1AV + MAGSDEVTEM(1,2.5)*B1SD
B2 = B2AV + MAGSDEVTEM(1,2.5)*B2SD
B2 = ANGLETEM(B2)

TEMP1=MAGSDEVTEM(1,2.5)
TEMP2=MAGSDEVTEM(1,2.5)

A0 = A0AV + TEMP1*A0SD

A1 = A1AV + (TEMP1*RHO + TEMP2*(1-RHO**2)**0.5)*A1SD
A2 = A2AV + MAGSDEVTEM(1,2.5)*A2SD
A2 = ANGLETEM(A2)

A3 = A3AV + MAGSDEVTEM(1,2.5)*A3SD
A4 = A4AV + MAGSDEVTEM(1,2.5)*A4SD
A4 = ANGLETEM(A4)

DO I=1,365
  NOISE = 0.0
  DO J=1,180
    FREQ = J + 2
    NOISE = NOISE + MAG(J)*SIN(FREQ * DATTIME(I)*2*PI + ANGLE(J))
  END DO
```

```
  BEAT = B0 + B1*SIN(DATTIME(I)*2*PI + B2)

  FREQ0 = A0

  FREQ1 = A1*SIN(DATTIME(I)*2*PI + A2)

  FREQ2 = A3*SIN(2*DATTIME(I)*2*PI + A4)

  DATVECT(I) = FREQ0 + FREQ1 + FREQ2 + BEAT*NOISE
END DO

RETURN

END SUBROUTINE DATGEN
!#####                    END OF SUBTROUTINE DATGEN                    #####

!#####                     START FUNCTION ANGLETEM                     #####
!# It calculates the condition angle based on THETA - used by temperature subroutine
!# Inputs : THETA
REAL*4 FUNCTION ANGLETEM(THETA)

IMPLICIT NONE

REAL*4:: THETA
REAL*4, PARAMETER:: PI = 3.14159265358979

 DO WHILE (ABS(THETA) > PI)
  THETA = THETA * (1 - ABS(THETA)**(-1)*2.0*PI)
END DO
ANGLETEM = THETA

END FUNCTION ANGLETEM
!#####                    END FUNCTION CONDITIONANGLE                  #####

!#####                    START OF MAGSDEVTEM FUNCTION                 #####
REAL*4 FUNCTION MAGSDEVTEM(MIDUM,BOUND)
! Function returns normally distributed numbers but removes outliers (i.e. values +/-
2.5stds).  This function is based on mrand.m written by L. Parrott and R. Kok; its main
purpose is for use in daily average temperature modeling.

! Modified so that the bounds with which to identify outliers are given as an argument. YC
Sun Mar 26 04
! copy used by temperature subroutines
```

```
INTEGER*4:: MIDUM
REAL*4::BOUND
REAL*4:: GASDEVTEM

MAGSDEVTEM = BOUND+1.0
 DO WHILE(ABS(MAGSDEVTEM) > BOUND)
  MAGSDEVTEM = GASDEVTEM(MIDUM)
END DO

!MAGSDEV = 6.0
! DO WHILE(ABS(MAGSDEV) > 5.0)
!  MAGSDEV = GASDEV(MIDUM)
!END DO

END FUNCTION MAGSDEVTEM
!#####                    END OF MAGSDEVTEM FUNCTION                    #####

!######                    START FUNCTION GASDEVTEM                    #####
!# Returns a normally distributed deviate with zero mean and unit variance, using ran(idum)
as the source of uniform
!# deviates.  Reprinted by Lael from Numerical Recipes in C 2 Ed., p. 289
! copy used by temperature subroutines
!# Inputs : GIDUM
REAL*4 FUNCTION GASDEVTEM(GIDUM)

INTEGER*4:: GIDUM

INTEGER*4:: ISET

REAL*4:: GSET,FAC,RSQ,V1,V2

REAL*4:: RANDTEM
SAVE ISET,GSET
 DATA ISET/0/

IF (ISET == 0) THEN
  RSQ = 1.
  DO WHILE(RSQ >= 1. .OR. RSQ == 0.)
    V1 = 2. * RANDTEM(GIDUM) - 1
    V2 = 2. * RANDTEM(GIDUM) - 1
    RSQ = V1**2 + V2**2
  END DO
  FAC = SQRT(-2.*LOG(RSQ)/RSQ)
  GSET = V1 * FAC
  GASDEVTEM = V2 * FAC
  ISET = 1
```

```
ELSE
  GASDEVTEM = GSET
  ISET = 0
END IF

END FUNCTION GASDEVTEM
!#####                    END FUNCTION GASDEV                    #####

!#####              START OF FUNCTION PROGRAM RANDTEM              #####
REAL*4 FUNCTION RANDTEM(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with
! Bays-Durham shuffle and added safeguards. Returns a uniform random deviate between
0.0 and 1.0 (exlusive of the
! endpoint values). Call with idum a negative integer to initialize; thereafter, do not alter
idum between successive
! deviates in sequence. RNMX should approximate the largest floating value that is less
than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
```

```
IF (IDUM <= 0) THEN                    !initialize
  IDUM=MAX(-IDUM,1)                    !be sure to prevent IDUM=0
  IDUM2=IDUM
  DO J=NTAB+8,1,-1                     !load the shuffle table (after 8 warm-ups)
    K=IDUM/IQ1
    IDUM=IA1*(IDUM-K*IQ1)-K*IR1
    IF (IDUM < 0) IDUM=IDUM+IM1
    IF (J <= NTAB) IV(J)=IDUM
  END DO
  IY=IV(1)
END IF

K=IDUM/IQ1                             !start here when not initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1            !compute IDUM=MOD(IA1*IDUM,IM1)
without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2          !compute IDUM2=MOD(IA2*IDUM2,IM2),
likewise
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                            !will be in the range 1:NTAB
IY=IV(J)-IDUM2                         !here IDUM is shuffled, IDUM and IDUM2 are
combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
RANDTEM=MIN(AM*IY,RNMX)                !because users don't expect endpoint
values
RETURN

END FUNCTION RANDTEM
!#####          END OF FUNCTION PROGRAM RANDTEM          #####

!#####          START SUBROUTINE SPLINE          #####
SUBROUTINE SPLINE(DATS,IDAY,OLDVALUE,IERROR2)
!From Numerical Recipes in Fortran, 2nd Edition. p. 109
!Given arrays x(1:n) and y(1:n) containing a tabulated function, i.e. y_i=f(x_i), with
!x_1 < x_2 < ... <x_n, and given points yp1 and ypn for the first derivative of the
interpolating
!function at points 1 and n, respectively, this routine returns an array y2(1:n) of length n
!which contains the second derivatives fo the interpolating function at the tabulated points
x_i.
!If yp1 and ypn are equal to 1X10^30 or larger, the routine is signaled to set the
corresponding
!boundary condition for a natural spline, with zero second derivative on that boundary.
!Parameter: NMAX is the largest anticipated value of n. - REMOVED - EDITED BY TRL
!
```

```
!INPUTS: DATS,IDAY,OLDVALUE
!OUTPUTS: YA, Y2, IERROR2

!integer parameter
INTEGER*4, PARAMETER:: N=13

!dimension real variables
REAL*4, DIMENSION(1095):: DATS                          !vector of three years of
DATs
REAL*4, DIMENSION(13):: XA, YA, Y2                      !time array, temp array,
spline array
REAL*4, DIMENSION(13):: U                               !working variable

!real variables
REAL*4:: OLDVALUE                                       !the temp. at 'midnight' for
IDAY-1
REAL*4:: AMPUP, AMPDOWN                                 !amplitude parameters for
SPLINE
REAL*4:: YP1, YPN                                       !boundary conditions
REAL*4:: P, QN, SIG, UN                                 !working variables
REAL*8:: DAT_YESTERDAY, DAT_TODAY, DAT_TOMORROW         !daily average temps:
IDAY-1,IDAY,IDAY+1
REAL*8:: MIN_YESTERDAY, MIN_TODAY, MIN_TOMORROW         !temps at 5h: IDAY-
1,IDAY,IDAY+1
REAL*8:: MAX_YESTERDAY, MAX_TODAY, MAX_TOMORROW         !temps at 14h: IDAY-
1,IDAY,IDAY+1
REAL*8:: DAT_DAY1, DAT_DAY5                             !daily average temps: IDAY-
2,IDAY+2

!integer variables
INTEGER*4:: IDAY                                        !day input from call
INTEGER*4:: IERROR2                                     !error code, 0=none detected
INTEGER*4:: I, K                                        !working variables

COMMON /BLOCK03/    AMPUP,AMPDOWN

COMMON /BLOCK04/    XA,YA,Y2                            !XA used by TEMPERAT,
SPLINE, SPLINT
                                                        !YA,Y2 used by SPLINE,
                                                    SPLINT

IERROR2=0

!calculate values for YP1 and YPN
YP1 = 0.0
YPN = 0.0
```

```
!set DATS for the five days
DAT_DAY1 = DATS(IDAY+363)
DAT_YESTERDAY = DATS(IDAY+364)
DAT_TODAY = DATS(IDAY+365)
DAT_TOMORROW = DATS(IDAY+366)
DAT_DAY5 = DATS(IDAY+367)

!calculate temps at 05:00 for IDAY-1, IDAY, and IDAY+1
MIN_YESTERDAY = ((DAT_DAY1 + DAT_YESTERDAY)/2 + 273)*AMPDOWN - 273
MIN_TODAY = ((DAT_YESTERDAY + DAT_TODAY)/2 + 273)*AMPDOWN - 273
MIN_TOMORROW = ((DAT_TODAY + DAT_TOMORROW)/2 + 273)*AMPDOWN - 273

!calculate temps at 14:00 for IDAY-1, IDAY, and IDAY+1
MAX_YESTERDAY = (DAT_YESTERDAY + 273)*AMPUP - 273
MAX_TODAY = (DAT_TODAY + 273)*AMPUP - 273
MAX_TOMORROW = (DAT_TOMORROW + 273)*AMPUP - 273

!place those temperatures in YA vector
YA(1) = MIN_YESTERDAY
YA(2) = DAT_YESTERDAY
YA(3) = MAX_YESTERDAY
YA(4) = DAT_YESTERDAY
YA(5) = OLDVALUE
YA(6) = MIN_TODAY
YA(7) = DAT_TODAY
YA(8) = MAX_TODAY
YA(9) = DAT_TODAY
YA(10) = MIN_TOMORROW
YA(11) = DAT_TOMORROW
YA(12) = MAX_TOMORROW
YA(13) = DAT_TOMORROW


IF (YP1 > 99E30) THEN                                   !The lower boundary condition
is set either to be "natural"
  Y2(1) = 0.0
  U(1) = 0.0
ELSE                                                    !or else to have a specified
first derivative.
  Y2(1) = -0.5
  U(1) = (3./(XA(2) - XA(1))) * ((YA(2) - YA(1))/(XA(2) - XA(1)) - YP1)
END IF

DO I=2,N-1                                              !This is the decomposition
loop of the tridiagonal
```

```
  SIG = (XA(I) - XA(I-1)) / (XA(I+1) - XA(I-1))                    !algorithm.  Y2 and U are used
for temporary
  P = SIG*Y2(I-1) + 2                                              !storage of the decomposed
factors.
  Y2(I) = (SIG - 1.)/P
  U(I) = (6.*((YA(I+1) - YA(I))/(XA(I+1) - XA(I)) - (YA(I) - YA(I-1))/(XA(I) - XA(I-1))) /
(XA(I+1) - XA(I-1)) - SIG*U(I-1))/P
END DO

IF (YPN > 99E30) THEN                                              !The upper boundary
condition is set either to be "natural"
  QN = 0.0
  UN = 0.0
ELSE                                                               !or else to have a specified
first derivative.
  QN = 0.5
  UN = (3./(XA(N) - XA(N-1))) * (YPN - (YA(N) - YA(N-1))/(XA(N) - XA(N-1)))
END IF

Y2(N) = (UN - QN*U(N-1)) / (QN*Y2(N-1) + 1.)

DO K=N-1,1,-1                                                      !This is the backsubstitution
loop of the tridiagonal algorithm.
  Y2(K) = Y2(K) * Y2(K+1) + U(K)
END DO

RETURN

END SUBROUTINE SPLINE
!#####                          END SUBROUTINE SPLINE                          #####

!#####                         START SUBROUTINE SPLINT                         #####
SUBROUTINE SPLINT(TIME,RESULT2,IERROR3)
!From Numerical Recipes in Fortran, 2nd Edition.  p. 110
!Given the arrays xa(1:n) and ya(1:n) of length n, which tabulate a function (the the xa_i's in
order), and given the array y2a(1:n),m which is the output from SPLINE above, and given a
valueof x, this routine returns a cubic spline interpolated value y.
!
!INPUTS: TIME, YA, Y2
!OUTPUTS: RESULT2, IERROR3

!integer parameter
INTEGER, PARAMETER:: N=13
```

```
!dimensioned real variables
REAL*4, DIMENSION(N):: XA, YA, Y2                    !time array, temp array,
spline array

!real variables
REAL*4:: TIME                                        !time of day in seconds +
86400.0
REAL*4:: RESULT2                                     !temperature in deg. C -
returned to TEMPERAT
REAL*4:: A, B, H

!integer variables
INTEGER*4:: IERROR3                                  !error code - 0=none
detected
INTEGER*4:: K, KHI, KLO                              !working variables

!common block
COMMON /BLOCK04/    XA,YA,Y2                         !XA used by TEMPERAT,
SPLINE, SPLINT                                       !YA,Y2 used by SPLINE,
SPLINT

IERROR3=0


KLO = 1                                              !We will find the right place
in the table by means of bisection
KHI = N                                              !This is optimal if sequential
calls to this routine are at random values of x.  If sequential calls in order, and closely
spaced, one would do better to store previous values of  KLO and KHI and test if they
remain appropriate on the next call.


1 IF (KHI-KLO > 1) THEN
  K = (KHI + KLO)/2
  IF (XA(K) > TIME) THEN
    KHI = K
  ELSE
    KLO = K
  END IF
GOTO 1
END IF                                               !KLO and KHI now bracket
the input value of x.

H = XA(KHI) - XA(KLO)

IF (H == 0) THEN                                     !The XA's must be distinct
```

```
 PRINT *
 PRINT *, "Message from subroutine SPLINT"
 PRINT *, "Bad XA input in SPLINT"
 IERROR3=1
 RETURN
END IF

A = (XA(KHI) - TIME) / H
B = (TIME - XA(KLO)) / H
RESULT2 = A*YA(KLO) + B*YA(KHI) + ((A**3 - A)*Y2(KLO) + (B**3 -
B)*Y2(KHI))*(H**2)/6


RETURN

END SUBROUTINE SPLINT
!#####                    END SUBROUTINE SPLINT                    #####

!*******************************************************************************

!#####                    START SUBROUTINE RADIAT                    #####
SUBROUTINE RADIAT(ICASE,IYEAR,IDAY,TIME,IRAD,RI,IERROR,RADFILE)
!Subroutine to obtain values for  a flag for whether it is daytime IRAD and radiation
intensity (watt per sq. meter) RI for a given IYEAR, IDAY, and TIME.  Also included as an
input is a case switch (ICASE) to determine whether it is an initialization call or normal
operation and the name (or an indicator) for which file contains the parameter values for
the subroutine.  Outputs also include an error code (IERROR).

IMPLICIT NONE
!declare dimensioned REAL*4 variables
REAL*4, DIMENSION(365):: DORTIME                !time array used for sinusoids
(1/365 - 365/365)
REAL*4, DIMENSION(183):: MAGAVPL                !residual magnitude averages
array
REAL*4, DIMENSION(183):: MAGSDPL                !residual magnitude standards
deviation array
REAL*4, DIMENSION(365):: DPNAVPL                !daily positive noise averages
array
REAL*4, DIMENSION(365):: DPNSDPL                !daily positive noise standard
deviations array
REAL*4, DIMENSION(365):: DNNAVPL                !daily negative noise averages
array
REAL*4, DIMENSION(365):: DNNSDPL                !daily negative noise standard
deviations array
REAL*4, DIMENSION(365):: DORS                   !vector of Daily Overall
Radiation values for  a year
```

```
REAL*4, DIMENSION(8760):: YATTN                    !hourly attenuation factors
for a year
REAL*4, DIMENSION(24*6+1):: DRIARRAY               !radiation intensities for a
day at ten minute intervals

!declare REAL*4 variables
REAL*4, PARAMETER:: PI = 3.14159265358979          !needed to treat LAT
REAL*4, PARAMETER:: EPS = 1E-6                     !needed to treat LAT
REAL*4:: LAT                                       !the latitude of ecosystem
location
REAL*4:: MINDOR                                    !minimum DOR for the
location
REAL*4:: A0AV, A1AV, A2AV, A3AV, A4AV              !ave. param. values for DOR
frequency sinusoids (0,1,2 freqs)
REAL*4:: A0SD, A1SD, A2SD, A3SD, A4SD              !stdev. param values for DOR
frequency sinusoids (0,1,2 freqs)
REAL*4:: B0AV, B1AV, B2AV                          !ave. param. values for DOR
beat sinusoid
REAL*4:: B0SD, B1SD, B2SD                          !stdev. param. values for DOR
beat sinusoid
REAL*4:: D0AV, D1AV, D2AV, D3AV, D4AV, D5AV        !ave. param. values for DOR
noise - residual
REAL*4:: D0SD, D1SD, D2SD, D3SD, D4SD, D5SD        !stdev. param. values for DOR
noise - residual
REAL*4:: E0AV, E1AV, E2AV, E3AV, E4AV, E5AV        !ave. param. values for DOR
noise - postive
REAL*4:: E0SD, E1SD, E2SD, E3SD, E4SD, E5SD        !stdev. param. values for DOR
noise - positive
REAL*4:: F0AV, F1AV, F2AV, F3AV, F4AV, F5AV        !ave. param. values for DOR
noise - negative
REAL*4:: F0SD, F1SD, F2SD, F3SD, F4SD, F5SD        !stdev. param. values for DOR
noise - negative
REAL*4:: RHO, RHO2                                 !correlation coefficient: A0
and A1, B0 and B1
REAL*4:: PNAV, PNSD                                !positive noise occurence
percentage ave. and s.dev.

REAL*4:: SIGN                                      !sign of input latitude (north
or south hemisphere)
REAL*4:: TIME                                      !time of day, values
0.0~86400.0
REAL*4:: RI                                        !radiation intensity at a given
time (returned to weather subroutine)
REAL*4:: DARRAYDT                                  !delta time in DRIARRAY (?)
REAL*4:: DPTTIME                                   !the array element on which
TIME falls
```

```
REAL*4:: TEMP                                   !temporary variable

!declare INTEGER*4 variables
INTEGER*4, DIMENSION(24*6+1):: DAYTIME          !half daytime, used to see if
TIME is in daylight hours
INTEGER*4:: ICASE                               !operation indicator:
0=initialization, 1=normal operation
INTEGER*4:: IYEAR                               !year of call from simulation,
values 1~10,000
INTEGER*4:: IDAY                                !day of call from simulation,
values 1~365
INTEGER*4:: IRAD                                !indicates day or night:
0=night, 1=daylight
INTEGER*4:: IERROR                              !error code: 0=no error
detected, 1=ICASE out of range, 2=IYEAR out of range, 3=IDAY out of range, 4=TIME out
of range, 5=error from DORGEN ...
INTEGER*4:: IERROR1                             !error code from DORGEN
INTEGER*4:: IERROR2                             !error code from ATTNYEAR
INTEGER*4:: IERROR3                             !error code from DRIGEN
INTEGER*4:: ISEED                               !seed for the random number
generator
INTEGER*4:: IYEARC                              !'current' year (year of last
call)
INTEGER*4:: IDAYC                               !'current' day (day of last
call)
INTEGER*4:: I                                   !counter
INTEGER*4:: FREQ                                !temporary variable used for
frequency calculations
INTEGER*4:: DPT                                 !day array index (?)

!declare CHARACTER variables
CHARACTER (LEN=14):: RADFILE                    !name of the input file for
the radiation parameters
CHARACTER (LEN=10):: TEXTLINE                   !dummy input line

!declare functions
REAL*4:: RANDRAD                                !random number generator
for radiation

!common blocks
!radiation parameters and variables used by DORGEN
COMMON /BLOCK05/
A0AV,A0SD,A1AV,A1SD,A2AV,A2SD,A3AV,A3SD,A4AV,A4SD,B0AV,B0SD,B1AV,B1SD,B2
AV,B2SD, MINDOR,PNAV,PNSD
```

```
COMMON /BLOCK06/
DORTIME,MAGAVPL,MAGSDPL,DPNAVPL,DPNSDPL,DNNAVPL,DNNSDPL

!parameters used by subroutines associated with DRIGEN subroutine
COMMON /BLOCK07/    DORS, YATTN, LAT

COMMON /BLOCK08/    DAYTIME

COMMON /BLOCK11/    DRIARRAY

!set initial value of IERROR
IERROR = 0

!check ICASE value
IF (ICASE /= 0 .AND. ICASE /= 1) THEN
  IERROR = 1
  RETURN
END IF

!decide initialization or normal operation
IF (ICASE == 1) THEN
  GOTO 1000                                                    !normal operation
END IF

!initialization routine
!initialization part 1 - read in data
OPEN (UNIT=1,FILE=RADFILE)

100 FORMAT (1A10)
101 FORMAT (I10)
102 FORMAT (F10.2)
103 FORMAT (5F15.5)
104 FORMAT (3F15.5)
105 FORMAT (E15.5)
106 FORMAT (F10.5)
107 FORMAT (2F10.5)

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,101) ISEED

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,102) LAT
```

```
READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,102) MINDOR

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,103) A0AV, A1AV, A2AV, A3AV, A4AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,103) A0SD, A1SD, A2SD, A3SD, A4SD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,104) B0AV, B1AV, B2AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,104) B0SD, B1SD, B2SD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,105) D0AV
READ (1,105) D1AV
READ (1,105) D2AV
READ (1,105) D3AV
READ (1,105) D4AV
READ (1,105) D5AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,105) D0SD
READ (1,105) D1SD
READ (1,105) D2SD
READ (1,105) D3SD
READ (1,105) D4SD
READ (1,105) D5SD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,106) RHO

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,106) RHO2
```

```
READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,107) PNAV, PNSD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,105) E0AV
READ (1,105) E1AV
READ (1,105) E2AV
READ (1,105) E3AV
READ (1,105) E4AV
READ (1,105) E5AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,105) E0SD
READ (1,105) E1SD
READ (1,105) E2SD
READ (1,105) E3SD
READ (1,105) E4SD
READ (1,105) E5SD

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,105) F0AV
READ (1,105) F1AV
READ (1,105) F2AV
READ (1,105) F3AV
READ (1,105) F4AV
READ (1,105) F5AV

READ (1,100) TEXTLINE
READ (1,100) TEXTLINE
READ (1,105) F0SD
READ (1,105) F1SD
READ (1,105) F2SD
READ (1,105) F3SD
READ (1,105) F4SD
READ (1,105) F5SD

CLOSE (UNIT=1)


!initialization part 2 - initialization of variables for DORGEN
!initialize random number generator
TEMP = RANDRAD(ISEED)
```

```
!fill DORTIME vector
DO I=1,365
  DORTIME(I) = FLOAT(I)/365
END DO

!calculate polynomials for residual noise magnitude: average and standard deviation
DO I=1,183
  FREQ = I - 1
  MAGAVPL(I) = D0AV + D1AV*FREQ + D2AV*FREQ**2 + D3AV*FREQ**3 +
D4AV*FREQ**4 + D5AV*FREQ**5
  MAGSDPL(I) = D0SD + D1SD*FREQ + D2SD*FREQ**2 + D3SD*FREQ**3 +
D4SD*FREQ**4 + D5SD*FREQ**5
END DO

!calculate polynomials for daily positive & negative noise magnitudes: average and standard
deviation
DO I=1,365
  DPNAVPL(I) = E0AV + E1AV*I + E2AV*I**2 + E3AV*I**3 + E4AV*I**4 + E5AV*I**5
  DPNSDPL(I) = E0SD + E1SD*I + E2SD*I**2 + E3SD*I**3 + E4SD*I**4 + E5SD*I**5
  DNNAVPL(I) = F0AV + F1AV*I + F2AV*I**2 + F3AV*I**3 + F4AV*I**4 + F5AV*I**5
  DNNSDPL(I) = F0SD + F1SD*I + F2SD*I**2 + F3SD*I**3 + F4SD*I**4 + F5SD*I**5
END DO

!initialization part 3 - adjust LAT to get rid of the invalid TAN(PI/2)
LAT=(LAT*PI/180.0)                                    !translate LAT value from
degrees to radians
SIGN=LAT/ABS(LAT)
LAT=SIGN*MIN(ABS(LAT),PI/2.0-EPS)                     !adjusted latitude, to avoid
infinite values for LAT = 90 or -90 degree

