NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

NUMERICAL INTEGRATION ROUTINES FOR THE GEST SIMULATION ENVIRONMENT

by

Michael Ian Barrett

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Master of Science
in

Systems Science
(Department of Computer Science)
University of Ottawa
Ottawa, Ontario, Canada.

ISBN 0-315-46714-2

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

University of Ottawa requires the signature of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

The advances of the applications of artificial intelligence in modelling and simulation are currently an important area of research in the field of simulation. A rationale is presented to explain why this paradigm shift from traditional simulation is inevitable. Additionally, current applications of artificial intelligence in simulation environments are discussed along with some promising directions for the future.

A particular advanced simulation environment based on the Gest language which demonstrates an application of artificial intelligence techniques in modelling and simulation is highlighted. The main correspondences between simulation programs written in the Gest language and their procedural equivalents are given. This, along with some test examples showing the structure of the specification program and its procedural equivalent provide good insight for the Gest translator to be built.

The run time library for the Gest environment is developed. It is that part of the environment responsible for actually driving the simulation at run time after the model building process and the necessary translation have been completed. It includes various integration algorithms for tackling some of the non-stiff initial value problems in continuous simulation. The features of a robust simulation run time library are discussed and suggestions as to how this can be ensured in the Gest environment are proposed.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my thesis supervisor, Professor Tuncer I. Oren, for suggesting the thesis topic, and his invaluable guidance in seeing through its completion. He was kind and patient, and spent considerable amount of his precious time in directing the research.

I would also like to express my sincere thanks to all my colleagues and friends for their help and support during the course of this research. Finally, I would like to thank the AUCC for their financial support, without which this work could not have been accomplished.

TABLE OF CONTENTS

DETAILED TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Simulation is being increasingly accepted as a powerful tool for decision support. Its growth in several independent fields continues to rise with the increasing availability of computers. As a direct consequence, in the last few years, many simulation systems have begun to provide aid to the user in the model building as well as other stages in the simulation process. This aid has become increasingly important since although the users may be experts in their respective fields, their knowledge of the simulation process may be such that software tools providing advanced assistance would be welcomed. As a result, it continues to be the focus of current research to improve the simulation process using artificial intelligence techniques to meet the demands of simulation today.

1.1 AIM AND STRUCTURE OF THESIS

This thesis consists of two parts. The first part presents some of the latest developments in the field of simulation, in particular simulation environments, and how they will shape the future of the simulation process as the synergy between artificial intelligence and simulation becomes more complete. The second part looks at an advanced environment based on the Gest language (Ören 1984b) which is an application of artificial intelligence in modelling and

simulation environments. This part constitutes the major purpose of the study which is two-fold. The first contribution consists of providing pertinent information to aid in the construction of the Gest translator for the environment. This involves giving detailed information on the structures of the programs written in the specification language Gest, their procedural equivalents (expressed in Fortran 77) and the correspondence between the two as well as test examples included in the appendices of the thesis which practically illustrate this. The second contribution is in providing run time software for the execution of continuous simulation models within the Gest environment at the run time stage, once the model translation has been completed. The software is used in driving the test examples for the translator. Finally, the quality assurance of the software is discussed. Part one of the thesis embodies chapters one and two while part two embodies chapters three to seven.

In the latter part of chapter one, we discuss the basic concepts of traditional simulation. In Chapter two, the paradigm shift from traditional simulation to artificial intelligence applications in simulation is discussed. Some of the current possibilities in the field are explained in detail. These include the use of artificial intelligence in modelling and simulation environments as well as quality assurance in modelling and simulation in the artificial intelligence era. The latter discusses new classes of

quality assurance concepts and techniques which are needed for advanced simulation environments. Finally, possibilities for future research in the field are enumerated.

Chapter three explains an advanced software environment based on the Gest language. Also, detailed explanations of Gest specification programs and their equivalent procedural programs for single and several component models are provided. Chapter four introduces the integration algorithms that will comprise the run-time library and discusses which algorithms are most advantageous for different types of problems of the non-stiff initial value type. In addition, the general purpose algorithms along with the multi-rate integration algorithm are presented in detail. Chapter five expounds the discontinuity problem along with the processing algorithm to deal with systems having such problems. In chapter six, we provide suggestions as to how quality assurance of the run - time library may be maintained within the Gest environment. Furthermore, the concept of guidance by the advanced simulation environment Gest in the selection of an appropriate integration algorithm for the model of interest is introduced. Chapter seven discusses the conclusions and the further studies that would enhance the existing research.

1.2 CONCEPTS OF TRADITIONAL SIMULATION

Simulation is a vastly used powerful decision support tool. The simulation technology has evolved over the years

from building models of systems in the form of diagrams, to more sophisticated physical prototype models to the construction of models in logical rather than physical form with the advent of the computer (Nielsen 1986). Simulation is important in decision support, control, and in aiding human knowledge processes by allowing theories to be tested practically (Birtwistle 1985). Computer simulation based on mathematical models has been used with great success for many years by engineers and scientists for analysis and design. Over the last couple of decades, the advancement and acceptance of simulation as a method of decision analysis is in sharp contrast to the early view by operations research practitioners who considered simulation as a last resort when all else failed (Wagner 1975). This advancement has been as a direct result of development in computer hardware and software making more powerful computers available at decreasing costs as well as recent advances in simulation systems and simulation methodology. This has further resulted in use and growth of quantitative simulation in other areas such as business models and management support activities. Traditionally, the major strength of a simulation model is in providing a prediction as to the consequences of a selected course of action under some experimental conditions usually referred to as a scenario (Reddy and Fox 1986). A typical application where a simulation model may be used by a decision support system is in the evaluation of two alternate courses of action to

improve the productivity of a factory. Simulation can be summed up as being experimentation with dynamic models (Ören 1986a) and the simulation model can be viewed simply as a special type of knowledge. The model is driven under some experimental conditions to generate model behaviour through time.

In the literature, two main types of simulation are discussed. These consist of discrete simulation and continuous simulation. In discrete simulation or simulation with discrete-change models, the changes which occur to the behaviour of the model are discrete and can be represented by step functions. Continuous simulation which is simulation with continuous-change models uses as its behaviour generation technique a numerical algorithm to solve a set of ordinary differential equations. However, it should be noted that a simulation study may involve a system which requires a variety of models. A systematization of the study of different types of simulation reveals a very large number of possibilities. A detailed taxonomy of types of simulation is given by Ören (1984c).

1.2.1 Elements Of A Simulation Study

In traditional simulation, as shown in Fig 1.1 the main elements of a study are the following : 1) Specification of the simulation study 2) Behaviour generation and 3) Model behaviour and its processing (Ören 1984c). Specification of

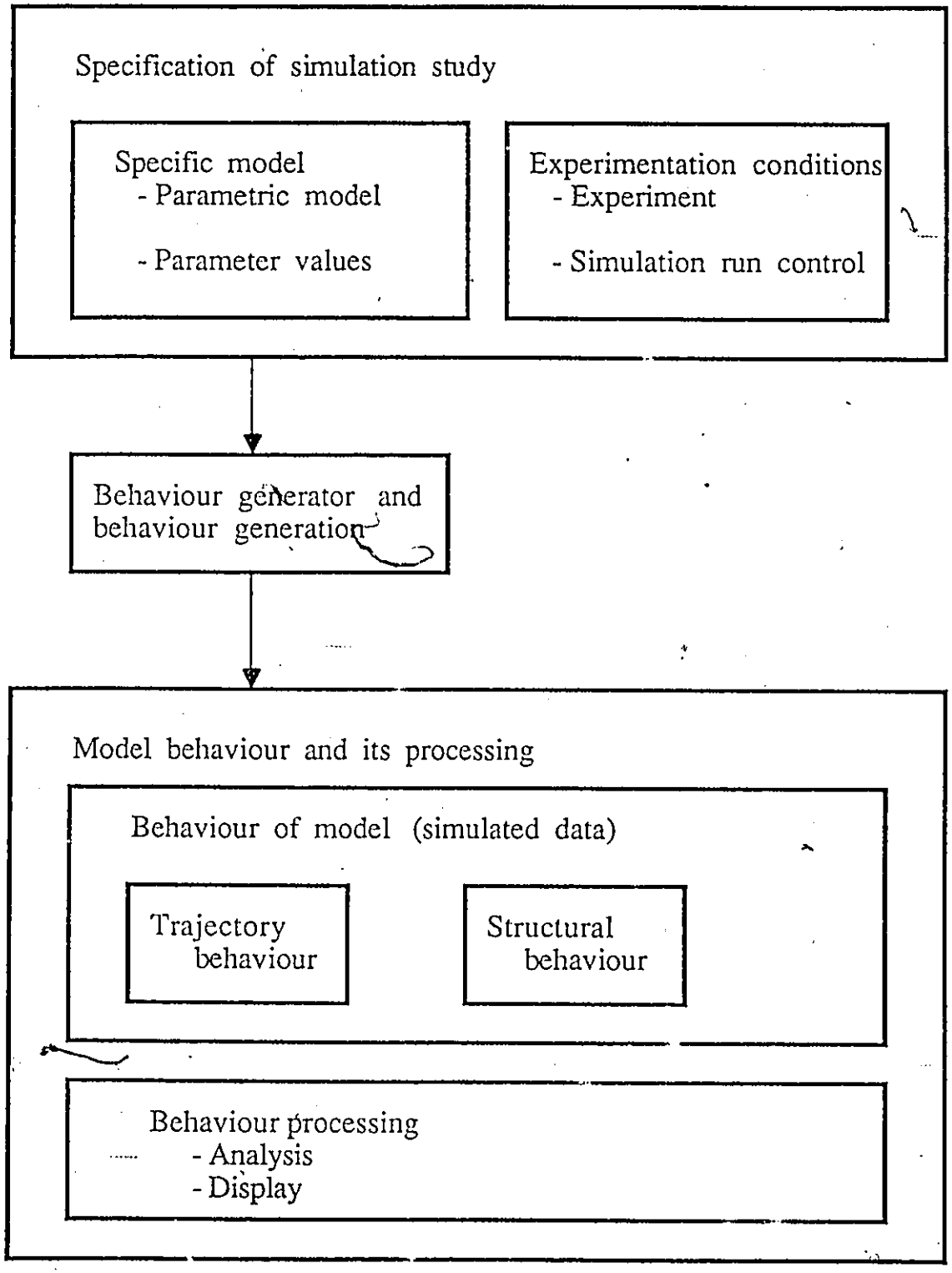


Figure 1.1 Fundamental Elements of a Simulation Study

the study requires knowledge of a parametric model and its associated parameter values as well as experimentation knowledge.

Behaviour generation is usually carried out in a computer by a behaviour generator that drives the model which comprises a parametric model and associated parameter set, under the specified experimental conditions to generate model behaviour. In simulation, model behaviour can be either trajectory behaviour or structural behaviour. In trajectory behaviour, a sequence of values of descriptive variables of interest, indexed with time, are generated. Then, using numerical or statistical techniques, the behaviour can be reduced to a figure of merit. In structural behaviour, the behaviour of a system is generated to depict a time indexed sequence of structural representations of the system. This is done under the control of a fixed or variable set of goals such as in the growth of plants in botany. After the simulation run, behaviour can be processed for analysis, compression and display purposes.

Traditional simulation is iterative. If necessary, the process is repeated for the parametric model, developing another scenario and the new experiment run. There are no design aids for developing the model. The modification of the experimental conditions and the model itself is controlled by the modeler and not incorporated into the system as an automatic computerized feature.

1.2.2 Traditional Simulation Languages

For some time now, traditional simulation languages have consisted of declarative as opposed to procedural types. Typical procedural simulation programs use a third generation language such as Fortran and traditionally have the control and data information components pertaining to the model combined in the model structure. These simulation languages are procedural since the program explicitly specifies the steps which must be performed to reach a solution.

Some continuous system simulation languages used in the former case such as ACSL, make the coding activities considerably easier. However, their execution point of view necessitates conceiving simulation programs in three sections: pre-run activities, run-time activities, and post-run activities as discussed in detail in chapter 3. This view was previously seen mainly due to the limited abilities of computers but has continued for an unnecessarily long time. As a result, traditional simulation still needs a great deal of expertise to translate the needs, goals, and objectives of the user into the appropriate model, using the right data, executed under the correct design of experiment and analyzed appropriately (Shannon et. al. 1985).

As a final remark, the principal characteristics of traditional simulation are emphasised as :

1) Knowledge processing of the system is limited only

to numerical processing.

2) Model is integrated with specification of experiment and behaviour generation.

CHARTER 2

ARTIFICIAL INTELLIGENCE IN SIMULATION

2.1 THE PARADIGM SHIFT FROM TRADITIONAL SIMULATION TO ARTIFICIAL INTELLIGENCE APPLICATIONS IN SIMULATION

2.1.1 Introduction To Artificial Intelligence And Expert Systems

Artificial intelligence deals with the discovery of techniques that will allow us to design and program machines which are able to emulate certain characteristics of human thought such as the ability to learn, solve problems and understand small contexts of human language. Artificial intelligence research which looks at how humans acquire, organize, and use knowledge can be categorized into :-

1) Attempting to duplicate or imitate the rational knowledge processing abilities in the same manner humans do. An example includes natural language query systems which allow the user to ask questions of a data base without using special commands.

2) Ability to use knowledge and learn so as to duplicate the results of learned skills or expertise without concern for whether it is exactly the process used by the human expert. This is often termed expert systems or knowledge-based systems. These are knowledgeable systems which may increase their knowledge by learning and have advanced knowledge processing characteristics including symbolic knowledge processing.

Expert systems research has grown out of more general concerns in artificial intelligence and is considered by some researchers to be "applied artificial intelligence." An expert system is a computing system capable of representing and reasoning about some knowledge-rich domain, such as internal medicine or geology, with a view to solving problems and giving advice (Jackson 1985). Feigenbaum, one of the pioneers in research and development of expert systems, has defined an expert system as:

" .. an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. Knowledge necessary to perform at such a level, plus the inference procedures used,can be thought of as a model of the expertise of the best practitioners of the field."

(Harmon,P., King,D. 1985)

2.1.2 Rationale For And Description Of The Paradigm Shift

Changes of paradigm always seem to take an unnecessarily long period of time primarily due to dogmatic thinking in most cases. However, there is a need to see simulation in a new light and it is inevitable that artificial intelligence will have a tremendous impact on simulation as we know it today (Ören 1984c). This need arises out of the fact that simulation is being utilized more and more in a variety of

fields and it is clear that traditional systems could benefit from further enhancements. The original requirements of such systems can be further enhanced via model processing abilities as well as experiment control and evaluation of simulation results. A variety of tools and techniques from the artificial intelligence field for working with knowledge models have only recently been applied to traditional simulation.

Artificial intelligence and simulation are based on models to generate new knowledge. In the case of a rule-based system, inference is used to generate knowledge. In the case of simulation, knowledge is generated by driving the model under experimental conditions. In particular, there is a methodological similarity between simulation models and expert systems (Shannon et. al. 1985, Lehmann 1987). These include :

1) Expert systems and simulation both represent in a modular form knowledge and expertise about a system and its behaviour in specific domains. The knowledge representation involves the use of events and processes in discrete event simulation and frames, rules, etc. in expert systems.

2) The control flow in expert systems and simulation both depend on logical decisions in contrast to numerical values calculated in other programs. The control or inference mechanism for an event driven

simulation can be described as a next-event time scheduling similar to inference mechanisms such as forward or backward chaining in expert systems.

3) There are similarities in the description and processing of uncertainty in a model as well as in an expert system. In stochastic simulation, this uncertain knowledge is expressed by probabilities and in an expert system application by means like certainty factors, fuzzy logic etc.

As a result techniques and results from artificial intelligence and particularly expert systems appear to offer a new and exciting technology for simulation application.

The paradigm shift in simulation is towards the use of artificial intelligence techniques in simulation. Knowledge-based simulation is the name adopted to describe this synergy. It deals specifically with knowledge-based techniques for model building and processing such as execution and analysis. This area of research in artificial intelligence concerned with building of tools and expert aids has had a significant effect on the modelling and simulation field. The main benefits achieved using an artificial intelligence based simulation tool include :

1) Aids in improving the process of transcribing the real world problem into the simulation thus shortening the development time of the model.

2) Structural changes can be carried out a lot quicker with faster synthesis and/or modification of simulation by the user.

3) Reduces the time required by users to work with the system since there is no need to learn a simulation language and develop modelling skills.

4) Improves the average user's quality of interpreting simulation runs and case studies.

Systems utilizing such tools would be responsible for the following developments to name a few :

1) A simulation query system which the user can use to determine the behaviour of some model or models in a model base, maybe with respect to a particular problem.

2) The creation of models for a specific problem in a well understood domain via a modelling environment.

3) Use a behaviour processing system to provide for analysis and display.

2.2. CURRENT APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN SIMULATION ENVIRONMENTS

In the past few years due to the increasing abilities of computers on symbolic processing and the general maturing of the simulation field, the focus of simulation languages and

packages has changed and is being replaced by a more comprehensive concept, namely simulation environments (Ören 1986b) . The creation of such environments is one of the possibilities for artificial intelligence to have an impact on the simulation field. These environments should be created to take advantage of current computer capabilities for data base management and graphics to name a few. The simulation environment is the term used to denote a comprehensive computer-aided design and simulation system. Furthermore, Henricksen (1983) defines an integrated simulation environment as "a collection of software tools for designing, writing, and validating models; writing and verifying simulation programs; preparing model input data; analyzing model output data; and designing and carrying out experiments with models." This is essentially the goal of artificial intelligence based expert simulation systems, in addition to embedding as much expertise into the simulation software as possible.

There is a clear need for simulation environments which provide an integrated and comprehensive set of tools for all areas of simulation (Langer and Schmidt 1987). Within a simulation system, three main kinds of environments are desirous: 1) Simulation modelling environment 2) Model processing environment 3) Environment for analysis of simulation results. These environments are discussed in detail in the following sections.

2.2.1 Simulation Modelling Environments

The simulation modelling environment is concerned with the model building stage of the simulation process. The environment assists the user in the model building process by providing a variety of software tools. Advanced user/system interfaces are commonly provided to simplify the user's task of specifying the model. Additionally, during the model creation, an advisory system guides the user with explanation thus ensuring model completeness and certification.

A major concern of current software developers is the interface between the user and the computer. Artificial intelligence researchers are among those developing user friendly interfaces to improve this communication. Advanced user/system interfaces enable the user to focus on the problem definition and analysis of results. It helps to form an advanced user/system interface for a simulation model for use by an inexperienced simulationist. The simulationist is able to develop and modify the model without requiring previous knowledge about the simulation language or programming environment Such interfaces are significant contributions to the field providing better user control of his/her model and thus increased efficiency in the modelling process (Khoshnevis and Austin 1986).

As proposed by Ören (1986b), in an interactive environment, the user would have a display unit on which

several concurrent, overlapping or non-overlapping windows may be seen. The windows may consist of: 1) Design and simulation windows to display the design and trajectories of descriptive variables. 2) Active icons that can be selected for model design. 3) Dialog diary to display system messages and user invoked tasks and operations. 4) Instruction window to specify to the system a particular operation which has to be operated on a given object related with a task. Recent simulation modelling environments provide similar advanced user/system interfaces.

The knowledge base simulation system developed by Reddy and Fox (1986) uses a window-based schema editor, which offers a fairly general facility to create or alter schemata. The model is a collection of schema representation language schemata that represents physical and abstract system entities. Model creation is simply the creation of schemata that represent the model's entities, including specifications for their event behaviour and interconnections with other schemata. One window displays the schema being edited, while the other window accepts commands to manipulate that schema.

In the intelligent simulation model generator (Guariso et. al. 1987), the interface is incorporated into the LISP environment and input and output are mainly performed via menus, forms representing the frames and graphics. The slots of the frames provided are filled by the user, the editing

being performed by extensive use of windows. An example is the use of the graphic editing window in creating compound models or models with several component models.

The Simplex II Simulation System (Langer and Schmidt 1987) is easy to use due to its menu-driven user guidance. For each complex system feature a menu is provided, which includes the necessary commands to perform all possible functions of this feature. These commands manipulate individual model components (create, copy, modify etc.) and also manage the various model banks. In the Gest environment which is explained in detail, in chapter 3, the system aids in the generation of the Gest source program using a model template generator which generates appropriate templates or frames whose slots are to be filled by the user.

Finally, it should be realized that it is technologically feasible to build natural language interfaces for domain specific applications. Additionally, a natural language representation of continuous systems is feasible since they have only a few types of variations in the transition logic (Khoshnevis and Austin 1986). It should be noted, however that while natural language interfaces are feasible solutions to human interface problems one must assess and determine when such an interface is feasible. Rich (1984) outlines seven key factors that should be considered in such a choice.

A truly successful understanding system incorporates knowledge about language, grammar, semantics and the specific domain of the problem. A natural language interface for model building is appropriate because of the variety and complexity of the inputs for problems formulation.

An advisory system is an expert system that uses a knowledge base and inference system to provide the user with relevant expert advice on fault finding as the consultation proceeds. It is a decision support tool that guides the user in correcting the fault. Simulation knowledge must be separated from domain knowledge. As would be expected, then, two different types of advisory systems are perceived in simulation, namely those with expertise in simulation and those with expertise in an application domain. Two main types of advisory systems are proposed. These consist of: 1) expert systems knowledgeable in the simulation methodology and 2) expert systems knowledgeable in the domain knowledge.

Another important feature of the advisory system is that they should possess explanation facilities. These facilities generate explanations so as to try to address or substantiate the accountability of guidance given by advisory systems. They do this by systematically relating to the chains of reasoning employed by rule-based systems. This is done by backtracking through the processes of inference used in generating the advice and presenting the

user with the knowledge used at each stage. One of the main compelling reasons for the requirements that advisory systems should be able to explain their reasoning and justify their conclusions in a manner that is intelligible to users is (Jackson 1985): 1) The users of automatic advisory systems need to be convinced that the reasoning behind a conclusion is substantially correct, and that the solution proposed is appropriate to their particular case. Finally, the advisory system has certification functions which are responsible for certifying the model.

The knowledge-based simulation system (Reddy and Fox 1986) contains a language and an interpreter so as to specify model consistency and completeness rules. Additionally, the logic programming techniques are incorporated in the Schema Representation Language. In this way, the knowledge-based simulation system evaluates the model using the interpreter's trace facility to aid in finding and explaining errors such as missing schemata and schemata with inconsistent slot values. The advisory system in the intelligent model generator (Guariso et. al. 1987) performs consistency checks and informs the user, forcing him to correct the errors found or to quit his current action.

The Simplex II Simulation System performs only syntactic checks on models to ensure they are syntactically correct. The advisor for modelling present in the Gest environment is

called Magest . Magest uses its available knowledge to assist the user in the model specification and performs several syntactic as well as semantic checks to ensure the model's certification. These checks consist primarily of checks for completeness, correctness, and compatibility. The available knowledge consists of: 1) problem independent knowledge concerning the morphology of Gest programs 2) incremental knowledge obtained from and about user specified Gest programs. The details of the rules and facts used by Magest to perform semantic checks can be seen in the recent article by Ören and Sheng (1988).

The use of the advisory systems and advanced user/system interfaces outlined above makes the modelling simulation environment within the simulation system a useful tool in aiding the user in the model building process.

2.2.2 Experimentation Specification And Model Processing Environments

The experimentation specification and model processing environments may be responsible for a variety of functions such as specification of experiments, simulation execution, and data collection. The experimentation environment would consist primarily of an experimentation advisor, an experimentation base, and its manager (Ören 1986b). The advisor may have several functions including:

1) Specification of experimentation including

specification of variables to be observed and monitored. The specification of the variables to be observed includes behaviour compression types (e.g. statistical), behaviour display period (beginning, end) and frequency (communication interval). An example of the specification of variables to be monitored involves having audio/visual warnings when a critical variable reaches a threshold value.

2) In the case of multiple runs, experimental conditions can be modified based on specification of generic experimental conditions. In the simplest case, the system can directly link to a model different experimental conditions specified by the user. It may have the ability to conduct a series of experiments without human intervention. Individual experiments should be improvements on past experiments.The improvements are automatically achieved by rule bases detailing the diagnosis and correction heuristics (Reddy and Fox 1986). The rules must allow for reasoning between several model scenarios. The Simplex II Simulation System (Langer and Schmidt 1987) contains an experimentation environment. This environment provides a menu experiment which allows various experiments to be performed on the model.

Symbolic model processing is possible during the

execution of the simulation. At the end of each simulation run, a rule can be activated from a rule base that will suggest changes to the model for the next run. After running many of these goal-directed experiments, we hope to get scenarios that come close to satisfying the simulation goal (Reddy and Fox 1986). A limit on the number of experiments prevents the system from getting lost while determining a desirable scenario.

Data collection is concerned with recording the changes in the value of a parameter. This is usually done either by constantly monitoring the parameter or by sampling the parameter (Reddy 1987) . The parameter to be monitored usually has attached to it instruments for measuring current values to be displayed. In the knowledge-based simulation system, monitoring is done by attaching demons to slots and the process of data collection is similar to measuring instruments. Usually, only data relevant to a particular goal is gathered and analyzed and the user is required to manually instrument each schema. The menu item called Monitor in the the Simplex II Simulation System is available for the user to select which model variables are to be recorded during the experiment for subsequent analysis and display, the menu present being subsequently used to graphically output the results.

2.2.3 Environment For Analysis of Simulation Results

This environment is primarily concerned with analysis of the simulation results. This is a recent feature in simulation systems. Within the knowledge-based simulation system, scenarios can be rated to measure the goodness or badness of simulation results. This is important so that the analyst can fine tune the input parameters to the model in order to bring the values of output variables within a desired range. The Simplex II Simulation System has available a menu item called Analyze which contains methods for the mathematical analysis of data which has been recorded by the monitor.

It is clear that current applications of artificial intelligence in simulation environments have made a positive impact on the aid given to the user throughout the simulation process. Current systems tend to place more emphasis on the modelling simulation environment with less emphasis on the other environments described above especially the environment for analysis of results. This trend is changing and future systems can be expected to have a variety of tools available in the three main environments.

2.3 DIRECTIONS FOR FUTURE RESEARCH

The latest developments in the area of simulation

strongly indicate that AI based simulation systems will be the trend in the near future. Many areas such as manufacturing are adopting this trend and the first steps and attempts in this direction have been taken (Shannon et. al. 1985).

Although a plethora of ideas could be suggested, Oren (1987b) Shannon et. al. (1985) and Elzas (1986b) have proposed that major developments in artificial intelligence based simulation environments can be expected to be seen in:

1) Improved dialogue facilities with modelling and simulation systems especially those possessing model bases via advanced intelligent interfaces. These interfaces can be improved using AI techniques such as the use of natural language programming for problem specification.

2) Future model bases which should provide access to models and submodels developed earlier in similar applications. They could be stored using a specification formalism in a number of domain specific model bases (Kettenis 1986). The model base would consist of a number of clearly written, well documented models.

3) Expert systems for modelling and simulation consultation. These systems would be very useful in providing advice as to which model in a model base is best suited for a particular application.

4) Model search and inferencing from assemblies of model components in model bases for a particular domain.

5) Intelligent simulation model generators to perform automated simulation program generation from structured model and experimental frame specification.

6) Advanced methodological concepts involving the possibility of generating experimental condition and interpretation of results within simulation environments.

7) Advanced tools to cope with the complexity of systems. This would involve design of other knowledge-based tools such as KEE (Fikes and Kehler 1985).

8) Expanding the knowledge available in current knowledge based environments to include domain specific knowledge on different application areas.

9) New knowledge representation structures in artificial intelligence environments for modelling and simulation support systems.

10) Input data management systems capable of storing, manipulating and retrieving data for specific

simulation experiments. It would require the ability to distinguish which data is needed for the different experiments.

11) An experiment supervisor which would make decisions based on the output of simulation runs e.g. sample sizes, number of runs needed etc.

In the last couple of years, the field of artificial intelligence in simulation has grown tremendously and continues to mature. Several books already exist on the topic: Birtwistle (1985) Elzas, Ören and Zeigler (1986), Holmes (1985), Kerchoffs, Vansteenkiste, and Zeigler (1986), Luker and Adelsberger (1986), Luker and Birtwistle (1987), Widman, Loparo, and Helman (1987) whilst others are now in preparation such as Elzas, Oren, and Zeigler (1988 ± In Preparation). It is quite clear that some of the developments discussed above will become a reality in the near future thus realizing yet more advanced simulation environments than those available today.

CHAPTER 3

GEST AND ITS ENVIRONMENT

3.1 SOFTWARE ENVIRONMENT FOR GEST

3.1.1 The Gest Language

Gest, which stands for General System Theory Implementor, is based on the axiomatic system theory of Wymore (1967) and departs radically from other simulation languages in that the specifications of the model and the experiment are totally separated. It is the first model and simulation specification language. As a result, it is declarative in nature acting as documentation easily comprehensible by users.

In the Gest language, a program has several distinct parts (Oren 1984b):

1) Mathematical Model
2) Parameter Set
3) Experimentation(s) and
4) Output Module(s)

The model consists of a parametric model and associated set(s) of parameter values. The experimentation is the specification of experimental conditions which have to be applied to the model. The output module is the specification of the output program to be used to display the results of the simulation study. The definition of a Gest program expressed in Backus-Naur form (BNF) is as

follows:

```
Program =
    "Program" Program Identifier
     Parametric Model Specification
    [{Parameter Set}]
     {Experiment}
     {Run Specification}
     {Output Module Specification}
     "End Program" Program Identifier
```

In the BNF representation, non-terminals are enclosed within " ", { } indicate repetition, [ ] indicate options, [{ }] indicate repetition of optional elements.

3.1.2 Elements of Gest Environment

The world view of Gest makes it a good basis for a comprehensive modelling and simulation environment. This environment is a major project currently being implemented at the University of Ottawa. It will provide assistance to users in the development of models in several application areas. As depicted in Figure. 3.1, the elements of the Gest environment consist of the following main components:

1) Gest executive
2) Magest system
3) Gest Translator

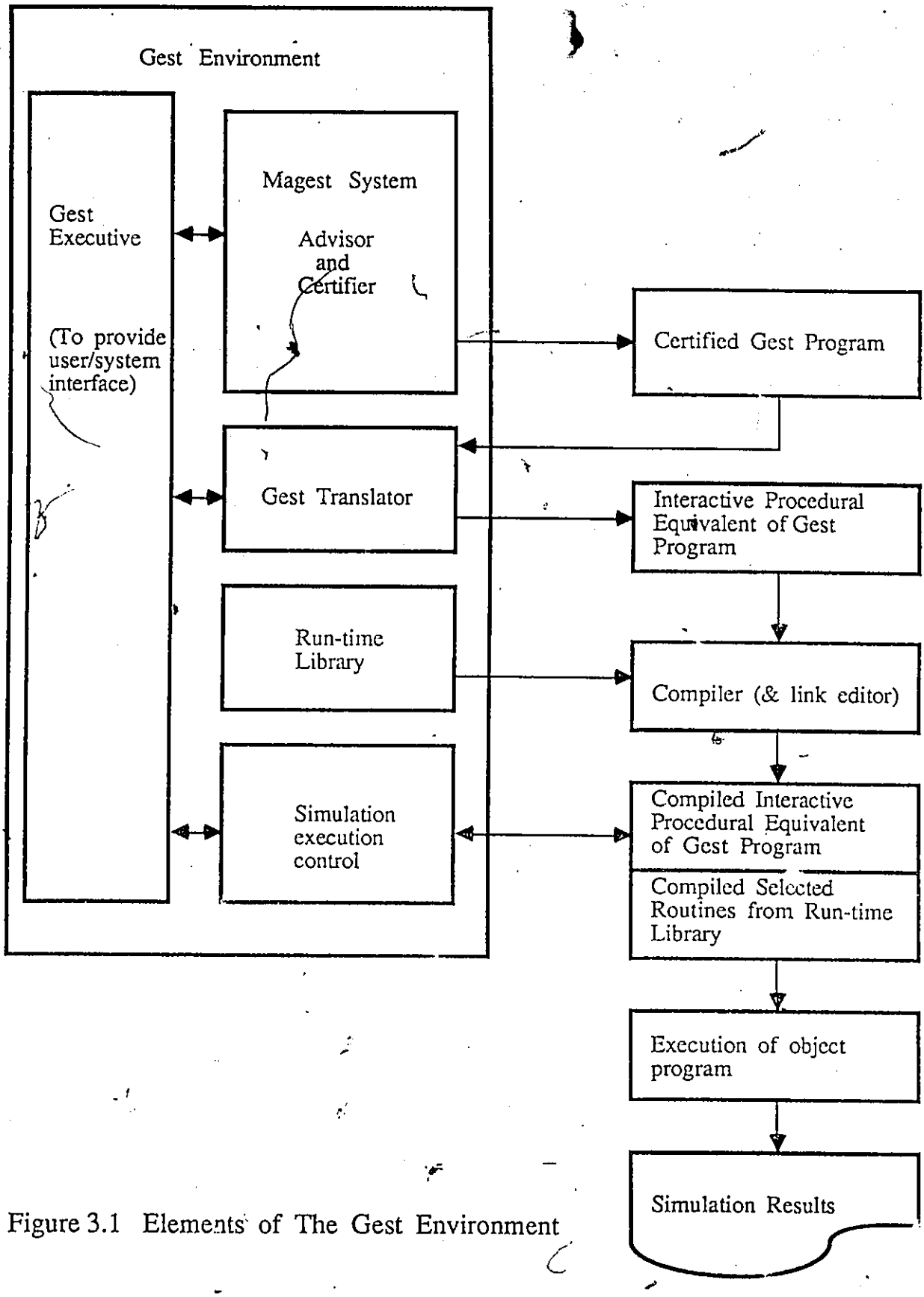


Figure 3.1 Elements of The Gest Environment

4) Gest Run-Time Library

5) Simulation execution control

The Gest executive is the user/system interface that assists the user at each stage of the simulation process. The executive is linked to the Magest system which provides assistance in the model building process. Additionally, it is connected to the simulation execution control, so that the user may be interactively involved in the execution of the simulation runs. Magest which stands for modelling advisor and certifier for Gest programs is a knowledge-based modelling and simulation system. It provides a user/oriented specification and certification environment for Gest. It has basically two modes of operation, the guidance mode and the certification mode. It also has access to a template file from which it can select templates and then tailor them dynamically.

In its template generating mode, Magest provides a dynamic template of the Gest source program or frame-like structure whose slots are to be filled by the user. As an example, for every component model, a template of static and dynamic structures are generated. The user then fills in the relevant model information in the appropriate slots. This is illustrated in Section 3.2.1. In this way, Magest guides the user in the specification of a model, parameter set, and experimentation.

During the generation of the Gest source program in the above manner, the certifier and advisor program uses its available knowledge to assist the user in the specification of his model and also performs several syntactic as well as semantic checks. These checks consist primarily of checks for completeness, correctness and compatibility of the specification.

The Magest system provides computer assistance by ensuring consistency and completeness of the specifications. The assistance provided by the system to the user while specifying his Gest program, also involves knowledge-based help on parameters to be specified as well as details on the experimentation specification pertaining to each component model such as initialization of all state variables.

Finally, once a Gest program passes these checks made by Magest, Magest then certifies the program making it ready for translation by the Gest Translator into an interactive procedural program.

The Translator will translate the declarative Gest program into an interactive procedural program. This program is then compiled by the procedural language compiler to allow for future execution.

The Translator recognizes three distinct levels of activities namely:

1) Pre-Run Activities

2) Run-Time Activities

3) Post-Run Activities

The pre-run activities involve the declaration of variables, initialization of state variables, parameters, input values, and run-control variables.

The run-time activities are concerned with experimentation with the model. These activities involve the determination of model inputs, outputs and the calls to the relevant integration algorithm in the Gest run-time library for integration. In the case of a model with several component models not utilizing the multi-rate integration algorithm, the translator lumps the differential equations of each component model into a derivative subroutine consisting of a single collection of differential equations. This derivative subroutine is used by the integration algorithm to integrate the model. In the case where the multi-rate algorithm is used, the large model which has widely varying characteristics is partitioned into two component systems having similar characteristics. They additionally have independent error control and utilize the best suited algorithm for each component model. The Gest translator in this case translates the specification ensuring two derivative subroutines containing the differential equations describing the two component models exists.

The post-run activities involve display of model results in tabular as well as graphical forms.

For all the activities described above, the necessary information required by the translator to determine these activities is taken from the pertinent modules of the Gest specification. This correspondence is explained in detail in Section 3.2.1.3.

The Gest run-time library consists of a variety of integration algorithms to deal with the different situations that arise in continuous simulation. The user expresses his choice of integration algorithm dependent on his model's needs in the experimentation section of his Gest Program. The Magest system ensures that the chosen algorithm is indeed available in the run-time library and may give advice to the user on the suitability of the chosen algorithm to the model at hand.

Then, during the execution phase, after the relevant inputs and outputs have been determined, the next states are computed by utilizing the relevant integration algorithm within the Gest run-time library via a call statement along with the subroutine DERIV containing the necessary functions and information to carry out the computational steps. The details of the proposed Gest run-time library are given in Chapters 4, 5, and 6.

Finally, the simulation execution control allows the user

to be interactively involved with the procedural equivalent of the Gest program. This may include the different specifications of the simulation runs to be executed for the different scenarios desired by the user.

3.1.3. The Disspla Software

Within the IBM environment, the Disspla software is the graphics package utilized by the Gest environment in providing post-run graphics of the simulation results at the end of the simulation. This 'high-level' plotting language which requires a minimum number of CALLS to produce graphs is used to facilitate data plotting in the Gest environment. It is machine independent as well as graphics device independent, enabling the user to take full advantage of the characteristics of a variety of output devices.

Once the translated procedural simulation program is run, the program is linked to Disspla to give post-run plots of the results of the simulation model. The graphics subroutine in the procedural program which utilizes the library Fortran subroutines available in Disspla, first determines and rounds the maximum and minimum values of the elements of the arrays of the variables of interest. These variables of interest are the outputs of the simulation for which the trajectory plot is required. Once these values are determined, scale factors are automatically generated in the subroutine by simple calculations. These factors along

with the maximum and minimum values are then used as arguments for the Disspla subroutines to generate the desired trajectory plots. For each model, the only significant changes to the graphics subroutine would involve labelling of axes, and title names. Full details of the Disspla software can be seen in the Disspla user's guide (ISSC 1982). In each of the test examples for the translator provided in this thesis, a graphics subroutine utilizes Disspla routines for carrying out the desired data plots.

3.1.4. The Xmenu/e Software

The XMENU/E software is used to provide interactive ability in the procedural equivalent of the Gest program. The XMENU/E system is a collection of programs that allow interactive creation and manipulation of display data and menus. XMENU/E allows the user to create and use full screen menus quickly and easily under his application program. Menus are useful tools for presenting users with a formatted list of options and input areas. XMENU/E provides the following basic facilities : 1) The XMENU program which allows the user to interactively create and modify his/her menus. 2) XMENUSUB, a collection of subroutines which can be called to use menus from programs written in high level procedural languages such as Fortran and Pascal.

As a result, this software can be utilized as a tool for interactive procedural programs. For a particular model, the

menu consisting of model parameters, initial values, experimentation variables, and relevant user information is created. Menu creation is performed on a blank input screen using PF keys to assign fields. The menu can be viewed as comprising of a number of fields with particular characteristics called 'attributes'. The fixed information on the menu have the following attributes : 1) PROT - The user is not allowed to enter data into that field from the terminal. 2) SKIP - This attribute facilitates easy access to the slots that need to be filled by the user. 3) DIM - The intensity of the field is given a normal display. 4) NMDT - This represents a no modification tag since these fields will not be modified at any time.

The slots which need to be filled by the user have the following set of attributes : 1) NUMERIC - This is an unprotected field that allows the user to enter only numeric data from the terminal. This is an obvious advantage for our purposes of simulation ensuring model integrity. It should be noted, however, that this attribute is only operational from some terminals such as the IBM 3179G terminals in the current version of the software. 2) NOSKIP - This attribute ensures that the user enters the appropriate input data. 3) BRIGHT - This attribute highlights the input, giving a bright intensity. 4) MDT - This Modified Data Tag ensures that data in each input field will be returned to the program as user input. At each simulation run, the current values are used in the simulation.

The XMENU/E software is first linked to the user's disk. The interactive procedural program is run, and with the aid of the appropriate subroutine from the XMENUSUB, the appropriate menu is presented to the user. The user is requested to fill in the slots according to the model, after which, he returns control to the program by selecting an appropriate PF key (in the current implementation PF 10 or PF 12) as specified on the menu. With the aid of subroutines from the XMENUSUB, the program is able to use the values from the menu to run the simulation. Another subroutine prints the values used for that particular simulation run into a file which is available to the user. This process is repeated for each simulation run which the user desires to carry out. The XMENU/E procedures are utilized in most of the test examples in this thesis to provide the desired interactive abilities of the procedural equivalents of the Gest programs.

3.2 STRUCTURE OF GEST AND EQUIVALENT PROCEDURAL PROGRAMS

In this section, detailed structures of the two main types of Gest programs for continuous simulation and their equivalent in a procedural language are provided. These involve specifications of continuous change models which are a subset of the possibilities available in the Gest language. In particular, the two main types are a Gest program with a single continuous component model and the

Gest program with several component models. For Gest programs with a single continuous component model, the ability to specify an updateable continuous change model is discussed. In the case of the Gest program with several component models, the ability to specify the Gest program using the multi-rate integration is also discussed. The structures show how programs are developed to facilitate the simulation of a user's model, whether it is a purely continuous model, a piece-wise continuous model, or a large dynamic system. Also, the correspondence between the Gest programs and their procedural counterparts are provided along with the computational steps needed in the simulation of the different types of models discussed. This information along with the Gest examples provided in the appendices of this thesis will aid in the construction of the Gest translator for the environment.

3.2.1 Gest Program : One Component Model

3.2.1.1 Gest Program

As shown in Figures 3.2(a) and (b), the Gest specification of a single component model consists of :

1) A parametric continuous component model

2) At least one parameter set

```
Program (Name)
  Continuous Model (Name)
    Static Structure
      Inputs (including Random if any)
      States
      Outputs
      Auxiliary Variables
      Constants
      Parameters
      Tabular Functions
    End Static Structure
    Dynamic Structure
      Derivatives            : State equations for computing
                               values of state variables
      Output Function        : Function for computing output
        (optional)
      Update (optional)      : Model update made when a particular
                               condition is satisfied
    End Dynamic Structure
  End Continuous Model (Name)
```

Figure 3.2(a) Gest Specification of a Parametric Component Model

```
Parameter Set
    Assignment of Parameter Values
    Assignment of Tabular Function values
End Parameter Set

Experiment
    External Input Function
        |
    End External Input Function
    Time Initialization
    Initialization of states
    Integration utilizing some particular algorithm
    Interpolation used (if any)
    Assign user tolerance
    Assign frequency of communication
    Collect and save inputs and outputs
    Specify end time of simulation
End Experiment

Run
    Specify the triplet        : (Model,Parameter Set, Experiment)
    Post-Run Requirements (optional)
End Run

Output Module
    Print output heading
    List values of saved variables
    Plot values of saved variables
End Output Module

End Program (Name)
```

Figure 3.2(b) Gest Specification of Parameter Set, Experimentation and Output Module for a Component Model.

3) At least one experimentation specification
which consists of :
- an experiment
- a run
- an optional post-run specification

4) An Output Module

The continuous component model consists of two sections: 1) static structure section and the dynamic structure section. The static structure is descriptive, and contains all the model descriptive elements such as inputs, states etc. as shown along with their types and ranges of values where applicable. The dynamic structure contains information needed for behaviour generation. This includes 1) a derivative block which contains the derivatives of state variables along with any computations necessary for auxiliary variables. 2) An optional output block containing output function(s). In cases where an output variable is not identical to a state variable, or an auxiliary variable, it must be computed from an output function. 3) An optional update block. This block provides a model-oriented specification of any discontinuities that may exist in the model. This includes update of state variables or parameters and the conditions which trigger them.

The parameter set contains the values of all model parameters and tabular functions if any. The experiment block consists of the values of all experimentation

variables, states and inputs. In addition, the type of algorithm to be used in the simulation is specified by the user in this block along with the Interpolation used if any. The Run specification specifies the triplet (model, parameter set, and experiment) to be used in the simulation. This essentially defines the scenario used with the model. In a typical simulation study, the modeler may need to test his model under a number of different scenarios. Thus, a number of parameter sets and experiment blocks may exist for any particular model, and the model may be specified to be run with different combinations of parameter sets and experiments. Finally, the Output Module specifies the format of display of the saved variables of interest along with an appropriate heading.

