OF CATERPILLARS AND MODELS

by

Richard Terrell Steinhoff

B.Sc., Colorado State University, 1970

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

COMPUTER SCIENCE

We accept this thesis as conforming
to the required standard

The University of British Columbia

October 1975
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representative. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date October 9, 1975
An approach to modeling in general, its background, and its implementation can be described by examining its application to a particular system. The Western Tent Caterpillar is a productive subject of study because there is extensive data available for determining the relationships within the system. Individual differences between caterpillars and the weather are the dominant factors involved. They are incorporated into a set of ordinary differential equations that express the rate of change in the caterpillar population over time. These equations are numerically integrated to calculate the population at some terminal time. The results over a six year period agree reasonably well with observed field data. The three dimensional plot of the population gives a visual appreciation for the distribution over time.
# Table of Contents

1. Introduction ............................................. 1

2. The Background - A Blending of the Sciences .......... 4
   2.1 The Levels of Application .......................... 4
   2.2 Operations Research ................................. 5
   2.3 Biology ........................................... 7
   2.4 Mathematics and Computer Science ................. 9
   2.5 The History of the Tent Caterpillar Model ....... 15

3. The Set - The Tools for This Particular Problem ..... 18
   3.1 Spatial Boundaries ................................ 19
   3.2 Components of the System .......................... 19
   3.3 State and Driving Variables ....................... 20
   3.4 The Independent Variable .......................... 21
   3.5 Continuous or Discrete ............................. 21
   3.6 Stochastic Elements ............................... 24

4. The Script - In Retrospect .............................. 26
   4.1 Determination of the Major State Variables ........ 27
   4.2 Description of the Model ........................... 32
   4.3 Determination of the Parameter Values ............. 34

5. Validation ................................................ 40

6. Conclusions .............................................. 49

7. Bibliography ............................................ 51

8. Appendices .............................................. 52
   8.1 Numerical Methods .................................. 52
   8.2 Newton-Raphson Routines ........................... 53
   8.3 Constrained Targeting Routines ..................... 59
   8.4 Caterpillar Model Code ............................. 125
LIST OF FIGURES

I - State Variables In Average Weather ..................38
II - Derivatives Of State Variables In Average Weather ...39
III - Comparison Of Real And Simulated Populations ......42
IV - Weather And The Simulated Caterpillar Population ....44
V - Caterpillar Distribution In Mild Weather ...............45
VI - Caterpillar Distribution In Harsh Weather ............46
VII - Caterpillar Distribution In Mild Weather ............47
VIII - Caterpillar Distribution In Harsh Weather ..........48
ACKNOWLEDGEMENTS

Anyone who has written a masters thesis knows the large debt I owe to my advisors and friends for guiding and encouraging me through this endeavor. Roger Hart and Ilan Vertinsky have been helpful beyond anything I had expected. John Coulthard and Mark Dionne supported my graphic display efforts. Without them and the facilities available at the University of British Columbia Computing Center, this would not have been possible. Bill Webb's documentation program has made the laborious part of writing this just bearable. Since Ginny and I are one in all we do as husband and wife, she is the unsigned co-author of this thesis and fully deserves her half of the credit! The data used in building the model was due to the field work of Bill Wellington and organization of Bill Thompson. Together with Peter Cameron and Abe Lansberg, they kept my biology straight, my nose in the wind, and my eyes on the horizon.
1. INTRODUCTION

Man is constantly trying to formalize the rather nebulous and complex world around him so that he can understand it, and manipulate it. This formalization, in the broadest sense of the word, may take many forms: everything from poetry to detailed mathematical proofs. One of the intermediate forms is modeling - the art of defining a transformation from one space into another one, such that an entity in the original space has, in some sense, an "equivalent" entity in the second space which lends itself more easily to rigorous examination. By "equivalent", I mean "responds to a given set of stimuli in a similar manner - both quantitatively and qualitatively." Mathematical modeling means that the transformation is from the physical world into a mathematical representation of it. This requires an explicit statement of the relationships that exist within the entity being modeled.

My particular application of modeling is the western tent caterpillar. This was for a variety of reasons. One of the deciding reasons was that that area of application had to be meaningful to the world around me. Research in some areas would not only be useless and redundant, but totally incapable of being validated. When I found that there was some interest, and indeed some exciting progress in modeling tent caterpillars, that became the area of study. There were several other advantages in making this choice. In biological modeling one of the biggest hindrances is the lack of data on some of the important relationships in the system - missing
jigsaw pieces, if you will. This lack is not due to any biological oversight, but rather to the fact that such data, before modeling came of age, was too massive and complex to be assimilated by an individual where as it is easily "digested" by including it in the description of the overall model. The advantage with the tent caterpillars was that H.G. Wellington [6] has devoted a great deal of time in establishing the necessary relationships needed to describe the tent caterpillars.

This thesis will have three major thrusts. One feature of the thesis as a whole is that it will present a step by step approach to biological modeling from the computer scientist's viewpoint. I will indicate in the first chapters the general disciplines and tools needed to attack this area of research with special attention given to digital simulation. I was fortunate to be in a department with some flexibility and a vision for the wide range of applications of modeling. By enrolling in courses in other fields, I have come to some conclusions as to what others would find helpful if they were to study bio-simulation. I therefore submit these in the hope that others may learn by this experience. Although I will be discussing particulars in the latter chapters, I hope that the approaches outlined there will be of some use in other applications.

The second goal is to relate the various types of modeling and itemize their strengths and weaknesses to a variety of problems. From that I will present the important
characteristics of tent caterpillars and describe the type of modeling I chose for them.

Finally, I will discuss my model in detail, with special attention given to how the variables and constants of the system were derived and a presentation of the evidence supporting the hypothesis that this is a realistic model.

In retrospect, it's interesting that caterpillars should be the particular application of modeling that was most promising after examining a number of possible areas. The metamorphosis that the caterpillar undergoes in changing into a butterfly or moth has been the analogy used for rebirth. Jesus told a first century scholar, "you must be born again" (JOHN 3:3) and to the scholar that was completely incomprehensible, just as the caterpillar cannot fly until it becomes a completely new creature and transcends the limitations it had as a caterpillar. Wellington expressed that need in research "at the border of the unknown, one must consciously strive to escape from the mesh of former frames of reference, and to remain outside the generally accepted range of opinion concerning one's problem." The caterpillar has the key to a spiritual truth which is completely inapproachable by scientific investigation. Unless we escape our traditional "scientific" frame of reference when we come to explore the unknown side of our being - our spirits - we will never possess our wings to carry us above our reason!
2. THE BACKGROUND - A BLENDING OF THE SCIENCES

Mathematical modeling may be approached on one of a variety of levels and incorporate a wide spectrum of disciplines. In this chapter I will discuss the levels and the variety of skills from both the biological and physical sciences necessary to deal with the problems presented at the various levels. The history of the tent caterpillar study is also included.

2.1 THE LEVELS OF APPLICATION

In the attempt to build an overview of the structure these levels define, let me indicate the two extremes. The most detailed level is the study of the relationships between component parts of the system. In a biological environment (rather than, for example, a bio-chemical one), this would usually mean the quantitative effect of one biological element upon another. For example, the mountain lion population in a given area effects the deer population that area. This effect could be expressed as either a death rate or as a discrete event, that being the death of a deer. On this detailed level, the study would center on a particular aspect of the effect the mountain lion has on a deer. For example, it might focus on the effects on its feeding habits and model that relationship. This is in contrast with modeling the deer and mountain lion population as a whole.

On the other extreme, the macro level, the model as a whole could be used to make management decisions about the system, based upon certain constraints and goals to be
satisfied. A fish and game department might decide upon the number of game permits to be issued by finding an "optimal" strategy to harvest the deer. This strategy could be formulated and tested in a model.

The intermediate levels usually try to study overall trends in the system by reaching a fairly tenable description of the system and then altering the environment in which the system is operating. A particular model for deer may be subjected to a variety of catastrophic conditions (such as forest fires, epidemics, etc) and the recovery process examined.

The following chapter will discuss this intermediate level and the detailed level, since they were the areas of concern for this thesis. Let me briefly describe the least detailed level and the kinds of research in that area.

2.2 OPERATIONS RESEARCH

This macro level is known most frequently as operations research. It was born during World War II as a result of the problem of where to place a finite number of radar antennae such that the enemy airplanes could be most efficiently detected. Operations research is usually employed for the expressed purpose of making management type decisions. As a rule it does not require an in-depth understanding of the model itself, but depends upon it to react in the same way that the real life system would to external stimuli. From these reactions it forms a set of policies to be followed that will cause the system to behave in a specified manner. These
policies might either be of the yes-or-no variety, or they might be of the quantitative type - harvest so many individuals. A further complicating factor is the discrete nature of some decisions; one cannot harvest 1.5 individuals. The policies may take the form of simple constant terms (harvest 400 deer) or may depend upon the current state of the system (how many deer are present, how many mountain lions etc). These dependent policies are usually expressed as a weighted sum of the dependent variables or as a table of points which represent the given policy plotted against the possible state of the system.

Anyone interested in this aspect of modeling should have some background in operations research and familiarity with linear and non-linear programming and optimization theory. The material in this area describes a process to optimize some performance function subject to a set of constraints, either on variables in the system or on the control parameters. This usually requires the construction of a Jacobian (a matrix of partial derivatives of the constraints and performance index relative to these control parameters) and a set of operations on it. Hence, a strong background in

Courses at the University of British Columbia that would be useful for those interested in Operations research are:
- COMMERCE 310, 410, 411, 581, 582, 584, 586, 590, 681, 682; COMPUTER SCIENCE 406; ELECTRICAL ENGINEERING 467, 485, 555, 557, 568, 570; FORESTRY 331, 559; MATHEMATICS 140, 151, 156, 221, 321, 362.
matrix theory and vector analysis is also desirable.

2.3 BIOLOGY

There are some general background areas in biology that are helpful in modeling ecosystems. Obviously, the particular biology for a given system must be fully understood before the important components of the system can be isolated and their interaction can be formalized. It is vital that the computer scientist confers with a biologist who is knowledgeable in the problem. Indeed, one of the side effects of modeling that will be discussed latter is that new directions of study are needed to complete the unknown relationships discovered during the formalization of the model. Formalization often points out unsubstantiated assumptions in any discipline.

The general biological background that would be useful to a computer scientist in the understanding an ecosystem, would start with a basic understanding of botany and zoology, and then continue with advanced work which focuses on specific ecosystems. Special attention should be given to the transfers of energy and mass between components in the system and chemical cycles. This exposure to the complicated variety of relationships - along with a feeling for the relative importance of them is the real basis for building a biological
At this point - the formalization of the "biology" in the model - one has to confront the rather obvious question of the ability of a computer scientist to deal with some issues that are biological in nature. Science is a spectrum of knowledge in which each of the basic disciplines overlaps. In the purest form, a mathematician or computer scientist is a craftsman - turning out tools for understanding and manipulating the world around us. As soon as he becomes involved with a particular application, he begins to become a biologist, engineer, or social scientist. It has been argued that there is no "practical" need for the purist, but that neglects the contribution of the tool itself, its study, improvement and upkeep. So the computer scientist and mathematician can give to a biologist some well honed tools - numerical analysis, mathematical model construction, and simulation languages which incorporate the whole gamut of theoretical design and programming technique. It must be stressed that this is a far cry from the skills traditionally associated with a computer programmer, in the same way that a mathematician has grown past the multiplication table stage of early civilization. The biologist and computer scientist,

Courses at the University of British Columbia that would be informative for computer scientists who want to apply their modeling to biology would be: APPLIED SCIENCE 250; BIOLOGY 301,321,322; FORESTRY 405,409,505; ZOOLOGY 401,421,527.
then, need a continual dialogue as they approach the problem of bio-simulation.

There is one additional advantage by having a computer scientist approach the biological problem. That is: he does not have any preconceived notions or conceptual frameworks to bias his view of the biological "facts". For example, May [3] has written one of the most comprehensive and accepted summaries of the state of the art biological modeling, and his background is in electrical engineering! The above comments should not be misconstrued to mean that the biological understanding is not a requirement for successful modeling, but merely points out the need for cooperation between computer scientists and biologists in the field of bio-simulation.

2.4 MATHEMATICS AND COMPUTER SCIENCE

Something should be said about the contributions to modeling and computer simulation by mathematics and computer science. I have divided the areas of effort into numerical analysis, general algorithms, theory of simulation, and language design. The skills mentioned in these areas should be regarded as minimal recommendations for any computer scientist who aspires to bio-simulation1.

1 Specific courses at the University of British Columbia that should be the basis for a curriculum in bio-simulation are: COMPUTER SCIENCE 115, 215, 315, 350, 405, 505, 535; ELECTRICAL ENGINEERING 591; MATHEMATICS 155, 255, 256, 301.
Numerical analysis has been one of the most important aids to successful modeling. For the continuous system, numerical integration and differentiation techniques have been improved, so that they are more accurate and efficient. They also have the ability to dynamically change the integration step size depending upon the non-linearity of the rate equations. The most advanced techniques in this area are Predictor-Corrector and Runge-Kutta methods with eigenvalue step size modification. Advanced work has been done in handling "stiff" systems of differential equations. The question of whether a system is "stiff" or not can generally be left until one actually begins to simulate the system. If it is a stiff system, there should not be any doubt, once the integration has begun, because the state variables will become very large in absolute value. Since eigenvalues are used to alter the step sizes during numerical integration, a set of eigenvalue routines for general and special matrices have been developed. Often it is necessary to incorporate table orientated data into continuous system simulation programs. Numerical analysts have developed curve fitting to approximate the values in the table with least squares algorithms and splines. Linear equations that arise in simulation problems require methods to solve that special type of system if it cannot be solved analytically, so mathematicians developed programming and integer programming. State of the art in this area is the Revised Simplex Algorithm by Danzig.

The class of general algorithms or program techniques
that are particularly useful in simulation includes sorting, hashing, and data handling. Hashing is the process of transferring large set of entities into a small set of classes such that the number of classes is in some sense optimal, and that roughly the same number of entities are likely to fall in each class. Once a hashing algorithm has been decided upon for a given set of entities, one can search the set for a particular entity more efficiently by first going to the appropriate class. Hashing would be useful in discrete simulation problems that require one to find a given individual amount a large number of similar individuals. For example, in modeling a employment agency, one might want to find a particular job applicant in a file of several thousand applicants. Sorting is the process of making an ordered list of entities. This could be done by a variety of methods such as binary search, bubble sort, and sort/merge. Sorting is useful in block diagram simulations where the blocks must be sorted before they can used to direct the simulation. Data representation is one of the areas of most interest in computer science. Several special types have already been examined and used in this thesis. Trees, lists, and multiple-linked structures are all very useful in simulation. The linked list used to connect occupied grid squares in the caterpillar model saved unnecessary integration time.

In the purest form, simulation has been studied from several theoretical aspects. Stability is a crucial question in modeling. The mathematician is able to determine the
stability of some systems analytically and the computer scientist has developed steady state techniques to isolate transients in the system and determine their ultimate effect. Deadlock is another area of study. This is the state where two or more processes freeze uncompleted because neither can proceed until the other process is completed. As an example of this, consider the traffic situation where two cars want to turn left just after they pass each other, one coming from the east and one coming from the west. A line of cars begins to form behind both cars. Both cars are prevented from turning by the other car's lineup, and that lineup will not move until they turn. The question of deadlock is more prevalent in a discrete simulation, and involves a probability rather than an absolute state of deadlock or non-deadlock. Validation, which is also a theoretical area of computer science, will be discussed in chapter V.

The last area of computer science I want to describe is language design and implementation. That is an important subject given the following two facts: the computer is merely a very fast and accurate information processing machine with a memory for storing the input data, the results, and the instructions for which operation to do next, all of which ultimately are represented as numbers; humans recognize and think in symbols and nested processes relevant to a given problem or a class of problems. It is therefore important to have a language in which the problem (which in this case is simulation) can be easily formulated and for which there
exists an unambiguous mapping between that language and the computer's internal language (numbers).

A detailed history of computer languages is out of the scope of this thesis. The interested reader is referred to Jean Sammet's book [3] on programming languages and their history. In computer science, languages evolve in a survival of the fittest environment. Old and unused languages are usually replaced by more powerful ones. There is a certain amount of inertia in the system, but eventually, the poorer languages tend to die out. What follows is a subjective judgement on the most useful languages for simulation. No doubt in the near future people will read these lines and smile at the primitive languages suggested.

There are four relevant types of languages for simulation: general, discrete, block diagram, and continuous. The first category are languages for a broad range of scientific applications. ALGOL, and its predecessor FORTRAN, are the two most widely used. The design criteria for ALGOL-68 (which is an improved version of ALGOL) has been suggested, but the generality and complexity of the language have made it difficult to implement and there are to date no complete versions available. SIMULA-67 contains not only the abilities of ALGOL, but co-routines which could be used to simulate parallel processes. Unfortunately, it is not very widely used in comparison, but as it comes into its rightful appreciation by the scientific community, it will be used more and therefore be the best choice for a general language.
Discrete languages would be used for discrete simulation which will be contrasted with continuous simulation in chapter III. The most useful languages in this class are SIMSCRIPT and GPSS. GPSS is a block orientated language, the most logical, and the easiest to learn. SIMSCRIPT is a statement orientated language and more flexible than GPSS. SIMULA-67 also contains discrete simulation facilities.

Block orientated languages (the predecessor of the continuous system simulation languages) are used for a class of problems originally handled by an analog computer. The best of this class in common use is DSL/90. Its predecessors were MIDAS and MIMIC.

Finally, continuous system languages in common use are DYNAMO and CSMP, the latter being far and above the most widely used. For simple problems, these are sufficient, but there is still a need for a more powerful, interactive continuous system simulation language. A report on the desirable features of such a language was produced in 1967 [5]. This CSSL report specified design objectives that included the better features of CSMP, DSL/90, GPSS, and ALGOL. It also made an attempt to extend the flexibility of these languages to user defined operators and control structure. It promised graphic display of output and a conversational mode during the simulation. A full blown implementation has not been forthcoming, but the objectives are clear and one would expect in the near future to see a third generation language: CSSL.
2.5 THE HISTORY OF THE TENT CATERPILLAR MODEL

While in the general overview of modeling, it is helpful to look at some examples of applications of modeling. The range of modeling goes from political or social science and armament in developing countries to nuclear physics and control of the fission process in reactors. Its domain goes from energy cycles within the cell to examining the big bang theory in space.

The caterpillar problem itself has some history that should be related here. The Western Tent Caterpillar, *Malacosoma pluviale* (Dyar), has a population that fluctuates quite a bit from year to year. On the Saanich Peninsula of southeastern Vancouver Island, the population varied between several thousand to several million individual caterpillars over a period of ten years [10]. That increase and decrease over the years became the area of study in 1955 for several entomologists [6] and the subject of many subsequent papers in population dynamics [8][9][1]. The initial question that was asked was "why don't the caterpillars continue to increase without bound?" At the time that the question was first asked, the leading theory among those who study population dynamics was that populations were controlled by density dependent "regulators" in the ecosystem. These regulators might be predators of the population or the food source of the population. The idea is that as the population changed, there was a corresponding effect upon the predators or food source. If, for example, predators increase by some exponential
function of the number of prey, the prey would eventually cause enough predators to be around so that the prey are eaten faster than they can reproduce. The prey population would therefore be controlled. The key word here is dependence. Wellington [8] was just beginning to look at the tent caterpillars in 1948 with the idea of studying the effects of weather on its population. The more he observed them, the more he became convinced that density dependent regulation was not the only factor in the oscillation of populations. In the first place it is very clear that individual animal behavior and activity influences its ultimate survival. He succeeded in showing the variation in activity level within the tent caterpillar population and demonstrated its effect on overall colony survival. Further, although predators were few and food sources were usually unlimited, the caterpillar population did reach a peak in 1956 and began to decline. It had been his conviction for some time that weather was a very large determining factor in caterpillar mortality, and this became more substantiated as his studies progressed.

So there were two major break-throughs in this study: individual traits effecting overall population growth, in addition to weather, which is a density independent factor. Most of his observations between 1955 and 1965 exploited this new knowledge and were consequently quite helpful in constructing a model which incorporated these facts. His vision for the important relationships within the caterpillar population was the reason for the unusual amount of data which
now can be used to model the caterpillar ecosystem.

Subsequently, P. Cameron and W. Thompson et al [7], began to build a model for the caterpillar population. They had the initial task of transforming the field data on micro climate (local climate), tree height, egg masses, and initial populations to a form that could be readily used in a computer program. Their work in that area was invaluable to this thesis. Once that was done, they completed a model for the caterpillar system. Basically, it was a continuous simulation, although the rate equations were difference equations and the death rate was probabilistic. It was written in IBM 360 assembler so that each colony was handled as a separate sub-model. The results from that model are described in a latter chapter.
3. THE SET - THE TOOLS FOR THIS PARTICULAR PROBLEM

Once an application is decided on a dialogue should begin between the computer scientist and the biologist who has studied that particular system. There is a logical set of questions that must be answered before the model can be constructed. In this chapter, these questions will be taken in order and the ramifications of the answers pointed out in addition to explaining what the answers were in this particular study.

The processes of questioning and dialogue is obviously an iterative one. Following an initial set of discussions with W. Thompson, during which we touched on all these issues, I proceeded to read publications on the Western Tent Caterpillars which brought up further points needing clarification. These were discussed with W. Wellington. Finally, I had a long afternoon with the chief research person, P. Cameron. Although there were several iterations, the following questions remain the basis for discussion.
3.1 SPATIAL BOUNDARIES

The first question to be answered is: what are the spatial and time boundaries of the system? In some ways this begs the later question of what are the independent variables, but as a rule it turns out to be either spatial coordinates or time. Of course the system may be a subsystem of some larger system - say the universe - but reasonable boundaries can usually be drawn by looking at the environment of the entity or process under study. For the caterpillar study, the spatial range was the Saanich peninsula and so spatially, the choice was rather easy. Since the model is interested in overall trends, the time interval has to be rather large, a factor several times larger than the basic generation time of one year.