!initialization part 4 - set initial values for main variables
IF (IDAY == 1) THEN
  IYEARC = IYEAR - 1
  IDAYC = 365
ELSE
  IYEARC = IYEAR
  IDAYC = IDAY - 1
END IF

!fill up Daily Overall Radiation vector - DORS
CALL DORGEN(DORS,IERROR1)
IF (IERROR1 /= 0) THEN
  IERROR = 5
  RETURN
END IF
```

```
!fill attenuation stream variable - YATTN
!CALL ATTNYEAR(YATTN,IERROR2)
!IF (IERROR2 /= 0) THEN
!  IERROR = 6
!  RETURN
!END IF

!fill day variables
CALL DRIGEN(IDAYC,IERROR3)
IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

RETURN

!normal operation
1000 CONTINUE

!test range of IYEAR
IF (IYEAR<0 .OR. IYEAR>10000) THEN
  IERROR=2
  RETURN
END IF

!test range of IDAY
IF (IDAY<1 .OR. IDAY>365) THEN
  IERROR=3
  RETURN
END IF

!test range of TIME
IF (TIME<0.0 .OR. TIME>86400.0) THEN
  IERROR=4
  RETURN
END IF

!compare IYEAR to IYEARC
IF (IYEAR==IYEARC) THEN
  GOTO 2000                                          !same year
END IF

!for different year
!fill up Daily Overall Radiation vector - DORS
CALL DORGEN(DORS,IERROR1)
```

```
IF (IERROR1 /= 0) THEN
  IERROR = 5
  RETURN
END IF

!fill attenuation stream variable - YATTN
!CALL ATTNYEAR(YATTN,IERROR2)
!IF (IERROR2 /= 0) THEN
!  IERROR = 6
!  RETURN
!END IF

!fill day variables
CALL DRIGEN(IDAY,IERROR3)
IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

!calculate return value
!adjust time to test against spacing of given values in RI array - currently 10 minutes.
DARRAYDT = 600.0                                        !10 minutes
DPTTIME = 1.0 + TIME/DARRAYDT
DPT = FLOOR(DPTTIME)

!check if there's any sunlight, return RI, do interpolation if requires
IF (DAYTIME(DPT) == 0.0) THEN
  IRAD = 0
  RI = 0.0
ELSE
  IF (DPTTIME-DPT == 0.0) THEN                          !no need to interpolate
    RI = DRIARRAY(DPT)
  ELSE
    RI = DRIARRAY(DPT) + (DPTTIME-DPT)*(DRIARRAY(DPT+1) - DRIARRAY(DPT))
  END IF
  IRAD = 1
END IF

!reset IYEARC and IDAYC
IYEARC = IYEAR
IDAYC = IDAY

RETURN

!same year
2000 CONTINUE
```

```
!compare IDAY to IDAYC
IF (IDAY == IDAYC) THEN
  GOTO 3000                                                    !same day
END IF

!different day
!fill day variables
CALL DRIGEN(IDAY,IERROR3)
IF (IERROR3 /= 0) THEN
  IERROR = 7
  RETURN
END IF

!calculate return value
!adjust time to test against spacing of given values in DRI array - currently 10 minutes.
DARRAYDT = 600.0                                               !10 minutes
DPTTIME = 1.0 + TIME/DARRAYDT
DPT = FLOOR(DPTTIME)

!check if there's any sunlight, return DRI, do interpolation if requires
IF (DAYTIME(DPT) == 0.0) THEN
  IRAD = 0
  RI = 0.0
ELSE
  IF (DPTTIME-DPT == 0.0) THEN                                 !no need to interpolate
    RI = DRIARRAY(DPT)
  ELSE
    RI = DRIARRAY(DPT) + (DPTTIME-DPT)*(DRIARRAY(DPT+1) - DRIARRAY(DPT))
  END IF
  IRAD = 1
END IF

!reset IDAYC
IDAYC = IDAY

RETURN

!same day
3000 CONTINUE

!calculate return value
!adjust time to test against spacing of given values in RI array - currently 10 minutes.
DARRAYDT = 600.0                                               !10 minutes
DPTTIME = 1.0 + TIME/DARRAYDT
DPT = FLOOR(DPTTIME)
```

```
!check if there's any sunlight, return RI, do interpolation if requires
IF (DAYTIME(DPT) == 0.0) THEN
 IRAD = 0
 RI = 0.0
ELSE
 IF (DPTTIME-DPT == 0.0) THEN                    !no need to interpolate
  RI = DRIARRAY(DPT)
 ELSE
  RI = DRIARRAY(DPT) + (DPTTIME-DPT)*(DRIARRAY(DPT+1) - DRIARRAY(DPT))
 END IF
 IRAD = 1
END IF

RETURN

END SUBROUTINE RADIAT
!#####                    END SUBROUTINE RADIAT                    #####

!#####                  START OF SUBTROUTINE DORGEN                #####
SUBROUTINE DORGEN(DORS,IERROR1)
!Subroutine to generate one year's worth of Daily Overall Radiation values.
!Written by YCS, adapted by TRL June 1, 2005.

IMPLICIT NONE
!REAL*4 variables - input parameters (common block)
REAL*4:: MINDOR                                 !minimum DOR for the
location
REAL*4:: A0AV, A1AV, A2AV, A3AV, A4AV           !ave. param. values for DOR
frequency sinusoids (0,1,2 freqs)
REAL*4:: A0SD, A1SD, A2SD, A3SD, A4SD           !stdev. param values for DOR
frequency sinusoids (0,1,2 freqs)
REAL*4:: B0AV, B1AV, B2AV                       !ave. param. values for DOR
beat sinusoid
REAL*4:: B0SD, B1SD, B2SD                       !stdev. param. values for DOR
beat sinusoid
REAL*4:: D0AV, D1AV, D2AV, D3AV, D4AV, D5AV     !ave. param. values for DOR
noise - residual
REAL*4:: D0SD, D1SD, D2SD, D3SD, D4SD, D5SD     !stdev. param. values for DOR
noise - residual
REAL*4:: E0AV, E1AV, E2AV, E3AV, E4AV, E5AV     !ave. param. values for DOR
noise - postive
REAL*4:: E0SD, E1SD, E2SD, E3SD, E4SD, E5SD     !stdev. param. values for DOR
noise - positive
REAL*4:: F0AV, F1AV, F2AV, F3AV, F4AV, F5AV     !ave. param. values for DOR
noise - negative
```

```
REAL*4:: F0SD, F1SD, F2SD, F3SD, F4SD, F5SD          !stdev. param. values for DOR
noise - negative
REAL*4:: RHO, RHO2                                   !correlation coefficient: A0
and A1, B0 and B1
REAL*4:: PNAV, PNSD                                  !positive noise occurence
percentage ave. and s.dev.

!dimension REAL*4 variables - input (common block)
REAL*4,DIMENSION(365):: DORTIME                      !time array used for sinusoids
(1/365 - 365/365)
REAL*4,DIMENSION(183):: MAGAVPL                      !residual magnitude averages
array
REAL*4,DIMENSION(183):: MAGSDPL                      !residual magnitude standards
deviation array
REAL*4,DIMENSION(365):: DPNAVPL                      !daily positive noise averages
array
REAL*4,DIMENSION(365):: DPNSDPL                      !daily positive noise standard
deviations array
REAL*4,DIMENSION(365):: DNNAVPL                      !daily negative noise averages
array
REAL*4,DIMENSION(365):: DNNSDPL                      !daily negative noise standard
deviations array

!REAL*4 variables - non input
REAL*4, PARAMETER:: PI=3.14159265358979

REAL*4:: PNOISEPER                                   !positive noice occurence
percentage for the year
REAL*4:: A0,A1,A2,A3,A4
REAL*4:: B0,B1,B2
REAL*4:: FREQ0, FREQ1, FREQ2                         !freq=0, freq=1, and freq=2
parts of DOR signal
REAL*4:: BEAT                                        !the beat sinusoid part of the
DOR signal
REAL*4:: NOISE                                       !noise portion of DOR signal
(uses MAG and ANGLE, DATTIME and FREQ)
REAL*4:: FREQ                                        !working variable for
frequency calculations
REAL*4:: TEMP1, TEMP2                                !working variables
REAL*4:: RANDRAD, MAGSDEVRAD, ANGLERAD               !functions

!dimension REAL*4 variables - non input
REAL*4,DIMENSION(183):: MAG                          !magnitude array rebuilt from
ave. and s.dev. polynomials
REAL*4,DIMENSION(183):: ANGLE                        !phase angle array
```

```
REAL*4,DIMENSION(365):: NOISESIGN                    !the sign (pos/neg) of noise
for each day in year
REAL*4,DIMENSION(365):: DORS                         !one year's worth of DORs

!INTEGER*4 variables
INTEGER*4:: IERROR1                                  !error code 0=none detected
INTEGER*4:: I,J

!common blocks
COMMON /BLOCK05/
A0AV,A0SD,A1AV,A1SD,A2AV,A2SD,A3AV,A3SD,A4AV,A4SD,B0AV,B0SD,B1AV,B1SD,B2
AV,B2SD,MINDOR,PNAV,PNSD,RHO,RHO2

COMMON /BLOCK06/
DORTIME,MAGAVPL,MAGSDPL,DPNAVPL,DPNSDPL,DNNAVPL,DNNSDPL

IERROR1 = 0

!empty signal to prepare for new DORs
DO I=1,365
  DORS(I) = 0.0
END DO

!Generate residual portion of noise
DO I=1,183
  MAG(I) = ABS(MAGAVPL(I) + MAGSDEVRAD(1,2.5)*MAGSDPL(I))
  ANGLE(I) = (RANDRAD(1)-0.5) * 2*PI
END DO

!Calculate postive noise occurence percentage and NOISESIGN vector
PNOISEPER = PNAV + MAGSDEVRAD(1,1.0)*PNSD

DO I=1,365
  NOISESIGN(I) = PNOISEPER - RANDRAD(1)
END DO

!set up frequency 0, 1, and 2 parameters
TEMP1=MAGSDEVRAD(1,5.0)                              !notice different bounds in
random # generation
TEMP2=MAGSDEVRAD(1,1.5)

A0 = A0AV + TEMP1*A0SD

A1 = A1AV + (TEMP1*RHO + TEMP2*(1-RHO**2)**0.5)*A1SD
A2 = A2AV + MAGSDEVRAD(1,1.5)*A2SD
A2 = ANGLERAD(A2)
```

```
A3 = A3AV + MAGSDEVRAD(1,2.0)*A3SD
A4 = A4AV + MAGSDEVRAD(1,2.0)*A4SD
A4 = ANGLERAD(A4)

!set up beat frequency parameters
TEMP1 = MAGSDEVRAD(1,2.0)
TEMP2 = MAGSDEVRAD(1,2.0)

B0 = B0AV + TEMP1*B0SD
B1 = B1AV + (TEMP1*RHO2 + TEMP2*(1-RHO2**2)**0.5)*B1SD
B2 = B2AV + MAGSDEVRAD(1,5.0)*B2SD
B2 = ANGLERAD(B2)

!generate signal
DO I=1,365
  NOISE = 0.0
  DO J=1,183
    FREQ = J - 1
    NOISE = NOISE + MAG(J)*SIN(FREQ * DORTIME(I)*2*PI + ANGLE(J))
  END DO


  BEAT = B0 + B1*SIN(DORTIME(I)*2*PI + B2)

  FREQ0 = A0

  FREQ1 = A1*SIN(DORTIME(I)*2*PI + A2)

  FREQ2 = A3*SIN(2*DORTIME(I)*2*PI + A4)

  DORS(I) = FREQ0 + FREQ1 + FREQ2 + BEAT*NOISE
  DORS(I) = MAX(DORS(I), MINDOR)
END DO

!write (14,14001) (DORS(I),I=1,365)
!14001 format (f15.5)

RETURN

END SUBROUTINE DORGEN
!#####                    END OF SUBTROUTINE DORGEN                    #####

!#####                    START FUNCTION ANGLERAD                    #####
!# It calculates the condition angle based on THETA - used by radiation subroutines
!# Inputs : THETA
```

```
REAL*4 FUNCTION ANGLERAD(THETA)

IMPLICIT NONE

REAL*4:: THETA
REAL*4, PARAMETER:: PI = 3.14159265358979

 DO WHILE (ABS(THETA) > PI)
  THETA = THETA * (1 - ABS(THETA)**(-1)*2.0*PI)
END DO
ANGLERAD = THETA

END FUNCTION ANGLERAD
!#####                    END FUNCTION CONDITIONANGLE                    #####

!#####                    START OF MAGSDEVRAD FUNCTION                    #####
REAL*4 FUNCTION MAGSDEVRAD(MIDUM,BOUND)
! Function returns normally distributed numbers but removes outliers (i.e. values +/-
2.5stds).  This function is based on mrand.m written by L. Parrott and R. Kok; its main
purpose is for use in daily average temperature modeling.
! Modified so that the bounds with which to identify outliers are given as an argument. YC
Sun Mar 26 04
! copy used by radiation subroutines

INTEGER*4:: MIDUM
REAL*4::BOUND
REAL*4:: GASDEVRAD

MAGSDEVRAD = BOUND+1.0
 DO WHILE(ABS(MAGSDEVRAD) > BOUND)
  MAGSDEVRAD = GASDEVRAD(MIDUM)
END DO

!MAGSDEV = 6.0
! DO WHILE(ABS(MAGSDEV) > 5.0)
!  MAGSDEV = GASDEV(MIDUM)
!END DO

END FUNCTION MAGSDEVRAD
!#####                    END OF MAGSDEVRAD FUNCTION                    #####

!#####                    START FUNCTION GASDEVRAD                    #####
!# Returns a normally distributed deviate with zero mean and unit variance, using ran(idum)
as the source of uniform deviates.  Reprinted by Lael from Numerical Recipes in C 2 Ed., p.
289
! copy used by radiation subroutines
```

```
!# Inputs : GIDUM
REAL*4 FUNCTION GASDEVRAD(GIDUM)

INTEGER*4:: GIDUM

INTEGER*4:: ISET

REAL*4:: GSET,FAC,RSQ,V1,V2

REAL*4:: RANDRAD
SAVE ISET,GSET
 DATA ISET/0/

IF (ISET == 0) THEN
  RSQ = 1.
  DO WHILE(RSQ >= 1. .OR. RSQ == 0.)
    V1 = 2. * RANDRAD(GIDUM) - 1
    V2 = 2. * RANDRAD(GIDUM) - 1
    RSQ = V1**2 + V2**2
  END DO
  FAC = SQRT(-2.*LOG(RSQ)/RSQ)
  GSET = V1 * FAC
  GASDEVRAD = V2 * FAC
  ISET = 1
ELSE
  GASDEVRAD = GSET
  ISET = 0
END IF

END FUNCTION GASDEVRAD
!####                          END FUNCTION GASDEV                          ####

!####                       START OF FUNCTION RANDRAD                       ####
FUNCTION RANDRAD(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform
random deviate between 0.0 and 1.0 (exlusive of the endpoint values). Call with idum a
negative integer to initialize; thereafter, do not alter idum between successive deviates in
sequence. RNMX should approximate the largest floating value that is less than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
```

```
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4:: RANDRAD
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
IF (IDUM <= 0) THEN                                    !initialize
 IDUM=MAX(-IDUM,1)                                     !be sure to prevent IDUM=0
 IDUM2=IDUM
 DO J=NTAB+8,1,-1                                      !load the shuffle table (after
8 warm-ups)
   K=IDUM/IQ1
   IDUM=IA1*(IDUM-K*IQ1)-K*IR1
   IF (IDUM < 0) IDUM=IDUM+IM1
   IF (J <= NTAB) IV(J)=IDUM
 END DO
 IY=IV(1)
END IF

K=IDUM/IQ1                                             !start here when not
initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1                            !compute
IDUM=MOD(IA1*IDUM,IM1) without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2                          !compute
IDUM2=MOD(IA2*IDUM2,IM2), likewise
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                                            !will be in the range 1:NTAB
```

```
IY=IV(J)-IDUM2                                  !here IDUM is shuffled,
IDUM and IDUM2 are combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
RANDRAD=MIN(AM*IY,RNMX)                         !because users don't expect
endpoint values
RETURN

END FUNCTION RANDRAD
!#####                END OF FUNCTION PROGRAM RANDRAD                #####

!#####              END OF FUNCTION SUBROUTINE ATTNYEAR              #####
SUBROUTINE ATTNYEAR(YATTN,IERROR2)
!subroutine to generate hourly attenuation factors for one year
!written by YCS, edited by TRL: May 30, 2005

IMPLICIT NONE

INTEGER*4:: I                                   !counter
INTEGER*4:: IERROR2                             !error code
REAL*4:: RANNSE                                 !random noise
REAL*4:: RDUMMY1, RDUMMY2                       !dummy variables
REAL*4:: RANDRAD                                !random number generator
function
REAL*4, DIMENSION(8760):: YATTN                 !array of hourly attenuation
factors for one year

IERROR2 = 0

!an attenuation stream is made with an autocorrelation series (for all 8760 hours in one
year)
RDUMMY1 = (RANDRAD(1) + 1.0) / 2.0

YATTN(1) = RDUMMY1

DO I=2,8760
  RANNSE = RANDRAD(1) - 0.5
  RDUMMY2 = RDUMMY1 + RANNSE
  RDUMMY2 = (RDUMMY2 + 1.0) / 2.5
  RDUMMY1 = RDUMMY2
  YATTN(I) = RDUMMY2
END DO

END SUBROUTINE ATTNYEAR

!*******************************************************************************
```

```
!#####                    START OF SUBTROUTINE DRIGEN                    #####
SUBROUTINE DRIGEN(IDAY,IERROR3)
!subroutine to make daily radiation intensity array
!Creates one day's worth of radiation intensity values with an interval of 10 minutes.
!Information considered includes "real" radiation from FFT approach, the theoretical value,
and anattenuation stream generated with an autocorrelation series, as described in Lael's C
code.
!Each day's array consists of 24*6+1 values (from 0 sec to 86400 sec inclusive)
!written by YCS, edited by TRL June 1, 2005.

IMPLICIT NONE
!declare dimensioned REAL*4 variables
REAL*4, DIMENSION(24,6):: D10MATN                    !in a day, the attenuation
factor from the prelim ones, every 10 min
REAL*4, DIMENSION(24,6):: TENTHERAD                  !every 10-min theoretical rad
energy (kJ/m^2), hours(1-24)
REAL*4, DIMENSION(24):: HOURTHERAD                   !hourly theoretical rad energy
(kJ/m^2), hours (1-24)
REAL*4, DIMENSION(365):: DORS                        !array of DORs
REAL*4, DIMENSION(8760):: YATTN                      !years worth of hourly
attenuation values
REAL*4, DIMENSION(24*6+1):: DRIARRAY                 !array of day's worth of
radiation intensities

!declare REAL*4 variables
REAL*4:: ADJUSTFAC                                   !adjustment factor for solar
attenuation
REAL*4:: DOR                                         !daily overall radiation
(kJ/m^2)
REAL*4:: HALFDAY                                     !half daytime length in sec
REAL*4:: HOURATTFAC                                  !adjusted attenuation factor
for the hour
REAL*4:: LAT                                         !latitude
REAL*4:: THEORSUM, THEORSUM2                         !sum of theoretical rad
intensity (1-TENTHERAD, 2-prelim DRIARRAY)
REAL*4:: TEMP

!declare INTEGER*4 variables
INTEGER*4, DIMENSION(24*6+1):: DAYTIME               !half daytime, used to see if
ITIME is in daylight hours
INTEGER*4:: IDAY                                     !day of year
INTEGER*4:: IERROR3
INTEGER*4:: I,J
```

```
!common blocks
COMMON /BLOCK07/    DORS, YATTN, LAT

COMMON /BLOCK08/    DAYTIME

COMMON /BLOCK09/    TENTHERAD

COMMON /BLOCK10/    D10MATN

COMMON /BLOCK11/    DRIARRAY

IERROR3 = 0

!fill in arrays with 0
DO I=1,24*6+1
  DAYTIME(I) = 0
  DRIARRAY(I) = 0.0
END DO

!Calculate theoretical hourly solar energy; D10MATN is calculated from the hourly
attenuation factor from YATNARRAY.
!Then theoretical values are attenuated by the 10-min attenuation factor for preliminary
attenuation.
!Compare the sum of attenuated solar energy from that "real" one (DOR) obtained from the
FFT approach, which is store in YDORARRAY, an adjust factor can be calculated. This
adjust factor and the pliminary ones together represent the "final" attenuation factor and
is multiplied to the theoretical solar intensity, the results are kept in DRIARRAY, ready to
be read in the RADIATION subroutine

!call subroutine to generate theoretical solar energy for the day, every 10 mins
CALL THEORET(IDAY,LAT)

!call subroutine to generate preliminary attenutation factors for the day, every 10 mins
!CALL DAYATTN(IDAY,YATTN)



!retrieve "real" solar energy for the day from DORs array
DOR = DORS(IDAY)

!calculate the sum of theoretical values, after each is attenuated by a preliminary
attenuation factor
THEORSUM = 0.0
THEORSUM2 = 0.0
DO I=1,24
  DO J=1,6
```

```
!    THEORSUM = THEORSUM + TENTHERAD(I,J)*D10MATN(I,J)
   THEORSUM2 = THEORSUM2 + DRIARRAY((I-1)*6+J)*600   !*D10MATN(I,J)
 END DO
END DO

!write (15,15001) THEORSUM
!15001 format (F15.5)

!write (16,16001) THEORSUM2
!16001 format (F15.5)

TEMP = 0.99 * THEORSUM2

!calculate the adjust factor
IF (DOR > TEMP) THEN
 ADJUSTFAC = 0.99
ELSE
 ADJUSTFAC = DOR / THEORSUM2
END IF

!calculate the final radiation intensities and store in DRIARRAY
DO I=1,24
 DO J=1,6
   DRIARRAY((I-1)*6+J) = DRIARRAY((I-1)*6+J) * ADJUSTFAC * 1000     ! *
D10MATN(I,J) !convert units from J/s/m^2 to kJ/s/m^2 (W/m^2)
 END DO
END DO
DRIARRAY(6*24+1) = DRIARRAY(6*24+1) * ADJUSTFAC  * 1000   ! * D10MATN(24,6)
!last one in array

!write (13,13001) (DRIARRAY(I),I=1,24*6+1)
!13001 FORMAT (E15.5)

RETURN

END SUBROUTINE DRIGEN
!#####                        END OF SUBTROUTINE DRIGEN                        #####

!#####                      START OF SUBTROUTINE THEORET                       #####
SUBROUTINE THEORET(IDAY,LAT)
!- total solar energy for every 10 mins  (kJ/m^2) is calculated for a given day, as well
DRIARRAY is filled with theoretical values.(intensity kJ/sec/m^2)
!- total solar energy every 10 mins is integrated with a simple Euler method
!- also fill into DAYTIME, DRIARRAY with calculated theoretical intensity, along the way
of integration
!- LAT is already treated in INITRADIATION, to get rid of the invalid TAN(PI/2)
```

```
!- last updated by YC Sun 11:30 041604
!- adjusted by TRL June 1, 2005

IMPLICIT NONE
!declare dimensioned REAL*4 variables

REAL*4,DIMENSION(24*6+1)::DRIARRAY                !array of daily radiation
intensity values
REAL*4,DIMENSION(24)::HOURTHERAD                  !hourly theoretical rad engr
(kJ/m^2), hours(1~24)
REAL*4,DIMENSION(24,6)::TENTHERAD                 !every 10-min theoretical rad
engr (kJ/m^2), hours(1~24)

!declare REAL*4 variables
REAL*4,PARAMETER:: PI = 3.14159265358979
REAL*4,PARAMETER:: SOLCONST=1.37                  !solar constant (kJ/sec/M^2)
REAL*4:: COSTEMP                                  !fixed part in the radiation
calculation
REAL*4:: DOR                                      !daily overall radiation
REAL*4:: ENDSUM                                   !used in integration
REAL*4:: HALFDAY                                  !half daytime length in sec
REAL*4:: LAT                                      !latitude
REAL*4:: LOWER                                    !lower bound in sec
REAL*4:: MIDDLESUM                                !used in integration
REAL*4:: RTEMP, RTEMP1                            !temporary variable
REAL*4:: SINTEMP                                  !fixed part in the radiation
calculation
REAL*4:: SUNDECL                                  !sun declination factor
REAL*4:: SUNDIST                                  !sun earth distance factor
REAL*4:: UPPER                                    !upper bound in sec