3.2.1.2. Interactive Procedural Program

The interactive procedural program equivalent to a Gest program (for the single component model) has three main sections, the details of which are shown in Figure. 3.3. The first section deals with pre-run activities where all model variables and experimentation variables are initialized. These values are initialized from the information available in the Gest parameter set and experimentation module. If more than one simulation run is required, the specification

```
Program (Name)                    (For Single Continuous Component Model)
Pre-Run Activities                                        (Initial Segment)
    Declarations
    Initialization                                          (using XMENU)
        Parameters
        Tabular Functions
        State Variables
        External Input Variables
        Run-Control Variables
              Time : - Initial Time:            -TINIT
                     - End Time    :            -TFIN
                     - Computational Time:      -T
                     - Communication Time:      -TCOM
                     - Communication Interval: -COMINT
              Integration : - Tolerance

              Dummy Indices  : - Array Indices
                               - Counters (etc.)
    End Initialization
    Pre-Run Output
        Print Heading
        Print Values of Parameters
        Print Values of Initial States
    End Pre-Run Output
End Pre-Run Activities
Run-Time Activities                                       (Dynamic Segment)
    While  T <= TFIN
        While T <= TCOM
            If there is a time for a model update
                Then | Perform model update
            End If
            Compute external input(s)
            Compute model output(s)
            Compute next state(s)
            If a state value for a model update is detected
                Then | Locate the discontinuity point
                     | Perform model update
                     | Recompute next state(s)
            End If
            Store next state(s)
            Terminate simulation if condition given as
              a function of state variable is satisfied
            Time update : T = T + h
        End While
        Collect and save model output(s) at communication time
        Update TCOM: TCOM = TCOM + COMINT
    End While
    Collect and save model output(s) at TFIN
End Run-Time Activities
Post-Run Activities
    Post-Run Computations (if any)
    Display Output(s)
          -In Tabular Form
          -In Graphical Form
End Post-Run Activities
End Program (Name)
```

Figure 3.3 Interactive Procedural Program For A Component Model

is interactively carried out using the XMENU/E package. The menu is simply invoked and the necessary modifications made to the variables. After the reinitialization is performed, the simulation is run once more. This process is repeated for all the simulation runs. Also, as part of the pre-run activities, any pre-run computations or assignments of inputs and/or outputs are carried out. The second section contains all the activities that must be carried out during the run-time of the simulation. The run - time computations are expounded in Section 3.2.1.3. The last section involves the post-run activities that need to carried out. These include any computations to be carried out after the run as well as how the output is to be displayed after the simulation is finished. The output is listed in tabular form and with the aid of the DISSPLA software displayed in graphical form.

3.2.1.3 Correspondence Between Gest and Procedural Program

For each set of activities, i.e. the pre-run, run-time and post-run activities the corresponding parts of the Gest program are shown. The correspondence for the pre-run activities are shown in Figure. 3.4. In a nutshell, information included in the static structure, parameter set, and experimentation section form the components of the pre-run activities. It is from these sections of the Gest program that the initial values of the model parameters,

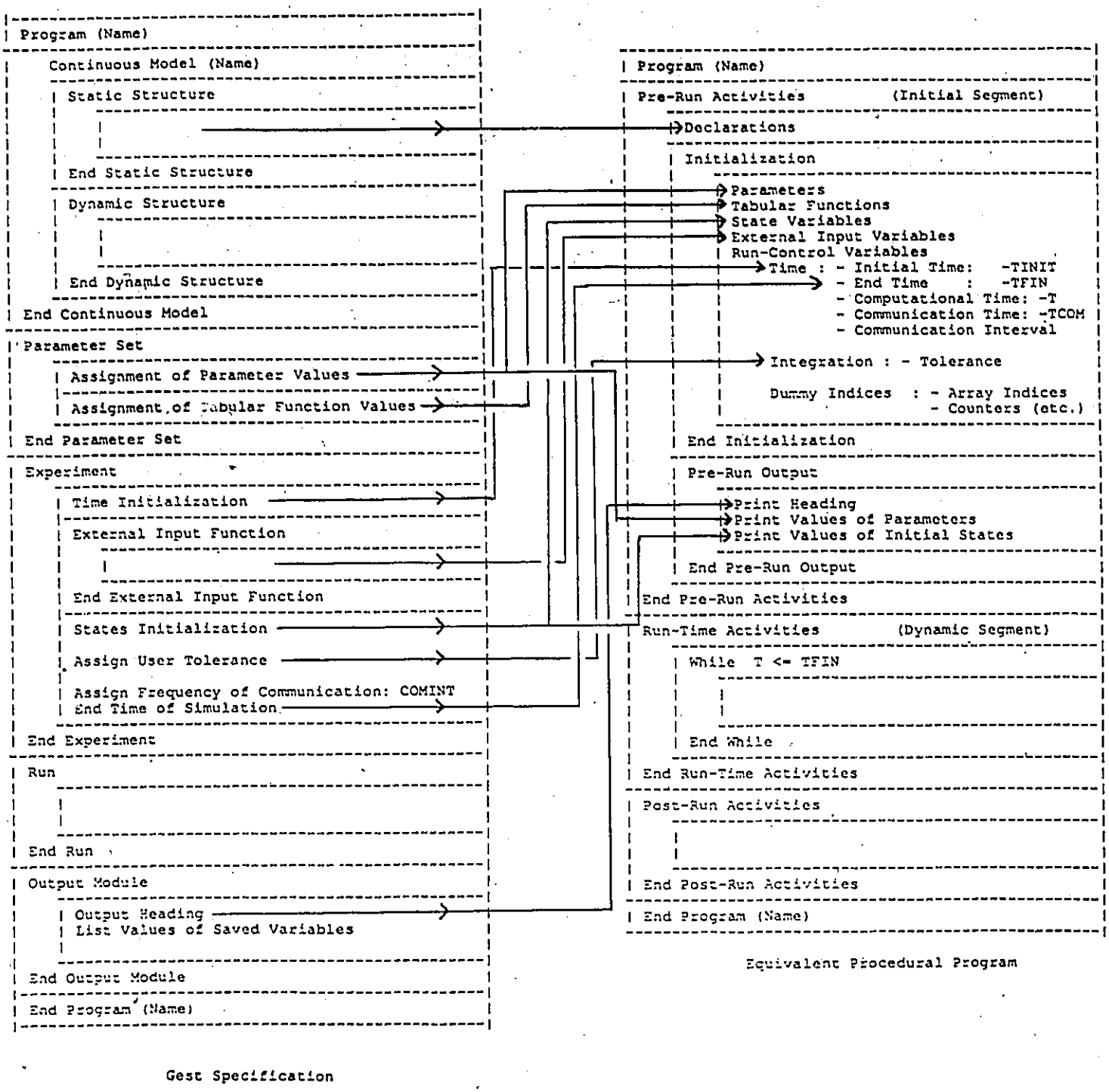


Figure 3.4 Correspondence Between Gest Specification and Equivalent Procedural Program For Pre-Run Activities

initial state values, external input variables, tabular functions, and run - control variables such as time and integration variables are determined. Preliminary information such as the heading in the output module of the Gest specification, additionally, provides the pre-run output in the procedural equivalent. Once the pre-run activities are completed, the run-time activities can proceed.

Figure 3.5 shows the correspondence for the run-time activities. The dynamic structure along with the triplet specification in the Run section provide the information required to perform the run-time activities of the procedural equivalent. Once the triplet (Model, Parameter Set, and Experiment) has been specified, the simulation run along with its activities can proceed. The run-time activities start with the computational time equal to the initial simulation time, and the integration algorithm is utilized to integrate over a communication interval or until a termination condition is met.

For each communication interval, a check is done for a possible discontinuity at a specific time. If one exists model update is performed. Any external inputs to the model are computed from the external input function along with an interpolation method if necessary. Model outputs are computed from the appropriate output function. The next states can be computed using the integration algorithm

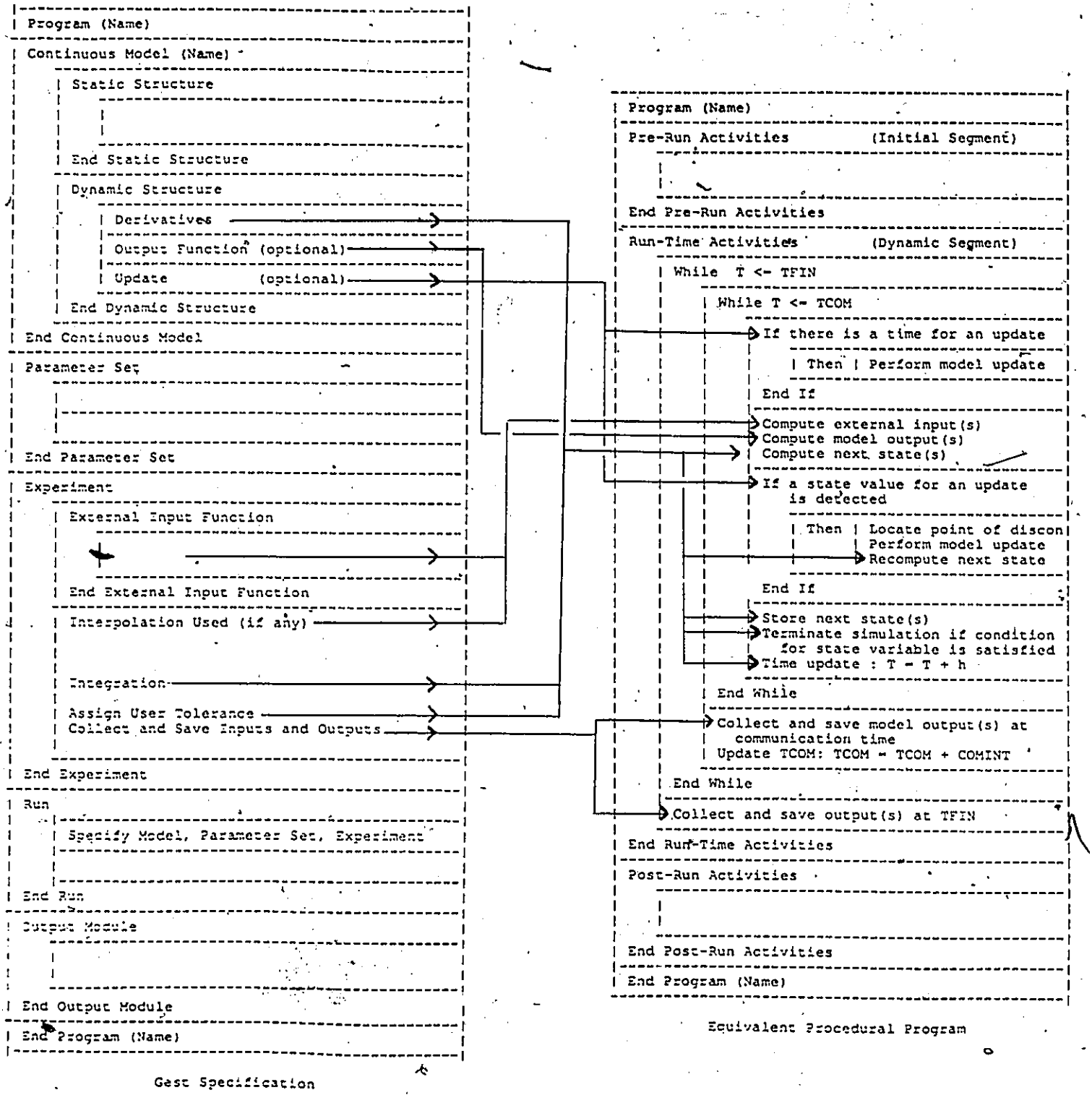
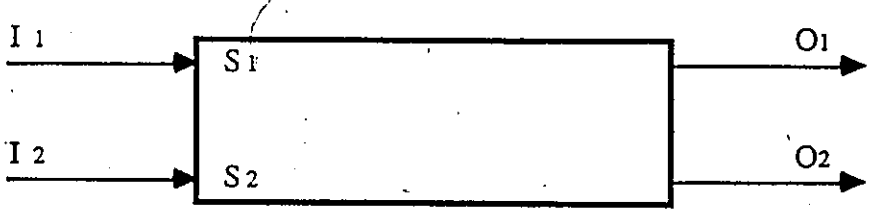


Figure 3.5 Correspondence Between Gest Specification and Equivalent Procedural Program For Run-Time Activities

chosen with the specified user tolerance, along with the initial state variable values and the derivatives that describe the model. At this point, a check is done to detect whether a discontinuity due to some state value exists. If it does, then the discontinuity point is located, the model update performed, and the next states recomputed. The next states are then stored, the termination condition for the simulation checked and finally the integration time updated. At the end of each communication time (TCOM), the model outputs are collected and saved for future display. Finally, the next communication time is computed using the information specified for the frequency of communication (COMINT). The process is repeated until the present computational time is greater than the specified end time of the simulation. Figs. 3.6(a) and (b) further clarify the computational steps that must be carried out during the run time of a single component model.

Finally, as shown in Figure 3.7, the Output module and the optional post-run requirements in the Run section provide all the information for the post-run activities of the procedural simulation program. These include carrying out all the post-run computations and displaying the output in tabular as well as graphical forms. The latter uses the Disspla package for this purpose.



f : Transition function
g : Output function

$S(T_N + H) = f(S(T_N), I[T_N, T_N + H])$
$O(T_N) = g(S(T_N))$

Initial Time : T_o
$T_1 = T_o + H$
End Time $T_2 = T_1 + H$
S : State Vector (S_1, S_2)
I : Input Vector (I_1, I_2)
O : Output Vecctor (O1,O2)

At Time T_o:

- Initialize State Variables
  $S_1(T_o)$: given
  $S_2(T_o)$: given
- Compute Initial Inputs
  $I_1(T_o)$ and $I_2(T_o)$
- Compute Outputs
  $O_1(T_o) = g_1(I(T_o), S(T_o))$
  $O_2(T_o) = g_2(I(T_o), S(T_o))$
- * - Compute Inputs $I_1(T_o, T_1]$ and $I_2(T_o, T_1]$
- Compute Next States
  $S_1(T_1) = f_1(S(T_o), I[T_o, T_1])$
  $S_2(T_1) = f_2(S(T_o), I[T_o, T_1])$
- Update Time
  $T_1 = T_o + H$

At Time T_1:

- Compute Outputs
  $O_1(T_1) = g_1(I(T_1), S(T_1))$
  $O_2(T_1) = g_2(I(T_1) S(T_1))$
- * - Compute Inputs $I_1(T_1, T_2]$ and $I_2(T_1, T_2]$
- Compute Next States
  $S_1(T_2) = f_1(S(T_1), I[T_1, T_2])$
  $S_2(T_2) = f_2(S(T_1), I[T_1, T_2])$
- Update Time
  $T_2 = T_1 + H$

At Time T_2:

$S_1(T_2)$ = given from last step
$S_2(T_2)$ = given from last step

- Compute Outputs
  $O_1(T_2) = g_1(I(T_2), S(T_2))$
  $O_2(T_2) = g_2(I(T_2), S(T_2))$

* (computed only at points needed for integration)

Figure 3.6(a) Computation of States and Outputs for a Single Component Model in the Interval $[T_o, T_2]$

		State		External Inputs		Outputs		
Time	Action	S_1	S_2	I_1	I_2	O_1	O_2	Comments
$T = T_o$	1) Given	$S_1(T_o)$	$S_2(T_o)$					Initialize State Variables
	2) Compute Outputs					$O_1(T_o)$	$O_2(T_o)$	Evaluate all Output Functions
	3) Given or Evaluate			$I_1[T_o, T_1]$	$I_2[T_o, T_1]$			Get values of External Inputs
	4) Compute Next States	$S_1(T_1)$	$S_2(T_1)$					Compute using integration algorithm, derivative section of model and initial states
	5) Update Time							$T_1 = T_o + H$
$T = T_1$	1) Given	$S_1(T_1)$	$S_2(T_1)$					All states are available from previous step
	2) Compute Outputs					$O_1(T_1)$	$O_2(T_1)$	Evaluate all Output Functions
	3) Given or Evaluate			$I_1[T_1, T_2]$	$I_2[T_1, T_2]$			Given or Computed from Input Function or Interpolated from Tabular Functions
	4) Compute Next States	$S_1(T_2)$	$S_2(T_2)$					Compute using integration algorithm, previous states and derivative section of model
	5) Update Time							$T_2 = T_1 + H$, Integration finished
$T = T_2$	1) Given	$S_1(T_2)$	$S_2(T_2)$					All states are available from previous step
	2) Compute Outputs					$O_1(T_2)$	$O_2(T_2)$	Evaluate all output functions

Figure 3.6(b) Computational Steps Required During Run Time of Single Component Model in Interval $[T_o, T_2]$

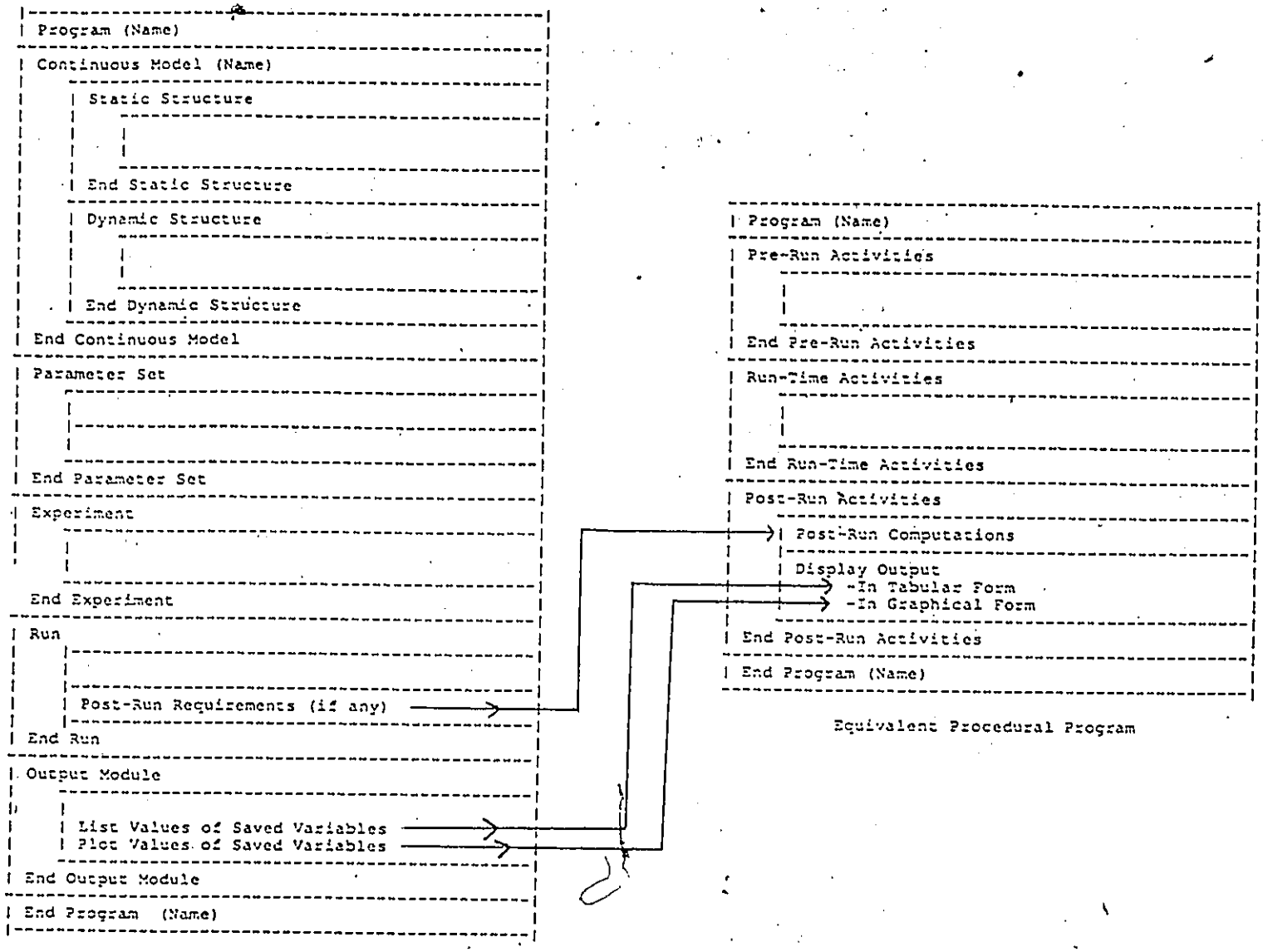


Fig 3.7 Correspondence Between Gest Specification And Equivalent Procedural Program For Post-Run Activities

3.2.2 Gest Program: Several Component Models

3.2.2.1 Coupled Model

In the case of a coupled model which is a Gest program with several component models, each component model is specified separately. Then the coupling or input/output relationships of the component models are specified. In Gest, the coupling of component models consists of : 1) external coupling where the equivalent external and internal inputs and outputs are specified. 2) Internal coupling where the input/output relationships between the component models are specified.

The example model Z given in Figure.3.8 consists of two component models M and N. As can be seen, both external and internal coupling exist for model Z and can be specified as follows : 1) External Coupling : The external input is given by IN and is specified as Z.IN. It corresponds to the internal input B of component model M. Hence, as shown, the coupling information is given by:

Z.IN --› M.B

Similarly, the internal output H of component model N corresponds to the output OUT of the model Z. As a result, the coupling information is given by :

Z.OUT ‹-- N.H

2) Internal coupling: For every input of each component model one distinguishes external inputs and internal inputs.

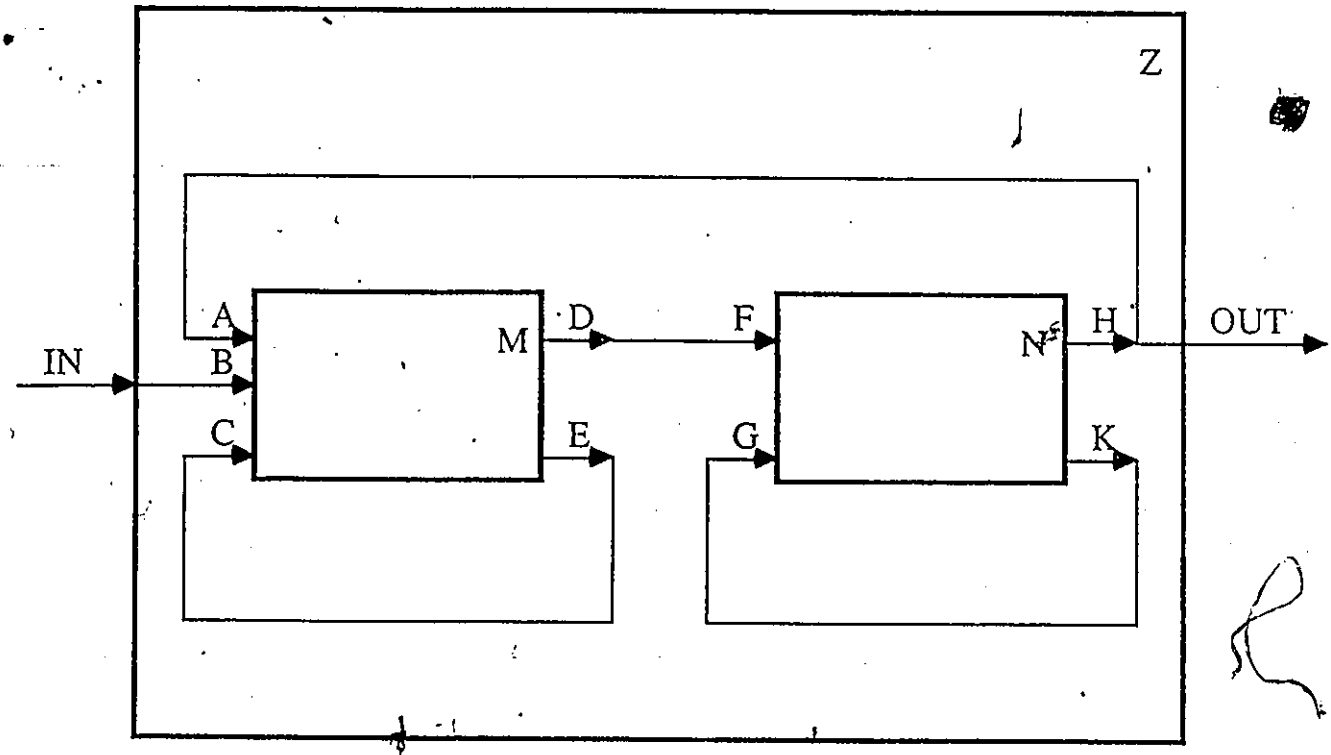


Figure 3.8 Pictorial Representation of Coupled Model Z With Component Models M and N

An external input of a component model is connected to an external input of the coupled model. All other inputs are internal inputs. Afterwards, for every internal input, one specifies from which output variable of which component model the values are provided. The internal coupling information that exists between the component systems M and N of model Z is given by :

M.A <-- N.H

M.C <-- M.E

N.F <-- M.D

N.G <-- N.K

It should be noted that the left hand side of the specification would be automatically generated by the Magest system of the Gest environment and the user would be asked to complete the specification.

To be able to specify a coupled model, one has to finish specification of each component model. Finally, for the parameter set and experimentation section, further modularization is required. The parameter set of the coupled model would consist of the specification of the parameters and tabular functions for each component model in a modular fashion. The experiment would consist of modular and separate specifications of the input functions and initial states for each component model. The specification of the Gest program consisting of several component models is shown in Figure 3.9.

```
Program (Name)                ( For Coupled Model)
Coupled Model (Name)
    Externals
        Inputs  -list of external inputs
        Outputs -list of external outputs
    End Externals
    Component Models -list of component models
    Macro Models     - list of macro models
    External Coupling
        Inputs  -list of equivalent external and internal inputs
        Outputs -list of equivalent external and internal outputs
    End External Coupling
    Internal Coupling
        For every component model
            For every input variable
                (not appearing in the external coupling)
                Specify from which output variable of which
                  component model the values are provided
            Loop
        Loop
    End Internal Coupling
End Coupled Model (Name)
Parameter Set
    For every  component model specify
        Parameter
        Tabular Functions
    Loop
End Parameter Set
Experiment
    For every component model specify
        Input Functions
        Initial States
    Loop
    Associated Time and Integration Knowledge for Simulation
End Experiment
Run
    Specify the Triplet          (Model, Parameter Set, Experiment)
    Post-Run Requirements        (Optional)
End Run
Output Module
    Print Output Heading
    List and Plot Variables of Interest
End Output Module
End Program (Name)
```

Figure 3.9 Gest Specification For A Coupled Model With Several Component Models

3.2.2.2 Multi-Rate Gest Program

The multi-rate Gest program specifies the model with several component models in such a way as to facilitate the use of the multi-rate integration algorithm. This algorithm is discussed in detail in Chapter 4. Essentially, a system having widely different dynamic characteristics is partitioned into two component systems having similar characteristics. These component systems are integrated using the best suited algorithms with independent error control. Hence, the main differences from a coupled Gest program are : 1) Two component models comprising of the Fast Continuous Model and Slow Continuous Model must be specified with the appropriate coupling as in the Gest Program for the coupled model. 2) In the experimentation section, for each component model an integration algorithm along with its allowed error is specified. The details of the specification are shown in Figure 3.10.

3.2.2.3 Interactive Procedural Program

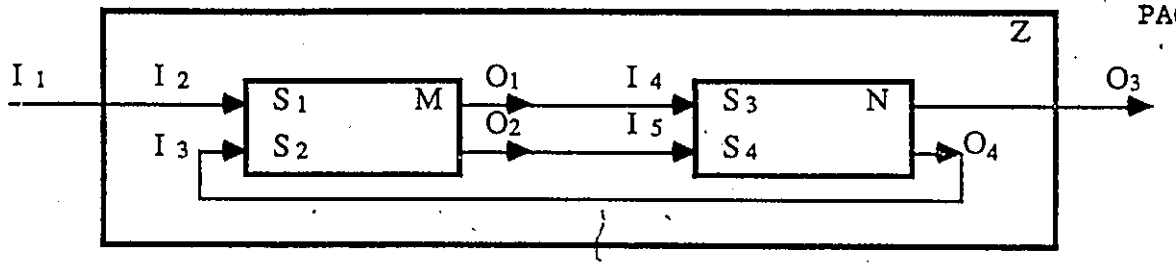
The primary distinction between the procedural program for a single component model and a coupled model with several component models is in the computation steps required within the run time activities as shown in Figure 3.11 and further elaborated in Figs. 3.12 (a) and (b) proceed in a similar manner to that of the single component model using the initial states, and the derivative

```
PROGRAM (NAME)        ( For Coupled Model Using Multi-Rate Algorithm)
Multi-Rate Model (Name)
    Externals
        Inputs  -list of external inputs
        Outputs -list of external outputs
    End Externals
    Component Models
        Slow Continuous Model (Name)
        Fast Continuous Model (Name)
    End Component Models
    Macro Models - list of macro models
    External Coupling
        Inputs  -list of equivalent external and internal inputs
        Outputs -list of equivalent external and internal outputs
    End External Coupling
    Internal Coupling
        For every component model
            For every input variable
                (not appearing in the external coupling)
                Specify from which output variable of which
                  component model the values are provided
            Loop
        Loop
    End Internal Coupling
End Multi-Rate Model (Name)
Parameter Set
    For every  component model (i.e. Fast model,Slow model) specify
        Parameter
        Tabular Functions
    Loop
End Parameter Set
Experiment
    For every component model (i.e. Fast model,Slow model) specify
        Integration Algorithm to be used
        User tolerance
        Input Functions
        Initial States
    Loop
    Specify Associated Time Knowledge for Simulation
End Experiment
Run
    Specify the Triplet          (Model, Parameter Set, Experiment)
    Post-Run Requirements        (Optional)
End Run
Output Module
    Print Output Heading
    List and Plot Variables of Interest
End Output Module
End Program (Name)
```

Figure 3.10 Specification For Multi-Rate Gest Program With Two Component Models

```
Program (Name)                                          (For Coupled Model)
  Pre-Run Activities                                    (Initial Segment)
    Initialization
      Parameters
      Tabular Functions
      State Variables
      External Input Variables
      Run-Control Variables
          Time
          Integration
          Dummy Indices
    End Initialization
    Pre-Run Output
      Print Heading
      Print Values of Parameters
      Print Values of Initial State(s)
    End Pre-Run Output
  End Pre-Run Activities
  Run-Time Activities                                   (Dynamic Segment)
    While  T <= TFIN
      While T <= TCOM
        If there is a time for a model update
          Then | Perform model update
        End If
        Compute external input(s)
        Compute model output(s)
        Use External Coupling to Get Some Internal Input
        Determine the remaining Internal Inputs by Using
          coupling information
        Compute next state(s)
        If a state value for a model update is detected
          Then | Locate the discontinuity point
               | Perform model update
               | Recompute next state(s)
        End If
        Store next state(s)
        Terminate simulation if condition given as
          a function of state variable is satisfied
        Time update : T = T + h
      End While
      Collect and save model output(s) at communication time
      Update TCOM: TCOM = TCOM + COMINT
    End While
    Collect and save model output(s) at TFIN
  End Run-Time Activities
  Post-Run Activities
    Post-Run Computations (if any)
    Display Output
      -In Tabular Form
      -In Graphical Form
  End Post-Run Activities
End Program (Name)
```

Figure 3.11 Equivalent Procedural Program For A Coupled Model With Several Component Models



f : Transition function
g : Output function

S_M: Model M State Vector (S_1, S_2)
I_M : Model M Input Vector (I_2, I_3)
O_M: Model M Output Vector (O_1, O_2)
$S(T_N + H) = f(S(T_N), I[T_N, T_N + H])$
$O(T_N) = g(S(T_N))$

Initial Time: T_O
$T_1 = T_O + H$
End Time: $T_2 = T_1 + H$
S_N: Model N State Vector (S_3, S_4)
I_N : Model N Input Vector (I_4, I_5)
O_N: Model N Output Vector (O_3, O_4)

At Time T_O:
- Initialize State Variables
  $S_M(T_O)$: given
  $S_N(T_O)$: given
- Compute Outputs
  $O_M(T_O) = g_1(S_M(T_O))$
  $O_N(T_O) = g_2(S_N(T_O))$
* - Compute Inputs $I_M[T_O, T_1]$ and $I_N[T_O, T_1]$
- Compute Next States
  $S_M(T_1) = f_1(S_M(T_O), I_M[T_O, T_1])$
  $S_N(T_1) = f_2(S_N(T_O), I_N[T_O, T_1])$
- Update Time
  $T_1 = T_O + H$

At Time T_1:
- Compute Outputs
  $O_M(T_1) = g_1(S_M(T_1))$
  $O_N(T_1) = g_2(S_N(T_1))$
* - Compute Inputs $I_M[T_1, T_2]$ and $I_N[T_1, T_2]$
- Compute Next States
  $S_M(T_2) = f_1(S_M(T_1), I_M[T_1, T_2])$
  $S_N(T_2) = f_2(S_N(T_1), I_N[T_1, T_2])$
- Update Time
  $T_2 = T_1 + H$

At Time T_2:
  $S_M(T_2)$ = given from last step
  $S_N(T_2)$ = given from last step
- Compute Outputs
  $O_M(T_2) = g_1(S_M(T_2))$
  $O_N(T_2) = g_2(S_N(T_2))$

* (computed only at points needed for integration)

Figure 3.12(a) Computation of States and Outputs of a Coupled Model Z With Component Models M and N in the Interval [T_O, T_2].

		Model M States	Model N States	External Input	Model M Inputs		Model M Outputs		Model N Inputs		Model N Outputs		
Time	Action	S_M	S_N	I_1	I_2	I_3	O_1	O_2	I_4	I_5	O_4	O_3	
$T = T_0$	1) Given	$S_M(T_0)$	$S_N(T_0)$										Initialize State Variables In all Component Models
	2) Compute Outputs						$O_1(T_0)$	$O_2(T_0)$			$O_4(T_0)$	$O_3(T_0)$	For all component models, evaluate output functions to get output variables
	3) Given or Evaluate			$I_1[T_0, T_1]$									All external input for all component models given or evaluated from input functions
	4) Determine Some Internal Inputs			$I_2(T_0)=I_1(T_0)$									Evaluate all external couplings
	5) Determine remaining internal inputs					$I_3(T_0)=O_4(T_0)$			$I_4(T_0)=O_1(T_0)$	$I_5(T_0)=O_2(T_0)$			Evaluate all internal couplings
	6) Compute next states	$S_M(T_1)$	$S_N(T_1)$										Compute using integration algorithm, initial states, and state derivatives
	7) Update Time												$T = T_0 + H = T_1$
$T = T_1$	1) Given	$S_M(T_1)$	$S_N(T_1)$										All states available from the previous step
	2) Compute Outputs												All external inputs for all component models given or evaluated from input functions
	3) Given or Evaluate			$I_1[T_1, T_2]$			$O_1(T_1)$	$O_2(T_1)$			$O_4(T_1)$	$O_3(T_1)$	For all component models, evaluate output functions to get output variables
	4) Determine Some Internal Inputs			$I_2(T_1)=I_1(T_1)$									Evaluate all external couplings
	5) Determine remaining internal inputs					$I_3(T_1)=O_4(T_1)$			$I_4(T_1)=O_1(T_1)$	$I_5(T_1)=O_2(T_1)$			Evaluate all internal couplings
	6) Compute next states	$S_M(T_2)$	$S_N(T_2)$										Compute using integration, algorithm, initial states, and state derivatives.
	7) Update Time												$T = T_1 + H = T_2$
$T = T_2$	1) Given	$S_M(T_2)$	$S_N(T_2)$										All states available from the previous step
	2) Compute Outputs						$O_1(T_2)$	$O_2(T_1)$			$O_4(T_2)$	$O_3(T_2)$	For all component models, evaluate output functions to get output variables

Figure 3.12(b) Computational Steps Required During Run Time of Coupled Model In Interval $[T_0, T_2]$

subroutine which is a single collection of the differential equations of the component models. However, at the end of each integration step, internal coupling between the component systems and external coupling may be necessary as opposed to the single component model where this does not occur. As a result, once the external input is computed, the coupling information is used to get the associated internal input. The output(s) of the model may also need to be determined from the output(s) of some component model(s) via the external coupling information. In addition, values of any remaining internal inputs are determined from the internal coupling information.

The same steps are carried out for the Multi-rate Gest program. However, the Multi-rate algorithm integrates the coupled model by integrating two derivative subroutines for the Slow and Fast continuous component models independently.

CHAPTER 4

GEST RUN-TIME LIBRARY - SECTION 1

4.1 INTRODUCTION

The run-time library for the simulation specification language Gest, consists of a variety of integration algorithms in the form of structured Fortran subprograms designed to generate the numerical solution of continuous models which can be described as non-stiff initial value problems.

The variety of algorithms to be included in such a run-time library is a very important decision since not all codes are equally effective in their handling and up to date, no single integration algorithm has been found to handle all kinds of problems equally well. As a result, users must be careful which algorithm they apply to particular situations. In the last decade, considerable research by Hull et. al. (1972), Hull (1980), and Shampine et. al. (1980) has been done to compare certain aspects of existing methods in order to identify those methods which would be good candidates for inclusion in such a library. In addition, apart from the numerous algorithms available, there are also as pointed out by Gear (1981) a number of major problem areas which require specialized algorithms. Thus, it has been decided that initially the run-time library should consist of algorithms to deal specifically

with a few problem areas with a view of expansion in the future. As stated previously, the set of problems chosen belong to the non-stiff initial value type. In particular, they consist of problems having known discontinuities, multi-rate methods, and general purpose methods.

The discontinuity handling algorithm is designed to deal with piece-wise continuous systems which have known time event discontinuities, state event discontinuities, or a combination of both. The multi-rate integration algorithm is included for use in the integration of large dynamic systems having widely different dynamic characteristics that can be efficiently partitioned into two component systems with similar characteristics. In addition, the coupling that exists between the component systems should involve few interconnections. Finally, some of the more popular general purpose algorithms are included to be used in the integration of systems which can be described by a fixed set of differential equations with state variables and first state derivatives both being continuous over the whole run. These include the fixed step Runge Kutta method, the variable step Runge-Kutta Fehlberg method and the variable step Adams Predictor Corrector Method. Each of these methods may be advantageous depending on such factors as the complexity of the model, the user's accuracy requirements, and the frequency of output stipulated. This is discussed at greater length in Chapter 6.

As described by Shampine et. al. (1980) the more popular methods for the solution of a non-stiff initial value problem are the Runge Kutta, Adams and extrapolation methods. Depending on the structure of the particular problem, one of these is usually more advantageous. Shampine et. al. (1980) concluded that extrapolation codes require a lot of storage and a high overhead. In addition, they are quite inefficient in dealing with problems exhibiting mild stiffness or requiring frequent output. It is recommended that they be exchanged with a variable order Runge Kutta in a library. However, the advantages of the latter are usually in coping with a wide range of tolerances and in the extreme instances handling a discontinuity in a derivative. The latter case is easily handled using the discontinuity handling algorithm described above. To deal with the range of tolerances, we have included a high order variable step Runge-Kutta Fehlberg method for stringent tolerances demanded by the user and a low order fixed step Runge-Kutta method for less stringent ones (Hull, T.E. et. al. 1972). Hence, in our library, the variable order Runge-Kutta method proposed by Shampine (1980) has been replaced by high and low order Runge-Kutta methods and an efficient discontinuity handling algorithm. As discussed by Shampine et.al. (1980), the Runge Kutta codes could not possibly replace the Adams especially for situations where function evaluations are relatively expensive. To this end, the variable step Adams Predictor Corrector method has been

included in the library.

In each of the sections 4.2 and 4.3, the general purpose algorithms and the multi-rate algorithm are discussed in detail. As explained in Chapter 3, the modeler specifies in the Experiment section of his Gest program the type of integration algorithm to be used to drive the particular model in question. Once the Gest program has been translated into an equivalent procedural program, the integration algorithm requested by the user is used to carry out the run-time activities of the model. Hence, one may perceive the integration algorithm to be embedded within the procedural simulation program.

4.2 GENERAL PURPOSE INTEGRATION ALGORITHMS

4.2.1 Fourth-Order Runge-Kutta

4.2.1.1 Description of Algorithm

The Fourth Order Runge-Kutta algorithm is one of a family of Runge Kutta formulas which are powerful numerical techniques for solving ordinary differential equations. These numerical techniques can be applied with varying degrees of success and effectiveness to virtually any differential equation which is in sharp contrast to the problem of trying to find an analytical solution to a differential equation (Hornbeck 1975). The Fourth Order Runge-Kutta Method is to be used in generating the solution of initial value problems and utilizes a fixed step size procedure.

If we consider the initial value problem :

$Y' = F(T, Y(t))$

with $Y(0) = Y_0$ (given) over the interval (T_0 , T_f) where Y' represents a set of n first order differential equations.

As illustrated in Figure 4.1, the objective of the Runge-Kutta technique is to advance the solution in steps such as $T_{j+1} = T_j + H$ over some interval $[T_0 , T_f]$ which is clearly an extrapolation principle (Hornbeck 1975). In using formulas of the Runge-Kutta type to solve such problems, the

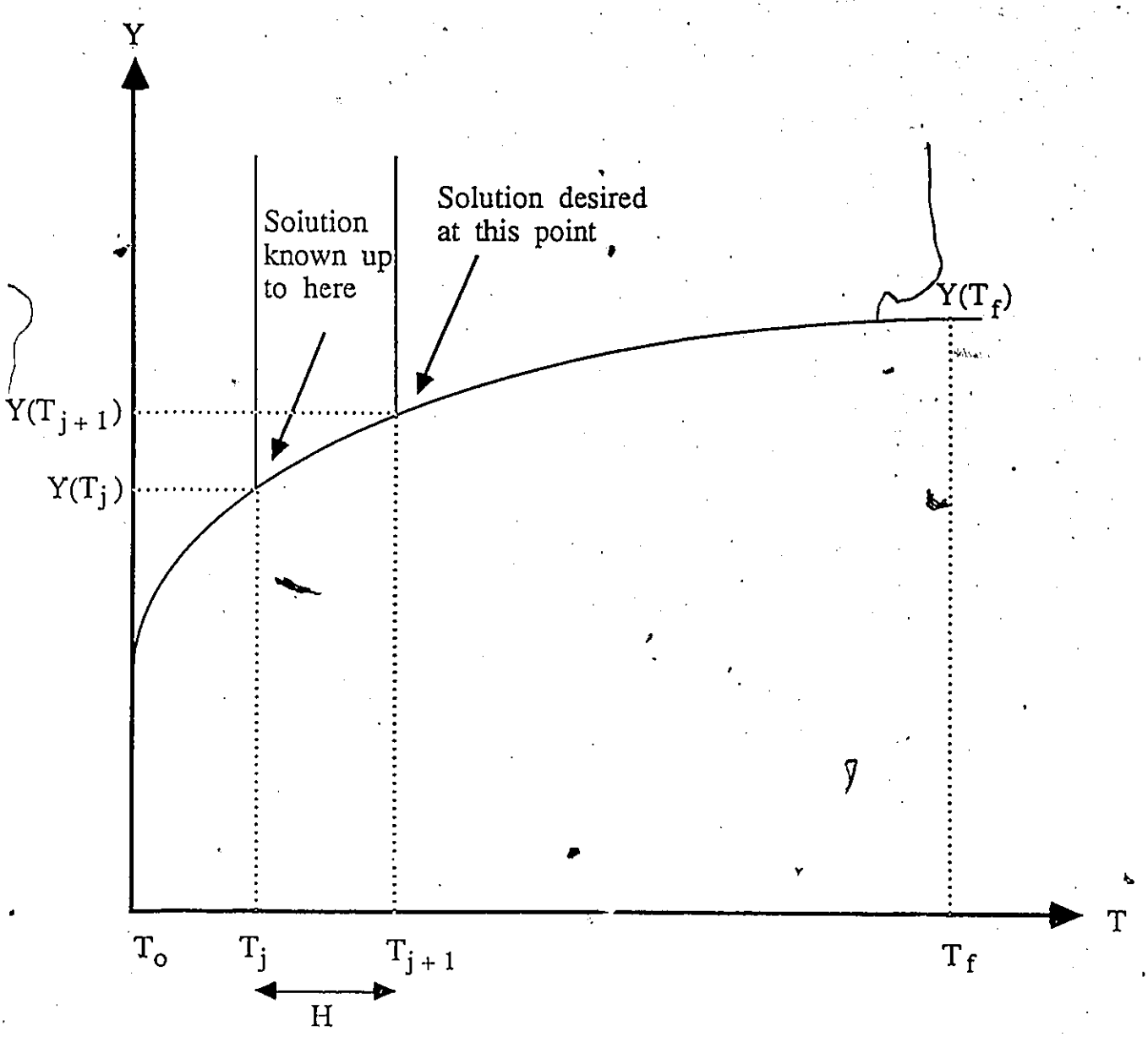


Figure 4.1 Solution Procedures for Runge-Kutta Methods

desired solution Y_{j+1} is obtained in terms of Y_j ,$F(T_j , Y_j)$ and $F(T,Y)$ evaluated for various estimated values of Y between T_j and $T_j + H$. The method is a self-starting one requiring no previous values for Y or $F(T,Y)$. Hence in the initial value problem above, at the first step, $T_j = 0$ and $Y_j = Y(0) = Y_0$, the method can be used to find $Y_{j+1} = Y(0 + H) = Y(H)$ and any other successive values of Y.

As applied to the initial value problem stated above, the formulas used by the fourth order Runge Kutta Method for the step from Y_j to Y_{j+1} are :

$$S1 = F(T_j , Y_j)$$

$$S2 = F(T_j + (H/2), Y_j + (S1 * H/2))$$

$$S3 = F(T_j + (H/2), Y_j + (S2 * H/2))$$

$$S4 = F(T_j + H, Y_j + (S3 * H))$$

$$Y_{j+1} = Y_j + (H/6)*(S1 + 2*S2 + 2*S3 + S4)$$

Graphically, as shown in Figure 4.2, one can consider S1....S4 as slopes and the values of $Y_j + (S1 * H/2)$, $Y_j + (S2 * H/2)$ as approximations to Y_j at $T_{j+1/2}$ and $Y_j + (S3 * H)$ as an approximation to Y at T_{j+1} .