3.2 COMPONENTS OF THE SYSTEM

The second question is answered by looking at the flow of energy or mass within the system and asking for the places where that energy or mass seems to pool. In the caterpillar model, we see the caterpillars themselves, their food source, their predators, the weather, their tents, and the caterpillar colony locations as the major components. Another way to view components is by the "process" which they are a part of. What do caterpillars do? They are laid as eggs, hatch, travel to and from egg mass to food source, eat, build a tent, rest, are eaten, expose themselves to weather, pupate, disperse and lay eggs. That, of course, is a very simplified statement of their activities. In actual fact, even their resting is a
complicated process, where the temperature, food intake, individual caterpillar characteristics, and position relative to each other, determine the length of the rest and consequently when the next feeding begins. In Chapter 4, I discuss how this list of components was pared down to a workable size of major elements in the system.

3.3 STATE AND DRIVING VARIABLES

From the list of system components, the third question regarding state variables arises. A state variable is one that uniquely determines one portion of the system's status for any given value of the independent variable. Hence all of the state variables together uniquely determine the particular state of the system as a whole at that particular instant. These state variables (or their derivatives) must be a function of some other variables in the system. In that sense, they are dependent variables. The only variable mentioned above for the caterpillar model that was not a state variable was weather. The next question to be considered is that of driving variables. A driving variable is one which is not a function of any other system variable except the independent variable. In this model, weather is the only driving variable. As I discuss in Chapter 4, weather is thought to be the strongest single influence on caterpillar activity and population size. Some systems have no driving variables - that is, each variable is a function of at least one variable besides time.
3.4 THE INDEPENDENT VARIABLE

In determining the independent variable, it is necessary to determine over what range the model's behavior is to be studied. As mentioned earlier, this is usually some period of time, but it can take other forms, such as a measurement of distance.

3.5 CONTINUOUS OR DISCRETE

Probably the question with the most ramifications is the one of continuity of the system being modeled. A continuous system is one in which the change in the system is ongoing at every instant, whereas in a discrete system, the change happens instantaneously at particular points in time. It has been said that the continuous is but the limit of the discrete and that the discrete is but the time slice picture of the continuous. That is, a system may involve the appearance and disappearance of individuals in the system, but as the number of individuals increases, the effect of one individual arriving or leaving the system approximates more and more a smooth continuous curve. Inversely, if one takes a smooth curve describing the population of a system over time, taking a sample of the system every change in time T where T becomes large, becomes a bar graph of individuals entering and leaving the system. All that is to say that the choice of continuous against discrete modeling depends more upon the magnitudes of the numbers involved, rather than the types of entities being modeled.

A rather controversial heuristic can be employed to make
this decision of continuity. With a few exceptions, applications in the biological and physical sciences would use continuous system simulations while those in commerce, transportation, and social sciences would probably use discrete system simulators. Additional insight in this matter can be gathered by comparing the advantages of continuous and discrete simulations for the particular application in question. Discrete systems are especially helpful in developing statistical information about the system: average delay, maximum and minimum delay, average process time etc. Discrete systems also deal with the well defined, complex, interactions between individuals in a system where "events", "activities", or "processes" take place. The important feeling one should have about "events", "activities", and "processes" is that they should be a well defined function between individuals in the system. In such a case, discrete simulation is the best approach. Continuous systems, on the other hand, deal with flow rates between segments of the system, and are not so concerned with statistics of the system, as they are with overall numbers in a given component. Continuous systems can deal with a large number of individuals, because the population is treated as a single number - the total population. Discrete systems can not deal with large populations, because they must handle each individual separately and for a large number, this requires too much time.

For the caterpillar model, the problem involves a large
number of individuals (approx., 1,000,000) and the system responses that are of interest are overall numbers of individuals. These two conditions caused me to choose a continuous model. As it turned out, one of the major features of the real world system was a periodic smooth curve that described the population over a long time span, so one naturally began to think in terms of continuous simulators.

The primary result of deciding on the continuity of the system, is the range of languages available for the simulation. If a discrete simulation was chosen, one could either use a general algorithmic language like ALGOL or SIMULA or a specialized discrete simulation language like GPSS. For the continuous system, the specialized language would be something like CSMP or BEDSOCS. Although these languages are more powerful than their predecessors, they are not available everywhere and are not understood by the majority of the scientific community. As I wanted to be able to run this program in any computing environment and allow others to use and modify it, I decided to use FORTRAN, which is far and above the most common scientific programming language, to simulate the continuous caterpillar system.
3.6 STOCHASTIC ELEMENTS

The final question relates to the stochastic nature of the system or the element of chance involved. For example, the rate of arrival of travelers at a border crossing is not constant, but random. The rate is therefore stochastic in nature and has a measure of uncertainty in it. This question is similar to the question of continuity in that the decisions are arbitrary rather than clear cut. Certainly in the real world, almost everything has an element of chance in it. Which deer a mountain lion may kill is, after the set of possible victims is established, a matter of which one it "comes upon" when it is hungry. The saving grace is that when large numbers are involved, one can sometimes ignore the stochastic elements. If most deer behave somewhat similarly, it doesn't matter which one of a large number of deer the mountain lion eats. The result is the same - the deer population decreases by one. Hence one could say one of two things: the probability of one deer dying on a given day is .01; the rate of mortality is approximately .01 deer/day. The deterministic statement is usually faster in simulation, and so if the difference between the two is small (as it was for the caterpillars), the deterministic model may be used. The first (and more detailed) model of tent caterpillars [7] by W. Thompson et al had many stochastic elements.

One additional clue in deciding on the stochastic characteristics is to see if the difference introduced by the chance element is magnified by some gross change in the state
at that point. Consider for a moment this problem when the females disperse at the end of the year. All year long the caterpillars are relatively stationary and reducing in numbers from several hundred per colony to a few per colony. Those caterpillars pupate and fly in some arbitrary (?) direction, some arbitrary (?) distance, and deposit their eggs—all 200 of them. Hence 200 caterpillars may be placed in any grid square. That large number of caterpillars "magnifies" the effect of chance in the migration of the moths to the point that I could no longer use a purely deterministic model. The migration direction and distance had to be stochastic.

A final personal bias—not entirely unfounded—is added here. As I have noted above, the real world is almost always discrete and stochastic. Consider the atom and random electrons which are the building blocks of matter to convince yourself of that fact. Nevertheless, it appears to me, a reasonable heuristic in modeling is that to strive for simplicity, continuity, and deterministic modeling. An optimistic engineer has been known to say "Nature is, on the whole, deterministic, continuous, and twice differentiable."
4. THE SCRIPT - IN RETROSPECT

I emphasize the title of this chapter. As is pointed out by Wellington's paper on approaches in population dynamics [12], the work published on a given subject area is almost always polished to the point that one reading it might assume that things naturally fell into place all the way along during the research and that the course of action was clear from the beginning of the investigation. That of course is never the case. If necessity is the mother of invention, then surely serendipity is the father. When I first decided upon caterpillars as my topic, I was going to study a detailed mortality model of the system, but as I pursued the various factors involved with mortality (such as random attack, starvation, non-random attack, desiccation and viral infection), I decided that the area that needed studying was a more general mortality model which had only a few terms dependent on the major state variables. So the work that resulted is seen here in retrospect not as a preset goal. In any application of modeling I would see a dynamic direction of effort that would change as new factors and understanding come to bear. One would begin to assemble the answers to the questions in the previous chapter, and as the model takes shape, one would see questions that require further study. Perhaps the whole model would be scrapped in favor of studying an interesting subsystem.
4.1 DETERMINATION OF THE MAJOR STATE VARIABLES

This chapter's text naturally falls into three topics: i) the determination of rate equations for the major state variables, ii) the detailed description of the caterpillar model, and the determination of the parameters in those rate equations. Let me review in detail the sequence of events that lead me to choose differential equations to model the relationships in the tent caterpillar community. There is some data that could be used to construct some complex and well defined relationships between individual caterpillars that effect their activity, eating habits and loss to predators. As an example of this, there was a study done on four colonies of caterpillars. Two of the colonies had a high percentage of active individuals and the other two had a low percentage of active individuals. One of each type was placed in a harsh weather situation and one of each type was placed in a mild weather situation. Detailed records were kept on the types of mortalities; predation by spiders, desiccation, etc. From that kind of data one might be lead to construct a very detailed mortality model, and since the data on these relationships is fairly rich, one might even pursue a model for a individual colony. This could even be a discrete model, if only a single colony was studied. Two factors kept me from this approach. Primarily, I was interested in the caterpillar population as a whole (which as pointed out earlier, was far too large for a discrete simulation). That was more of a personal preference, but when coupled with the fact that
weather was felt to be the overriding factor in caterpillar population dynamics, the scales were tipped in the direction a macro, continuous model.

In my discussions with Thompson, Cameron and Wellington, weather was alluded to again and again as the major controlling element. It is important to see that the biologist was the one to make this judgement. The computer scientist is required merely to confirm the supporting arguments for this judgement. In the case of caterpillars, the supporting evidence was more along the negative lines as to which things did not effect the population. In most biological systems, food supplies and predator populations regulate the prey population, but in this system it was not the case. Even in the most populated years, the food supply was almost unlimited. In a very few cases did a caterpillar colony totally defoliate its host tree. There was some starvation, of course, but that appeared to be from the inability to forage from food. There were also a few caterpillar predators. Wasps, spiders, ants and several insect parasites all prey on caterpillars, but usually they did not have a significant effect. Besides that, the predator and parasite populations were nearly independent of the prey populations because they had alternate hosts in years of low prey populations. The general feeling, then was that weather played the biggest role in controlling caterpillar populations.

Weather is a difficult thing to analyze quantitatively.
Most of its effects are indirect. Bad weather may keep caterpillars in their tents away from food and thereby kill them by starvation. Good weather may allow the caterpillars to build new tents, leaving the weaker, virus infected members of the colony behind in the previous tents and isolating what could have been an epidemic. Again, the effect is indirect.

Because these effects are indirect, one can only model them in general, and not by a direct, measurable relationship. This is where differential equations become quite helpful. Suppose a simple relationship between weather and the mortality could be expressed in differential equations that would fit the data for actual caterpillar systems. Even though the differential equations would be artificial, it could express these indirect relationships and would apply the effect uniformly in the system over time. As noted earlier, the overall system responses seem to be continuous. So I had answered two basic questions: the type of equations and the driving variable.

Next, I looked back over the early work of Wellington. Surely, his biggest contribution to biology was that individuals behavior, or rather types of individuals and their behavior, affect the overall characteristics of the ecosystem. The active individuals in the caterpillar colony lead the rest of the colony to new foraging areas. They are the first to terminate the rest periods of the colony. They also influence the number and shape of the tents constructed by the colony. Their dispersal distances as moths are much longer. The sluggish members of the colony retard its activity, but they
do reinforce its tent construction. What is the simplest statement that could be used to model these differences? One possibility would be to divide the caterpillars into groups by type. In the studies on the Saanich Peninsula, this is exactly what was done. The four activity groups decided upon were really very artificial. In reality, a spectrum of activity existed, ranging from those that were not even able to break out of their eggs, to those that were very independent and constantly active. Still, an important point can be made here. When faced with a population that is quite diversified in its behavior, one must try and divide it into prototypes and then model these groups of individuals as separate populations, so that the differences in individuals can be incorporated. This model required four rate equations, one for each type of individual. The four populations corresponding to these four types became the state variables of the system. Once the differences were established, the interaction between them was chosen to be a constant times the percentage the number of most active individuals were of the whole.

The one remaining factor to be incorporated in the state equations was the independent variable, time. It turns out that the mortality rate is not constant. Caterpillars shed their skin, and their age (instar) is measured by the number of times that has occurred. The mortality rate is quite high in early instars but as the caterpillar grows, it becomes less susceptible to many predators and diseases, such as ants which
can only attack young caterpillars. In the last instar, the caterpillar becomes vulnerable to fly parasitism. So the natural thing is to add constant multipliers and time to the state equations.

It was then possible to write the simplest differential equations that express the four state variables in terms of themselves, weather \((W)\), and time \((T)\). \(P(i)\) is the population of the ith type of caterpillar. The vectors \(A, B, C,\) and \(D\) are the parameter values to be determined later. This gives the following set of equations.

\[
\frac{dP(i)}{dT} = (A(i) \times B(i) \times T + C(i) \times W + D(i) \times PCT) \times P(i)
\]

where \(i\) goes from 1 to 4

\(PCT\) is \(p(1)/S\)

and \(S\) is the sum of the \(P(i)\).

These four equations became the hub of my model. Routine INTGRT takes the current values of the state variables and time and calculates the derivatives directly from these equations. That routine could be called by any integration scheme to calculate the state of the system at a new time \(T' = T + DT\). In the first stages of development, I used Euler's integration method to numerically integrate a day at a time. See Appendix I for a description of the Euler algorithm.
4.2 DESCRIPTION OF THE MODEL

We now return to the spatial considerations discussed in chapter III. I wanted to study the spatial distributions of caterpillars on the Saanich Peninsula. The approach used was very similar to the one used to break down the caterpillar population into prototypes. Suppose the peninsula was to be broken down into grid squares, and each grid square was assumed to be homogeneous in micro climate, tree height, and general caterpillar occupancy. Each grid square would then have a different set of four populations of caterpillars, but the same four differential equations, except that the weather in each grid square would be altered by a function of the micro climate before using it in the state differential equations. Routine IHTGRT, then, takes the current values of the four caterpillar populations and the local weather for one grid square and returns the derivatives which are integrated ahead one day at a time for all the grid squares in routine STEP. The grid squares with nonzero populations are kept in a double linked list so that each square need not be searched each day to see if it must be integrated. The main program is quite straightforward. It calls READ to input all the initial population data. It calls INITIAL to initialize all the variables, and then calls STEP for as many days as is required to integrate the system to the end of the year. DSPRS is called to change the remaining caterpillars to moths, disperse them randomly, and lay new egg masses. A new population distribution is then printed out by OUTPUT and the
process begins again until the number of years of simulation required are produced.

DSPRS and OUTPUT actually require further explanation. The problem in DSPRS is that by this time the caterpillars have pupated and turned into moths, so they need to fly in some random direction and distance and then deposit a quantity of eggs. One solution is to pick a random number between 0.0 and 360.0, calculate the sin and cos of that number, and go in that direction a random distance whose range of values is determined by the type of individual (note the individuals effect again). Even the number of eggs laid and the types of individuals resulting from them are dependent upon the prototype of the mother. That solution is sufficient, but it can be speeded up by noting that the maximum distance covered by a moth under any circumstances is twenty grid squares. Hence, with a finite number of equal arched rays, each grid square could be visited. Picture a bicycle wheel on a checker board. If enough spokes are added, each square would be touched by at least one spoke. It turns out, that the magic number of "spokes" it takes to cover the twenty squares is twenty eight. Thus, only that many sines and cosines allow the moths to visit any square within its range of flight. This fact saves the calculation of a sin and cos for each moth (remember, there are sometimes several hundred thousand of them!) each year! It was decided that if a moth lands in the ocean or on an uninhabited grid square, it is assumed to die.

OUTPUT allows the user to output the yearly caterpillar
distribution, either as a printer two dimensional plot with each grid position given a character from 1 to Z depending upon its total population, or as a three dimensional graphic relief map with the "altitude" of the relief corresponding to the caterpillar population. If the relief map is displayed continuously as the population changes from year to year, one begins to get a visual feeling for the behavior of the system over a long period of time.

4.3 DETERMINATION OF THE PARAMETER VALUES

The next step of model development was the most difficult one: the determination of reasonable values for the parameters A, B, C, D, and E in the differential equations describing the population rate of change. As pointed out earlier, these parameters could not be determined from direct relationships in the field data. Weather and the effects each type of caterpillar has on the other types are indirect effects. I decided to take a couple of average colonies in average weather and harsh weather and determine the parameters that would result in the final colony populations being as close to the observed values as possible. That meant that I had a non-linear system of differential equations to be solved as a two point boundary value problem. Given an initial set of conditions, and the differential equations as a function of the parameters A, B, C, and D, the parameters are varied so that the resulting final population meets a given terminal condition. The number of degrees of freedom was equal to the number of parameters (twelve). The number of initial and
terminal conditions was the number of state variables, times the number of test colonies (eight). Actually, I even experimented with three test colonies, but that meant the solution (if it existed) was quite difficult to find. After some effort, I reduced the number of test colonies to two.

At first, I took the simple minded approach of "twiddling" each of the parameter values until the criteria were nearly satisfied. It became like stuffing a full inner tube into a bicycle tire. As soon as I poked it in one place, it popped out in another. I needed something more powerful.

The second try was with a Newton-Raphson iteration technique. Given a error vector (E) which is how far away each terminal condition was from a given initial guess of the parameters, and a sensitivity matrix (S) which is the partial derivatives of the errors with respect to the parameters, one can compute a correction vector \( \mathbf{U} = -S'(SS')^{-1}E \). This can be used to compute a new set of parameters, a new error vector, and a new sensitivity matrix. The process is used iteratively until the errors are within some tolerance of zero. To calculate E for any given values of the parameters, one needs numerically integrate the state differential equations for one year. To calculate S for given values of the parameters (P), one takes the nominal values P(0) and perturbs them one at a time and calculates the corresponding E vector. S is then the values \( (E(0) - E(j))/(P(0) - P(i)) \) where j varies over the number of errors and i varies over the number of parameters. The procedure begins by calculating E(0) from the initial
guess on the parameters $P(0)$. That requires integrating the differential equations forward to the terminal time and subtracting the terminal population values from the desired ones. Next, one at a time, the parameters are increased by a small constant value and the differential equations are integrated to the terminal time where the new error values $E(j)$ are calculated. As each parameter is perturbed and the corresponding $E(j)$ is calculated, a new row is added to the sensitivity matrix $S$. When the process is finished, the vector $U$ is calculated and new parameters are calculated from $P' = P + U$. A new error vector $E(0)$ is then calculated by integrating the differential equations again. If it is within some tolerance of zero, the process is completed and $P'$ is printed out. Otherwise, $P'$ becomes $P$ and the process begins again.

This Newton-Raphson approach appeared to work quite well, until I examined the intermediate values of the derivatives of the state variables. To my horror, they were positive at some points in time. Mathematically, that was quite reasonable, but biologically, caterpillars do not reproduce or spontaneously appear during the year. The population must always decrease from the time the eggs are laid until the time when they pupate and disperse at the end of the year.

My first modification was to force the derivatives to be negative by including the negative of their absolute value. That turned out to be too artificial, and the Newton-Raphson scheme began to diverge. I really needed some way of
constraining the direction of the correction vector and some intermediate values of the derivatives. If I had wanted equality constraints, I could have just added them to the list of eight I was already targeting for, but I needed inequality constraints. I didn't want to constrain the derivatives unless they started to go positive, and I didn't want to constrain the correction vector unless the correction direction would violate known biological facts. These inequality constraints forced me to use a program developed earlier with some engineers in trajectory shaping[2]. Now I had the tools necessary to attack the problem. After modifying this program extensively and playing with the constraints so that the derivatives behaved properly, I came up with the following values for the parameters.

\[
\begin{align*}
A(i) & = 0.009960 & 0.12000 & 0.30700 & 0.009350 \\
B(i) & = -0.000685 & -0.000126 & -0.00055 & -0.000542 \\
C(i) & = -0.000151 & -0.003850 & -0.01000 & -0.000289 \\
D(i) & = -0.001000 & -0.001000 & -0.00100 & -0.001000 
\end{align*}
\]

The reader is referred to Appendix II for the source code for the Newton-Raphson routines and Appendix III for the inequality constraint programs. This Figure I and Figure II show the time history of both the state variables and their derivatives.
FIGURE I - STATE VARIABLES IN AVERAGE WEATHER
FIGURE II - DERIVATIVES OF STATE VARIABLES IN AVERAGE WEATHER
5. VALIDATION

One of the hardest things to accept as a computer scientist was the statement that when a model produces results that "fit" known observed facts, that it hasn't proved the correctness of the model. In fact, a model can never be shown to be an exact mapping from the original system, no matter how similar it may behave. No matter how consistently a model seems to duplicate known facts, it has not proved its correctness. The best one can do is make a good case for the likelihood of the model being a reasonable one. To an absolute and objective computer scientist, that is humbling realization!! One can, however, show a model to be incomplete or erroneous by a counter example where the real system behaves differently than the model.

Even though it is not possible to prove the correctness of a model, there are some ways to validate it or show the likelihood that it will behave in most cases as the real system would behave. These methods of validation must all be non-circular; that is, the data that they use for comparisons must not have been used in constructing the model. If one uses the temperature history of the Fraser river from 1950 to 1975 in constructing a salmon modeling program, he would not be able to validate his model by using the comparison of how the model behaved to how the real system behaved in 1960, since that was part of the data he used to build or modify his model in the first place. Ask someone a question. Now give them the answer. Now ask them the question again, and it is
surprising how many times that person will have the answer.
The data used in validating a model must be as unrelated as possible to the data used to construct the model in the first place.

Validation is usually done by comparing the state of the model after some long change in the independent variable with the real system. As noted before, this interval should be different than any interval used in "fine tuning" the model. Additional validation is possible by comparing the model's reaction to changes in the driving variables with the reaction of the real system. This type of validation is sometimes called sensitivity analysis. It centers upon the overall trends in and the resilience of the system and its model. Resilience is the ability of the system to return to a steady state after it is perturbed by some external force. In the caterpillar model, both types of validation were used. Figure III shows the model's caterpillar population, the Thompson et al model's result, and that of the real population from 1959 to 1963.
These years of data were not used in building the model. The two were hard to compare, because the field study count was in number of colonies and the model's output were in actual caterpillars. The number of caterpillars per colony varies from year to year. Generally the number of caterpillars per colony increases as the weather improves from 40 to 90. The number of colonies might be wrong by a factor of two. Considering the simplified nature of the state equations, these were remarkably close to the real values.
The second area of validation in the caterpillar model involved the model's reaction to weather. During the calculation of the system parameters discussed in chapter four, only two constant weather values had been used. Figure IV shows the real weather and the response of the model to it from 1959 to 1964.
Notice that when the weather improved from 1959 to 1963, the population increased, and in 1964 when the weather was harsh, the population decreased.