!declare INTEGER*4 variables
INTEGER*4, DIMENSION(24*6+1)::DAYTIME             !half daytime (in sec) to
check given time in daytime or not
INTEGER*4:: IDAY                                  !day of year
INTEGER*4:: DELTA                                 !deltatime in sec
INTEGER*4:: I,J,K                                 !counters
INTEGER*4:: ITEMP                                 !temporary variable
INTEGER*4:: PIECES                                !# of pieces an integration
span is divided into
INTEGER*4:: SUNLIGHTHOUR                          !number of hours with
sunlight

!common blocks
COMMON /BLOCK08/    DAYTIME
```

```
COMMON /BLOCK09/    TENTHERAD

COMMON /BLOCK11/    DRIARRAY

!calculate sunrise / sunset time, and sun-earth distance factor on radiation intensity
SUNDECL = 0.4093*SIN(2.0*PI*(IDAY + 9.0)/365.0 - PI/2.0)
SUNDIST = 1.0 + 0.033*COS(2.0*PI*IDAY/365.0)

!check daytime length
RTEMP = TAN(SUNDECL)*TAN(LAT)                    !interim value in calculating
HALFDAY

IF (RTEMP > 1.0) THEN                            !24 hour daytime
  HALFDAY = 43200.0
ELSEIF (RTEMP < -1.0) THEN                       !24 hour nighttime
  HALFDAY = 0.0
ELSE
  HALFDAY = 43200.0*ACOS(-RTEMP)/PI
END IF

SUNLIGHTHOUR = CEILING(HALFDAY/3600.0)           !the hour from noon that still
has sunlight

!calculate hourly radiation intensity (deltatime = 1 min)
SINTEMP = SIN(SUNDECL) * SIN(LAT)                !fixed part in the radiation
calculation
COSTEMP = COS(SUNDECL) * COS(LAT)                !fixed part in the radiation
calculation

!update the values for noon
IF (HALFDAY > 0.0) THEN                          !not in 24 dark area
  DAYTIME(73) = 1                                !the element right in the
middle
  RTEMP = 0.0
  DRIARRAY(73) = MAX(0.0,SOLCONST*SUNDIST*(COSTEMP*COS(RTEMP*PI/43200.0)
+ SINTEMP))
END IF

PIECES = 10
DELTA = 60
DO I=1,12                                        !counted from noon
  IF (I > SUNLIGHTHOUR) THEN
    DO J=1,6
      TENTHERAD(13-I,7-J) = 0.0                  !update TENTHERAD, indices
show actual order in array
      TENTHERAD(12+I,J) = 0.0                    !the symmetric one
```

```
    ITEMP = 73 - ((I-1)*6 + J)                          !index of DAYTIME and
DRIARRAY
    DAYTIME(ITEMP) = 0
    DRIARRAY(ITEMP) = 0.0
    ITEMP = 73 + ((I-1)*6 + J)                          !index of the symmetric
element
    DAYTIME(ITEMP) = 0
    DRIARRAY(ITEMP) = 0.0
   END DO
  ELSE
   DO J=1,6
    LOWER = (I-1)*3600.0 + (J-1)*600.0
    UPPER = LOWER + PIECES*DELTA
    ENDSUM = MAX(0.0,SOLCONST*SUNDIST*(COSTEMP*COS(LOWER*PI/43200.0) +
SINTEMP))

    RTEMP = MAX(0.0,SOLCONST*SUNDIST*(COSTEMP*COS(UPPER*PI/43200.0) +
SINTEMP))

    ENDSUM = ENDSUM + RTEMP

    IF (UPPER < HALFDAY) THEN                           !the whole 10 min with
sunlight
     !update THEDAYARR for UPPER, and also for the one on the other side of noon
     DAYTIME(73-((I-1)*6 + J)) = 1
     DRIARRAY(73-((I-1)*6 + J)) = RTEMP
     DAYTIME(73+((I-1)*6 + J)) = 1
     DRIARRAY(73+((I-1)*6 + J)) = RTEMP

     !sum MIDDLESUM for every min in this 10 min period (only 9 of them)
     MIDDLESUM = 0.0

     DO K=1,PIECES-1                                    !deltatime = 1 min
      RTEMP = LOWER + DELTA*K
      RTEMP1 = MAX(0.0,SOLCONST*SUNDIST*(COSTEMP*COS(RTEMP*PI/43200.0) +
SINTEMP))
       MIDDLESUM = MIDDLESUM + RTEMP1
     END DO

    ELSE                                                !have to check if within
daytime
     !update THEDAYARR for UPPER, and also for the one on the other side of noon
     DAYTIME(73-((I-1)*6 + J)) = 0
     DRIARRAY(73-((I-1)*6 + J)) = 0.0
     DAYTIME(73+((I-1)*6 + J)) = 0
```

```
      DRIARRAY(73+((I-1)*6 + J)) = 0.0

         !sum MIDDLESUM for every min in this 10 min (only 9 of them)
         MIDDLESUM = 0.0
         DO K=1,PIECES-1                                    !deltatime = 1 min
          RTEMP = LOWER + DELTA*K

          IF (RTEMP < HALFDAY) THEN                         !has sunlight
           RTEMP1 =
MAX(0.0,SOLCONST*SUNDIST*(COSTEMP*COS(RTEMP*PI/43200.0) + SINTEMP))
            MIDDLESUM = MIDDLESUM + RTEMP1
       END IF
      END DO
    END IF     !(UPPER < HALFDAY)

    TENTHERAD(13-I,7-J) = (ENDSUM + 2*MIDDLESUM)*DELTA/2.0
        !update TENTHERAD, indices show the actual order in array
    TENTHERAD(12+I,J) = TENTHERAD(13-I,7-J)

   END DO      !J=1,6
  END IF       !(I>SUNLIGHTHOUR)
END DO

!write (12,12001) (DRIARRAY(I),I=1,24*6+1)
!12001 FORMAT (E15.5)

RETURN

END SUBROUTINE THEORET
!#####                    END OF SUBTROUTINE THEORET                    #####

!#####                   START OF SUBTROUTINE DAYATTN                   #####
SUBROUTINE DAYATTN(IDAY,YATTN)
!Create smoothed attenduation factors for every 10 min in a day.  Values dervied from the
hourly factory stored in YATTN.
!Written by YCS.  Adapted by TRL June 2, 2005

IMPLICIT NONE
!declare REAL*4 variables
REAL*4, DIMENSION(24,6):: D10MATN                   !attenuation factor from the
prelim ones, every 10 min
REAL*4, DIMENSION(24*6):: RARRTEMP                  !temporary array
REAL*4, DIMENSION(8760):: YATTN                     !year of hourly attenuation
factor
REAL*4:: RTEMP                                      !temporary variable
```

```
!declare INTEGER*4 variables
INTEGER*4:: IDAY                                        !day of year
INTEGER*4:: I, J                                        !counters

!common blocks
COMMON /BLOCK10/    D10MATN

DO I=1,24
 !retrieve this hourly attenduation factor from the yearly attenuation factor array
 RTEMP = YATTN((IDAY-1)*24+1)
 !fill it as this hour's every 10 min attenuation factor
 DO J=1,6
  RARRTEMP((I - 1)*6 + J) = RTEMP
 END DO
END DO

!smooth along the whole day, except the last hour, which doesn't need it
DO I=1,23
 !7-point moving average
 DO J=1,6
  D10MATN(I,J) = (RARRTEMP((I-1)*6 + J) + RARRTEMP((I-1)*6 + J + 1) + RARRTEMP((I-
1)*6 + J + 2) + &
                  RARRTEMP((I-1)*6 + J + 3) + RARRTEMP((I-1)*6 + J + 4) + &
                                RARRTEMP((I-1)*6 + J + 5) + RARRTEMP((I-1)*6 + J + 6))
/ 7.0
 END DO
END DO

DO J=1,6
 D10MATN(24,J) = RARRTEMP(138 + J)
END DO

!do j=1,24
!  write (15,15001) (D10MATN(J,I),I=1,6)
!  15001 FORMAT (F8.5)
!END DO

RETURN

END SUBROUTINE DAYATTN
!#####                    END OF SUBTROUTINE DAYATTN                    #####

!*****************************************************************************
!*****************************************************************************
```

```
!#####                START OF FUNCTION PROGRAM RANDOM1                #####
FUNCTION RANDOM1(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform
random deviate between 0.0 and 1.0 (exlusive of the endpoint values). Call with idum a
negative integer to initialize; thereafter, do not alter idum between successive deviates in
sequence. RNMX should approximate the largest floating value that is less than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4:: RANDOM1
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
IF (IDUM <= 0) THEN                                    !initialize
 IDUM=MAX(-IDUM,1)                                     !be sure to prevent IDUM=0
 IDUM2=IDUM
 DO J=NTAB+8,1,-1                                      !load the shuffle table (after
8 warm-ups)
   K=IDUM/IQ1
   IDUM=IA1*(IDUM-K*IQ1)-K*IR1
   IF (IDUM < 0) IDUM=IDUM+IM1
   IF (J <= NTAB) IV(J)=IDUM
 END DO
```

```
 IY=IV(1)
END IF

K=IDUM/IQ1                                          !start here when not
initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1                         !compute
IDUM=MOD(IA1*IDUM,IM1) without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2                       !compute
IDUM2=MOD(IA2*IDUM2,IM2), likewise
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                                         !will be in the range 1:NTAB
IY=IV(J)-IDUM2                                      !here IDUM is shuffled,
IDUM and IDUM2 are combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
RANDOM1=MIN(AM*IY,RNMX)                             !because users don't expect
endpoint values
RETURN

END FUNCTION RANDOM1
!#####                  END OF FUNCTION PROGRAM RANDOM1                  #####


!#####                  START OF FUNCTION PROGRAM RANDOM2                #####
FUNCTION RANDOM2(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform
random deviate between 0.0 and 1.0 (exlusive of the endpoint values). Call with idum a
negative integer to initialize; thereafter, do not alter idum between successive deviates in
sequence. RNMX should approximate the largest floating value that is less than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
```

```
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4:: RANDOM2
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
IF (IDUM <= 0) THEN                                    !initialize
  IDUM=MAX(-IDUM,1)                                    !be sure to prevent IDUM=0
  IDUM2=IDUM
  DO J=NTAB+8,1,-1                                     !load the shuffle table (after
8 warm-ups)
    K=IDUM/IQ1
    IDUM=IA1*(IDUM-K*IQ1)-K*IR1
    IF (IDUM < 0) IDUM=IDUM+IM1
    IF (J <= NTAB) IV(J)=IDUM
  END DO
  IY=IV(1)
END IF

K=IDUM/IQ1                                             !start here when not
initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1                            !compute
IDUM=MOD(IA1*IDUM,IM1) without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2                          !compute
IDUM2=MOD(IA2*IDUM2,IM2), likewise
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                                            !will be in the range 1:NTAB
IY=IV(J)-IDUM2                                         !here IDUM is shuffled,
IDUM and IDUM2 are combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
RANDOM2=MIN(AM*IY,RNMX)                                !because users don't expect
endpoint values
RETURN
```

```
END FUNCTION RANDOM2
!#####                    END OF FUNCTION PROGRAM RANDOM2                    #####


!#####                    START OF FUNCTION PROGRAM RANDOM3                    #####
FUNCTION RANDOM3(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform
random deviate between 0.0 and 1.0 (exlusive of the endpoint values). Call with idum a
negative integer to initialize; thereafter, do not alter idum between successive deviates in
sequence. RNMX should approximate the largest floating value that is less than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4:: RANDOM3
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
IF (IDUM <= 0) THEN                                    !initialize
  IDUM=MAX(-IDUM,1)                                    !be sure to prevent IDUM=0
  IDUM2=IDUM
  DO J=NTAB+8,1,-1                                     !load the shuffle table (after
8 warm-ups)
    K=IDUM/IQ1
```

```
   IDUM=IA1*(IDUM-K*IQ1)-K*IR1
   IF (IDUM < 0) IDUM=IDUM+IM1
   IF (J <= NTAB) IV(J)=IDUM
  END DO
  IY=IV(1)
END IF

K=IDUM/IQ1                                          !start here when not
initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1                         !compute
IDUM=MOD(IA1*IDUM,IM1) without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2                       !compute
IDUM2=MOD(IA2*IDUM2,IM2), likewise
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                                         !will be in the range 1:NTAB
IY=IV(J)-IDUM2                                      !here IDUM is shuffled,
IDUM and IDUM2 are combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
RANDOM3=MIN(AM*IY,RNMX)                             !because users don't expect
endpoint values
RETURN

END FUNCTION RANDOM3
!#####                  END OF FUNCTION PROGRAM RANDOM3                  #####


!#####                  START OF FUNCTION PROGRAM RANDOM4                #####
FUNCTION RANDOM4(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform
random deviate between 0.0 and 1.0 (exlusive of the endpoint values). Call with idum a
negative integer to initialize; thereafter, do not alter idum between successive deviates in
sequence. RNMX should approximate the largest floating value that is less than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
```

```
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4:: RANDOM4
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
IF (IDUM <= 0) THEN                                    !initialize
  IDUM=MAX(-IDUM,1)                                    !be sure to prevent IDUM=0
  IDUM2=IDUM
  DO J=NTAB+8,1,-1                                     !load the shuffle table (after
8 warm-ups)
    K=IDUM/IQ1
    IDUM=IA1*(IDUM-K*IQ1)-K*IR1
    IF (IDUM < 0) IDUM=IDUM+IM1
    IF (J <= NTAB) IV(J)=IDUM
  END DO
  IY=IV(1)
END IF

K=IDUM/IQ1                                             !start here when not
initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1                            !compute
IDUM=MOD(IA1*IDUM,IM1) without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2                          !compute
IDUM2=MOD(IA2*IDUM2,IM2), likewise
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                                            !will be in the range 1:NTAB
IY=IV(J)-IDUM2                                         !here IDUM is shuffled,
IDUM and IDUM2 are combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
```

```
RANDOM4=MIN(AM*IY,RNMX)                                    !because users don't
expect endpoint values
RETURN

END FUNCTION RANDOM4
!#####                  END OF FUNCTION PROGRAM RANDOM4                  #####


!#####                  START OF FUNCTION PROGRAM RANDOM5                #####
FUNCTION RANDOM5(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform
random deviate between 0.0 and 1.0 (exlusive of the endpoint values). Call with idum a
negative integer to initialize; thereafter, do not alter idum between successive deviates in
sequence. RNMX should approximate the largest floating value that is less than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4:: RANDOM5
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
IF (IDUM <= 0) THEN                                        !initialize
  IDUM=MAX(-IDUM,1)                                        !be sure to prevent IDUM=0
```

```
 IDUM2=IDUM
 DO J=NTAB+8,1,-1                                !load the shuffle table (after 8
warm-ups)
   K=IDUM/IQ1
   IDUM=IA1*(IDUM-K*IQ1)-K*IR1
   IF (IDUM < 0) IDUM=IDUM+IM1
   IF (J <= NTAB) IV(J)=IDUM
 END DO
 IY=IV(1)
END IF

K=IDUM/IQ1                                       !start here when not initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1                      !compute
IDUM=MOD(IA1*IDUM,IM1) without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2                    !compute
IDUM2=MOD(IA2*IDUM2,IM2), likewise
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                                      !will be in the range 1:NTAB
IY=IV(J)-IDUM2                                   !here IDUM is shuffled, IDUM and
IDUM2 are combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
RANDOM5=MIN(AM*IY,RNMX)                          !because users don't expect endpoint
values
RETURN

END FUNCTION RANDOM5
!####                  END OF FUNCTION PROGRAM RANDOM5                  ####

!*****************************************************************************

!####                    START OF SUBROUTINE ASCEND                     ####

SUBROUTINE ASCEND(FLOWS,A)

!#############################################################
!####      Subroutine to calculate the system ascendency at each ecocycle      ####
!#############################################################

INTEGER*4, PARAMETER:: MAXSPEC=30                !maximum total numer of
species allowed

REAL*4, DIMENSION (MAXSPEC+2,MAXSPEC+2):: FLOWS
REAL*4:: VIGOR                                   !sum of all flows in matrix
```

```
REAL*4:: ORGANIZ                                    !(organization) average mutual
information - degree of constraint
REAL*4:: ROW                                        !sum of all flows from source
compartment
REAL*4:: COLUMN                                     !sum of all flows to sink
compartment
REAL*4:: TEMP1, TEMP2
REAL*4:: A                                          !system ascendency
INTEGER*4:: I, J, K

!calculate VIGOR
VIGOR=0.0

DO I=1,MAXSPEC+2
  DO J=1,MAXSPEC+2
    VIGOR=VIGOR+FLOWS(I,J)
  END DO
END DO

!WRITE (*,*) VIGOR

!calculate ORGANIZ
ORGANIZ=0.0

DO I=1,MAXSPEC+2
  ROW=0.0
  DO K=1,MAXSPEC+2
    ROW=ROW+FLOWS(I,K)
  END DO
  DO J=1,MAXSPEC+2
    IF (FLOWS(I,J) /= 0) THEN
      COLUMN=0.0
            DO K=1,MAXSPEC+2
              COLUMN=COLUMN+FLOWS(K,J)
            END DO
     TEMP1=(FLOWS(I,J)*VIGOR) / (ROW*COLUMN)
            TEMP2=(FLOWS(I,J)/VIGOR)*(LOG(TEMP1))
            ORGANIZ=ORGANIZ+TEMP2
          END IF
  END DO
END DO


!calculate ascendency A
A = VIGOR * ORGANIZ
```

```
RETURN

END SUBROUTINE ASCEND

!####                    END OF SUBROUTINE ASCEND                    ####

!*****************************************************************************

!/////////////////////////////////////////////////////////////////////////////
!####                        END OF SUBROUTINES                        ####
!/////////////////////////////////////////////////////////////////////////////
```

Appendix B

Sample input files for the virtual ecosystem model and simulation program. See Chapter 3, Section 3.3.1 for more information.

B.1 File "ecomod****.inp" - Values for the model parameters

```
! this is datafile ecomod****.inp which contains values for the model parameters
!
! Part 1 - random number seeds
  -209149   -898560
!
! Part 2 - ecosystem composition - Species:  20  1 17 11 12  2  6 19  7  4 31 32

! number of producer species (n1) and consumer species (n2)
      10        2
!
! minimum energy levels for species (1 x ntot)
!      1       2       3       4       5       6       7       8       9      10
    133.0     17.7     84.0    108.0    142.7    125.2     76.7     13.9    147.5    122.9
     48.8     84.6
!
! energy levels at birth for species (1 x ntot)
!      1       2       3       4       5       6       7       8       9      10
    213.5     32.1    146.3    313.6    248.4    198.6    214.5     25.1    232.2    333.5
    139.1    209.2
!
! energy threshold at which species can reproduce (1 x ntot)
!      1       2       3       4       5       6       7       8       9      10
   1580.4    261.5   1336.0   2448.1   1999.3   1747.5   1227.5    152.7   1552.8   1709.5
   1237.0   1811.9
!
! values of the energy quanta of the producers (1 x n1)
!      1       2       3       4       5       6       7       8       9      10
    0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50
!
! specific base metabolic rate for species (1 x ntot)
!      1       2       3       4       5       6       7       8       9      10
  0.8E-07  0.2E-06  0.6E-07  0.2E-06  0.2E-06  0.1E-06  0.1E-06  0.1E-06  0.2E-06
0.2E-06
  0.6E-06  0.4E-06
!
! low end of maximum age for species (1 x ntot) (units= day)
!      1       2       3       4       5       6       7       8       9      10
    1611      158      993     1295     1733     1512      901      109     1794     1484
     549     1001
```

```
!
! absolute maximum age for species (1 x ntot) (units= day)
!      1       2       3       4       5       6       7       8       9      10
     2167    2266    1628    1721    3206    2256    3481    2272    3618    2187
     3171    1831
!
! affect1: food affectedness of consumer species (1 x n2)
!    N1+1    N1+2    N1+3    N1+4    N1+5    N1+6    N1+7    N1+8    N1+9   N1+10
    0.621   0.501
!
!affect2: health affectedness of all species (1 x ntot) - used together with the INTER
matrix
!      1       2       3       4       5       6       7       8       9      10
     22.9    11.0    11.0    13.8    10.2     5.1     6.1     7.2    10.8    10.3
     14.0    12.0
!
! Part 3 - ecosystem structure
!
!food matrix (n2 x ntot)
!row 1 contains food preference values of species N1+1 for species 1, 2, 3, etc
!    1     2     3     4     5     6     7     8     9     10
!
  0.631 0.145 0.000 0.000 0.017 0.000 0.061 0.000 0.000 0.101
  0.000 0.044
!
  0.897 0.000 0.000 0.000 0.028 0.033 0.024 0.007 0.000 0.006
  0.006 0.000
!
!interaction matrix (refers to species healthness) (ntot x ntot)
!row 1 contains values for how species 1 is affected by species 1, 2, 3, etc.
!    1     2     3     4     5     6     7     8     9     10
!
  0.379 -0.986  0.767  0.130  0.414  0.365 -0.060  0.523 -0.021 -0.512
  0.210 -0.607
!
  0.626  0.977 -0.090 -0.404 -0.162  0.687 -0.427 -0.956 -0.997 -0.379
  0.260  0.928
!
  0.499  0.688 -0.674 -0.851  0.502 -0.888 -0.816 -0.675  0.392  0.571
  0.214  0.475
!
 -0.354 -0.242  0.792  0.755  0.346 -0.202 -0.704 -0.583 -0.910 -0.635
  0.283 -0.422
!
  0.816  0.315 -0.709 -0.190  0.839 -0.345 -0.728  0.328  0.882 -0.270
 -0.292 -0.026
```

```
!
  -0.968   0.254  -0.208  -0.660   0.567   0.238  -0.713  -0.905  -0.329   0.276
  -0.679   0.282
!
   0.812  -0.852   0.165  -0.382  -0.627  -0.558   0.750   0.982  -0.939  -0.560
  -0.287   0.040
!
   0.308  -0.039   0.575   0.138   0.271   0.981   0.928   0.068   0.877  -0.223
  -0.771  -0.811
!
   0.666   0.842   0.638  -0.847  -0.208   0.851  -0.441   0.758  -0.138   0.709
   0.038   0.191
!
   0.271  -0.900  -0.324  -0.009   0.609  -0.177  -0.550   0.260   0.195   0.522
   0.759   0.574
!
  -0.190  -0.085  -0.630   0.695  -0.693   0.740  -0.989  -0.362   0.183  -0.900
   0.421   0.002
!
  -0.599  -0.956  -0.739   0.303   0.454  -0.665   0.333  -0.728   0.509  -0.179
  -0.114   0.080
!
! Part 4 - initial state of system
!
! initial population sizes (1 x ntot)
!       1        2        3        4        5        6        7        8        9       10
    10000    10000    10000    10000    10000    10000    10000    10000    10000    10000
10000
       10       10
!
! end of file
```

B.2 File "ecosim****.inp" - Values for the simulation parameters

```
! this is datafile ecosim.inp which contains values for the simulation parameters

! name of the output file for this experiment
ecosys_0001.out

! random number seeds (units= no units)
   -646541  -2789863   -379834

! start day for the simulation (units= day)
       100
```

```
! start time for the simulation (units= sec)(must be < (86400-DELTIME); should be integer
multiple of DELTIME)
      0.0

! maximum number of days allowed for the simulation (units= day)
     1825

! time increment for the simulation (units= sec)(should be integer fraction of 86400)
   3600.0

! upper bound on total system energy (units= energy units)
 0.10E+09

! minimum time in which the system is allowed to double in size (units= year)
     0.02

! attenuation factor variable "alpha"
      4.0

! end of file
```

B.3 File "ecorad****.inp" - Values for the radiation subroutine parameters

```
!file 'ecorad****.inp' - parameters for the RADIAT subroutine - (Montreal)

!   ISEED
   -44561

! latitude (degree)
    45.47

!  MINDOR
   433.00

!        A0AV         A1AV         A2AV         A3AV         A4AV
   13120.27800    8829.81016      -1.34716    1037.22761       0.02965

!        A0SD         A1SD         A2SD         A3SD         A4SD
    456.60124     632.27509       0.05915     382.06399       0.85617

!        B0AV         B1AV         B2AV
    4447.52876    2295.43612      -1.14244

!        B0SD         B1SD         B2SD
    185.07364     208.38921       0.10119
```

```
! magav (D0AV through D5AV)
    9.92396E-02
   -5.21604E-04
    5.67420E-06
    2.07822E-08
   -5.60990E-10
    1.91221E-12

! magsd (D0SD through D5SD)
    5.73754E-02
   -9.11941E-04
    1.84082E-05
   -1.52860E-07
    5.41825E-10
   -6.77494E-13