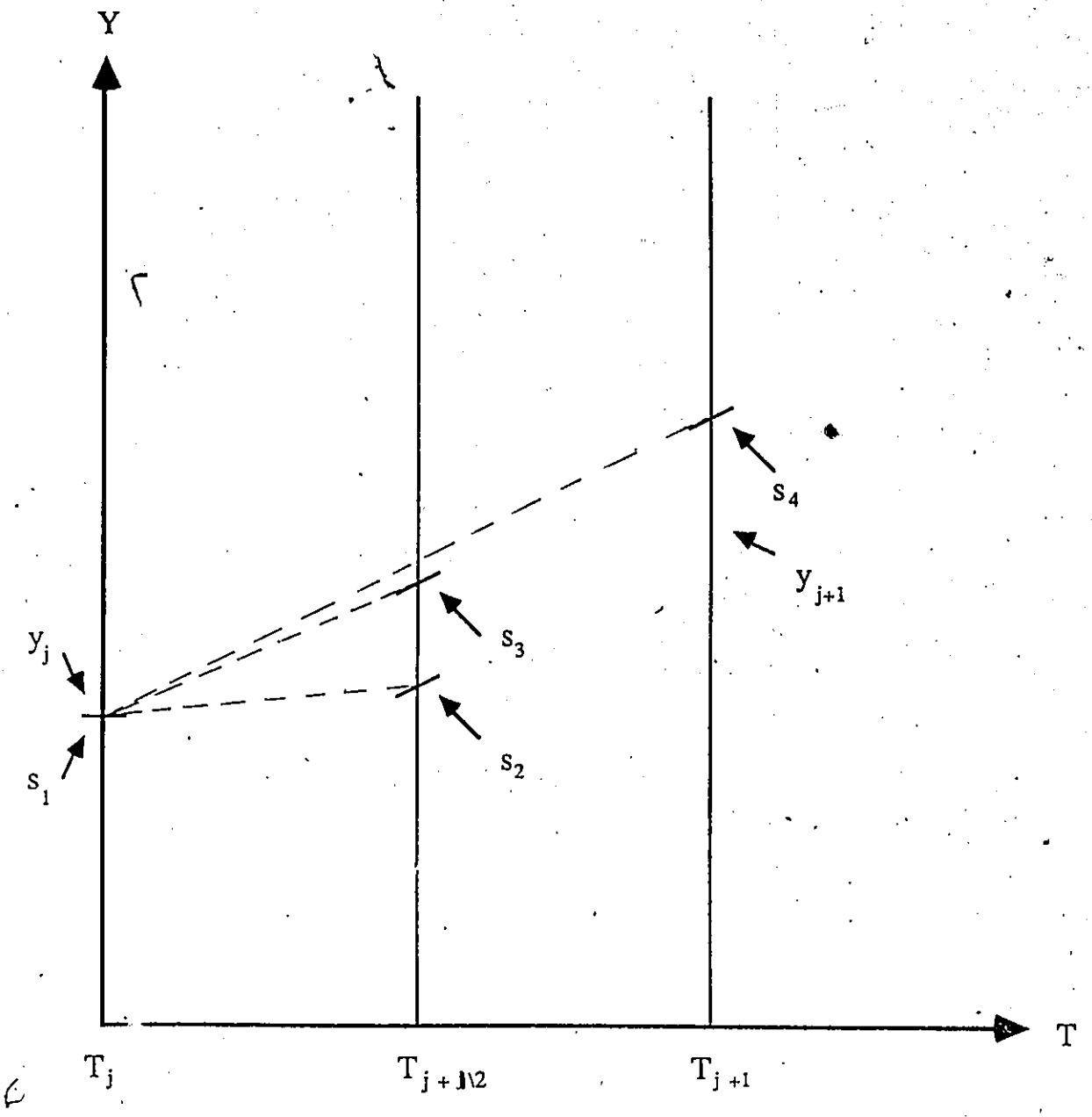


Figure 4.2 Graphical Representation of One Step of the Fourth Order Runge-Kutta Method

4.2.1.2 Structured Algorithm

Figures 4.3(a) and (b) show the details of the Fourth Order Runge-Kutta algorithm used in the run-time library.

List of variables for Fourth Order Runge-Kutta Algorithm

T	- Computational Time
TINIT	- Initial Time of Simulation
TFIN	- End Time of Simulation
TCOM	- Communication Time
COMINT	- Communication interval
H	- Current step size
HINIT	- Initial step size
HMIN	- Minimum allowable step size
HBOUND	- Upper bound on initial step size
Y	- Vector of state variables
N	- Dimension of vector
TOL	- User tolerance
FDER	- Maximum value of derivative functions at initial point
NFLAG	- = 0 TCOM has not been detected = 1 TCOM has been detected

Figure 4.3(a) List of variables used in RK4 Algorithm

```
Subroutine Fourth Order Runge-Kutta

                          Initialization

Set T = TINIT
Set NFLAG = 0                        (TCOM has not been detected)
Compute initial step size : HINIT
    HINIT = min(HBOUND, ((abs(0.5*TOL/FDER)) ** 0.5)/4)
Compute initial  Communication Time:
    TCOM = T + COMINT
Set H = HINIT
-------------------------------------------------------------------
While (T < TFIN)
    ---------------------------------------------------------------
    While (T < TCOM)
        -----------------------------------------------------------
        If (((TCOM - T) <= (HINIT + HMIN) AND (NFLAG = 0)))
            -------------------------------------------------------
            | Then | H = (TCOM - T)/2
                   | Set NFLAG = 1     (Communication time detected)
            -------------------------------------------------------
        End If
        -----------------------------------------------------------
        Compute next states for each state variable
        Update Time:
           T = T + H
        -----------------------------------------------------------
    End While
    ---------------------------------------------------------------
    Call OUTPUT (Y,N,T)
    Compute next TCOM:
        TCOM = T + COMINT
    ---------------------------------------------------------------
    If ((TCOM >= TFIN)
        -----------------------------------------------------------
        | Then | TCOM = TFIN
        -----------------------------------------------------------
    End If
    ---------------------------------------------------------------
    Set NFLAG = 0
    Set H = HINIT
    ---------------------------------------------------------------
End While
-------------------------------------------------------------------
CALL OUTPUT(Y,N,T)
-------------------------------------------------------------------
EndSubroutine Fourth Order Runge-Kutta
-------------------------------------------------------------------
```

Figure 4.3.(b) Details of The RK4 Algorithm

4.2.1.3 Program Structure

Figure 4.4 shows the pictorial view of a program using the Fourth Order Runge-Kutta method to drive the simulation of a single component model.

Main Program

The structure of the main program has three main blocks, namely pre-run activities run-time and post-run activities as discussed in Chapter 3. In the pre-run activities, a CALL is made to subroutine INIT. This subroutine invokes the XMENU and the user initializes all the state variables, parameters, and experimentation variables contained on the full screen menu. After this, any necessary pre-run computations are carried out as well as the pre-run output.

The run time activities of the main program involve a CALL to the RK4 algorithm. This integrates the system over the length of the simulation. Control is returned to the main program after the run is complete, at which point results are displayed tabular form and a CALL is made to the subroutine GRAPH in order to plot the results.

Subroutine INIT

This subroutine is linked to the MAIN program and utilizes the XMENU software within the IBM environment to allow the user to interactively initialize the appropriate model parameters, initial values, and other experimentation variables.

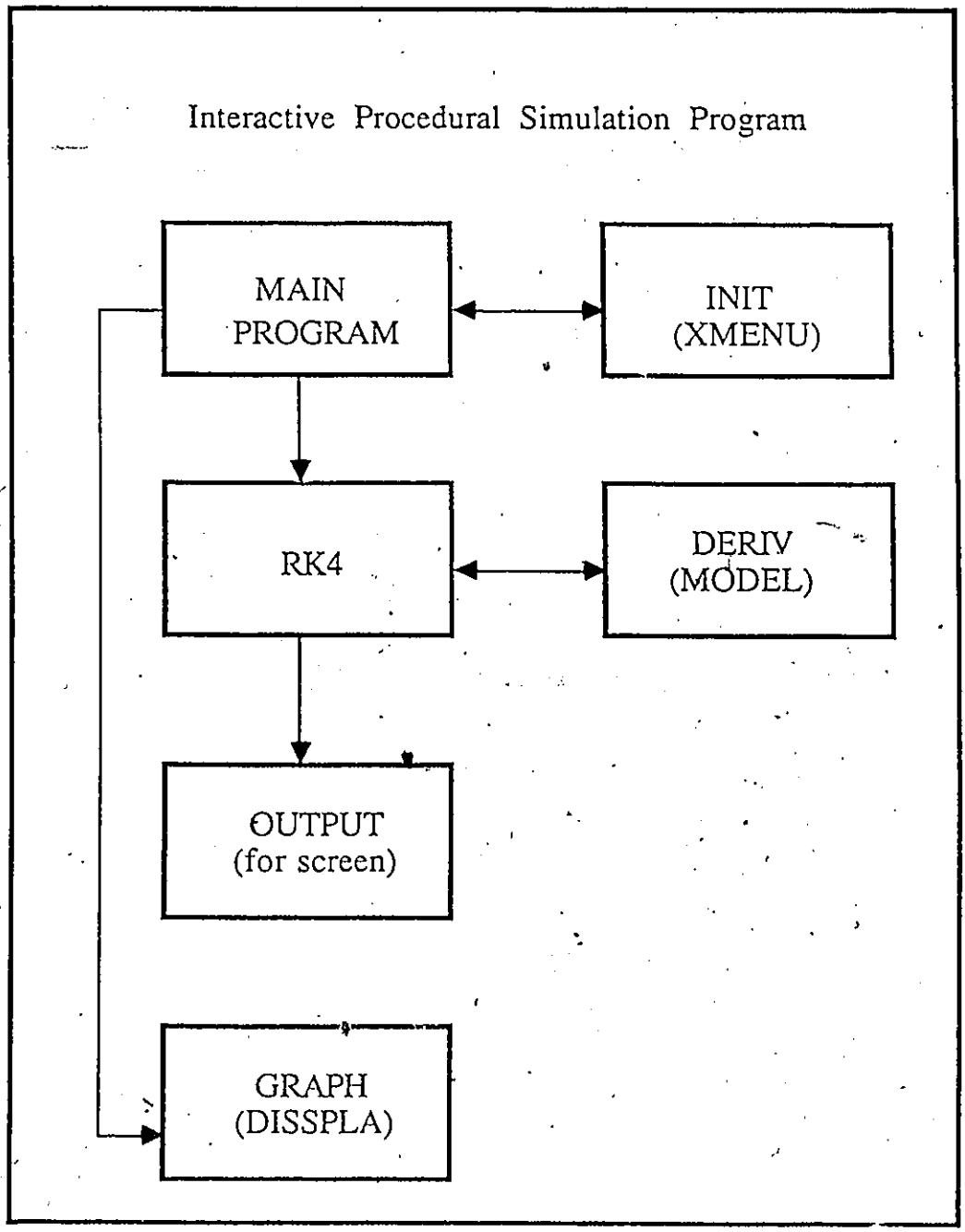


Figure 4.4 Modules of Interactive Procedural Simulation Program using the RK4 Algorithm

Subroutine RK4

This subroutine implements the fourth order Runge-Kutta integration method for integrating systems having frequent output points but not requiring a high level of accuracy. It makes calls at each stage of the integration to the DERIV subroutine. Additionally, at every communication time, a call is made to the OUTPUT subroutine.

Subroutine DERIV

This is a model dependent routine and contains the necessary functions and information to carry out the computational steps. These consist of the output function(s), the external input(s) if any exist, the internal and external coupling information, and the derivatives of the state variables that describe the model in the order shown in Figure 4.5. For a coupled model with several component models, the output functions that may exist for each component model are simply lumped together. In a similar way, as shown, the derivative functions that describe each component model are lumped together.

Subroutine OUTPUT

The values of the outputs at the communication intervals are stored and an interactive tabular display is shown to

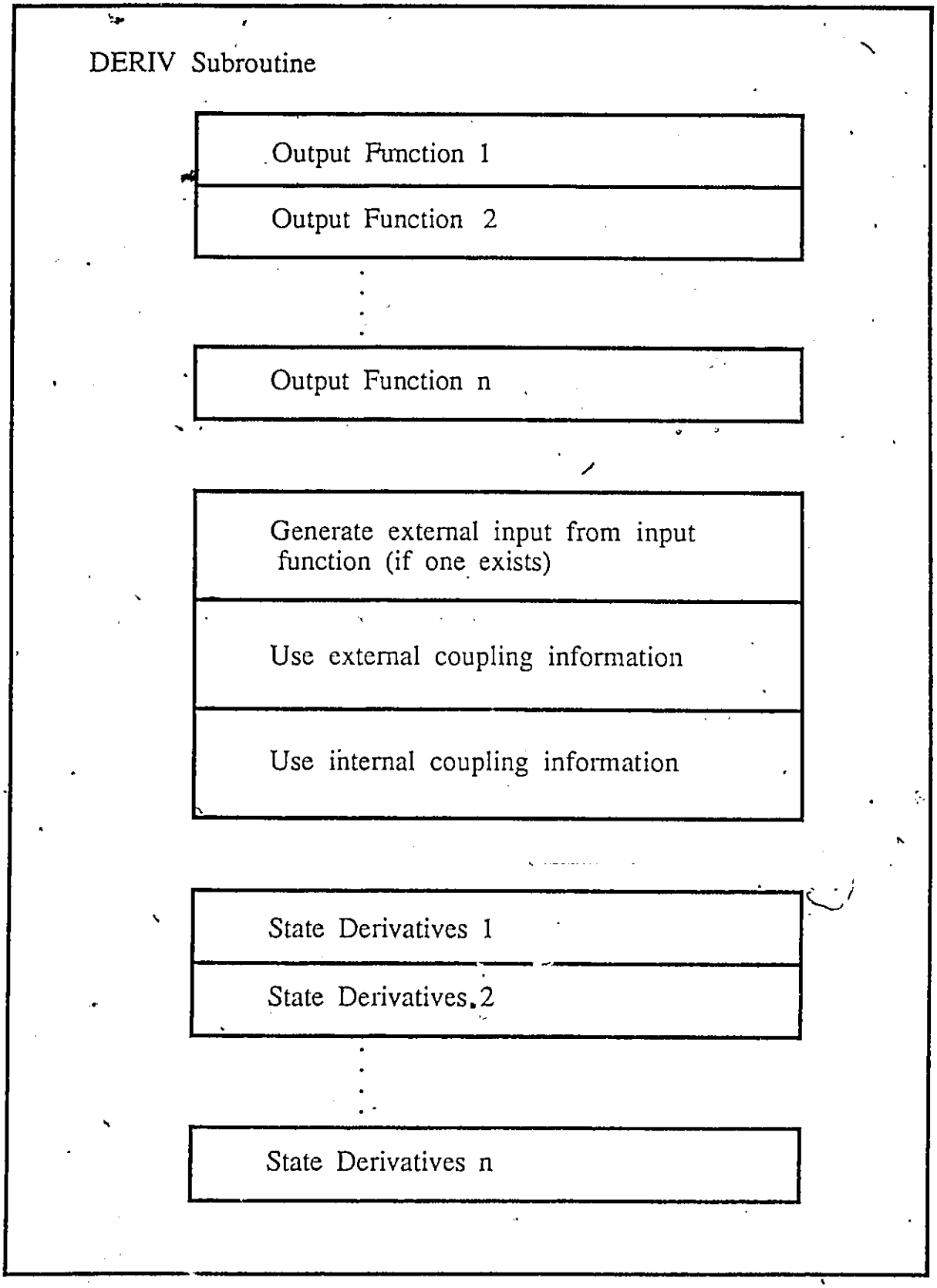


Figure 4.5 Details of DERIV Subroutine

the user during the simulation run.

Subroutine GRAPH

This subroutine linked to the MAIN program utilizes the Disspla software within the IBM environment to provide post-run hard copy graphic displays of the desired model outputs with time.

The fixed step RK4 algorithm is imbedded within the Diabetes Mellitus model example given in Appendix A and in the Forest Biomass model example given in Appendix F.

4.2.2 FIFTH-ORDER RUNGE-KUTTA FEHLBERG

4.2.2.1 Description of Algorithm

This method is a fifth order variable step algorithm and is another in the family of the Runge Kuttas. The algorithm involves computing six function evaluations, but then an estimate of the local truncation error is made available which is used as a basis to decide on the integration step size.

The basis is to compute two Runge-Kutta estimates for the new value of Y_{j+1} but of different orders of error. Hence, instead of comparing estimates of Y_{j+1} for H and H/2 as is necessary for the fixed step Runge-Kutta, one compares the estimates of Y_{j+1} using fourth and fifth order Runge-Kutta formulas and comparing the results. The formula for the step

Yj to Yj+1 is given as follows (Gerald and Wheatley 1984 P.310) where Y is the vector of state variables and F(T,Y) is the value of the derivative function at time T.

Compute:

```
K1 = H * F(T, Y(T))
K2 = H * F(T + H/4, Y(T) + K1/4)
K3 = H * F(T + 3*H/8, Y(T) + (3/32)*K1 + (9/32)*K2)
K4 = H * F(T + 12*H/13, Y(T) + (1932/2197)*K1
         - (7200/2197)*K2 + (7296/2197)*K3)
K5 = H * F(T + H, Y(T) + (439/216)*K1 - 8*K2
         + (3680/513)*K3 - (845/4104)*K4)
K6 = H * F(T + H/2, Y(T) - (8/27)*K1 + 2*K2
         - (3544/2565)*K3+(1859/4104)*K4-(11/40)*K5
ERR= ABS(K1/360-(128/4275)*K3 - (2197/75240)*K4 +
     K5/50 + (2/55)*K6)/H
```

State at T+H:

```
Y(T+H) = Y(T) + ((25/216)*K1 + (1408/2565)*K3 + (2197/4104) *
                K4 - K5/5)
```

Figure 4.6 illustrates the major parts of a variable step size integration algorithm.

```
Initialize integration step size and states
While (T <= TFIN)
    Compute state derivatives for each state variable
    Compute local truncation error estimates
    Determine the maximum local truncation error: ERRMAX
    If (ERRMAX < TOL)
        Then | Compute next states
             | Update Time: T = T + H
             | Update value of H
        Else | Compute reduced value of H  (reject previous step)
    End If
End While
```

Figure 4.6 Major Parts of a Variable Step Size Algorithm

4.2.2.2 STRUCTURED ALGORITHM

Figures 4.7(a), (b), and 4.8 show the details of the fifth order Runge-Kutta algorithm

used in the run-time library.

List of variables for fifth order Runge-Kutta Fehlberg Algorithm

T	- Computational Time
TFIN	- End Time of Simulation
TINIT	- Initial Time of Simulation
TCOM	- Communication Time
COMINT	- Communication interval
H	- Current step size
HINIT	- Initial step size
HPREV	- Previous step size
HMIN	- Minimum allowable step size
HBOUND	- Maximum allowable step size
ERR(N)	- Vector of local error estimates for all state variables
ERRMAX	- Maximum local error estimate
Y	- Vector of state variables
N	- Dimension of vector
TOL	- Integration error tolerance
NORD	- Order of method
FDER	- Maximum value of derivative functions at initial point
NFLAG	- = 0 TCOM has not been detected = 1 TCOM has been detected

Figure 4.7(a) List of Variables Used in RK Fehlberg Algorithm

```
Subroutine Runge-Kutta Fehlberg
Set NFLAG = 0                    (TCOM has not been detected)
Compute initial step size : HINIT
    HINIT = min(HBOUND, ((abs(0.5*TOL/FDER)) ** 0.5)/4)
Compute initial  Communication Time:
    T = T + COMINT
Set H = HINIT
While (T < TFIN)
    While (T < TCOM)
        If (((TCOM - T) <= (HINIT + HMIN) AND (NFLAG.EQ.0))
            Then | H = (TCOM - T)/2
                 | Set NFLAG = 1    (Communication time detected)
        End If
        Compute state derivatives (k1...k6) for each state variable
        Compute local error estimates for each state variable
        Determine the maximum local error estimate : ERRMAX
        If (ERRMAX <= TOL)
            Then | Update Time : T = T + H
                 | Compute new states at T
        End If
        If (NFLAG = 0)
            Then | Perform Step Size Control  (See Fig. 4.9)
        End If
    End While
    Call OUTPUT (Y,N,T)
    Compute next TCOM:
        TCOM = TCOM + COMINT
    If (TCOM >= TFIN)
        Then | TCOM = TFIN
    End If
    Set NFLAG = 0
End While
CALL OUTPUT(Y,N,T)
EndSubroutine Runge-Kutta Fehlberg
```

Figure 4.7.(b) Details of Fifth Order Runge Kutta Fehlberg Algorithm

```
Step Size Control
Set HPREV = H
If (ERRMAX NOT = 0)                    (Compute new step size)
    Then | H = 0.9 * HPREV * (TOL/ERRMAX) ** (1/D)
End If
If (H < HMIN)
    Then | Interrupt simulation run
         | Display values of:
         |       - Integration error tolerance: TOL
         |       - Minimum step size: HMIN
         |       - Current step size: H
         | User asked interactively by system to change HMIN and/TOL
         | If User changes value of HMIN or TOL
         |     Then | Simulation is allowed to proceed
         |     Else | Print message to alert user
         |          | Abort simulation run
         | End If
End If
```

Figure 4.8 Step Size Control for Variable Step Algorithm

4.2.2.3 Program Structure

The program structure for the program using the Runge-Kutta Fehlberg algorithm to drive the simulation of a model is identical to that of a program using the Fourth Order Runge-Kutta algorithm. As a result, no other details are given except to note that the fifth order Runge-Kutta Fehlberg method is used to integrate the system instead of the fourth order Runge-Kutta method.

The variable step Runge Kutta Fehlberg method is used in the Electronic Oscillator model in Appendix B.

4.2.3 FOURTH-ORDER ADAMS PREDICTOR-CORRECTOR METHOD

4.2.3.1 Description of Algorithm

This method belongs to the class of methods having formulas of the multi-step type. These formulas as depicted in Figure 4.9 require previous information for $T<T_j$. In general, such formulas may require the value of Yj and values of F(T,Y) at T_j , T_{j-1} , T_{j-2} and T_{j-3} to compute a solution for Y_{j+1} and are hence not self starting. This class of methods for the numerical solution of ordinary differential equations can be represented by the following formula:

$$Y_{j+1} = \sum_{i=1}^{k}(A_i * Y_{j+1-i}) + H * \sum_{i=0}^{k}(B_i * F_{j+1-i})$$

Special Cases

(i) If k=1, then single step method.

(ii) If k>1, then multi-step method used requiring past values to generate Y_{j+1} .

(iii) If $B_0 = 0$, then the method is termed an open method.

(iv) If B_0 is not equal to 0, then the method is termed a closed one.

This actual algorithm is a combination of open and closed

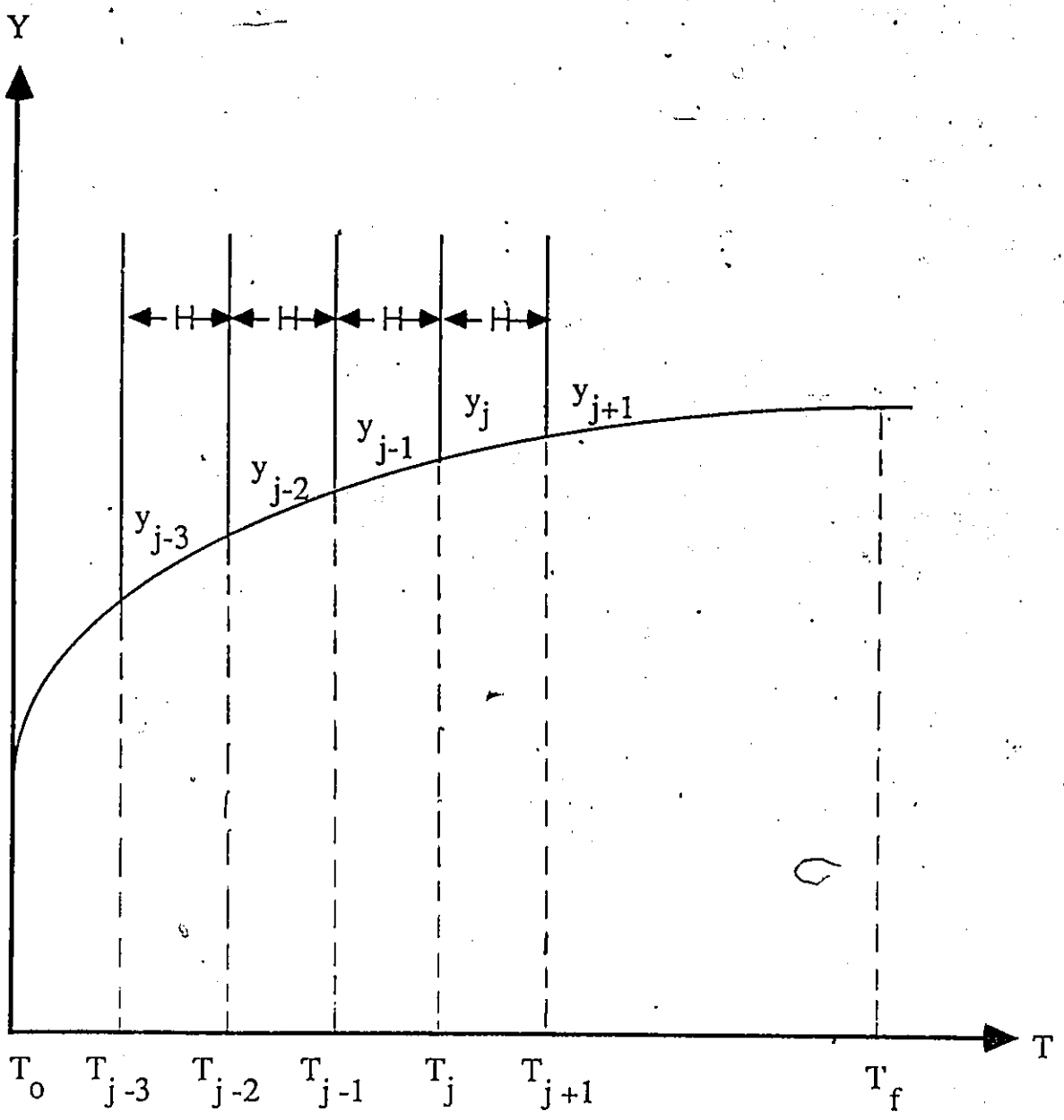


Figure 4.9 Solution Procedures for Multistep type Formulas Requiring Information for $T < T_j$

formulas seeking to utilize the advantages of both types of formulas to make a more efficient algorithm. The open formula used is the predictor method and is called the Adams-Bashforth method given by (Hornbeck 1975):

$$Y_{j+1} = Y_j + ((H/24) * (55*F_j - 59*F_{j-1} + 37*F_{j-2} - 9*F_{j-3})$$

The closed formula used is the corrector method and is called the Adams-Moulton method given by (Hornbeck 1975):

$$Yj+1 = Y_j + ((H/24) * (9*F_{j+1} - 19*F_j - 5*F_{j-1} + F_{j-2})$$

The algorithm's procedure involves firstly using a fourth order Runge-Kutta formula to obtain the values of Y and F for the first three step sizes beyond the initial condition. The open formula described by the Adams Bashforth is used to predict or estimate Y_{j+1} (p) the value of Y at the next step since the closed formula although accurate requires iterative procedure. Then using this predicted value of Y_{j+1} (p) as a first estimate, the closed formula described by the Adams Moulton can be iterated until the desired degree of convergence is obtained. A similar procedure for step size control as that of the Runge-Kutta Fehlberg is used. However, in this case, since the multi-step method requires equal step sizes for the starting values, any change in step size necessitates recalculating new starting values at that point using the Fourth Order Runge-Kutta algorithm. As a result, this algorithm is inefficient when

frequent output points are requested by the user. In addition, the tolerance specified by the user is used to calculate the initial step size as shown in the Runge-Kutta Fehlberg Algorithm.

Finally, the algorithm has embedded in its structure a stiffness indicator. As pointed out by Hull (1980), a desirable feature of a non-stiff method, is to provide a warning of any stiffness that might be experienced. In general terms, a problem exhibits stiffness if its solution components are slowly varying but solutions with very rapidly changing components are possible. These problems are physically important and as a result we have decided to follow the advice of Hull and include a rough measuring criterion for indicating where stiffness is likely to occur. The test is merely to monitor the number of steps required by the models to complete the integration using a low order method since stiffness is usually detected by such a method (Shampine and Gordon 1975 pp.143-145). If the actual number of steps required by the model to complete the integration exceeds the computed value, then the user is alerted as to the possibility of stiffness occuring in the model.

The limitations of the test must be realized. It is quite obvious that false returns can occur. It may be the case that the model actually requires low orders and a lot of work. The test would then return a false statement of stiffness. Thus, the test should not be seen as being

perfect but to identify situations which warrant further investigations as to the validity of the stiffness suspicions given by the test. The value of the number of steps used in the testing procedure is dependent on the interval and the starting step size.

Number of steps: NSTEPS = (TFIN - TINIT) / (HINIT * 10)

4.2.3.2. Structured Algorithm

Figures. 4.10 and 4.11 show the details of the Fourth Order Adams algorithm used in the run-time library.

List of variables for Fourth- Order Adams Algorithm

T	- Computational Time
TINIT	- Initial time of Simulation
TFIN	- End Time of Simulation
TCOM	- Communication Time
COMINT	- Communication interval
H	- Current step size
HINIT	- Initial step size
HMIN	- Minimum allowable step size
HBOUND	- Upper bound on initial step size
N	- Dimension of vector
Y	- Vector of state variables
Yp	- Predicted values of state variables
Yc	- Corrected values of state variables
Fn-1..Fn-3	- Values of three previous state derivatives
ERR(N)	- Vector of local error estimates for all state variables
ERRMAX	- Maximum local error estimate
TOL	- Integration error tolerance
NFLAG	- = 0 TCOM has not been detected = 1 TCOM has been detected
NSTEPS	- Number of steps for stiffness check
NCHECK	- Actual number of steps taken during simulation

Figure 4.10 List of Variables Used in Fourth-Order Adams Method

```
Subroutine Adams Predictor-Corrector Method
Compute initial step size HINIT
    HINIT = min(HBOUND, ((abs(0.5*TOL/FDER)) ** 0.5)/4)
Determine number of steps for stiffness check:
    NSTEPS = 10 * (TFIN - TINIT)/ HINIT
Set NCHECK = 0
Compute starting values using RK4 method
While (T < TFIN)
    While (T < TCOM)
        If (((TCOM - T) <= (H + HMIN)) AND (NFLAG = 0))
            Then | H = (TCOM - T)/2
                 | Set NFLAG = 1   (Communication time detected)
        End If
        Predict new states at next step using Adams-Basforth :
            Yp = Y + (H/24)*(55*(Fn)-59*(Fn-1)+37*(Fn-2)-9*(Fn-3))
        Compute corrected values of states using Adams-Moulton :
            Yc = Y + (H/24)*(9*(Fn+1)+19*(Fn)-5*(Fn-1)+(Fn-2))
        Compute local error estimate of all state variables :
            E = |Yc - Yp| / (10 * H)
        Determine the maximum local error estimate: ERRMAX
        If (ERRMAX > TOL)
            Then |                                  Reduce Step Size
                 | Perform step size control  (as in Figure 4.9)
                 | Reject computed state values
                 | Recompute starting values using RK4
                 | NCHECK = NCHECK + 3
            Else | Update time: T = T + H
                 | Store values of new states
                 | NCHECK = NCHECK + 1
        End If
        If ((ERRMAX <= (0.1*TOL))
            Then |                                  Augment Step Size
                 | Perform step size control   (as in Figure 4.9)
                 | If ((T + (3*H)) <= TCOM)
                 |     Then | Recompute starting values using
                 |          |   RK4
                 |          | NCHECK = NCHECK + 3
                 |     Else | H = HPREV
                 | End If
        End If
        If (NCHECK > NSTEPS)            (Perform stiffness check)
            Then | Alert user that model stiffness is suspected
        End If
    End While
    Call OUTPUT(Y,N,T)
    Compute next TCOM :
        TCOM = TCOM + COMINT
    If ((TCOM >= TFIN) .AND. (T NOT = TFIN))
        Then | TCOM = TFIN
    End If
    Set NFLAG = 0
End While
Call OUTPUT(Y,N,T)
End Subroutine Adams Predictor-Corrector Method
```

Figure 4.11. Details of The Adams Predictor Corrector Method

4.2.3.3 Program Structure

Figure 4.12 shows the pictorial view of a program using the Adams Predictor Corrector Method to drive the simulation of a model.

Main Program

This is very similar to the main program for the Runge-Kutta Method. However, it makes CALLS to the Adams subroutine to carry out the run-time activities for the model.

Subroutine INIT

This subroutine is linked to the MAIN program and utilizes the XMENU software within the IBM environment to allow the user to interactively initialize the appropriate model parameters, initial values, and other experimentation variables.

Subroutine ADAMS

This subroutine contains the integration algorithm for the Adams Predictor Corrector Method. It is called from the main program to carry out the run-time activities. In carrying out the integration, it makes subsequent CALLS to the Runge-Kutta starter subroutine as necessary, and to the ADCOMP subroutine.

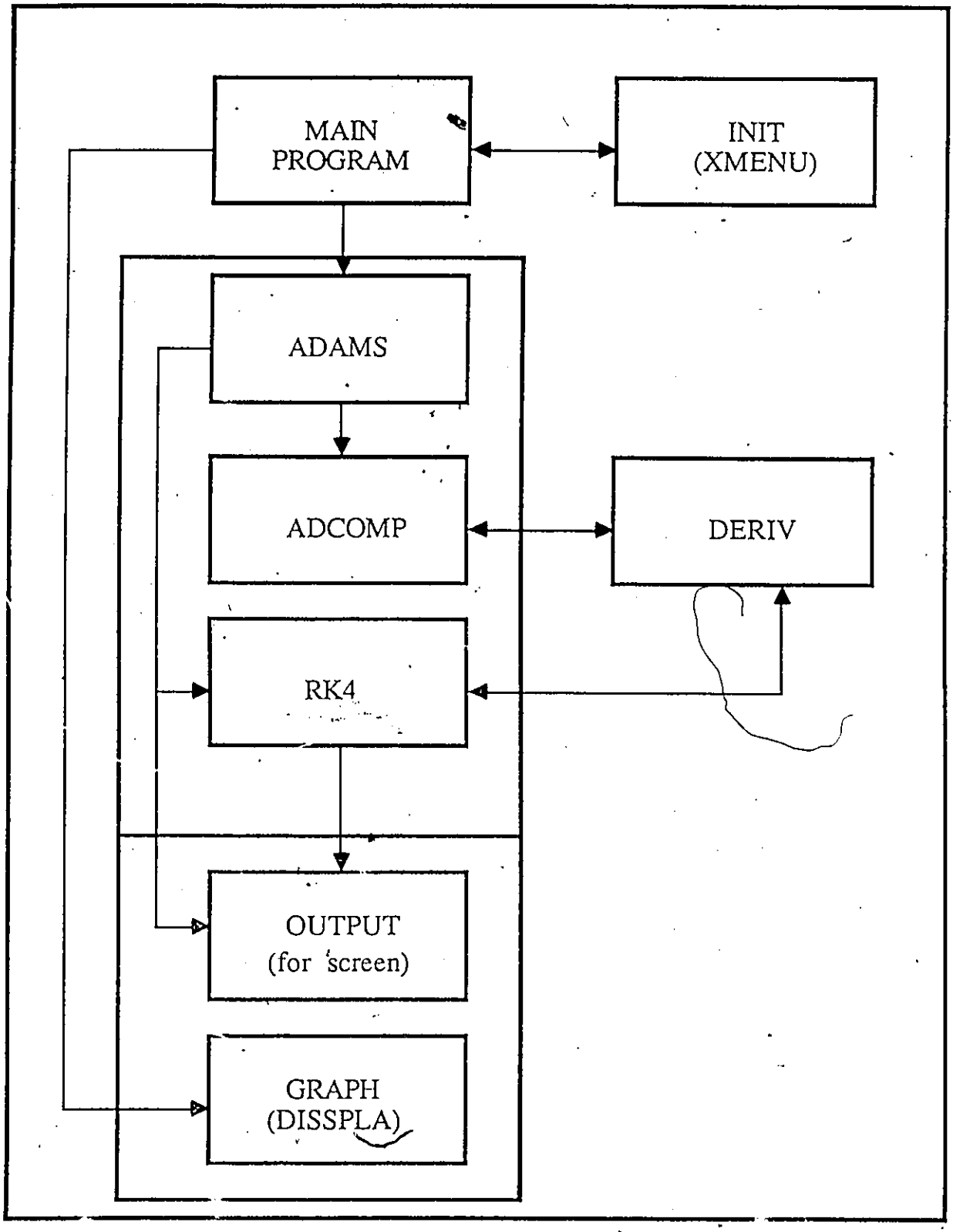


Figure 4.12 Modules of an Interactive Procedural Program using Adams Method

Subroutine RK4

This subroutine uses the Runge-Kutta algorithm as a starter to determine the three previous function evaluations needed by the Adams Method. It is invoked at the beginning of the simulation from the subroutine Adams and at each step size change. It makes CALLS at each stage of the integration to the DERIV subroutine to determine the values of the derivatives of the states.

Subroutine DERIV

This is a model dependent routine containing the necessary functions and information to carry out the computational steps as explained in section 4.2.1.3. It is used by the starter RK 4 subroutine and the subroutine ADCOMP.

Subroutine ADCOMP

This subroutine performs the calculations of the predictor and corrector formulas for the Adams Method. It utilizes the subroutine DERIV in carrying out its calculations.

Subroutine OUTPUT

The values of the outputs at each communication time are stored and an interactive tabular display is shown to the user during the simulation run. This subroutine is called

from the subroutine ADAMS.

Subroutine GRAPH

This subroutine linked to the MAIN program utilizes the Disspla software within the IBM environment to provide post-run hard copy graphic displays of the desired model outputs with time.

The variable step Fourth Order Adams method is imbedded within the Rudder Control model in Appendix C.

4.3. MULTI-RATE INTEGRATION ALGORITHM

4.3.1. Introduction

This particular type of integration algorithm has been included in the library to deal specifically with the simulation of large dynamic systems which involve relatively few interconnections. This essentially means that relatively few of the outputs (usually state variables in the absence of an output function) of some component system are inputs to some component system (Birta 1980). In practice, large dynamic systems, such as those in many chemical and petroleum processes, require frequent experimentation with the initial value problem that describes the process. As a result, what the integration does is to reduce the CPU time required by the process especially in cases where a system having widely different

characteristics can be partitioned into two or more component systems with similar characteristics. This is done without sacrificing the overall accuracy of the solution (Palusinski 1985).

The conventional approach to solving a large system consisting of component systems is simply to group together all the equations describing the component systems into one subroutine and then solve them as a single collection of differential equations. However, using this approach in a system comprising of two component systems having varied frequency components would involve significant computational inefficiency. This would arise, since the integration would proceed at a rate totally dependent on the most severe conditions experienced amongst the component systems. This occurs when the maximum local error estimate of the state variables provided by the integration formula exceeds the user's specified tolerance. As a result, some component systems which could have been integrated using a larger step size are unable to do so.

Birta (1980) has considered this problem along with a particular implementation of a quasi-parallel method. In addition, he suggests and formulates possible measures for evaluating the computational advantages of such a method. Finally, he provides results of numerical experiments that provide insight into the question of trade-offs between accuracy and computational gain.

The quasi-parallel method allows the solution to each component system to move forward independently in time. It does this by using an interpolation/extrapolation scheme based on a cubic polynomial to compute the interconnection variables when they are required by the integration of a particular component system. The coefficients of the polynomial are determined from data at two previously computed time points on the trajectory of the interconnecting variable. In addition, a strategy is used to sequence the advancement of the various component systems forward in time so as to minimize extrapolation errors involved in handling the interconnecting variables.

The conclusion of the study using the quasi-parallel method was that its effectiveness was limited to the case where not more than three interconnecting variables were involved. The method's accuracy was also found to be dependent on the order of the integration method and the order of the interpolation being the same. This is again a limiting factor since the computational requirement of the interpolation procedure increases significantly with the order. Even for a third order method whose accuracy is not very high, the cubic polynomial interpolation already has significant computational requirements. Hence, the use of such a method as a potentially efficient way of handling the simulation of large dynamic systems may be hard to justify especially if high order integration methods are to be employed.

Brosilow (1985) suggests a modular integration technique. This allows computational speed through the parallel processing of individual component systems and allows the best integration methods to be used for each component system. Independent error control used in the individual dynamic simulators ensures that the proper step size is taken in each component system. Hence the efficiency of the overall simulation is not adversely influenced by the step size in any single component system. Several coordination algorithms including a new Newton type algorithm are suggested to deal with the interconnection problem and to keep the various component systems in time synchronization. These coordination algorithms have essentially three phases after initialization:

1) Estimation of error,

2) Adjustment of time horizon or interconnection variables, and

3) Computation of interconnection variables for the next time horizon.

The conclusion of such a method is that modular integration methods are robust and efficient particularly supported by the theoretical analysis of the stability and convergence properties of the methods as presented by Brosilow (1985). As indicated previously, the software for such an algorithm, is developed on parallel processors.

4.3.2. DESCRIPTION OF THE ALGORITHM

The algorithm implemented does not involve parallel processing and hence is based on the method by Palusinski (1985) for the integration of two time scale systems. The large dynamic system is characterized by the following initial value problem:

$$Z' = p(Z,T);\ Z = Z(0)$$

$T \in [0, T_f]$ where vector Z is composed of rapidly varying components X, as well as slowly varying components Y over the interval T where dim Z = dim X + dim Y. It is assumed that the single large dynamic system can be partitioned into two individual component systems which are not independent of one another, one a so called fast component system and the other a slow component system with specified initial conditions as follows:-

$$X' = F(X,T,Y) \quad \text{- fast component system}$$

$$Y' = G(Y,T,X) \quad \text{- slow component system}$$

$$X_0 = X(0)$$

$$Y_0 = Y(0)$$

In general, coupling exists between the component systems such that the outputs of one component system may be inputs to another. In integrating the resulting two time scale systems, two routines are utilized. The Slow Routine is used to integrate the slow component system while the Fast Routine is used in the fast component system integration.

Each component system is integrated independently using different step sizes and independent error control. The explicit Runge Kutta Merson algorithm is used in the integration of both the Slow and Fast Routines. The formula used by the Runge-Kutta Merson algorithm for the step Y_j to Y_{j+1} is :

$$K1 = H * F(T,Y),$$

$$K2 = H * F(T + H/3, Y + K1/3),$$

$$K3 = H * F(T + H/3, Y + K1/6 + K2/6),$$

$$K4 = H * F(T + H/2, Y + K1/8 + (3/8)*K3),$$

$$K5 = H * F(T + H, Y + K1/2 - (3/2)*K3 + 2*K4);$$

$$Y_{j+1} = Y_j + (K1 + 4*K4 + K5)/6$$

$$E = H * abs(K1/3 - (3/2)*K3 + (4/3)*K4 - K5/6)$$

The underlying idea in the multi-rate integration approach is to carry out the following steps. The Slow Routine firstly computes the values of the slow state variables at each step size of the integration. It does this by utilizing imbedded Runge-Kutta schemes to compute each K1...K5 at each stage as depicted by the above formula.

The imbedded formulas for computing intermediate approximations using explicit Runge Kutta schemes are (Palusinski 1986, P. 273):

1) First-Order formula for computing the approximation of the solution Y at $T = T_j + H/3$ also called Euler formula is:

$$Y_{j+1/3} = Y_j + H * K1/3 \quad ;$$

$$K1 = F(Y_j , T_j , X_j)$$

2) Second-Order formula for computing the approximation of Y at the same time also called improved Euler formula defined as:

$$Y_{j+1/3} = Y_j + H * (K1 + K2)/6 \quad ;$$

$$K1 = F(Y_j , T_j , X_j)$$
$$K2 = F(Y_j + H * K1/3, T_j + H/3, X_{j+1/3})$$

3) Third-Order formula for computing the approximation of Y at T + H/2 is obtained from the third order formula (Lambert 1973) by taking half of the step and has the form:

$$Y_{j+1/2} = Y_j + H * (K1 + 3*K3)/8$$

$$K1 = F(Y_j , T_j , X_j)$$

$$K2 = F(Y_j + H * K1/3, T_j + H/3, X_{j+1/3})$$

$$K3 = F(Y_j + H*(K1 + K2)/6, T_j + H/3, X_{j+1/3})$$

Additionally, the Fast Routine integration uses the conventional Runge Kutta Merson Method to compute the values of the fast system variables over time horizons which are multiples of the Slow Routine step size during each step size of the Slow Routine, in particular at H/3, H/2 and H. Depending on the coupling resulting from the system partition, interpolation/extrapolation techniques are used to determine values of interconnecting Slow Routine state variables Y necessary for the Fast Routine integration at

each step size of the integration. The following interpolation/extrapolation scheme is used in the integration algorithm at the different stages of each integration step:

Linear Interpolation:

1) $Y(T) = Y_j + (T - T_j)$;

$$T_j <= T <= T_j + H/3$$

2) $Y(T) = Y_j + H/3 + (T - T_j + H/3)*(-K1/4-K2+(9/4)*K3)$;

$$T_j + H/3 <= T <= T_j + H/2$$

Extrapolation using Euler Method:

$Y(T) = Y_j + H/2 + (T-T_j + H/2)*K4$;

$$T_j + H/2 <= T <= T_j + H$$

4.3.3 Structured Algorithm

Figures 4.13(a),(b), and 4.14 show the details of the multi-rate integration algorithm used in the run-time library.