A clearer picture of what happens spatially can be found by plotting the caterpillar population on a two dimensional grid. Figure V is a plot of the population on the Saanich Peninsula in 1960 when the weather was quite mild.
FIGURE V - CATERPILLAR DISTRIBUTION IN MILD WEATHER

Figure VI is a plot of the population in 1964 when the weather was quite harsh.
Note that the milder weather allowed the caterpillars to infect areas with relatively hostile local climate, but in harsh weather, only the areas with the best local climate were occupied. The refuge areas into which the caterpillars receded in years of harsh weather agrees with field observations. Figures VII and VIII demonstrate another method of plotting the mild and harsh weather distributions.
FIGURE VIII - CATERPILLAR DISTRIBUTION IN HARSH WEATHER
6. CONCLUSIONS

One of the most encouraging things that happened, was that with relatively simple state equations expressed as ordinary differential equations, there was reasonable agreement with the real ecosystem. The simplified nature of the model is quite appealing in that it makes its functioning quite straightforward and allows for the modification of the state differential equations. Furthermore, the graphics display of the caterpillar population on a two dimensional grid (which produced figures V and VI) gave a visual understanding of what is taking place. Currently, work is being done on taking a movie of such displays over a long period of time, and should give added appreciation for the functioning of refuge areas and the response of the model to varied weather conditions.

Areas of future study would undoubtedly include extending the simple differential equations to include additional terms, such as cross product terms between weather, time, and percent of type I caterpillars. Perhaps an extension such as $E(i) \times W \times T + P(i) \times W \times PCT$ would be a good place to start. One would tend to expect better agreement with the real ecosystem as more detail is added to the differential equations in the model. Of course new terms in the differential equations would necessitate a new set of parameter values. Experience has shown that the determination of those values should not be limited to examining a single colony in two weather conditions, but should be done using a whole grid square of
colonies at one time.

Another area that would be interesting to address, involves long periods of hypothetical weather. Using the six years of known weather to create realistic long term weather, it would be productive to study extended periods of harsh and mild weather. The results would show how responsive the model is overall to weather and might be used to predict the conditions under which extinction or saturation might occur.

The visual plots produce by the model hold the greatest prospect for future use. The movie included with this thesis demonstrates very clearly the important part visual images can play in understanding the functioning of a simulated system. That discovery is not only relevant to bio-simulation, but is a powerful technique to be applied to modeling in general.
BIBLIOGRAPHY


APPENDIX I - NUMERICAL METHODS
EULERS METHOD

The general equations

1) \( D(i) = \frac{dP(i)}{dt} = P(i) \) where \( P_i \) is a function

2) \( P'(i) = P(i) + D(i) \times (T-T_0) \) of \( P \) and \( T \)

The general process

i) calculate vector \( D \) from equation 1

ii) calculate vector \( P' \) from equation 2

iii) \( P' \) now becomes \( P \) and \( T_0 \) becomes \( T \)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION X(12), G(10), GX(10,12), THI(3),
       DX1(3), DX2(12)

DATA THI /0.000/
N = 0
NF = 0
NC = 10
FACT2 = .5
READ (1,5) X
WRITE (6,3) X
CALL STATE (10,12,NC,X,G,GX,GNEW)
WRITE (6,3) FACT2
WRITE (6,2) GNEW
WRITE (6,3) (G(I),I=1,NC)
DO 9 J = 1, 12
  9 WRITE (6,3) (GX(I,J),I=1,NC)
10 CONTINUE
CALL DX (NC,12,10,G,GX,THI,DX1,DX2)
WRITE (6,3) DX2
READ (5,5) FACT2
DO 101 I = 1, 12
  101 X(I) = X(I) - DX2(I)*FACT2
C WRITE (6,3) FACT2
WRITE (6,3) X
GOLD = GNEW
CALL STATE (10,12,NC,X,G,GX,GNEW)
N = N + 1
WRITE (6,2) GNEW
WRITE (6,3) (G(I),I=1,NC)
DO 102 J = 1, 12
  102 WRITE (6,3) (GX(I,J),I=1,NC)
IF (GNEW .LT. GOLD) GO TO 11.
FACT2 = FACT2/2
NF = 0
GO TO 20
11 CONTINUE
IF ((DLOG10(GOLD)-DLOG10(GNEW)) .GT. 1.0) GO TO 12
NF = NF + 1
IF (NF .LT. 3) GO TO 20
FACT2 = FACT2*2
IF (FACT2 .GT. 1) FACT2 = 1
NF = 0
GO TO 20
12 CONTINUE
FACT2 = 1.0
NF = 0
20 CONTINUE
IF (GNEW .LT. 1.0D-10) GO TO 10
WRITE (6,1) (X(I),I=1,12)
STOP
1 FORMAT (1X, 6(D17.10,3X))
APPENDIX II - NEWTON-RAPHSON ROUTINES
MAIN PROGRAM

2 FORMAT (10X,10H ERRORMAG ,D14.7)
3 FORMAT (1X, 10(2X,D10.3))
4 FORMAT (I2,I2)
5 FORMAT (D8.2)
END
SUBROUTINE STATE (L,M,N,XO,G0,GX,GMAG)
C CALCULATES THE STATE OF THE SYSTEM
C GO IS THE CURRENT ERROR
C GX IS THE CURRENT SENSITIVITY MATRIX
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION X(20), XO(M), G(20), G0(N), GX(L, M), GSUM(20)
DATA NTIMES /-1/
DO 10 I = 1, N
GSUM(I) = 0
10 CONTINUE
DO 20 I = 1, M
X(I) = XO(I)
20 CONTINUE
CALL ERROR(M, N, XO, G0)
GMAG = 0
DO 30 J = 1, N
GMAG = GMAG + DABS (G0(J))
30 CONTINUE
WRITE (6,1) (G0(I), I=1,N)
NTIMES = NTIMES + 1
C IF (MOD(NTIMES,4) .NE. 0) RETURN
DO 100 I = 1, B
X(I) = X0(I) • 1.QD-2*X0(I)
CALL ERROR(M,N,X,G)
DO 40 J = 1, N
GX(J,I) = (G(J) - G0(J))/(X(I)-X0(I))
GSDM(J) = GSUM(J) + DABS (GX (J, I))
GSUM(J) = GSUM(J)/N
40 CONTINUE
X(I) = XO(I)
100 CONTINUE
DO 120 J = 1, N
GSUM(J) = GSUM(J)/N
IF (GSUM(J) .EQ. 0) GSUM(J) = 1.0D0
G0(J) = G0(J)/GSUM(J)
120 CONTINUE
DO 140 J = 1, N
GX(J,I) = GX(J,I)/GSUM(J)
140 CONTINUE
160 CONTINUE
RETURN
1 FORMAT (1X,10(2X,D10.3))
END
SUBROUTINE ERROR (M, N, X, G)

C CALCULATES THE ERROR BY INTEGRATING THE
C EQUATIONS TO THE FINAL TIME

IMPLICIT REAL*8 (A-H, O-Z)
COMMON /COM/ C(12)
DIMENSION X(M), G(N), P(13), PO(13), PD(13),
Q(13)

DATA P0 /0.0D0, 
1 5.0D1, 7.1D1, 1.12D2, 3.0D0, 
2 5.0D1, 7.1D1, 1.12D2, 3.0D0, 
2 2.6D1, 4.5D1, 9.9D1, 1.5D1/ 
DO 10 I = 1, 13
P(I) = P0(I)
10 CONTINUE
DO 20 I = 1, M
C(I) = X(I)
20 CONTINUE
DO 40 J = 2, 13
C IF (P(J) .LT. 1) P(J) = 0.0D0
30 CONTINUE
WRITE (6, 123) P
123 FORMAT (1X, 13D10.3)
40 CONTINUE
G(1) = P(2) - 3.0D0
G(2) = P(3) - 3.6D1
G(3) = P(4) - 1.1D1
G(4) = P(5) - 7.0D0
G(5) = P(6) - 1.5D0
G(6) = P(7) - 0.0D0
G(7) = P(8) - 0.0D0
G(8) = P(9) - 0.0D0
G(9) = 6.0D0-P(10)-P(11)-P(12)
G(10) = P(13) - 0.0D0
RETURN
END
SUBROUTINE AUXRK(X,F)

C CALCULATES THE DERIVATIVES (F) OF THE
C STATE VARIABLES (X)

IMPLICIT REAL*8 (A-H,O-Z)
COMMON /COM/ C (12)
DIMENSION X (1), F (1), W (3), PCT (3)
DATA W /2.5D1, 7.5D1, 5.46D2/
DO 20 I = 1, 3
    I4 = 4*(I-1) + 1
    T = 0.0D0
    DO 10 J = 1, 4
        K = I4 + J
        T = T + DABS(X(K))
    10 CONTINUE
    PCT(I) = DABS(X(I4+1))/T - 2.0D-1
    PCT(I) = PCT(I)*1.0D-1
    IF (PCT(I) .GT. 0) PCT(I) = 0.0D0
20 CONTINUE
DO 40 I = 2, 13
    I2 = I + 2
    J = I2/4
    K = 3*(I2-4*J)
    F(I) = (C(K+1) + C(K+2)*X(1) + C(K+3)*W(J) + PCT(J))*DABS(X(I))
    IF (F(I) .GT. 0) F(I) = 0
40 CONTINUE
RETURN
END
SUBROUTINE DX (NC,NV,MC,G,GX,THIX,DX1,DX2,IS)

C CALCULATES THE CORRECTION VECTOR DX2 GIVEN G AND GX

C IMPLICIT REAL*8 (A-H,O-Z)
COMMON /MATEXP/ JEXP
DIMENSION G(NC), GX(MC,NV), THIX(NV)
DIMENSION DX1(NV), DX2(NV), GXI(15,15),
       TEMP(15,15)
CALL DGPROD (GX,GX,GXI,NC,NV,NC,MC,15,0,1,1)
CALL DINVRT (GXI,NC,15,DET,COND)
DET = DET*1.0D1**JEXP
WRITE (6,2) DET, COND
IF (DET .EQ. 0) GO TO 400
CALL DGPROD (GX,GXI,TEMP,NV,NC,MC,15,15,1,0,0)
IF (IS .EQ. 1) GO TO 200
CALL DGMATV (TEMP,G,DX2,NV,NC,15)
IF (IS .EQ. 2) GO TO 300
200 CONTINUE
CALL DGMULT (TEMP,GX,GXI,NC,NV,15,MC,15)
DO 260 I = 1, NV
   DO 240 J = 1, NV
      IF (I .EQ. J) GO TO 220
      GXI(I,J) = -GXI(I,J)
240 CONTINUE
220 CONTINUE
GXI(I,J) = 1 - GXI(I,J)
240 CONTINUE
CALL DGMATV (GXI,THIX,DX1,NV,NV,15)
300 RETURN
400 WRITE (6,1) JEXP
STOP
1 FORMAT (1X,I4,25H *** SINGULAR MATRIX)
2 FORMAT (10H DET = ,D14.7, 10H COND = , D14.7)
END
C      COMMON /SEARCH/ GOES HERE
2  C COMMON /SERVC/ GOES HERE
3  EQUIVALENCE (JACOB,ACOB)
4  DIMENSION JACOB(1), ISAV(200), JSAV(200)
C
6  DATA NSAV /0/
7  NCOM = (LOCF(TEMP(1)) - LOCF(ACOB)) / 4
8  CONTINUE
9  IF (NSAV .EQ. 0) GO TO 60
10  DO 20 I = 1, NCOM
11     JACOB(I) = 0
12     CONTINUE
13  DO 40 I = 1, NSAV
14     J = ISAV(I)
15     JACOB(J) = JSAV(I)
16     CONTINUE
17  CALL RSEARC
18  CONTINUE
19  CALL REGION
20  CONTINUE
21  DO 80 I = 1, NCOM
22     IF (JACOB(I) .EQ. 0) GO TO 80
23     NSAV = NSAV + 1
24     ISAV(NSAV) = I
25     JSAV(NSAV) = JACOB(I)
26     CONTINUE
27  CALL MINMYS
28  NSAV = 0
29  CONTINUE
30  GO TO 10
31  END
APPENDIX III – CONSTRAINED TARGETING ROUTINES

BLOCK DATA

1  BLOCK DATA
2  C  COMMON /SEARCH/ GOES HERE
3  C  COMMON /SERVC/ GOES HERE
4  DATA TEMP /25*0.0 /
5  DATA STEMPE/25*0.0 /
6  DATA I /0 /
7  DATA J /0 /
8  DATA L /0 /
9  DATA M /0 /
10  DATA N /0 /
11  DATA NULL /1HU /
12  DATA N00 /0 /
13  DATA N01 /1 /
14  DATA N02 /2 /
15  DATA N03 /3 /
16  DATA N04 /4 /
17  DATA N05 /5 /
18  DATA N06 /6 /
19  DATA N07 /7 /
20  DATA N08 /8 /
21  DATA N09 /9 /
22  DATA N10 /10 /
23  DATA FP5 / .5 /
24  DATA FP1 /1.0 /
25  DATA FP2 /2.0 /
26  DATA FP3 /3.0 /
27  DATA FP4 /4.0 /
28  DATA FP5 /5.0 /
29  DATA FP6 /6.0 /
30  DATA FP7 /7.0 /
31  DATA FP8 /8.0 /
32  DATA FP9 /9.0 /
33  DATA FP10 /10.0 /
34  C  DATA RPD /171243575065045123518B/
35  C  DATA DPR /172571227340646170208B/
36  DATA XINF /10000000000.0 /
37  C
38  C
39  DATA CONEPS/ 89.9 /
40  1  5*1 /
41  DATA CONSEX/ .000001 /
42  DATA CTHA / .5 /
43  DATA FITERR/ .000001 /
44  DATA GAMAX / 3.0 /
45  DATA ICGM / 63 /
46  DATA .ISTART/ 63 /
47  DATA IDEB / 0 /
48  DATA IDEPVR/ 25*0 /
49  DATA KASE / 1 /
50  DATA MAXITR/ 10 /
51  DATA MODEW / 1 /
52  DATA NDEPV / 0 /
53  DATA NFLAG / 0 /
DATA NINDV / 0 /
DATA OPT / 0.0 /
DATA PCTCC / .3 /
DATA PCTOLD / .3 /
DATA PERT / 25*1.0E-4 /
DATA PGEPS / 1.0 /
DATA P2MIN / 1.0 /
DATA SRCHM / 0.0 /
DATA STMINP / .01 /
DATA STPMAK / 1.0E+10 /
DATA U / 25*0.0 /
DATA WCON / 100.0 /
DATA WOPT / 1.0 /
DATA WU / 25*1.0 /
END
SUBROUTINE ABT (A, B, C, L, M, N)

C*** PERFORMS MATRIX MULTIPLICATION
C
C = A * TRS(B)
C
MATRICES ARE ASSUMED TO BE STORED IN COLUMN ORDER

DIMENSION A(L,M), B(N,M), C(L,N)

DIMENSION A(1), B(1), C(1)

DO 60 I = 1, L
DO 40 J = 1, N
IC = I + (J-1)*L
C(IC) = 0.0
DO 20 K = 1, M
IA = I + (K-1)*L
IB = J + (K-1)*N
C(IC) = C(IC) + A(IA)*B(IB)
CONTINUE
CONTINUE
CONTINUE
RETURN
END
SUBROUTINE AUXRK

COMMON /COM/ C (12)
DIMENSION X (1), P (1), W (3), PCT (3)
DATA W /2.5E1, 7.5E1, 5.46E2/

DO 20 I = 1, 3
I4 = 4 * (I - 1) + 1
T = 0.0E0
DO 10 J = 1, 4
K = I4 + J
T = T + ABS (X (K))
10 CONTINUE
IF (T .NE. 0) PCT (I) = ABS (X (I4 + 1)) / T - 2.0E-1
PCT (I) = PCT (I) * 1.0E-1
IF (PCT (I) .GT. 0) PCT (I) = 0.0E0

DO 40 I = 2, 13
I2 = I + 2
J = I2 / 4
K = 3 * (I2 - 4 * J)
XX = X (I)
IF (XX .LT. -1) XX = 1.0 / ABS (XX) ** 3
IF (XX .LT. 0) XX = ABS (XX)
P (I) = (C (K + 1) + C (K + 2) * X (1) + C (K + 3) * W (J) + PCT (J)) * XX
IF (P (I) .GT. 0) P (I) = 0
40 CONTINUE
RETURN
SUBROUTINE BTW (B, W, D, L, M, N)

C*** PERFORMS THE INDICATED MATRIX MULTIPLICATION
C MATRICES ARE ASSUMED TO BE STORED IN COLUMN ORDER.

C D = TRS(B) * W
C DIMENSION B(M,L), W(M,N), D(L,N)

DIMENSION B(1), W(1), D(1)

DO 60 I=1,L
IM1M=M*(I-1)
DO 40 J=1,N
JM1L=L*(J-1)
JM1M=M*(J-1)
ID=JM1L+I
D(ID)=0.0
DO 20 K=1,M
IB=IM1M+K
IW=JM1M+K
D(ID)=D(ID)+B(IB)*W(IW)
20 CONTINUE
40 CONTINUE
60 CONTINUE
RETURN
END
SUBROUTINE BUCKET(X, Y, N, XX, YY, NP)

C***

C THIS ROUTINE REARRANGES AN ARRAY X AND THE CORRESPONDING ELEMENTS OF Y IN ASCENDING ORDER. N IS THE NUMBER OF ELEMENTS IN EACH.

C NP IS A POINTER TO THE FIRST ELEMENT OF Y THAT IS LESS THAN THE NEXT ELEMENT. NP IS ZERO IF IT DOES NOT EXIST. ORDERED VALUES ARE RETURNED IN XX AND YY.

C***

DIMENSION X(1), Y(1), XX(1), YY(1), XS(10)

DO 10 I = 1, N
10 XS(I) = X(I)

DO 30 I = 1, N
20 NSMAL = 1

DO 20 J = 1, N
30 CONTINUE

XX(I) = X(NSMAL)
YY(I) = Y(NSMAL)
XS(NSMAL) = 1.E10

30 CONTINUE

NP = 0

DO 40 I = 2, N
40 CONTINUE

RETURN

NP = I-1
RETURN
END
SUBROUTINE CGM (F0, G, GMAG)

C THIS SUBROUTINE CALCULATES A DU BY THE
C CONJUGATE GRADIENT METHOD

C WITH AN INITIAL STEEPEST DESCENT STEP WHEN
C ICGM IS NONZERO

C

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE
EQUIVALENCE (GSQ, TEMP(1)), (CON, TEMP(2))

C

DIMENSION G(1)
GSQ=GMAG**2
CON=GSQ/GSQOLD
IF (ICGM.EQ.0) GO TO 40

C

C INITIALIZATION PASS

C

L=0
DO 20 I=1, NINDV
S(I)=-G(I)
DU(I)=S(I)
IF (IDAV.EQ.0) GO TO 20
L=L+I
H(L) = FP1
GP(I) = G(I)
20 CONTINUE
GO TO 100

C

C CONJUGATE STEPS

C

40 CONTINUE
IF (IDAV.EQ.0) GO TO 60
CALL DGM (G)
GO TO 100

60 CONTINUE
DO 80 I=1, NINDV
S(I)=CON*S(I)-G(I)
DU(I)=S(I)
80 CONTINUE

GO TO 100

40 GSQOLD=GSQ
ICGM=0
RETURN
END
SUBROUTINE COMBIN
C  COMMON /SEARCH/ GOES HERE
IF (IND(1) .NE. 0) GO TO 40
M = IND(2)
N = IND(3)
DO 20 I = 1, N
IND(I) = I
20 CONTINUE
I = 1
K = N
GO TO 120
40 CONTINUE
IF (K .EQ. 0) GO TO 100
L = M + K - N
IF (IND(K) .LT. L) GO TO 60
K = K - 1
GO TO 40
60 CONTINUE
IND(K) = IND(K) + 1
I = I + 1
80 CONTINUE
IF (K .EQ. N) GO TO 120
IND(K+1) = IND(K) + 1
K = K + 1
GO TO 80
100 CONTINUE
IND(1) = -1
120 CONTINUE
RETURN
END
SUBROUTINE CUBMIN(A, XMIN, YMIN)

C INPUTS
A=ARRAY OF CUBIC POLYNOMIAL
COEFFICIENTS. A(1) IS CONST
TERM, A(2) IS LINEAR TERM, A(3) IS
QUADRATIC TERM, AND
A(4) IS CUBIC TERM.