!     RHO (A0 to A1)
   0.55174

!    RHO2 (B0 to B1)
   0.35574

!    PNAV      PNSD
   0.54977   0.01391

! dpnav (E0AV through E5AV)
    6.73317E-01
    2.58597E-03
    7.49953E-05
   -1.12215E-06
    4.73178E-09
   -6.22206E-12

! dpnsd (E0SD through E5SD)
    3.93449E-01
   -1.86361E-03
    8.29983E-05
   -8.67030E-07
    3.34443E-09
   -4.26600E-12

! dnnav (F0AV through F5AV)
   -7.03378E-01
   -1.74981E-02
    2.44556E-04
   -1.50007E-06
```

```
   4.00604E-09
  -3.75995E-12

! dnnsd (F0SD through F5SD)
   3.95866E-01
   1.01834E-02
  -1.64697E-04
   1.23122E-06
  -3.91411E-09
   4.30624E-12
```

B.4 File "ecotem****.inp" - Values for the temperature subroutine parameters

```
!file 'ecotem****.inp' - parameters for TEMPERAT subroutine - Vancouver A0AV+10

!   ISEED
   -48927

!    A0AV     A1AV     A2AV     A3AV     A4AV
  19.68582   7.06066  -1.87590   1.09537  -0.14956

!    A0SD     A1SD     A2SD     A3SD     A4SD
   0.40035   0.49859   0.08062   0.39782   0.33524

!    B0AV     B1AV     B2AV
   1.81692   0.66674   1.70905

!    B0SD     B1SD     B2SD
   0.15905   0.24835   0.21634

! MAGAV (D0AV through D5AV)
   3.37923E-01
  -7.97406E-03
   1.12432E-04
  -9.49622E-07
   4.21953E-09
  -7.39887E-12

! MAGSD (D0SD through D5SD)
   1.79879E-01
  -5.13337E-03
   9.59917E-05
  -1.01137E-06
   5.18424E-09
  -1.00021E-11
```

```
!     RHO (not actual rho for vancouver)
  -0.50000

!   AMPUP,  AMPDOWN
     0.975     1.020
```

Appendix C
Sample output files. See Chapter 3, Section 3.3.1 for more information.

C.1 File "ecosys**.out" – Main output file**

```
********************************************** START OF OUTPUT FROM INITIALIZATION PHASE
**********************************************

THE OUTPUT FILE NAME FOR THIS EXPERIMENT IS (THIS FILE):        ecosys0001.out

ALL MATRICES SUCCESSFULLY FILLED AT INITIALIZATION

********** OUTPUT FROM INITIALIZATION PHASE PART 1 - DATA INPUT AND OUTPUT INTO "PAPER" FILE
**********

PARAMETER VALUES FOR THE SIMULATION:
MAXIMUM POPULATION SIZE ALLOWED FOR THIS SIMULATION RUN:  100000000
MAXIMUM NUMBER OF SPECIES ALLOWED FOR THIS SIMULATION RUN:   30
SEEDS FOR RANDOM NUMBER GENERATOR FOR THE SIMULATION:    -34038  -896428  -735507
START DAY FOR THE SIMULATION:  100
START TIME FOR THE SIMULATION:       0.0 SECONDS
MAXIMUM NO. DAYS FOR THE SIMULATION:  1825 DAYS
TIME INCREMENT FOR THE SIMULATION:    3600.0 SECONDS
UPPER BOUND ON TOTAL SYSTEM ENERGY:  0.10E+11 ENERGY UNITS
MINIMUM DOUBLING TIME FOR THE SYSTEM:      0.02 YEAR
ALPHA VALUE FOR ATTENUATION FACTOR CALCULATION:       4.0

PARAMETER VALUES FOR THE MODEL:
SEEDS FOR RANDOM NUMBER GENERATOR FOR THE MODEL:    -209149   -898560
NUMBER OF PRODUCER SPECIES:  10
NUMBER OF CONSUMER SPECIES:   2
TOTAL NUMBER OF SPECIES IN SYSTEM:   12   ***** NOTE: CALCULATED
THE ENERGY MINIMA FOR THE SPECIES ARE:
    133.0    17.7    84.0   108.0   142.7   125.2    76.7    13.9   147.5   122.9
```

48.8 84.6
THE BIRTH ENERGIES FOR THE SPECIES ARE:
213.5 32.1 146.3 313.6 248.4 198.6 214.5 25.1 232.2 333.5
139.1 209.2
THE REPRODUCTION THRESHOLD ENERGIES FOR THE SPECIES ARE:
1580.4 261.5 1336.0 2448.1 1999.3 1747.5 1227.5 152.7 1552.8 1709.5
1237.0 1811.9
THE ENERGY QUANTUM MAGNITUDES FOR THE PRODUCER SPECIES ARE:
0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50
THE SPECIFIC METABOLIC RATES FOR THE SPECIES ARE:
0.800E-07 0.200E-06 0.600E-07 0.200E-06 0.200E-06 0.100E-06 0.100E-06 0.100E-06 0.200E-06 0.200E-06
0.600E-06 0.400E-06
THE LOW END OF MAXIMUM AGES FOR THE SPECIES (IN DAYS) ARE:
1611 158 993 1295 1733 1512 901 109 1794 1484
549 1001
THE ABSOLUTE MAXIMUM AGES FOR THE SPECIES (IN DAYS) ARE:
2167 2266 1628 1721 3206 2256 3481 2272 3618 2187
3171 1831
THE LOW END OF MAXIMUM AGES FOR THE SPECIES (IN SECONDS) ARE: ****** NOTE: CALCULATED
0.13919E+09 0.13651E+08 0.85795E+08 0.11189E+09 0.14973E+09 0.13064E+09 0.77846E+08 0.94176E+07
0.15500E+09 0.12822E+09 0.47434E+08 0.86486E+08
THE ABSOLUTE MAXIMUM AGES FOR THE SPECIES (IN SECONDS) ARE: ****** NOTE: CALCULATED
0.18723E+09 0.19578E+09 0.14066E+09 0.14869E+09 0.27700E+09 0.19492E+09 0.30076E+09 0.19630E+09
0.31260E+09 0.18896E+09 0.27397E+09 0.15820E+09
THE FOOD AFFECTEDNESS VALUES FOR THE CONSUMER SPECIES ARE:
0.621 0.501
THE HEALTH AFFECTEDNESS VALUES FOR THE SPECIES ARE:
22.9 11.0 11.0 13.8 10.2 5.1 6.1 7.2 10.8 10.3
14.0 12.0

THE FOOD PREFERENCE VALUES FOR THE CONSUMERS ARE:

0.631 0.145 0.000 0.000 0.017 0.000 0.061 0.000 0.000 0.101
0.000 0.044
0.897 0.000 0.000 0.000 0.028 0.033 0.024 0.007 0.000 0.006
0.006 0.000

HOW SPECIES I (ROW) IS AFFECTED BY SPECIES J (COLUMN) - THE INTERACTION VALUES ARE:

0.379 -0.986 0.767 0.130 0.414 0.365 -0.060 0.523 -0.021 -0.512
0.210 -0.607
0.626 0.977 -0.090 -0.404 -0.162 0.687 -0.427 -0.956 -0.997 -0.379
0.260 0.928
0.499 0.688 -0.674 -0.851 0.502 -0.888 -0.816 -0.675 0.392 0.571
0.214 0.475
-0.354 -0.242 0.792 0.755 0.346 -0.202 -0.704 -0.583 -0.910 -0.635
0.283 -0.422
0.816 0.315 -0.709 -0.190 0.839 -0.345 -0.728 0.328 0.882 -0.270
-0.292 -0.026
-0.968 0.254 -0.208 -0.660 0.567 0.238 -0.713 -0.905 -0.329 0.276
-0.679 0.282
0.812 -0.852 0.165 -0.382 -0.627 -0.558 0.750 0.982 -0.939 -0.560
-0.287 0.040
0.308 -0.039 0.575 0.138 0.271 0.981 0.928 0.068 0.877 -0.223
-0.771 -0.811
0.666 0.842 0.638 -0.847 -0.208 0.851 -0.441 0.758 -0.138 0.709
0.038 0.191
0.271 -0.900 -0.324 -0.009 0.609 -0.177 -0.550 0.260 0.195 0.522
0.759 0.574
-0.190 -0.085 -0.630 0.695 -0.693 0.740 -0.989 -0.362 0.183 -0.900
0.421 0.002
-0.599 -0.956 -0.739 0.303 0.454 -0.665 0.333 -0.728 0.509 -0.179

```
   -0.114     0.080
THE INITIAL POPULATION SIZES ARE:
    10000    10000    10000    10000    10000    10000    10000    10000    10000    10000
       10       10

********************** END OF OUTPUT FROM INITIALIZATION PHASE PART 1 ****************************


**** OUTPUT FROM INITIALIZATION PHASE PART 2 - RANDOM INITIALIZATION PART OF THE SYSTEM COMPOSITION ****

VALUES OBTAINED FROM INITIALIZATION CALLS TO RANDOM NUMBER GENERATORS FOR MODEL (ISEED4,ISEED5) ARE:
0.768891  0.303142
THE TOTAL INITIAL POPULATION SIZE IS (IPOP1):  100020

-------------------- STATS OF THE POPULATIONS AT INITIALIZATION TIME --------------------
SPECIES PRESENT(0/1) POPULATION    START       STOP
   1       1       10000          1       10000
   2       1       10000       10001       20000
   3       1       10000       20001       30000
   4       1       10000       30001       40000
   5       1       10000       40001       50000
   6       1       10000       50001       60000
   7       1       10000       60001       70000
   8       1       10000       70001       80000
   9       1       10000       80001       90000
  10       1       10000       90001      100000
  11       1          10      100001      100010
  12       1          10      100011      100020
```

********** END OF OUTPUT FROM INITIALIZATION PHASE PART 2 **********

****** OUTPUT FROM INITIALIZATION PHASE PART 3 - SETTING UP, SETTING COUNTERS, ETC. ******

THE LENGTH OF THE WHAT MATRIX CONTENTS (IPOP2) WAS SET AT (IPOP1 = TOTAL LIVING POPULATION): 100020
THE NUMBER OF ECOCYCLES PER DAY IS: 24
THE NEW VALUE OF DELTIME WAS SET AT (IN SECONDS): 3600.00
THE MAXUMUM POSSIBLE NUMBER OF ECOCYCLES IS: 43800
NOTE: START DATA CORRECTED FOR "ILLEGAL" START TIME SPECIFICATION DURING FIRST DELTA-TIME OF THE DAY
THE STARTING DAY WAS SET AT (STARTDAY, BEFORE FIRST ITERATION INCREMENTATION): 99
THE STARTING TIME WAS SET AT (STARTTIME, BEFORE FIRST ITERATION INCREMENTATION, IN SECONDS): 86400.
THE STARTING DAY CYCLE NUMBER IS (ICYCLE, BEFORE FIRST ITERATION INCREMENTATION): 24
THE STARTING ECOCYCLE NUMBER IS (IECOCYCLE, BEFORE FIRST ITERATION INCREMENTATION): 0
THE STARTING DAY NUMBER IS (IDAY, BEFORE FIRST ITERATION INCREMENTATION): 99
THE STARTING YEAR NUMBER IS (IYEAR, BEFORE FIRST ITERATION INCREMENTATION): 1
INITIAL TIME-OF-DAY VALUE AT END OF TIME INCREMENT (TIME, BEFORE FIRST INCREMENTATION): 86400.
TOTAL SIMULATION TIME ELAPSED (TTIME, BEFORE FIRST INCREMENTATION): 0.
TOTAL TIME ELAPSED SINCE START OF FIRST YEAR OF SIMULATION (TOTALTIME, BEFORE FIRST INCREMENTATION): 8553600.
TOTAL INITIAL ENERGY IN THE SYSTEM FROM SUMMING VALUES IN THE "WHAT" MATRIX: 0.79965E+08
ENERGY CONTENTS OF THE SPECIES AT VERY START OF SIMULATION:

1	0.89414E+07
2	0.14780E+07
3	0.74182E+07
4	0.13852E+08
5	0.11170E+08
6	0.97376E+07

```
      7   0.72626E+07
      8   0.88718E+06
      9   0.89874E+07
     10   0.10215E+08
     11   0.71792E+04
     12   0.85248E+04
TOTAL ENERGY IN SYSTEM FROM SUMMING SPECIES ENERGIES:     0.79965E+08
THE VALUE OBTAINED FROM SUMMING INDIVIDUAL ENERGIES WAS:     0.79965E+08
MINIMUM DOUBLING TIME FOR THE SYSTEM:  0.63072E+06 SECONDS
MAXIMUM POSSIBLE POWER INTO THE SYSTEM:  0.15855E+05 ENERGY UNITS/SECOND
VALUES OBTAINED FROM INIT CALLS TO RANDOM NUMBER GENERATORS FOR SIMULATION (ISEED1,ISEED2,ISEED3) ARE:
0.583290  0.507911  0.067789
TOTAL ENERGY IN PRODUCER SPECIES:    0.79949E+08

****** MESSAGES FROM SUBROUTINE WEATHER DURING INITIALIZATION ******

TEMPERATURE SUBROUTINE INITIALIZED

RADIATION SUBROUTINE INITIALIZED
THE VALUE OF ALPHA FOR THE ATTENUATION FACTOR CURVE IS:    4.00000
THE CORRESPONDING VALUE OF X FOUND FOR THE ATTENUATION FACTOR CURVE IS:    0.67376
THE NORMALIZATION VALUE FOR THE ATTENUATION FACTOR CURVE IS:    0.57663
****** END OF MESSAGES FROM SUBROUTINE WEATHER DURING INITIALIZATION ******

********************** END OF OUTPUT FROM INITIALIZATION PHASE PART 3 ****************************

************************** END OF OUTPUT FROM INITIALIZATION PHASE *********************************
```

```
******************************** START OF OUTPUT FROM ITERATION PHASE ********************************


*************** START OF OUTPUT FROM ECOCYCLE      1 *************
IECOCYCLE, ICYCLE, IYEAR, IDAY:        1       1       1      100
AT END OF THIS ECOCYCLE: TIME, TTIME, TOTALTIME WILL BE:      3600.0     3600.0   8557200.0
TOTAL ENERGY AT START OF THIS ECOCYLE:    0.79965E+08
SPECIES, POPULATION, AND SPECIES ENERGY AT START OF THIS ECOCYCLE:
   1    10000    0.89414E+07
   2    10000    0.14780E+07
   3    10000    0.74182E+07
   4    10000    0.13852E+08
   5    10000    0.11170E+08
   6    10000    0.97376E+07
   7    10000    0.72626E+07
   8    10000    0.88718E+06
   9    10000    0.89874E+07
  10    10000    0.10215E+08
  11       10    0.71792E+04
  12       10    0.85248E+04
TOTAL ENERGY IN PRODUCER SPECIES AT START OF THIS ECOCYCLE IS:    0.79949E+08
FROM WEATHER: TEMPERAT, IRAD, ENERIN:    26.0   0    0.00000E+00
POPULATIONS AFTER DEATH FROM OLD AGE:
   1     9999
   2    10000
   3    10000
   4    10000
```

5	10000
6	10000
7	10000
8	9999
9	10000
10	10000
11	10
12	10

DEATHS DUE TO OLD AGE:

1	1
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	0
10	0
11	0
12	0

RELATIVE RATE FOR INHERENT METABOLISM (PRODUCERS ONLY, DEPENDENT ON TEMPERATURE): 1.34704

DEATHS DUE TO STARVATION:

1	0
2	0
3	0
4	0
5	0
6	0

```
  7     0
  8     0
  9     0
 10     0
 11     0
 12     0
TOTAL METABOLIZED ENERGY PER SPECIES:
  1   0.34682E+04
  2   0.14335E+04
  3   0.21584E+04
  4   0.13435E+05
  5   0.10834E+05
  6   0.47221E+04
  7   0.35219E+04
  8   0.43015E+03
  9   0.87165E+04
 10   0.99069E+04
 11   0.15507E+02
 12   0.12276E+02
TOTAL ENERGY IN SYSTEM AFTER DEATH FROM OLD AGE AND FROM INHERENT METABOLISM:     0.79905E+08
POPULATIONS, ENERGIES, AND RELATIVE ENERGIES AFTER INHERENT METABOLISM AND DEATH FROM STARVATION:
  1     9999    0.89363E+07   0.11184
  2    10000    0.14766E+07   0.01848
  3    10000    0.74161E+07   0.09281
  4    10000    0.13839E+08   0.17319
  5    10000    0.11159E+08   0.13966
  6    10000    0.97329E+07   0.12181
  7    10000    0.72591E+07   0.09085
  8     9999    0.88660E+06   0.01110
```

```
    9    10000    0.89786E+07   0.11237
   10    10000    0.10205E+08   0.12771
   11       10    0.71637E+04   0.00009
   12       10    0.85126E+04   0.00011
THE FEEDNOT PROBABILITIES ARE (CONSUMER SPECIES ONLY):
  0.99374   0.99089
THE FEED PROBABILITIES ARE:
  0.00474   0.00003   0.00000   0.00000   0.00020   0.00000   0.00030   0.00000   0.00000   0.00099
  0.00000   0.00000
  0.00814   0.00000   0.00000   0.00000   0.00040   0.00036   0.00014   0.00000   0.00000   0.00007
  0.00000   0.00000
THE HEALTH VECTOR IS:
  0.00667  -0.01263  -0.01309  -0.00506   0.00741  -0.04028  -0.03758   0.05872   0.01116   0.00846
 -0.01113  -0.00515
CHECKING UP ON THE POPULATIONS TO SEE IF THEY STILL MAKE SENSE:
SPECIES#   PRESENCE   POPULATION   TOTAL SPECIES ENERGY
--------   --------   ----------   --------------------
    1         1         9999          0.89371E+07
    2         1        10000          0.14766E+07
    3         1        10000          0.74161E+07
    4         1        10000          0.13839E+08
    5         1        10000          0.11159E+08
    6         1        10000          0.97329E+07
    7         1        10000          0.72591E+07
    8         1         9999          0.88727E+06
    9         1        10000          0.89787E+07
   10         1        10000          0.10205E+08
   11         1           10          0.71637E+04
   12         1           10          0.85126E+04
```

TOTAL # INDIVIDUALS FROM SUMMING: 100018 IPOP1: 100018 INDEX K AT: 100018
BIRTHS THIS CYCLE:

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0

POPULATION CHECK: TOTAL LIVING POPULATION (IPOP1)= 100018; SUM OF SPECIES POPULATIONS= 100018
FINAL REPORT AT END OF ECOCYCLE 1:

I	IWHO	IPOPS	ENERGY
1	1	9999	0.89363E+07
2	1	10000	0.14766E+07
3	1	10000	0.74161E+07
4	1	10000	0.13839E+08
5	1	10000	0.11159E+08
6	1	10000	0.97329E+07
7	1	10000	0.72591E+07
8	1	9999	0.88660E+06
9	1	10000	0.89787E+07
10	1	10000	0.10205E+08
11	1	10	0.71637E+04

```
  12     1       10    0.85126E+04
THE TOTAL ENERGY CONTENT OF THE SYSTEM AT THE END OF ECOCYCLE       1 IS:     0.79905E+08
ASCENDENCY FOR ECOCYCLE:        0.00000
************************ END OF OUTPUT FROM ECOCYCLE      1 ************************

...

*************** START OF OUTPUT FROM ECOCYCLE   43800  *************
IECOCYCLE, ICYCLE, IYEAR, IDAY:      43800       24        6       99
AT END OF THIS ECOCYCLE: TIME, TTIME, TOTALTIME WILL BE:      86400.0 157680000.0 166233600.0
TOTAL ENERGY AT START OF THIS ECOCYLE:     0.95895E+10
SPECIES, POPULATION, AND SPECIES ENERGY AT START OF THIS ECOCYCLE:
   1  1974018    0.10733E+10
   2        0    0.00000E+00
   3  8708189    0.85162E+10
   4        0    0.00000E+00
   5        0    0.00000E+00
   6        0    0.00000E+00
   7        0    0.00000E+00
   8        0    0.00000E+00
   9        0    0.00000E+00
  10        0    0.00000E+00
  11        0    0.00000E+00
  12        0    0.00000E+00
TOTAL ENERGY IN PRODUCER SPECIES AT START OF THIS ECOCYCLE IS:    0.95895E+10
FROM WEATHER: TEMPERAT, IRAD, ENERIN:    17.8   0    0.00000E+00
POPULATIONS AFTER DEATH FROM OLD AGE:
   1  1973922
   2        0
```

3 8707926
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0

DEATHS DUE TO OLD AGE:

1 96
2 0
3 263
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0

RELATIVE RATE FOR INHERENT METABOLISM (PRODUCERS ONLY, DEPENDENT ON TEMPERATURE): 0.89752

DEATHS DUE TO STARVATION:

1 0
2 0
3 0
4 0

```
  5     0
  6     0
  7     0
  8     0
  9     0
 10     0
 11     0
 12     0
TOTAL METABOLIZED ENERGY PER SPECIES:
  1   0.27747E+06
  2   0.00000E+00
  3   0.17171E+07
  4   0.00000E+00
  5   0.00000E+00
  6   0.00000E+00
  7   0.00000E+00
  8   0.00000E+00
  9   0.00000E+00
 10   0.00000E+00
 11   0.00000E+00
 12   0.00000E+00
TOTAL ENERGY IN SYSTEM AFTER DEATH FROM OLD AGE AND FROM INHERENT METABOLISM:     0.91698E+10
POPULATIONS, ENERGIES, AND RELATIVE ENERGIES AFTER INHERENT METABOLISM AND DEATH FROM STARVATION:
  1  1973922    0.10723E+10   0.11694
  2        0    0.00000E+00   0.00000
  3  8707926    0.80975E+10   0.88306
  4        0    0.00000E+00   0.00000
  5        0    0.00000E+00   0.00000
  6        0    0.00000E+00   0.00000
```

```
   7        0    0.00000E+00   0.00000
   8        0    0.00000E+00   0.00000
   9        0    0.00000E+00   0.00000
  10        0    0.00000E+00   0.00000
  11        0    0.00000E+00   0.00000
  12        0    0.00000E+00   0.00000
THE FEEDNOT PROBABILITIES ARE (CONSUMER SPECIES ONLY):
  0.99507   0.99140
THE FEED PROBABILITIES ARE:
  0.00493   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
  0.00000   0.00000
  0.00860   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
  0.00000   0.00000
THE HEALTH VECTOR IS:
  0.03150   0.00000  -0.04876   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
  0.00000   0.00000
CHECKING UP ON THE POPULATIONS TO SEE IF THEY STILL MAKE SENSE:
SPECIES#   PRESENCE   POPULATION   TOTAL SPECIES ENERGY
--------   --------   ----------   --------------------
   1         1      1973922          0.10730E+10
   2         0            0          0.00000E+00
   3         1      8707926          0.85147E+10
   4         0            0          0.00000E+00
   5         0            0          0.00000E+00
   6         0            0          0.00000E+00
   7         0            0          0.00000E+00
   8         0            0          0.00000E+00
   9         0            0          0.00000E+00
  10         0            0          0.00000E+00
```

```
   11        0          0          0.00000E+00
   12        0          0          0.00000E+00
TOTAL # INDIVIDUALS FROM SUMMING:  10681848   IPOP1:  10681848   INDEX K AT:  10681848
BIRTHS THIS CYCLE:
   1    0
   2    0
   3    0
   4    0
   5    0
   6    0
   7    0
   8    0
   9    0
  10    0
  11    0
  12    0
POPULATION CHECK: TOTAL LIVING POPULATION (IPOP1)=  10681848;   SUM OF SPECIES POPULATIONS=  10681848
FINAL REPORT AT END OF ECOCYCLE  43800:
   I  IWHO  IPOPS       ENERGY
-----------------------------------------------------------------
   1    1  1973922   0.10730E+10
   2    0        0   0.00000E+00
   3    1  8707926   0.85146E+10
   4    0        0   0.00000E+00
   5    0        0   0.00000E+00
   6    0        0   0.00000E+00
   7    0        0   0.00000E+00
   8    0        0   0.00000E+00
   9    0        0   0.00000E+00
```

```
  10     0        0    0.00000E+00
  11     0        0    0.00000E+00
  12     0        0    0.00000E+00
THE TOTAL ENERGY CONTENT OF THE SYSTEM AT THE END OF ECOCYCLE   43800 IS:     0.95876E+10
ASCENDENCY FOR ECOCYCLE:       0.00000
************************ END OF OUTPUT FROM ECOCYCLE   43800 ************************


***************************** END OF OUTPUT FROM ITERATION PHASE **********************************
TIME TO RUN SIMULATION: 647117.12SECONDS


*******************************************************************************************
*******************************************************************************************
********************************** END OF OUTPUT ************************************
*******************************************************************************************
*******************************************************************************************
```

C.2 File "asc**.out" – Ascendency output file**
System ascendency for each ecocycle

```
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
   5445.94629
  13176.31055
  17462.06836
  20667.47070
  22841.69336
  24982.68164
  23831.88477
  23972.79688
  22203.57422
  19092.16211
  16056.94043
  11192.81250
   4909.95557
    193.23129
    120.30851
    209.11815
    144.45876
      0.00000
    578.46271
    251.09726
      0.00000
      0.00000
      0.00000
      0.00000
  73142.46875
 118685.96094
 129234.27344
 133324.26562
 132698.18750
 148617.79688
 154286.14062
 184753.89062
 164802.75000
 157112.57812
 135201.79688
 105257.75000
```

etc.

C.3 File "eat**.out" – Who eats who output file**

Quantity eaten of each species in the ecosystem (column) by a given consumer (row) for each ecocycle of iteration. Note: Breaks between ecocycles added manually.

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

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   1   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   0   0   0   0   0   0   0   0   0   0   0
1   0   0   0   0   0   0   0   0   0   0   0
```

etc.