List of variables for Multi-Rate Integration Algorithm

T	- Computational Time
TINIT	- Initial Time of Simulation
TFIN	- End Time of Simulation
TCOM	- Communication Time
THOR	- Next Horizon Time
COMINT	- Communication interval
H	- Current step size
HINIT	- Initial step size
HSMIN	- Minimum allowable step size for slow system
HFMIN	- Minimum allowable step size for fast system
HBOUND	- Upper bound on initial step size
Y	- Vector of slow system state variables
X	- Vector of fast system state variables
N	- Dimension of slow system state vector
M	- Dimension of fast system state vector
ERR(N)	- Vector of local error estimates for all slow system state variables
ERRMAX	- Maximum local error estimate of slow system state variables
ERF(M)	- Vector of local error estimates for all fast system state variables
ERFMAX	- Maximum local error estimate of fast system state variables
TOLERS	- Slow system error tolerance
TOLERF	- Fast system error tolerance
FDER	- Maximum value of derivative functions at initial time
NFLAG	- = 0 TCOM has not been detected = 1 TCOM has been detected

Figure 4.13(a) List of Variables Used in the Multi-Rate Algorithm

```
Subroutine Multi-Rate Algorithm
                                        Slow Routine- Modified RK Merson
Set NFLAG = 0                           (TCOM has not been detected)
Compute initial step size
    HINIT = min(HBOUND, ((abs(0.5*TOL/FDER)) ** 0.5)/4)
Compute initial TCOM
While (T < TFIN)
    While (T < TCOM)
        If (((TCOM - T) <= (HINIT + HMIN)) AND (NFLAG = 0))
            Then | H = (TCOM - T)/2
                 | NFLAG = 1          (Communication time detected)
        End If
        Compute K1:                     Y  - Slow system variables
                                                 at T
             K1 = DERIVS(Y, T, X)       X  - Fast system variables
                                                 at T
        Use Imbedded Euler Method to predict
            Y  at T + H/3 :
               Y + H/3(p) = Y +  (H/3)*K1
        CALL Fast  Routine to determine X + H/3
        Compute K2:
               K2 =  DERIVS(Y + H/3(p), T + H/3, X + H/3)
        Use Imbedded Improved Euler Method to
            correct Y  at T + H/3:
               Y + H/3 = Y + (H/6)*(K1-K2)
        Compute K3:
               K3 = DERIVS(Y + H/3, T + H/3,X + H/3)
        Use Imbedded third order RK formula to estimate
            Y  at T + H/2:
               Y + H/2 = Y + (H/8)*(K1-3*K3)
        CALL Fast  Routine to determine X + H/2
        Compute K4:
            K4 = DERIVS(Y + H/8*(K1-3*K3, T-H/2, X + H/2)
        CALL Fast  Routine to determine X  + H
        Compute K5:
            K5 = DERIVS(Y + H/2)*K1 - 3*(H/2)*K3 + 2*H*K4,
                        T + H, X + H)
        Compute local error estimates of states :ERR(N)
        Determine maximum local error estimate : ERRMAX
        If (ERRMAX <= TOLERS)
            Then | Update Time: T = T + H
                 | Compute new Y at T
        End If
        If ((NFLAG = 0)
            Then | Perform step size control (as in Figure 4.8)
        End If
    End While
    CALL OUTPUT(Y, N, T)
    Compute next TCOM :
                          TCOM = TCOM + COMINT
    If ((TCOM >= TFIN)
        Then | TCOM = TFIN
    End If
    Set NFLAG = 0
End While
CALL OUTPUT(Y, N, T)
End Subroutine Multi-Rate Algorithm
```

Figure 4.13.(b) Details of Slow Routine of Multi-Rate Algorithm

```
Subroutine Fast Routine                                    (RK-Merson Method)
Compute initial step size
    HINIT = min(HBOUND, ((abs(0.5*TOL/FDER)) ** 0.5)/4)
While (T <= THOR)
    If (THOR    = T + H/3)
        Then | Linearly interpolate for values of input variables
             | from the slow system between [T, T + H/3]
        ElseIf (THOR  =   T + H/2)
            Then | Linearly interpolate for values of input
                 | variables from the slow system between
                 | [T, T + H/2]
        ElseIf (THOR  = T + H)
            Then | Extrapolate utilizing Euler's Method for
                 | values of input variables from the slow system
                 | between [T + H/2, T + H]
    EndIf
    Compute state derivatives K1..K5 for each state variable
    Compute local error estimates of all state variables: ERF(M)
    Determine the maximum local error estimate : ERFMAX
    If (ERFMAX <= TOLERF)
        Then | Update Time: T = T + H
             | Compute new states at T
    End If
    Perform step size control                    (as in Figure 4.8)
End While
End Subroutine Fast Routine
```

Figure 4.14 Details of Fast Routine of Multi-Rate Algorithm

4.3.4 Program Structure

Figure 4.15 shows the pictorial view of a program using the multi-rate integration method to drive the simulation of a model.

Main Program

This is very similar to the main program for the Runge-Kutta Method. However, it calls to the subroutine SRATE to carry out the run-time activities for the model.

Subroutine INIT

This subroutine is linked to the MAIN program and utilizes the XMENU software within the IBM environment to allow the user to interactively initialize the appropriate model parameters, initial values, and other experimentation variables.

Subroutine SRATE

This subroutine uses the variable step Runge Kutta Merson method to integrate the slow routine during the simulation. However, the values of the state variables at the different stages of each step size are computed using imbedded formulas for computing intermediate approximations using explicit Runge-Kutta schemes. These formulas use the subroutine DERIVS in their computations.

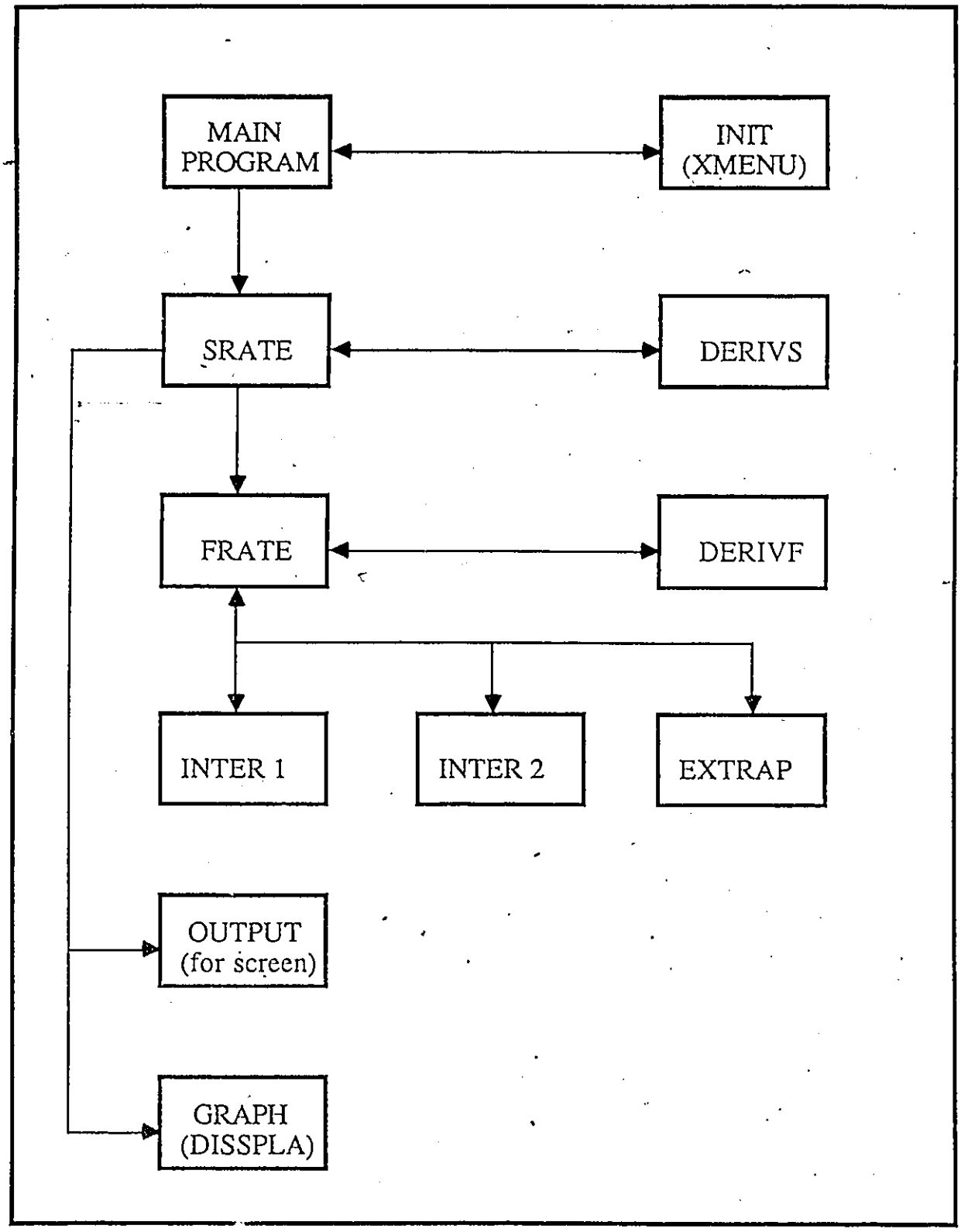


Figure 4.15 Modules of an Interactive Procedural Simulation Program using Multi-Rate Algorithm

Subroutine FRATE

Called by the subroutine SRATE, it uses the normal variable step Runge-Kutta Merson to integrate the fast routine during the simulation using the subroutine DERIVF in its computations.

Subroutine INTER1

This subroutine determines the value of the slow system interconnection variables by linear interpolation during the fast routine integration between T_j and $T_j + H/3$

Subroutine INTER2

This subroutine determines the value of the slow system interconnection variables by linear interpolation during the Slave Routine integration between $T_j + H/3$ and $T_j + H/2$

Subroutine EXTRAP

This subroutine determines the value of the slow system interconnection variables by extrapolation using Euler's Method during the fast routine integration between $T_j + H/3$ and $T_j + H/2$

Subroutine DERIVS

This is a model dependent subroutine and contains the necessary functions and information to carry out the computational steps needed for the integration of the slow

routine.

Subroutine DERIVF

This is a model dependent subroutine and contains the necessary functions and information to carry out the computational steps needed for the integration of the Fast Routine.

Subroutine OUTPUT

The values of the outputs at each communication time are stored and an interactive tabular display is shown to the user during the simulation run. This subroutine is called from the subroutine SRATE.

Subroutine GRAPH

This subroutine linked to the MAIN program utilizes the Disspla software within the IBM environment to provide post-run hard copy graphic displays of the desired model outputs with time.

The multi-rate integration algorithm is imbedded within the Electronic Oscillator Model in Appendix G.

CHAPTER 5

GEST RUN-TIME LIBRARY - SECTION 2

5.1 THE DISCONTINUITY PROBLEM

It is quite common to encounter discontinuities in the simulation of continuous systems and it is one of the main problem areas in the solving of ordinary differential equations. In general, the continuous simulation model will be comprised of the set of ordinary differential equations having the form

$$Y' = F(T, Y(T))$$

where Y represents the state vector (Y1,...Yn) and each F represents the corresponding system derivative function.

The types of discontinuities associated with such systems can be classified as derivative discontinuities and jump discontinuities (Ören 1987d). Derivative discontinuities correspond to a model update as shown in Figure 5.1. This update may involve a model change which can take place by replacing the function which specifies the derivative of a state variable or by merely updating the value of one or more parameters which appear in the derivative function. Jump derivatives refer to the reinitialization of a state variable and correspond to a sudden change of a state variable as shown in Figure 5.2. Finally, it should be further realized that both jump and derivative discontinuities can take place simultaneously as shown in

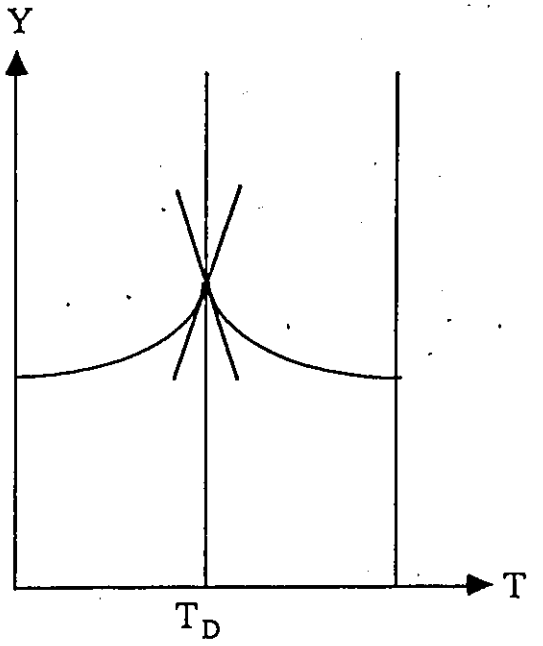


Figure 5.1 Derivative Discontinuity

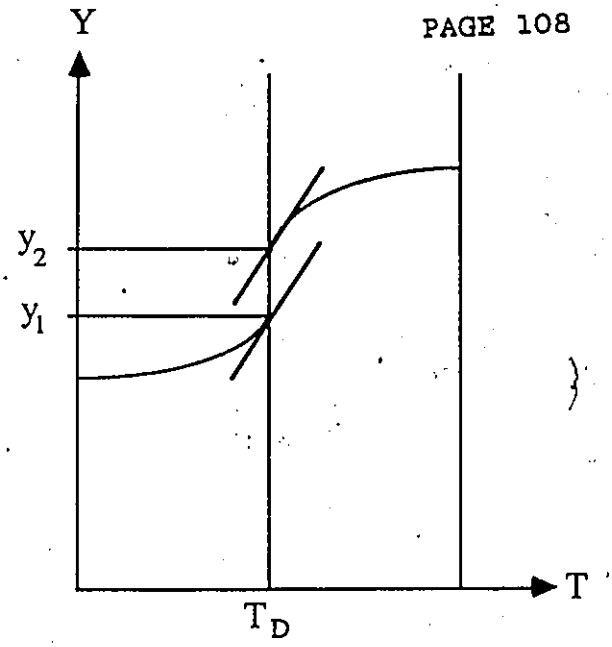


Figure 5.2 Jump Discontinuity

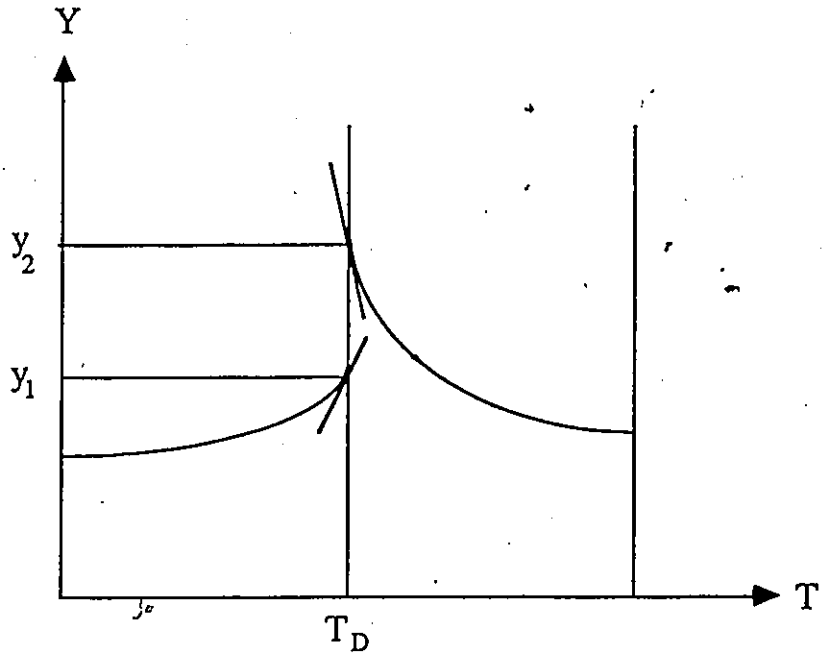


Figure 5.3 Jump and Derivative Discontinuity

Figure 5.3.

In the literature (Cellier 1979), from the implementation point of view, two types of discontinuity processing are discussed. These consist of 1) time events and 2) state events. If the discontinuity time is known a priori, one can schedule a time event at that point. In the processing of a time event, the step size simply has to be reduced to the scheduled event time. When the integration has reached the event time, depending on the nature of the discontinuity i.e. derivative or jump discontinuity, the model update or state variable reinitialization can be performed. In the case of state events, time is not known a priori, and they are triggered once some pre-specified state condition is met. An example of such a condition could be when a state variable reaches some threshold value.

In the processing of state event discontinuities, the conditions which determine the changes in the model can be expressed as a set of discontinuity functions or as more commonly referred to in the literature switching functions S_i (T,Y(T)). i = 1...m such that an actual model change occurs when one of the

$$S_i (T,Y(T)) = 0$$

Thus it can be clearly seen that the discontinuity processing problem for state events reduces to firstly detecting the zeros of the discontinuity functions using an appropriate detection scheme. Upon detection, the unknown

event times must be accurately located using an iteration procedure inherent in the procedure. Once located, the step size will be adjusted so that the integration proceeds right up to the point of discontinuity. At that point, the appropriate model updates and/or reinitialization of state variables as determined by the discontinuity function(s) are triggered.

5.2 REVIEW OF PREVIOUS WORKS

A number of methods have been developed to deal with this problem especially since it was clear that standard ordinary differential equation solvers were both inadequate and inefficient. The early solvers dealt with the discontinuities by estimating the local truncation error. Based on the error, the integration would either continue or the step size would be halved and the integration step repeated. This causes excessive creeping around the discontinuities (Ellison 1981)

The most common methods involve the use of switching functions, although few such as Gear and Osterby (1984) look at processing discontinuities in situations where external indicators such as pre-defined switching functions, are not available. In this method, the magnitude and order of the discontinuity is estimated based on examining the local truncation error and its estimate. The method involves a predictor-corrector integration process. However, in our discussion, we concentrate on the methods where switching functions are known.

Carver's method (Carver 1978) for detecting and processing discontinuities represents them as switching functions, as mentioned earlier, which are triggered when particular conditions are fulfilled. The switching functions are differentiated by the user and attached to the

system of equations as an auxiliary set. Also, scaled derivatives up to the order of the integration q are stored so that the prediction of Y at time T + H is given by a set of equations. From these equations, an accurate estimate of the discontinuities eliminates the creeping mentioned above. The detection of the crossover points where the switching function passes through zero results in determining an H1 such that the set of equations mentioned earlier is zero in the neighbourhood of S(T). This reduces to the problem of solving a qth order polynomial for S(H1). However, a simpler criterion involving the product of the switching functions and their derivative at the interval points is usually adequate.

Cellier (1979) introduces a distinction for continuous systems possessing discontinuities and those that do not. The following definitions clarify this distinction:

" Combined systems are systems described, either during the whole period under investigation or during a part of it, by a fixed or variable set of differential equations where at least one state variable or one state derivative is not continuous over a simulation run. "

(Cellier 1979, P. 17)

" Continuous systems are systems described by a fixed set of differential equations with state variables and first state derivatives both being continuous over the whole simulation run. "

(Cellier 1979,P. 17)

Cellier considers the two types of discontinuities discussed earlier and conveys the fact that continuous simulation has to be performed either up to the next scheduled event time or until a state condition is met which triggers execution of a state event, whichever comes first. His detection criterion involves realizing the integration step where a discontinuity function changes its sign for the first time . Within this step, it is clear that a state event has taken place. In the location of the discontinuity, he suggests that although any good iteration procedure can solve the problem, an iteration procedure with a quadratic or cubic convergence ratio should be used so that the iteration procedure could converge as soon as possible for efficiency. In addition, for software robustness requirements, the procedure should have an infinite convergence range. For these above reasons, he proposes a combination of the inverse Hermite Interpolation (fast convergence) with Regula-Falsi (unlimited convergence range) methods.

Ellison (1981) uses a similar approach to the problem by classifying types of discontinuities as time and state events. His method claims that time events can be immediately located, while the state events require one inverse interpolation. With this breakdown of events the

inefficiency produced by other procedures (e.g. Carver 1978) for systems which involve time events only or mixed time and state events is eliminated.

In the integration of the system where time and state events both occur separately, we will know the time of the 'time' event but cannot advance the step until it has been concluded that no state event occurs before this. If a particular condition for the state event has been met, the time of occurrence dictates the new integration step. The location of this time is determined as accurately as the tolerance designated by the user. The method is an adaption of the Shampine method and uses a fairly simple threshold detection criterion for detecting the discontinuity. Additionally, it utilizes the inverse third order Hermite interpretation procedure requiring values of y and y' at t_j and t_{j+1} for location of the discontinuity.

Birta et.al. (1985) utilize a robust procedure for dealing with the problem. The procedure assumes the presence of state events only. It considerably improves on the detection criterion suggested by Carver and others. The criterion is based on a cubic polynomial model requiring values of the switch functions and their derivatives at the interval boundaries. The only real limitation occurring if three stationary points exist in the interval. The location problem is dealt within a two stage procedure. In the first stage, the regula falsi is first carried out for all

switching functions identified as having a cross over in the interval. The evaluation of Si at points in the interval as required by the process is done using estimated x values obtained from a fifth-order Hermite Interpretation. The second stage uses the Newton Raphson method to refine the results produced by the first stage.

5.3 PROPOSED INTEGRATION ALGORITHM

The method chosen for use in the run-time library is a modified version of the adaptive order algorithm developed by Ören and Ma (1986). Essentially, it incorporates the breakdown of events as illustrated by Cellier and Ellison and discussed earlier with the adaptive order algorithm. The detection criterion of the algorithm utilizes the values of the discontinuity functions at the boundaries of each interval whilst the location of the discontinuity is found by using an inverse aitken interpolation method with variable order. The algorithm is further enhanced by its ability to deal with systems which involve time events only, state events only, or mixed time and state events. The following advantages provide a good base as to why this algorithm was chosen for use in the run-time library:

1) The fact that the code does not require the derivatives of the switching functions makes the code simpler and less of a burden to the modeler.

2) The procedure utilizes inverse interpolation, the

robustness of which depends on the existence of the inverse function. From a mathematical viewpoint, (Ma and Liu 1986) one need only check the monotonicity of the switch function over the particular integration interval to guarantee this existence. The fact that the software requirements could be considered on a strict mathematical base further enhances the choice of the method as the discontinuity processing algorithm in the library.

3) The order of the interpolation being variable and dependent on the tolerance specified by the user is a very useful component since unnecessary computations will not be carried out. This is so since as the order increases, the number of components of the aitken's table increases.

4) The performance of the algorithm is very good since the convergence rate increases as the order of the interpolation increases.

5) The main benefit of using an inverse interpolation should be realized. A unique solution exists for the time at which the interpolation polynomial is zero. In addition, the process will be quicker than if the direct polynomial method is used which may have several solutions to choose the correct one from.

6) The efficiency achieved by the algorithm being able to handle systems involving time events only, state events only, or mixed state and time events is clearly advantageous over other methods.

7) The iterative process of the aitken's method makes it a very useful method to implement on the computer.

The main disadvantage of the algorithm exists in its detection scheme. The detection criterion involving the product of the values of the switch functions at the interval boundaries may not always be totally reliable, for example, in cases where an even number of cross-overs occur in the same step interval as Figure 5.4 illustrates. In such cases, the algorithm has no chance to detect the crossovers. Such discontinuities are labelled short-living discontinuity functions (Cellier 1979). However, as Gear and Osterby (1984) point out, the detection of a discontinuity should involve a very low cost set of tests since they have to be activated, as in our case, at each step of the integration routine. Additionally, to circumvent this problem the size of the integration step has been restricted to a value HMAX. In keeping the step size relatively small, the probability of missing a discontinuity will be smaller.

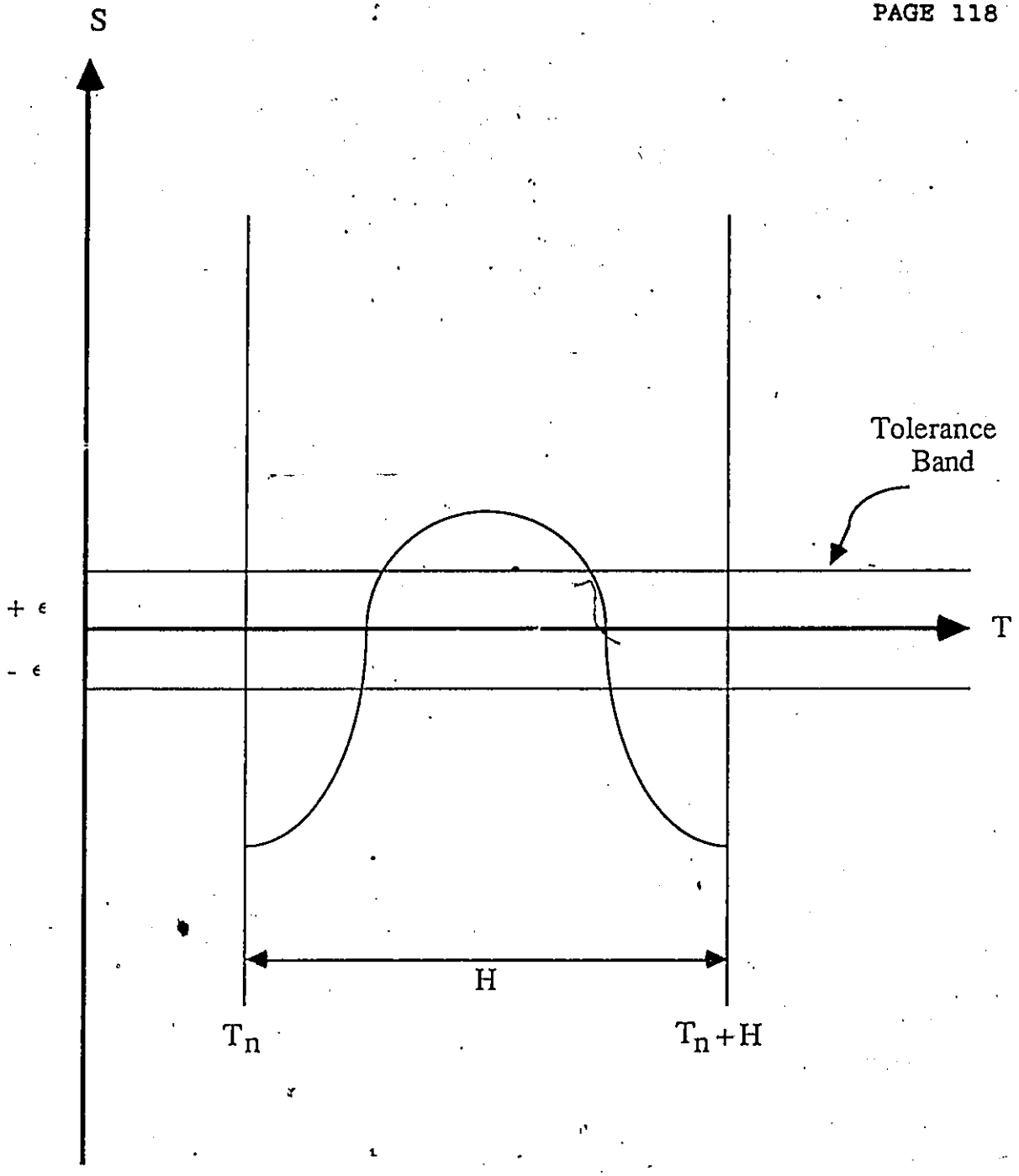


Figure 5.4 Even Number (2) of Cross overs in a Step Size

5.3.1 Details of Algorithm

As discussed previously, the algorithm effectively deals with systems having both derivative and jump discontinuities expressed as time and state conditions. Thus, for the sake of clarity, we discuss how the algorithm deals with systems having each of the different types of events.

5.3.1.1 Time Events

In the situation where the system exhibits time events only, the algorithm is somewhat simplified. The number of time events and their exact locations are easily predetermined by the user, who then inputs them to the algorithm. The time values are input in the form of a vector in ascending order of their occurrence. The normal routine using the variable step Runge-Kutta Merson is used to drive the simulation until it is evident that in the next integration step the time of the event will have been surpassed.

At this point, the step size of the integration is adjusted and the Modified Runge-Kutta Merson method is utilized to step right up to the discontinuity point. The model is then updated as specified by the time event. This information is given in a user-supplied subroutine called MODUPD. The process is repeated for the next time event if one exists or if not, the normal Runge-Kutta Merson

integration routine drives the simulation to the end time TFIN. The integration subroutines are discussed later in this section. The step size control is identical to the one used in the Runge- Kutta Fehlberg Method.

5.3.1.2 State Events

In the case where state event discontinuities only are known to exist, the values of the discontinuity functions are calculated at the end of each integration step. A cross over is detected once the product of the previous and present values of a discontinuity function is found to be negative as shown in Figure 5.5.

At this point, the algorithm starts its search to determine the precise location of the crossover within the user's tolerance. This reduces to determining the time of the zero of the corresponding discontinuity function S(T,Y(T)) = 0. Using the inverse interpolation method, we simply change the two axes which are 'value of discontinuity function' and 'time' as illustrated by Figures 5.6 and 5.7. Mathematically, the inverse function could be written as T = T(S). At this point, the time T can be approximated by an n degree polynomial, T being available at n corresponding points of S. Thus, using the known values of n points and the Aitken's interpolation formula we can derive T which is corresponding to S = 0.

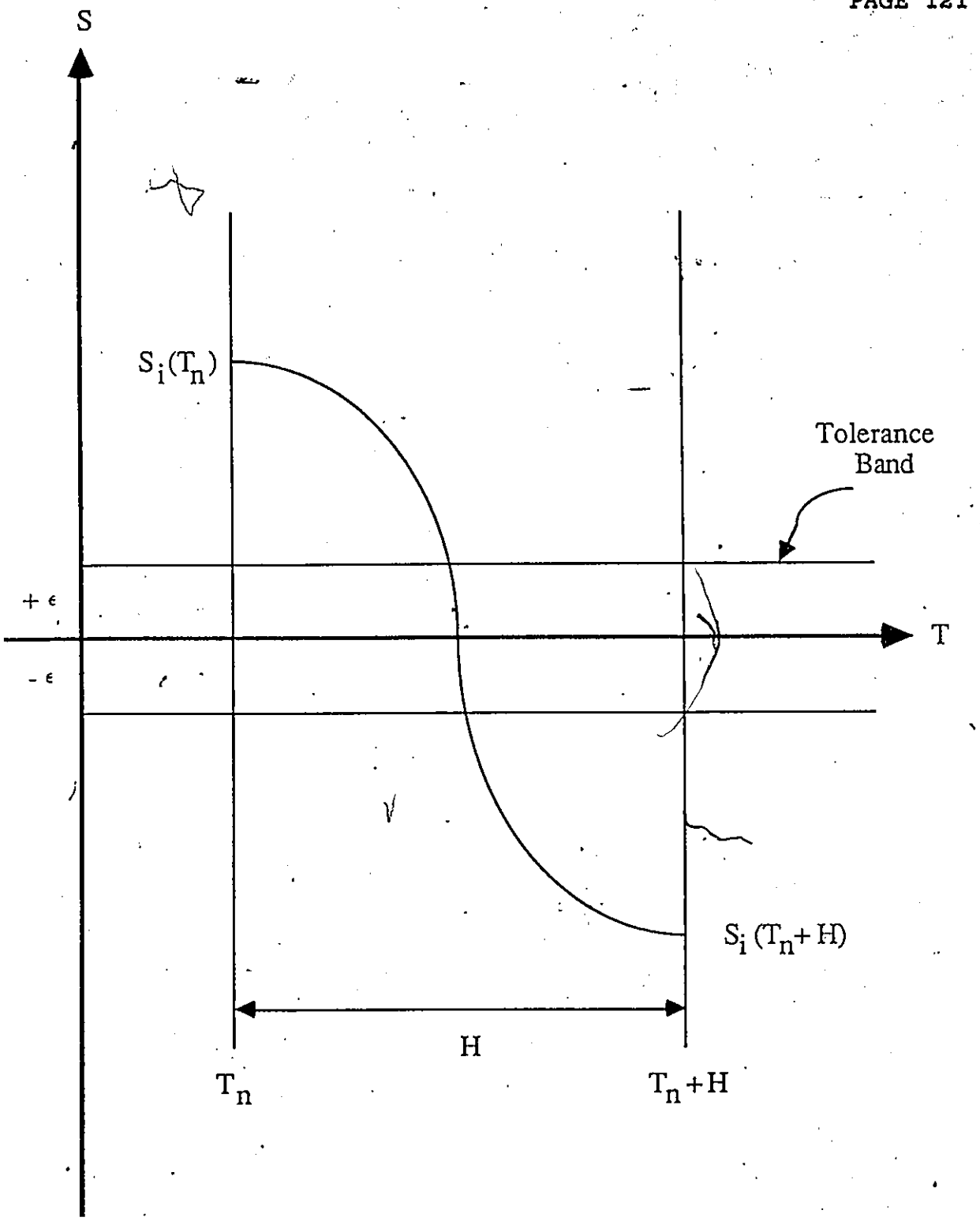


Figure 5.5 Cross-Over Detected in this Step Size

T	T_1	T_2 . . .	T_n
S(T , Y(T))	$S(T_1)$	$S(T_2)$ . . .	$S(T_n)$

Figure 5.6 Initial Interpolation Table

S (T)	$S(T_1)$	$S(T_2)$ . . .	$S(T_n)$
T =T(S)	T_1	T_2	T_n

Figure 5.7 Interpolation Table With Axes Changed

The adaptive algorithm will start with linear interpolation using the previous and current values of the discontinuity function. The first approximation of the time T(2,2) can be calculated from the linear interpolation formula:

$$T(2,2) = \frac{1}{F(2)-F(1)} * \begin{vmatrix} T(1,1) & F(1)-\overline{F} \\ T(2,1) & F(2)-\overline{F} \end{vmatrix}$$

where F(1)and F(2) are values of the discontinuity functions at the boundaries of the last step and F is the value of the discontinuity functions we require equal to zero.

Aitken interpolation table now becomes:

F(1)	T(1,1)	
F(2)	T(2,1)	T(2,2)

The integration step is repeated using the step size H = (T(2,2) - T(1,1)) and the new corresponding value of the discontinuity function F(3) is calculated. If the absolute value of F(3) lies within the band of tolerance, the interpolation procedure is stopped. If not, the order of interpolation is increased so that quadratic interpolation can be effected by linear interpolation over two independent linear interpolation polynomials to determine T(3,3). The table, ready for quadratic interpolation will have three points:-

F(1)	T(1,1)	
F(2)	T(2,1)	T(2,2)

F(3) T(3,1)

where T(3,1)-T(2,2). We calculate T(3,2) in a similar way to T(2,2) and then compute T(3,3) by :-

$$T(3,3) = \frac{1}{F(3)-F(2)} * \begin{vmatrix} T(2,2) & F(2)-\bar{F} \\ T(3,2) & F(3)-\bar{F} \end{vmatrix}$$

The process is repeated again, determining the new corresponding value of the discontinuity function F(4) and checking to see if it lies within the tolerance. If not, one could effect cubic interpolation by linear interpolation over two independent quadratic interpolation polynomials. This interpolative process is continued until the tolerance is met.

In the implementation of the algorithm, the iterative inverse Aitkens's interpolation formula is summed up by

FOR K=2 to N
FOR I=K to N

$$T(I,K) = \frac{1}{F(I)-F(K-1)} * \begin{vmatrix} T(K-1,K-1),(F(K-1)-DELTA) \\ T(I,K-1),(F(I)-DELTA) \end{vmatrix}$$

where DELTA = -0.5 * sign f(d(N)*TOL),
D(N) = nth value of the stored derivative functions of the last step
TOL = the user tolerance

The LELTA ensures that the S lies within the specified tolerance boundaries totally on the 'opposite' side of the axis, so that at the computed t, S will have changed sign.

Robust checks are carried out for each successive iteration of the interpolation procedure from the start. The validity of the interpolation table generated is checked. Any useless points are discarded and the table generated once more. As an example, using the table above, T(2,2) would be discarded if it was found to be greater than T(2,1) or less than T(1,1). Additionally, at each iteration, the Aitken interpolation table is rearranged according to the ascending order of the discontinuity functions. The algorithm also has a subroutine which handles multi-discontinuities and does a non-monotony check. When several discontinuities occur during an integration step, the routine considers the one which crosses the axis first. This check is repeated for each iterative cycle.

5.3.1.3 Time And State Events

In the final case, the system may have both time and state events. The number and times of the time events will be known exactly, being predetermined by the user. We will try to advance each time event but must first conclude that no state events occur before each one of them . If one or more state events are detected, we must locate their times once again using the inverse Aitken interpolation method as described previously. At each state event, the model is updated as stipulated by the event. Similarly, once a time event is reached, the model must also be updated.

Initially, as long as it has been confirmed that a time event will not occur in the next step, the normal Runge-Kutta Merson method is used to drive the simulation, the discontinuity algorithm checking at each step for signs of a state event discontinuity. If a state event is detected, the point is located and the modified Runge-Kutta Merson method used to drive the system up to that point. The model update by the state event is carried out, and then the same routine used to continue the integration of the system to the point of the time event discontinuity. Once a time event has been identified to be present in the next step, the modified Runge-Kutta method is used to integrate the system up to the point of discontinuity, as long as a state event is not detected. Model update as required by the time event is then carried out. This process is repeated for each time event interval, and then finally up to the end of the simulation.

5.3.2 Structured Algorithm

Figures 5.8 - 5.11 show the details of the discontinuity algorithm used in the run-time library. Figure 5.8 gives the list of variables while Figure 5.9 shows the global structure of the algorithm. Figure 5.10 shows the details for processing systems where time events only exist. Figure 5.11 shows the details for cases where state events or mixed time and state events occur.

List of variables for Discontinuity Processing Algorithm

T	- Computational Time
TINIT	- Initial time of Simulation
TFIN	- End Time of Simulation
TCOM	- Communication Time
THOR	- Next Horizon Time
TE	- Vector of times at which time events occur
COMINT	- Communication interval
H	- Current step size
HINIT	- Initial step size
HMIN	- Minimum allowable step size
HMAX	- Maximum allowable step size
HBOUND	- Upper bound on initial step size
N	- Dimension of vector
Y	- Vector of state variables
ERR(N)	- Vector of local error estimates for all state variables
ERRMAX	- Maximum local error estimate
TOL	- Integration error tolerance
FDER	- Maximum value of derivative functions at initial time
NTEALL	- Flag showing model has time events only
NUM	- Flag counter
NUMTE	- The number of time events existing in the model
NTIMEV	- Flag showing Time Event taking place
NTIMUP	- Flag showing time update took place

Figure 5.8 List of variables used in the Discontinuity Processing Method

```
Discontinuity Processing Algorithm
Compute initial step size : HINIT
    HINIT = min(HBOUND,  ((abs(0.5*TOL/FDER)) ** 0.5)/4)
Set H      = HINIT
Set NTIMEV = 0
Compute communication time TCOM:
    TCOM = T + COMINT
While (T < TFIN)
    Determine next time horizon THOR
    If ((NUM <= NUMTE) AND (NTIMUP = 1))
        Then | If (TE(NUM) < TCOM)
             |    Then | THOR   = TE(NUM)
             |          | NTIMUP = 0
             |    Else | THOR   = TCOM
             |         | NTIMUP = 0
             | End If
    End If
    If ((NUM > NUMTE) AND (NTIMUP = 1))
        Then | If (TFIN > TCOM)
             |    Then | THOR   = TCOM
             |          | NTIMUP = 0
             |    Else | THOR   = TFIN
             |         | NTIMUP = 0
             | End If
    End If
    While ((T + H) < THOR)        Time horizon reached in next step
        If (TEALL = 1)          Check if all events are time events
            Then | Integrate system for time events only
                 |                              (See Figure 5.9)
            Else | Integrate system for state events and/or time
                 | and state events          (See Figure 5.10)
        End If
    End While
                                        Modified Runge Kutta Merson
    HDIS = THOR - T
    H    = HDIS
    Compute state values at THOR
    Update Time : T = T + H
    Set NTIMEV = 1
    If (THOR = TE(NUM)
        Then | NTIMEV = 1
    End If
End While
Call OUTPUT(Y,N,T)
End Discontinuity Processing Algorithm
```

Figure 5.9: Global Structure of Discontinuity Processing Algorithm

```
Processing of Time Events only
If (NTIMEV = 1)                                   Time Event Reached
    Then | CALL MODUPD                            Update model as
         |                                        specified by time event
End If
                                                  Normal Runge Kutta Merson Method
Compute state derivatives K1..K5 for each state variable
Compute local error estimates for each state variable : ERR(I)
Determine the maximum local error estimate: ERRMAX
If (ERRMAX <= TOL)
    Then | Update Time : T = T + H
         | Store values of new states
End If
Perform step size control     (as for Runge Kutta Fehlberg)
End Time Events only Processing
```

Figure 5.10 Processing of Time Events only

```
Processing of State Event or Time and/or State Event
                                                        Initialization
Set NTIMEV = 0
Set NTIMUP = 0
Compute value of discontinuity function(s)
If (Integration is at first step)
  Then | Compute and store state variables at T using the normal
       |   Runge Kutta Merson Method.
       | Compute value of discontinuity function(s)
End If
Use detection criterion to check for a state event in [T, T+H]
If (No. of Iterations NOT = 1)
  Then | If (State Event not present)
       |   Then | If (NTIMEV = 1)
       |        |   Then | Call MODUPD        (Model Update
       |        |        |                     for Time Event)
       |        |        | NTIMUP = 1         (Integration at
       |        |        |                     Time Event)
       |        |   Else | Store previous state values
       |        |        | Compute state values at T + H
       |        |        |   using normal RK Merson method
       |        | End If
       |   Else | Build Aitken Interpolation Table
       |        | Locate time of discontinuity by Aitken
       |        |   Interpolation
       |        | Check validity of Interpolation Table
       |        | Compute state values at interpolation time
       |        |   using the Modified RK Merson Method
       | End If
  Else | If ((State events present) AND (Values within tolerance))
                                                  No. of Iterations = 1
       |   Then | Call MODUPD                (Model Update due
       |        |                             to State Event)
       |        | If (NTIMEV = 0)
       |        |   Then | Store previous state values
       |        |        | Compute state values at T + H
       |        |        |   using the normal RK Merson
       |        |   Else | Compute state values at Time
       |        |        |   Event using the modified RK
       |        |        |   Merson Method
       |        | End If
       |   Else |   (State event values outside tolerance band)
       |        | Prepare for higher order interpolation
       |        | Sort table values
       |        | Check for monotonicity and select crossover
       |        | Compute time of state event by Aitken
       |        |   Interpolation
       |        | Check validity of Interpolation Table
       |        | Compute states at interpolated time using
       |        |   the modified RK Merson method
       | End If
End If
End Processing of State event or Time and/or State event
```

Fig. 5.11: Processing of State events or Time and/or State events

5.3.3 Program Structure

Figure 5.12 shows the pictorial view of a program using the discontinuity processing method to drive the simulation of a model. For systems known to have time event discontinuities only, subroutines DISFUN, AITKEN, SORT, and SELECT are not utilized.

MAIN Program

This is very similar to the main program for the Runge-Kutta Method. However, it makes calls to the subroutine DISCON to carry out the run-time activities for the model.

Subroutine INIT

This subroutine is linked to the MAIN program and utilizes the XMENU software within the IBM environment to allow the user to interactively initialize the appropriate model parameters, initial values, and other experimentation variables.

Subroutine DISCON

This subroutine is linked directly to the MAIN program and contains the body of the discontinuity processing algorithm. It makes calls to other subroutines which are involved in the integration of the model.