C OUTPUTS
XMIN=ABSCISSA VALUE OF MINIMUM OF CUBIC
YMIN=ORDINATE VALUE OF MINIMUM OF CUBIC

1 FORMAT( 'CUBIC DEGENERATES TO A
STRAIGHT LINE,'/)
2 FORMAT( 'CUBIC HAS NO EXTREMA,'/)
3 FORMAT( 'CUBIC DEGENERATES TO A
QUADRATIC WITH NO
MINIMUM,'/)

C
DIMENSION
A(4)

DATA
R2 /2.0/
R3 /3.0/
TP10 /1.0E+10/

C
IF (A(4)) 10,20,10
20 CONTINUE
IF (A(3)) 30,40,50
40 CONTINUE
WRITE(6,1)
GO TO 60
30 CONTINUE
WRITE(6,3)
GO TO 60
34 CONTINUE
50 CONTINUE
XMIN=-A(2)/(R2*A(3))
GO TO 70
10 CONTINUE
THRA4=R3*A(4)
DISC=A(3)**2-A(2)*THRA4
IF (DISC) 80,80,90
80 CONTINUE
C
WRITE (6,2)
C
60 CONTINUE
XMIN=TP10
YMIN=TP10
GO TO 100
C
90 CONTINUE
DISC=SQRT(DISC)
XMIN=(-A(3)+DISC)/THRA4
APPENDIX III - CONSTRAINED TARGETING Routines

SUBROUTINE CUBMIN

50 70 CONTINUE
51 YMIX=A(1)+XMIN*(A(2)+XMIN*(A(3)+XMIN*A(4)))
52 100 CONTINUE
53 RETURN
54 END
SUBROUTINE DELTU

DELTU CALCULATES THE DIRECTION OF SEARCH BY THE METHOD SPECIFIED
BY SRCHM, SCALES THE VECTOR AND CALLS ULIMIT AND TEST

COMMON /SEARCH/ GOES HERE

INTEGER SRCHM

CALL WUCAL

CALL GMAG

BRANCH ON SRCHM

GO TO (20, 40, 60, 80), SRCHM

CONJUGATE GRADIENT METHOD

ICGM=63

CALL CGM (P2N0M, G2, G2MAG)

GO TO 100

STEPEST DESCENT METHOD

CONTINUE

CALL SDM

GO TO 100

CGM OF G1

CONTINUE

ICGM=63

CALL CGM (P1N0M, G1, G1MAG)

GO TO 100

PROJECTED GRADIENT METHOD

CONTINUE

CALL PGM

CALL UNITDU

IF (SRCHM .EQ. 4) CALL GABDD

RETURN

END
SUBROUTINE DGM (G)

C*** COMPUTES DU BY DAVIDONS DEFLECTED GRADIENT METHOD

C
C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE
C
EQUIVALENCE (P,DU) , (PP,S)
DIMENSION P(25), PP(25), G(25)
DATA ZERO,ONE/0.0,-1.0/
N = NINDV
M=N*(N+1)/2

C COMPUTE NEW DEFORMATION MATRIX
DO .20 J=1,N
DG(J)=G(J)-GP(J)
20 CONTINUE

DO 100 J=1,B
HDG(J)=ZERO
DO 100 L=1,N
IF (J-L) 60,60,40
40 JLA=L+J*(J-1)/2
GO TO 80
60 JLA=J+L*(L-1)/2
80 HDG(J)=HDG(J) + H(JLA) *DG(L)
100 CONTINUE

PPTDG=ZERO
DGTHDG=ZERO
DO 120 J=1,N
PPTDG=PPTDG+PP(J) *DG(J)
DGTHDG=DGTHDG+DG(J) *HDG(J)
120 CONTINUE
TEMP1=GAMASS/PPTDG
TEMP2=ONE/DGTHDG
DO 140 L=1,N
DO 140 J=1,L
JLA=J+L*(L-1)/2
A=PP(J) *PP(L) *TEMP1
B=HDG(J) *HDG(L) *TEMP2
H(JLA)=H(JLA) +A-B
140 CONTINUE

C COMPUTE NEW SEARCH DIRECTION
DO .220 J=1,N
P(J)=ZERO
DO .220 L=1,N
IF (J-L) 180,180,160
160 JLA=L+J*(J-1)/2
GO TO 200
180 JLA=J+L*(L-1)/2
200 P(J)=P(J) -H(JLA) *G(L)
220 CONTINUE
APPENDIX III - CONSTRAINED TARGETING Routines
SUBROUTINE DGM

53 PP(J) = P(J)
54 GP(J) = G(J)
55 240 CONTINUE
56 RETURN
57 END
SUBROUTINE DIGDIF(M, N, NDIF)
DIMENSION MTAB(8)
DATA MTAB /9999999, 999999, 99999, 9999, 999, 99, 9, 0/
L = LXOR(M, N)
DO 10 I = 1, 8
   IF (L .GT. MTAB(I)) GO TO 20
10 CONTINUE
I = 9
20 NDIF = 9 - I
WRITE (6,1) M, N, L, I
1 FORMAT (10H M, N, L, I , 4(5X, Z8))
RETURN
END
SUBROUTINE DISPLY

CC COMMON /SEARCH/ GOES HERE
CC COMMON /SERVC/ GOES HERE
C EQUIVALENCE (IR1,I)

CC
C DIMENSION MSG(3)
C DATA IPP/15230700000100000000B/
C DATA LRAP1/0/
C IF (LRAP1.NE.0) GO TO 20
C LRAP1=2-LOCF(MSG)
C IR1=LOCF(MSG)
C IPP=IPP+IR1
C20 CONTINUE
C TEMP(1)=ABS(P1NOM)
C ENCODE (12,80,STEMP(1)) TEMP(1)
C ENCODE (12,80,TEMP(1)) P2NOM
C IF (P2NOM.EQ.0.0) P2NOM=1.0E-10
C IR1 = 0
C IF (P2NOM .NE. 0.0) IR1 =
C ALOG10(ABS(P2NOM))

C I=1H+
C IF (P2NOM.GE.1.0) GO TO 40
C I=1H-
C IR1=1-IR1
C40 CONTINUE
C ENCODE (28,100,MSG) NSTEP,STEMP(1),TEMP(1),
C I,IR1,CTHA
C60 CONTINUE
C IF (MSG(LRAP1).NE.0) GO TO 60
C MSG(LRAP1) = IPP
RETURN

C80 FORMAT (E12.6)
C100 FORMAT (I2,1X,A8,1X,A4,1HE,A1,I1,1X,F8.5)
END
SUBROUTINE DROP

C COMMON SEARCH GOES HERE
C COMMON SERVC GOES HERE

IF (NAC .GT. NINDV) GO TO 40
CONTINUE
CALL REVISE
IF (IDEB .NE. 0) WRITE (6,3) ICD
IF (ICD .EQ. 0) RETURN
GO TO 20

MORE ACTIVE CONSTRAINTS THAN CONTROLS

40 CONTINUE
DP1 = -1.0E+10
IND(1) = 0
IND(2) = NAC - NEQC
IND(3) = NINDV - NEQC
IF (IND(2) * IND(3) .LE. 0) GO TO 300
DO 60 I = 1, NDEPV
ISTC(I) = ITC(I)
60 CONTINUE

MAKE ONE ITERATION PER COMBINATION

80 CONTINUE
CALL COMBIN
IF (IND(1) .LT. 0) GO TO 200
J = 0
K = 1
DO 100 I = 1, NDEPV
ITC(I) = 0
IF (ISTC(I) .EQ. 0) GO TO 100
J = J + 1
IF (J .NE. IND(K)) GO TO 100
K = K + 1
ITC(I) = 63
100 CONTINUE
NTC = NINDV - NEQC
WRITE (6,5) (ITC(I), I = 1, NDEPV)
CALL UPDATS
CALL REVISE
IF (ICD .EQ. 0) GO TO 400
WRITE (6,7) (ITC(I), I = 1, NDEPV)
DO 140 J = 1, NDEPV
IF (ISTC(J) .EQ. 0) GO TO 140
IF (ITC(J) .NE. 0) GO TO 140
DO 120 I = 1, NINDV
K = J + (I-1) * NDEPV
TEMP(I) = ACOB(K)
120 CONTINUE
CALL MATPY (PG1, TEMPI, STEMPI, NO1, NINDV, NO1)
APPENDIX III - CONSTRAINED TARGETING ROUTINES

SUBROUTINE DROP

53 WRITE (6,6) STEHP(1)
54 IF (STEHP(1)) 140,140,80
55 140 CONTINUE
56 TEMP(1) = SP(PG1,G1,NINDV)/PG1MAG
57 IF (IDEB .NE. 0) WRITE (6,4) TEMP(1), (ITC(I), I=1,NDEPV)
58 IF (TEMP(1) .LT. DP1) GO TO 80
59 DP1 = TEMP(1)
60 DO 150 I = 1, NDEPV
61 IBTC(I) = ITC(I)
62 150 CONTINUE
63 GO TO 80
64 C
65 C  FEASIBLE DIRECTION FOUND
66 C
67 200 CONTINUE
68 DO 220 I = 1, NDEPV
69 ITC(I) = IBTC(I)
70 220 CONTINUE
71 CALL UPDATS
72 RETURN
73 C
74 C  ERROR MESSAGES
75 C
76 300 CONTINUE
77 WRITE (6,1)
78 GO TO 500
79 400 CONTINUE
80 WRITE (6,2)
81 500 CONTINUE
82 NFLAG = -1
83 RETURN
84 1 FORMAT (44H-*** MORE EQUALITY CONSTRAINTS THAN CONTROLS)
85 2 FORMAT (49H-*** PROBLEM SOLVED === NAC IS GREATER THAN NINDV)
86 3 FORMAT (10H-CONTRAINT, I2, 8H DROPPED)
87 4 FORMAT (7H-DP1 = ,E21.14, 7H ITC = ,25(I1, 1X))
88 5 FORMAT (20H-COMBINATION OF ITC ,25(I1,1X))
89 6 FORMAT (37H-NEGATIVE DU DOTTED WITH CONSTRAIN = ,E21.14)
90 7 FORMAT (20H-REVISION OF ITC ,25(I1,1X))
91 END
SUBROUTINE ERROR (I,J)
C***ERROR
C* ERROR-PRINT ERR/R MESSAGE AND DETERMIN IF FATAL
C*
DATA MSK1 /77000000000000000000B/
DATA IBK1 /55000000000000000000B/
CC
10 CONTINUE
CALL PAGER (3)
WRITE (6,100) I,J
20 CONTINUE
STOP
RETURN
100 FORMAT (10X,10H**ERROR**, 2X,A10,A10 )
END
SUBROUTINE FGAMA (IS)

C*** DETERMINES THE P1 AND P2 ASSOCIATED WITH A PARTICULAR GAMA IN THE DIRECTION OF SEARCH BY CHANGING THE CONTROLS: U = U0 + GAMMA*DU

COMMON /SEARCH/ GOES HERE
COMMON /SERVC/ GOES HERE

GAMAS=STEP(IS)
CALL TRAJ
P1TRY(IS)=P1
P2TRY(IS)=P2
IF (INTRI1 .EQ. 0) RETURN

ESTIMATE NET PERFORMANCE

IF (NAC .EQ. 0) GO TO 15
CALL HATPY (EHAP,EA,TEHP(IS),NINDV,NAC,1), P1TRY(IS)=P1TRY(IS)+SP(G1,TEMP(IS)^HINDV)
CONTINUE
JS=IS-1
SAVIT(1,JS)=P1
SAVIT(2,JS)=P2
DO 19 I=1,NDEPV
SAVIT(I+2,JS)=E(I)
CONTINUE
RETURN
END
SUBROUTINE FOPMIN(X,Y,XMIN,YMIN,IERR)

C INPUTS
C X=ARRAY OF ABSCISSA VALUES OF FOUR
C SAMPLE POINTS
C Y=ARRAY OF ORDINATE VALUES OF FOUR
C SAMPLE POINTS

C OUTPUTS
C XMIN=ABSCISSA VALUE OF MINIMUM OF
C ABOVE CUBIC
C YMIN=ORDINATE VALUE OF MINIMUM OF
C ABOVE CUBIC
C IERR=FLAG WHOSE NON-ZERO VALUE
C INDICATES THAT TWO OF THE
C GIVEN X VALUES ARE IDENTICAL

DIMENSION
1 X(4),Y(4),A(4)

DATA
1 NO /0 /
2 ,N1 /1 /

C IERR=NO
B21=X(2)-X(1)
B31=X(3)-X(1)
B32=X(3)-X(2)
B41=X(4)-X(1)
B42=X(4)-X(2)
B43=X(4)-X(3)
TEMP1=B21*B31*B41
IF (TEMP1) 10,20,10
10 CONTINUE
F1=Y(1)/TEMP1
TEMP1=B21*B32*B42
IF (TEMP1) 30,20,30

F1=-Y(2)/TEMP1
TEMP1=B31*B32*B43
IF (TEMP1) 40,20,40

F1=-Y(3)/TEMP1
TEMP1=B41*B42*B43
F4=-Y(4)/TEMP1
GO TO 50

A(4)=-(F1+F2+F3+F4)
A(3)=(X(2)*X(3)*X(4))*F1+(X(1)*X(3)+...
SUBROUTINE POPMIN

\[
X(4) \times F_2 + (X(1) + X(2) + X(4)) \times F
\]
\[
+ (X(1) + X(2) + X(3)) \times F_4
\]

50 \quad B_{21} = X(2) \times X(1)

51 \quad B_{31} = X(3) \times X(1)

52 \quad B_{32} = X(3) \times X(2)

53 \quad B_{41} = X(4) \times X(1)

54 \quad B_{42} = X(4) \times X(2)

55 \quad B_{43} = X(4) \times X(3)

56 \quad A(2) = -((B_{32} + B_{42} + B_{43}) \times F_1 + (B_{31} + B_{41} + B_{43}) \times F_2 + (B_{21} + B_{41} + B_{42}) \times F_3 + (B_{21} + B_{31} + B_{32}) \times F_4)

57 \quad A(1) = Y(1) - (X(1) + A(2) \times X(1) + A(3) + X(1) \times A(4))

59 \quad \text{CALL CUBMIN(A, XMIN, YMIN)}

60 \quad \text{CONTINUE}

61 \quad \text{RETURN}

62 \quad \text{END}
SUBROUTINE GABDD

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE

STPMAX = 1.0E+10
DO 40 I = 1, NDEPV
IF (IDEPVR(I) .EQ. 0) GO TO 40
IF (ENOM(I) .LE. 1.0) GO TO 40
DO 20 J = 1, NINDV
K = I + (J-1)*NDEPV
20 TEMP(J) = ACOB(K)
10 CONTINUE
CALL HATPY (TEMP(1),DU,STEMP(1),1,NINDV,1)
IF (STEMP(1) .GE. 0) GO TO 40
TEMP(1) = -ENOM(I)/STEMP(1)
IF (TEMP(1) .GT. STPMAX) GO TO 40
STPMAX = TEMP(1)
40 CONTINUE
RETURN
END
SUBROUTINE GENHIN (X,Y,DYDX1,EPSA,YES,HIN)

C*** FINDS A MINIMUM TO THE RIGHT OF X(1), BUT TO THE LEFT EPSA(2).
C TRIAL STEPS ARE STORED IN ORDERED PAIRS (X, Y). DYDX1 IS THE SLOPE
C OF THE FUNCTION AT X(1). ZERO SLOPE IS IGNORED IN THE ESTIMATION
C PROCESS. USER MAY PRESET ANY ELEMENTS OF THE X AND Y ARRAYS.
C YES CONTAINS THE PREDICTED VALUES OF THE FUNCTION BASED UPON THE
C CURVE FIT. IF ANY PREDICTED VALUE IS LESS THAN EPSA(4) PERCENTAGE
C FROM THE ACCUAL VALUE, THE MINIMIZATION PROCESS IS TERMINATED.
C IF ANY TWO CONSECUTIVE X VALUES ARE LESS THAN EPSA(3) PERCENTAGE
C APART, CONVERGENCE IS ASSUMED. MIN IS SET TO THE NUMBER OF THE
C ORDERED PAIR THAT COMES CLOSEST TO THE MINIMUM. A NEGATIVE MIN
C INDICATES THAT A WELL DEFINED MINIMUM DID NOT EXIST. A ZERO MIN
C INDICATES THAT NO MINIMUM EXISTS. EPSA(2) IS THE MAXIMUM PERCENT
C THAT X IS ALLOWED TO BE REDUCED BY IF THE FUNCTION INCREASES
C
C DIMENSION X(6), Y(6), XX(5), YY(5), EPSA(4), YES(4)

DO 20 I=1,4
YES(I)=0.0
20 CONTINUE
IF (DYDX1) 40,360,440
CSLOPE NEGATIVE
CFILL IN NEEDED VALUES FOR FIRST FIT
40 IF (Y(1) .EQ. 0.0) CALL .FGAMA (1)
IF (Y(2) .NE. 0.0) GO TO 60
IF (X(2) .EQ. 0.0) X(2)=1.0
CALL .FGAMA (2)
GO TO 80
60 IF (Y(3) .NE. 0.0) GO TO 120
CTWO POINT ONE SLOPE
80 CALL TPOSM (X,Y,DYDX1,X(3),YES)
IF (X(3) .LT. EPSA(1)*X(2)) X(3)=EPSA(1)*X(2)
IF (X(3) .GT. EPSA(2)) X(3)=EPSA(2)
IF (ABS((X(3)-X(2))/X(2)) .GT. EPSA(3)) GO TO 100
MAX=2
100 GO TO 300
CALL .FGAMA (3)
SUBROUTINE GENMIN

CCHECK QUADRATIC FIT
IF (Y(3) .EQ. 0.0) Y(3) = 1.0E-10
IF (ABS((YES(1) - Y(3)) / Y(3)) .GE. EPSA(4)) GO TO 120

MIN = 3
GO TO 460

CTHREE POINT ONE SLOPE
120 CALL THPOSH (X, Y, DYDX1, X(4), Y(2), ISER)
140 CONTINUE

IF (X(4) .LT. EPSA(1) * X(3)) X(4) = EPSA(1) * X(3)
IF (X(4) .GT. EPSA(2)) X(4) = EPSA(2)
IF (ABS((X(4) - X(3)) / X(3)) .LE. EPSA(3)) GO TO 150
IF (ABS((X(4) - X(2)) / X(2)) .GT. EPSA(3)) GO TO 160

MAX = 3
GO TO 300

160 CALL FGAMA(4)

CCHECK CUBIC FIT
IF (Y(4) .EQ. 0.0) Y(4) = 1.0E-10
IF (ABS((YES(2) - Y(4)) / Y(4)) .GE. EPSA(4)) GO TO 180

MIN = 4
GO TO 460

CFIND THREE BEST POINTS FOR NEXT FIT
180 CALL BUCKET (X, Y, YY, NP)
IF (NP .GT. 1) GO TO 220
MAX = 4
GO TO 300
NP = NP - 1

220 CALL THPM (XX(NP), YY(NP), X(5), YES(3))

CTHREE POINT
IF (X(5) .LT. EPSA(1) * X(4)) X(5) = EPSA(1) * X(4)
IF (X(5) .GT. EPSA(2)) X(5) = EPSA(2)
IF (ABS((X(5) - X(4)) / X(4)) .GT. EPSA(3)) GO TO 200
IF (ABS((X(5) - X(3)) / X(3)) .LE. EPSA(3)) GO TO 200
IF (ABS((X(5) - X(2)) / X(2)) .LE. EPSA(3)) GO TO 200

CALL FGAMA(5)

CCHECK QUADRATIC (NO SLOPE) FIT
IF (Y(5) .EQ. 0.0) Y(5) = 1.0E-10
IF (ABS((YES(3) - Y(5)) / Y(5)) .GE. EPSA(4)) GO TO 240

MIN = 5
GO TO 460

240 XX(NP + 3) = X(5)
YY(NP + 3) = Y(5)

CFOUR POINT CUBIC FIT
CALL FOPMIN (XX(NP), YY(NP), X(6), YES(4), ISER)
IF (X(6) .LT. EPSA(1) * X(5)) X(6) = EPSA(1) * X(5)
IF (X(6) .GT. EPSA(2)) X(6) = EPSA(2)
IF (ABS((X(6) - X(5)) / X(5)) .GT. EPSA(3)) GO TO
SUBROUTINE GENHIN

260

MAX=5
GO TO 300
260 CONTINUE
CALL FGAMA (6)
C CHECK CUBIC (NO SLOPE) FIT
IF (Y(6).EQ.0.0) Y(6)=1.0E-10
IF (ABS((YES(4)-Y(6))/Y(6)).GE.EPSA(4)) GO TO 280

MIN=6
GO TO 460
MAX=6
MIN=1
CFIND BEST POINT AND RETURN
MINSV=MIN
DO 340 I=1,MAX
IF (Y(MIN).GT.Y(I)) MIN=I
340 CONTINUE
IF (MIN.EQ.1) GO TO 440
IF (MINSV.NE.MIN) MIN=-MIN
RETURN
CSLOPE OF ZERO INDICATES ONLY POINTS ARE TO BE USED FOR THE CURVE FIT
IF (Y(1).NE.0.0) GO TO 380
CALL FGAMA (1)
380 IF (Y(2).NE.0.0) GO TO 400
IF (X(2).EQ.0.0) X(2)=.75
CALL FGAMA (2)
400 IF (Y(3).NE.0.0) GO TO 420
IF (X(3).EQ.0.0) X(3)=1.25
CALL FGAMA (3)
420 CALL THPM (X,Y,X(4),YES(2))
GO TO 140
CSLOPE POSITIVE
MIN=0
RETURN
MAX=MIN
GO TO 320
END
SUBROUTINE GMAG

C*** CALCULATES THE MAGNITUDE OF THE GRADIENTS

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE

INTEGER SRCHM
IF (OPT.EQ.0. OR .SRCHM.EQ.1) GO TO 20
G1MAG=SQRT(SP(G1,G1,NINDV))
IF (G1MAG.EQ.0.0) CALL ERROR (10H G1MAG=0, ,
10HCK INPUTS )

C CONTINUE
IF (NDEPV.EQ.0) GO TO 40
G2MAG=SQRT(SP(G2,G2,NINDV))
CONTINUE
RETURN
END
SUBROUTINE GRAD

GRAD CALCULATES THE GRADIENTS TO EACH OF THE TARGETS AND TO THE OPTIMIZATION INDEX WITH RESPECT TO THE CONTROLS

COMMON /SEARCH/ GOES HERE
COMMON /SERVC/ GOES HERE
INTEGER SRCHM
EQUIVALENCE (ISUB,I)