Appendix D
Species definitions spreadsheets. See Chapter 4 for how these spreadsheets were used.

D.1 Species attributes

D.1.1 Values of species attributes

See Table 3.2 (p. 28) for definitions of attribute variable names.

	ENERMIN	ENERBIR	ENERREP	ENERQUAN	XMETAB
P1	17.7	32.1	261.5	0.50	2.18E-07
P2	125.2	198.6	1747.5	0.50	1.17E-07
P3	121.6	316.0	2843.5	0.50	1.66E-07
P4	122.9	333.5	1709.5	0.50	1.54E-07
P5	45.9	136.5	745.2	0.50	2.43E-07
P6	76.7	214.5	1227.5	0.50	1.06E-07
P7	147.5	232.2	1552.8	0.50	2.13E-07
P8	99.4	262.7	2153.7	0.50	9.63E-08
P9	61.3	96.4	754.3	0.50	2.44E-07
P10	137.8	395.1	3859.1	0.50	6.44E-08
P11	108.0	313.6	2448.1	0.50	1.87E-07
P12	142.7	248.4	1999.3	0.50	2.04E-07
P13	148.2	421.9	2736.3	0.50	1.61E-07
P14	14.3	31.0	172.5	0.50	6.15E-08
P15	132.1	361.7	2022.0	0.50	1.75E-07
P16	25.2	48.7	394.5	0.50	2.20E-07
P17	84.0	146.3	1336.0	0.50	6.34E-08
P18	45.3	107.2	840.7	0.50	8.23E-08
P19	13.9	25.1	152.7	0.50	1.22E-07
P20	133.0	213.5	1580.4	0.50	7.71E-08
C1	19.3	34.4	250.0	-	5.31E-07
C2	38.2	109.4	763.1	-	3.61E-07
C3	24.7	37.5	204.5	-	3.89E-07
C4	70.9	133.2	971.3	-	5.89E-07
C5	61.3	109.4	559.0	-	5.19E-07
C6	13.7	34.5	280.1	-	4.44E-07
C7	125.9	279.4	2322.0	-	4.39E-07
C8	17.1	38.2	357.9	-	3.75E-07
C9	139.0	226.3	1653.6	-	5.65E-07
C10	149.5	289.4	1476.6	-	4.17E-07
C11	48.8	139.1	1237.0	-	6.10E-07
C12	84.6	209.2	1811.9	-	3.86E-07
C13	28.7	59.7	584.4	-	4.74E-07
C14	75.1	163.6	929.5	-	4.41E-07
C15	113.6	183.8	1439.7	-	4.95E-07
C16	123.1	307.2	2473.8	-	5.85E-07
C17	126.3	210.1	1896.2	-	4.26E-07
C18	145.3	390.2	2225.8	-	5.32E-07
C19	125.3	350.4	3043.5	-	6.06E-07
C20	135.0	324.8	2694.6	-	5.90E-07

MINMAXAGE	MAXMAXAGE	AFFECT1	AFFECT2
158	2266	-	11.0
1512	2256	-	5.1
1467	2275	-	17.4
1484	2187	-	10.3
513	3034	-	9.5
901	3481	-	6.1
1794	3618	-	10.8
1187	2669	-	15.9
707	2425	-	14.3
1671	3379	-	5.2
1295	1721	-	13.8
1733	3206	-	10.2
1802	2808	-	16.6
114	1663	-	8.0
1599	2667	-	24.1
252	3272	-	14.2
993	1628	-	11.0
505	875	-	8.7
109	2272	-	7.2
1611	2167	-	22.9
177	1744	0.708	23.2
415	2519	0.285	6.2
245	723	0.545	11.3
827	1913	0.507	16.7
707	2460	0.475	13.9
107	2183	0.401	21.3
1521	2021	0.444	19.1
150	2436	0.700	10.9
1686	1938	0.546	5.1
1819	2396	0.643	20.9
549	3171	0.621	14.0
1001	1831	0.501	12.0
295	1057	0.600	21.7
880	2604	0.323	17.6
1366	3396	0.365	6.8
1486	2956	0.665	21.8
1526	3315	0.747	18.3
1766	2427	0.605	14.9
1513	1815	0.373	14.4
1635	3178	0.402	5.7

D.1.2 Formulas for generating values of species attributes

		ENERMIN	ENERBIR
P1	=RAND()	=B2*(150-10)+10	=C2*(RAND()*(3-1.5)+1.5)
P2	=RAND()	=B3*(150-10)+10	=C3*(RAND()*(3-1.5)+1.5)
P3	=RAND()	=B4*(150-10)+10	=C4*(RAND()*(3-1.5)+1.5)
P4	=RAND()	=B5*(150-10)+10	=C5*(RAND()*(3-1.5)+1.5)
P5	=RAND()	=B6*(150-10)+10	=C6*(RAND()*(3-1.5)+1.5)
P6	=RAND()	=B7*(150-10)+10	=C7*(RAND()*(3-1.5)+1.5)
P7	=RAND()	=B8*(150-10)+10	=C8*(RAND()*(3-1.5)+1.5)
P8	=RAND()	=B9*(150-10)+10	=C9*(RAND()*(3-1.5)+1.5)
P9	=RAND()	=B10*(150-10)+10	=C10*(RAND()*(3-1.5)+1.5)
P10	=RAND()	=B11*(150-10)+10	=C11*(RAND()*(3-1.5)+1.5)
P11	=RAND()	=B12*(150-10)+10	=C12*(RAND()*(3-1.5)+1.5)
P12	=RAND()	=B13*(150-10)+10	=C13*(RAND()*(3-1.5)+1.5)
P13	=RAND()	=B14*(150-10)+10	=C14*(RAND()*(3-1.5)+1.5)
P14	=RAND()	=B15*(150-10)+10	=C15*(RAND()*(3-1.5)+1.5)
P15	=RAND()	=B16*(150-10)+10	=C16*(RAND()*(3-1.5)+1.5)
P16	=RAND()	=B17*(150-10)+10	=C17*(RAND()*(3-1.5)+1.5)
P17	=RAND()	=B18*(150-10)+10	=C18*(RAND()*(3-1.5)+1.5)
P18	=RAND()	=B19*(150-10)+10	=C19*(RAND()*(3-1.5)+1.5)
P19	=RAND()	=B20*(150-10)+10	=C20*(RAND()*(3-1.5)+1.5)
P20	=RAND()	=B21*(150-10)+10	=C21*(RAND()*(3-1.5)+1.5)
C1	=RAND()	=B22*(150-10)+10	=C22*(RAND()*(3-1.5)+1.5)
C2	=RAND()	=B23*(150-10)+10	=C23*(RAND()*(3-1.5)+1.5)
C3	=RAND()	=B24*(150-10)+10	=C24*(RAND()*(3-1.5)+1.5)
C4	=RAND()	=B25*(150-10)+10	=C25*(RAND()*(3-1.5)+1.5)
C5	=RAND()	=B26*(150-10)+10	=C26*(RAND()*(3-1.5)+1.5)
C6	=RAND()	=B27*(150-10)+10	=C27*(RAND()*(3-1.5)+1.5)
C7	=RAND()	=B28*(150-10)+10	=C28*(RAND()*(3-1.5)+1.5)
C8	=RAND()	=B29*(150-10)+10	=C29*(RAND()*(3-1.5)+1.5)
C9	=RAND()	=B30*(150-10)+10	=C30*(RAND()*(3-1.5)+1.5)
C10	=RAND()	=B31*(150-10)+10	=C31*(RAND()*(3-1.5)+1.5)
C11	=RAND()	=B32*(150-10)+10	=C32*(RAND()*(3-1.5)+1.5)
C12	=RAND()	=B33*(150-10)+10	=C33*(RAND()*(3-1.5)+1.5)
C13	=RAND()	=B34*(150-10)+10	=C34*(RAND()*(3-1.5)+1.5)
C14	=RAND()	=B35*(150-10)+10	=C35*(RAND()*(3-1.5)+1.5)
C15	=RAND()	=B36*(150-10)+10	=C36*(RAND()*(3-1.5)+1.5)
C16	=RAND()	=B37*(150-10)+10	=C37*(RAND()*(3-1.5)+1.5)
C17	=RAND()	=B38*(150-10)+10	=C38*(RAND()*(3-1.5)+1.5)
C18	=RAND()	=B39*(150-10)+10	=C39*(RAND()*(3-1.5)+1.5)
C19	=RAND()	=B40*(150-10)+10	=C40*(RAND()*(3-1.5)+1.5)
C20	=RAND()	=B41*(150-10)+10	=C41*(RAND()*(3-1.5)+1.5)

ENERREP	ENERQUAN	XMETAB
=D2*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D3*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D4*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D5*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D6*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D7*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D8*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D9*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D10*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D11*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D12*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D13*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D14*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D15*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D16*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D17*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D18*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D19*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D20*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D21*(RAND()*(10-5)+5)	0.5	=RAND()*(0.00000025-0.00000005)+0.00000005
=D22*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D23*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D24*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D25*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D26*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D27*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D28*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D29*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D30*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D31*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D32*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D33*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D34*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D35*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D36*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D37*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D38*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D39*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D40*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035
=D41*(RAND()*(10-5)+5)	-	=RAND()*(0.00000065-0.00000035)+0.00000035

MINMAXAGE	MAXMAXAGE	AFFECT1
=B2*(1825-60)+60	=RAND()*(3650-(H2+30))+(H2+30)	-
=B3*(1825-60)+60	=RAND()*(3650-(H3+30))+(H3+30)	-
=B4*(1825-60)+60	=RAND()*(3650-(H4+30))+(H4+30)	-
=B5*(1825-60)+60	=RAND()*(3650-(H5+30))+(H5+30)	-
=B6*(1825-60)+60	=RAND()*(3650-(H6+30))+(H6+30)	-
=B7*(1825-60)+60	=RAND()*(3650-(H7+30))+(H7+30)	-
=B8*(1825-60)+60	=RAND()*(3650-(H8+30))+(H8+30)	-
=B9*(1825-60)+60	=RAND()*(3650-(H9+30))+(H9+30)	-
=B10*(1825-60)+60	=RAND()*(3650-(H10+30))+(H10+30)	-
=B11*(1825-60)+60	=RAND()*(3650-(H11+30))+(H11+30)	-
=B12*(1825-60)+60	=RAND()*(3650-(H12+30))+(H12+30)	-
=B13*(1825-60)+60	=RAND()*(3650-(H13+30))+(H13+30)	-
=B14*(1825-60)+60	=RAND()*(3650-(H14+30))+(H14+30)	-
=B15*(1825-60)+60	=RAND()*(3650-(H15+30))+(H15+30)	-
=B16*(1825-60)+60	=RAND()*(3650-(H16+30))+(H16+30)	-
=B17*(1825-60)+60	=RAND()*(3650-(H17+30))+(H17+30)	-
=B18*(1825-60)+60	=RAND()*(3650-(H18+30))+(H18+30)	-
=B19*(1825-60)+60	=RAND()*(3650-(H19+30))+(H19+30)	-
=B20*(1825-60)+60	=RAND()*(3650-(H20+30))+(H20+30)	-
=B21*(1825-60)+60	=RAND()*(3650-(H21+30))+(H21+30)	-
=B22*(1825-60)+60	=RAND()*(3650-(H22+30))+(H22+30)	=RAND()*(0.75-0.25)+0.25
=B23*(1825-60)+60	=RAND()*(3650-(H23+30))+(H23+30)	=RAND()*(0.75-0.25)+0.25
=B24*(1825-60)+60	=RAND()*(3650-(H24+30))+(H24+30)	=RAND()*(0.75-0.25)+0.25
=B25*(1825-60)+60	=RAND()*(3650-(H25+30))+(H25+30)	=RAND()*(0.75-0.25)+0.25
=B26*(1825-60)+60	=RAND()*(3650-(H26+30))+(H26+30)	=RAND()*(0.75-0.25)+0.25
=B27*(1825-60)+60	=RAND()*(3650-(H27+30))+(H27+30)	=RAND()*(0.75-0.25)+0.25
=B28*(1825-60)+60	=RAND()*(3650-(H28+30))+(H28+30)	=RAND()*(0.75-0.25)+0.25
=B29*(1825-60)+60	=RAND()*(3650-(H29+30))+(H29+30)	=RAND()*(0.75-0.25)+0.25
=B30*(1825-60)+60	=RAND()*(3650-(H30+30))+(H30+30)	=RAND()*(0.75-0.25)+0.25
=B31*(1825-60)+60	=RAND()*(3650-(H31+30))+(H31+30)	=RAND()*(0.75-0.25)+0.25
=B32*(1825-60)+60	=RAND()*(3650-(H32+30))+(H32+30)	=RAND()*(0.75-0.25)+0.25
=B33*(1825-60)+60	=RAND()*(3650-(H33+30))+(H33+30)	=RAND()*(0.75-0.25)+0.25
=B34*(1825-60)+60	=RAND()*(3650-(H34+30))+(H34+30)	=RAND()*(0.75-0.25)+0.25
=B35*(1825-60)+60	=RAND()*(3650-(H35+30))+(H35+30)	=RAND()*(0.75-0.25)+0.25
=B36*(1825-60)+60	=RAND()*(3650-(H36+30))+(H36+30)	=RAND()*(0.75-0.25)+0.25
=B37*(1825-60)+60	=RAND()*(3650-(H37+30))+(H37+30)	=RAND()*(0.75-0.25)+0.25
=B38*(1825-60)+60	=RAND()*(3650-(H38+30))+(H38+30)	=RAND()*(0.75-0.25)+0.25
=B39*(1825-60)+60	=RAND()*(3650-(H39+30))+(H39+30)	=RAND()*(0.75-0.25)+0.25
=B40*(1825-60)+60	=RAND()*(3650-(H40+30))+(H40+30)	=RAND()*(0.75-0.25)+0.25
=B41*(1825-60)+60	=RAND()*(3650-(H41+30))+(H41+30)	=RAND()*(0.75-0.25)+0.25

AFFECT2
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5
=RAND()*(25-5)+5

D.2 Food preferences of consumer species

D.2.1 Values for food preferences of consumer species

Species preferred (rows) by a given consumer (columns) have a value of 1. Species that the consumer will not eat have a value of 0. See Chapter 3 for more information on consumer food preferences.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
P1	0	0	0	1	1	0	1	1	1	0	1	0	1	1
P2	0	0	0	1	1	0	0	0	1	0	0	1	1	1
P3	1	0	0	0	0	1	0	0	1	0	0	0	0	1
P4	0	0	0	1	0	1	1	0	1	0	1	1	1	1
P5	0	1	1	0	0	1	0	1	0	0	1	1	1	0
P6	0	0	1	0	1	1	0	1	0	0	1	1	1	1
P7	1	1	1	1	0	0	0	1	1	1	0	0	1	1
P8	1	0	1	1	1	0	0	1	1	0	1	1	0	1
P9	0	1	0	0	1	0	1	1	1	1	1	1	1	1
P10	0	0	0	0	1	1	0	0	0	0	1	1	0	0
P11	1	1	1	1	0	0	0	1	1	1	0	0	0	1
P12	1	1	0	0	1	0	1	1	0	0	1	1	0	0
P13	0	1	0	0	1	1	1	1	1	1	1	0	0	1
P14	1	0	0	1	0	1	1	1	1	0	1	1	1	0
P15	1	0	1	0	1	1	1	1	1	1	0	0	1	1
P16	0	0	0	1	0	1	0	1	0	0	1	0	0	0
P17	1	0	0	0	1	0	1	0	0	0	0	0	1	0
P18	1	0	1	0	0	1	0	0	1	1	1	1	0	1
P19	1	0	0	1	0	0	1	0	1	0	0	1	0	1
P20	1	1	0	0	1	0	0	1	1	0	1	1	0	1
C1	0	0	0	0	0	0	0	0	1	1	1	0	1	1
C2	0	0	0	0	0	0	0	1	0	0	1	1	0	1
C3	0	0	0	0	0	0	1	0	0	1	1	1	1	1
C4	0	0	0	0	0	0	1	1	0	1	0	0	0	0
C5	0	0	0	0	0	0	0	1	0	1	0	1	0	1
C6	0	0	0	0	0	0	0	0	1	1	1	1	1	0
C7	0	0	0	0	0	0	0	0	1	0	1	0	0	0
C8	0	0	0	0	0	0	1	0	1	1	1	0	0	0
C9	0	0	0	0	0	0	0	1	0	1	0	0	0	1
C10	0	0	0	0	0	0	0	1	1	0	1	0	1	0
C11	0	0	0	0	0	0	0	1	0	0	0	1	1	0
C12	0	0	0	0	0	0	1	1	1	0	1	0	0	1
C13	0	0	0	0	0	0	0	1	0	0	0	0	0	0
C14	0	0	0	0	0	0	1	0	1	0	1	0	0	0
C15	0	0	0	0	0	0	0	0	0	1	0	1	1	0
C16	0	0	0	0	0	0	0	1	1	0	1	1	1	0
C17	0	0	0	0	0	0	1	1	1	1	1	0	0	0
C18	0	0	0	0	0	0	0	1	1	0	1	1	0	0
C19	0	0	0	0	0	0	0	1	0	1	0	0	0	1
C20	0	0	0	0	0	0	0	0	0	1	1	0	1	0

C15	C16	C17	C18	C19	C20
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1
1	1	1	0	1	1
1	1	0	0	0	0
0	1	1	1	1	0
1	0	1	1	0	0
0	1	1	0	0	1
1	0	1	1	0	0
0	0	1	1	0	0
0	1	1	1	1	1
1	1	0	0	0	1
1	0	1	0	1	1
1	0	0	1	1	1
1	1	1	1	1	0
1	1	1	0	1	1
0	1	0	1	0	0
1	0	1	1	1	0
0	1	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	0	1	0

Legend of special cells

- avoid cannibalism
- herbivores avoid consumers
- carnivores avoid producers

D.2.2 Formulas for generating values of consumer food preferences

Note: Columns of repeated formulas redacted for space; cells coded as 0 to avoid cannibalism (yellow cells above) change rows to match to the consumer in the column.

	C1 through C6	C7 through C14	C15 through C20
P1	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P2	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P3	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P4	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P5	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P6	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P7	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P8	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P9	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P10	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P11	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P12	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P13	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P14	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P15	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P16	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P17	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P18	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P19	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
P20	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)	0
C1	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C2	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C3	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C4	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C5	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C6	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C7	0	0	=IF(RAND()>0.5,1,0)
C8	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C9	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C10	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C11	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C12	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C13	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C14	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C15	0	=IF(RAND()>0.5,1,0)	0
C16	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C17	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C18	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C19	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)
C20	0	=IF(RAND()>0.5,1,0)	=IF(RAND()>0.5,1,0)

D.3 Health interactions

D.3.1 Values of health interactions

How – negatively, positively, or neutrally – and how strongly a given species (row) is affected by each species in the ecosystem (column). See Chapter 3 for more information on health interactions with the ecosystem.

	P1	P2	P3	P4	P5	P6	P7	P8
P1	0.977	0.687	-0.311	-0.379	0.378	-0.427	-0.997	0.502
P2	0.254	0.238	-0.864	0.276	-0.853	-0.713	-0.329	-0.047
P3	0.749	0.762	0.162	-0.213	0.193	-0.334	0.211	-0.221
P4	-0.900	-0.177	0.130	0.522	-0.024	-0.550	0.195	-0.446
P5	-0.412	0.933	-0.118	-0.840	0.923	-0.782	0.011	0.766
P6	-0.852	-0.558	0.719	-0.560	-0.061	0.750	-0.939	-0.509
P7	0.842	0.851	-0.663	0.709	0.192	-0.441	-0.138	-0.048
P8	0.494	-0.181	0.031	-0.286	0.273	0.695	0.245	0.041
P9	0.004	0.076	-0.485	-0.486	-0.099	0.626	0.772	-0.285
P10	-0.841	-0.436	-0.481	-0.832	-0.516	0.201	-0.639	-0.832
P11	-0.242	-0.202	0.238	-0.635	-0.138	-0.704	-0.910	0.841
P12	0.315	-0.345	0.883	-0.270	-0.003	-0.728	0.882	-0.669
P13	-0.131	0.392	0.097	0.394	-0.465	0.026	-0.458	0.039
P14	0.691	0.176	0.085	-0.552	-0.526	0.921	0.301	-0.293
P15	0.509	-0.838	0.468	0.096	-0.105	-0.761	-0.407	0.460
P16	0.539	0.484	-0.148	-0.027	0.767	0.311	0.414	0.913
P17	0.688	-0.888	-0.406	0.571	0.843	-0.816	0.392	0.278
P18	0.832	-0.032	-0.976	-0.930	-0.969	0.882	-0.700	-0.194
P19	-0.039	0.981	0.216	-0.223	0.839	0.928	0.877	-0.678
P20	-0.986	0.365	-0.659	-0.512	0.860	-0.060	-0.021	-0.389
C1	0.864	-0.426	0.438	-0.433	-0.117	0.997	-0.183	-0.452
C2	-0.015	0.747	0.269	0.436	0.613	0.232	-0.664	-0.984
C3	0.875	-0.127	-0.186	-0.482	0.854	0.371	-0.353	-0.418
C4	0.541	-0.424	-0.268	-0.723	-0.856	-0.024	0.267	0.801
C5	-0.425	-0.723	0.435	-0.296	0.103	0.780	0.670	0.730
C6	-0.768	0.344	0.474	0.067	-0.566	0.327	-0.234	0.258
C7	0.850	-0.736	0.117	-0.452	0.885	0.070	0.526	-0.577
C8	0.115	-0.387	0.075	-0.804	-0.030	0.287	-0.432	0.549
C9	0.652	-0.742	0.456	-0.215	-0.138	0.382	0.638	-0.863
C10	-0.146	0.914	-0.900	-0.778	-0.575	0.322	-0.939	0.153
C11	-0.085	0.740	0.332	-0.900	0.841	-0.989	0.183	0.440
C12	-0.956	-0.665	-0.772	-0.179	0.535	0.333	0.509	0.033
C13	0.694	-0.692	0.726	0.594	-0.972	-0.031	0.461	-0.114
C14	-0.071	-0.859	0.508	-0.303	0.329	0.769	-0.642	-0.330
C15	-0.806	0.843	-0.327	0.452	0.750	0.541	0.937	0.290
C16	0.095	-0.803	-0.348	-0.391	-0.670	0.748	-0.288	-0.825
C17	0.539	-0.331	-0.133	-0.689	0.820	-0.456	-0.869	0.123
C18	0.538	0.068	-0.308	-0.822	0.778	-0.511	0.070	-0.529
C19	-0.831	-0.082	-0.492	-0.107	0.678	0.881	-0.679	0.444
C20	0.180	0.027	0.130	-0.112	-0.122	-0.922	0.817	0.026