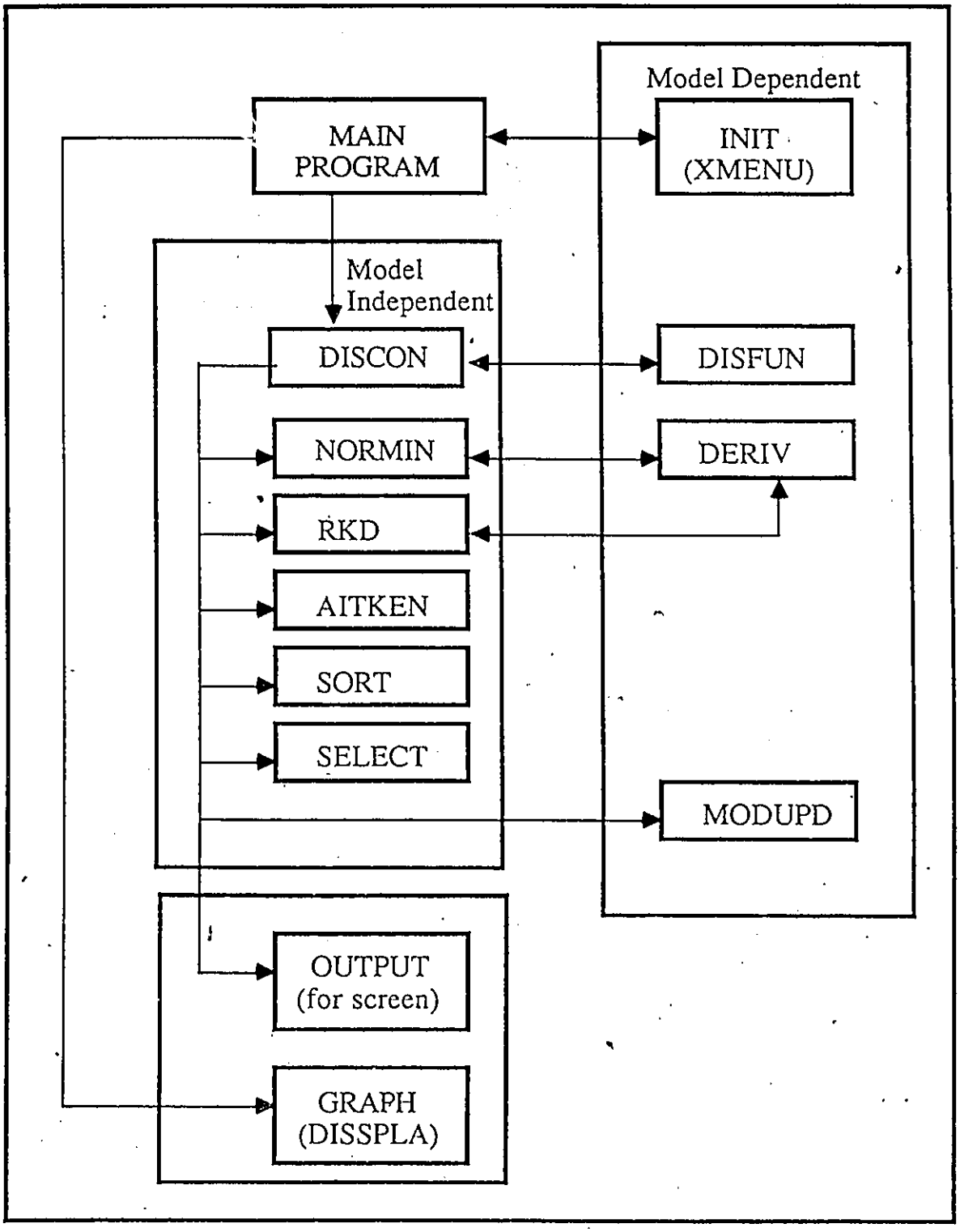


Figure 5.12 Modules of an Interactive Procedural Program using the Discontinuity Processing Algorithm

Subroutine RKD

This subroutine utilizes the variable step RK-Merson method to drive the simulation right up to the discontinuity point which has been detected in the last step. As soon as a discontinuity is detected the normal routine is stopped and the step is repeated with a new step size HDIS = THOR - T computed by the algorithm dealing with the discontinuity. This step size, assuming the estimated error does not go beyond the specified tolerance, allows us to step right up to the discontinuity point computed by the algorithm and pass it. If the error does go beyond the specified tolerance, the integration will be repeated over a multiple of subintervals, using a fraction of the step size, as is necessary, until the last step ends up exactly on the discontinuity point located by the algorithm.

Subroutine NORMIN

This subroutine utilizes the variable step RK-Merson method to drive the simulation at each step where a discontinuity has not been detected. Integration using the normal Runge-Kutta Merson algorithm continues throughout the simulation until an event discontinuity (time or state) occurs or the end of the simulation does, whichever comes first. In the former case, the normal routine is disabled and the modified routine takes over as explained above. As soon as the point of discontinuity is reached and the model updated, control is returned to the normal routine.

Subroutine MODUPD

This subroutine contains the model update information. This consists of the update of state variables or parameters and the conditions which should trigger them. It is a model independent subroutine.

Subroutine AITKEN

This subroutine is responsible for generating the interpolation table in the quest for determining the time of the state event discontinuity. The validity of the interpolation table generated is also checked and any useless points are discarded and the table generated once more.

Subroutine SORT

This subroutine is model independent. It is responsible for rearranging the interpolation table in ascending order of the discontinuity functions after each iterative interpolation.

Subroutine DISFUN

This subroutine is model dependent and contains the discontinuity functions that exist in the model. The zeros of these functions indicate when a model update should occur as a result of a state event.

Subroutine SELECT

This subroutine selects the useful points which will increase the convergence rate and discards those points which should produce absurd interpolation results . It also performs a monotonic check.

Subroutine OUTPUT

The values of the outputs at each communication time are stored and an interactive tabular display is shown to the user during the simulation run. This subroutine is called from the subroutine DISCON.

Subroutine GRAPH

This subroutine linked to the MAIN program utilizes the Disspla software within the IBM environment to provide post-run hard copy graphic displays of the desired model outputs with time.

CHAPTER 6

GEST RUN-TIME LIBRARY - SECTION 3

6.1 QUALITY ASSURANCE IN MODELLING AND SIMULATION IN THE ARTIFICIAL INTELLIGENCE ERA

6.1.1 Introduction

The validation of models and verification of simulation programs are extremely important factors especially in large scale simulation studies. Additionally, the topic requires attention in light of the fact that huge amounts of money are being spent annually in developing, using and maintaining computer models worldwide. It is widely accepted that modelling and simulation may add other sources of error to the large number of errors that may exist in basic computerization. As a result, quality assurance along with reliability of models used and associated software is of primordial importance. As defined by Ören (1986c): "Quality Assurance of an element of a simulation study is an assessment of its characteristics to certify its acceptability or to recommend ways and means for its improvement or to recommend its rejection".

Quality assurance for traditional modelling and simulation has been well elaborated in the literature over the years and is only briefly discussed here. Ören (1987a) gives a global taxonomy of studies and references that deal with the topic. The main focus, however, in this section is to discuss the currently advanced quality assurance

concepts.

The main problems that are investigated to ensure the quality assurance in traditional modelling and simulation include :

1) Applicability of validity and verification of experimental conditions.

2) Computational validity

3) Experiment and Model Compatibility

4) Formal checks for model integrity, consistency, completeness and comprehensibility

5) Software quality assurance of the Run Time Library

In addition, due to the increasing applications of simulation studies, the development of an assessment methodology is essential. The quality assessment studies fall under two main categories (Ören 1984a). They are : 1) Descriptive assessment which consists of syntactic assessment, morphological assessment, and semantic assessment. 2) Normative assessment consisting of pragmatic assessment and ethical assessment.

6.1.2 ADVANCED QUALITY ASSURANCE CONCEPTS IN MODELLING AND SIMULATION

With the advent of advanced knowledge processing paradigms such as knowledge based systems or the like in

complex systems, a new class of quality assurance techniques are required. The need for such techniques are to :

1) Avoid additional sources of errors to already complex systems.

2) Take advantage of artificial intelligence techniques to eliminate some types of errors.

As a result, three additional areas of advanced quality assurance techniques have been formulated by Ören (1987a):

1) Quality assurance in cognizant modelling and simulation

2) Cognizant quality assurance in modelling and simulation

3) Cognizant quality assurance in cognizant modelling and simulation

Cognizant simulation or artificial intelligence based simulation is the term given to a simulation where the modelling environment and/or simulation system have cognitive abilities. Cognizant simulation systems may allow goal directed modification of a model's structure during a simulation study while cognizant simulation environments can be used to specify models, parameters, etc., as well as to monitor the simulation process. It is clear that such a system requires special needs for quality assurance. Traditional concepts that must be extended to such systems include adequacy, completeness, consistency and integrity of

both static and dynamic knowledge bases which may include model and rule bases which are critical components of current expert and knowledge based systems.

For static knowledge bases, techniques need to be developed and implemented to ensure knowledge base adequacy, completeness and consistency. Dynamic knowledge bases may exist in an advanced simulation environment such as those that need updating as a result of machine learning to provide knowledge rich environments for specifications and executions of simulations. In such a case, one needs to additionally ensure knowledge base integrity. Each time a need for an update occurs, correctness and consistency of the knowledge base needs to be assured via relevant syntactic and semantic checks. This should help to ensure that catastrophes do not occur when knowledge processing takes place. A similar need occurs in rule-based simulation systems. In such systems, the rule base must be carefully monitored since logical programming can quite easily give way to erroneous answers. This can occur as the inference engine can function and generate answers even in the absence of a rule or if another rule is added to the original rule-base. As a result, automated consistency checks of the rule-base are critical after every change of rules.

There are two main ways in which cognizant quality assurance can be utilized in modelling and simulation (Ören 1986c). In the first case, one can use cognizant techniques

such as rule based programming to enhance quality assurance operations by building a rule base for such operations.

Secondly, one can assure quality by using cognizant techniques to guide the user in the specification phase of the simulation in order to eliminate some types of error such as morphological and semantic ones (Ören 1987a). This "on line" approach allows immediate remedial action at each stage, say, of a specification rather than to check the existence of cases which need to be improved as quality assured. This feature is present in Magest (Aytac and Ören 1986) which is an advisory system for simulation modelling that carries out a variety of semantic checks to assure completeness and compatibility of specifications. The system additionally, provides in its advisory mode, user friendly explanation and can also suggest alternatives to correct them.

The third area pertaining to cognizant quality assurance in cognizant modelling and simulation is in its early stages of development. Concepts and techniques need to be generated to deal with this problem. Typical techniques should include the use of a dynamically enhanceable knowledge base containing knowledge or knowledge processing knowledge relevant to quality assurance operations for quality assurance of cognizant simulation or simulative environments (Ören 1987a).

6.2 QUALITY ASSURANCE OF THE RUN TIME LIBRARY IN THE GEST ENVIRONMENT

The quality assurance of the run-time library is an important subset of the quality assurance required in traditional modelling and simulation. This is easily conceivable if one realizes the role of the run-time library in the continuous simulation process. The library is the engine which is responsible for driving the simulation to determine its resulting behaviour. The aim of this section is to highlight the quality assurance considerations present in the run-time library. Also, potential ways in which the Gest environment can be utilized in the future to enhance the quality assurance is discussed in this section and in section 6.3. This can be seen as an important subset of cognizant quality assurance in modelling and simulation.

As discussed in chapter 3, the information about the specific model to be simulated and the conditions under which the results are to be obtained are specified by the user in the Gest language with the aid of the Magest system. After translation of the specification by the Gest translator, the program is executed using the Gest run-time library. The latter is developed with careful consideration for the attribute of 'robustness' as specified by Cellier (1979). In doing so, the user is relieved as much as possible of those aspects that deal with the execution of his model such as the integration step size and other associated parameters. The user is not required to provide

any information he does not have at his disposal but instead is allowed to concentrate on those factors involving the specification of his model in Gest . As a result, in the implementation of the run-time library, the user when specifying a discontinuous model, is given the option of specifying a maximum step size. As mentioned earlier in Chapter 5, this is important in reducing the probability of the detection criterion missing a discontinuity. For the other algorithms in the run-time library, there is no need to specify a maximum step size, HMAX. This would unnecessarily restrict the rate of integration which should be controlled by the tolerance or accuracy of the integration required by the user. However, in each algorithm of the run-time library, the option of specifying a minimum step size is given. This is important so as to prevent the possibility of a run-time error occuring due to overflow exception. This can occur if the algorithms are used in the integration of stiff or oscillating models where the step size control mechanism can force the step size to be reduced to a very small value for the user tolerance specified. On the other hand, the value of HMIN should not be relatively large, since this would cause the integration to be interrupted unnecessarily. Once the values are specified, the Magest system could do appropriate simple checks to determine their acceptability. These checks involve determining whether the given values are valid ones. The accuracy and stability of the methods is dependent on their

values being positive and less than one. Additionally, the value of HMAX for the discontinuity processing algorithm must be greater than the value specified for HMIN. The users are alerted if the specified values are invalid, at which point Magest would suggest an appropriate optimal range and request the user to make the appropriate changes. If the user does not specify any values then the algorithm automatically assigns the above mentioned values for these variables.

The user is not allowed to specify the initial step size for the integration in all of the algorithms of the run-time library except the Fourth-Order Runge-Kutta method where it is allowed as an option. This feature is a significantly important one which some available codes neglect and the user is requested to guess some initial step size. This is an unsatisfactory procedure to which users should not be subjected. Instead of specifying the step size, the step size controlled integration algorithms require the user to provide a tolerance for the accuracy of the results desired. In all the algorithms in the run-time library, the algorithms are automatically assigned an initial step size based on the formula derived by Shampine and Gordon (1975). It is given by:

HINIT = min(HBOUND, ((abs(0.5 * TOL/FDER)) ** 0.5)/4)

where

HBOUND is the upper bound on the initial step size

TOL is the user specified tolerance

FDER is the maximum value of the derivative functions at the initial point

A value for HBOUND should be specified by the user. This value which indicates the general scale of the problem is important so as to prevent any difficulty due to a state derivative being zero in the computation of the initial step size HINIT. As mentioned earlier, some flexibility has been allowed for specifying an initial step size. In some cases, the user may have a good reason for specifying an initial step size for his/her simulation run. In these cases, the user should use the Fourth-Order Runge-Kutta method which has this option available. However, before proceeding with this step size the algorithm checks to see whether the value of HBOUND has been exceeded. If so, the user is warned that this is the case, and is given the option to use the value specified for the initial step size or to use the value of HBOUND.

As mentioned above, the value of TOL is important in controlling the rate or step size of the integration. Magest could also carry out checks on the value of TOL supplied by the user to ensure its validity. It must necessarily be a positive number and should be very small less than one.

The run-time library also has internal mechanisms to control the integration process and also to indicate when

the algorithm is unable to proceed properly. These include:

1) Ensuring that the integration proceeds efficiently. This requires that the integration advance as close to optimal as possible at each step. This is pertinent to all algorithms in the library excluding the Fourth Order Runge-Kutta. In some available codes, this aspect is not considered seriously. The step size update procedure is carried out quite arbitrarily. The following step size control scheme proposed by Birta (1980) ensures an efficient update close to optimality.

```
H = 0.9 * HPREV * ((TOL/E)) ** 1/D
```

Then,

$$HNEW = \begin{cases} H & 1/D \leq H/HPREV \leq D \\ HPREV/D & H < HPREV / D \\ D*HPREV & H > D*HPREV \end{cases}$$

E represents MAXj (Ej)

Ej represents local truncation error estimate

TOL represents the specified user tolerance

D represents the order of the method

2) In the discontinuity processing algorithm, at the end of each integration step size, a check is made to ascertain whether the new step size produced by the algorithm is

larger than the value of HMAX. If it is, the value of the new step size is replaced by the value of HMAX, thus ensuring that the algorithm does not ever proceed with a step size greater than HMAX as desired.

3) For all the algorithms in the run-time library, the user is alerted and the integration suspended if the new step size produced by the algorithm is smaller than HMIN. The system has dialogue with the user to try to rectify the situation by requesting the user to either specify a smaller HMIN or a less stringent TOL if practically feasible so that the integration step may proceed past this point. If it is not practically feasible, then the simulation is aborted and the user requested to investigate the model closely.

4) A check is done to ensure that an output point is not required at a step size less than or equal to HMIN after the next integration step is taken. If such a case exists, the new step size is replaced by the value (TCOM - T)/2 where TCOM is the communication time where output is required, and T is the present computational time. This will avoid unnecessary suspension of the integration and possible abortion since the step size H would be less than or equal to HMIN. The integration is repeated with this step size until TCOM is reached. Once the output point is reached, the step size update is carried out according to the step size control scheme described above in all algorithms except the Fixed Step Runge-Kutta method where the initial step size is

again employed. The above features present in the run-time library are responsible for ensuring its 'robustness'.

6.3 FUTURE POSSIBILITIES FOR INTEGRATION ALGORITHM SELECTION WITHIN THE GEST ENVIRONMENT

In the run-time library, a range of integration algorithms are offered for the user to choose from in carrying out the simulation of his continuous model. Unfortunately, even in recently developed languages the user is not aided in determining the optimal choice of algorithm from amongst those available in the run-time library for simulating his specific model. This non-trivial problem is further complicated by the fact that in the majority of cases, the users though knowledgeable about the problem domain area may not be in a position to make such a decision. The end result of such a situation is usually a poor choice, the user specifying some algorithm he may be familiar with or merely using the default option that may be available in the run-time library.

For the Gest environment, this limitation can be somewhat overcome. It is proposed that Magest perform checks on the model provided by the user to determine whether his specified algorithm is indeed a suitable and optimal choice. If Magest is not in agreement, it advises the user as to which algorithm, is the most appropriate one to be chosen from the Gest run-time library and the reasons for its

conclusions. This task requires considerable research and experimentation to develop and validate appropriate rules for giving advice on the choice of a particular integration algorithm for a specific model. The aim of this section is to provide insight and information as to what factors should be considered and utilized in the formulation of such rules. A well developed set of rules would be a valuable asset of such an environment and would further enhance its abilities as a computer aided modelling tool.

In the current run-time library, the user is able to choose from five integration algorithms. These consist of : 1) Fixed Step Runge-Kutta 2) Variable Step Runge-Kutta Fehlberg 3) Adams Predictor Corrector Method 4) Discontinuity Processing Algorithm and 5) Multi-Rate Integration Algorithm.

The first three algorithms are termed general purpose algorithms, while the discontinuity processing algorithm and Multi-Rate algorithm are specialized algorithms.

The rules that could be utilized by Magest in advising that the discontinuity algorithm be selected in the simulation of a particular system is quite straightforward.

1)

IF the model has a 'model update' section

THEN the discontinuity processing algorithm should be used.

2)

IF there is an "if then else" structure in the derivative section

THEN the discontinuity processing algorithm should be used.

The rules that could be used by Magest in advising the multi-rate integration algorithm be selected is not as straightforward. Firstly, it must be determined under which conditions the algorithm can be advantageously utilized. For coupled models with several component models the advantage of using the algorithm should be dependent on the number of interconnections that exist in the internal coupling section of the specification. Although, Palusinski (1985) does not make direct reference to it for his algorithm, the results of his experiments indicate as would be expected that the CPU time savings decreases with the number of interconnection variables. This is understandable in light of the fact that although considerable CPU time is achieved by partitioning the original system into two component systems with similar characteristics, the amount of interpolation required in integrating the resulting structure increases with the number of interconnection variables. It is clear that at some number of interconnection variables, the amount of interpolation

required does not warrant the use of the multi-rate method. As explained previously, Birta (1980) conducted numerical experiments to investigate this problem for his algorithm and concluded that a maximum of three interconnection variables should be allowed for his method to be feasible. From the numerical experiments given by Palusinski (1985) this number seems like a suitable rough benchmark for his algorithm. Needless to say, as mentioned earlier, detailed experimentation should be carried out to provide a much more accurate benchmark. It has previously been cited that the method is usually advantageous for large dynamic systems. Another question arises as to how one defines large. Additionally, the algorithm is particularly favorable when the dimension of the slow system is significantly greater than the fast system. Experimentation is required to develop rules based on the above information which Magest can then use in providing the user with advice on the applicability of the Multi-Rate integration algorithm.

In general, two situations may occur. Firstly, the user has been able to partition the system beforehand and has used the Gest definition for a system employing the use of multi-rate integration, that is, slow and fast component models have been specified etc. In this case, Magest would carry out the appropriate checks based on the rules developed. If the conditions of the checks do hold then Magest ensures that the multi-rate integration algorithm is used. If one or more of the conditions is violated, then

Magest advises the user that the multi-rate algorithm must be used because of the structure of the model specification, but that no advantage in terms of CPU savings may be achieved by using the algorithm. At this point, the user may use the algorithm and run the simulation or respecify the simulation and use an appropriate general purpose algorithm with aid from Magest once more.

The other possibility involves a system being specified with as a coupled model as opposed to a multi-rate model which satisfies the conditions of the checks made by Magest for the applicability of multi-rate integration. For such a system , Magest indicates that the multi-rate integration algorithm could be potentially advantageous if a suitable partition for establishing a slow and fast system could be determined by the user. Magest does not allow the multi-rate algorithm to be used until the system has been properly partitioned and respecified for use by the multi-rate integration. At this point, the user has the option of merely using a general purpose algorithm with the aid of Magest, or determining a suitable partitioning and then respecifying his model so that the multi-rate integration can be used. The partitioning problem up till now has been left entirely up to the responsibility of the user. The user must examine the qualitative features of the process in order to determine a suitable partition (Palusinski 1985). In chapter 7, an expert monitor is proposed to aid the user in the partitioning problem.

Once an appropriate partitioning is found (if possible), then the user re-specifies the model in the appropriate form including a fast and slow system etc. Magest then ensures multi-rate integration is used for the simulation. This procedure is hardly worthwhile for a simulation that is to be carried out only once. However, in practice this is usually not the case. Large dynamic processes require frequent experimentation with the initial value problem that describes the process, and from this point of view the procedure could very well prove effective.

The choice of a particular general purpose algorithm involves consideration of a few factors while examining the system. It is clear from previous discussions , the conditions under which one of the general purpose algorithms is considered for selection. 1) An update section is not present in the dynamic structure of the specification nor does an 'if then else' structure occur in the derivative section. 2) The model specified as a multi-rate model satisfies the conditions under which the multi-rate integration algorithm is potentially advantageous. 3) The system is specified as a coupled model even though the use of the multi-rate algorithm could have potential advantages. However, for some reason, such as a suitable partition not able to be defined, the user desires not to use the multi-rate integration and Magest allows the flexibility of using a general purpose algorithm which it gives advice on.

Once it has been ascertained by Magest that a general purpose algorithm is to be used in the integration, three main factors should be considered and investigated to develop appropriate rules to be used by Magest in advising on the best choice of general purpose algorithm to be employed. These consist of : 1) the accuracy requirements of the model 2) the frequency of output required by the model and 3) the expense of the function evaluations in the model.

As Cellier (1979) points out, the CPU cost to execute a specific model can be graphed versus the required relative accuracy as shown in Figure 6.1. The rules for accuracy could be based on the rule of thumb cited by Cellier and may have the form

IF the tolerance specified by the user
>= 10 **(-4)

THEN the Fixed Step Runge Kutta
algorithm is favoured

IF the tolerance specified by the user
< 10 **(-4)

THEN the Adams method or the Runge Kutta
Fehlberg algorithm is favoured.

For the frequency of output criteria, Cellier (1979)

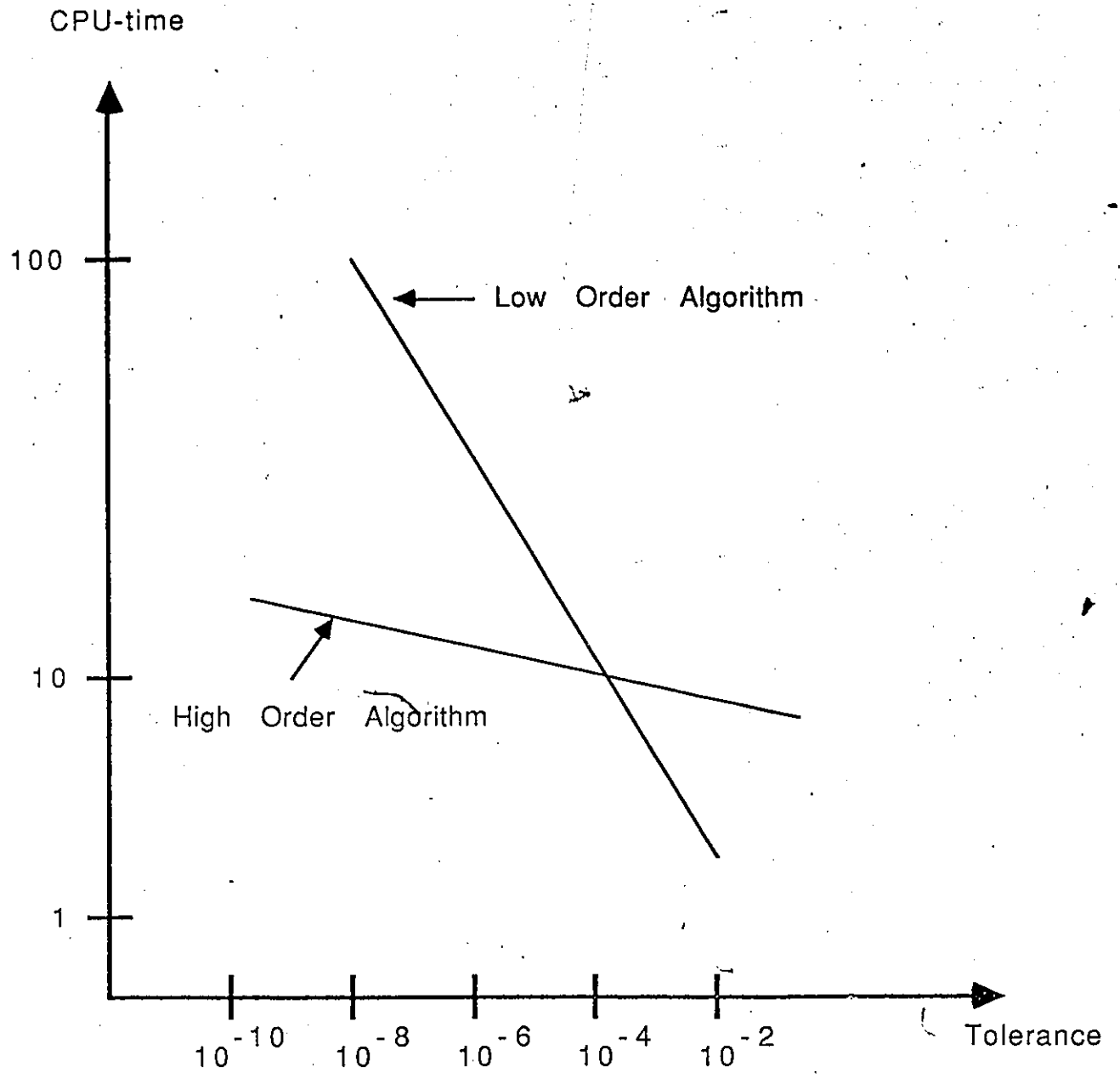


Figure 6.1 CPU-time versus Tolerance Requirements for Low and High Order Algorithms

shows the accumulated CPU-cost graphed versus the simulation clock for one step and multi-step integration as shown in Figure 6.2. The graph illustrates the fact that one step methods are more economic initially because of their self starting mechanism. However, the multi-step algorithm becomes more economic over a longer period of time provided the problem is a purely continuous one. Finally, the low order single step algorithm favours continuous problems where frequent output is required as opposed to their higher order equivalents. Frequency of output could be easily determined by Magest, from the communication interval specified and the values for the start and end times of the simulation as specified by the user. The number of output points is given simply by (TFIN - TINIT)/COMINT. In summarizing, factors should be considered in developing rules for this criteria would be :

IF a high frequency of output is computed from the appropriate information in the specification

THEN the Fixed step Runge Kutta algorithm is favoured.

IF a low frequency of output is computed from the appropriate information in the specification

THEN the Runge Kutta Fehlberg or the Adams Predictor algorithm is favoured.

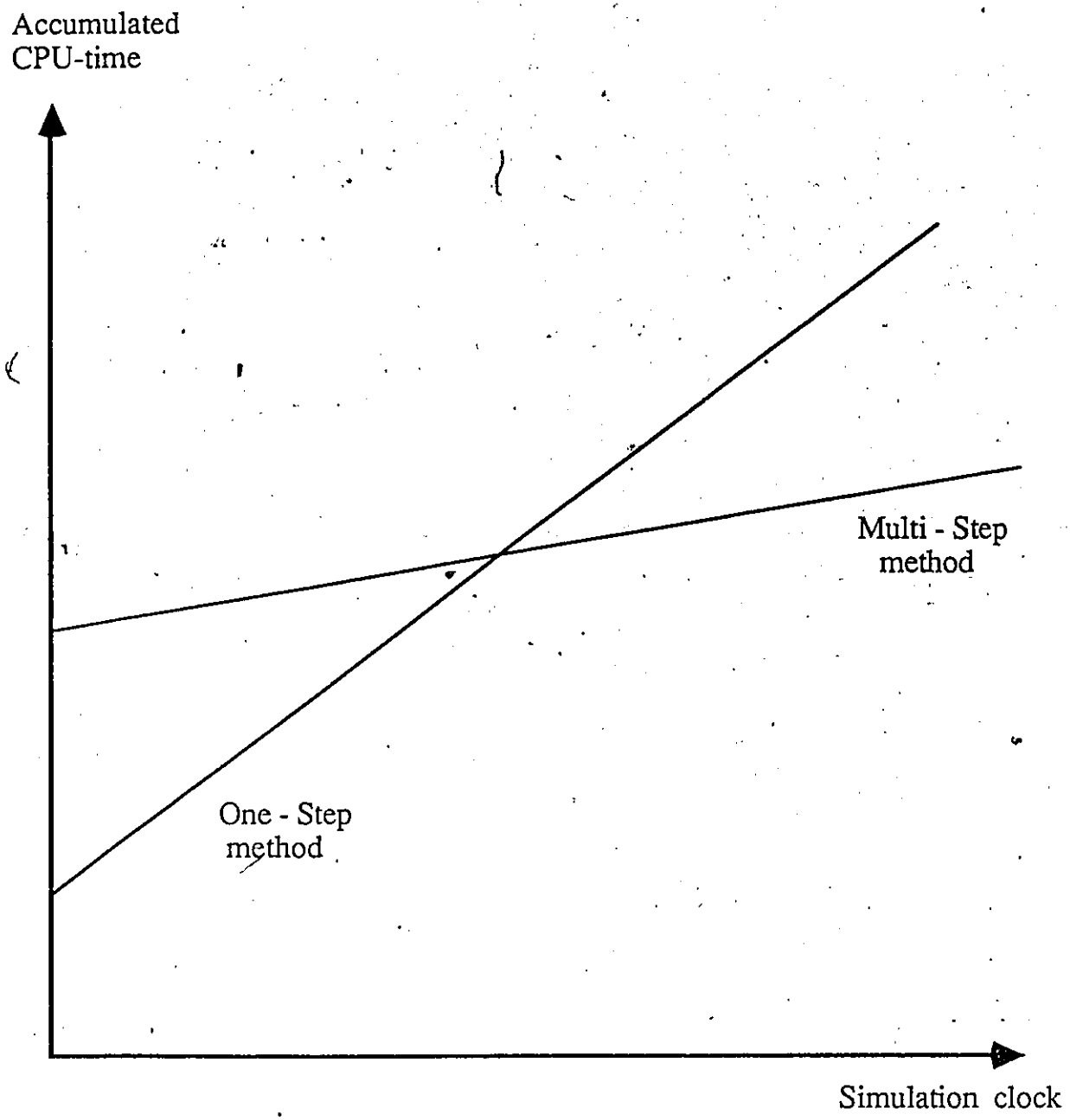


Figure 6.2 Acccumulated CPU-time Versus Simulation Clock for One-Step and Multi-Step Integration

The final criteria concerning the expense of function evaluations is investigated by Magest determining a relative complexity factor from the equations describing the model. A suitable benchmark is required for this non-trivial problem to distinguish a high complexity model from a low complexity one. A high complexity model favours the use of the Adams predictor corrector algorithm while a low complexity model favours one of the one step Runge Kutta methods. The following factors should be used in developing rules for this criteria to be used by Magest in recommending a general purpose algorithm for a particular model.

IF the model has been ascertained to have a high complexity factor

THEN the Adams Predictor Corrector Method is favoured.

IF the model has been ascertained to have a low complexity factor

THEN either the Fixed Step Runge Kutta or the Runge Kutta Fehlberg algorithm is favoured.

Figure 6.3 summarizes the conditions under which each of the general purpose algorithms is favoured based on the factors described above. As should be evident, a number of possible combinations of factors may exist for a particular model. However, the number of rules could be significantly

	Accuracy		Model Complexity		Output Frequency	
	Low $<10^{-4}$	High $>=10^{-4}$	Low	High	Low	High
Fixed Step Size Runge -Kutta Method	✓		✓			✓
Variable Step Size Runge -Kutta Fehlberg Method		✓	✓		✓	
Adams Predictor Corrector Method		✓		✓	✓	

Figure 6.3 Factors Affecting Selection of General Purpose Algorithms

reduced as follows. Firstly, because of the problems associated with restarting the Adams method, Magest should only consider this algorithm when the user specifies a long communication interval. Secondly, the accuracy requested by the user via the specified value for TOL should be deemed the most important criteria. This is a viable assumption in light of the fact that, with the increasing power of computers, the other factors are becoming less and less important.

These considerations not only drastically reduce the number of rules needed but improves the ability of Magest to select the appropriate general purpose algorithm despite the combinations of factors that may have to be considered for any particular model.

As a result, the following factors can be summarized which could be used in the development of rules for Magest in selecting an appropriate general purpose algorithm for a particular model.

1) The Adams Predictor Corrector Method would be advantageous in all cases where the model has a high complexity factor and a high tolerance is specified by the user.

2) The Runge-Kutta Fehlberg Method would be advantageous in all cases where the model has a low complexity factor and a high tolerance is specified by the user.

In all other cases, the Fourth Order Runge-Kutta Method would be recommended for use.

In a nutshell, each of the algorithms in the run-time library may be advantageous for a particular model. The Magest system should be able to provide advice concerning the selection of the integration algorithm for a model. Further research and detailed experimentation utilizing the knowledge above should result in the development of a set of rules that Magest can use to this end.

CHAPTER 7

CONCLUSIONS AND FURTHER STUDY

In this thesis, several issues concerning the applications of Artificial Intelligence in Simulation are discussed. The main theme, however, deals with advanced simulation environments, in particular the one based on 'Gest' as discussed in Chapter Three. The correspondence shown between the structures of the Gest specification programs and their equivalent procedural programs along with the test examples demonstrate how the translator for the Gest environment is to function thus providing aid in its construction. The run-time library for the Gest environment has been developed to handle non-stiff initial value problems and the algorithms are used to drive the test examples in the thesis.

A number of areas for further study are possible apart from those mentioned as future directions for artificial intelligence applications in simulation environments in section four of Chapter two. As indicated previously in the thesis, one should use the algorithm that is most advantageous for a certain class of problems. As a result, it is highly recommended that the run time library be expanded to include other algorithms for handling the different problem areas in continuous simulation as outlined by Gear (1981). In particular, an algorithm for handling stiff systems, preferably Gear's algorithm (Gear 1971)

should be included.

It is recommended that a set of rules to be used by Magest in advising the user as to the integration algorithm to be selected from the run-time library for a particular model be well developed using section 6.2 of chapter 6 as a base and then implemented. As stressed in section 6.2, this will require further research and a series of experimentations to determine the relative importance of these suggestions. The successful implementation of this set of rules would further enhance the computer aided modelling abilities of the Gest environment.

As mentioned in Chapter 6, the development and implementation of the intelligent monitor during simulation run time could be a useful aid to the user in dealing with the partitioning problem for use of multi-rate integration. The user would first shorten the length of the simulation time so that the simulation is run for a sufficient period of time that the model is behaving in 'steady state'. This usually occurs a fair length of time after the initial transient time has elapsed and is estimated from the dynamics of the process. Additionally, a variable step method is chosen to drive the simulation. During the simulation, the monitor would collect statistics giving the weight each state had in controlling the step size. This, it would determine by studying the local error estimates of the state variables at each step. The use of the monitor in this

way could aid in the partitioning process. If certain state variables have large error estimates responsible for slowing the rate at which the system integration proceeds, then these can be identified as being potential fast system variables. It is necessary to incorporate into the final decision of the partitioning any a priori knowledge of the qualitative features of the system. Preliminary studies done using the electronic oscillator model confirmed the usefulness of the monitor in aiding the user to determine an appropriate partitioning. Additionally, assuming the run-time library has been expanded as outlined above, this 'intelligent' monitor could be further enhanced by incorporating into the system functions of stiffness detection, order selection and abnormal phenomena processing as discussed by Xiong and Song (1986). Appropriate rules for these situations would be collected and put into a knowledge base. The knowledge base would have a learning mode as discussed by Xiong and Song and the inference strategy would be a sequential one.

The present post-run graphics ability of the run-time library could be improved to include interactive abilities so that the trajectory of the simulation can be monitored during each stage of the simulation.

Further studies into the choice of interpolation algorithm for subsystem interfacing could be extremely useful. Palusinski (1985) mentions this as being very

important in the design of a multi-rate integration algorithm for partitioned system simulation. This is also a major consideration and handicap in the use of the quasi-parallel method proposed by Birta as discussed in Chapter four. If this handicap could be overcome, the proposed algorithm by Birta (1980) could be most advantageous if the large dynamic system consisted of component systems with known general characteristics such as discontinuous and stiff component systems for example. With the expanded version of the run-time library, the use of the best integration algorithms for each component system can be employed so that each component system moves forward in time independently. In this case, then, it is clear that such a method would have distinct advantages over the traditional method of dealing with a system having several component systems with known general characteristics. Gear's algorithm (Gear 1971) would be utilized in the integration of the stiff component system while the discontinuity algorithm would be utilized for the discontinuous component system.

Alternatively, the implementation of the algorithm proposed by Brosilow (1985) would be most advantageous for a system having component systems with known general characteristics since it would allow the parallel processing of the individual component systems using the best integration algorithms for each component system. However, this method would require software that runs on parallel processors.

APPENDICES. EXAMPLES FOR GEST TRANSLATOR

Appendix A. THE DIABETES MELLITUS MODEL: NORMAL PERSON

A.1 Problem Definition

This example utilizes the Differential Model of Diabetes Mellitus (Davies 1976) to simulate the blood sugar and insulin levels of a normal person over a typical day. The model for the normal person consists of two differential equations describing the state derivatives of the model, the food source input, and the associated parameter set. The state variables consist of the sugar level and insulin level in the blood. The Fourth Order Runge-Kutta method is used as the method of integration. Figure A.1 shows the pictorial representation of this component model. The results are identical to those given in the article by Davies.

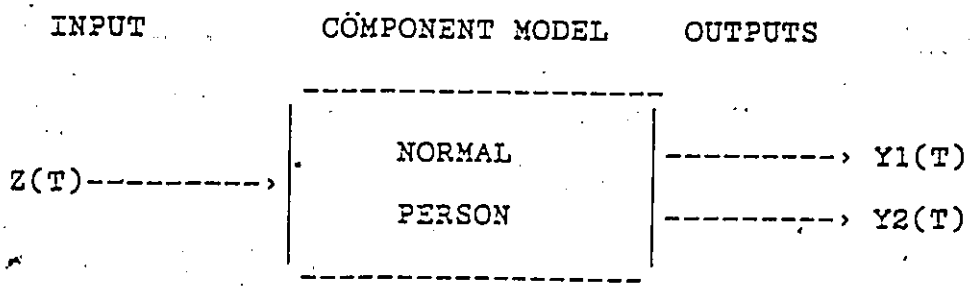


Figure A.1 System Representation For Normal Person

A.2 Gest Specification of The Diabetes Mellitus Model: Normal Person

```
Program StudyOfNormalPerson

ContinuousModel NormalPerson
  StaticStructure
    Inputs
          Z                    (* Food Input                              *)
    States
          Y1,                  (* Blood Sugar Level                       *)
          Y2                   (* Blood Insulin Level                     *)
    Outputs
          Y1,                  (* Blood Sugar Level                       *)
          Y2                   (* Blood Insulin Level                     *)
    AuxiliaryVariables
          H                    (* Step Function                           *)

    Constants YI1, YI2
      YI1 = 100                (* Initial Sugar Level                     *)
      YI2 = 0                  (* Initial Insulin Level                   *)
    EndConstants

    Parameters
          A1,                  (* Sensitivity of sugar level
                                     gradient to insulin                   *)
          A2,                  (* Sensitivity of sugar level
                                  gradient to low blood sugar
                                  levels                                   *)
          A3,                   (* Sensitivity of sugar level
                                  gradient to the input                    *)
          B1,                  (* Sensitivity of insulin level
                                  gradient to high blood sugar
                                  levels                                   *)
          B2,                  (* Sensitivity of insulin level
                                  gradient to insulin levels               *)
          QFOOD[3],            (* Quantity of food at three
                                  meal times                               *)
          TFOOD[3],            (* Meal times                              *)
          KFOOD[3]             (* Delay parameter                         *)
  EndStaticStructure

  DynamicStructure
    Derivatives
      If ((Y1 - YI1) < 0.0)
         Then H = 0
         Else H = 1
      EndIf
      Y1' = -A1*Y1*Y2 + A2*(YI1-Y1)*(1-H) + A3*Z
      Y2' =  B1*(Y1-YI1)*H - B2*Y2
    EndDerivatives
  EndDynamicStructure
EndModel NormalPerson

Parameter Set 1
  Parameters
      A1       = 0.05
      A2       = 1.0
      A3       = 4.0
      B1       = 0.5
      B2       = 2.0
      QFOOD[1] = 50
      QFOOD[2] = 100
      QFOOD[3] = 100
```

```
        KFOOD[1] = 4.0
        KFOOD[2] = 4.0
        KFOOD[3] = 4.0
        TFOOD[1] = 8
        TFOOD[2] = 13
        TFOOD[3] = 20
    EndParameters
  EndParameter Set 1


  Parameter Set 2
    Parameters
        A1       = 0.05
        A2       = 1.0
        A3       = 4.0
        B1       = 0.5
        B2       = 2.0
        QFOOD[1] = 50
        QFOOD[2] = 100
        QFOOD[3] = 100
        KFOOD[1] = 6.0
        KFOOD[2] = 6.0
        KFOOD[3] = 6.0
        TFOOD[1] = 8
        TFOOD[2] = 13
        TFOOD[3] = 20
    EndParameters
  EndParameterSet 2


  Experiment 1
      TimeUnitIs Hour
        StartTime = 0
      IntegrateBy Fourth-Order Runge Kutta.
        RelError = 0.001
      Communicate at every 0.5 Hour
      SimulateUntil Time = 24
      InitialStates
        YI1 = 100.0
        YI2 = 0.0
      EndInitialStates
      SaveOutputs Y1,Y2
      Input Function Z
        At Times TFOOD(I)
          IF (T.GE.TFOOD(I))
              THEN Z = QFOOD(I)*EXP(-KFOOD(I)*(T-TFOOD(I)))
            ELSE Z = 0
          ENDIF
      EndInput Function Z
  EndExperiment 1


  Run 1
    ToObserve NormalPerson
      WithParameterSet 1
      InExperiment 1
      WithPostRun
        OutputModule 1 On Printer
      EndPostRun
  EndRun 1

  Run 2
    ToObserve NormalPerson
      WithParameterSet 2
      InExperiment 1
      WithPostRun
```

```
      OutputModule 1 On Printer
    EndPostRun
EndRun 2


OutputModule 1
  New Page
  Heading ForFirstPage 1 Line
    'Study of NormalPerson'
  List Parameters And InitialConditons
  PlotandList Y1,Y2
EndOutputModule 1


EndProgram StudyOfNormalPerson
```

```
C  A.3    Equivalent Fortran Program With Results For                    DIA00020
C         Diabetes Mellitus Model: Normal Person                         DIA00030
                                                                         DIA00040
                                                                         DIA00050
                                                                         DIA00060
                                                                         DIA00070
C                          Variable List                                 DIA00080
C                                                                        DIA00090
C Input  : N              --Dimension of the Vector of State Variables   DIA00100
C                                                                        DIA00110
C          Y(1)-Y(N)      --Initial Values of State Variables            DIA00120
C                                                                        DIA00130
C          TINIT          --Initial Time of Simulation                   DIA00140
C                                                                        DIA00150
C          TCOM           --Communication Time                           DIA00160
C                                                                        DIA00170
C          T              --Computational ,Time                          DIA00180
C                                                                        DIA00190
C          TFIN           --End Time of Simulation                       DIA00200
C                                                                        DIA00210
C          COMINT         --Communication Interval                       DIA00220
C                                                                        DIA00230
C          HBOUND         --Upper Bound on Initial Step Size             DIA00240
C                                                                        DIA00250
C          HMIN           --Minimum step size integration is             DIA00260
C                            allowed to proceed at                       DIA00270
C                                                                        DIA00280
C          HINIT          --Starting Integration Step Size               DIA00290
C                                                                        DIA00300
C          H              --Current Integration Step Size                DIA00310
C                                                                        DIA00320
C          TOL            --Tolerance or Accuracy specified by user      DIA00330
C                                                                        DIA00340
C Output : TIME           --Vector of times of Output Values             DIA00350
C                                                                        DIA00360
C          YVAL           --Vector of Values of Y1 at Output Points      DIA00370
C                                                                        DIA00380
C          YVAL3          --Vector of Values of Y2 at Output Points      DIA00390
C Program Variables :                                                    DIA00400
C          Y(N)           --State Variables Vector                       DIA00410
C                                                                        DIA00420
C          DY(N)          --Derivative Vector of State Variables         DIA00430
C                                                                        DIA00440
C        DY1(15)-DY4(15)--Working Vectors in Integration Routine         DIA00450
C                          capable of integrating systems with up        DIA00460
C                            to 15 state variables.                      DIA00470
C                                                                        DIA00480
C        F1(15)-F3(15)   --Working Vectors in Integration Routine        DIA00490
C                          capable of integrating systems with up        DIA00500
C                          to 15 state variables.                        DIA00510
C                                                                        DIA00520
C          NFLAG          --Flag Which Signals Collection of Output      DIA00530
C                            Variables.                                  DIA00540
C                                                                        DIA00550
C          HPOS           --Intermediate variable used in computing HINIT DIA00560
C                                                                        DIA00570
C          NPL            --Counter used for Graphics Routine.           DIA00580
C                                                                        DIA00590
C        . FDER           --Maximum value of derivative functions at     DIA00600
C                            initial time                                DIA00610
C                                                                        DIA00620
                                                                         DIA00630
      PROGRAM MAIN                                                       DIA00640
```

```
C       This is a test program utilizing the Fourth Order                DIA00650
C       Runge-Kutta Integration Algorithm to drive the                   DIA00660
C       Simulation of A Component Model DiabetesMellitus.                DIA00670
                                                                         DIA00680
      IMPLICIT REAL (A-H,O-Z)                                            DIA00690
      DIMENSION Y(5)                                                     DIA00700
      REAL*4 YVAL1(300),YVAL2(300),TIME1(300)                            DIA00710
      EXTERNAL DERIV                                                     DIA00720
      COMMON/DIFF/YVAL(342),YVAL3(342),TIME(342),NPL                     DIA00730
      COMMON/PAR/YI(2),TFOOD(3),KFOOD(3),QFOOD(3),A1,A2,A3,B1,B2,Z       DIA00740
      COMMON/SAME/PU(24)                                                 DIA00750
      CHARACTER*1 CHOICE                                                 DIA00760
                                                                         DIA00770
      PRINT*,'Welcome To The Diabetes Mellitus Simulator'                DIA00780
      PRINT*,'    '                                                      DIA00790
      PRINT*,'Would You Like To Perform a Simulation ? (Y/N) '           DIA00800
      READ*, CHOICE                                                      DIA00810
      IF (CHOICE .EQ. 'Y') THEN                                          DIA00820
200   CALL INIT                                                          DIA00830
                                                                         DIA00840
C                                  Initialization from Menu              DIA00850
C                                                                        DIA00860
C                                  Initialize Parameters                 DIA00870
199   A1        = PU(1)                                                  DIA00880
      A2        = PU(2)                                                  DIA00890
      A3        = PU(3)                                                  DIA00900
      B1        = PU(4)                                                  DIA00910
      B2        = PU(5)                                                  DIA00920
      QFOOD(1)  = PU(6)                                                  DIA00930
      QFOOD(2)  = PU(7)                                                  DIA00940
      QFOOD(3)  = PU(8)                                                  DIA00950
      TFOOD(1)  = PU(9)                                                  DIA00960
      TFOOD(2)  = PU(10)                                                 DIA00970
      TFOOD(3)  = PU(11)                                                 DIA00980
      KFOOD(1)  = PU(12)                                                 DIA00990
      KFOOD(2)  = PU(13)                                                 DIA01000
      KFOOD(3)  = PU(14)                                                 DIA01010
                                                                         DIA01020
                                                                         DIA01030
                                                                         DIA01040
                                                                         DIA01050
C                                  Initialize State Variables            DIA01060
      N         = 2                                                      DIA01070
      Y(1)      = PU(16)                                                 DIA01080
      Y(2)      = PU(17)                                                 DIA01090
      YI(1)     = Y(1)                                                   DIA01100
      YI(2)     = Y(2)                                                   DIA01110
                                                                         DIA01120
C                                  Initialize External Input             DIA01130
C                                      Variable                          DIA01140
      Z         = PU(15)                                                 DIA01150
                                                                         DIA01160
                                                                         DIA01170
C                                  Initialize Run Control Variables      DIA01180
      TINIT     = PU(18)                                                 DIA01190
      TFIN      = PU(19)                                                 DIA01200
      COMINT    = PU(20)                                                 DIA01210
      HMIN      = PU(21)                                                 DIA01220
```