GAMAS = 1.0
DO 20 I = 1, NINDV
DU(I) = 0.0
20 CONTINUE

DO 140 K = 1, NINDV
DU(K) = PERT(K)
IF (NDEPV.EQ.0) GO TO 60
DO 40 L = 1, NDEPV
E(L) = ENOM(L)
40 CONTINUE

P1 = P1NOM
KS = K
CALL TRAJ
K = KS

DU(K) = 0.0
G2(K) = 0.0
IF (OPT.EQ.0 .OR. SRCHM.EQ.1) GO TO 80
G1(K) = (P1 - P1NOM) / PERT(K)
CALL PAD (P1NOM, P1, 0)
80 CONTINUE

IF (NDEPV.EQ.0) GO TO 120
DO 100 L = 1, NDEPV
ISUB = L + (K - 1) * NDEPV
CALL PAD (ENOM(L), E(L), 0)
ACOB(ISUB) = (E(L) - ENOM(L)) / PERT(K)
G2(K) = G2(K) + PP2 * ACOB(ISUB) * ENOM(L)
100 CONTINUE

120 CONTINUE
CALL PAD (PERT(K), U(K), N01)
140 CONTINUE
RETURN
END
SUBROUTINE INVH (A, N)

C INPUT
C A(N,N) MATRIX
C N = DIMENSIONS OF A

C OUTPUT
C A(N,N) MATRIX ***** A (OUTPUT) = INVERSE OF A (INPUT)

DIMENSION A(1), XB(25), RTEST(25), IX(25), IY(25)

CALL ZERO (RTEST, 25, 1)
CALL ZERO (XB, 25, 1)

C MAIN LOOP

DO 100 I = 1, N
   S1 = 0.
   DO 40 JJ = 1, N
      IF (XB(JJ)) 40, 1, 40
   1 LA = (JJ - 1) * N
      DO 4 J = 1, N
         IF (RTEST(J)) 4, 2, 4
      2 NB = LA + J
         S2 = ABS (A(NB))
      3 S1 = S2
      LC = JJ
      NC = J
      NG = NB
      CONTINUE
   40 CONTINUE

IF (S1) 5, 5, 6
5 CALL ZERO (A, N, N)
WRITE (6, 200)
200 FORMAT (45HOINVERSION OF A SINGULAR MATRIX WAS ATTEMPTED)
RETURN

   IX(I) = NC
   IY(I) = LC
   RTEST(NC) = 10.
   XB(LC) = A(NG)

C CLOBBER LOOP

NAA = (LC - 1) * N
DO 8 K = 1, N
   IF (NC - K) 9, 8, 9
7 DO 13 KL = 1, N
   IF (KL - LC) 131, 13, 131
131 NI = (KL - 1) * N
   NJ = NAA + K
   NF = NI + K
APPENDIX III — CONSTRAINED TARGETING ROUTINES

SUBROUTINE INVM

51    NH = NI+NC
52    A(NP) = A(NP)-A(NJ)*A(NH)/XB(LC)
53 13 CONTINUE
54     8 CONTINUE
55     DO 11 KK=1,N
56     NE = NAA+KK
57     11 A(NE) = -A(NE)/XB(LC)
58     A(NG) = 1.
59   C END OF CLOBBER LOOP
60   C
61 100 CONTINUE
62 C END OF MAIN LOOPS, START OF POSITIONING LOOPS
63 C
64 C (500 LOOP IS COLUMN REPOSITIONING)
65 C
66 DO 500 I=1,N
67 DO 501 J=1,N
68 NY = (IY(J)-1)*N+I
69 501 RTEST(J) = A(NY)
70 500 A(NY) = RTEST(J)
71 C (600 LOOP IS ROW REPOSITIONING)
72 C
73 DO 600 I=1,N
74 NX = (I-1)*N
75 DO 601 J=1,N
76 NY = NX+IX(J)
77 NZ = IY(J)
78 601 RTEST(NZ) = A(NY)
79 600 A(NZ) = RTEST(J)
80 C END OF POSITIONING LOOPS ***** START COMPUTING INVERSE
81 C
82 DO 800 I=1,N
83 D = XB(I)
84 DO 800 J=1,N
85 NZ = (J-1)*N+I
86 800 A(NZ) = A(NZ)/D
87 C RETURN
88 END
SUBROUTINE ITERO

C PROGRAM ITERO IS THE MAIN ROUTINE TO OBTAIN THE RESULTS OF THE ITERATION USING COMMON SEARCH AND SERVC BLOCKS.

C COMMON SEARCH GOES HERE
C COMMON SERVC GOES HERE

INTEGER SRCHM

C LABELS DATED IN

DATA LCTHA /4H CTH/
DATA LCP /4H CP/
DATA LDP1DS/4H DP1/
DATA LDP2DS/4H DP2/
DATA LDU /4H DU/
DATA LDUMAG/4H DUM/
DATA LE /4H E(I)/
DATA LGAMA /4H GAM/
DATA LGAMAS/4H GAM/
DATA LG1 /4H G1/
DATA LG1MAG/4H G1M/
DATA LG2 /4H G2/
DATA LG2MAG/4H G2M/
DATA LIAC /4H IAC/
DATA LNAC /4H NAC/
DATA LPCTCC/4H PCT/
DATA LPERT /4H PER/
DATA LPG1 /4H PG1/
DATA LPG1M /4H PG1/
DATA LP1 /4H P1/
DATA LP1TRY/4H P1T/
DATA LP2 /4H P2/
DATA LP2TRY/4H P2T/
DATA LRATIO/4H RAT/
DATA LSMAT /4H SMA/
DATA LSTPMX/4H STP/
DATA LU /4H U(I)/
DATA LUMAG /4H UMA/
DATA LWP1 /4H WP1/
DATA LWVEC /4H WVE/
DATA LWU /4H WU(/
DATA LYPRED/4H YPR/

C OUTPUT BLOCK

IF (NOMF.NE.0) GO TO 20
L=(NINDV+5)/6
L=3*L
M=(NDEPV+5)/6
M=2*M
I = L+M+11
CALL PAGER(I)
APPENDIX III — CONSTRAINED TARGETING ROUTINES 91

SUBROUTINE ITERO

54 WRITE (6, 620) .
55 GO TO 300 .
56 20 CONTINUE .
57 CALL PAGER (1000) .
58 IF (NSTEP .NE. 1) GO TO 40 .
59 WRITE (6, 480) .
60 GO TO 300 .
61 40 CONTINUE .
62 M=NSTEP-1 .
63 WRITE (6, 500) M .
64 80 CONTINUE .
65 TEMP (1)=SECONS .
66 CALL TIME (1,0,IJKLNM) .
67 SECONS = IJKLNM .
68 TEMP (1)=SECONS-TEMP (1) .
69 WRITE (6, 560) LCP,TEMP (1) .
70 IF (IDEB.EQ.0) GO TO 100 .
71 WRITE (6, 560) LPERT, (PERT (I),I=1,NINDV) .
72 100 CONTINUE .
73 IF (NDEPV.EQ.0) GO TO 140 .
74 DO 120 J=1,NDEPV .
75 WRITE (6, 560) LSMAT, (ACOB (J+I*NDEPV-NDEPV), I=1,NINDV) .
76 120 CONTINUE .
77 140 CONTINUE .
78 IF (OPT.EQ.0.0.OR.SRCHM.EQ.1) GO TO 160 .
79 WRITE (6, 560) LG1, (G1 (I),I=1,NINDV) .
80 WRITE (6, 560) LG1HAG,G1MAG .
81 160 IF (NDEPV.EQ.0) GO TO 180 .
82 WRITE (6, 560) LG2, (G2 (I),I=1,NINDV) .
83 WRITE (6, 560) LG2MAG,G2MAG .
84 180 IF (OPT.EQ.0.0.OR.SRCHM.EQ.1) GO TO 200 .
85 WRITE (6, 560) LPG1, (PG1(I),I=1,NINDV) .
86 WRITE (6, 560) LPG1H,PG1MAG .
87 200 CONTINUE .
88 IF (IDEB.EQ.1.AND.NSTEP.EQ.2) WRITE (6, 560) LWVEC, (WU(I),I=1,NINDV) 1) .
89 90 IF (ITERF .EQ. 0) GO TO 280 .
91 IF (NEQC .EQ. NDEPV) GO TO 220 .
92 J = JAC(1) .
93 WRITE (6, 580) LNAC, J .
94 WRITE (6, 580) LIAC, (JAC(I+1),I=1,J) .
95 220 CONTINUE .
96 IF (IDEB .EQ. 1) WRITE (6, 560) LRATIO, (RATIO(I),I=1,4) .
97 TEMP (1)=ARCCOS (CTHA) *DPR .
98 WRITE (6, 560) LCTHA,TEMP (1) .
99 WRITE (6, 560) LDP1DS,ZP1DS .
100 WRITE (6, 560) LDP2DS,ZP2DS .
101 WRITE (6, 560) LSTPHX,ZMAX .
102 WRITE (6, 560) LUMAG,ZMAG .
103 WRITE (6, 560) LPCTCC,PCTCC .
104 WRITE (6, 570) LGAMAS,ZTEP .
APPENDIX III - CONSTRAINED TARGETING ROUTINES
SUBROUTINE ITERO

WRITE (6,570) LP1TRY,Z1TRY
WRITE (6,570) LP2TRY,Z2TRY
WRITE (6,570) LYPRED,N00,N00,(ZYES(I),I=1,4)
280 CONTINUE
IF (ITERF .EQ. 2) GO TO 300
IF (NEQC .EQ. NDEPV) GO TO 290
WRITE (6,580) LNAC,NAC
WRITE (6,580) LIAC,(IAC(I),I=1,NAC)
290 CONTINUE
WRITE (6,560) LDP1DS,DP1DS
WRITE (6,560) LDP2DS,DP2DS
WRITE (6,560) LSTPMX,STPHAX
WRITE (6,560) LUMAG,UHAG
WRITE (6,560) LDUMAG,DUMAG
WRITE (6,570) LGAMAS,STEP
WRITE (6,570) LP1TRY,P1TRY
WRITE (6,570) LP2TRY,P2TRY
FOO = 0.0
WRITE (6,570) LYPRED,FOO,FOO,(YES(I),I=1,4)

C ABREVIATED OUTPUT FOR TRIAL STEPS BEGINS HERE

DO 320 I=1,NINDV
TEMP(I)=U(I)+GAMAS*DU(I)
320 CONTINUE
IF (NOHF.NE.0) GO TO 340
WRITE (6,560) LGAMAS,GAMAS
WRITE (6,560) LDU,(DU(I),I=1,NINDV)
340 CONTINUE
WRITE (6,560) LWU,(TEMP(I),I=1,NINDV)
DO 360 I=1,NINDV
TEMP(I)=TEMP(I)/HU(I)
360 CONTINUE
WRITE (6,560) LU,(TEMP(I),I=1,NINDV)
380 CONTINUE
IF (NDEPV.EQ.0) GO TO 420
WRITE (6,560) LE,(E(I),I=1,NDEPV)
390 CONTINUE
IF (OPT.EQ.0) GO TO 440
WRITE (6,560) LWP1,P1
395 TEMP(1)=P1/WOPT
400 WRITE (6,560) LP1,TEMP(1)
405 GO TO 460
420 CONTINUE
IF (IDEB .EQ.-O) RETURN
IF (NOMF .EQ. 0) RETURN
CALL PAGER (1000)
460 CONTINUE
WRITE (6,600)
149 440 CONTINUE
150 WRITE (6,600)
151 460 CONTINUE
WRITE (6,560) LP2,P2
IF (IDEB .EQ. 0) RETURN
IF (NOMF .EQ. 0) RETURN
CALL PAGER (1000)
RETURN
157 C
C FORMATS

480 FORMAT (23H *** NOMINAL TRAJECTORY///)
500 FORMAT (22H *** ITERATION NUMBER , I2///)
560 FORMAT (A4, 5(E15.8,7X),E15.8/6(7X,E15.8)/6(7X,E15.8)/6(7X,E15.8)/6(7X,E15.8))
570 FORMAT (A4,5(E15.8,7X),E15.8)
580 FORMAT (A4,13X,5(I2,20X),I2/6(20X,I2)/6(20X,I2)/20X,I2)
600 FORMAT (1H )
620 FORMAT (///15H-*** TRIAL STEP)

END
SUBROUTINE MATPY (A, B, C, L, M, N)
DIMENSION A(1), B(1), C(1)

C******************************************************************************
DO 100 I=1, L
DO 100 J=1, N
JM1 = J-1
JM1M = JM1*M
II = JM1*L+I
C(II) = 0,
DO 100 K=1, M
JJ = (K-1)*L+I
KK = JM1M+K
100 C(II) = C(II)+A(JJ)*B(KK)
C
RETURN
END
SUBROUTINE MINMYS

MINMYS MINIMIZES THE OPTIMIZATION INDEX
WHILE SATISFYING THE

COMMON /SEARCH/ GOES HERE
COMMON /SERVC/ GOES HERE

INTEGER SRCHM

ITERATION INITIALIZATION

IF (SRCHM.NE.5) GO TO 20
SRCHM=1
IDAV=63
CONTINUE
DO 40 I = 1, NINDV
IF (MODEW .NE. 0) GO TO 40
U(I)=U(I)*SU(I)
PERT(I)=PERT(I)*SU(I)
CONTINUE
40 CONTINUE
60 CONTINUE
CTHAT=COS(CONEPS(1)*RPD)
NEQC = 0
DO 70 I = 1, NDEPV
IAC(I) = I
IF (IDEPVR(I) .EQ. 0) NEQC = NEQC + 1
70 CONTINUE
NAC = NDEPV
IF (OPT .EQ. 0) GO TO 75
IF (SRCHM .NE. 4) GO TO 75
NETF = 63
CONTINUE
75 CONTINUE

ITERATION LOOP

DO 100 NSTEP=1,100
CALL NOMINL
IF (NFLAG .NE. 0) GO TO 120
GRADIENTS COMPUTED
CALL GRAD
IF (NFLAG .NE. 0) GO TO 120
DIRECTION OF SEARCH CALCULATED AND
CONVERGENCE FLAG SET
CALL DELTU
IF (NFLAG .NE. 0) GO TO 120
C

C
SUBROUTINE HINMYS

C FIND THE BEST STEP SIZE IN THE DIRECTION OF SEARCH

C
ITERF = 0
IF (NETF .EQ. 0) GO TO 80
IF (INTRBL .NE. 0) GO TO 80
ITERF = 2
CALL TRYIT1
IF (NFLAG .NE. 0) GO TO 120
I = NTC + NEQC
IF (I .EQ. 0) GO TO 100
ITERF = 1
80 CONTINUE
CALL TRYIT2
IF (NFLAG .NE. 0) GO TO 120

C UPDATE NOMINAL CONTROLS

C
CALL UPNOM

C
100 CONTINUE
120 CONTINUE
GAMAS = 0.0
CALL NOHINL
NFLAG = 0
RETURN
C
END
SUBROUTINE NOMINL

C*** RUNS A NOMINAL TRAJECTORY, SAVING THE STATE VARIABLES AT THE BEGINNING OF EACH PHASE, CALLS TEST AND DISPLY.

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE
DMAG=SQRT(SP(U,U,NINDV))
NOMF=63

CALL TRAJ

P1NOM=P1
P2NOM = 0.0
IF (NDEPV.EQ.0) GO TO 40
NTC = 0
DO 20 I=1,NDEPV
ENOM(I)=E(I)
ITC(I) = 0.0
IF (IDEPVR(I) .EQ. 0) GO TO 10
IF (ENOM(I) .GT. 10.0) GO TO 20
ITC(I) = 63
NTC = NTC + 1
10 CONTINUE
P2NOM = P2NOM + ENOM(I)**2
20 CONTINUE
40 CONTINUE
CALL DISPLY
CALL TEST
RETURN

CALL ERROR (10HUNUSABLE ,10HNOMINAL )
NFLAG=-1
RETURN
END
SUBROUTINE PAD(A,B,MOD)
INTEGER TRACE
DIMENSION LOCREF(1)
DATA MIN, MAX /17, -1/
DATA NAD /0/.
IF (MOD .NE. 0) GO TO 10
CALL DIGDIF(A,B,NDIF1)
X = A
IF (ABS(B) .GT. ABS(A)) X = B
CALL DIGDIF (A+X, B+X, NDIF2)
NDIF = MIN0(NDIF1, NDIF2)
IF (NDIF .GT. MAX) MAX = NDIF
IF (NDIF .EQ. 0) RETURN
IF (NDIF .LT. MIN) MIN = NDIF
RETURN
10 CONTINUE
IF (MIN .GE. 4 .OR. MAX .GE. 6) GO TO 20
AS = A
A = A*10.0**MIN0(4-MIN,6-MAX)
GO TO 30
20 IF (MAX .LE. 6 .OR. MIN .LE. 4) GO TO 40
AS = A
A = A/10.0**MIN0(MIN-4,MAX-6)
IF (B .EQ. 0.0) GO TO 30
IF (ABS(A/B) .LT. 1.0E-10) A = SIGN(B*1.0E-10, A)
30 CONTINUE
NAD = NAD + 1
IF (A .LT. 1000.0 .AND. NAD .LT. 10) GO TO 50
A = AS
40 CONTINUE
NAD = 0
50 CONTINUE
WRITE (6,1) MIN, MAX
1 FORMAT (7H0MIN =, I3/, 7H MAX =, I3/)
END
SUBROUTINE PAGER (N)
DIMENSION HEADER(25)

DATA NPAGE /0 /
DATA KOUNT /0/
DATA HEADER /25*4H /

KOUNT=KOUNT+N
IF (KOUNT.LE.60 ) RETURN
IF (KOUNT.GT.500) GO TO 20
KOUNT=N+3
NPAGE=NPAGE+1
WRITE (6,1000) NPAGE, (HEADER (L),L=1,25)
1000 FORMAT (1H1,120X,5HPAGE ,I4, /1X,25A4/)
RETURN
CONTINUE
KOUNT=3
GO TO 10
END
SUBROUTINE PGM

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE

CALL UPDATS
C THIS CARD SEEMS UNNECESSARY TO ME,
C BUT I LEAVE IT HERE JUST IN CASE THERE WAS A
C SOME REASON FOR IT.
C IF (OPT .EQ. 0.0) GO TO 40
IF (INTRY1 .NE. 0) GO TO 40
IF (INTRBL .NE. 0) GO TO 40
CALL DROP
40 CONTINUE
IF (NAC .LE. NINDV) GO TO 50
C LEAST SQUARES STEP
C CALL BTW (SMAT, SMAT, SSTI, NINDV, NDEPV, NINDV)
CALL INVM(SSTI, NINDV)
CALL ABT (SSTI, SHAT, EHAM, NINDV, NINDV, NDEPV)
INTRBL = 63
50 CONTINUE
K = 0
DO 60 I = 1, NINDV
G2(I) = 0.0
IF (NAC .EQ. 0) GO TO 60
DO 55 J = 1, NAC
K = K + 1
EMAP(K) = -EMAP(K)
L = IAC(J)
EA(J) = ENOM(L)
M = J + (I-1)*NAC
G2(I) = G2(I) + FP2*EA(J)*SMAT(M)
55 CONTINUE
60 CONTINUE
P2NOM = 0.0
IF (NAC .NE. 0) P2NOM = SP(EA, EA, NAC)
IF (NETF .EQ. 0) GO TO 70
IF (INTRBL .NE. 0) GO TO 70
IF (INTRY1 .NE. 0) GO TO 70
C CTHA = 0.0
IF (ABS(PG1MAG) .GT. 1.0E-8) CTHA = SP(G1, PG1, NINDV)/(G1MAG*PG1MAG)
IF (CTHA .GT. 1.0) CTHA = 1.0
IF (ISTART .NE. 0) GO TO 63
IF (NACS .NE. NAC) GO TO 62
IF (IDEPVR(20) .NE. 0) GO TO 62
IF (NAC .EQ. 0) GO TO 65
DO 61 I = 1, NAC
IF (IACS(I) .NE. IAC(I)) GO TO 62
61 CONTINUE
SUBROUTINE PGH

GO TO 65

CONTINUE

ISTART = 63

CONTINUE

RESTART DAVIDON

L = 0

DO 64 I = 1, NINDV

DU(I) = -PG1(I)

S(I) = DU(I)

GP(I) = PG1(I)

L = L + 1

H(L) = FP1

CONTINUE

GO TO 66

CONTINUE

CALL DGH (PG1)

CONTINUE

NACS = NAC

DO 67 I = 1, NAC

IACS(I) = IAC(I)

CONTINUE

GO TO 100

CONTINUE

CALL HATPY (EHAP,EA,DU,NINDV,NAC,1)

CONTINUE

RETURN

END
SUBROUTINE PPT (P,C,H,N,S)

C
C*****P(MXN) *TRS(P) = C(MXN)
C
DIMENSION P(1), C(1)
C
DO 200 I=1,M
DO 200 J=1,N
IJK = I+J-1)*M
C(IJK) = C(IJK)*S
DO 100 K=1,N
KM1SM = (K-1)*M
IK = I+KM1SM
JK = J+KM1SM
100 C(IJK) = C(IJK) + P(IK)*P(JK)
JI = J+(I-1)*M
200 C(JI) = C(IJ)
C
RETURN
END
SUBROUTINE REVISE

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE
C
10 CONTINUE
ICD = 0
IF (NAC .EQ. 0) GO TO 200
TEMP(1) = PG1MAG/G1MAG
IF (TEMP(1) .GT. PGEPS) GO TO 200
CALL MATPY (SSI, SMAT, PROJ, NAC, NAC, NINDV)
CALL MATPY (PROJ, G1, TEMP(1), NAC, NINDV, 1)
DO 60 I = 1, NAC
   60 STEMP(1) = 0.0
   DO 40 J = 1, NINDV
      K = I + (J-1)*NAC
      STEMP(1) = STEMP(1) + SMAT(K)**2
   40 CONTINUE
   TEMP(I) = TEMP(I) * SQRT(STEMP(1))
   WRITE (6, 1) I, TEMP(I)
200 CONTINUE
   STEMP(1) = 0.0
   DO 80 I = 1, NAC
      J = IAC(I)
      IF (IDEPVR(J) .EQ. 0) GO TO 80
      IF (TEMP(I) .GT. STEMP(1)) GO TO 80
      K = IAC(I)
      STEMP(1) = TEMP(I)
   80 CONTINUE
   IF (STEMP(1) .GE. -1.0E-6) GO TO 200
   ICD = K
   ITC(ICD) = 0
   NTC = NTC - 1
   CALL UPDATS
200 CONTINUE
RETURN
1 FORMAT (13H-***COMPONENT,I2,3H = ,E21.14)
END
SUBROUTINE RSEARCH