P9	P10	P11	P12	P13	P14	P15	P16	P17
-0.264	0.947	-0.404	-0.162	-0.879	0.260	0.223	-0.315	-0.090
-0.488	0.442	-0.660	0.567	-0.251	0.696	0.931	0.051	-0.208
0.211	0.737	0.195	-0.155	0.586	-0.866	-0.240	0.215	0.016
0.548	-0.189	-0.009	0.609	0.849	0.643	-0.449	0.503	-0.324
-0.449	-0.015	-0.556	-0.569	-0.333	-0.551	-0.999	0.173	0.579
0.522	0.152	-0.382	-0.627	0.047	0.167	-0.905	0.976	0.165
0.229	-0.493	-0.847	-0.208	-0.958	-0.524	0.102	0.893	0.638
-0.015	0.266	-0.634	-0.650	-0.583	0.147	-0.309	0.179	-0.055
0.380	0.436	0.970	-0.819	-0.734	-0.101	0.217	0.110	0.171
-0.976	-0.195	-0.562	-0.173	0.588	-0.251	-0.572	0.885	-0.625
-0.011	0.920	0.755	0.346	-0.564	0.658	-0.739	0.641	0.792
-0.732	-0.438	-0.190	0.839	-0.138	0.834	-0.432	0.670	-0.709
-0.310	-0.558	-0.265	-0.613	0.683	-0.377	0.348	0.721	-0.211
0.729	0.037	-0.555	-0.569	0.842	-0.991	-0.522	0.469	0.786
-0.652	0.393	-0.277	-0.027	0.733	-0.324	-0.383	0.509	0.250
0.641	-0.114	-0.684	-0.985	0.469	0.474	-0.745	0.268	0.342
-0.740	-0.451	-0.851	0.502	-0.675	-0.241	-0.863	0.601	-0.674
0.251	-0.834	0.874	-0.789	-0.106	0.028	-0.607	-0.641	0.465
0.850	-0.517	0.138	0.271	-0.177	0.771	0.634	0.586	0.575
0.914	-0.945	0.130	0.414	0.393	-0.271	0.198	-0.514	0.767
-0.558	-0.076	-0.853	0.408	-0.193	-0.985	0.768	0.981	0.349
-0.206	0.072	-0.546	0.741	-0.085	0.370	-0.357	0.558	-0.066
-0.473	-0.510	-0.363	0.730	0.823	-0.826	0.586	0.547	0.601
-0.762	-0.555	0.348	0.692	-0.728	0.781	0.545	-0.982	0.693
-0.701	0.396	-0.355	0.304	-0.332	-0.036	-0.031	-0.917	0.313
-0.769	-0.187	0.521	-0.552	0.147	0.464	0.409	0.216	-0.095
0.251	0.332	-0.535	0.300	0.576	0.101	0.418	0.106	-0.297
0.609	-0.034	-0.692	-0.451	0.912	-0.726	-0.276	0.570	0.359
0.723	-0.330	-0.075	0.959	0.063	-0.333	0.961	0.745	0.233
-0.208	0.423	-0.522	-0.553	0.273	0.168	0.023	0.433	0.373
-0.129	-0.670	0.695	-0.693	-0.180	0.811	-0.054	0.444	-0.630
-0.454	-0.925	0.303	0.454	0.309	0.293	-0.299	-0.360	-0.739
-0.606	0.022	0.694	-0.378	-0.989	0.557	0.054	0.247	0.674
-0.776	0.915	-0.904	-0.444	0.981	-0.854	-0.875	0.610	0.165
0.332	0.785	-0.818	0.320	-0.581	0.877	-0.266	-0.409	0.375
-0.388	-0.488	0.899	0.415	-0.048	-0.480	-0.117	-0.057	-0.200
0.942	-0.631	-0.967	-0.537	0.386	0.699	0.813	-0.441	0.517
-0.026	-0.132	-0.545	0.517	-0.239	-0.738	0.975	0.286	-0.273
-0.479	-0.269	-0.710	0.570	0.938	0.616	-0.679	0.204	0.144
0.940	-0.772	0.643	0.521	0.752	-0.185	-0.553	-0.806	-0.944

P18	P19	P20	C1	C2	C3	C4	C5	C6
0.871	-0.956	0.626	0.337	-0.609	0.448	0.458	-0.220	-0.418
0.144	-0.905	-0.968	-0.701	0.009	-0.717	-0.071	0.543	-0.525
0.259	0.742	0.145	0.794	0.018	-0.880	-0.453	0.973	0.316
0.138	0.260	0.271	0.446	-0.430	-0.915	-0.275	-0.609	0.041
-0.715	-0.201	0.834	0.845	-0.431	0.484	-0.612	0.349	0.104
-0.199	0.982	0.812	-0.980	0.237	-0.572	0.653	-0.362	0.159
0.819	0.758	0.666	0.576	0.623	0.278	-0.704	-0.487	-0.928
-0.297	-0.618	-0.518	-0.670	-0.460	0.032	-0.454	-0.047	0.257
0.022	-0.745	0.779	-0.879	0.516	-0.267	-0.089	-0.310	-0.077
0.792	-0.663	-0.136	-0.972	-0.827	-0.130	0.385	0.876	-0.353
-0.699	-0.583	-0.354	0.648	0.835	0.303	0.626	0.042	-0.238
0.846	0.328	0.816	0.745	0.863	0.406	0.959	0.920	0.779
0.875	0.389	-0.761	0.048	-0.783	0.235	-0.941	0.379	0.838
0.520	0.593	-0.954	0.076	-0.016	-0.196	0.065	-0.723	-0.426
-0.918	0.027	-0.455	0.781	0.809	-0.203	-0.039	-0.087	-0.756
-0.488	-0.653	0.892	-0.836	0.477	-0.323	0.065	-0.022	-0.381
0.389	-0.675	0.499	0.305	-0.693	-0.307	0.460	-0.775	-0.440
0.083	0.610	0.576	-0.620	0.626	0.299	-0.727	0.551	-0.117
-0.414	0.068	0.308	0.485	-0.344	-0.071	-0.867	-0.589	0.225
-0.458	0.523	0.379	-0.035	-0.764	-0.745	0.793	-0.918	-0.980
0.555	-0.630	-0.670	-0.708	-0.934	0.152	0.731	0.604	-0.885
-0.647	-0.218	-0.218	0.288	0.881	-0.676	0.655	0.148	-0.977
-0.922	0.822	0.204	0.878	0.638	0.015	0.351	0.326	0.125
-0.424	0.882	-0.318	-0.967	0.322	0.911	-0.834	-0.721	0.381
-0.534	-0.983	0.839	-0.822	-0.163	0.778	-0.514	0.970	0.693
-0.030	0.081	-0.210	0.029	0.335	-0.746	-0.297	0.358	0.694
-0.497	0.462	0.029	-0.074	-0.934	-0.336	0.055	0.976	0.215
0.729	-0.106	-0.353	0.239	-0.957	-0.436	-0.731	-0.896	-0.561
-0.378	-0.423	-0.621	-0.520	-0.462	-0.209	0.711	-0.627	0.344
0.673	-0.635	0.672	0.929	-0.112	-0.669	0.008	-0.156	0.008
-0.783	-0.362	-0.190	0.773	-0.061	-0.642	-0.191	0.629	0.413
0.532	-0.728	-0.599	0.655	-0.101	-0.953	0.794	-0.819	-0.395
-0.091	-0.015	-0.721	-0.955	0.085	0.936	0.030	0.453	0.293
-0.178	-0.417	-0.401	0.363	-0.144	-0.738	0.180	-0.255	-0.515
-0.977	-0.546	-0.588	-0.936	0.879	-0.223	0.600	-0.020	0.195
-0.216	-0.716	0.745	-0.587	-0.620	-0.903	0.040	-0.878	-0.140
-0.052	-0.414	0.280	-0.603	-0.945	0.777	-0.513	-0.449	0.903
0.409	-0.432	-0.415	0.633	-0.250	0.306	0.585	0.145	0.214
-0.673	0.113	0.444	-0.377	-0.959	0.344	-0.159	0.227	-0.252
0.861	-0.889	0.983	-0.935	0.511	0.997	-0.035	0.410	0.220

C7	C8	C9	C10	C11	C12	C13	C14	C15
-0.906	-0.427	0.046	0.940	0.260	0.928	0.222	-0.566	0.624
0.282	0.812	0.727	-0.993	-0.679	0.282	0.551	0.489	0.833
-0.043	-0.746	-0.322	0.004	-0.140	0.366	0.122	0.735	-0.479
-0.414	-0.292	0.018	0.448	0.759	0.574	0.845	-0.683	0.644
0.326	0.372	0.873	0.869	0.449	-0.607	-0.737	0.536	-0.456
-0.618	-0.872	0.799	0.318	-0.287	0.040	-0.677	-0.391	0.124
-0.657	-0.484	-0.835	-0.094	0.038	0.191	0.479	-0.310	0.347
-0.373	0.564	0.997	-0.552	0.451	0.838	0.252	-0.184	-0.678
-0.970	-0.787	0.815	0.694	0.923	0.188	-0.370	0.758	0.242
0.157	0.424	0.791	-0.874	-0.733	0.282	0.297	-0.505	0.385
0.913	-0.887	-0.161	0.568	0.283	-0.422	-0.734	-0.808	-0.299
-0.642	0.886	0.367	-0.522	-0.292	-0.026	-0.318	-0.099	-0.205
-0.032	0.293	-0.755	-0.658	0.979	-0.645	-0.878	-0.870	0.387
-0.351	-0.634	0.183	-0.362	0.361	-0.474	-0.432	0.642	0.841
-0.795	-0.047	-0.721	-0.291	-0.858	-0.130	0.568	-0.425	0.449
0.773	-0.267	0.573	-0.762	-0.806	-0.325	0.247	-0.084	0.924
-0.642	0.169	-0.407	0.928	0.214	0.475	0.210	-0.983	-0.368
0.757	-0.710	0.146	0.729	-0.084	-0.170	-0.201	-0.653	0.628
0.832	0.672	-0.805	-0.356	-0.771	-0.811	-0.655	0.853	-0.242
0.122	-0.795	-0.763	-0.039	0.210	-0.607	-0.422	-0.723	-0.270
-0.960	-0.084	0.801	-0.067	0.464	0.549	-0.122	-0.649	-0.960
0.012	-0.171	-0.065	0.276	-0.608	0.384	0.865	-0.584	-0.529
-0.030	-0.858	0.137	-0.922	-0.480	-0.338	0.788	0.745	-0.745
-0.257	0.724	0.270	0.258	-0.467	0.499	-0.846	0.812	-0.452
0.396	0.431	0.103	0.577	0.283	0.779	0.915	0.431	-0.635
-0.576	-0.310	-0.959	0.287	0.081	0.669	-0.593	-0.965	-0.939
0.556	-0.095	0.361	0.348	-0.622	-0.775	0.485	0.924	-0.858
-0.203	0.243	0.229	-0.047	-0.678	-0.498	0.633	-0.308	0.999
-0.647	0.141	0.328	-0.262	-0.675	0.972	-0.723	0.958	0.308
0.143	0.815	0.931	0.329	0.708	0.803	0.911	0.023	-0.750
-0.242	0.597	0.653	-0.357	0.421	0.002	-0.379	0.715	0.161
0.105	-0.356	0.910	0.633	-0.114	0.080	0.039	0.977	0.516
0.263	-0.194	-0.137	-0.102	-0.585	-0.439	-0.599	-0.945	0.432
0.945	0.851	0.803	-0.273	-0.276	-0.615	-0.994	0.216	0.877
0.860	0.498	-0.575	-0.242	-0.589	0.259	0.901	0.746	0.165
-0.276	-0.387	0.938	0.786	-0.748	0.008	0.262	0.220	0.277
-0.497	0.101	0.670	-0.261	-0.426	-0.605	-0.786	0.999	0.891
-0.693	-0.816	0.288	-0.517	0.962	0.991	0.212	-0.962	0.449
-0.619	0.944	-0.580	-0.002	-0.530	-0.479	0.702	-0.304	0.294
0.357	0.982	0.195	-0.138	0.393	0.196	0.266	-0.098	0.925

C16	C17	C18	C19	C20
0.824	0.790	0.099	-0.426	-0.422
0.550	0.160	0.881	0.187	0.405
-0.662	0.753	-0.382	-0.479	-0.684
-0.442	0.394	0.180	-0.889	-0.345
-0.585	-0.415	-0.088	0.121	0.725
0.398	-0.052	-0.725	0.778	-0.071
0.690	-0.564	0.707	-0.849	0.814
0.942	-0.721	0.084	0.449	0.504
-0.372	-0.177	0.540	0.355	-0.584
0.459	0.320	0.663	0.916	-0.803
0.266	0.906	-0.943	-0.941	-0.293
0.276	-0.053	0.368	-0.821	0.013
0.399	-0.742	-0.945	0.305	0.544
-0.929	0.008	0.255	-0.436	0.085
-0.800	-0.094	-0.616	0.971	0.679
0.154	-0.825	-0.183	0.928	-0.859
0.832	-0.741	-0.453	-0.243	-0.863
0.222	0.386	-0.108	-0.698	0.847
-0.379	0.046	0.831	-0.048	0.464
0.545	-0.438	0.402	0.065	0.805
0.852	0.999	0.664	-0.277	0.631
0.429	0.389	0.525	-0.278	0.817
0.436	0.028	0.940	-0.659	-0.254
-0.229	-0.904	-0.886	-0.338	0.859
-0.412	0.122	0.870	-0.471	0.368
0.492	0.866	-0.308	0.701	-0.201
-0.376	0.967	-0.633	-0.272	0.287
0.134	0.857	-0.534	-0.873	-0.541
0.280	-0.127	0.055	-0.176	0.427
0.614	0.794	0.938	0.833	-0.558
0.950	0.115	-0.150	-0.588	0.729
0.222	0.230	0.158	-0.256	-0.060
-0.333	0.270	-0.581	0.706	0.125
-0.013	0.075	0.466	0.685	0.263
0.869	-0.767	0.198	0.036	0.582
-0.050	-0.877	0.451	-0.925	0.177
0.241	0.043	0.747	-0.397	0.194
0.870	0.318	0.206	0.352	0.226
-0.287	-0.609	0.165	0.517	0.695
-0.027	0.068	0.520	0.540	-0.060

D.3.2 Formulas for generating values of health interactions

Formula for generating values between -1 and 1 is the same for all cells in the active area of the spreadsheet.

	P1 through C20
P1	=RAND()*(1-(-1))+(-1)
P2	=RAND()*(1-(-1))+(-1)
P3	=RAND()*(1-(-1))+(-1)
P4	=RAND()*(1-(-1))+(-1)
P5	=RAND()*(1-(-1))+(-1)
P6	=RAND()*(1-(-1))+(-1)
P7	=RAND()*(1-(-1))+(-1)
P8	=RAND()*(1-(-1))+(-1)
P9	=RAND()*(1-(-1))+(-1)
P10	=RAND()*(1-(-1))+(-1)
P11	=RAND()*(1-(-1))+(-1)
P12	=RAND()*(1-(-1))+(-1)
P13	=RAND()*(1-(-1))+(-1)
P14	=RAND()*(1-(-1))+(-1)
P15	=RAND()*(1-(-1))+(-1)
P16	=RAND()*(1-(-1))+(-1)
P17	=RAND()*(1-(-1))+(-1)
P18	=RAND()*(1-(-1))+(-1)
P19	=RAND()*(1-(-1))+(-1)
P20	=RAND()*(1-(-1))+(-1)
C1	=RAND()*(1-(-1))+(-1)
C2	=RAND()*(1-(-1))+(-1)
C3	=RAND()*(1-(-1))+(-1)
C4	=RAND()*(1-(-1))+(-1)
C5	=RAND()*(1-(-1))+(-1)
C6	=RAND()*(1-(-1))+(-1)
C7	=RAND()*(1-(-1))+(-1)
C8	=RAND()*(1-(-1))+(-1)
C9	=RAND()*(1-(-1))+(-1)
C10	=RAND()*(1-(-1))+(-1)
C11	=RAND()*(1-(-1))+(-1)
C12	=RAND()*(1-(-1))+(-1)
C13	=RAND()*(1-(-1))+(-1)
C14	=RAND()*(1-(-1))+(-1)
C15	=RAND()*(1-(-1))+(-1)
C16	=RAND()*(1-(-1))+(-1)
C17	=RAND()*(1-(-1))+(-1)
C18	=RAND()*(1-(-1))+(-1)
C19	=RAND()*(1-(-1))+(-1)
C20	=RAND()*(1-(-1))+(-1)

Appendix E

Source code and input files of the system creator program. See Chapter 4, Section 4.1.2 for more information on the program.