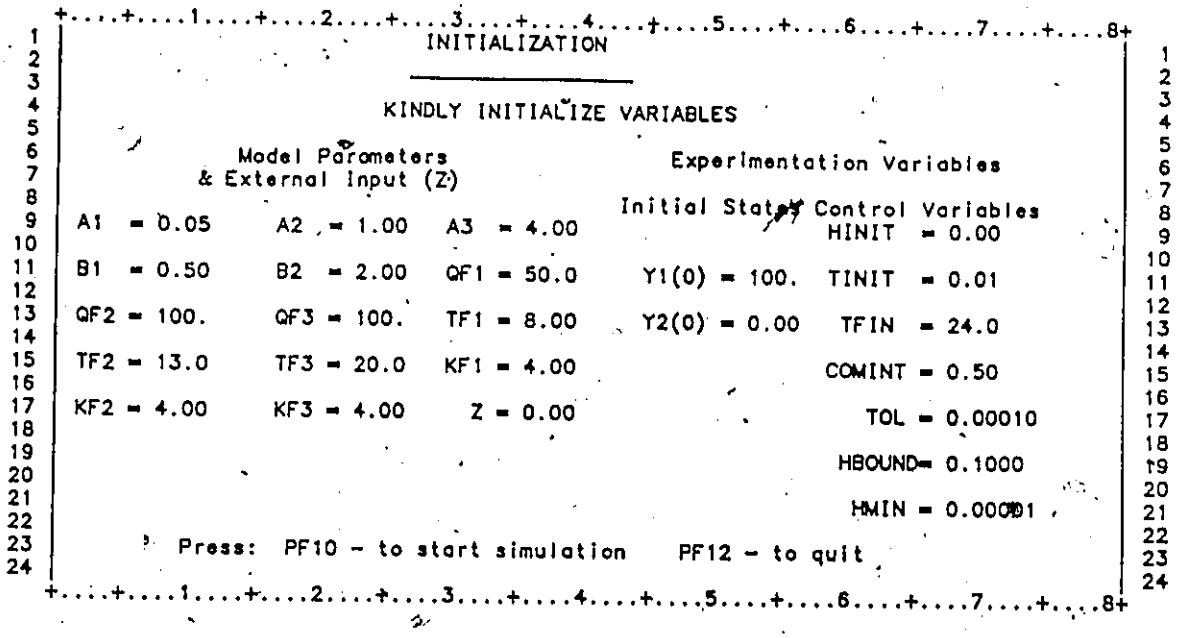
```
      HBOUND    = PU(22)                                                  DIA01230
C                                  Tolerance Specified by user for       DIA01240
C                                  computation of initial step size      DIA01250
C                                                                        DIA01260
      TOL       = PU(23)                                                  DIA01270
      HINIT     = PU(24)                                                  DIA01280
                                                                          DIA01290
      T         = TINIT                                                   DIA01300
                                                                          DIA01310
                                                                          DIA01320
                                                                          DIA01330
                                                                          DIA01340
      NPL       = 1                                                       DIA01350
C                                  Pre-Run Output :                      DIA01360
C                                           Set Up Output Format and     DIA01370
C                                           Output Initial Values.       DIA01380
C                                                                        DIA01390
      WRITE(8,17)                                                       DIA01400
   17 FORMAT(1X,'Method:',1X,'Fixed Step Fourth-Order Runge-Kutta'/19X/  DIA01410
     +5X,'Study Of Normal Person'/19X/                                  DIA01420
     +1X,'Problem: Component Model DiabetesMellitus'/19X/4X,'Time',7X,  DIA01430
     +'Value of Blood Sugar',4X,'Value of Blood Insulin')               DIA01440
      PRINT*,'   '                                                      DIA01450
      WRITE(8,21) T,Y(1),Y(2)                                           DIA01460
   21 FORMAT(1X,F8.3,5X,F8.3,5X,F8.3)                                   DIA01470
      PRINT*,'   '                                                      DIA01480
      TIME(NPL) = T                                                     DIA01490
      YVAL(NPL) = Y(1)                                                  DIA01500
      YVAL3(NPL) = Y(2)                                                 DIA01510
C                                  Run-Time Activities :                 DIA01520
C                                           Integrate by Fourth-Order    DIA01530
C                                           Runge-Kutta Method.          DIA01540
C                                                                        DIA01550
      CALL RK4(T,Y,N,TFIN,H,HINIT,HBOUND,HMIN,TOL,COMINT,DERIV)         DIA01560
C                                  Post-Run Output:                      DIA01570
C                                           Tabulate Values of State     DIA01580
C                                           Variables and Time at        DIA01590
C                                           Output Points.               DIA01600
C                                                                        DIA01610
      DO 99 I = 2,NPL                                                   DIA01620
         WRITE(8,87)TIME(I),YVAL(I),YVAL3(I)                            DIA01630
   87    FORMAT(1X,F8.3,5X,F8.3,5X,F8.3)                                DIA01640
         PRINT*,'   '                                                  DIA01650
   99 CONTINUE                                                          DIA01660
C                                           Plot Values of State         DIA01670
C                                           Variables with Time          DIA01680
      CALL ANYDEV                                                       DIA01690
      CALL GRAPH(YVAL,TIME,NPL)                                         DIA01700
      CALL GRAPH1(YVAL3,TIME,NPL)                                       DIA01710
      PRINT*,'Do you wish to carry out another simulation (Y/N) ?'      DIA01720
      READ*,CHOICE                                                      DIA01730
      IF (CHOICE.EQ.'Y') THEN                                           DIA01740
         CALL INIT                                                      DIA01750
         GOTO 199                                                       DIA01760
      ENDIF                                                             DIA01770
      ENDIF                                                             DIA01780
      STOP                                                              DIA01790
      END                                                               DIA01800
                                                                        DIA01810
```

```
      SUBROUTINE DERIV(T,Y,DY,N)                                        DIA02900
      IMPLICIT REAL(A-H,O-Z)                                            DIA02910
      DIMENSION Y(N),DY(N)                                              DIA02920
      COMMON/PAR/YI(2),TFOOD(3),KFOOD(3),QFOOD(3),A1,A2,A3,B1,B2,Z      DIA02930
C         This is the derivative evaluation                             DIA02940
C         subroutine where the user specifies                           DIA02950
C         the system derivatives that describe                          DIA02960
C         his model. As an example, the model                           DIA02970
C         Diabetes Mellitus is used.                                    DIA02980
C                                                                       DIA02990
      IF ( (Y(1) - YI(1) .LT. 0.0)) THEN                                DIA03000
         H = 0.0                                                        DIA03010
      ELSE                                                              DIA03020
         H = 1.0                                                        DIA03030
      ENDIF                                                             DIA03040
      IF ( T.LT.TFOOD(1)) THEN                                          DIA03050
         Z = 0.0                                                        DIA03060
      ELSE IF ((T.GE.TFOOD(1)) .AND. (T.LE.TFOOD(2))) THEN              DIA03070
         Z = QFOOD(1) * EXP((-KFOOD(1)) * (T - TFOOD(1)))               DIA03080
      ELSE IF ((T.GE.TFOOD(2)).AND.(T.LE.TFOOD(3))) THEN                DIA03090
         Z = QFOOD(2) * EXP((-KFOOD(2)) * (T - TFOOD(2)))               DIA03100
      ELSE IF (T.GE.TFOOD(3)) THEN                                      DIA03110
         Z = QFOOD(3) * EXP((-KFOOD(3)) * (T - TFOOD(3)))               DIA03120
      ENDIF                                                             DIA03130
                                                                        DIA03140
      DY(1) = (-A1 * Y(1) * Y(2)) + A2 * (YI(1) - Y(1)) * (1-H)         DIA03150
     +        + A3 * Z                                                  DIA03160
                                                                        DIA03170
      DY(2) = B1 * (Y(1) - YI(1)) * H - (B2 * Y(2))                     DIA03180
      RETURN                                                            DIA03190
      END                                                               DIA03200
                                                                        DIA03210
                                                                        DIA03220
      SUBROUTINE GRAPH(YVAL,TIME,NPL)                                   DIA03230
C        This Subroutine Provides A                                     DIA03240
C         Graphical Display Of The                                      DIA03250
C         Component Model. Value Of                                     DIA03260
C         Y Is Plotted Against Time.                                    DIA03270
                                                                        DIA03280
      IMPLICIT REAL*4(A-H,O-Z)                                          DIA03290
      DIMENSION YVAL(NPL),TIME(NPL)                                     DIA03300
      VALMAX = YVAL(NPL)                                                DIA03310
      VALMIN = YVAL(1)                                                  DIA03320
      DO 13 I = 1,NPL                                                   DIA03330
         IF (YVAL(I).GT.VALMAX) THEN                                    DIA03340
            VALMAX = YVAL(I)                                            DIA03350
         ELSE IF(YVAL(I).LT.VALMIN) THEN                                DIA03360
            VALMIN = YVAL(I)                                            DIA03370
         ENDIF                                                          DIA03380
   13 CONTINUE                                                          DIA03390
      MAXVAL = VALMAX + 1                                               DIA03400
      MINVAL = VALMIN - 1                                               DIA03410
      SPAC1 = (TIME(NPL) - TIME(1))/10.0                                DIA03420
      SPAC2 = (MAXVAL - MINVAL)/10.0                                    DIA03430
      CALL HWROT('MOVIE')                                               DIA03440
      CALL SWISSM                                                       DIA03450
      CALL PAGE(8.5,11.0)                                               DIA03460
      CALL AREA2D(6.0,8.0)                                              DIA03470
                                                                        DIA03480
```

```
      CALL HEADIN('COMPONENT MODEL DIABETES MELLITUS$',100,1.5,2)      DIA03490
      CALL HEADIN('STUDY OF NORMAL PERSON$',100,-1.15,2)               DIA03500
      CALL XNAME('TIME(HOURS)$',100)                                   DIA03510
      CALL YNAME('BLOOD SUGAR(PPM)$',100)                              DIA03520
      CALL GRAF(TIME(1),SPAC1,TIME(NPL),MINVAL,SPAC2,MAXVAL)           DIA03530
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                      DIA03540
C  RATIONAL SPLINE INTERPOLATION                                       DIA03550
C     CALL RASPLN (1.5)                                                DIA03560
C     CALL THKCRV (.015)                                               DIA03570
C     CALL THKRND (0.)                                                 DIA03580
      CALL CURVE(TIME,YVAL,NPL,0)                                      DIA03590
      CALL ENDPL(0)                                                    DIA03600
      RETURN                                                           DIA03610
      END                                                              DIA03620
                                                                       DIA03630
      SUBROUTINE GRAPH1(YVAL3,TIME,NPL)                                DIA03640
C        This Subroutine Provides A                                    DIA03650
C        Graphical Display Of The                                      DIA03660
C        Component Model. Value Of                                     DIA03670
C        Y Is Plotted Against Time.                                    DIA03680
                                                                       DIA03690
      IMPLICIT REAL*4(A-H,O-Z)                                         DIA03700
      DIMENSION YVAL3(NPL),TIME(NPL)                                   DIA03710
      VALMAX = YVAL3(NPL)                                              DIA03720
      VALMIN = YVAL3(1)                                                DIA03730
      DO 13 I = 1,NPL                                                  DIA03740
         IF (YVAL3(I).GT.VALMAX) THEN                                  DIA03750
            VALMAX = YVAL3(I)                                          DIA03760
         ELSE IF(YVAL3(I).LT.VALMIN) THEN                              DIA03770
            VALMIN = YVAL3(I)                                          DIA03780
         ENDIF                                                         DIA03790
   13 CONTINUE                                                         DIA03800
      MAXVAL = VALMAX + 1                                              DIA03810
      MINVAL = VALMIN - 1                                              DIA03820
      SPAC1 = (TIME(NPL) - TIME(1))/10.0                               DIA03830
      SPAC2 = (MAXVAL - MINVAL)/10.0                                   DIA03840
      CALL HWROT('MOVIE')                                              DIA03850
      CALL SWISSM                                                      DIA03860
      CALL PAGE(8.5,11.0)                                              DIA03870
      CALL AREA2D(6.0,8.0)                                             DIA03880
      CALL HEADIN('COMPONENT MODEL DIABETES MELLITUS$',100,1.5,2)      DIA03890
      CALL HEADIN('STUDY OF NORMAL PERSON$',100,-1.15,2)               DIA03900
      CALL XNAME('TIME (HOURS)$',100)                                  DIA03910
      CALL YNAME('BLOOD INSULIN(PPM)$',100)                            DIA03920
      CALL GRAF(TIME(1),SPAC1,TIME(NPL),MINVAL,SPAC2,MAXVAL)           DIA03930
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                      DIA03940
C  RATIONAL SPLINE INTERPOLATION                                       DIA03950
C     CALL RASPLN (1.5)                                                DIA03960
C     CALL THKCRV (.015)                                               DIA03970
C     CALL THKRND (0.)                                                 DIA03980
      CALL CURVE(TIME,YVAL3,NPL,0)                                     DIA03990
      CALL ENDPL(0)                                                    DIA04000
      RETURN                                                           DIA04010
      END                                                              DIA04020
                                                                       DIA04030
      SUBROUTINE OUTPUT(Y,N,T)                                         DIA04040
C        This Subroutine stores values of Output Variables             DIA04050
C        at communication times. Additionally, it provides             DIA04060
C        an interactive tabular display of the results.                DIA04070
```

```
      IMPLICIT REAL (A-H,O-Z)                                          DIA04080
      DIMENSION Y(N)                                                   DIA04090
      COMMON/DIFF/YVAL(342),YVAL3(342),TIME(342),NPL                   DIA04100
                                                                       DIA04110
      WRITE(6,23) T,Y(1),Y(2)                                          DIA04120
   23 FORMAT(1X,F8.3,5X,F8.3,5X,F8.3)                                  DIA04130
      NPL = NPL + 1                                                    DIA04140
      TIME(NPL) = T                                                    DIA04150
      YVAL(NPL) = Y(1)                                                 DIA04160
      YVAL3(NPL)= Y(2)                                                 DIA04170
      RETURN                                                           DIA04180
      END                                                              DIA04190
                                                                       DIA04200
                                                                       DIA04210
                                                                       DIA04220
      SUBROUTINE INIT                                                  DIA04230
                                                                       DIA04240
      INTEGER NUM,RC,RLEN                                              DIA04250
C     REAL RES1                                                        DIA04260
      CHARACTER*8 KEY,MENUN                                            DIA04270
      CHARACTER*8 FLDN(24)                                             DIA04280
      CHARACTER*4 CC                                                   DIA04290
      COMMON/SAME/PU(24)                                               DIA04300
                                                                       DIA04310
C        Xmenu used for Parameters and  Initial Conditions             DIA04320
                                                                       DIA04330
      FLDN(1) = 'A1'                                                   DIA04340
      FLDN(2) = 'A2'                                                   DIA04350
      FLDN(3) = 'A3'                                                   DIA04360
      FLDN(4) = 'B1'                                                   DIA04370
      FLDN(5) = 'B2'                                                   DIA04380
      FLDN(6) = 'QFOOD1'                                               DIA04390
      FLDN(7) = 'QFOOD2'                                               DIA04400
      FLDN(8) = 'QFOOD3'                                               DIA04410
      FLDN(9) = 'TFOOD1'                                               DIA04420
      FLDN(10)= 'TFOOD2'                                               DIA04430
      FLDN(11)= 'TFOOD3'                                               DIA04440
      FLDN(12)= 'KFOOD1'                                               DIA04450
      FLDN(13)= 'KFOOD2'                                               DIA04460
      FLDN(14)= 'KFOOD3'                                               DIA04470
      FLDN(15)= 'Z'                                                    DIA04480
      FLDN(16)= 'Y1'                                                   DIA04490
      FLDN(17)= 'Y2'                                                   DIA04500
      FLDN(18)= 'TINIT'                                                DIA04510
      FLDN(19)= 'TFIN'                                                 DIA04520
      FLDN(20)= 'COMINT'                                               DIA04530
      FLDN(21)= 'HMIN'                                                 DIA04540
      FLDN(22)= 'HBOUND'                                               DIA04550
      FLDN(23)= 'TOL'                                                  DIA04560
      FLDN(24)= 'HINIT'                                                DIA04570
                                                                       DIA04580
      MENUN = 'DIABETES'                                               DIA04590
      CALL MCLSCR(NUM,RC)                                              DIA04600
      CALL MLOAD(NUM,RC,MENUN)                                         DIA04610
      CALL MPFMAP(NUM,RC,1)                                            DIA04620
      CALL MALARM(NUM,RC,1)                                            DIA04630
      CALL MPRINT(NUM,RC,2,PROUT)                                      DIA04640
      CALL MDSPRD(NUM,RC,KEY)                                          DIA04650
C     CALL MEXIT(NUM,RC)                                               DIA04660
```

```
      IF (KEY .EQ. 'PF10') GOTO 25                                    DIA04670
      IF (KEY .EQ. 'PF12') STOP                                       DIA04680
   25 DO 23 I = 1,17                                                  DIA04690
         CALL MSCR2D(NUM,RC,FLDN(I),CC,4,RLEN)                        DIA04700
         CALL MCVTCF(PU(I),RC,CC,4,1)                                 DIA04710
   23 CONTINUE                                                        DIA04720
      DO 29 I = 18,24                                                 DIA04730
         CALL MSCR2D(NUM,RC,FLDN(I),CC,7,RLEN)                        DIA04740
         CALL MCVTCF(PU(I),RC,CC,7,1)                                 DIA04750
   29 CONTINUE                                                        DIA04760
      RETURN                                                          DIA04770
      END                                                             DIA04780
```



Menu For The Diabetes Mellitus Model: Normal Person

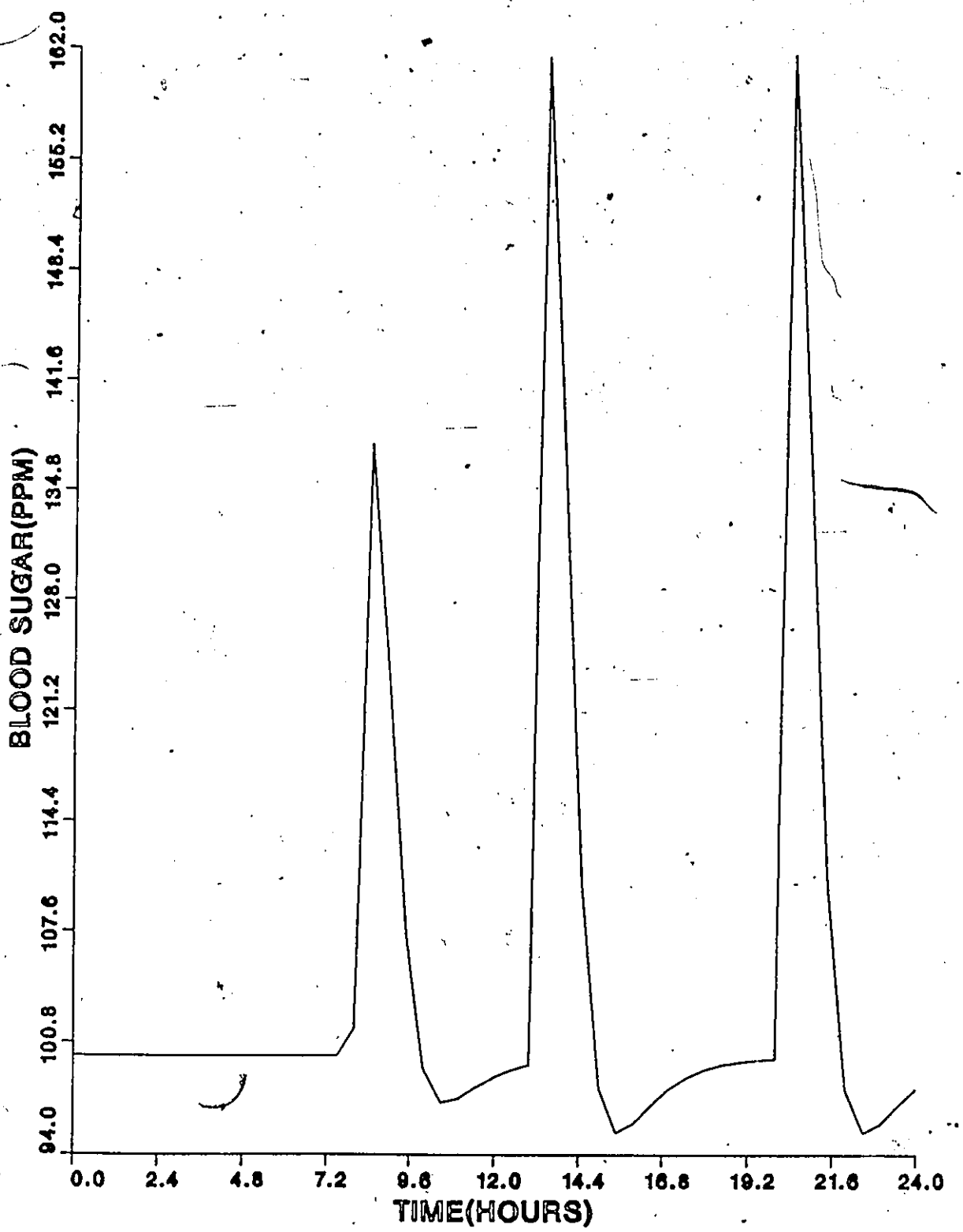
Method: Fixed Step Fourth-Order Runge-Kutta

Problem: Component Model DiabetesMellitus
Study of Normal Person

Time	Blood Sugar	Blood Insulin
0.000	100.000	0.000
0.500	100.000	0.000
1.000	100.000	0.000
1.500	100.000	0.000
2.000	100.000	0.000
2.500	100.000	0.000
3.000	100.000	0.000
3.500	100.000	0.000
4.000	100.000	0.000
4.500	100.000	0.000
5.000	100.000	0.000
5.500	100.000	0.000
6.000	100.000	0.000
6.500	100.000	0.000
7.000	100.000	0.000
7.500	100.000	0.000
8.000	101.667	0.000
8.500	137.675	4.838
9.000	123.274	6.586
9.500	107.539	4.565
10.000	99.275	2.030
10.500	97.199	0.747
11.000	97.436	0.275
11.500	98.125	0.101
12.000	98.744	0.037
12.500	99.194	0.014
13.000	99.495	0.005
13.500	161.532	7.227
14.000	136.847	10.455
14.500	110.583	7.076
15.000	98.069	3.025
15.500	95.374	1.113
16.000	95.928	0.410
16.500	97.060	0.151
17.000	98.041	0.055
17.500	98.747	0.020
18.000	99.216	0.008
18.500	99.515	0.003
19.000	99.703	0.001
19.500	99.819	0.000
20.000	99.889	0.000
20.500	181.794	7.269
21.000	136.907	10.492
21.500	110.560	7.090
22.000	98.038	3.027
22.500	95.353	1.114
23.000	95.915	0.410
23.500	97.051	0.151
24.000	98.036	0.055

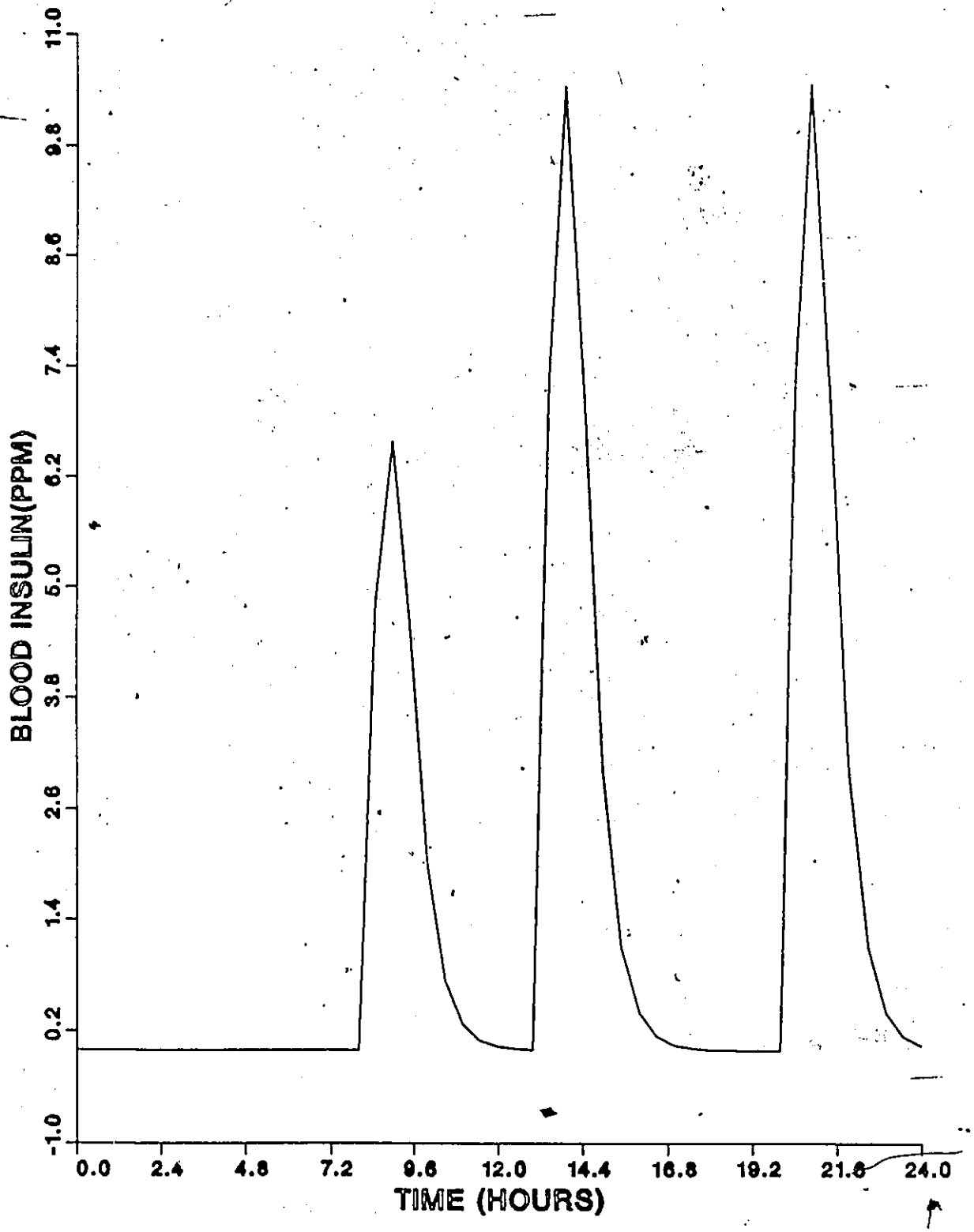
COMPONENT MODEL DIABETES MELLITUS

STUDY OF NORMAL PERSON



COMPONENT MODEL DIABETES MELLITUS

STUDY OF NORMAL PERSON



Appendix B. THE ELECTRONIC OSCILLATOR MODEL

B.1 Problem Definition

This test example is of the electronic oscillator model as described by Palusinski (1985). It consists of six main blocks namely:

- An Oscillator
- A Squaring Operation
- A Sixth-Order Lumped-Parameter Model Of Thermistor Inertia and Non-Linearity
- A Summing Element
- A Square Root Operation
- The Inertia Of Feedback

The oscillator is modelled by the Van der Pol differential equation and the non-linearities are represented by elementary functions of one variable. The resulting single component model is integrated using the Fifth-Order Runge-Kutta Fehlberg over an interval of 20 seconds and the parameter set suggested by Palusinski (1985, p.269). The initial states were assigned the values as specified in the Gest specification of the model in Section B.2.

B.2 Gest Specification of The Electronic Oscillator Model

```
Program StudyOfElectronicOscillator

ContinuousModel ElectronicOscillator
  StaticStructure
    States
        Y1,
        Y2,
        X1,
        X2,
        X3,
        X4,
        X5,
        X6,
        V,
        S
    Outputs
        X3,
        S
    Parameters
        R1,                (* Resistance in kOhms                  *)
        R2,                (* Resistance in kOhms                  *)
        R3,                (* Resistance in kOhms                  *)
        R4,                (* Resistance in kOhms                  *)
        R5,                (* Resistance in kOhms                  *)
        R6,                (* Resistance in kOhms                  *)
        C1,                (* Capacitance in microFarads           *)
        C2,                (* Capacitance in microFarads           *)
        C3,                (* Capacitance in microFarads           *)
        C4,                (* Capacitance in microFarads           *)
        C5,                (* Capacitance in microFarads           *)
        C6                 (* Capacitance in microFarads           *)
  EndStaticStructure
  DynamicStructure
    Derivatives
      X1' = 2*Y1*Y2/R1 + (X2-X1)/R1*C1
      X2' = X1/R2*C1 - (1/R2*C1 + 1/R2*C2)*X2 + X3/R1*C1
      X3' = X2/R3*C2 - (1/R3*C2 + 1/R3*C3)*X3 + X4/R3*C3
      X4' = X3/R4*C3 - (1/R4*C3 + 1/R4*C4)*X4 + X5/R4*C4
      X5' = X4/R5*C4 - (1/R5*C4 + 1/R5*C5)*X5 + X6/R5*C5
      X6' = X5/R6*C5 - (1/R6*C5 + 1/R6*C6)*X6
      Y1' = Y2
      Y2' = -64*Y1 + 7.5*(2 + S. - Y1**2.0) * Y2
       V' = X6/C6
       S' = 0.02*Abs(36 - 5.29 * Sqrt(Abs(V))) ** 1.2 *
            Sign(Abs(36 - 5.29 * Sqrt(Abs(V))) - 0.02*S)
    EndDerivatives
  EndDynamicStructure
EndModel ElectronicOscillator

Parameter Set 1
  Parameters
    R1 = 8.0
    R2 = 8.0
    R3 = 8.0
    R4 = 8.0
    R5 = 8.0
    R6 = 8.0
    C1 = 6.25
    C2 = 4.25
    C3 = 1.25
    C4 = 1.25
```

```
      C5 = 1.25
      C6 = 1.25
    EndParameters
  EndParameter Set 1

  Experiment 1
      TimeUnitIs Second
        StartTime = 0.0
      IntegrateBy Runge-Kutta Fehlberg
        Tol = 0.5
      Communicate at every 0.2 Second
      SimulateUntil Time = 20.0
      InitialStates
        X1 = 80.0
        X2 = 85.0
        X3 = 40.0
        X4 = 45.0
        X5 = 50.0
        X6 = 55.0
        Y1 = 60.0
        Y2 = 70.0
         V = 60.0
         S = 65.0
      EndInitialStates
      SaveOutputs X3,S
    EndContinuousModelElectronicOscillator
  EndExperiment 1

  Run 1
    ToObserve ElectronicOscillator
      WithParameterSet 1
      InExperiment 1
      WithPostRun
        OutputModule 1 On Printer
      EndPostRun
  EndRun 1


  OutputModule 1
    New Page
    Heading ForFirstPage 1 Line
      'Study of ElectronicOscillator'
    List Parameters And InitialConditions
    PlotandList X3,S
  EndOutputModule 1

  Program StudyOfElectronicOscillator
```

```
C  B.3 Equivalent Fortran Program With Results For                    FEL00010
C      Electronic Oscillator Model                                    FEL00020
                                                                      FEL00030
                                                                      FEL00040
                                                                      FEL00050
C                          Variable List                              FEL00060
C                                                                     FEL00070
C Input  : N              --Dimension of the Vector of State Variables FEL00080
C                                                                     FEL00090
C          Y(1)-Y(N)      --Initial Values of State Variables        FEL00100
C                                                                     FEL00110
C          TCOM           --Communication Time                       FEL00120
C                                                                     FEL00130
C          T              --Computational Time                       FEL00140
C                                                                     FEL00150
C          TFIN           --End Time of Simulation                   FEL00160
C                                                                     FEL00170
C                                                                     FEL00180
C          HBOUND         --Upper Bound on Initial Step Size         FEL00190
C                                                                     FEL00200
C          HMIN           --Minimum step size integration is         FEL00210
C                           allowed to proceed at                     FEL00220
C                                                                     FEL00230
C          HINIT          --Starting Integration Step Size           FEL00240
C                                                                     FEL00250
C          H              --Integration Step Size                    FEL00260
C                                                                     FEL00270
C          TOL            --Tolerance or Accuracy specified by user  FEL00280
C                                                                     FEL00290
C          COMINT         --Interval at which output of simulation is FEL00300
C                           required                                  FEL00310
C                                                                     FEL00320
C          ORD            --ORDER OF THE INTEGRATION METHOD          FEL00330
C                                                                     FEL00340
C                                                                     FEL00350
C Output : TIME           --Vector of times of Output Values         FEL00360
C                                                                     FEL00370
C          YVAL           --Vector of values of Y at Output Points   FEL00380
C                                                                     FEL00390
C Program Variables :                                                 FEL00400
C          Y(N)           --State Variables Vector                   FEL00410
C                                                                     FEL00420
C          DY(N)          --Derivative Vector of State Variables     FEL00430
C                                                                     FEL00440
C          DY1(15)-DY4(15)--Working Vectors in Integration Routine   FEL00450
C                             capable of integrating systems with up  FEL00460
C                             to 15 state variables.                  FEL00470
C                                                                     FEL00480
C          F1(15)-F3(15)  --Working Vectors in Integration Routine   FEL00490
C                             capable of integrating systems with up  FEL00500
C                             to 15 state variables.                  FEL00510
C                                                                     FEL00520
C          NFLAG          --Flag Which Signals Collection of Output  FEL00530
C                           Variables.                                FEL00540
C                                                                     FEL00550
C          HPOS           --Intermediate variable used in computing HINIT FEL00560
C                                                                     FEL00570
C          NPL            --Counter used for Graphics Routine.       FEL00580
C                                                                     FEL00590
```

```
C                                                                       FEL00600
C       ERR(N)          --Values of Local Error Estimates of State      FEL00610
C                       ---Variables                                    FEL00620
C                                                                       FEL00630
C       ERRMAX          --Maximum value of Local Error Estimates        FEL00640
C                                                                       FEL00650
C                                                                       FEL00660
C     This Test Program Demonstrates the use of the Variable            FEL00670
C     Step Runge-Kutta Fehlberg Integration Algorithm in the            FEL00680
C     Simulation of a Coupled Model.                                    FEL00690
                                                                        FEL00700
      PROGRAM MAIN                                                      FEL00710
      IMPLICIT REAL*4 (A-H,O-Z)                                         FEL00720
      DIMENSION Y(10),DY(10),ERR(10)                                    FEL00730
      COMMON/DIFF/YVAL(400),TIME(400),YVAL3(400),NPL                    FEL00740
      COMMON/PARS/C1,C2,C3,C4,C5,C6,R1,R2,R3,R4,R5,R6                   FEL00750
      COMMON/XMEN/PU(28)                                                FEL00760
      CHARACTER*1 CHOICE                                                FEL00770
      EXTERNAL DERIV                                                    FEL00780
                                                                        FEL00790
      PRINT*,'Welcome to the Electronic Oscillator Model'               FEL00800
      PRINT*,'    '                                                     FEL00810
      PRINT*,'Would You Like To Perform a Simulation ? (Y/N) '          FEL00820
      READ*, CHOICE                                                     FEL00830
      IF (CHOICE .EQ. 'Y') THEN                                         FEL00840
        CALL INIT                                                       FEL00850
                                                                        FEL00860
                                                                        FEL00870
C                                  Initialization from Menu             FEL00880
C                                                                       FEL00890
C                                     Initialize Parameters             FEL00900
                                                                        FEL00910
199   R1 = PU(1)                                                        FEL00920
      R2 = PU(2)                                                        FEL00930
      R3 = PU(3)                                                        FEL00940
      R4 = PU(4)                                                        FEL00950
      R5 = PU(5)                                                        FEL00960
      R6 = PU(6)                                                        FEL00970
      C1 = PU(7)                                                        FEL00980
      C2 = PU(8)                                                        FEL00990
      C3 = PU(9)                                                        FEL01000
      C4 = PU(10)                                                       FEL01010
      C5 = PU(11)                                                       FEL01020
      C6 = PU(12)                                                       FEL01030
                                                                        FEL01040
C                                     Initialize State Variables        FEL01050
      N    = 10                                                         FEL01060
      Y(1) = PU(13)                                                     FEL01070
      Y(2) = PU(14)                                                     FEL01080
      Y(3) = PU(15)                                                     FEL01090
      Y(4) = PU(16)                                                     FEL01100
      Y(5) = PU(17)                                                     FEL01110
      Y(6) = PU(22)                                                     FEL01120
      Y(7) = PU(18)                                                     FEL01130
      Y(8) = PU(19)                                                     FEL01140
      Y(9) = PU(20)                                                     FEL01150
      Y(10) =PU(21)                                                     FEL01160
C                                     Initialize Run Control            FEL01170
C                                        Variables                      FEL01180
```

```
      T      = PU(23)                                                  FEL01190
      TFIN   = PU(24)                                                  FEL01200
      COMINT= PU(25)                                                   FEL01210
      HBOUND= PU(27)                                                   FEL01220
      HMIN   = PU(28)                                                  FEL01230
      TOL    = PU(26)                                                  FEL01240
      ORD    = 5                                                       FEL01250
                                                                       FEL01260
      NPL    = 1                                                       FEL01270
C                                  Pre-Run Output :                    FEL01280
C                                           Set Up Output Format and OutFEL01290
C                                           put Initial Values of T and FEL01300
      WRITE(8,17)                                                      FEL01310
   17 FORMAT(1X,'Method:',10X,'Variable Step Runge-Kutta Fehlberg'/10X/ FEL01320
     + 1X,'Problem:',10X,'Coupled Model ElectronicOscillator'/10X/1X,  FEL01330
     + 'Time',10X,'Value of X3',10X,'Value of S')                      FEL01340
      PRINT*,'    '                                                    FEL01350
      WRITE(8,21) T,Y(1),Y(6)                                          FEL01360
   21 FORMAT(1X,F8.3,15X,F8.3,15X,F8.3)                                FEL01370
      TIME(1) = T                                                      FEL01380
      YVAL(1) = Y(1)                                                   FEL01390
      YVAL3(1) = Y(6)                                                  FEL01400
      CALL RKFEL(T,Y,N,TFIN,H,ORD,HBOUND,HMIN,COMINT,DERIV,TOL)        FEL01410
                                                                       FEL01420
C                                  Post-Run Output:                    FEL01430
C                                           Tabulate Values of State    FEL01440
C                                           Variables and Time at       FEL01450
C                                           Output Points               FEL01460
      DO 99 I = 2,NPL                                                  FEL01470
         WRITE(8,87)TIME(I),YVAL(I),YVAL3(I)                           FEL01480
   87    FORMAT(1X,F7.2,15X,F8.3,15X,F8.3)                             FEL01490
         PRINT*,'    '                                                FEL01500
   99 CONTINUE                                                         FEL01510
C                                           Plot Values of State        FEL01520
C                                           Variables with Time         FEL01530
      CALL ANYDEV                                                      FEL01540
      CALL GRAPH(YVAL,TIME,NPL)                                        FEL01550
      CALL GRAPH1(YVAL3,TIME,NPL)                                      FEL01560
         PRINT*,'Do you wish to carry out another simulation (Y/N) ?'  FEL01570
         READ*,CHOICE                                                  FEL01580
         IF (CHOICE.EQ.'Y') THEN                                       FEL01590
            CALL INIT                                                  FEL01600
            GOTO 199                                                   FEL01610
          ENDIF                                                        FEL01620
      ENDIF                                                            FEL01630
      STOP                                                             FEL01640
      END                                                              FEL01650
                                                                       FEL03160
                                                                       FEL03170
      SUBROUTINE GRAPH(YVAL ,TIME ,NPL)                                FEL03180
C         This Subroutine Provides A Graphical Display Of The          FEL03190
C         Component Model. Value Of S is Plotted Against Time.         FEL03200
                                                                       FEL03210
      DIMENSION YVAL (NPL),TIME (NPL)                                  FEL03220
      MINT = TIME(1)                                                   FEL03230
      MAXT = TIME(NPL)                                                 FEL03240
      VALMAX = YVAL (NPL)                                              FEL03250
      VALMIN = YVAL (1)                                                FEL03260
      DO 14 I = 1,NPL                                                  FEL03270
```

```
      IF (YVAL (I).GT.VALMAX) THEN                                    FEL03280
         VALMAX = YVAL (I)                                            FEL03290
      ELSE IF (YVAL (I).LT.VALMIN) THEN                               FEL03300
         VALMIN = YVAL (I)                                            FEL03310
      END IF                                                          FEL03320
   14 CONTINUE                                                        FEL03330
      MAXVAL = VALMAX + 1                                             FEL03340
      MINVAL = VALMIN - 1                                             FEL03350
      SPAC1 = (MAXT - MINT)/10.0                                      FEL03360
      SPAC2 = (MAXVAL - MINVAL)/10.0                                  FEL03370
      CALL HWROT('MOVIE')                                             FEL03380
      CALL PAGE(8.5,11.0)                                             FEL03390
      CALL AREA2D(6.0,8.0)                                            FEL03400
      CALL SWISSM                                                     FEL03410
      CALL HEADIN('ELECTRONIC OSCILLATOR MODEL$',100,1.5,2)           FEL03420
      CALL HEADIN('PLOT OF S WITH TIME$',100,-1.15,2)                 FEL03430
      CALL XNAME('TIME(SECS)$',100)                                   FEL03440
      CALL YNAME('VALUE OF S$',100)                                   FEL03450
      CALL GRAF(MINT,SPAC1,MAXT,MINVAL,SPAC2,MAXVAL)                  FEL03460
C  Draw Curve with Rounded and Thickened Line and                     FEL03470
C  Rational Spline Interpolation                                      FEL03480
      CALL RASPLN (1.5)                                               FEL03490
      CALL THKCRV (.015)                                              FEL03500
      CALL THKRND (0.)                                                FEL03510
      CALL CURVE(TIME ,YVAL ,NPL,1)                                   FEL03520
      CALL ENDPL(0)                                                   FEL03530
      RETURN                                                          FEL03540
      END                                                             FEL03550
                                                                      FEL03560
                                                                      FEL03570
      SUBROUTINE GRAPH1(YVAL3,TIME1,NPL)                              FEL03580
C          This Subroutine Provides A Graphical Display Of The        FEL03590
C          Component Model. Value Of S is Plotted Against Time.       FEL03600
                                                                      FEL03610
      DIMENSION YVAL3(NPL),TIME1(NPL)                                 FEL03620
      MINT = TIME1(1)                                                 FEL03630
      MAXT = TIME1(NPL)                                               FEL03640
      VALMAX = YVAL3(NPL)                                             FEL03650
      VALMIN = YVAL3(1)                                               FEL03660
      DO 14 I = 1,NPL                                                 FEL03670
         IF (YVAL3(I).GT.VALMAX) THEN                                 FEL03680
            VALMAX = YVAL3(I)                                         FEL03690
         ELSE IF (YVAL3(I).LT.VALMIN) THEN                            FEL03700
            VALMIN = YVAL3(I)                                         FEL03710
         END IF                                                       FEL03720
   14 CONTINUE                                                        FEL03730
      MAXVAL = VALMAX + 1                                             FEL03740
      MINVAL = VALMIN - 1                                             FEL03750
      SPAC1 = (MAXT - MINT)/10.0                                      FEL03760
      SPAC2 = (MAXVAL - MINVAL)/10.0                                  FEL03770
      CALL HWROT('MOVIE')                                             FEL03780
      CALL PAGE(8.5,11.0)                                             FEL03790
      CALL AREA2D(6.0,8.0)                                            FEL03800
      CALL SWISSM                                                     FEL03810
      CALL HEADIN('STUDY OF ELECTRONIC OSCILLATOR MODEL$',100,1.5,2)  FEL03820
      CALL HEADIN('PLOT OF X3 WITH TIME$',100,-1.15,2)                FEL03830
      CALL XNAME('TIME(SECS)$',100)                                   FEL03840
      CALL YNAME('VALUE OF X3$',100)                                  FEL03850
      CALL GRAF(MINT,SPAC1,MAXT,MINVAL,SPAC2,MAXVAL)                  FEL03860
```

```
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                        FEL03870
C  RATIONAL SPLINE INTERPOLATION                                         FEL03880
      CALL RASPLN (1.5)                                                 FEL03890
      CALL THKCRV (.015)                                                FEL03900
      CALL THKRND (0.)                                                  FEL03910
      CALL CURVE(TIME1,YVAL3,NPL,1)                                     FEL03920
      CALL ENDPL(0)                                                     FEL03930
      RETURN                                                            FEL03940
      END                                                               FEL03950
                                                                        FEL03960
                                                                        FEL03970
      SUBROUTINE OUTPUT(Y,N,T)                                          FEL03980
C        This Subroutine stores values of Output Variables              FEL03990
C        at communication times. Additionally, it provides              FEL04000
C        an interactive tabular display of the results.                 FEL04010
                                                                        FEL04020
      IMPLICIT REAL*4 (A-H,O-Z)                                         FEL04030
      DIMENSION Y(N)                                                    FEL04040
      COMMON/DIFF/YVAL(400),TIME(400),YVAL3(400),NPL                    FEL04050
                                                                        FEL04060
      WRITE(6,23) T,Y(1),Y(6)                                           FEL04070
   23 FORMAT(1X,F5.2,3X,F6.3,3X,F9.3)                                   FEL04080
      NPL = NPL + 1                                                     FEL04090
      TIME(NPL) = T                                                     FEL04100
      YVAL(NPL) = Y(1)                                                  FEL04110
      YVAL3(NPL) = Y(6)                                                 FEL04120
      RETURN                                                            FEL04130
      END                                                               FEL04140
                                                                        FEL04150
                                                                        FEL04160
      SUBROUTINE DERIV(T,Y,DY,N)                                        FEL04170
      IMPLICIT REAL*4 (A-H,O-Z)                                         FEL04180
      DIMENSION Y(N),DY(N)                                              FEL04190
      COMMON/PARS/C1,C2,C3,C4,C5,C6,R1,R2,R3,R4,R5,R6                   FEL04200
                                                                        FEL04210
C                                Subroutine consists of Slow System     FEL04220
C                                Derivatives which are specified by     FEL04230
C                                the user for his particular model.     FEL04240
      DY(1) = Y(10)/(R3*C2)-(1/(R3*C2)+1/(R3*C3))*Y(1) +                FEL04250
     +        Y(2)/(R3*C3)                                              FEL04260
      DY(2) = Y(1)/(R4*C3)-(1/(R4*C3)+1.0/(R4*C4))*Y(2) + Y(3)/(R4*C4)  FEL04270
                                                                        FEL04280
      DY(3) = Y(2)/(R5*C4)-(1/(R5*C4)+1/(R5*C5))*Y(3) + Y(4)/(R5*C5)    FEL04290
                                                                        FEL04300
      DY(4) = Y(3)/(R6*C5) - (1.0/(R6*C5) + 1.0/(R6*C6))*Y(4)           FEL04310
                                                                        FEL04320
      DY(5) = Y(4)/C6                                                   FEL04330
                                                                        FEL04340
      DY(6) = 0.02 * ABS(36.0 - 5.29*SQRT(ABS(Y(5))))**1.2*             FEL04350
     +        SIGN(1.0,ABS(36.0-5.29*SQRT(ABS(Y(5))))) - 0.02*Y(6)      FEL04360
                                                                        FEL04370
      DY(7) = Y(8)                                                      FEL04380
                                                                        FEL04390
      DY(8) = -64.0*Y(7) + 7.5*(2.0+Y(6)-(Y(7)*Y(7)))*Y(8)              FEL04400
                                                                        FEL04410
      DY(9) = 2.0*Y(7)*Y(8)/R1 + (Y(10)-Y(9))/(R1*C1)                   FEL04420
                                                                        FEL04430
      DY(10)=Y(9)/(R2*C1)-Y(10)*(1.0/(R2*C1)+1.0/(R2*C2))+Y(1)/(R1*C1)  FEL04440
                                                                        FEL04450
```

```
      RETURN                                                          FEL04460
      END                                                             FEL04470
                                                                      FEL04480
                                                                      FEL04490
                                                                      FEL04500
                                                                      FEL04510
      SUBROUTINE INIT                                                 FEL04520
                                                                      FEL04530
      INTEGER NUM,RC,RLEN                                             FEL04540
      CHARACTER*8 KEY,MENUN                                           FEL04550
      CHARACTER*8 FLDN(28)                                            FEL04560
      CHARACTER*4 CC                                                  FEL04570
      COMMON/XMEN/PU(28)                                              FEL04580
                                                                      FEL04590
C     Xmenu used for Parameters and  Initial Conditions               FEL04600
                                                                      FEL04610
      FLDN(1) = 'R1'                                                  FEL04620
      FLDN(2) = 'R2'                                                  FEL04630
      FLDN(3) = 'R3'                                                  FEL04640
      FLDN(4) = 'R4'                                                  FEL04650
      FLDN(5) = 'R5'                                                  FEL04660
      FLDN(6) = 'R6'                                                  FEL04670
      FLDN(7) = 'C1'                                                  FEL04680
      FLDN(8) = 'C2'                                                  FEL04690
      FLDN(9) = 'C3'                                                  FEL04700
      FLDN(10) = 'C4'                                                 FEL04710
      FLDN(11) = 'C5'                                                 FEL04720
      FLDN(12) = 'C6'                                                 FEL04730
      FLDN(13) = 'X3'                                                 FEL04740
      FLDN(14) = 'X4'                                                 FEL04750
      FLDN(15) = 'X5'                                                 FEL04760
      FLDN(16) = 'X6'                                                 FEL04770
      FLDN(17) = 'V'                                                  FEL04780
      FLDN(18) = 'Y1'                                                 FEL04790
      FLDN(19) = 'Y2'                                                 FEL04800
      FLDN(20) = 'X1'                                                 FEL04810
      FLDN(21) = 'X2'                                                 FEL04820
      FLDN(22) = 'S'                                                  FEL04830
      FLDN(23) = 'TINIT'                                              FEL04840
      FLDN(24) = 'TFIN'                                               FEL04850
      FLDN(25) = 'COMINT'                                             FEL04860
      FLDN(26) = 'TOL'                                                FEL04870
      FLDN(27) = 'HBOUND'                                             FEL04880
      FLDN(28) = 'HMIN'                                               FEL04890
                                                                      FEL04900
      MENUN = 'OSSIE'                                                 FEL04910
      CALL MCLSCR(NUM,RC)                                             FEL04920
      CALL MLOAD(NUM,RC,MENUN)                                        FEL04930
      CALL MPFMAP(NUM,RC,1)                                           FEL04940
      CALL MALARM(NUM,RC,1)                                           FEL04950
      CALL MPRINT(NUM,RC,2,PROUT)                                     FEL04960
      CALL MDSPRD(NUM,RC,KEY)                                         FEL04970
C     CALL MEXIT(NUM,RC)                                              FEL04980
      IF (KEY .EQ. 'PF10') GOTO 25                                    FEL04990
      IF (KEY .EQ. 'PF12') STOP                                       FEL05000
   25 DO 23 I = 1,22                                                  FEL05010
         CALL MSCR2D(NUM,RC,FLDN(I),CC,4,RLEN)                        FEL05020
         CALL MCVTCF(PU(I),RC,CC,4,2)                                 FEL05030
   23 CONTINUE                                                        FEL05040
```

```
29  DO 33 I = 23,28                                                  FEL05050
        CALL MSCR2D(NUM,RC,FLDN(I),CC,7,RLEN)                         FEL05060
        CALL MCVTCF(PU(I),RC,CC,7,2)                                  FEL05070
33    CONTINUE                                                        FEL05080
      RETURN                                                          FEL05090
    . END                                                             FEL05100
                                                                      FEL05110
                                                                      FEL05120
```

```
sk 10
    +....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8+
 1  |                              INITIALIZATION                                      |  1
 2  |                              --------------                                      |  2
 3  |                                                                                  |  3
 4  |                          KINDLY INITIALIZE VARIABLES                             |  4
 5  |                                                                                  |  5
 6  |      Model Parameters                                  Experimentation Variables |  6
 7  |                                                                                  |  7
 8  |                                        Initial States          Control Variables |  8
 9  |                                                                                  |  9
10  |                                                                                  | 10
11  | C1 = 6.25 mF  R1 = 8.00 kOhms X3(0) = 40.0 Y1(0) = 20.0  TINIT  = 0.00           | 11
12  |                                                                                  | 12
13  | C2 = 4.25 mF  R2 = 8.00 kOhms X4(0) = 45.0 Y2(0) = 70.0   TFIN  = 20.0           | 13
14  |                                                                                  | 14
15  | C3 = 1.25 mF  R3 = 8.00 kOhms X5(0) = 50.0 X1(0) = 80.0  COMINT = 0.20           | 15
16  |                                                                                  | 16
17  | C4 = 1.25 mF  R4 = 8.00 kOhms X6(0) = 55.0 X2(0) = 85.0    TOL  = 0.50000        | 17
18  |                                                                                  | 18
19  | C5 = 1.25 mF  R5 = 8.00 kOhms V(0) = 60.0   S(0) = 65.0  HBOUND= 0.10            | 19
20  |                                                                                  | 20
21  | C6 = 1.25 mF  R6 = 8.00 kOhms                             HMIN = 0.00001         | 21
22  |                                                                                  | 22
23  |        Press:  PF10 - to start simulation    PF12 - to quit                      | 23
24  |                                                                                  | 24
    +....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8+
```