RSEARCH - READS NAMELIST SEARCH

COMMON /SEARCH/ GOES HERE

COMMON /SERVC/ GOES HERE

INTEGER SRCHM

REAL INDPH

NAMELIST/SEARCH/

1 CONEPS, CONSEX, FITERR, GAMAX, IDEB, IDEPVR, KASE, MAXITR, MODEW

2, NDEPV, NINDV, OPT, PCTCC, PERT, PGEPS, P2MIN, SRCHM, STMNP

3, U, WCON, WOPT, WU

READ (5, SEARCH)

RETURN

END
SUBROUTINE SDM

C*** CALCULATES DU BY THE STEEPEST DESCENT METHOD (-G2)

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE

DO 20 I=1,NINDV
  DU(I)=-G2(I)
20 CONTINUE
RETURN
END
FUNCTION SP (X,Y,N)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* PURPOSE Computes the inner product of two N-dimensional vector

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

DIMENSION

SP=0.0
DO 10 I=1,N
   SP=SP+X(I)*Y(I)
10 CONTINUE
RETURN
END
SUBROUTINE TEST

C THIS ROUTINE TESTS TO SEE IF THE CONVERGENCE CRITERIA HAVE BEEN MET

C COMMON SEARCH GOES HERE
C COMMON SERVC GOES HERE

INTEGER SRCHM

IF (ABS(P2NOM) GT P2MIN) GO TO 50
IF (NETF .EQ. 0) GO TO 20
IF (CTHA .GT. CTHAT) GO TO 60

C CONVERGENCE CRITERIA MET

20 CONTINUE
WRITE (6,160)
NF1AG=63
RETURN

50 CONTINUE
IF (IOPT .NE. 0) GO TO 60
IF (NETF .EQ. 0) GO TO 60
INTRBL = 63
60 IF (NSTEP .LT. 3) GO TO 120

C COMPARE U,P1,P2,G2, WITH THE PREVIOUS ITERATION

RATIO(1)=ABS((UMAG-OLDU)/UMAG)
IF (P1NOM .NE. 0.0) RATIO(2)=ABS((P1NOM-OLDP1)/P1NOM)
RATIO(3)=ABS((P2NOM-OLDP2)/P2NOM)
IF (G2MAG .NE. 0) RATIO(4) = ABS((G2MAG-OLDG2)/G2MAG)

NVOTE=0
DO 80 I=1,4
IF (RATIO(I) .GT. CONEPS(I+1)) NVOTE=1
80 CONTINUE
IF (KREEP .NE. 0 OR NVOTE .NE. 0) GO TO 100

C IF THERE IS NO CHANGE 2 ITERATIONS IN A ROW, STOP

C WRITE (6,180).
GO TO 40
100 CONTINUE
KREEP=NVOTE
120 OLDU=UMAG
OLDP1=P1NOM
OLDP2=P2NOM
OLDG2=G2MAG
IF (NSTEP.LE.MAXITR) GO TO 140
SUBROUTINE TEST

49 NFLAG=-1
50 WRITE (6,200)
51 CONTINUE
52 RETURN
53 C
54 FORMAT (20H-*** PROBLEM SOLVED )
55 FORMAT (55H-*** NO CHANGE IN STATE DURING 2
56 CONSECUTIVE ITERATIONS 1)
57 FORMAT (20H-*** ITERATION LIMIT)
58 END

PROBLEM SOLVED
NO CHANGE IN STATE DURING 2
CONSECUTIVE ITERATIONS
SUBROUTINE THPM(X,Y,XMIN,YMIN)
DIMENSION X(3), Y(3)
X12 = X(1) - X(2)
X23 = X(2) - X(3)
X31 = X(3) - X(1)
ANUM = X23*Y(1) + X31*Y(2) + X12*Y(3)
XS12 = X12*(X(1)+X(2))
XS23 = X23*(X(2)+X(3))
XS31 = X31*(X(3)+X(1))
DENOM = XS12*X(3) + XS23*X(1) + XS31*X(2)
A = -ANUM/DENOM
IF (A .GT. 0.0) GO TO 10
XMIN = 1.0E10
YMIN = -1.0E10
RETURN
10 BNUM = XS23*Y(1) + XS31*Y(2) + XS12*Y(3)
B = BNUM/DENOM
XMIN = -0.5*B/A
C = Y(1) - X(1)*(B + X(1)*A)
YMIN = C + XMIN*(B + XMIN*A)
RETURN
END
SUBROUTINE THPOSM(X,Y,DYDX1,XMIN,YMIN,IERR)

C INPUTS
X=ARRAY OF ABSCISSA VALUES OF THREE
SAMPLE POINTS
Y=ARRAY OF ORDINATE VALUES OF THREE
SAMPLE POINTS
DYDX1=SLOPE AT FIRST SAMPLE POINT

C OUTPUTS
XMIN=ABSCISSA VALUE OF MINIMUM OF
THREE SAMPLE POINTS AND THE SLOPE
AT FIRST POINT
YMIN=ORDINATE VALUE OF MINIMUM OF
ABOVE CUBIC
IERR=FLAG WHOSE NON-ZERO VALUE
INDICATES THAT TWO OF THE
GIVEN X VALUES ARE IDENTICAL.

C
DIMENSION
1 X(3) Y(3) A(4)

C DATA
1 ONE /1. /
2 /TWO /2. /
3 /THREE /3. /

C DATA
1 NO /0 /
2 /N1 /1 /

C IERR=NO
IF (X (3) -X (2) ) 30,80,40
30 CONTINUE
AMBDA=X(2)-X(1)
IF (AMBDA) 70,80,70
80 CONTINUE
IERR=N1
GO TO 60

70 CONTINUE
ALPHA=(X (3) -X (1) ) /AMBDA
FALAM=Y (3)
FLAM=Y (2)
GO TO 50

40 CONTINUE
AMBDA=X (3) -X (1)
IF (AMBDA) 90,80,90
90 CONTINUE
ALPHA=(X (2) -X (1) ) /AMBDA
FALAM=Y (2)
FLAM=Y (3)

50 CONTINUE
IF (ALPHA) 100,80,100
APPENDIX III - CONSTRAINED TARGETING ROUTINES
SUBROUTINE THPOSM

100 CONTINUE
F0 = Y(1)
A(1) = F0
POP = DYDX1
A(2) = POP
ALPHAS = ALPHA**2
APSFLM = ALPHAS * FLAM
OPALP = ONE + ALPHA
OMALP = ONE - ALPHA
APSAMS = AMBDA**2 * ALPHAS
ALFOP = ALPHA * AMBDA * POP
A(4) = ((APSFLM - FALAM) / OMALP + ALFOP +
         POP * OPALP) / (APSAMS * AMBDA)
A(3) = ((FALAM - ALPHA * APSFLM) / OMALP -
         ALFOP * OPALP - POP * (OPALP + ALPHAS

1 /APSAMS
CALL CUBMIN(A, XMIN, YMIN)
60 CONTINUE
RETURN
END
SUBROUTINE TPOSM(X, Y, DYDX1, XMIN, YMIN)
DIMENSION X(2), Y(2)
X2MX1 = X(2) - X(1)
A = (Y(2) - Y(1))/X2MX1**2 - DYDX1/X2MX1
IF (A .GT. 0.0) GO TO 10
XMIN = 1.0E10
YMIN = -1.0E10
RETURN
A2 = 2.0*A
B = DYDX1 - A2*X(1)
XMIN = -B/A2
C = Y(1) - X(1)*(B + X(1)*A)
YMIN = C + XMIN*(B + XMIN*A)
RETURN
END
SUBROUTINE TRAJ

C*** RUNS THE TRAJECTORY PHASE BY PHASE, SETING IN THE INDEPENDENT
C AND SAVING THE DEPENDENT VARIABLES AT THE APPROPRIATE PHASES.
C THE CP TIME FOR THE TRAJECTORY AND OTHER OPTIMIZATION DATA ARE OUT
C
C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE

INTEGER SRCHM
C COMMON /COM/ C(12)
DIMENSION P(13), PO(13), PD(13), Q(13)
DIMENSION DEPTL(25)

DATA DEPTL /25*.01/,
DATA PO /0.0D0,
1  5.0D1,7.1D1,1.12D2,3.0D0,
2  5.0D1,7.1D1,1.12D2,3.0D0,
2  2.6D1,4.5D1,9.9D1,1.5D1/
CALL TIME (1,0,1)
TEMPT = I
DO 999 I = 1, NINDV
   TEMP(I) = (U(I) + GAMAS*DU(I))/HO(I)
999 CONTINUE
C   GO TO (1010,1020,1030), KASE
1010 CONTINUE
C   E(1) = (TEMP(3)-FPP5*(TEHP(1)**2+
F P2*TEMP(2)**2))/DEPTL(1)
C   E(2) = (TEMP(2)-FP1)/DEPTL(2)
C   E(3) = (TEMP(1) - FPP5)/DEPTL(3)
C   E(4) = (TEMP(1) + TEMP(2) - FP2)/DEPTL(4)
C   E(5) = (FP4- (TEMP(1)+TEMP(2)))/DEPTL(5)
C   P1 = TEMP(3)
C   GO TO 2000
C1020 CONTINUE
C   E(1) = (TEMP(3)-FPP5*(TEMP(1)**2+
TEMP(2)**2))/DEPTL(1)
C   E(2) = (TEMP(2)-FP3+FPP5*(FP5*TEMP(1)+
TEMP(1)**2))/DEPTL(2)
C   P1 = TEMP(3)
C   GO TO 2000
C1030 CONTINUE
C   TEMP(10) = (TEMP(1)-FP1)**2
C   TEMP(11) = (TEMP(1)+FP1)**2
C   E(1) = (TEMP(10)-TEMP(2))/DEPTL(1)
C   E(2) = (TEMP(10)+TEMP(2))/DEPTL(2)
C   E(3) = (TEMP(11)-TEMP(2))/DEPTL(3)
C   E(4) = (TEMP(11)+TEMP(2))/DEPTL(4)
C   E(5) = (FP1-(TEMP(1)**2 +

```plaintext
C         P1 = TEMP(1)+TEMP(2)
C2000     CONTINUE
DO 10 I = 1, 13
P(I) = P0(I)
10 CONTINUE
DO 20 I = 1, NINDV
C(I) = TEMP(I)
20 CONTINUE
PD(1) = 1.0
CALL AUXRK(P,PD)
IF (NOMF .NE. 0) WRITE (7,1) (P(J), J=2,13), (PD(J), J=2,13)
DO 40 I = 1, 80
CALL RK(P,PD,Q,1.0E-2,13,100)
IF (I .NE. 1) GO TO 30
E(6) = -PD(2)*1000.0
E(7) = -PD(3)*1000.0
E(8) = -PD(4)*1000.0
E(9) = -PD(5)*1000.0
30 CONTINUE
IF (NOHF .NE.,0) WRITE (7,1) (P(J), J=2,13), (PD(J), J=2,13)
CC IF (P(J) .LT. 1) P(J) = 0.0D0
C30 CONTINUE
CC WRITE (6,123) P
C123 FORMAT (1X,13D10.3)
40 CONTINUE
E(1) = P(2) - 3.0E0
E(2) = P(3) - 3.6E1
E(3) = P(4) - 1.1E1
E(4) = P(5) - 7.0E0
E(5) = AMAX1(P(6),0.0)+AMAX1(P(7),0.0)+
          AMAX1(P(8),0.0) - 1.5E0
78 C     E(6) = P(7) - 0.0E0
79 C     E(7) = P(8) - 0.0E0
80 C     E(8) = P(9) - 0.0E0
81 C     E(10) = -PD(5)*1000.0
82 C     E(11) = -PD(3)*1000.0
83 C     E(12) = -PD(4)*1000.0
84 C     E(13) = AMAX1(P(10),0.0)+AMAX1(P(11),0.0)+
          AMAX1(P(12),0.0) - 1.5E0
E(14) = -PD(5)*1000.0
P2 = 0.0
IF (NAC .EQ. 0) GO TO 88
DO 85 I = 1, NAC
J = IAC(I)
EA(I) = E(J)
85 CONTINUE
P2 = SP(EA,EA,NAC)
88 CONTINUE
IF (SRCHM .EQ. 1) P2=P1+WCON*P2
IF (NOMF .NE. 0) GO TO 90
IF (IDEB .EQ. 0) GO TO 100
```
APPENDIX III — CONSTRAINED TARGETING ROUTINES

SUBROUTINE TRAJ

97      CALL TIME (1,0,I)
98      TEMP(1) = I - TEMPT
99      WRITE (6,120) TEMP(1)
100     CALL PAGER (NO2)
101     IF (GAMAS .EQ. 1.0) GO TO 100
102 90     CONTINUE
103     CALL ITERO
104 100    CONTINUE
105     RETURN
106 C
107 1     FORMAT (12(1X,E10.3))
108 120     FORMAT (25H *** TRAJECTORY CP TIME =,F7.3/)
109     END
SUBROUTINE TRYIT1

C*** MINIMIZES NET PERFORMANCE

C
COMMON /SEARCH/ GOES HERE
COMMON /SERVC/ GOES HERE
DIMENSION EPS(4)

INTRY1 = 63
IF (IOPT .EQ. 0) GO TO 40

ADJUST PCTCC BASED UPON PREVIOUS ITERATION

IF (IHADIT.NE.0) GO TO 20
PCTCC = FP2*PCTOLD
GO TO 40
CONTINUE
PCTCC=PCTOLD*FPP5
CONTINUE
IHADIT = 0
DP1DS=SP(DU,G1,NINDV)
DP2DS=SP(DU,G2,NINDV)
EPS(1) = STMINP
EPS(2) = AMIN1(STPMAX,PCTCC*UMAG)
EPS(3) = CONSEX
EPS(4) = FITERR

C  INITIALIZE TRIAL STEP ARRAY

DO 80 I=1,6
STEP(I)=0.0
P1TRY(I)=0.0
P2TRY(I)=0.0
IF (I.GT. 4) GO TO 80
YES(I) = 0.0
CONTINUE
P1TRY(1)=P1NOM
P2TRY(1)=P2NOM
IF (NAC .EQ. 0) GO TO 95
CALL MATPY (EMAP,EA,TEMP(I),NINDV,NAC,1)
P1TRY(I)+SP(G1,TEMP(I),NINDV)
CONTINUE
STEP(2) = FPP5*EPS(2)

MINIMIZE PERFORMANCE

CALL GENMIN (STEP,P1TRY,DP1DS,EPS,YES,MIN)
IF (MIN.NE.0) GO TO 100
WRITE (6,180)
IF (ISTART .NE. 0) NFLAG = -1
I = 1
GAMAS = 0
SUBROUTINE TRYIT1

110 CONTINUE

PCTOLD = PCTCC*FPP5
IF (GAMAS .NE. STPMAX) PCTOLD = GAMAS/UMAG
ZP1DS=DP1DS
ZP2DS=DP2DS
ZMAG = UMAG
ZMAX = STPMAX
DO 120 J = 1, 6
ZTEP(J) = STEP(J)
Z1TRY(J) = P1TRY(J)
Z2TRY(J) = P2TRY(J)
IF (J .GT. 4) GO TO 120
ZYES(J) = YES(J)
120 CONTINUE
JAC(1) = NAC
DO 130 I = 1, NAC
JAC(I+1) = IAC(I)
130 CONTINUE
C
C UPDATE THE CONTROLS
C
I = MIN - 1
P1NOM=SAVIT(1,I)
P2NOM=SAVIT(2,I)
NTC = 0
DO 140 J=1,NDEPV
ENOM(J)=SAVIT(J+2,I)
IF (IDADEPR(J) .EQ. 0) GO TO 140
IF (ENOM(J) .GT. 1.0) GO TO 140
ITC(J) = 63
NTC = NTC + 1
140 CONTINUE
CALL UPNOM
C
C CALCULATE A TARGETING DU
C
IF (NFLAG .EQ. 0) CALL DELTU
160 CONTINUE
IOPT = 63
INTRY1 = 0
RETURN
C
180 FORMAT (20H OPTIMIZATION UPHILL)
END
SUBROUTINE TRYIT2

C     MINIMIZES THE ERROR
C
C     COMMON /SEARCH/ GOES HERE
C     COMMON /SERVC/ GOES HERE
C
INTEGER SRCHM
DIMENSION EPS(4)
DP1DS=SP(DU,G1,NINDV)
DP2DS=SP(DU,G2,NINDV)
IF (INTRBL .NE. 0) GO TO 10
IF (NETF .EQ. 0) GO TO 10
DP2DS = 0.0
10 CONTINUE
INTRBL = 0

C     INITIALIZE TRIAL STEP ARRAY
C
DO 20 I=1,6
P1TRY(I)=0.0
P2TRY(I)=0.0
STEP(I)=0.0
20 CONTINUE
P1TRY(1)=P1NOM
P2TRY(1)=P2NOM
EPS(1) = STMINP
EPS(2) = AMIN1 (STPMAX,GAMAX*DUMAG)
EPS(3) = CONSEX
EPS(4) = FITERR
STEP(2)=ABS(FP2*P2NOM/DP2DS)
C IF (SRCHM. EQ. 4) STEP (2) = DUMAG
IF (SRCHM. EQ. 4) READ (5,1) STEP (2)
1 FORMAT (E10.0)
STEP(2) = AMIN1 (STEP(2), EPS(2))
IF (DP2DS .EQ. 0.0) STEP (3) = AMIN1
(1.25*DUMAG,EPS(2))
IF (SRCHM. NE. 4) GO TO 40
CALL FGAMA (N02)
IF (P2.GE.P2MIN) GO TO 40
MIN=2
GO TO 60
40 CONTINUE

C     MINIMIZE ERROR
C
CALL GENMIN (STEP,P2TRY,DP2DS,EPS,YES,MIN)
IF (MIN.NE.0) GO TO 60
WRITE (6,100)
GAMAS = 0.0
IF (P2NOM .LT. P2MIN) GO TO 80
IF (ITERF .NE. 0) GO TO 80
NFLAG=-1
SUBROUTINE TRYIT2

RETURN
CONTINUE
I=IABS(MIN)
GAMAS=STEP(I)
P2=P2TRY(I)
IF (P2.LT.P2MIN) RETURN
CONTINUE
INTRBL=63
IHADIT=1
RETURN
C
100 FORMAT (27H DIRECTION OF SEARCH UPHILL)
END
SUBROUTINE UNITDU

C UNITIZE DU

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE

DUMAG = SQRT(SP(DU, DU, NINDV))
IF (DUMAG .EQ. 0.0) RETURN
DO 20 I = 1, NINDV
  DU(I) = DU(I) / DUMAG
20 CONTINUE
RETURN
END
SUBROUTINE UPDATS

C COMMON /SEARCH/ GOES HERE

C COMMON /SERVC/ GOES HERE

M = 0
NAC = NEQC + NTC

DO 60 I = 1, NDEPV
IF (IDEPVR(I) .EQ. 0) GO TO 40
IF (ITC(I) .EQ. 0) GO TO 60

CONTINUE

M = M + 1
IAC(M) = I

DO 50 J = 1, NINDV
K = M + (J - 1)*NAC
L = I + (J - 1)*NDEPV
SMAT(K) = ACOB(L)

CONTINUE

50 CONTINUE

60 CONTINUE

C CALCULATE THE PROJECTED GRADIENT

IF (NAC .NE. 0) GO TO 70
DO 65 I = 1, NINDV
PG1(I) = G1(I)

CONTINUE

65 CONTINUE

GO TO 120

70 CONTINUE

CALL PPT (SMAT, SSTI, NAC, NINDV, 0.0)
CALL INVH (SSTI, NAC)
CALL BTB (SMAT, SSTI, ECHAP, NINDV, NAC, NAC)
IF (OPT .EQ. 0.0) RETURN
CALL MATPY (ECHAP, SMAT, PROJ, NINDV, NAC, NINDV)
IF (NAC .LE. NINDV) GO TO 75
CALL ZERO (PG1, NINDV, NO1)
GO TO 120

75 CONTINUE

DO 100 I = 1, NINDV
PG1(I) = 0.0
DO 80 J = 1, NINDV
K = I + (J-1)*NINDV
TEMP(1) = 0.0
IF (I .EQ. J) TEMP(1) = 1.0
PG1(I) = PG1(I) + (TEMP(1) - PROJ(K)) * G1(J)

80 CONTINUE

100 CONTINUE

120 CONTINUE

PG1MAG = SQRT(SP(PG1, PG1, NINDV))
RETURN

END
SUBROUTINE UPNOM

C COMMON /SEARCH/ GOES HERE
C COMMON /SERVC/ GOES HERE

DO 20 I=1,NINDV
   U(I)=U(I)+DU(I)*GAMAS
   DU(I)=0.0
20 CONTINUE
IF (IDAV .NE. 0) GAMASS = GAMAS/DUMAG
GAMAS=0
RETURN
END
SUBROUTINE WUCAL

*** CALCULATES THE WEIGHTING FOR THE CONTROLS

COMMON /SEARCH/ GOES HERE
COMMON /SERVC/ GOES HERE
INTEGER SRCHM

IF (NSTEP.NE.1) RETURN
IF (INTRY1 .NE. 0) RETURN
IF (NINDV.EQ.0) RETURN
IF (MODEW.EQ.0) RETURN
G2MAG=SQRT(SP(G2,G2,NINDV))