E.1 Source code of the system creator program

```
PROGRAM SYSTEMS
!
! program to create systems for use with the virtual ecosystem program (input files)
!
IMPLICIT NONE
!integers
INTEGER:: I, J
INTEGER:: ITEMP1, ITEMP2
INTEGER:: NTOT, N1, N2                                  !total species, producer, &
consumers in system
INTEGER:: PREY                                          !number of prey species for a
given consumer
INTEGER:: PREF_COUNT                                    !counter for number of
preferences assigned
INTEGER:: ISEED1,ISEED2,ISEED3                          !random number seeds for
the simulation
INTEGER:: ISEED4,ISEED5                                 !random number seeds for
the model
INTEGER:: ISTARTDAY                                     !start day of the simulation
(in days + 1 <= STARTDAY <= 365 )
INTEGER:: MAXDAYS                                       !maximum total number of
days possible in the simulation, after the start day (days)
INTEGER, DIMENSION(40):: MINMAXAGE_IN, MAXMAXAGE_IN     !age range
values (input)
INTEGER, DIMENSION(20,40):: FOOD_IN                     !food preferences for
CONSUMERS (input)
INTEGER, DIMENSION(:), ALLOCATABLE:: MINMAXAGE_OUT, MAXMAXAGE_OUT
        !age range values (output)
INTEGER, DIMENSION(:), ALLOCATABLE:: PROD, CONS         !producer and consumer
species numbers
INTEGER, DIMENSION(:), ALLOCATABLE:: POPS               !initial populations
INTEGER, DIMENSION(:,:), ALLOCATABLE:: FOOD_POINT       !array container 0 or 1
pointers for food preferences


!reals
REAL:: RANDOM
REAL:: TEMP1, TEMP2
REAL:: PREF                                             !'amount' of preference left
to be doled out
```

```
REAL:: ALPHA                                    !variable used in the
calculation of the attenuation factor for energy input (no units)
REAL:: DELTIME                                  !the time increment used for
the simulation (s) (should be integer fraction of 86400)
REAL:: DBLETIME                                 !minimum doubling time for
the system (year)
REAL:: ENERMAX                                  !maximum possible power into
the system (energy units/second)
REAL:: STARTTIME                                !start time during the day of
the simulation (in seconds, <86400, should be integer multiple of DELTIME)
REAL, DIMENSION(40):: ENERMIN_IN, ENERBIR_IN, ENERREP_IN        !energy values
(input)
REAL, DIMENSION(20):: ENERQUAN_IN               !PRODUCER energy quanta
size (input)
REAL, DIMENSION(40):: XMETAB_IN                         !specific base
metablic rate (input)
REAL, DIMENSION(20):: AFFECT1_IN                        !CONSUMER hunting
ability (input)
REAL, DIMENSION(40):: AFFECT2_IN                        !health sensitivity
(input)
REAL, DIMENSION(40,40):: INTER_IN               !health interaction values
(input)
REAL, DIMENSION(:), ALLOCATABLE:: ENERMIN_OUT, ENERBIR_OUT, ENERREP_OUT
        !energy values (output)
REAL, DIMENSION(:), ALLOCATABLE:: ENERQUAN_OUT  !PRODUCER energy quanta
size (output)
REAL, DIMENSION(:), ALLOCATABLE:: XMETAB_OUT            !specific base
metablic rate (output)
REAL, DIMENSION(:), ALLOCATABLE:: AFFECT1_OUT           !CONSUMER hunting
ability (output)
REAL, DIMENSION(:), ALLOCATABLE:: AFFECT2_OUT           !health sensitivity
(output)
REAL, DIMENSION(:,:), ALLOCATABLE:: FOOD_OUT            !food preferences for
CONSUMERS (output)
REAL, DIMENSION(:,:), ALLOCATABLE:: INTER_OUT           !health interaction
values (output)

!character arrays
CHARACTER (LEN=14):: MODFILE                    !name of model input file
CHARACTER (LEN=14):: SIMFILE                    !name of simulation input file
CHARACTER (LEN=14):: FILENAME                   !name of output file
CHARACTER (LEN=4):: SIM                         !simulation number (text
form)
CHARACTER (LEN=130):: TEXTLINE                  !dummy text lines

!read in species, food, and health interaction data input files
```

```
OPEN (UNIT=1, FILE='species.txt')

READ(1,100) TEXTLINE
100 FORMAT (A1)
READ(1,100) TEXTLINE
READ(1,101) (ENERMIN_IN(I),I=1,40)
101 FORMAT (10F10.1)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,101) (ENERBIR_IN(I),I=1,40)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,101) (ENERREP_IN(I),I=1,40)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,102) (ENERQUAN_IN(I),I=1,20)
102 FORMAT (10F10.2)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,103) (XMETAB_IN(I),I=1,40)
103 FORMAT (10E10.3)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,104) (MINMAXAGE_IN(I),I=1,40)
104 FORMAT (10I10)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,104) (MAXMAXAGE_IN(I),I=1,40)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,105) (AFFECT1_IN(I),I=1,20)
105 FORMAT (10F10.3)

READ(1,100) TEXTLINE
READ(1,100) TEXTLINE
READ(1,101) (AFFECT2_IN(I),I=1,40)

CLOSE(UNIT=1)
OPEN(UNIT=1, FILE='food.txt')
```

```
READ(1,100) TEXTLINE
DO I=1,20
  READ(1,106) (FOOD_IN(I,J),J=1,40)
  106 FORMAT (40I5.0)
END DO

CLOSE(UNIT=1)
OPEN(UNIT=1, FILE='health.txt')

READ(1,100) TEXTLINE
DO I=1,40
  READ(1,107) (INTER_IN(I,J),J=1,40)
  107 FORMAT(40F10.3)
END DO

CLOSE(UNIT=1)

!initialize random number generator
TEMP1=RANDOM(-32747)


!!write sets of input files
!open simulation number file
OPEN (UNIT=2, FILE='sims.txt')

READ(1,100) TEXTLINE
1000 READ (2,200, END=1001) SIM
200 FORMAT (A4)

PRINT *, SIM

WRITE (SIMFILE,700) SIM
700 FORMAT ('ecosim',A4,'.inp')

WRITE (MODFILE,701) SIM
701 FORMAT ('ecomod',A4,'.inp')

!open set of input files to be written
OPEN (UNIT=3, FILE=SIMFILE)
OPEN (UNIT=4, FILE=MODFILE)

!choose number of species between 2 and 30 and number of producers and consumers
TEMP1=RANDOM(1)
```

```
NTOT=TEMP1*(31-2)+2                                    !real number
truncated to integer
IF (NTOT>30) THEN                                      !should never be used, but
just in case
  NTOT=TEMP1*(30-2)+2
END IF

IF (NTOT==2) THEN
  N1=1
  N2=1
ELSE IF (NTOT<21) THEN
  TEMP1=RANDOM(1)
  N1=TEMP1*((NTOT-1)-(NTOT/2))+(NTOT/2)                !real number
truncated
  N2=NTOT-N1
ELSE
  TEMP1=RANDOM(1)
  N1=TEMP1*(20-(NTOT/2))+(NTOT/2)                      !real number
truncated
  N2=NTOT-N1
END IF

PRINT *, NTOT, N1, N2

!allocate vectors and matrices
ALLOCATE (PROD(1:N1))
ALLOCATE (CONS(1:N2))
ALLOCATE (ENERMIN_OUT(1:NTOT))
ALLOCATE (ENERBIR_OUT(1:NTOT))
ALLOCATE (ENERREP_OUT(1:NTOT))
ALLOCATE (ENERQUAN_OUT(1:N1))
ALLOCATE (XMETAB_OUT(1:NTOT))
ALLOCATE (MINMAXAGE_OUT(1:NTOT))
ALLOCATE (MAXMAXAGE_OUT(1:NTOT))
ALLOCATE (AFFECT1_OUT(1:N2))
ALLOCATE (AFFECT2_OUT(1:NTOT))
ALLOCATE (FOOD_POINT(1:N2,1:NTOT))
ALLOCATE (FOOD_OUT(1:N2,1:NTOT))
ALLOCATE (INTER_OUT(1:NTOT,1:NTOT))
ALLOCATE (POPS(1:NTOT))

!choose producer species
DO I=1,N1
  2000  TEMP1=RANDOM(1)
  PROD(I)=TEMP1*(21-1)+1                               !real number truncated
  IF (PROD(I)>20) THEN                                 !just in case
```

```
   PROD(I)=TEMP1*(20-1)+1
  END IF
  IF (I>1) THEN
   DO J=1,I-1
          IF (PROD(J)==PROD(I)) THEN                    !check whether
species has already been chosen
            GOTO 2000
           ELSE
            CONTINUE
           END IF
         END DO
  END IF
END DO

PRINT *, (PROD(I),I=1,N1)

!choose consumer species
DO I=1,N2
  2001  TEMP1=RANDOM(1)
  CONS(I)=TEMP1*(41-21)+21                          !real number truncated
  IF (CONS(I)>40) THEN                              !just in case
   CONS(I)=TEMP1*(40-21)+21
  END IF
  IF (I>1) THEN
   DO J=1,I-1
          IF (CONS(J)==CONS(I)) THEN                !check whether species has
already been chosen
            GOTO 2001
           ELSE
            CONTINUE
           END IF
         END DO
  END IF
END DO

PRINT *, (CONS(I),I=1,N2)

!build output vectors
DO I=1,N1                                           !values for producers
  ITEMP1=PROD(I)                                    !producer in question (spp# 1-
20)
  ENERMIN_OUT(I)=ENERMIN_IN(ITEMP1)
  ENERBIR_OUT(I)=ENERBIR_IN(ITEMP1)
  ENERREP_OUT(I)=ENERREP_IN(ITEMP1)
  ENERQUAN_OUT(I)=ENERQUAN_IN(ITEMP1)
  XMETAB_OUT(I)=XMETAB_IN(ITEMP1)
```

```
 MINMAXAGE_OUT(I)=MINMAXAGE_IN(ITEMP1)
 MAXMAXAGE_OUT(I)=MAXMAXAGE_IN(ITEMP1)
 AFFECT2_OUT(I)=AFFECT2_IN(ITEMP1)
END DO

DO I=N1+1,NTOT                                          !values for consumers
 ITEMP1=CONS(I-N1)                                      !consumer in question (spp#
21-40)
 ENERMIN_OUT(I)=ENERMIN_IN(ITEMP1)
 ENERBIR_OUT(I)=ENERBIR_IN(ITEMP1)
 ENERREP_OUT(I)=ENERREP_IN(ITEMP1)
 XMETAB_OUT(I)=XMETAB_IN(ITEMP1)
 MINMAXAGE_OUT(I)=MINMAXAGE_IN(ITEMP1)
 MAXMAXAGE_OUT(I)=MAXMAXAGE_IN(ITEMP1)
 AFFECT1_OUT(I-N1)=AFFECT1_IN(ITEMP1-20)
 AFFECT2_OUT(I)=AFFECT2_IN(ITEMP1)
END DO

PRINT *, 'VECTORS DONE'

!build food preference matrix - NOTE: there is no guarantee a consumer will have ANY
prey spp
DO I=1,N2
 ITEMP1=CONS(I)                                         !this is the consumer in
question
 DO J=1,N1
  ITEMP2=PROD(J)                                        !this is the possible prey
species(producer)
  FOOD_POINT(I,J)=FOOD_IN(ITEMP1-20,ITEMP2)             !set poniter to 0 or 1
 END DO
 DO J=1,N2
  ITEMP2=CONS(J)                                        !this is the possible prey
species(consumer)
       FOOD_POINT(I,J+N1)=FOOD_IN(ITEMP1-20,ITEMP2)     !set pointer to 0 or 1
 END DO
 PREY=0                                                 !zero counter for number of
prey
 DO J=1,NTOT
       IF (FOOD_POINT(I,J)==1) THEN
         PREY=PREY+1                                    !increment prey counter
         FOOD_OUT(I,J)=0.005                            !give baseline amount of
preference
       END IF
 END DO
 PREF=1-0.005*PREY                                      !set 'amount' of preference
left to be distributed
```

```
 PREF_COUNT=0                                          !set counter for number of
preferences assigned to 0
 DO J=1,NTOT
  IF (FOOD_POINT(I,J)==1) THEN
         PREF_COUNT=PREF_COUNT+1                       !increment preference
counter
          IF (PREF>0) THEN
      IF (PREF_COUNT<PREY) THEN
            TEMP1=RANDOM(1)
                 FOOD_OUT(I,J)=FOOD_OUT(I,J)+TEMP1*PREF     !assign preference
value between 0.005 and PREF
                 PREF=PREF-TEMP1*PREF                  !subtract assigned pref value
for PREF
           ELSE
            FOOD_OUT(I,J)=FOOD_OUT(I,J)+PREF           !last value equals remaining
PREF + 0.005
           END IF
          ELSE
           FOOD_OUT(I,J)=0.005
          END IF
         ELSE
          FOOD_OUT(I,J)=0.000
         END IF
 END DO
END DO

PRINT *, 'FOOD MATRIX DONE'

!build health interaction matrix
DO I=1,N1                                              !values for producers
 ITEMP1=PROD(I)                                        !producer in question (spp# 1-
20)
 DO J=1,N1
  ITEMP2=PROD(J)                                       !interaction species
(producer)
        INTER_OUT(I,J)=INTER_IN(ITEMP1,ITEMP2)
 END DO
 DO J=1,N2
  ITEMP2=CONS(J)                                       !interaction species
(consumer)
        INTER_OUT(I,J+N1)=INTER_IN(ITEMP1,ITEMP2)
 END DO
END DO
DO I=N1+1,NTOT                                         !values for consumers
 ITEMP1=CONS(I-N1)                                     !consumer in question (spp#
21-40)
```

```
  DO J=1,N1
    ITEMP2=PROD(J)                                    !interaction species
(producer)
         INTER_OUT(I,J)=INTER_IN(ITEMP1,ITEMP2)
  END DO
  DO J=1,N2
    ITEMP2=CONS(J)                                    !interaction species
(consumer)
         INTER_OUT(I,J+N1)=INTER_IN(ITEMP1,ITEMP2)
  END DO
END DO

PRINT *, 'INTER MATRIX DONE'

!determine initial populations
DO I=1,N1
  POPS(I)=10000                                       !all producers start at 10,000
END DO
DO I=N1+1,NTOT
  ITEMP1=CONS(I-N1)                                   !consumer in question (spp#
21-40)
  IF (ITEMP1<27) THEN
    POPS(I)=1000                                      !herbivores start at 1000
  ELSE IF (ITEMP1>27 .AND. ITEMP1>34) THEN
    POPS(I)=100                                       !omnivores start at 100
  ELSE
    POPS(I)=10                                        !carnivores start at 10
  END IF
END DO

!calculate random number seeds for ecomod file
TEMP1=RANDOM(1)
ISEED4=TEMP1*(1-1000000)
TEMP1=RANDOM(1)
ISEED5=TEMP1*(1-1000000)

!set variables for ecosim file
TEMP1=RANDOM(1)
ISEED1=TEMP1*(1-1000000)
TEMP1=RANDOM(1)
ISEED2=TEMP1*(1-1000000)
TEMP1=RANDOM(1)
ISEED3=TEMP1*(1-1000000)
ISTARTDAY=100
STARTTIME=0.0
MAXDAYS=1825
```

```
DELTIME=3600.0
ENERMAX=0.10E+11
DBLETIME=0.02
ALPHA=4.0

!write files
!write ecosim file
WRITE (3,300)
300 FORMAT ('! this is datafile ecosim.inp which contains values for the simulation
parameters')
WRITE (3,301) TEXTLINE
301 FORMAT (A1)
WRITE (3,302)
302 FORMAT ('! name of the output file for this experiment')
WRITE (3,303) SIM
303 FORMAT ('ecosys',A4,'.out')
WRITE (3,301) TEXTLINE
WRITE (3,304)
304 FORMAT ('! random number seeds (units= no units)')
WRITE (3,305) ISEED1,ISEED2,ISEED3
305 FORMAT (3I10)
WRITE (3,301) TEXTLINE
WRITE (3,306)
306 FORMAT ('! start day for the simulation (units= day)')
WRITE (3,307) ISTARTDAY
307 FORMAT (I10)
WRITE (3,301) TEXTLINE
WRITE (3,308)
308 FORMAT ('! start time for the simulation (units= sec)(must be < (86400-DELTIME);
should be integer multiple of DELTIME)')
WRITE (3,309) STARTTIME
309 FORMAT (F10.1)
WRITE (3,301) TEXTLINE
WRITE (3,310)
310 FORMAT ('! maximum number of days allowed for the simulation (units= day)')
WRITE (3,311) MAXDAYS
311 FORMAT (I10)
WRITE (3,301) TEXTLINE
WRITE (3,312)
312 FORMAT ('! time increment for the simulation (units= sec)(should be integer fraction
of 86400)')
WRITE (3,313) DELTIME
313 FORMAT (F10.2)
WRITE (3,301) TEXTLINE
WRITE (3,314)
314 FORMAT ('! upper bound on total system energy (units= energy units)')
```

```
WRITE (3,315) ENERMAX
315 FORMAT (E10.2)
WRITE (3,301) TEXTLINE
WRITE (3,316)
316 FORMAT ('! minimum time in which the system is allowed to double in size (units=
year)')
WRITE (3,317) DBLETIME
317 FORMAT (F10.2)
WRITE (3,301) TEXTLINE
WRITE (3,318)
318 FORMAT ('! attenuation factor variable "alpha"')
WRITE (3,319) ALPHA
319 FORMAT (F10.1)
WRITE (3,301) TEXTLINE
WRITE (3,320)
320 FORMAT ('! end of file')

!write model input file
WRITE (4,400)
400 FORMAT ('! this is datafile ecomod.inp which contains values for the model
parameters')
WRITE (4,401) TEXTLINE
401 FORMAT (A1)
WRITE (4,402)
402 FORMAT ('! Part 1 - random number seeds')
WRITE (4,403) ISEED4,ISEED5
403 FORMAT (2I10)
WRITE (4,401) TEXTLINE
WRITE (4,404) (PROD(I),I=1,N1),(CONS(I),I=1,N2)
404 FORMAT ('! Part 2 - ecosystem composition - Species: ',30I3)
WRITE (4,401)
WRITE (4,405)
405 FORMAT ('! number of producer species (n1) and consumer species (n2)')
WRITE (4,406) N1,N2
406 FORMAT (2I10)
WRITE (4,401) TEXTLINE
WRITE (4,407)
407 FORMAT ('! minimum energy levels for species (1 x ntot)')
WRITE (4,408)
408 FORMAT ('!       1       2       3       4       5       6       7       8       9       10')
WRITE (4,409) (ENERMIN_OUT(I),I=1,NTOT)
409 FORMAT (10F10.1)
WRITE (4,401) TEXTLINE
WRITE (4,410)
410 FORMAT ('! energy levels at birth for species (1 x ntot)')
WRITE (4,411)
```

```
411 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,412) (ENERBIR_OUT(I),I=1,NTOT)
412 FORMAT (10F10.1)
WRITE (4,401) TEXTLINE
WRITE (4,413)
413 FORMAT ('! energy threshold at which species can reproduce (1 x ntot)')
WRITE (4,414)
414 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,415) (ENERREP_OUT(I),I=1,NTOT)
415 FORMAT (10F10.1)
WRITE (4,401) TEXTLINE
WRITE (4,416)
416 FORMAT ('! values of the energy quanta of the producers (1 x n1)')
WRITE (4,417)
417 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,418) (ENERQUAN_OUT(I),I=1,N1)
418 FORMAT (10F10.2)
WRITE (4,401) TEXTLINE
WRITE (4,419)
419 FORMAT ('! specific base metabolic rate for species (1 x ntot)')
WRITE (4,420)
420 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,421) (XMETAB_OUT(I),I=1,NTOT)
421 FORMAT (10E10.1)
WRITE (4,401) TEXTLINE
WRITE (4,422)
422 FORMAT ('! low end of maximum age for species (1 x ntot) (units= day)')
WRITE (4,423)
423 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,424) (MINMAXAGE_OUT(I),I=1,NTOT)
424 FORMAT (10I10)
WRITE (4,401) TEXTLINE
WRITE (4,425)
425 FORMAT ('! absolute maximum age for species (1 x ntot) (units= day)')
WRITE (4,426)
426 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,427) (MAXMAXAGE_OUT(I),I=1,NTOT)
427 FORMAT (10I10)
WRITE (4,401) TEXTLINE
WRITE (4,428)
428 FORMAT ('! affect1: food affectedness of consumer species (1 x n2)')
WRITE (4,429)
429 FORMAT ('!    N1+1     N1+2     N1+3     N1+4     N1+5     N1+6     N1+7     N1+8
N1+9    N1+10')
WRITE (4,430) (AFFECT1_OUT(I),I=1,N2)
430 FORMAT (10F10.3)
```

```
WRITE (4,401) TEXTLINE
WRITE (4,431)
431 FORMAT ('!affect2: health affectedness of all species (1 x ntot) - used together with
the INTER matrix')
WRITE (4,432)
432 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,433) (AFFECT2_OUT(I),I=1,NTOT)
433 FORMAT (10F10.1)
WRITE (4,401) TEXTLINE
WRITE (4,434)
434 FORMAT ('! Part 3 - ecosystem structure')
WRITE (4,401) TEXTLINE
WRITE (4,435)
435 FORMAT ('!food matrix (n2 x ntot)')
WRITE (4,436)
436 FORMAT ('!row 1 contains food preference values of species N1+1 for species 1, 2, 3,
etc')
WRITE (4,437)
437 FORMAT ('!     1      2      3      4      5      6      7      8      9     10')
DO I=1,N2
  WRITE (4,401) TEXTLINE
  WRITE (4,438) (FOOD_OUT(I,J),J=1,NTOT)
END DO
438 FORMAT (10F8.3)
WRITE (4,401) TEXTLINE
WRITE (4,439)
439 FORMAT ('!interaction matrix (refers to species healthness) (ntot x ntot)')
WRITE (4,440)
440 FORMAT ('!row 1 contains values for how species 1 is affected by species 1, 2, 3, etc.')
WRITE (4,441)
441 FORMAT ('!     1      2      3      4      5      6      7      8      9     10')
DO I=1,NTOT
  WRITE (4,401) TEXTLINE
  WRITE (4,442) (INTER_OUT(I,J),J=1,NTOT)
END DO
442 FORMAT (10F8.3)
WRITE (4,401) TEXTLINE
WRITE (4,443)
443 FORMAT ('! Part 4 - initial state of system')
WRITE (4,401) TEXTLINE
WRITE (4,444)
444 FORMAT ('! initial population sizes (1 x ntot)')
WRITE (4,445)
445 FORMAT ('!      1       2       3       4       5       6       7       8       9       10')
WRITE (4,446) (POPS(I),I=1,NTOT)
446 FORMAT (10I10)
```

```
WRITE (4,401) TEXTLINE
WRITE (4,447)
447 FORMAT ('! end of file')

!close files
CLOSE (UNIT=3)
CLOSE (UNIT=4)

!deallocate allocatable arrays
DEALLOCATE (PROD)
DEALLOCATE (CONS)
DEALLOCATE (ENERMIN_OUT)
DEALLOCATE (ENERBIR_OUT)
DEALLOCATE (ENERREP_OUT)
DEALLOCATE (ENERQUAN_OUT)
DEALLOCATE (XMETAB_OUT)
DEALLOCATE (MINMAXAGE_OUT)
DEALLOCATE (MAXMAXAGE_OUT)
DEALLOCATE (AFFECT1_OUT)
DEALLOCATE (AFFECT2_OUT)
DEALLOCATE (FOOD_POINT)
DEALLOCATE (FOOD_OUT)
DEALLOCATE (INTER_OUT)
DEALLOCATE (POPS)

!return for next sim number
GOTO 1000

!ending phase
1001 CONTINUE

CLOSE (UNIT=2)

END PROGRAM SYSTEMS

FUNCTION RANDOM(ISEED)
! From Numerical Recipes in Fortran, po.272-273. Long period ( > 2x10^18 ) random-number
generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform
random deviate between 0.0 and 1.0 (exlusive of the endpoint values). Call with idum a
negative integer to initialize; thereafter, do not alter idum between successive deviates in
sequence. RNMX should approximate the largest floating value that is less than 1.

IMPLICIT NONE

INTEGER*4,PARAMETER:: IA1=40014
INTEGER*4,PARAMETER:: IA2=40692
```

```
INTEGER*4:: IDUM
INTEGER*4,SAVE:: IDUM2 = 123456789
INTEGER*4,PARAMETER:: IM1=2147483563
INTEGER*4,PARAMETER:: IM2=2147483399
INTEGER*4,PARAMETER:: IMM1=IM1-1
INTEGER*4,PARAMETER:: IQ1=53668
INTEGER*4,PARAMETER:: IQ2=52774
INTEGER*4,PARAMETER:: IR1=12211
INTEGER*4,PARAMETER:: IR2=3791
INTEGER*4,INTENT(IN):: ISEED
INTEGER*4,SAVE:: IY = 0
INTEGER*4:: J
INTEGER*4:: K

INTEGER*4,PARAMETER:: NTAB=32
INTEGER*4,PARAMETER:: NDIV=1+IMM1/NTAB
INTEGER*4,DIMENSION(NTAB),SAVE:: IV = (NTAB*0)

REAL*4,PARAMETER:: AM=1.0/IM1
REAL*4,PARAMETER:: EPS=1.2E-7
REAL*4:: RANDOM
REAL*4,PARAMETER:: RNMX=1.0-EPS

IDUM=ISEED
IF (IDUM <= 0) THEN                                    !initialize
  IDUM=MAX(-IDUM,1)                                          !be sure to prevent
IDUM=0
  IDUM2=IDUM
  DO J=NTAB+8,1,-1                                     !load the shuffle table (after
8 warm-ups)
    K=IDUM/IQ1
    IDUM=IA1*(IDUM-K*IQ1)-K*IR1
    IF (IDUM < 0) IDUM=IDUM+IM1
    IF (J <= NTAB) IV(J)=IDUM
  END DO
  IY=IV(1)
END IF

K=IDUM/IQ1                                             !start here when not
initializing
IDUM=IA1*(IDUM-K*IQ1)-K*IR1                            !compute
IDUM=MOD(IA1*IDUM,IM1) without overflows
IF ( IDUM < 0 ) IDUM=IDUM+IM1
K=IDUM2/IQ2
IDUM2=IA2*(IDUM2-K*IQ2)-K*IR2                          !compute
IDUM2=MOD(IA2*IDUM2,IM2), likewise
```

```
IF ( IDUM2 < 0 ) IDUM2=IDUM2+IM2
J=1+IY/NDIV                                    !will be in the range 1:NTAB
IY=IV(J)-IDUM2                                 !here IDUM is shuffled,
IDUM and IDUM2 are combined
IV(J)=IDUM
IF ( IY < 1 ) IY=IY+IMM1
RANDOM=MIN(AM*IY,RNMX)                                 !because users don't
expect endpoint values
RETURN

END FUNCTION RANDOM
```

E.2 Input files for the system creator program

E.2.1 Input file of all species attribute values

```
!species attribute values for all possible species
!ENERMIN
    17.7    125.2    121.6    122.9     45.9     76.7    147.5     99.4     61.3    137.8
   108.0    142.7    148.2     14.3    132.1     25.2     84.0     45.3     13.9    133.0
    19.3     38.2     24.7     70.9     61.3     13.7    125.9     17.1    139.0    149.5
    48.8     84.6     28.7     75.1    113.6    123.1    126.3    145.3    125.3    135.0

!ENERBIR
    32.1    198.6    316.0    333.5    136.5    214.5    232.2    262.7     96.4    395.1
   313.6    248.4    421.9     31.0    361.7     48.7    146.3    107.2     25.1    213.5
    34.4    109.4     37.5    133.2    109.4     34.5    279.4     38.2    226.3    289.4
   139.1    209.2     59.7    163.6    183.8    307.2    210.1    390.2    350.4    324.8

!ENERREP
   261.5   1747.5   2843.5   1709.5    745.2   1227.5   1552.8   2153.7    754.3
3859.1
  2448.1   1999.3   2736.3    172.5   2022.0    394.5   1336.0    840.7    152.7   1580.4
   250.0    763.1    204.5    971.3    559.0    280.1   2322.0    357.9   1653.6   1476.6
  1237.0   1811.9    584.4    929.5   1439.7   2473.8   1896.2   2225.8   3043.5
2694.6

!ENERQUAN (PRODUCERS ONLY)
    0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50
    0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50     0.50

!XMETAB
 2.18E-07 1.17E-07 1.66E-07 1.54E-07 2.43E-07 1.06E-07 2.13E-07 9.63E-08 2.44E-
07 6.44E-08
 1.87E-07 2.04E-07 1.61E-07 6.15E-08 1.75E-07 2.20E-07 6.34E-08 8.23E-08 1.22E-
07 7.71E-08
```

5.31E-07 3.61E-07 3.89E-07 5.89E-07 5.19E-07 4.44E-07 4.39E-07 3.75E-07 5.65E-07 4.17E-07
6.10E-07 3.86E-07 4.74E-07 4.41E-07 4.95E-07 5.85E-07 4.26E-07 5.32E-07 6.06E-07 5.90E-07

!MINMAXAGE
158 1512 1467 1484 513 901 1794 1187 707 1671
1295 1733 1802 114 1599 252 993 505 109 1611
177 415 245 827 707 107 1521 150 1686 1819
549 1001 295 880 1366 1486 1526 1766 1513 1635

!MAXMAXAGE
2266 2256 2275 2187 3034 3481 3618 2669 2425 3379
1721 3206 2808 1663 2667 3272 1628 875 2272 2167
1744 2519 723 1913 2460 2183 2021 2436 1938 2396
3171 1831 1057 2604 3396 2956 3315 2427 1815 3178

!AFFECT1 (CONSUMERS ONLY)
0.708 0.285 0.545 0.507 0.475 0.401 0.444 0.700 0.546 0.643
0.621 0.501 0.600 0.323 0.365 0.665 0.747 0.605 0.373 0.402

!AFFECT2
11.0 5.1 17.4 10.3 9.5 6.1 10.8 15.9 14.3 5.2
13.8 10.2 16.6 8.0 24.1 14.2 11.0 8.7 7.2 22.9
23.2 6.2 11.3 16.7 13.9 21.3 19.1 10.9 5.1 20.9
14.0 12.0 21.7 17.6 6.8 21.8 18.3 14.9 14.4 5.7