Menu For Electronic Oscillator Model
Using Runge-Kutta Fehlberg Algorithm

Method: Variable Step Runge-Kutta Fehlberg

Problem: Coupled Model ElectronicOscillator

Time	Value of X3	Value of S
0.00	40.000	65.000
0.20	40.358	64.777
0.40	40.705	64.574
0.60	41.041	64.392
0.80	41.366	64.229
1.00	41.682	64.086
1.20	41.989	63.960
1.40	42.286	63.852
1.60	42.573	63.761
1.80	42.852	63.686
2.00	43.124	63.627
2.20	43.387	63.583
2.40	43.641	63.553
2.60	43.888	63.538
2.80	44.129	63.537
3.00	44.360	63.548
3.20	44.585	63.573
3.40	44.804	63.609
3.60	45.015	63.658
3.80	45.220	63.718
4.00	45.419	63.790
4.20	45.611	63.873
4.40	45.797	63.967
4.60	45.976	64.070
4.80	46.151	64.184
5.00	46.319	64.308
5.20	46.481	64.441
5.40	46.638	64.584
5.60	46.790	64.735
5.80	46.936	64.896
6.00	47.077	65.065
6.20	47.214	65.243
6.40	47.344	65.428
6.60	47.470	65.622
6.80	47.592	65.823
7.00	47.708	66.033
7.20	47.820	66.248
7.40	47.927	66.472
7.60	48.030	66.704
7.80	48.128	66.941
8.00	48.222	67.184
8.20	48.312	67.436
8.40	48.398	67.694
8.60	48.480	67.958
8.80	48.557	68.227
9.00	48.630	68.503
9.20	48.700	68.787
9.40	48.766	69.074

9.60	48.828	69.366
9.80	48.887	69.668
10.00	48.942	69.970
10.20	48.993	70.280
10.40	49.041	70.595
10.60	49.086	70.917
10.80	49.127	71.242
11.00	49.165	71.573
11.20	49.199	71.906
11.40	49.231	72.250
11.60	49.259	72.592
11.81	49.284	72.949
12.01	49.306	73.303
12.20	49.324	73.651
12.40	49.340	74.012
12.60	49.353	74.378
12.80	49.364	74.748
13.00	49.373	75.122
13.20	49.380	75.501
13.40	49.384	75.882
13.60	49.387	76.267
13.80	49.387	76.657
14.00	49.385	77.052
14.20	49.382	77.447
14.40	49.376	77.847
14.60	49.368	78.252
14.80	49.358	78.661
15.00	49.346	79.069
15.20	49.333	79.484
15.40	49.317	79.901
15.60	49.300	80.323
15.80	49.280	80.748
16.00	49.260	81.172
16.20	49.237	81.603
16.40	49.212	82.040
16.60	49.186	82.476
16.80	49.158	82.916
17.00	49.128	83.358
17.20	49.097	83.802
17.40	49.064	84.249
17.60	49.030	84.700
17.80	48.994	85.155
18.00	48.956	85.611
18.20	48.917	86.076
18.40	48.877	86.527
18.60	48.835	86.991
18.80	48.790	87.460
19.00	48.745	87.928
19.20	48.699	88.399
19.40	48.653	88.873
19.60	48.605	89.349
19.80	48.557	89.828
20.00	48.535	90.122

Appendix C. THE RUDDER CONTROL MODEL

C.1 Problem Definition

This test example uses the Adams Predictor Corrector Method to carry out the integration. The example is adapted from the article by Elzas (1979). As can be seen from the results, the control responds quickly and the trajectories follow closely within the first one-tenth of a second. The full specification of the model is shown in Section C.2.

C.2 Gest Specification of The Rudder Control Model

```
Program StudyOfRudderControl

ContinuousModel RudderControl
  StaticStructure
    States Y1, Y2
      RangeOf  Y1 = ( >= -10.0, <=10.0 )
      RangeOf  Y2 = ( >= -20.0, <=20.0 )
    Outputs Y1,Y2
    Parameters A
  EndStaticStructure

  DynamicStructure
    Derivatives
      Y1' = Y2
      Y2' = A*( 1.0 - Y1**2 - Y2**2 )*( Y2 - Y1 )
    EndDerivatives
  EndDynamicStructure
EndModel RudderControl


ParameterSet 1
  Model RudderControl
    Parameter
      A = 5.0
    EndParameter
  EndModel RudderControl
EndParameterSet 1


Experiment 1
    TimeUnitIs  Second
      StartTime = 0.0
    IntegrateBy  AdamsPredictorCorrector
      RelError = 0.00001
    Communicate at every 0.10 Second
    SimulateUntil Time = 1.0

  Model RudderControl
    InitialStates
      Y1  =  2.0
      Y2  = -2.0
    EndInitialStates
    SaveOutputs  Y1, Y2
  EndModel RudderControl
EndExperiment 1


Run 1
  ToObserve RudderControl
    WithParameterSet 1
    InExperiment 1
    WithPostRun
      OutputModule 1 On Printer
    EndPostRun
EndRun 1


OutputModule 1
  New Page
  Heading ForFirstPage 1 Line
    Study of RudderControl
```

```
  List Parameters And InitialConditions
  PlotandList  Y1,Y2
EndOutputModule 1

EndProgram StudyOfRudderControl
```

```
C  C.3    Equivalent Procedural Program With Results                    ADA00020
C         For Rudder Control Model                                      ADA00030
                                                                        ADA00040
                                                                        ADA00050
C                          Variable List                                ADA00060
C                                                                       ADA00070
C Input  : N              --Dimension of the Vector of State Variables  ADA00080
C                                                                       ADA00090
C          Y(K,1)-Y(K,N)--Initial Values of State Variables           ADA00100
C                                                                       ADA00110
C          T              --Computational Time                         ADA00120
C                                                                       ADA00130
C          TINIT          --Initial Time                               ADA00140
C                                                                       ADA00150
C          TFIN           --End Time of Simulation                     ADA00160
C                                                                       ADA00170
C          TCOM           --Communication Time                         ADA00180
C                                                                       ADA00190
C          COMINT         --Communication Interval                     ADA00200
C                                                                       ADA00210
C                                                                       ADA00220
C                                                                       ADA00230
C          H              --Integration Step Size                      ADA00240
C                                                                       ADA00250
C          HINIT          --Starting Integration Step Size             ADA00260
C                                                                       ADA00270
C          HBOUND         --Upper Bound On Initial Step Size           ADA00280
C                                                                       ADA00290
C          HMIN           --Minimum step size integration is           ADA00300
C                           allowed to proceed at                       ADA00310
C                                                                       ADA00320
C          TOL            --Tolerance or Accuracy specified by user    ADA00330
C                                                                       ADA00340
C          NORD           --Order of the Integration Method            ADA00350
C                                                                       ADA00360
COUTPUT :T.               --Vector of times of Output Values           ADA00370
C                                                                       ADA00380
C        Y(K,1)-Y(K,N)    --Vector of Values of Y at Output Points     ADA00390
C                                                                       ADA00400
CPROGRAM VARIABLES :                                                    ADA00410
C        Y(K,N)           --State Variables Vector                     ADA00420
C                                                                       ADA00430
C        DY(N)            --Derivative Vector of State Variables       ADA00440
C                                                                       ADA00450
C        VP(N)            --Vector of Predictor Variables              ADA00460
C                                                                       ADA00470
C        VC(N)            --Vector of Corrector Variables              ADA00480
C                                                                       ADA00490
C        ERR(N)           --Vector of Local Error Estimates            ADA00500
C                                                                       ADA00510
C        ERRMAX           --Variable containing Maximum Error          ADA00520
C                                                                       ADA00530
C        DY1(N)-DY4(N)-Working Vectors in R-K Starter Routine          ADA00540
C                                                                       ADA00550
C        P1(N)-P3(N)  --Working Vectors in R-K Starter Routine         ADA00560
C                                                                       ADA00570
C        T1-T3          --Variables storing three time values generated by ADA00580
C                         the R-K Starter                               ADA00590
```

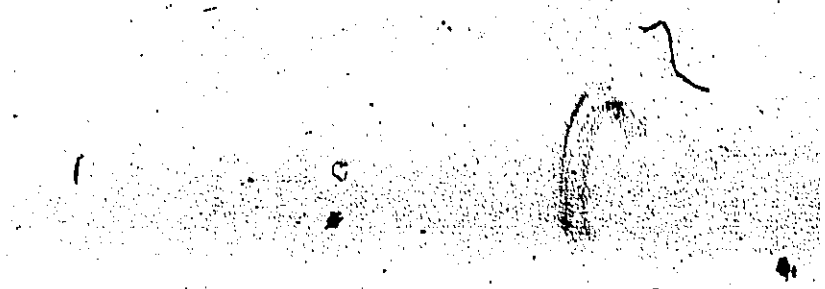
```
C                                                                       ADA00600
C       Y1-Y3        --Variables storing three values of State VariablesADA00610
C                       generated by the R-K Starter.                   ADA00620
C                                                                       ADA00630
C       HPOS         --Intermediate variable used in determining HINIT  ADA00640
C                                                                       ADA00650
C       NPL          --Counter used for Graphics Routine                ADA00660
C                                                                       ADA00670
C       NSTEPS       --Number Of Steps For Stiffness Check              ADA00680
C                                                                       ADA00690
C       NCHECK       --Actual Number Of Steps Taken During Simulation   ADA00700
C                                                                       ADA00710
C       NFLAG        --Signals A Communication Time                     ADA00720
C                                                                       ADA00730
C                                                                       ADA00740
      PROGRAM MAIN                                                      ADA00750
C       This test program illustrates                                   ADA00760
C       the utilization of the Fourth-                                  ADA00770
C       Order Variable Adams Method                                     ADA00780
C       in the simulation of a single                                   ADA00790
C       Component Model                                                 ADA00800
                                                                        ADA00810
C                                  VARIABLE DECLARATION                 ADA00820
      IMPLICIT REAL*4(A-H,O-Z)                                          ADA00830
      DIMENSION T(6000),Y(6000,5)                                       ADA00840
      COMMON/DIFF/ YVAL1(300),YVAL2(300),TIME(300),NPL                  ADA00850
      COMMON/PAR/A                                                      ADA00860
      COMMON/SAME/PU(9)                                                 ADA00870
      EXTERNAL DERIV                                                    ADA00880
      CHARACTER*1 CHOICE                                                ADA00890
                                                                        ADA00900
      PRINT*,'Welcome To The Rudder Control Simulator'                  ADA00910
      PRINT*,'    '                                                     ADA00920
      PRINT*,'Would You Like To Perform a Simulation ? (Y/N) '          ADA00930
      READ*, CHOICE                                                     ADA00940
      IF (CHOICE .EQ. 'Y') THEN                                         ADA00950
200   CALL INIT                                                         ADA00960
                                                                        ADA00970
C                                Initialization from Menu               ADA00980
C                                        Initialize Parameter           ADA00990
199   A = PU(1)                                                         ADA01000
                                                                        ADA01010
C                                        Initialize State Variables     ADA01020
      N = 2                                                             ADA01030
      Y(1,1) =  PU(2)                                                   ADA01040
      Y(1,2) =  PU(3)                                                   ADA01050
      IF ((Y(1,1).LT.-10.0).OR. (Y(1,1).GT.10.0)) THEN                  ADA01060
         PRINT*,' Sorry,This Model Requires Y to be greater than or     ADA01070
     +equal to -10 and less than or equal to 10. Kindly                 ADA01080
     +Enter "Y" and Reinitialize Y  '                                   ADA01090
         READ*, CHOICE                                                  ADA01100
         IF (CHOICE .EQ. 'Y') GOTO 200                                  ADA01110
      ENDIF                                                             ADA01120
      IF ((Y(1,2).LT.-20.0).OR. (Y(1,2).GT.20.0)) THEN                  ADA01130
         PRINT*,' Sorry,This Model Requires Y1 to be greater than or    ADA01140
     +equal to -20 and less than or equal to 20. Kindly                 ADA01150
     +Enter "Y" and Reinitialize Y1 '                                   ADA01160
         READ*, CHOICE                                                  ADA01170
         IF (CHOICE .EQ. 'Y') GOTO 200                                  ADA01180
```

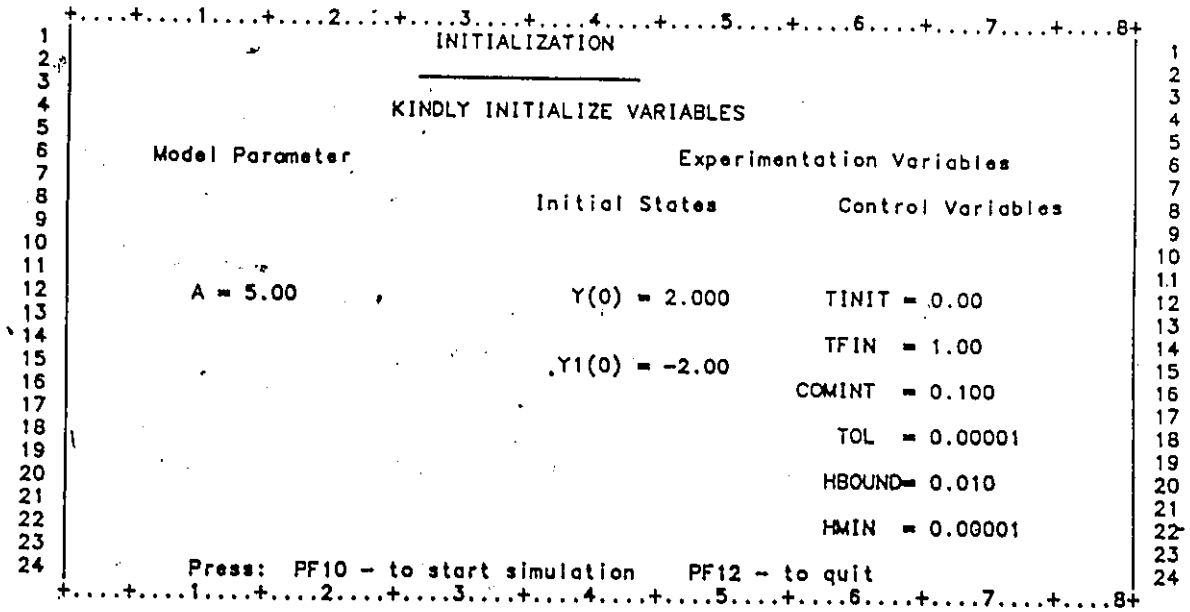
```
      ENDIF                                                              ADA01190
C                                        Initialize Run Control Variables ADA01200
      T(1)   = PU(4)                                                     ADA01210
      TFIN   = PU(5)                                                     ADA01220
      COMINT= PU(6)                                                      ADA01230
      HBOUND= PU(7)                                                      ADA01240
      HMIN   = PU(8)                                                     ADA01250
C                                      Tolerance Specified by user for  ADA01260
C                                      computation of initial step size ADA01270
C                                                                       ADA01280
      TOL    = PU(9)                                                     ADA01290
                                                                         ADA01300
                                                                         ADA01310
                                                                         ADA01320
                                                                         ADA01330
      NPL    = 1                                                         ADA01340
      ORD    = 4                                                         ADA01350
C                                      Pre-Run Output :                 ADA01360
C                                               Set up Output Format and ADA01370
C                                               Output Initial Values of ADA01380
C                                               T(1) and Y(1,1)          ADA01390
      WRITE(8,17)                                                        ADA01400
   17 FORMAT(1X,'Method:',19X,'Variable Step Adams'/1X,'Time',17X,       ADA01410
     +       'VALUE OF Y1',7X,'VALUE OF Y2')                             ADA01420
      PRINT*,'   '                                                      ADA01430
      WRITE(8,21) T(1),Y(1,1),Y(1,2)                                     ADA01440
   21 FORMAT(1X,F5.2,5X,F8.3,5X,F8.3)                                    ADA01450
      PRINT*,' '                                                         ADA01460
      TIME(NPL) = T(1)                                                   ADA01470
      YVAL1(NPL)= Y(1,1)                                                 ADA01480
      YVAL2(NPL)= Y(1,2)                                                 ADA01490
C                                    Run-Time Activities                ADA01500
C                                         Integrate by Fourth-Order     ADA01510
C                                         Adams Predictor Method        ADA01520
                                                                         ADA01530
                                                                         ADA01540
      CALL ADAMS(T,Y,N,TFIN,H,ORD,HBOUND,HMIN,DERIV,COMINT,TOL,K)        ADA01550
C                                                                       ADA01560
C                                      Post-Run Output:                 ADA01570
C                                               Tabulate Values of State ADA01580
C                                               Variables and Time at    ADA01590
C                                               Output Points.           ADA01600
C                                                                       ADA01610
      DO 99 I = 2,NPL                                                    ADA01620
         WRITE(8,87)TIME(I),YVAL1(I),YVAL2(I)                           ADA01630
   87    FORMAT(1X,F5.2,5X,F8.3,5X,F8.3)                                ADA01640
         PRINT*,'   '                                                  ADA01650
   99 CONTINUE                                                          ADA01660
                                                                        ADA01670
C                                               Plot Values of State    ADA01680
C                                               Variables with Time     ADA01690
      CALL ANYDEV                                                       ADA01700
      CALL GRAPH(YVAL1,YVAL2,TIME,NPL)                                  ADA01710
      PRINT*,'Do you wish to carry out another simulation (Y/N) ?'      ADA01720
      READ*,CHOICE                                                      ADA01730
      IF (CHOICE.EQ.'Y') THEN                                           ADA01740
         CALL INIT                                                      ADA01750
         GOTO 199                                                       ADA01760
      ENDIF                                                             ADA01770
```

```
      ENDIF                                                              ADA01780
      STOP                                                               ADA01790
      END                                                                ADA01800
                                                                         ADA01810
                                                                         ADA04330
      SUBROUTINE DERIV(T,Y,DY,N,K)                                       ADA04340
      IMPLICIT REAL*4(A-H,O-Z)                                           ADA04350
      DIMENSION Y(6000,5),DY(6000,5),T(6000)                             ADA04360
      COMMON/PAR/A                                                       ADA04370
C         This is the derivative evaluation                              ADA04380
C         subroutine where the user specifies                            ADA04390
C         the system derivatives that describe                           ADA04400
C         his model. As an example, the model                            ADA04410
C         Rudder Control is used.                                        ADA04420
                                                                         ADA04430
      DY(K,1) = Y(K,2)                                                   ADA04440
      DY(K,2) = A*(1.0-(Y(K,1)*Y(K,1))-(Y(K,2)*Y(K,2))) *                ADA04450
     +         (Y(K,2) - Y(K,1))                                         ADA04460
                                                                         ADA04470
C        WRITE (6,23) DY(K,1),DY(K,2),Y(K,1),Y(K,2),K,T(K-1)             ADA04480
C 23     FORMAT(1X,F9.4,1X,F9.4,1X,F9.4,1X,F9.4,1X,I3,1X,F9.4)           ADA04490
                                                                         ADA04500
                                                                         ADA04510
                                                                         ADA04520
                                                                         ADA04530
      RETURN                                                             ADA04540
      END                                                                ADA04550
                                                                         ADA04560
                                                                         ADA04920
      SUBROUTINE GRAPH(YVAL1,YVAL2,TIME1,NPL)                            ADA04930
C         This Subroutine Provides A                                     ADA04940
C         Graphical Display Of The                                       ADA04950
C         Component Model. Value Of                                      ADA04960
C         Y Is Plotted Against Time.                                     ADA04970
                                                                         ADA04980
      IMPLICIT REAL*4(A-H,O-Z)                                           ADA04990
      DIMENSION YVAL1(NPL),YVAL2(NPL),TIME1(NPL)                         ADA05000
      VALMAX = YVAL2(NPL)                                                ADA05010
      VALMIN = YVAL2(1)                                                  ADA05020
      DO 13 I = 1,NPL                                                    ADA05030
         IF (YVAL2(I).GT.VALMAX) THEN                                    ADA05040
            VALMAX = YVAL2(I)                                            ADA05050
         ELSE IF(YVAL2(I).LT.VALMIN) THEN                                ADA05060
            VALMIN = YVAL2(I)                                            ADA05070
         ENDIF                                                           ADA05080
   13 CONTINUE                                                           ADA05090
      MAXVAL = VALMAX + 1                                                ADA05100
      MINVAL = VALMIN - 1                                                ADA05110
      SPAC1 = (TIME1(NPL) - TIME1(1))/10.0                               ADA05120
      SPAC2 = (MAXVAL - MINVAL)/10.0                                     ADA05130
      CALL HWROT('MOVIE')                                                ADA05140
      CALL SWISSM                                                        ADA05150
      CALL PAGE(8.5,11.0)                                                ADA05160
      CALL AREA2D(6.0,8.0)                                               ADA05170
      CALL HEADIN('Study Of Rudder Control Model$',100,1.5,3)            ADA05180
      CALL HEADIN('Adams Predictor Corrector Method$',100,1.25,3)        ADA05190
      CALL HEADIN('Fourth-Order$',100,-1.15,3)                           ADA05200
      CALL XNAME('Time(seconds)$',100)                                   ADA05210
      CALL YNAME('Values Of Y$',100)                                     ADA05220
```

```
      CALL GRAF(TIME1(1),SPAC1,TIME1(NPL),MINVAL,SPAC2,MAXVAL)          ADA05230
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                       ADA05240
C  RATIONAL SPLINE INTERPOLATION                                        ADA05250
C     CALL RASPLN (1.5)                                                 ADA05260
      CALL THKCRV (.015)                                                ADA05270
      CALL THKRND (0.)                                                  ADA05280
      CALL CURVE(TIME1,YVAL1,NPL,1)                                     ADA05290
      CALL CURVE(TIME1,YVAL2,NPL,1)                                     ADA05300
      CALL ENDPL(0)                                                     ADA05310
      RETURN                                                            ADA05320
      END                                                               ADA05330
                                                                        ADA05340
      SUBROUTINE OUTPUT(Y,N,TNOW,K)                                     ADA05350
C        This Subroutine stores values of Output Variables              ADA05360
C        at communication times. Additionally, it provides              ADA05370
C        an interactive tabular display of the results.                 ADA05380
                                                                        ADA05390
      IMPLICIT REAL*4 (A-H,O-Z)                                         ADA05400
      DIMENSION Y(6000,N)                                               ADA05410
      COMMON/DIFF/YVAL1(300),YVAL2(300),TIME(300),NPL                   ADA05420
                                                                        ADA05430
      WRITE(8,23) TNOW,Y(K-1,1),Y(K-1,2)                                ADA05440
   23 FORMAT(1X,F8.3,5X,F8.3,5X,F8.3)                                   ADA05450
      NPL = NPL + 1                                                     ADA05460
      TIME(NPL) = TNOW                                                  ADA05470
      YVAL1(NPL)= Y(K-1,1)                                              ADA05480
      YVAL2(NPL)= Y(K-1,2)                                              ADA05490
      RETURN                                                            ADA05500
      END                                                               ADA05510
                                                                        ADA05520
                                                                        ADA05530
      SUBROUTINE INIT                                                   ADA05540
                                                                        ADA05550
      INTEGER NUM,RC,RLEN                                               ADA05560
C     REAL RES1                                                         ADA05570
      CHARACTER*8 KEY,MENUN                                             ADA05580
      CHARACTER*8 FLDN(9)                                               ADA05590
      CHARACTER*4 CC                                                    ADA05600
      COMMON/SAME/PU(9)                                                 ADA05610
                                                                        ADA05620
C        Xmenu used for Parameters and  Initial Conditions              ADA05630
                                                                        ADA05640
      FLDN(1) = 'A'                                                     ADA05650
      FLDN(2) = 'Y'                                                     ADA05660
      FLDN(3) = 'Y1'                                                    ADA05670
      FLDN(4) = 'TINIT'                                                 ADA05680
      FLDN(5) = 'TFIN'                                                  ADA05690
      FLDN(6) = 'COMINT'                                                ADA05700
      FLDN(7) = 'HBOUND'                                                ADA05710
      FLDN(8) = 'HMIN'                                                  ADA05720
      FLDN(9) = 'TOL'                                                   ADA05730
                                                                        ADA05740
      MENUN = 'RUDDER'                                                  ADA05750
      CALL MCLSCR(NUM,RC)                                               ADA05760
      CALL MLOAD(NUM,RC,MENUN)                                          ADA05770
      CALL MPFMAP(NUM,RC,1)                                             ADA05780
      CALL MALARM(NUM,RC,1)                                             ADA05790
      CALL MPRINT(NUM,RC,2,PROUT)                                       ADA05800
      CALL MDSPRD(NUM,RC,KEY)                                           ADA05810
```

```
C     CALL MEXIT(NUM,RC)                                          ADA05820
      IF (KEY .EQ. 'PF10') GOTO 25                                ADA05830
      IF (KEY .EQ. 'PF12') STOP                                   ADA05840
   25 DO 23 I = 1,9                                               ADA05850
         CALL MSCR2D(NUM,RC,FLDN(I),CC,7,RLEN)                    ADA05860
         CALL MCVTCF(PU(I),RC,CC,7,2)                             ADA05870
   23 CONTINUE                                                    ADA05880
      RETURN                                                      ADA05890
      END                                                         ADA05900
                                                                  ADA05910
                                                                  ADA05920
                                                                  ADA05930
                                                                  ADA05940
                                                                  ADA05950
                                                                  ADA05960
                                                                  ADA05970
                                                                  ADA05980
```





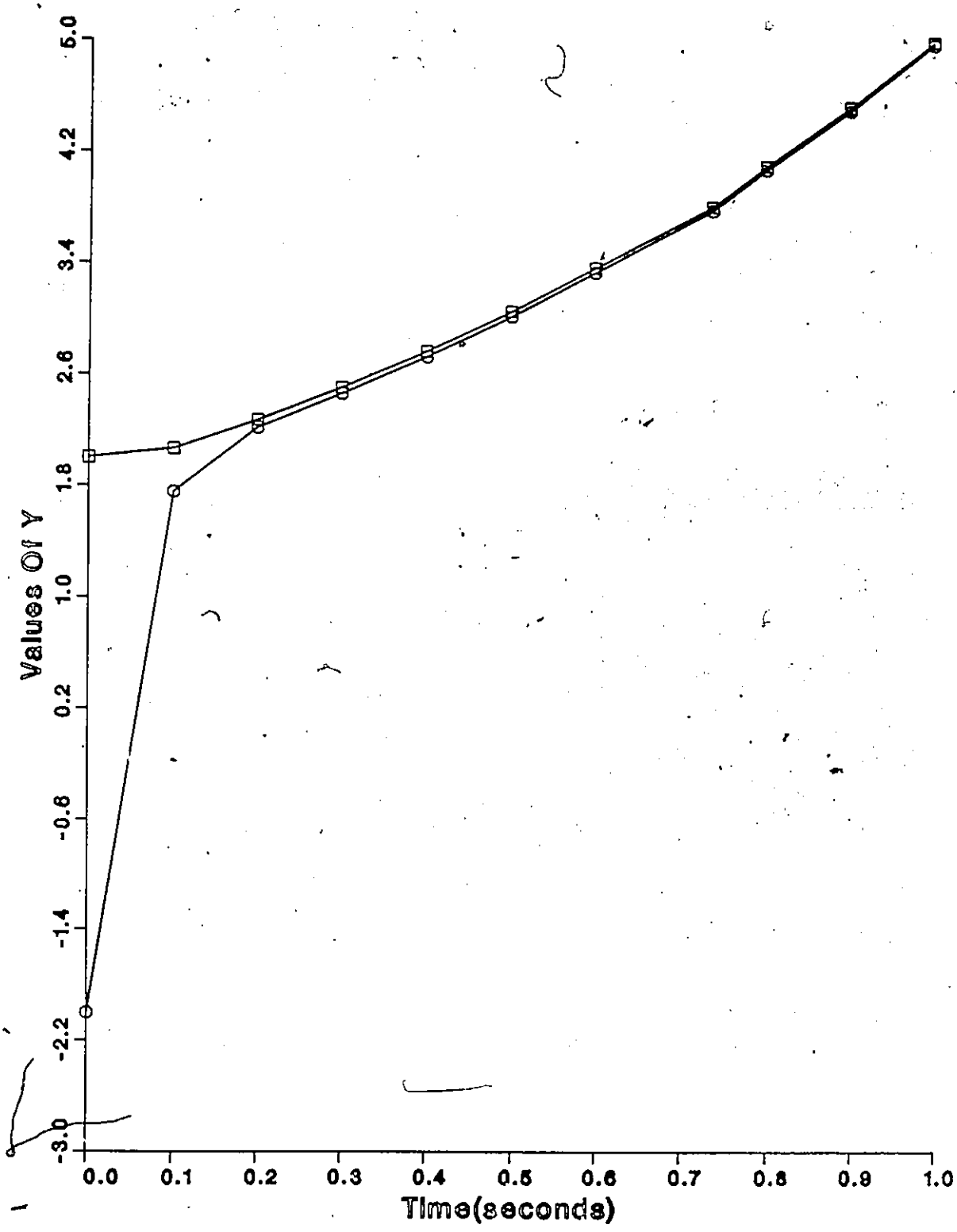
Menu For Rudder Control Model

Problem: Study Of The Rudder Control Model

Method: Adams Predictor Corrector Method

Time	VALUE OF Y1	VALUE OF Y2
0.00	2.000	-2.000
0.10	2.067	1.758
0.20	2.271	2.215
0.30	2.504	2.460
0.40	2.765	2.725
0.50	3.050	3.015
0.60	3.364	3.332
0.70	3.809	3.781
0.80	4.103	4.077
0.90	4.531	4.509
1.00	5.006	4.985

Study Of Rudder Control Model
Adams Predictor Corrector Method
Fourth-Order



Appendix. D THE BOUNCING BALL MODEL

D.1 Problem Definition

This model is used to demonstrate the use of the discontinuity processing algorithm in the processing of state events. The trajectory of the height of the ball with time is observed. The ball, being initially at rest is dropped from a height of 10m. The model consists of a simple set of differential equations. To prevent the hazardous detection of an erroneous cross-over which could occur in this model, the height of the ball is set to zero in the MODUPD subroutine once the ball hits the ground (i.e. the state event has been detected and located). If this is not supplied, erroneous results would occur. These erroneous results caused by the false detection of a crossover are as a result of the slope of the switching function changing but the sign of the function remaining the same. The algorithm may allow the ball to go underground for a short distance instead of bouncing. This is due to the inverse aitken interpolation terminating as soon as the tolerance is met. The algorithm integrates the system to a small distance underground and the model update at this point allows the ball to start its upward climb. However, in the next integration step, the algorithm detects a change in sign once more and hence an erroneous 'cross-over'. The 'new' point of discontinuity is located and the update involving

the change in sign of the velocity once again activated. The ball would thus start its downward descent and this process is continually repeated giving rise to an incorrect trajectory whereby the ball oscillates within a small range about the origin.