DO 200 I=1,NINDV
IF (NEQC .EQ. 0) GO TO 20
GO TO (20,40,80,120,140,150), MODEW

UNITIZED CONTROL WEIGHTING

20 CONTINUE
WU(I) = 1.0
IF (U(I).EQ.0.0) GO TO 160
WU(I) = FP1/ABS(U(I))
GO TO 160

DICKHAN WEIGHTING (NOT RECOMMENDED)

40 CONTINUE
TEMP(1) = 0.0
DO 60 J=1,NDEPV
IR1=J+ (I-1) *NDEPV
TEMP(1)= TEMP(1) + ABS(ACOB(IR1))
60 CONTINUE
WU(I) = TEMP(1)*WU(I)
GO TO 160

STEINHOFF WEIGHTING (DEFINITELY NOT RECOMMENDED)

80 CONTINUE
WU(I) = 1.0
IF (U(I).EQ.0.0) GO TO 160
WU(I) = 0.0
DO 100 J=1,NDEPV
IR1=J+ (I-1) *NDEPV
WU(I)=WU(I)+ABS((ACOB(IR1)*E(J))/U(I))
100 CONTINUE
GO TO 160

GRADIENT WEIGHTING

WU(I)=ABS(G2(I))/G2MAG
SUBROUTINE WUCAL

53 GO TO 160
54 C
55 C AVERAGED GRADIENT AND CONTROL WEIGHTING
56 C
57 140 CONTINUE
58 IF (G2(I) .EQ. 0.0) GO TO 20
59 TEMP(I)=FP1/G2(I)
60 WU(I)=(FP10*U(I)+.1*TEMP(I))/(U(I)**2+
       TEMP(I)**2)
61 GO TO 160
62 C
63 C HOUSTON WEIGHTING
64 C
65 150 CONTINUE
66 TEMP(I)=0.0
67 DO 155 J = 1, NDEPV
68 K = J+(I-1)*NDEPV
69 TEMP(I)=TEMP(I)+
       ALOG10 (ACOB(K)**2/ACOB(J)**2)
70 155 CONTINUE
71 WU(I)=SQRT (FP10**TEMP(I)/WU(I))
72 C
73 C WEIGHT THE CONTROL, ITS PERTURBATION, ITS LIMITS, AND ITS GRADIENTS
74 C
75 160 CONTINUE
76 TEMP(I)=WU(I)
77 U(I)=U(I)*TEMP(I)
78 PERT(I)=PERT(I)*TEMP(I)
79 G1(I)=G1(I)/TEMP(I)
80 G2(I)=G2(I)/TEMP(I)
81 PG1(I)=PG1(I)/TEMP(I)
82 DO 180 J = 1, NDEPV
83 IR1=J+(I-1)*NDEPV
84 ACOB(IR1)=ACOB(IR1)/TEMP(I)
85 180 CONTINUE
86 200 CONTINUE
87 UMAG = SQRT (SP(U,U,NINDV))
88 RETURN
89 END
SUBROUTINE ZERO (A, M, N)
DIMENSION A(1)

C********************************************
MN = M * N
DO 100 I = 1, MN
   100 A(I) = 0.
C
RETURN
END
PROGRAM CATS

UNIT 0 HAS THE GENERAL INPUTS
UNIT 1 HAS THE WEATHER DATA IN THE FORM
   OF IYEAR, IDAY, IWEATHER
UNIT 2 HAS THE MICRO CLIMATE IN FORM 6012
UNIT 3 HAS THE TREE HEIGHTS IN FORM 6012
UNIT 4 HAS THE BOUNDARIES OF THE INHABITABLE AREAS ON THE GRID
UNIT 6 IS THE OUTPUT FILE FOR PRINTING
UNIT 7 IS THE OUTLINE COORDINATES FOR PLOTTING
UNIT 8 IS THE OUTPUT FILE FOR PLOTS

FOR THE ITH, JTH GRID SQUARE:

G(1, I, J) IS THE NUMBER OF TYPE I CATS
G(2, I, J) IS THE NUMBER OF TYPE II CATS
G(3, I, J) IS THE NUMBER OF TYPE III CATS
G(4, I, J) IS THE NUMBER OF TYPE IV CATS
G(5, I, J) IS THE TOTAL NUMBER OF CATS
G(6, I, J) IS THE DERIVATIVE OF TYPE I
G(7, I, J) IS THE DERIVATIVE OF TYPE II
G(8, I, J) IS THE DERIVATIVE OF TYPE III
G(9, I, J) IS THE DERIVATIVE OF TYPE IV
G(10, I, J) IS THE MICRO CLIMATE NUMBER
G(11, I, J) IS THE TREE HEIGHT NUMBER
G(12, I, J) IS THE I OF PREVIOUS SQUARE
G(13, I, J) IS THE J OF PREVIOUS SQUARE
G(14, I, J) IS THE I OF FOLLOWING SQUARE
G(15, I, J) IS THE J OF FOLLOWING SQUARE

INTEGER*2 IBND

COMMON /CM/
   IDAY, IYEAR, NDAY
   NYEAR, IHEAD, JHEAD
   IMAX, JMAX, IODAY
   IOPLOT, IOSAVE, IOTOTL
   IOTYPE, INPT
   MX
   MAX, MIY
   ITURN, ITILT
   DX, DY
   PHAX, DX
   NBD(110)
   ETAB(5, 3), CTAB(28), STAB(28)
   XB (200), YB (200)
APPENDIX IV - CATERPILLAR MODEL CODE

MAIN PROGRAM

C. IWITH(3,500), IBND(2,1400), G(15,60,110)
DIMENSION IG(15,60,110)
EQUIVALENCE (IG,G)

CALL DFAULT (*0=TUIT:MAT.SAN.0 *)
CALL DFAULT (*1=TUIT:MAT.REAL.1 *)
CALL DFAULT (*2=TUIT:MAT.MICRO.2 *)
CALL DFAULT (*3=TUIT:MAT.TREEH.3 *)
CALL DFAULT (*4=TUIT:MAT.ONE.4 *)
CALL DFAULT (*5=*SINK*)
CALL DFAULT (*6=TUIT:MAT.TRACED *)
CALL DFAULT (*7=TUIT:MAT.TRACED *)
CALL DFAULT (*8=-CAT.PLOTS *)

CALL READ
IF (IOPLOT .NE. 0) CALL PLOTI
IYEAR = 1
IWPT = 1
20 CONTINUE
CALL INITIAL
IF (IODAY .LE. 0) CALL OUTPUT
IF (IYEAR .GT. NYEAR) GO TO 100
40 CONTINUE
CALL STEP
IF (IDAY .EQ. IABS(IODAY) .OR. IODAY .EQ. 0) CALL OUTPUT
IDAY = IDAY + NDAY
IF (IDAY .LE. 80) GO TO 40
IF (IODAY .LT. 0) CALL OUTPUT
CALL DSPRS
IYEAR = IYEAR + 1
GO TO 20
100 CONTINUE
STOP
END
APPENDIX IV - CATERPILLAR MODEL CODE

BLOCK DATA

GENERAL CONSTANT INITIAL VALUES

DTAB IS THE DISTANCE ADULT MOTHS CAN FLY
ETAB IS THE NUMBER OF EGGS AN ADULT LAYS
STAB ARE SINES FOR 0 TO 90 DEGREES
CTAB ARE COSINES FOR 0 TO 90 DEGREES

COMMON /CM/ GOES HERE

DATA C1 ./00996., 120., 307., 00935/
DATA C2 /-.000685, -.00126, -.00055, -.000542/
DATA C3 /-.00151, -.00385, -.0100, -.000289/
DATA C4 /-.01, -.01, -.01, -.01/
DATA DTAB / 20.0, 12.0, 5.0 /
DATA ETAB / 236.0, 50.0, 71.0, 112.0, 3.0, 181.0, 21.0, 46.0, 101.0, 13.0, 140.0, 7.0, 18.0, 85.0, 30.0/
DATA STAB / 1.00000000E-10, .58144829E-01, .11609291E+00, .17364818E+00, .23061587E+00, .28680323E+00, .34202014E+00, .39607977E+00, .44879918E+00, .50000000E+00, .54950898E+00, .59715859E+00, .64278761E+00, .68624164E+00, .72737364E+00, .7660444E+00, .80212319E+00, .83548781E+00, .8660254E+00, .89363264E+00, .91821611E+00, .93969262E+00, .95798951E+00, .97304487E+00, .98480775E+00, .99323836E+00, .99830816E+00, .10000000E+01 /
DATA CTAB / .10000000E+01, .99323836E+00, .98480775E+00, .97304487E+00, .95798951E+00, .93969262E+00, .91821611E+00, .89363264E+00, .8660254E+00, .83548781E+00, .80212319E+00, .7660444E+00, .72737364E+00, .68624164E+00, .64278761E+00, .59715859E+00, .54950898E+00, .50000000E+00, .44879918E+00, .39607977E+00, .34202014E+00, .28680323E+00 /
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>*</td>
<td>(0.23061587 \times 10^0)</td>
</tr>
<tr>
<td>37</td>
<td>*</td>
<td>(1.7364818 \times 10^0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1.1609291 \times 10^0)</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>(5.8144829 \times 10^{-1})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1.00000000 \times 10^{-10})</td>
</tr>
</tbody>
</table>

END
SUBROUTINE READ

READS THE INPUT TO THE PROGRAM

COMMON /CM/ GOES HERE

DIMENSION IMP1(60), IMP2(60)

IMAX  MAXIMUM GRID SIZE IN Y
JMAX  MAXIMUM GRID SIZE IN X
IDAY  CURRENT DAY OF THE YEAR
     (INITIALLY, THE DAY ON WHICH TO START THE SIMULATION)
NDAY  NUMBER OF DAYS AT A TIME TO BE INTEGRATED
NYEAR  NUMBER OF YEARS TO SIMULATE
       (SIMULATION ENDS WHEN IYEAR = NYEAR OR THE FIRST DAY OF NYEAR + 1)
IODAY 0 THE DAY TO OUTPUT
       = 0  OUTPUT EVERY DAY
       [ 0  OUTPUT FIRST AND LAST DAY AND IABS OF IODAY
IOPLOT 2 CALCOMP PLOT LINED
       = 1  CALCOMP PLOT OUTLINED
       = 0  PRINTER DIAGRAM
       =-1 ADAGE PLOT OUTLINED
       =-2 ADAGE PLOT LINED
IOSAVE 0 PLOT IMMEDIATELY
       = 0  SAVE IT PLOTS ON FILE 8
IOTOTL 0 DON'T OUTPUT TOTALS AS NUMBERS
       = 0  OUTPUT TOTALS AS NUMBERS
IOTYPE 0 NO TYPES OUTPUT
       = 1  TYPE 1
       = 2  TYPE 2
       = 3  TYPE 3
       = 4  TYPE 4
       = 5  TOTAL OF THE TYPES
MIX  MIN X AFTER ROTATION
MAX  MAX X AFTER ROTATION
MIY  MIN Y AFTER ROTATION
MAY  MAX Y AFTER ROTATION
MAXPOP  MAX POP ON ANY GRID SQUARE
       (USED FOR SCALING PLOTS)
ITURN  ANGLE RIGHT OF STRAIGHT TO VIEW PLOTS
ITILT  ANGLE ABOVE HORIZONTAL TO VIEW PLOTS

CALL FREAD (0,'17I: ',IMAX,JMAX,IDAY,NDAY,
       NYEAR,IODAY,IOPLOT,IOSAVE
       IOTOTL,IOTYPE,MIX,MAX,MIY,MAY,
       MAXPOP,ITURN,ITILT)

CALL ZERO
CALL FREAD (-2,'ENDFILE',1)

CONTINUE
CALL FREAD (0, '2I:', I, J, &20)
CALL FREAD (*',*REAL VECTOR:', G(1, I, J), 4)
G(5, I, J) = 0
DO 10 K = 1, 4
  G(5, I, J) = G(5, I, J) + G(K, I, J)
10 CONTINUE
GO TO 5
20 CONTINUE
I = 1
30 CONTINUE
CALL FREAD (1, 'INTEGER VECTOR:', INWETH(1, I), 3, &60)
I = I + 1
GO TO 30
DO 60 I = 1, IMAX
READ (2, 1) (IMP1(J), J = 1, JMAX)
READ (3, 1) (IMP2(J), J = 1, JMAX)
DO 80 J = 1, JMAX
  IF (IMP2(J) .EQ. 0) IMP2(J) = -2
  IF (IMP2(J) .EQ. 4) IMP2(J) = -2
  IF (IMP2(J) .EQ. 3) IMP2(J) = 0
  IMP2(J) = IMP2(J) + 1
  IG(10, J, I) = IMP1(J)
  IG(11, J, I) = IMP2(J)
80 CONTINUE
100 CONTINUE
K = 0
DO 200 I = 1, IMAX
  READ (4, 2) N
  NBND(I) = N
  READ (4, 3) ((IBND(L, J+K), L = 1, 2), J = 1, N)
  K = K + N
200 CONTINUE
DO 300 I = 1, 200
  CALL FREAD (7, '2R:', YB(I), XB(I))
300 CONTINUE
RETURN
1 FORMAT (60I2)
2 FORMAT (I2)
3 FORMAT (40(I2, A1))
SUBROUTINE ZERO

C CALLED ONCE TO INITIAL THE G ARRAY TO 0.0

C COMMON /CM/ GOES HERE

JP1 = JMAX + 1
DO 100 I = 1, IMAX
DO 100 J = 1, JP1
DO 100 K = 1, 15
G(K, J, I) = 0.0
100 CONTINUE
RETURN
END
SUBROUTINE PLOTI

C PLOT INITIALIZATION

C XB AND YB ARE THE COORDINATES OF THE
C ROTATED OUTLINE

C COMMON /CM/ GOES HERE
SCALE = 60.0/HAXPOP
CALL ROTATI(FLOAT(ITURN),90.0-ITILT)
DX = 4.0/(MAX-MIX)
DY = 4.0/(MAX-MIY)
IF (IABS(IOPLOT) .NE. 1) RETURN
DO 20 I = 1, 200
X = XB(I)
Y = YB(I)
Z = 0
CALL ROTAT(Y,X,Z)
XB(I) = (X-MIX)*DX
YB(I) = (Y-MIY)*DY
IF (IOPLOT .EQ. -1) CALL PLOTB(XB(I),YB(I),2)

20 CONTINUE
RETURN
1 FORMAT (10X,F3.0,10X,F3.0)
END
SUBROUTINE PLOTB(X,Y,IP)

C PLOT BRANCH ROUTINE
C CALLS ADAGE PLOT IF IOPLOT [ 0
C ELSE USES CALCOMP PLOTS
C
C COMMON /CM/ GOES HERE
IF (IOPLOT .GT. 0) GO TO 20
CALL PLOTA (X,Y,IP)
RETURN
CONTINUE
CALL PLOT (X,Y,IP)
RETURN
END
SUBROUTINE PLOTA (X,Y,IPEN)
C ADAGE PLOT ROUTINE
C COMMON /CM/ GOES HERE
LOGICAL*1 PENS(6000)
LOGICAL FIRST / .TRUE. /
DIMENSION XS(6000), YS(6000)
DATA MAXVEC /6000/
DATA ISPOT,NSPOT / 2*1 /
IF( .NOT. FIRST ) GO TO 35
IF (NSPOT .NE. 1) GO TO 25
CALL AGTCON (XS(1),4,5,0)
CALL AGTCON (XS(2),6,3,0)
CALL AGTCON (XS(3),8,6,0)
CALL AGTCON (XS(4),10,2,0)
XS(5) = 0
NSPOT = 6
CALL AGTCON (LITON,34,1,0)
CALL AGTCON (LITOFF,36,1,0)
IPAGT = 0
25 CONTINUE
IF (IOPLOT .EQ. -1) GO TO 30
CALL ADAGEB (XS,NSPOT-1,1,.TRUE.,2)
FIRST = .FALSE.
26 GO TO 35
27 CONTINUE
28 IF (IPEN .EQ. 3) GO TO 32
CALL AGTCVT (XS(NSPOT),X*2.0,Y*2.0,IPAGT,0)
NSPOT = NSPOT + 1
IPAGT = 1
32 RETURN
33 CONTINUE
34 CALL ADAGEB (XS,NSPOT-1,1,.TRUE.,2)
35 FIRST = .FALSE.
36 CONTINUE
37 C*****CHECK FOR A PEN OF -3 ... ALL THROUGH *****
38 IF( IPEN .LE. -3 ) GO TO 400
IF( ISPOT .LT. MAXVEC ) GO TO 200
PRINT 62 , IDAY, IYEAR
62 FORMAT ( '0 VECTOR OVERFLOW IDAY, IYEAR = ', I3, 10X, I3)
STOP
200 CONTINUE
XS( ISPOT ) = X
YS( ISPOT ) = Y
PENS( ISPOT ) = IPEN .EQ. 2
ISPOT = ISPOT + 1
RETURN
400 ISPOT = ISPOT - 1
IF( ISPOT .LE. 0 ) RETURN
DO 450 I = 1 , ISPOT
450 C*****SET THE PEN BIT **************
IPAGT = 0
IF ( PENS ( I ) ) IPAGT = 1
XAGT = XS(I) * 2.0
CALL AGTCVT ( XS(I), XAGT, YS(I) * 2.0, IPAGT, 0)
CONTINUE
IF ( ISPOT .LT. 6000 ) ISPOT = ISPOT + 1
CALL AGTCVT ( XS(ISPOT), 0, 0, 0, 1 )
CALL ADAGEB ( XS, ISPOT, NSPOT, .FALSE., 2)
CALL WAITB (100)
CALL ADAGEB ( LITON, 1, 5 , .FALSE., 2)
CALL WAITB (150)
CALL ADAGEB ( LITOFF, 1, 5, .FALSE., 2)
CALL WAITB (50)
ISPOT = 1
RETURN
END
SUBROUTINE ADAGEB (I,J,K,L,M)

C ADAGE BRANCH ROUTINE
C SAVES THE DATA INSTEAD OF DISPLAYING IT
C IF IOSAVE NON-ZERO
C
C COMMON /CM/ GOES HERE
DIMENSION I(J)
IF (IOSAVE .NE. 0) GO TO 100
CALL AGTDSP (I,J,K,L,M)
RETURN
100 CONTINUE
WRITE (8) M,L,K,J,I
RETURN
END
SUBROUTINE WAITB (N)

C WAITING BRANCH ROUTINE
C CALLS FOR A WAIT DURING DISPLAY UNLESS
C THE DATA IS BEING SAVED FOR DISPLAY LATER

C COMMON /CM/ GOES HERE
DATA I, J, K, L, M /1, 1, -1, 1, 1/
IF (IOSAVE .NE. 0) GO TO 100
CALL RTWAIT (N)
RETURN

100 CONTINUE
K = -N
WRITE (8) I, J, K, L, M
RETURN
END
SUBROUTINE INITAL

INITIALIZES THE G ARRAY

IHEAD AND JHEAD POINT TO THE FIRST
NON-ZERO GRID SQUARE IN G. THEN EACH
NON-ZERO GRID SQUARE POINTS TO THE
NEXT NON-ZERO ONE AND THE PREVIOUS ONE.
THIS CREATES A DOUBLE LINKED LIST
OF NON-ZERO GRID SQUARES.