E.2.2 Input file of consumer food preferences

```
! preferences of  a given consumer (row) for each species
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 1
0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 0 0 1 1 1 1
0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 1 0 1 1
0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 0 0 1 1
0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 1 1 1 0 0
0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0
0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0 1
1 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0
1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 0 0 0
1 0 1 0 0 0 1 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0
1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0
0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0
0 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0
0 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

E.2.3 Input file of health interaction values

! file of health interaction values how a given species (row) is affected by each species in the ecosystem (column)

0.977 0.687 -0.311 -0.379 0.378 -0.427 -0.997 0.502 -0.264 0.947
-0.404 -0.162 -0.879 0.260 0.223 -0.315 -0.090 0.871 -0.956 0.626
0.337 -0.609 0.448 0.458 -0.220 -0.418 -0.906 -0.427 0.046 0.940
0.260 0.928 0.222 -0.566 0.624 0.824 0.790 0.099 -0.426 -0.422
0.254 0.238 -0.864 0.276 -0.853 -0.713 -0.329 -0.047 -0.488 0.442
-0.660 0.567 -0.251 0.696 0.931 0.051 -0.208 0.144 -0.905 -0.968 -
0.701 0.009 -0.717 -0.071 0.543 -0.525 0.282 0.812 0.727 -0.993 -
0.679 0.282 0.551 0.489 0.833 0.550 0.160 0.881 0.187 0.405
0.749 0.762 0.162 -0.213 0.193 -0.334 0.211 -0.221 0.211 0.737
0.195 -0.155 0.586 -0.866 -0.240 0.215 0.016 0.259 0.742 0.145
0.794 0.018 -0.880 -0.453 0.973 0.316 -0.043 -0.746 -0.322 0.004 -
0.140 0.366 0.122 0.735 -0.479 -0.662 0.753 -0.382 -0.479 -0.684
-0.900 -0.177 0.130 0.522 -0.024 -0.550 0.195 -0.446 0.548 -0.189
-0.009 0.609 0.849 0.643 -0.449 0.503 -0.324 0.138 0.260 0.271
0.446 -0.430 -0.915 -0.275 -0.609 0.041 -0.414 -0.292 0.018 0.448
0.759 0.574 0.845 -0.683 0.644 -0.442 0.394 0.180 -0.889 -0.345
-0.412 0.933 -0.118 -0.840 0.923 -0.782 0.011 0.766 -0.449 -0.015
-0.556 -0.569 -0.333 -0.551 -0.999 0.173 0.579 -0.715 -0.201 0.834
0.845 -0.431 0.484 -0.612 0.349 0.104 0.326 0.372 0.873 0.869
0.449 -0.607 -0.737 0.536 -0.456 -0.585 -0.415 -0.088 0.121 0.725
-0.852 -0.558 0.719 -0.560 -0.061 0.750 -0.939 -0.509 0.522 0.152
-0.382 -0.627 0.047 0.167 -0.905 0.976 0.165 -0.199 0.982 0.812 -
0.980 0.237 -0.572 0.653 -0.362 0.159 -0.618 -0.872 0.799 0.318 -
0.287 0.040 -0.677 -0.391 0.124 0.398 -0.052 -0.725 0.778 -0.071
0.842 0.851 -0.663 0.709 0.192 -0.441 -0.138 -0.048 0.229 -0.493
-0.847 -0.208 -0.958 -0.524 0.102 0.893 0.638 0.819 0.758 0.666
0.576 0.623 0.278 -0.704 -0.487 -0.928 -0.657 -0.484 -0.835 -0.094
0.038 0.191 0.479 -0.310 0.347 0.690 -0.564 0.707 -0.849 0.814
0.494 -0.181 0.031 -0.286 0.273 0.695 0.245 0.041 -0.015 0.266
-0.634 -0.650 -0.583 0.147 -0.309 0.179 -0.055 -0.297 -0.618 -0.518 -
0.670 -0.460 0.032 -0.454 -0.047 0.257 -0.373 0.564 0.997 -0.552
0.451 0.838 0.252 -0.184 -0.678 0.942 -0.721 0.084 0.449 0.504
0.004 0.076 -0.485 -0.486 -0.099 0.626 0.772 -0.285 0.380 0.436
0.970 -0.819 -0.734 -0.101 0.217 0.110 0.171 0.022 -0.745 0.779 -
0.879 0.516 -0.267 -0.089 -0.310 -0.077 -0.970 -0.787 0.815 0.694
0.923 0.188 -0.370 0.758 0.242 -0.372 -0.177 0.540 0.355 -0.584
-0.841 -0.436 -0.481 -0.832 -0.516 0.201 -0.639 -0.832 -0.976 -0.195
-0.562 -0.173 0.588 -0.251 -0.572 0.885 -0.625 0.792 -0.663 -0.136 -
0.972 -0.827 -0.130 0.385 0.876 -0.353 0.157 0.424 0.791 -0.874 -
0.733 0.282 0.297 -0.505 0.385 0.459 0.320 0.663 0.916 -0.803
-0.242 -0.202 0.238 -0.635 -0.138 -0.704 -0.910 0.841 -0.011 0.920
0.755 0.346 -0.564 0.658 -0.739 0.641 0.792 -0.699 -0.583 -0.354

0.648 0.835 0.303 0.626 0.042 -0.238 0.913 -0.887 -0.161 0.568
0.283 -0.422 -0.734 -0.808 -0.299 0.266 0.906 -0.943 -0.941 -0.293
0.315 -0.345 0.883 -0.270 -0.003 -0.728 0.882 -0.669 -0.732 -0.438
-0.190 0.839 -0.138 0.834 -0.432 0.670 -0.709 0.846 0.328 0.816
0.745 0.863 0.406 0.959 0.920 0.779 -0.642 0.886 0.367 -0.522 -
0.292 -0.026 -0.318 -0.099 -0.205 0.276 -0.053 0.368 -0.821 0.013
-0.131 0.392 0.097 0.394 -0.465 0.026 -0.458 0.039 -0.310 -0.558
-0.265 -0.613 0.683 -0.377 0.348 0.721 -0.211 0.875 0.389 -0.761
0.048 -0.783 0.235 -0.941 0.379 0.838 -0.032 0.293 -0.755 -0.658
0.979 -0.645 -0.878 -0.870 0.387 0.399 -0.742 -0.945 0.305 0.544
0.691 0.176 0.085 -0.552 -0.526 0.921 0.301 -0.293 0.729 0.037
-0.555 -0.569 0.842 -0.991 -0.522 0.469 0.786 0.520 0.593 -0.954
0.076 -0.016 -0.196 0.065 -0.723 -0.426 -0.351 -0.634 0.183 -0.362
0.361 -0.474 -0.432 0.642 0.841 -0.929 0.008 0.255 -0.436 0.085
0.509 -0.838 0.468 0.096 -0.105 -0.761 -0.407 0.460 -0.652 0.393
-0.277 -0.027 0.733 -0.324 -0.383 0.509 0.250 -0.918 0.027 -0.455
0.781 0.809 -0.203 -0.039 -0.087 -0.756 -0.795 -0.047 -0.721 -0.291 -
0.858 -0.130 0.568 -0.425 0.449 -0.800 -0.094 -0.616 0.971 0.679
0.539 0.484 -0.148 -0.027 0.767 0.311 0.414 0.913 0.641 -0.114 -
0.684 -0.985 0.469 0.474 -0.745 0.268 0.342 -0.488 -0.653 0.892 -
0.836 0.477 -0.323 0.065 -0.022 -0.381 0.773 -0.267 0.573 -0.762 -
0.806 -0.325 0.247 -0.084 0.924 0.154 -0.825 -0.183 0.928 -0.859
0.688 -0.888 -0.406 0.571 0.843 -0.816 0.392 0.278 -0.740 -0.451
-0.851 0.502 -0.675 -0.241 -0.863 0.601 -0.674 0.389 -0.675 0.499
0.305 -0.693 -0.307 0.460 -0.775 -0.440 -0.642 0.169 -0.407 0.928
0.214 0.475 0.210 -0.983 -0.368 0.832 -0.741 -0.453 -0.243 -0.863
0.832 -0.032 -0.976 -0.930 -0.969 0.882 -0.700 -0.194 0.251 -0.834
0.874 -0.789 -0.106 0.028 -0.607 -0.641 0.465 0.083 0.610 0.576 -
0.620 0.626 0.299 -0.727 0.551 -0.117 0.757 -0.710 0.146 0.729 -
0.084 -0.170 -0.201 -0.653 0.628 0.222 0.386 -0.108 -0.698 0.847
-0.039 0.981 0.216 -0.223 0.839 0.928 0.877 -0.678 0.850 -0.517
0.138 0.271 -0.177 0.771 0.634 0.586 0.575 -0.414 0.068 0.308
0.485 -0.344 -0.071 -0.867 -0.589 0.225 0.832 0.672 -0.805 -0.356 -
0.771 -0.811 -0.655 0.853 -0.242 -0.379 0.046 0.831 -0.048 0.464
-0.986 0.365 -0.659 -0.512 0.860 -0.060 -0.021 -0.389 0.914 -0.945
0.130 0.414 0.393 -0.271 0.198 -0.514 0.767 -0.458 0.523 0.379 -
0.035 -0.764 -0.745 0.793 -0.918 -0.980 0.122 -0.795 -0.763 -0.039
0.210 -0.607 -0.422 -0.723 -0.270 0.545 -0.438 0.402 0.065 0.805
0.864 -0.426 0.438 -0.433 -0.117 0.997 -0.183 -0.452 -0.558 -0.076
-0.853 0.408 -0.193 -0.985 0.768 0.981 0.349 0.555 -0.630 -0.670 -
0.708 -0.934 0.152 0.731 0.604 -0.885 -0.960 -0.084 0.801 -0.067
0.464 0.549 -0.122 -0.649 -0.960 0.852 0.999 0.664 -0.277 0.631
-0.015 0.747 0.269 0.436 0.613 0.232 -0.664 -0.984 -0.206 0.072
-0.546 0.741 -0.085 0.370 -0.357 0.558 -0.066 -0.647 -0.218 -0.218
0.288 0.881 -0.676 0.655 0.148 -0.977 0.012 -0.171 -0.065 0.276 -
0.608 0.384 0.865 -0.584 -0.529 0.429 0.389 0.525 -0.278 0.817

0.875 -0.127 -0.186 -0.482 0.854 0.371 -0.353 -0.418 -0.473 -0.510
-0.363 0.730 0.823 -0.826 0.586 0.547 0.601 -0.922 0.822 0.204
0.878 0.638 0.015 0.351 0.326 0.125 -0.030 -0.858 0.137 -0.922 -
0.480 -0.338 0.788 0.745 -0.745 0.436 0.028 0.940 -0.659 -0.254
0.541 -0.424 -0.268 -0.723 -0.856 -0.024 0.267 0.801 -0.762 -0.555
0.348 0.692 -0.728 0.781 0.545 -0.982 0.693 -0.424 0.882 -0.318 -
0.967 0.322 0.911 -0.834 -0.721 0.381 -0.257 0.724 0.270 0.258 -
0.467 0.499 -0.846 0.812 -0.452 -0.229 -0.904 -0.886 -0.338 0.859
-0.425 -0.723 0.435 -0.296 0.103 0.780 0.670 0.730 -0.701 0.396
-0.355 0.304 -0.332 -0.036 -0.031 -0.917 0.313 -0.534 -0.983 0.839 -
0.822 -0.163 0.778 -0.514 0.970 0.693 0.396 0.431 0.103 0.577
0.283 0.779 0.915 0.431 -0.635 -0.412 0.122 0.870 -0.471 0.368
-0.768 0.344 0.474 0.067 -0.566 0.327 -0.234 0.258 -0.769 -0.187
0.521 -0.552 0.147 0.464 0.409 0.216 -0.095 -0.030 0.081 -0.210
0.029 0.335 -0.746 -0.297 0.358 0.694 -0.576 -0.310 -0.959 0.287
0.081 0.669 -0.593 -0.965 -0.939 0.492 0.866 -0.308 0.701 -0.201
0.850 -0.736 0.117 -0.452 0.885 0.070 0.526 -0.577 0.251 0.332
-0.535 0.300 0.576 0.101 0.418 0.106 -0.297 -0.497 0.462 0.029 -
0.074 -0.934 -0.336 0.055 0.976 0.215 0.556 -0.095 0.361 0.348 -
0.622 -0.775 0.485 0.924 -0.858 -0.376 0.967 -0.633 -0.272 0.287
0.115 -0.387 0.075 -0.804 -0.030 0.287 -0.432 0.549 0.609 -0.034
-0.692 -0.451 0.912 -0.726 -0.276 0.570 0.359 0.729 -0.106 -0.353
0.239 -0.957 -0.436 -0.731 -0.896 -0.561 -0.203 0.243 0.229 -0.047 -
0.678 -0.498 0.633 -0.308 0.999 0.134 0.857 -0.534 -0.873 -0.541
0.652 -0.742 0.456 -0.215 -0.138 0.382 0.638 -0.863 0.723 -0.330
-0.075 0.959 0.063 -0.333 0.961 0.745 0.233 -0.378 -0.423 -0.621 -
0.520 -0.462 -0.209 0.711 -0.627 0.344 -0.647 0.141 0.328 -0.262 -
0.675 0.972 -0.723 0.958 0.308 0.280 -0.127 0.055 -0.176 0.427
-0.146 0.914 -0.900 -0.778 -0.575 0.322 -0.939 0.153 -0.208 0.423
-0.522 -0.553 0.273 0.168 0.023 0.433 0.373 0.673 -0.635 0.672
0.929 -0.112 -0.669 0.008 -0.156 0.008 0.143 0.815 0.931 0.329
0.708 0.803 0.911 0.023 -0.750 0.614 0.794 0.938 0.833 -0.558
-0.085 0.740 0.332 -0.900 0.841 -0.989 0.183 0.440 -0.129 -0.670
0.695 -0.693 -0.180 0.811 -0.054 0.444 -0.630 -0.783 -0.362 -0.190
0.773 -0.061 -0.642 -0.191 0.629 0.413 -0.242 0.597 0.653 -0.357
0.421 0.002 -0.379 0.715 0.161 0.950 0.115 -0.150 -0.588 0.729
-0.956 -0.665 -0.772 -0.179 0.535 0.333 0.509 0.033 -0.454 -0.925
0.303 0.454 0.309 0.293 -0.299 -0.360 -0.739 0.532 -0.728 -0.599
0.655 -0.101 -0.953 0.794 -0.819 -0.395 0.105 -0.356 0.910 0.633 -
0.114 0.080 0.039 0.977 0.516 0.222 0.230 0.158 -0.256 -0.060
0.694 -0.692 0.726 0.594 -0.972 -0.031 0.461 -0.114 -0.606 0.022
0.694 -0.378 -0.989 0.557 0.054 0.247 0.674 -0.091 -0.015 -0.721 -
0.955 0.085 0.936 0.030 0.453 0.293 0.263 -0.194 -0.137 -0.102 -
0.585 -0.439 -0.599 -0.945 0.432 -0.333 0.270 -0.581 0.706 0.125
-0.071 -0.859 0.508 -0.303 0.329 0.769 -0.642 -0.330 -0.776 0.915
-0.904 -0.444 0.981 -0.854 -0.875 0.610 0.165 -0.178 -0.417 -0.401

0.363 -0.144 -0.738 0.180 -0.255 -0.515 0.945 0.851 0.803 -0.273 -
0.276 -0.615 -0.994 0.216 0.877 -0.013 0.075 0.466 0.685 0.263
-0.806 0.843 -0.327 0.452 0.750 0.541 0.937 0.290 0.332 0.785
-0.818 0.320 -0.581 0.877 -0.266 -0.409 0.375 -0.977 -0.546 -0.588 -
0.936 0.879 -0.223 0.600 -0.020 0.195 0.860 0.498 -0.575 -0.242 -
0.589 0.259 0.901 0.746 0.165 0.869 -0.767 0.198 0.036 0.582
0.095 -0.803 -0.348 -0.391 -0.670 0.748 -0.288 -0.825 -0.388 -0.488
0.899 0.415 -0.048 -0.480 -0.117 -0.057 -0.200 -0.216 -0.716 0.745 -
0.587 -0.620 -0.903 0.040 -0.878 -0.140 -0.276 -0.387 0.938 0.786 -
0.748 0.008 0.262 0.220 0.277 -0.050 -0.877 0.451 -0.925 0.177
0.539 -0.331 -0.133 -0.689 0.820 -0.456 -0.869 0.123 0.942 -0.631
-0.967 -0.537 0.386 0.699 0.813 -0.441 0.517 -0.052 -0.414 0.280 -
0.603 -0.945 0.777 -0.513 -0.449 0.903 -0.497 0.101 0.670 -0.261 -
0.426 -0.605 -0.786 0.999 0.891 0.241 0.043 0.747 -0.397 0.194
0.538 0.068 -0.308 -0.822 0.778 -0.511 0.070 -0.529 -0.026 -0.132
-0.545 0.517 -0.239 -0.738 0.975 0.286 -0.273 0.409 -0.432 -0.415
0.633 -0.250 0.306 0.585 0.145 0.214 -0.693 -0.816 0.288 -0.517
0.962 0.991 0.212 -0.962 0.449 0.870 0.318 0.206 0.352 0.226
-0.831 -0.082 -0.492 -0.107 0.678 0.881 -0.679 0.444 -0.479 -0.269
-0.710 0.570 0.938 0.616 -0.679 0.204 0.144 -0.673 0.113 0.444 -
0.377 -0.959 0.344 -0.159 0.227 -0.252 -0.619 0.944 -0.580 -0.002 -
0.530 -0.479 0.702 -0.304 0.294 -0.287 -0.609 0.165 0.517 0.695
0.180 0.027 0.130 -0.112 -0.122 -0.922 0.817 0.026 0.940 -0.772
0.643 0.521 0.752 -0.185 -0.553 -0.806 -0.944 0.861 -0.889 0.983 -
0.935 0.511 0.997 -0.035 0.410 0.220 0.357 0.982 0.195 -0.138
0.393 0.196 0.266 -0.098 0.925 -0.027 0.068 0.520 0.540 -0.060

E.2.4 Input file of simulation numbers

Numbers to be added to each set of files created by the system creator program.

```
! simulation numbers
0001
0002
0003
0004
0005
0006
0007
0008
0009
0010

etc.
```

Appendix F

Sample from database of simulation results – ecosystem initial state, ecosystem final state, and values for applied measures. See Chapter 6 for more information.

Sim #	$N1_0$	$N2_0$	$NTOT_0$
1	10	2	12
2	2	2	4
3	19	5	24
4	8	5	13
5	11	8	19
6	1	2	3
7	15	2	17
8	18	3	21
9	4	4	8
10	18	7	25
11	11	3	14
12	18	8	26
13	17	3	20
14	2	2	4
15	11	12	23
16	7	8	15
17	2	2	4
18	15	11	26
19	13	3	16
20	19	6	25
21	14	2	16
22	1	2	3
23	16	7	23
24	3	2	5
25	9	3	12
26	14	3	17
27	7	3	10
28	1	2	3
29	10	5	15
30	8	3	11
31	11	5	16
32	5	6	11
33	12	6	18
34	15	9	24
35	9	2	11
36	6	4	10
37	12	5	17
38	2	2	4
39	13	11	24
40	17	13	30
...	**...**	**...**	**...**
minimum:	1	1	2
maximum:	19	15	30
average:	9	5	14

SPP_0
20, 1, 17, 11, 12, 2, 6, 19, 7, 4, 31, 32
16, 5, 30, 22
8, 9, 20, 1, 4, 16, 11, 13, 10, 3, 12, 7, 6, 17, 14, 18, 15, 5, 19, 24, 35, 33, 40, 30
4, 10, 13, 18, 15, 14, 17, 9, 33, 32, 40, 34, 22
15, 10, 6, 1, 17, 8, 7, 11, 2, 13, 9, 27, 21, 23, 36, 38, 33, 34, 28
10, 26, 32
8, 19, 5, 18, 10, 20, 9, 15, 13, 12, 14, 17, 2, 1, 7, 23, 28
4, 20, 16, 11, 8, 2, 3, 6, 10, 9, 15, 12, 14, 5, 7, 1, 18, 19, 22, 21, 31
13, 10, 4, 9, 27, 28, 21, 39
9, 8, 16, 11, 2, 12, 6, 5, 1, 14, 15, 4, 17, 10, 18, 7, 13, 20, 40, 23, 34, 28, 37, 39, 21
4, 3, 6, 12, 8, 20, 19, 15, 13, 2, 11, 31, 33, 40
8, 5, 15, 4, 6, 12, 1, 16, 19, 13, 2, 7, 18, 14, 9, 3, 11, 17, 38, 21, 24, 35, 28, 33, 22, 31
9, 19, 15, 8, 5, 16, 6, 11, 18, 10, 1, 14, 13, 12, 20, 7, 3, 26, 22, 28
20, 11, 34, 21
5, 14, 18, 16, 6, 19, 17, 10, 13, 11, 2, 25, 23, 33, 22, 36, 30, 32, 28, 37, 31, 39, 38
18, 19, 14, 20, 15, 16, 4, 28, 29, 31, 23, 34, 30, 40, 36
8, 20, 23, 37
11, 19, 14, 4, 9, 6, 13, 15, 5, 7, 3, 1, 8, 10, 20, 32, 39, 28, 38, 40, 26, 21, 35, 31, 27, 24
6, 15, 1, 7, 8, 14, 10, 2, 5, 20, 4, 17, 13, 33, 25, 28
17, 19, 2, 15, 7, 14, 8, 5, 4, 16, 6, 20, 1, 10, 18, 3, 12, 11, 13, 34, 28, 23, 35, 37, 24
12, 20, 1, 16, 18, 6, 14, 11, 5, 4, 19, 15, 10, 3, 30, 35
10, 21, 22
10, 14, 2, 20, 6, 12, 8, 15, 7, 19, 13, 4, 18, 17, 11, 1, 32, 34, 30, 39, 29, 40, 38
13, 6, 4, 21, 29
4, 8, 11, 7, 13, 20, 3, 15, 19, 29, 21, 33
11, 14, 1, 5, 13, 9, 12, 15, 20, 6, 7, 2, 8, 18, 32, 25, 26
2, 4, 17, 1, 3, 13, 8, 37, 24, 25
13, 21, 31
6, 15, 8, 14, 5, 20, 1, 13, 17, 10, 32, 39, 40, 30, 36
2, 4, 11, 19, 14, 6, 12, 15, 25, 23, 40
7, 11, 12, 14, 15, 1, 10, 2, 5, 17, 4, 39, 28, 30, 34, 26
4, 14, 17, 13, 18, 40, 28, 29, 30, 38, 23
10, 6, 3, 7, 15, 5, 11, 16, 4, 18, 19, 1, 31, 32, 33, 22, 36, 29
15, 9, 4, 2, 19, 20, 8, 11, 7, 6, 12, 14, 13, 5, 16, 33, 30, 36, 25, 34, 27, 26, 28, 40
20, 9, 3, 17, 15, 7, 19, 13, 12, 28, 39
3, 8, 9, 5, 7, 10, 33, 21, 34, 32
15, 3, 1, 2, 5, 9, 18, 12, 4, 11, 20, 16, 40, 38, 28, 33, 34
8, 16, 23, 24
6, 4, 10, 5, 18, 2, 8, 15, 12, 11, 9, 19, 17, 30, 40, 29, 32, 39, 36, 28, 26, 38, 25, 22
4, 1, 10, 9, 18, 3, 8, 20, 17, 2, 13, 19, 16, 11, 5, 14, 6, 25, 39, 40, 23, 29, 33, 37, 32, 21, 31, 36, 22, 27
...

$N1_n$	$N2_n$	$NTOT_n$	SPP_n	P_{N1}	P_{N2}	P
2	0	2	20, 17	0.20000	0.00000	0.16667
1	0	1	16	0.50000	0.00000	0.25000
4	0	4	20, 10,17,14	0.21053	0.00000	0.16667
3	0	3	10, 14,17	0.37500	0.00000	0.23077
2	0	2	10, 15	0.18182	0.00000	0.10526
0	0	0	none	0.00000	0.00000	0.00000
4	0	4	10, 20, 14, 17	0.26667	0.00000	0.23529
3	0	3	20, 10, 14	0.16667	0.00000	0.14286
1	0	1	10	0.25000	0.00000	0.12500
4	0	4	14, 17, 10, 20	0.22222	0.00000	0.16000
5	0	5	6, 8, 20, 19, 2	0.45455	0.00000	0.35714
2	0	2	14, 17	0.11111	0.00000	0.07692
3	0	3	10, 14, 20	0.17647	0.00000	0.15000
0	0	0	none	0.00000	0.00000	0.00000
3	0	3	14, 17, 10	0.27273	0.00000	0.13043
3	0	3	19, 14, 20	0.42857	0.00000	0.20000
2	0	2	8, 20	1.00000	0.00000	0.50000
3	0	3	14, 10, 20	0.20000	0.00000	0.11538
4	0	4	14, 10, 20, 17	0.30769	0.00000	0.25000
4	0	4	17, 14, 20, 10	0.21053	0.00000	0.16000
3	0	3	20, 14, 10	0.21429	0.00000	0.18750
1	0	1	10	1.00000	0.00000	0.33333
7	0	7	10, 14, 20, 6, 8, 19, 17	0.43750	0.00000	0.30435
1	0	1	6	0.33333	0.00000	0.20000
3	0	3	8, 20, 19	0.33333	0.00000	0.25000
6	0	6	14, 20, 6, 2, 8, 18	0.42857	0.00000	0.35294
1	0	1	17	0.14286	0.00000	0.10000
0	0	0	none	0.00000	0.00000	0.00000
4	0	4	14, 20, 17,10	0.40000	0.00000	0.26667
4	0	4	2, 19, 14, 6	0.50000	0.00000	0.36364
3	0	3	14, 10, 17	0.27273	0.00000	0.18750
2	0	2	14, 17	0.40000	0.00000	0.18182
4	0	4	10, 6, 18, 19	0.33333	0.00000	0.22222
6	0	6	2, 19, 20, 8, 6, 14	0.40000	0.00000	0.25000
2	0	2	20, 17	0.22222	0.00000	0.18182
1	0	1	10	0.16667	0.00000	0.10000
3	0	3	2, 18, 20	0.25000	0.00000	0.17647
0	0	0	none	0.00000	0.00000	0.00000
2	0	2	10, 17	0.15385	0.00000	0.08333
...	**...**	**...**	**...**	**...**	**...**	**...**
94	2	94	**survived** **minimum:**	0.00000	0.00000	0.00000
6	98	6	**failed** **maximum:**	1.00000	0.12500	0.66667
			average:	0.34957	0.00236	0.20750
			full successes:	8	0	0

	Conn	Frac Dim
	0.5909	1.1451
	0.1667	1.1132
	0.2870	1.1567
	0.4000	1.1544
	0.3958	1.1459
	0.7500	1.0459
	0.4688	1.1494
	0.5000	1.1550
	0.2500	1.1505
	0.3155	1.1495
	0.3590	1.1426
	0.3900	1.1421
	0.5263	1.1514
	0.8333	1.0658
	0.3636	1.1491
	0.4643	1.1506
	0.1667	1.1472
	0.3709	1.1479
	0.6444	1.1491
	0.3403	1.1506
	0.1667	1.1457
	0.0000	1.1507
	0.4156	1.1419
	0.3750	1.1405
	0.6364	1.1453
	0.5417	1.1460
	0.4074	1.1457
	0.5000	1.1349
	0.2571	1.1449
	0.2333	1.1501
	0.3733	1.1503
	0.4000	1.1471
	0.4216	1.1506
	0.3865	1.1461
	0.3500	1.1472
	0.4722	1.1556
	0.3750	1.1411
	0.5000	1.0484
	0.3755	1.1434
	...	**...**
minimum:	0.00000	1.03180
maximum:	0.83333	1.15670
average:	0.36110	1.13976

Appendix G

Screenshot of case base with the Induce-It program active in Excel. See Chapter 6 for more information about the case base and Induce-It.