COMMON /CM/ GOES HERE

IRST = 0
IHEAD = 0
JHEAD = 0
DO 40 I = 1, IMAX
   DO 20 J = 1, JMAX
      IF (G(5,J,I) .EQ. 0) GO TO 20
      IF (IRST .NE. 0) GO TO 10
      G(12,J,I) = 0.0
      G(13,J,I) = 0.0
      IS = I
      JS = J
      IHEAD = IS
      JHEAD = JS
      IRST = 1
      CONTINUE
   10 IG(14,JS,IS) = I
      IG(15,JS,IS) = J
      IG(12,J,I) = IS
      IG(13,J,I) = JS
      IS = I
      JS = J
   20 CONTINUE
   40 CONTINUE
      IG(14,JS,IS) = 0
      IG(15,JS,IS) = 0
      IF (IYEAR .NE. 1) IDAY = 1
      IF (IHEAD .EQ. 0) CALL HANGUP
      RETURN
END
SUBROUTINE HANGUP

C
C CALLED IF THE CATERPILLARS BECOME EXTINCT
C
COMMON /CM/ GOES HERE
WRITE (6,1) IDAY, IYEAR
STOP
1 FORMAT (30H DISASTER...THEY ALL DIED ON , I2,14HTH DAY IN THE , I2,10HTH YEAR!!)
END
SUBROUTINE OUTPUT

C DEPENDS ON WHICH TYPE OF OUTPUT IS DESIRED

C COMMON /CM/ GOES HERE

LOGICAL L1, L2
EQUVALENCE (L1,IOTYPE), (L2,MASK)

DO 100 II = 1, 5
MASK = 2** (II-1)
L2 = L1 .AND. L2
IF (MASK .EQ. 0) GO TO 100
CALL TOTALS (II)
IF (IOPLOT .NE. 0) GO TO 40
CALL PRINTER (II)
GO TO 100

40 CONTINUE
CALL PLOTTER (II)

100 CONTINUE
RETURN

END
SUBROUTINE TOTALS (II)

COMMON /CM/ GOES HERE

PMAX = 0
GTOT = 0
I = IHEAD
J = JHEAD

20 CONTINUE
GTOT = GTOT + G(II,J,I)
IF (G(II,J,I) .GT. PMAX) PMAX = G(II,J,I)
IS = IG(14,J,I)
IF (IS .EQ. 0) GO TO 25
J = IG(15,J,I)
I = IS
GO TO 20

25 CONTINUE
IF (IOTOTL .EQ. 0) GO TO 100
IF (IOPLOT .EQ. 0) GO TO 30
WRITE (6,2) IYEAR,IDAY, PMAX,GTOT
GO TO 100

30 CONTINUE
WRITE (6,1) IYEAR, IDAY, PMAX, GTOT

RETURN

1 FORMAT (30H-STATE OF SYSTEM IN YEAR
12X,I2,10H DAY ,I2/,
30H POP MAX USED FOR SCALING , E14.7/
30H TOTAL POPULATION , E14.7)

2 FORMAT (30H-STATE OF SYSTEM IN YEAR
12X,I2,10H DAY ,I2/,
30H POP MAX USED FOR SCALING , E14.7/
30H TOTAL POPULATION , E14.7)
SUBROUTINE PRNTER (II)
C
C OUTPUTS POPULATION DISTRIBUTIONS ON THE PRINTER
C
C COMMON /CM/ GOES HERE
INTEGER*2 ISYM, LEN, LINE
DIMENSION LINE(63), ISYM(36)
DATA LINE /63*1H /
DATA LEN /126/
DATA ISYM /1H,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,
         1H1A,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,
         1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,
         1HU,1HV,1HW,1HX,1HY,1H2/
1 IF (PMAX .LE. 0) GO TO 100
2 PMAX = 35/PMAX
3 M = 0
4 DO 40 I = 1, IMAX
5      DO 30 J = 1, JHAX
6          IS = G(II,J,I)*PMAX + 1.99999
7      30 CONTINUE
8      LINE(2) = LINE(1)
9      KMAX = NBD(I)
10     DO 35 K = 1, KMAX
11       M = M + 1
12      35 CONTINUE
13      L = IBND(1,M) + 1
14      IF (LINE(L) .NE. LINE(1) .AND. L .LE. JHAX)
15          WRITE (6,1)
16      LINE(L) = IBND(2,M)
17     40 CONTINUE
18     100 CONTINUE
19     RETURN
20 1 FORMAT (20H BOUNDARY ERROR )
21 RETURN
22 END
SUBROUTINE PLOTER (II)

C

C OUTPUTS POPULATION DISTRIBUTIONS ON THE PLOTTER

C

C

IHIST = 0  PREVIOUS POINT = 0

IHIST = 1  PREVIOUS POINT = 0

IHIST = 2  TWO PREVIOUS POINTS = 0

C

C

JHIST = 0  PREVIOUS POINT ON GRID

JHIST = 1  PREVIOUS POINT OFF GRID

JHIST = 2  TWO PREVIOUS POINTS OFF GRID

C

C

XO  PREVIOUS X COORDINATE

YO  PREVIOUS Y COORDINATE

C

C COMMON /CM/ GOES HERE

CALL HSLOT (0.0,0.0,0,CLEAR)

IF (IOPLOT .NE. 1) GO TO 3

IIP = 3

DO 2 I = 1, 200

CALL PLOTB (XB(I), YB(I), IIP)

IIP = 2

2 CONTINUE

3 CONTINUE

K = 1

DO 14 I = 1, IMAX

IHIST = 0

JHIST = 0

L = NBD(I) • K - 1

JS = IBND(1,K) - 1

JF = IBND(1,L) - 1

IP = -1

DO 13 J = JS, JF

IF (J .NE. 0) GO TO 4

Z = 0

IHIST = 1

JHIST = 1

GO TO 125

4 CONTINUE

IF (IHIST .NE. 2) GO TO 6

IF (IABS(IOPLOT) .EQ. 1) GO TO 5

IF (G(10,J,I) .NE. 0) GO TO 5

IHIST = 0

JHIST = 1

GO TO 12

5 CONTINUE

IF (G(II,J,I) .NE. 0) GO TO 55

XO = J

YO = I

GO TO 13

55 CONTINUE

CALL ROTAT(YO,XO,ZO)
SUBROUTINE PLOTER

54 X = (XO-MIX)*DX
55 Y = (YO-MIY)*DY
56 IIP = -2
57 IF (IABS(IOPLOT) .EQ. 1) IIP = -1
58 CALL HSPLOT(X,Y,IIP,CLEAR)
59 IHIST = 0
60 GO TO 12
61 CONTINUE
62 IF (IABS(IOPLOT) .EQ. 1) GO TO 10
63 IF (JHIST .NE. 2) GO TO 8
64 IF (G(10,J,I) .NE. 0) GO TO 7
65 XO = J
66 YO = I
67 ZO = 0
68 GO TO 13
69 CONTINUE
70 CALL ROTAT(YO,XO,ZO)
71 X = (XO-MIX)*DX
72 Y = (YO-MIY)*DY
73 IIP = -1
74 CALL HSPLOT(X,Y,IIP,CLEAR)
75 JHIST = 0
76 IHIST = 1
77 GO TO 10
78 CONTINUE
79 IF (G(10,J,I) .EQ. 0) GO TO 85
80 JHIST = 0
81 GO TO 10
82 CONTINUE
83 IF (IHIST .EQ. 0) GO TO 9
84 XO = J
85 YO = I
86 ZO = 0
87 JHIST = 2
88 GO TO 13
89 CONTINUE
90 JHIST = 1
91 GO TO 12
92 CONTINUE
93 IF (G(I1,J,I) .EQ. 0) GO TO 105
94 IHIST = 0
95 GO TO 12
96 CONTINUE
97 IF (IHIST .EQ. 0) GO TO 11
98 XO = J
99 YO = I
100 ZO = 0
101 IHIST = 2
102 GO TO 13
103 CONTINUE
104 IHIST = 1
105 CONTINUE
106 Z = G(I1,J,I)
107 IF (Z .NE. 0) Z = ALOG10(Z) * SCALE
SUBROUTINE PLOTER

108 CONTINUE
109 X = J
110 Y = I
111 CALL ROTAT(Y,X,Z)
112 X = (X-MIX)*DX
113 Y = (Y-MIY)*DY
114 CALL HSPLOT(X,Y,IP,CLEAR)
115 IP = -2
116 CONTINUE
117 K = L + 1
118 CONTINUE
119 CALL PLOTB(12.0,0.0,-3)
120 RETURN
121 END
SUBROUTINE ROTAT

C
C ROTATES 3-D VECTOR TO PROPER PERSPECTIVE
C
C*************************************************************************************************
C* ROTAT
C*
C* ROTATE THE X,Y,Z COORDINATES
C*
C*************************************************************************************************

XT = X
YT = Y
ZT = Z
X = XT*XC1 - YT*XC2 + ZT*XC3
Y = XT*YC1 + YT*YC2 - ZT*YC3
Z = XT*ZC1 + YT*ZC2 + ZT*ZC3
RETURN
ENTRY ROTATI( TURN, TILT )

555 ASPECT = 0.0
DATA RC / .1745329E-1 /
T = TURN * RC
STURN = SIN( T )
CTURN = COS( T )
T = TILT * RC
STILT = SIN( T )
CTILT = COS( T )
T = ASPECT * RC
SASP = SIN( T )
CASP = COS( T )

XC1 = CTURN*CTILT
XC2 = STURN*CTILT
XC3 = STILT
YC1 = STURN*CASP + CTURN*STILT*SASP
YC2 = CTURN*CASP - STURN*STILT*SASP
YC3 = CTILT*SASP
ZC1 = STURN*SASP - CTURN*STILT*CASP
ZC2 = CTURN*SASP + STURN*STILT*CASP
ZC3 = CTILT*CASP
RETURN
END
SUBROUTINE HSPLOT( XX , YY , IPENN, CLEAR )

C***********************************************************************
C* SUBROUTINE TO FACILITATE THE PLOTTING
C* OF SURFACES
C*
C* DIAGRAMS. HSPLOT WILL NOT PERMIT A LINE TO
C* BE DRAWN AT A YHEIGHT
C* LESS THAN THE YHEIGHT REGISTERED IN THE
C* YHT ARRAY FOR ANY X
C*
C* POSITION.
C*
C*
C* THUS IF A SURFACE IS DRAWN WITH THE AREA
C* NEAREST THE
C* OBSERVER BEING DRAWN FIRST HIDDEN LINES WILL BE
C* REMOVED
C* FROM THE SURFACE BY THIS ROUTINE
C*
C*
C* THE OUTPUT IS TO ROUTINE 'PLOT' AND IS
C* DESIGNED TO INTERFACE
C* DIRECTLY TO THE PLOT ROUTINES. ON OUTPUT:
C* IPEN = 2 IS PEN DOWN
C* = 3 IS PEN UP.
C*
C*
C* RESTRICTION: PLEASE - NO PLOT LONGER THEN 30
C* INCHES.
C*
C* IF...
C* IPEN = +1 , +2 , +3 THE MOVEMENT IS MADE
C* WITHOUT CHECKING.
C* IPEN = 0 THE YHT ARRAY IS CLEARED TO -0.5.
C* IPEN = -1 THE PEN IS MOVED VIRTUALLY. IF
C* THE POSITION INDICATED
C* IS IN THE CLEAR THE PEN IS MOVED
C* THERE UP. IF THE
C* POSITION IS HIDDEN THE PEN IS NOT
C* ACTUALLY MOVED
C* BUT THE POSITION INDICATORS ARE
C* UPDATED.
C* IPEN = -2 THE MOVEMENT IS MADE WITH
C* CHECKING AND THE YHT ARRAY
C* IS UPDATED.
C* IPEN = -3 THE MOVEMENT IS MADE WITH
SUBROUTINE HSPLT

C* CHECKING BUT THE YHT ARRAY
C* IS NOT UPDATED
C* IPEN = -4 THE MOVEMENT IS ONLY SIMULATED,
C* THE YHT ARRAY IS
C* UPDATED BUT THE PEN IS NOT
C* ACTUALLY MOVED.
C*
C* CLEAR - WILL BE RETURNED .TRUE. IF THE PEN
C* WAS ENTIRELY IN THE
C* CLEAR (NOT HIDDEN IN ANY WAY). IT
C* WILL BE .FALSE. OTHERWISE*
C* IT IS USED PRIMARILY TO OPTIMIZE PEN
C* UP AND DOWN MOTION BY *
C* ELIMINATING THE NEED TO UNNECESSARILY
C* LIFT THE PEN IF A
C* LINE IS TO BE RETRACED.
C*
C*

DIMENSION YHT(3201)

C*** CHECKING BUT THE YHT ARRAY
C* IS NOT UPDATED
C* IPEN = -4 THE MOVEMENT IS ONLY SIMULATED.
C* THE YHT ARRAY IS
C* UPDATED BUT THE PEN IS NOT
C* ACTUALLY MOVED.
C*
C* CLEAR - WILL BE RETURNED .TRUE. IF THE PEN
C* WAS ENTIRELY IN THE
C* CLEAR (NOT HIDDEN IN ANY WAY). IT
C* WILL BE .FALSE. OTHERWISE*
C* IT IS USED PRIMARILY TO OPTIMIZE PEN
C* UP AND DOWN MOTION BY *
C* ELIMINATING THE NEED TO UNNECESSARILY
C* LIFT THE PEN IF A
C* LINE IS TO BE RETRACED.
C*
C*

C******************************
INTEGER YHT(3201)
C
C IF PEN = .TRUE. THE PEN IS IN AN AREA WHERE
C PLOTTING MAY OCCUR

C
LOGICAL CLEAR
LOGICAL PEN , FLAG4 , FLAG3
DATA NERROR/0/
X = XX
Y = YY
IPEN = IPENN
IF ( IPEN .NE. 0 ) GO TO 100
DO 50 I = 1, 3201
50 YHT(I) = (-0.5)
PEN = .TRUE.,
RETURN
IF ( IPEN .LT. 0 ) GO TO 200
CALL PLOTB(X,Y,IPEN)
XO = X
YO = Y
IX = X * 100.0 + 0.5
IX = IX + 200
PEN = .TRUE.
IF ( YHT(IX) .GT. Y + 0.005 ) PEN = .FALSE.
RETURN
IF ( IPEN .NE. (-1) ) GO TO 201
XO = X
YO = Y
SUBROUTINE HSPLOT

69 IX = X * 100.0 + 0.5
70 IX = IX + 200
71 PEN = .FALSE.
72 IF( YHT( IX ) .GT. Y + 0.005) RETURN
73 PEN = .TRUE.
74 CALL PLOTB( X, Y, 3)
75 RETURN
76 CONTINUE

C****** 'CLEAR' WILL BE TRUE IF THE LINE IS NOT HIDDEN ON ANY ********
77 C****** PORTION OF THE JOURNEY

78 CLEAR = PEN

C****** RANGE OF THE X MOVEMENT
80 IF( X .LT. ( -1.9 ) .OR. X .GT. .30.0 ) GO TO 99
81 IX = X * 100.0 + 0.5
82 IXO = XO * 100.0 + 0.5
83 DX = IABS( IX - IXO )
84
C**** THE Y INCREMENT
85 DYI = YO
86 IF( DX .NE. 0.0 ) DYI = DYI / DX
87 ADYI = ABS( DYI )
88 IXEND = IX + 200
89 IXLOC = IXO + 200
90 INC = 1
91 IF( IXLOC .GT. IXEND ) INC = ( -1 )
92 IXE = IXEND + INC
93 YS = YO

C
C PREAMBLE -

C CHECK FOR NO X MOVEMENT.
99 IF( DX .NE. 0.0 ) GO TO 340
100 YS = Y
101 YSOLD = YS
102 IXLO = IXLOC

C
103 C*************** START OF MAIN LOOP

104 CONTINUE

105 C
106 300 CONTINUE
107 IF( YHT( IXLOC ).GT. YS + 0.005) GO TO 330
108 C
109 PEN IS IN THE CLEAR SO UPDATE THE 'YHT' ARRAY AND MOVE
110 C THE PEN UP TO THE LAST POSITION WITH THE PEN UP IF THE LAST
111 C POSITION WAS HIDDEN.
112 C
113 IF( PEN ) GO TO 350
114 IXLOC = FLOAT( IXLO - 200 ) * 0.01
115 C
116 C USE THE Y-VALUE THAT WILL CAUSE
SUBROUTINE HSPLOT

THE SMALLEST VARIATION IN THE Y DIRECTION.

YPLOT = YSOLD

IF( ADYI.GT. ABS(YHT(IXLO) - YHT(IXLOC)) ) YPLOT = YHT(IXLO)

CALL PLOTB(XLOC, YPLOT, +3)

PEN = .TRUE.

YHT(IXLOC) = YS

GO TO 340

CONTINUE

THE PEN IS HIDDEN - CAN'T PLOT IN THIS AREA. IF THE LAST POSITION WAS IN THE CLEAR MOVE THE PEN THERE DOWN.

IF( .NOT., PEN ) GO TO 340

XLOC = FLOAT(IXLO - 200) * 0.01

YPLOT = YS

IF( ADYI.GT. ABS(YHT(IXLO) - YHT(IXLOC)) ) YPLOT = YHT(IXLOC)

CALL PLOTB(XLOC, YPLOT, +2)

PEN = .FALSE.

CLEAR = .FALSE.

CONTINUE

YSOLD = YS

YS = YS + DYI

IXLO = IXLOC

IXLOC = IXLOC + INC

IF( IXLOC .NE. IXE ) GO TO 300

C************** END OF MAIN LOOP**************

SAVE THE LAST POSITION. DON'T ACTUALLY MOVE THE PEN IF IT IS HIDDEN.

XO = X

YO = Y

IF( .NOT., PEN ) RETURN

CALL PLOTB(X, Y, 2)

RETURN

CONTINUE

IF( NERROR.GT.0 ) RETURN

NERROR = NERROR + 1

WRITE( 6, 991 ) X

991 FORMAT( 1H0, 51HERRO IN CALL TO 'HSPLOT', ATTEMPT TO PLOT AT 1= , G16.8 , 9H INCHES. / )

RETURN

END
SUBROUTINE STEP

C GOES THROUGH LINK LIST OF G'S ONCE TO
C INTEGRATE AHEAD ONE STEP OF N DAYS.
C
C IHPT IS THE POINTER TO THE WEATHER FOR THIS
C DAY
C
C COMMON /CM/ GOES HERE
C
CONTINUE
10 IF (IYEAR .LT. IWETH(2,IHPT)) GO TO 40
11 IF (IYEAR .EQ. IWETH(2,IHPT)) GO TO 20
12 IWPT = IWPT + 1
13 GO TO 10
14 20 CONTINUE
15 IF (IDAY .LE. IWETH(1,IHPT)) GO TO 40
16 IWPT = IWPT + 1
17 GO TO 20
18 40 CONTINUE
19 100 I = IHEAD
20 J = JHEAD
21 50 CONTINUE
22 CALL INTGRT (G(1,J,I),G(1,J,I),IWETH(3, IWPT))
23 IF (G(5,J,I) .NE. 0) GO TO 60
24 IF (IG(12,J,I) .NE. 0) GO TO 55
25 IHEAD = IG(14,J,I)
26 JHEAD = IG(15,J,I)
27 IF (IHEAD .EQ. 0) CALL HANGUP
28 IG(12,JHEAD,IHEAD) = 0
29 IG(13,JHEAD,IHEAD) = 0
30 GO TO 60
31 55 CONTINUE
32 IP = IG(12,J,I)
33 JP = IG(13,J,I)
34 IG(14,JP,IP) = IG(14,J,I)
35 IG(15,JP,IP) = IG(15,J,I)
36 IP = IG(14,J,I)
37 IF (IP .EQ. 0) GO TO 80
38 JP = IG(15,J,I)
39 IG(12,JP,IP) = IG(12,J,I)
40 IG(13,JP,IP) = IG(13,J,I)
41 60 CONTINUE
42 II = IG(14,J,I)
43 J = IG(15,J,I)
44 I = II
45 IF (I .NE. 0) GO TO 50
46 80 CONTINUE
47 RETURN
48 END
SUBROUTINE INTGRT (GS, IGS, IW)

C INTEGRATES ONE GRID SQUARE NDAYS AHEAD IN TIME

C COMMON /CM/ GOES HERE

DIMENSION GS(12), IGS(12)
PCT1 = GS(1)/GS(5) - .2
IF (PCT1 .GT. 0) PCT1 = 0
LOCW = IW*IGS(10)
GS(5) = 0
DO 20 I = 1, 4
GS(I+5) = (C1(I) + C2(I)*IDAY + C3(I)*LOCW + C4(I)*PCT1)*GS(I)
GS(I) = GS(I) + GS(I+5)*NDAY
IF (GS(I) .LT. 1) GS(I) = 0
GS(5) = GS(5) + GS(I)
CONTINUE
RETURN
END
SUBROUTINE DSPRS

C DISPERSES THE ADULT MOTHs TO NEW GRID SQUARES AND UPDATES FOR NEW EGG MASSES

C COMMON /CM/ GOES HERE

DO 4 I = 1, IMAX
DO 4 J = 1, JMAX
DO 4 K = 5, 9
G(K,J,I) = 0
CONTINUE
I = IHEAD
J = JHEAD

IASUB = RAND(1.0)
CONTINUE
X = I - .5
Y = J - .5
DO 100 K = 1, 3
L = G(K,J,I)
IF (L .EQ. 0) GO TO 100
DO 95 H = 1, L
IASUB = FRAND(1.0)*108.0 + 1.0
III = I
JJJ = J
IF (IASUB .LT. 82) GO TO 10
INC = 1
JNC = -1
II = I
JJ = J - 1
N = 110 - IASUB
S = -STAB(N)
C = -CTAB(N)
GO TO 40
CONTINUE
IF (IASUB .LT. 56) GO TO 20
INC = -1
JNC = -1
II = I - 1
JJ = J - 1
N = IASUB - 54
S = -STAB(N)
C = -CTAB(N)
GO TO 40
CONTINUE
IF (IASUB .LE. 28) GO TO 30
INC = -1
JNC = 1
II = I - 1
JJ = J
N = 56 - IASUB
APPENDIX IV - CATERPILLAR MODEL CODE
SUBROUTINE DSPRS

54  S = STAB(N)
55  C = -CTAB(N)
56  GO TO 40
57  30 CONTINUE
58  INC = 1
59  JNC = 1
60  II = I
61  JJ = J
62  S = STAB(IASUB)
63  C = CTAB(IASUB)
64  40 CONTINUE
65  DT = FRAND(2.0)*DTAB(K)
66  D1 = (JJ-Y)/S
67  D2 = (II-X)/C
68  DOLD = 0
69  D = 0
70  50 CONTINUE
71  DIF = D1 - D2
72  IF (ABS(DIF) > 1.0E-3) GO TO 60
73  D = D + (D1-DOLD)*IABS(IG(11,JJJ,III))
74  II = II + INC
75  JJ = JJ + JNC
76  III = III + INC
77  JJJ = JJJ + JNC
78  DOLD = D1
79  D1 = (JJ-Y)/S
80  D2 = (II-X)/C
81  GO TO 80
82  60 CONTINUE
83  IF (DIF < 0) GO TO 70
84  D = D + (D2-DOLD)*IABS(IG(11,JJJ,III))
85  II = II + INC
86  III = III + INC
87  DOLD = D2
88  D2 = (II-X)/C
89  GO TO 80
90  70 CONTINUE
91  D = D + (D1-DOLD)*IABS(IG(11,JJJ,III))
92  JJ = JJ + JNC
93  JJJ = JJJ + JNC
94  DOLD = D1
95  D1 = (JJ-Y)/S
96  80 CONTINUE
97  IF (D > DT) GO TO 85
98  IF (II > IMAX) GO TO 95
99  IF (JJ > JMAX) GO TO 95
100  IF (II < 1) GO TO 95
101  IF (JJ < 1) GO TO 95
102  GO TO 50
103  85 CONTINUE
104  IF (IG(11,JJJ,III) = -1) GO TO 95
105  DO 90 N = 5, 9
106  G(N,JJJ,III) = G(N,JJJ,III) + ETAB(N-4,K)
107  90 CONTINUE
SUBROUTINE DSPRS

II = IG(14,J,I)
J = IG(15,J,I)
I = II
IF (I .GT. 0) GO TO 5

DO 500 I = 1, IMAX
DO 500 J = 1, JMAX
DO 500 K = 1, 4
G(K,J,I) = G(K+5,J,I)

500 CONTINUE
RETURN
END