D.2 Gest Specification of The Bouncing Ball Model

```
Program StudyOfBouncingBall

ContinuousModel BouncingBall
  StaticStructure
    States  Y1,Y2          (* Y1 vertical velocity              *)
    Outputs Y2             (* Y2 height                         *)
    Constants G,K
      G = 9.807            (* G gravity constant                *)
      K = 0.8              (* K bouncing constant               *)
    EndConstants
  EndStaticStructure
  DynamicStructure
    Derivatives
      Y1' = -G
      Y2' = Y1
    EndDerivatives
    Updates
         When Y2 Crosses 0.0 ( Tolerance 0.01 ),
           Y1 = -K*Y1 ,
           Y2  = 0
    EndUpdates
  EndDynamicStructure
EndModel BouncingBall

Experiment 1
    TimeUnitIs Second
    InitialStates
      Y1 = 0
      Y2 = 10.0
    EndInitialStates
    IntegrateBy DiscontinuityProcessingAlgorithm
    Communicate at every 0.1 Second
    Display OutputModule 1 on Screen and on LaserQMS
    SimulateUntil Time = 10.0 Seconds
EndExperiment 1

Run 1
  ToObserve BoucingBall
    InExperiment 1
EndRun 1

OutputModule 1
  Heading ForFirstPage 1 Line
    StudyOfBouncingBall
  PlotAndList Y2 Versus Time
EndOutputModule 1

EndProgram StudyOfboucingBall
```

```
C  D.3     EQUIVALENT FORTRAN PROGRAM WITH RESULTS                    BAL00030
C          FOR BOUNCING BALL MODEL                                    BAL00040
                                                                      BAL00050
                                                                      BAL00060
                                                                      BAL00070
C                          Variable List                              BAL00080
C                                                                     BAL00090
C Input  : N              --Dimension of the Vector of State Variables BAL00100
C                                                                     BAL00110
C          NDF            --Dimension of the Vector of Discontinuity BAL00120
C                           Functions                                 BAL00130
C                                                                     BAL00140
C          Y(1)-Y(N)      --Initial Values of State Variables        BAL00150
C                                                                     BAL00160
C          G              --Constant used in differential equation for V BAL00170
C                                                                     BAL00180
C          K              --Constant used in Model Update            BAL00190
C                                                                     BAL00200
C          T              --Computational Time                       BAL00210
C                                                                     BAL00220
C          TINIT          --Initial Time Of Simulation               BAL00230
C                                                                     BAL00240
C          TFIN           --End Time of Simulation                   BAL00250
C                                                                     BAL00260
C          TCOM           --Communication Time                       BAL00270
C                                                                     BAL00280
C          COMINT         --Communication Interval                   BAL00290
C                                                                     BAL00300
C          TE             --Vector Of Times At Which Time            BAL00310
C                           Events Occur                              BAL00320
C                                                                     BAL00330
C          THOR           --Horizon Time                             BAL00340
C                                                                     BAL00350
C          H              --Integration Step Size                    BAL00360
C                                                                     BAL00370
C          HINIT          --Starting Integration Step Size           BAL00380
C                                                                     BAL00390
C          HMAX           --Maximum step size integration is         BAL00400
C                           allowed to proceed at                     BAL00410
C                                                                     BAL00420
C          HBOUND         --Upper Bound On Initial Step Size         BAL00430
C                                                                     BAL00440
C          HMIN           --Minimum step size integration is         BAL00450
C                           allowed to proceed at                     BAL00460
C                                                                     BAL00470
C          TOL            --Tolerance or Accuracy specified by user  BAL00480
C                                                                     BAL00490
C          NORD           --Order of the Integration Method          BAL00500
C                                                                     BAL00510
COUTPUT  :T                --Vector of times of Output Values         BAL00520
C                                                                     BAL00530
C         YVAL(N)          --Vector of Output at Output Points        BAL00540
C                                                                     BAL00550
C         YVALT(N)         --Vector of Output at Output Points        BAL00560
C                                                                     BAL00570
CPROGRAM VARIABLES :                                                  BAL00580
C         Y(N)          --State Variables Vector                      BAL00590
C                                                                     BAL00600
C         DY(N)         --Derivative Vector of State Variables        BAL00610
```

```
C                                                                       BAL00620
C       ER(N)          --Vector of Local Error Estimates                BAL00630
C                                                                       BAL00640
C       ERMAX          --Variable containing Maximum Error              BAL00650
C                                                                       BAL00660
C       HPOS           --Intermediate variable used in determining HINIT BAL00670
C                                                                       BAL00680
C       NPL            --Counter used for Graphics Routine              BAL00690
C                                                                       BAL00700
C       YVAL,TIME      --Single-Precision copies of output Vectors      BAL00710
C                        needed to run Disspla Graphics Package         BAL00720
C                                                                       BAL00730
C       YLAST(N)       --Last Values of State Variables                 BAL00740
C                                                                       BAL00750
C       DF(DDF)        --Present Values of Discontinuity Functions      BAL00760
C                                                                       BAL00770
C       PDF(DDF)       --Previous Values of Discontinuity Functions     BAL00780
C                                                                       BAL00790
C       F(10,5)        --Working Arrays used in Aitken Interpolation    BAL00800
C                        Table                                          BAL00810
C                                                                       BAL00820
C       TS(10,10)      --Working Arrays used in Aitken Interpolation    BAL00830
C                        Table                                          BAL00840
C                                                                       BAL00850
C       NUM            --Identifies the Discontinuity Function which    BAL00860
C                        has changed its sign in the preceding Step     BAL00870
C                                                                       BAL00880
C       NOT            --A Flag used to indicate whether the value of the BAL00890
C                        Discontinuity Function is within the Tolerance BAL00900
C                        Limit                                          BAL00910
C                                                                       BAL00920
C       NUMTE          --Number of Time Events                          BAL00930
C                                                                       BAL00940
C       ITER           --Iteration Number                               BAL00950
C                                                                       BAL00960
C       NPL            --Counter for plot of Output                     BAL00970
C                                                                       BAL00980
C       NTIMEV         --Flag Showing Time Event Taking Place           BAL00990
C                                                                       BAL01000
C       NFIRST         --Flag Showing Initial Step to be Processed      BAL01010
C                                                                       BAL01020
C       NTEALL         --Flag Showing Discon. are all Time Events       BAL01030
C                                                                       BAL01040
C       NTIMUP         --Flag Showing Time Update took place            BAL01050
C                                                                       BAL01060
C       G              --Constant Used In Differential Equation For V   BAL01070
C                                                                       BAL01080
C       C              --Parameter Used In Model Update                 BAL01090
C                                                                       BAL01100
C                                                                       BAL01110
C                              Variable Declaration                     BAL01120
C                                                                       BAL01130
      IMPLICIT REAL*8 (A-H,O-Z)                                         BAL01140
      DIMENSION Y(5)                                                    BAL01150
      REAL*4 TIME1(500),YVAL2(500),YVAL3(500)                           BAL01160
      COMMON/DIFF/YVAL(500),TIME(500),YVAL1(500),NPL                    BAL01170
      COMMON/FLAG/NUMTE,NTIMEV,NFIRST,NTEALL                            BAL01180
      COMMON/CON/G                                                      BAL01190
      COMMON/UPD/C                                                      BAL01200
```

```
      COMMON/WHY/COMINT,NOUT,NTIMUP                                     BAL01210
      COMMON/INTEG/ORD                                                  BAL01220
      EXTERNAL DERIV                                                    BAL01230
C                                  Initialization                       BAL01240
C                                       Initialize Constants            BAL01250
      G      = 9.807D0                                                  BAL01260
      C      = 0.8                                                      BAL01270
C                                                                       BAL01280
C                                       Initialize State Variables      BAL01290
      Y(1) = 0.0                                                        BAL01300
      Y(2) = 10.0                                                       BAL01310
C                                       Initialize Run Control Variables BAL01320
      T    = 0.0                                                        BAL01330
      TFIN   = 10.0                                                     BAL01340
      HMIN   = 0.000001                                                 BAL01350
      TOL    = 0.00001                                                  BAL01360
      ORD    = 4                                                        BAL01370
      COMINT= 0.10                                                      BAL01380
      NDF    = 1                                                        BAL01390
      N      = 2                                                        BAL01400
      NPL    = 1.                                                       BAL01410
      NUMTE = 0                                                         BAL01420
      NTIMEV = 0                                                        BAL01430
C                              Flag Showing Initial Step to be Processed BAL01440
      NFIRST = 0                                                        BAL01450
C                              Flag Showing Discon. are all Time Events BAL01460
      NTEALL = 0                                                        BAL01470
C                              Flag Showing Time Update Took Place      BAL01480
      NTIMUP = 1                                                        BAL01490
C                                  Pre-Run Output :                     BAL01500
C                                       Set up Output Format            BAL01510
                                                                        BAL01520
      WRITE(8,17)                                                       BAL01530
   17 FORMAT(1X,'Problem:',19X,'Study Of Bouncing Ball Model')          BAL01540
      WRITE(8,79)                                                       BAL01550
   79 FORMAT(1X,'Method:',19X,'Discontinuity Processing Algorithm '     BAL01560
     + /30X,'Based On Runge-Kutta Merson'                               BAL01570
     +       /10X/1X,'Time',6X,'Height ',6X,'Velocity')                 BAL01580
                                                                        BAL01590
       PRINT *,'       '                                                BAL01600
       TIME(1) = 0.0                                                    BAL01610
       YVAL1(1) = 10.0                                                  BAL01620
                                                                        BAL01630
      CALL DISCON(T,Y,TE,N,TFIN,HMIN,DERIV,TOL,NDF)                     BAL01640
                                                                        BAL01650
      DO 33 I = 1,NPL                                                   BAL01660
         TIME1(I) = TIME(I)                                             BAL01670
         YVAL2(I) = YVAL(I)                                             BAL01680
         YVAL3(I) = YVAL1(I)                                            BAL01690
   33 CONTINUE                                                          BAL01700
C                                  Post Run Output                      BAL01710
                                                                        BAL01720
      DO 59 I = 1,NPL                                                   BAL01730
         WRITE(8,23)TIME(I),YVAL1(I),YVAL(I)                            BAL01740
   23    FORMAT(1X,F5.3,5X,F8.3,5X,F8.3)                                BAL01750
   59 CONTINUE                                                          BAL01760
C     PRINT*, 'VALUE OF D IS',D                                         BAL01770
      CALL ANYDEV                                                       BAL01780
      CALL GRAPH(TIME1,YVAL3,NPL)                                       BAL01790
```

```
      STOP                                                              BAL01800
      END                                                               BAL01810
                                                                        BAL01820
                                                                        BAL01830
      SUBROUTINE DISFUN(Y,T,M,L,DF)                                     BAL05330
      IMPLICIT REAL*8 (A-H,O-Z)                                         BAL05340
      DIMENSION DF(L),Y(M)                                              BAL05350
C        This Subprogram is Responsible For Evaluating The Value        BAL05360
C        of The Discontinuity or Switching Function.The User Creates    BAL05370
C        This Module According to His Model.                            BAL05380
      DF(1)=Y(2)                                                        BAL05390
      RETURN                                                            BAL05400
      END                                                               BAL05410
                                                                        BAL05420
                                                                        BAL05430
      SUBROUTINE DERIV(T,Y,DY,N)                                        BAL05440
      IMPLICIT REAL*8 (A-H,O-Z)                                         BAL05450
      DIMENSION  Y(N),DY(N)                                             BAL05460
      COMMON/CON/G                                                      BAL05470
C        This is the derivative evaluation Subroutine where the         BAL05480
C        user  should write the model's system derivatives.             BAL05490
                                                                        BAL05500
      DY(1) = -G                                                        BAL05510
      DY(2)= Y(1)                                                       BAL05520
      RETURN                                                            BAL05530
      END                                                               BAL05540
                                                                        BAL05550
                                                                        BAL07960
      SUBROUTINE GRAPH(TIME1,YVAL3,NPL)                                 BAL07970
                                                                        BAL07980
C        This Subroutine Provides A Graphical Display of The            BAL07990
C        BouncingBall Model.Height is plotted against Time.             BAL08000
      DIMENSION TIME1(NPL),YVAL3(NPL)                                   BAL08010
      VALMAX = YVAL3(NPL)                                               BAL08020
      VALMIN = YVAL3(1)                                                 BAL08030
      DO 13 I = 1,NPL                                                   BAL08040
         IF (YVAL3(I).GT.VALMAX) THEN                                   BAL08050
            VALMAX = YVAL3(I)                                           BAL08060
         ELSE IF(YVAL3(I).LT.VALMIN) THEN                               BAL08070
            VALMIN = YVAL3(I)                                           BAL08080
         ENDIF                                                          BAL08090
   13 CONTINUE                                                          BAL08100
      MAXVAL = VALMAX + 1                                               BAL08110
      MINVAL = VALMIN - 1                                               BAL08120
      SPAC1 = (TIME1(NPL) - TIME1(1))/10.0                              BAL08130
      SPAC2 = (MAXVAL - MINVAL)/10.0                                    BAL08140
      CALL HWROT('MOVIE')                                               BAL08150
      CALL SWISSM                                                       BAL08160
      CALL PAGE(8.5,11.0)                                               BAL08170
      CALL AREA2D(6.0,8.0)                                              BAL08180
      CALL HEADIN('Study Of Bouncing Ball Model$',100,1.5,3)            BAL08190
      CALL HEADIN('Discontinuity Processing Algorithm$',100,1.25,3)     BAL08200
      CALL HEADIN('Based On RK-Merson$',100,-1.15,3)                    BAL08210
      CALL XNAME('TIME(sec[redacted])$',100)                            BAL08220
      CALL YNAME('Height(m[redacted],100)                               BAL08230
      CALL GRAF(TIME1(1),SPAC1,TIME1(NPL),0.0,SPAC2,MAXVAL)             BAL08240
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                       BAL08250
C  RATIONAL SPLINE INTERPOLATION                                        BAL08260
      CALL RASPLN (1.5)                                                 BAL08270
```

```
      CALL THKCRV (.015)                                                BAL08280
      CALL THKRND (0.)                                                  BAL08290
      CALL CURVE(TIME1,YVAL3,NPL,0)                                     BAL08300
      CALL ENDPL(0)                                                     BAL08310
      RETURN                                                            BAL08320
      END                                                               BAL08330
                                                                        BAL08340
                                                                        BAL08350
                                                                        BAL08360
      SUBROUTINE OUTPUT(Y,N,T)                                          BAL08370
      IMPLICIT REAL*8 (A-H,O-Z)                                         BAL08380
      DIMENSION Y(N)                                                    BAL08390
      COMMON/DIFF/YVAL(500),TIME(500),YVAL1(500),NPL                    BAL08400
                                                                        BAL08410
      WRITE(6,23) T,Y(2),Y(1)                                           BAL08420
   23 FORMAT(1X,F5.2,3X,F9.3,3X,F9.3)                                   BAL08430
      NPL = NPL + 1                                                     BAL08440
      TIME(NPL) = T                                                     BAL08450
      YVAL(NPL) = Y(1)                                                  BAL08460
      YVAL1(NPL) = Y(2)                                                 BAL08470
      RETURN                                                            BAL08480
      END                                                               BAL08490
                                                                        BAL08500
      SUBROUTINE MODUPD(DF,NDF,T,Y,N,TOL)                               BAL08510
C        This subroutine is model dependent and is supplied by          BAL08520
C        the user. It contains all time and state events for            BAL08530
C        the model involved.                                            BAL08540
      DIMENSION Y(N),DF(N)                                              BAL08550
      COMMON/UPD/C                                                      BAL08560
      IF ((DF(1).GE.-TOL) .AND. (DF(1).LE.0)) THEN                      BAL08570
         Y(1) = -(C * Y(1))                                             BAL08580
         Y(2) = 0.0                                                     BAL08590
      ENDIF                                                             BAL08600
      RETURN                                                            BAL08610
      END                                                               BAL08620
                                                                        BAL08630
                                                                        BAL08640
```

Method: Discontinuity Processing Algorithm
Based On Runge-Kutta Merson

Problem: Study Of Bouncing Ball Model

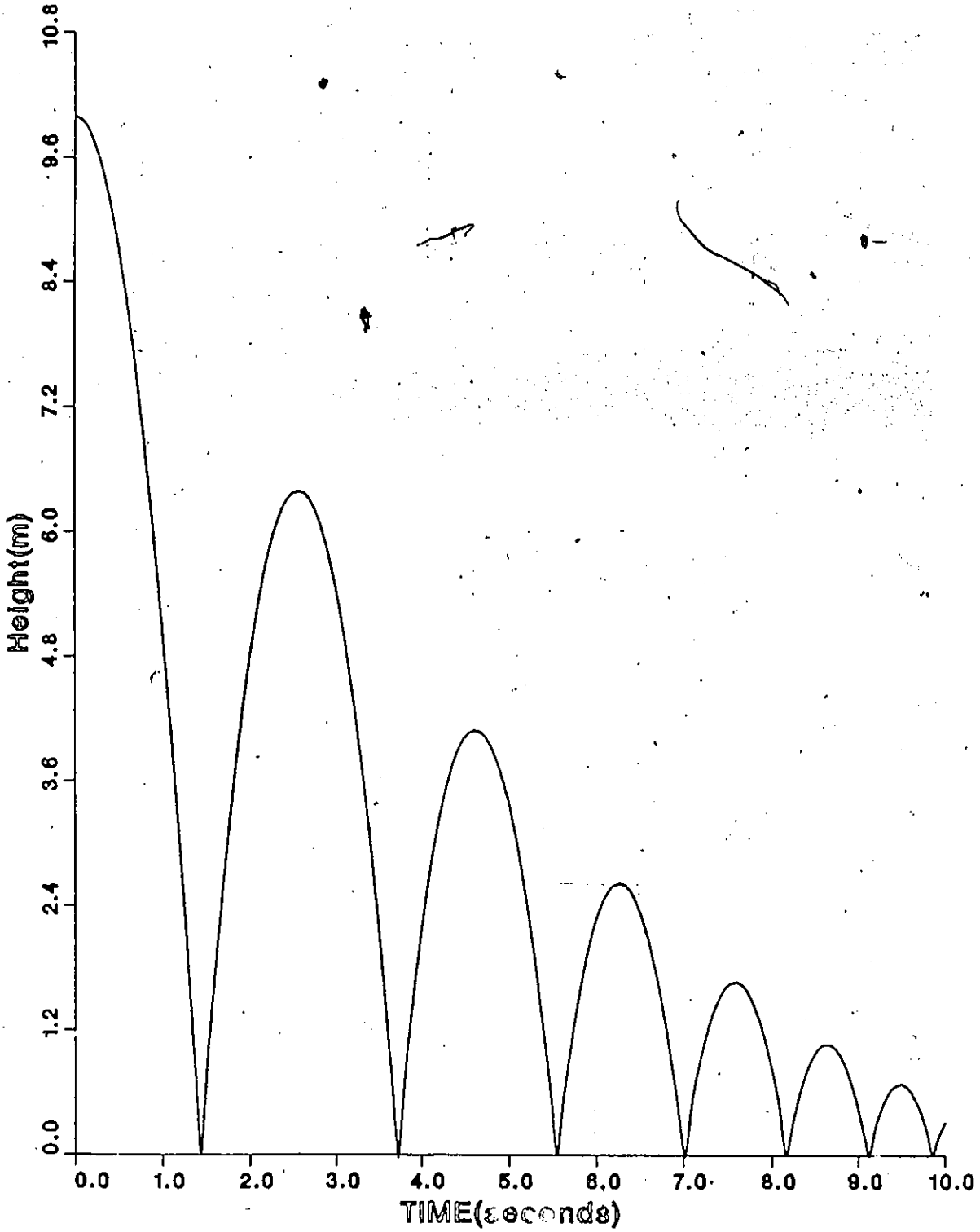
Time (T)	Height (Y2)
0.000	10.000
0.100	9.951
0.200	9.804
0.300	9.559
0.400	9.215
0.500	8.774
0.600	8.235
0.700	7.597
0.800	6.862
0.900	6.028
1.000	5.096
1.100	4.067
1.200	2.939
1.300	1.713
1.400	0.389
1.428	0.000
1.500	0.781
1.600	1.781
1.700	2.684
1.800	3.489
1.900	4.195
2.000	4.804
2.100	5.314
2.200	5.727
2.300	6.041
2.400	6.257
2.500	6.376
2.600	6.396
2.700	6.318
2.800	6.142
2.900	5.868
3.000	5.495
3.100	5.025
3.200	4.457
3.300	3.791
3.400	3.026
3.500	2.164
3.600	1.203
3.700	0.144
3.713	0.000
3.800	0.743
3.900	1.505
4.000	2.169
4.100	2.735
4.200	3.202
4.300	3.572
4.400	3.844
4.500	4.017
4.600	4.092
4.700	4.070
4.800	3.949
4.900	3.730
5.000	3.413
5.100	2.999
5.200	2.486
5.300	1.875
5.400	1.165
5.500	0.358

Method: Discontinuity Processing Algorithm
Based On Runge-Kutta Merson

Problem: Study Of Bouncing Ball Model

Time	Height
5.541	0.000
5.600	0.407
5.700	1.017
5.800	1.529
5.900	1.943
6.000	2.259
6.100	2.476
6.200	2.596
6.300	2.618
6.400	2.541
6.500	2.367
6.600	2.094
6.700	1.723
6.800	1.255
6.900	0.688
7.000	0.023
7.003	0.000
7.100	0.509
7.200	0.939
7.300	1.271
7.400	1.504
7.500	1.640
7.600	1.677
7.700	1.616
7.800	1.458
7.900	1.201
8.000	0.846
8.100	0.393
8.173	0.000
8.200	0.120
8.300	0.503
8.400	0.789
8.500	0.976
8.600	1.065
8.700	1.057
8.800	0.950
8.900	0.745
9.000	0.442
9.100	0.041
9.109	0.000
9.200	0.294
9.300	0.522
9.400	0.653
9.500	0.686

Study Of Bouncing Ball Model
Discontinuity Processing Algorithm
Based On RK-Merson



Appendix. E THE SAFETY PARACHUTE MODEL

E.1 Problem Definition

This model illustrates the use of the discontinuity processing algorithm in the processing of time and state events. The simulation simulates the trajectory of a parachutist starting at a height of 1000m at the point of leaving the aeroplane, to the specific time of popping the parachute at T = 5 seconds, to the point when the emergency parachute pops at a height of 900m, and the subsequent descent. The aeroplane is assumed to be flying at a fairly level trajectory before allowing the parachutist to jump. Thus the initial horizontal component of velocity is given to be large as opposed to the upward vertical component of velocity which is very small. The world view taken is that the upward vertical velocity and the easterly horizontal component of velocity, were both positive. The reverse directions were negative. The results displayed by the simulation are as expected. They indicate a drastic reduction in the horizontal component of the velocity and a quick change of an upward to a downward component of velocity in the first second. The horizontal component of velocity continues to fall at a reduced rate while the downward vertical component of velocity increases rapidly and gradually levels off. At the time of the first discontinuity, it is assumed that the parachute fails

opening only partially and hence increasing the drag coefficient slightly. This point occurs at T = 5 where the vertical component of velocity decreases only slightly. When the safety parachute is pulled, the drag coefficient increases and the vertical component of velocity reduces sharply. The descent hereafter sees the horizontal component diminish to zero over time with the vertical component stabilizing at a constant velocity until the parachutist finally lands.

E.2 Gest Specification of The Safety Parachute Model

```
Program StudyOfSafetyParachute

ContinuousModel SafetyParachute
  StaticStructure
    States Y1,Y2,Y3              (* Y2 Vertical Component of Velocity   *)
    Outputs Y1,Y2                (* Y1 Horizontal Component of Velocity *)
    AuxiliaryVariables V         (* V Velocity  , Y3 Height             *)
    Constants G
      G = 9.807
    EndConstants
    Parameters M,D,TE            (* TE Parachute Opening Time (s)
                                    D Drag Coef.  (kg/m)
                                    M Mass (kg)                         *)
  EndStaticStructure

  DynamicStructure
    Derivatives
      Y1' = -(D*V*Y1)/M
      Y2' = -G-(D*V*Y2)/M
      V   = Sqrt(Y1**2+Y2**2)
      Y3' = Y2
    EndDerivatives
    Updates

        When Time = TE
          D = 1.5                (* Update of parameter                 *)

        When Y3 = 900
          D = 7.0                (* Update of parameter                 *)

    EndUpdates
  EndDynamicStructure

EndModel SafetyParachute

ParameterSet 1
  Model Parachute
    Parameters
      TE = 5.0
      M = 70
      D = 0.7
    EndParameters
  EndModel Parachute
EndParameterSet1


Experiment 1
    TimeUnitIs Second
    InitialStates
      Y1 = 800.0
      Y2 = 10.0
      Y3 = 1000.0
    EndInitialStates
    IntegrateBy Discontinuity Algorithm
    Communicate at every 0.3 Second
    Display OutputModule 1 on Screen and on LaserQMS
    SimulateUntil Time = 20.0
EndExperiment 1
```

```
Run 1
  ToObserve Parachute
    WithParameterSet 1
    InExperiment 1
EndRun 1


OutputModule 1
  NewPage
  Heading ForFirstPage 1 Line
    StudyofSafetyParachute
  PlotandList Y1,Y2,Y3 Versus Time
EndOutputModule 1


EndProgram StudyOfFaultyParachute
```

```
C  E.3    Equivalent Fortran Program With Results                    SAF00010
C         For Safety Parachute Model                                 SAF00020
C                                                                    SAF00030
C                                                                    SAF00040
C                                                                    SAF00050
C                                                                    SAF00060
C                          Variable List                             SAF00070
C                                                                    SAF00080
C Input  : N              --Dimension of the Vector of State Variables SAF00090
C                                                                    SAF00100
C          NDF            --Dimension of the Vector of Discontinuity SAF00110
C                           Functions                                SAF00120
C                                                                    SAF00130
C          Y(1)-Y(N)      --Initial Values of State Variables       SAF00140
C                                                                    SAF00150
C          G              --Constant used in differential equation for V SAF00160
C                                                                    SAF00170
C          K              --Constant used in Model Update           SAF00180
C                                                                    SAF00190
C          T              --Computational Time                      SAF00200
C                                                                    SAF00210
C          TINIT          --Initial Time Of Simulation              SAF00220
C                                                                    SAF00230
C          TFIN           --End Time of Simulation                  SAF00240
C                                                                    SAF00250
C          TCOM           --Communication Time                      SAF00260
C                                                                    SAF00270
C          COMINT         --Communication Interval                  SAF00280
C                                                                    SAF00290
C          TE             --Vector Of Times At Which Time           SAF00300
C                           Events Occur                             SAF00310
C                                                                    SAF00320
C          THOR           --Horizon Time                            SAF00330
C                                                                    SAF00340
C          H              --Integration Step Size                   SAF00350
C                                                                    SAF00360
C          HINIT          --Starting Integration Step Size          SAF00370
C                                                                    SAF00380
C          HMAX           --Maximum step size integration is        SAF00390
C                           allowed to proceed at                    SAF00400
C          HBOUND         --Upper Bound On Initial Step Size        SAF00410
C                                                                    SAF00420
C          HMIN           --Minimum step size integration is        SAF00430
C                           allowed to proceed at                    SAF00440
C                                                                    SAF00450
C          TOL            --Tolerance or Accuracy specified by user SAF00460
C                                                                    SAF00470
C          NORD           --Order of the Integration Method         SAF00480
C                                                                    SAF00490
COUTPUT :T                --Vector of times of Output Values        SAF00500
C                                                                    SAF00510
C         YVAL(N)         --Vector of Output at Output Points       SAF00520
C                                                                    SAF00530
C         YVAL1(N)        --Vector of Output at Output Points       SAF00540
C                                                                    SAF00550
CPROGRAM VARIABLES :                                                 SAF00560
C         Y(N)          --State Variables Vector                   SAF00570
C                                                                    SAF00580
C         DY(N)         --Derivative Vector of State Variables     SAF00590
C                                                                    SAF00600
C         ER(N)         --Vector of Local Error Estimates          SAF00610
C                                                                    SAF00620
C         ERMAX         --Variable containing Maximum Error        SAF00630
                                                                     SAF00640
```

```
C                                                                       SAF00650
C       HPOS          --Intermediate variable used in determining HINIT SAF00660
C                                                                       SAF00670
C       NPL           --Counter used for Graphics Routine               SAF00680
C                                                                       SAF00690
C       YVAL,TIME     --Single Precision copies of output Vectors       SAF00700
C                       needed to run Disspla Graphics Package          SAF00710
C                                                                       SAF00720
C       YLAST(N)      --Last Values of State Variables                  SAF00730
C                                                                       SAF00740
C       DF(DDF)       --Present Values of Discontinuity Functions       SAF00750
C                                                                       SAF00760
C       PDF(DDF)      --Previous Values of Discontinuity Functions      SAF00770
C                                                                       SAF00780
C       F(10,5)       --Working Arrays used in Aitken Interpolation     SAF00790
C                       Table                                           SAF00800
C                                                                       SAF00810
C       TS(10,10)     --Working Arrays used in Aitken Interpolation     SAF00820
C                       Table                                           SAF00830
C                                                                       SAF00840
C       NUM           --Identifies the Discontinuity Function which     SAF00850
C                       has changed its sign in the preceding Step      SAF00860
C                                                                       SAF00870
C       NOT           --A Flag used to indicate whether the value of the SAF00880
C                       Discontinuity Function is within the Tolerance  SAF00890
C                       Limit                                           SAF00900
C                                                                       SAF00910
C       NUMTE         --Number of Time Events                           SAF00920
C                                                                       SAF00930
C       ITER          --Iteration Number                                SAF00940
C                                                                       SAF00950
C       NPL           --Counter for plot of Output                      SAF00960
C                                                                       SAF00970
C       NTIMEV        --Flag Showing Time Event Taking Place            SAF00980
C                                                                       SAF00990
C       NFIRST        --Flag Showing Initial Step to be Processed       SAF01000
C                                                                       SAF01010
C       NTEALL        --Flag Showing Discon. are all Time Events        SAF01020
C                                                                       SAF01030
C       NTIMUP        --Flag Showing Time Update took place             SAF01040
C                                                                       SAF01050
                                                                        SAF01060
C                              Variable Declaration.                    SAF01070
                                                                        SAF01080
      IMPLICIT REAL*8 (A-H,O-Z)                                         SAF01090
      DIMENSION Y(5),TE(5)                                              SAF01100
      REAL*4 TIME1(500),YVAL2(500),YVAL3(500),YVAL4(500)                SAF01110
      COMMON/DIFF/YVAL(500),TIME(500),YVAL1(500),Y4(500),NPL            SAF01120
      COMMON/FLAG/NUMTE,NTIMEV,NFIRST,NTEALL                            SAF01130
      COMMON/DER/D,G,M                                                  SAF01140
      COMMON/WHY/COMINT,NOUT,NTIMUP                                     SAF01150
      COMMON/INTEG/NORD                                                 SAF01160
      EXTERNAL DERIV                                                    SAF01170
C                               Initialization                          SAF01180
C                                    Initialize Constants               SAF01190
      M     = 70.                                                       SAF01200
      G     = 9.807D0                                                   SAF01210
      D     = 0.7                                                       SAF01220
C                                    Initialize State Variables         SAF01230
      Y(1) = 800.0                                                      SAF01240
      Y(2) = 10.0                                                       SAF01250
      Y(3) = 1000.0                                                     SAF01260
C                                    Initialize Run Control Variables   SAF01270
      T     = 0.0                                                       SAF01280
      TFIN  = 20.0                                                      SAF01290
      TE(1) = 5.0                                                       SAF01300
```

```
C     TE(2) = 7.02                                                      SAF01310
      COMINT = 0.3                                                      SAF01320
      HMIN  = 0.0000001                                                 SAF01330
      TOL   = 0.00001                                                   SAF01340
      NORD  = 4                                                         SAF01350
      NDF   = 1                                                         SAF01360
      N     = 3                                                         SAF01370
      NPL   = 1                                                         SAF01380
      NUMTE = 1                                                         SAF01390
C                                  FLAG SHOWING TIME EVENT TAKING PLACE SAF01400
      NTIMEV = 0                                                        SAF01410
C                               FLAG SHOWING INITIAL STEP TO BE PROCESSED SAF01420
      NFIRST = 0                                                        SAF01430
C                               FLAG SHOWING DISCON. ARE ALL TIME EVENTS SAF01440
      NTEALL = 0                                                        SAF01450
C                               FLAG SHOWING TIME UPDATE TOOK PLACE     SAF01460
      NTIMUP = 1                                                        SAF01470
C                                    Pre-Run Output :                   SAF01480
C                                         Set up Output Format          SAF01490
                                                                        SAF01500
      WRITE(8,17)                                                       SAF01510
   17 FORMAT(1X,'Problem:',1X ,'Study Of Safety Parachute Model'        SAF01520
     + /13X,'Time And State Events')                                    SAF01530
      WRITE(8,11)                                                       SAF01540
   11 FORMAT(1X,'Method:',1X ,'Discontinuity Processing Algorithm')     SAF01550
                                                                        SAF01560
       PRINT *,'       '                                                SAF01570
                                                                        SAF01580
      TIME(1) = 0.0                                                     SAF01590
      YVAL(1) = 800.0                                                   SAF01600
      YVAL1(1)= 10.0                                                    SAF01610
      Y4(1)   = 1000.0                                                  SAF01620
      CALL DISCON(T,Y,TE,N,TFIN,HMIN,DERIV,TOL,NDF)                     SAF01630
                                                                        SAF01640
      DO 33 I = 1,NPL                                                   SAF01650
         TIME1(I) = TIME(I)                                             SAF01660
         YVAL2(I) = YVAL(I)                                             SAF01670
         YVAL3(I) = YVAL1(I)                                            SAF01680
         YVAL4(I) =Y4(I)                                                SAF01690
   33 CONTINUE                                                          SAF01700
C                                    Post Run Output                    SAF01710
                                                                        SAF01720
         DO 59 I = 1,NPL                                                SAF01730
            WRITE(8,23)TIME(I),YVAL(I),YVAL1(I),Y4(I)                   SAF01740
   23       FORMAT(1X,F6.2,5X,F8.3,5X,F8.3,5X,F8.3)                     SAF01750
   59    CONTINUE                                                       SAF01760
C     PRINT*, 'VALUE OF D IS',D                                         SAF01770
      CALL ANYDEV                                                       SAF01780
      CALL GRAPH(TIME1,YVAL3,NPL)                                       SAF01790
      CALL GRAPH(TIME1,YVAL2,NPL)                                       SAF01800
      CALL GRAPH(TIME1,YVAL4,NPL)                                       SAF01810
      STOP                                                              SAF01820
      END                                                               SAF01830
                                                                        SAF01840
                                                                        SAF01850
                                                                        SAF01860
                                                                        SAF05340
      SUBROUTINE DISFUN(Y,T,M,L,DF)                                     SAF05350
      IMPLICIT REAL*8 (A-H,O-Z)                                         SAF05360
      DIMENSION DF(L),Y(M)                                              SAF05370
C        This subprogram is responsible for evaluating the value        SAF05380
C        of the discontinuity or switching function. The user creates   SAF05390
C        this module according to his model.                            SAF05400
                                                                        SAF05410
      DF(1) = Y(3) - 900                                                SAF05420
      RETURN                                                            SAF05430
```

```
      END                                                                SAF05440
                                                                         SAF05450
                                                                         SAF05460
                                                                         SAF05470
      SUBROUTINE DERIV(T,Y,DY,N)                                         SAF05480
      IMPLICIT REAL*8 (A-H,O-Z)                                          SAF05490
      DIMENSION  Y(N),DY(N)                                              SAF05500
      COMMON/DER/D,G,M                                                   SAF05510
C                              This is the derivative evaluation         SAF05520
C                              subroutine where the user should          SAF05530
C                              write the model's system derivatives.     SAF05540
      V = SQRT(Y(1)*Y(1)+Y(2)*Y(2))                                      SAF05550
      DY(1) = -(D*V*Y(1))/M                                              SAF05560
      DY(2)= -G -(D*V*Y(2))/M                                            SAF05570
      DY(3)= Y(2)                                                        SAF05580
      RETURN                                                             SAF05590
      END                                                                SAF05600
                                                                         SAF05610
                                                                         SAF05620
                                                                         SAF08020
      SUBROUTINE GRAPH(TIME1,YVAL3,NPL)                                  SAF08030
                                                                         SAF08040
C        This Subroutine Provides A Graphical Display of The             SAF08050
C        Bouncing Ball Model.Height is plotted against Time.             SAF08060
      DIMENSION TIME1(NPL),YVAL3(NPL)                                    SAF08070
      VALMAX = YVAL3(NPL)                                                SAF08080
      VALMIN = YVAL3(1)                                                  SAF08090
      DO 13 I = 1,NPL                                                    SAF08100
         IF (YVAL3(I).GT.VALMAX) THEN                                    SAF08110
            VALMAX = YVAL3(I)                                            SAF08120
         ELSE IF(YVAL3(I).LT.VALMIN) THEN                                SAF08130
            VALMIN = YVAL3(I)                                            SAF08140
         ENDIF                                                           SAF08150
   13 CONTINUE                                                           SAF08160
      MAXVAL = VALMAX + 1                                                SAF08170
      MINVAL = VALMIN - 1                                                SAF08180
      SPAC1 = (TIME1(NPL) - TIME1(1))/10.0                               SAF08190
      SPAC2 = (MAXVAL - MINVAL)/10.0                                     SAF08200
      CALL HWROT('MOVIE')                                                SAF08210
      CALL SWISSM                                                        SAF08220
      CALL PAGE(8.5,11.0)                                                SAF08230
      CALL AREA2D(6.0,8.0)                                               SAF08240
      CALL HEADIN('Study Of Safety Parachute Model$',100,1.5,3)          SAF08250
      CALL HEADIN('Discontinuity Processing Algorithm$',100,1.25,3)      SAF08260
      CALL HEADIN('Based On RK-Merson$',100,-1.15,3)                     SAF08270
      CALL XNAME('Time(seconds)$',100)                                   SAF08280
      CALL YNAME('Height(m)$',100)                                       SAF08290
      CALL GRAF(TIME1(1),SPAC1,TIME1(NPL),MINVAL,SPAC2,MAXVAL)           SAF08300
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                        SAF08310
C  RATIONAL SPLINE INTERPOLATION                                         SAF08320
      CALL RASPLN (1.5)                                                  SAF08330
      CALL THKCRV (.015)                                                 SAF08340
      CALL THKRND (0.)                                                   SAF08350
      CALL CURVE(TIME1,YVAL3,NPL,0)                                      SAF08360
      CALL ENDPL(0)                                                      SAF08370
      RETURN                                                             SAF08380
      END                                                                SAF08390
                                                                         SAF08400
                                                                         SAF08410
      SUBROUTINE OUTPUT(Y,N,T)                                           SAF08420
      IMPLICIT REAL*8 (A-H,O-Z)                                          SAF08430
      DIMENSION Y(N)                                                     SAF08440
      COMMON/DIFF/YVAL(500),TIME(500),YVAL1(500),Y4(500),NPL             SAF08450
                                                                         SAF08460
      WRITE(6,23) T,Y(2),Y(1)                                            SAF08470
   23 FORMAT(1X,F5.2,3X,F9.3,3X,F9.3)                                    SAF08480
```

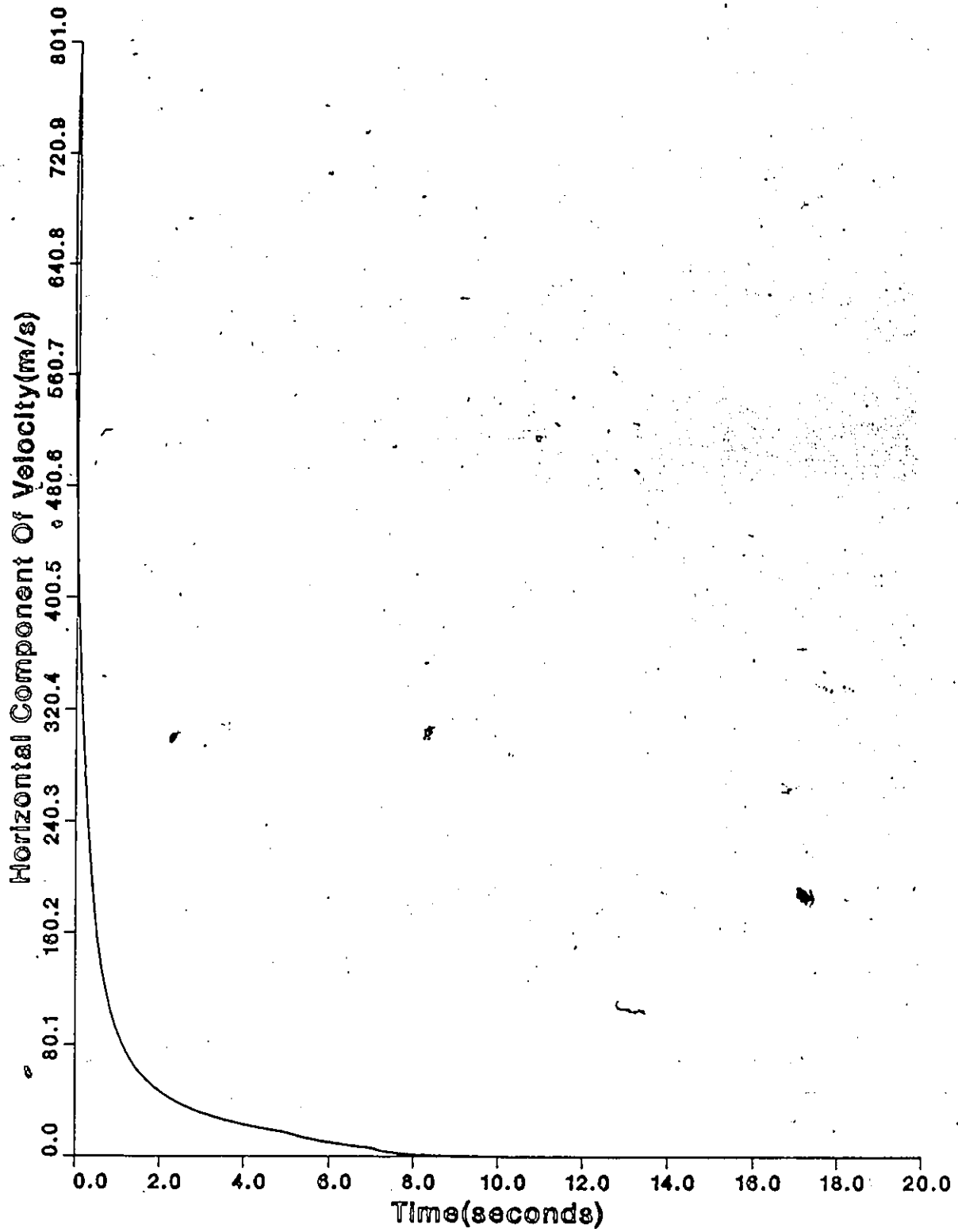
```
      NPL = NPL + 1                                                    SAF08490
      TIME(NPL) = T                                                    SAF08500
      YVAL(NPL) = Y(1)                                                 SAF08510
      YVAL1(NPL) = Y(2)                                                SAF08520
      Y4(NPL)    = Y(3)                                                SAF08530
      RETURN                                                           SAF08540
      END                                                              SAF08550
                                                                       SAF08560
      SUBROUTINE MODUPD(DF,NDF,T,Y,N,TOL)                              SAF08570
C        This subroutine is model dependent and is supplied by         SAF08580
C        the user. It contains all time and state events for           SAF08590
C        the model involved.                                           SAF08600
      DIMENSION Y(N),DF(N)                                             SAF08610
      COMMON/DER/D,G,M                                                 SAF08620
                                                                       SAF08630
      IF (T.EQ.5.0) THEN                                               SAF08640
         D = 1.5                                                       SAF08650
      ENDIF                                                            SAF08660
                                                                       SAF08670
      IF ((DF(1).GE.-TOL   ) .AND. (DF(1).LE.0   )) THEN               SAF08680
         D = 7.0                                                       SAF08690
      ENDIF                                                            SAF08700
      RETURN                                                           SAF08710
      END                                                              SAF08720
                                                                       SAF08730
                                                                       SAF08740
```

Problem: Study Of Safety Parachute Model
Time And State Events

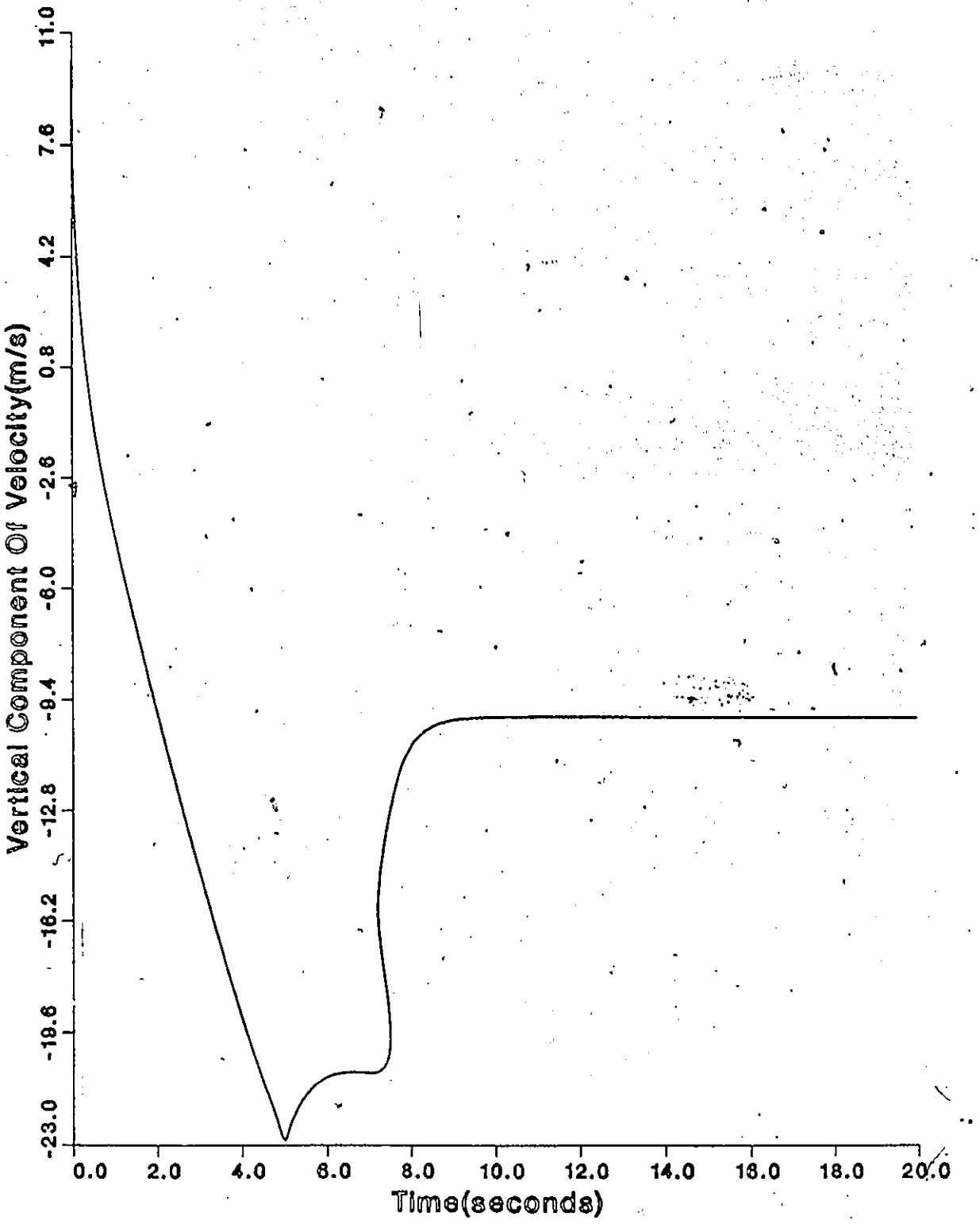
Method: Discontinuity Processing Method

Time (T)	Velocity (Y1)	Velocity (Y2)	height (Y3)
0.00	800.000	10.000	1000.000
0.30	235.299	1.038	1001.219
0.60	137.933	-1.725	1001.082
0.90	97.553	-3.732	1000.254
1.20	75.440	-5.494	998.866
1.50	61.467	-7.146	996.968
1.80	51.816	-8.735	994.585
2.10	44.726	-10.280	991.731
2.40	39.275	-11.789	988.420
2.70	34.928	-13.263	984.661
3.00	31.359	-14.699	980.466
3.30	28.355	-16.091	975.846
3.60	25.773	-17.433	970.816
3.90	23.516	-18.718	965.392
4.20	21.513	-19.940	959.592
4.50	19.716	-21.091	953.435
4.80	18.088	-22.169	946.944
5.00	17.083	-22.844	942.442
5.10	16.086	-22.463	940.178
5.40	13.565	-21.650	933.571
5.70	11.566	-21.181	927.154
6.00	9.937	-20.928	920.841
6.30	8.581	-20.811	914.583
6.60	7.436	-20.776	908.347
6.90	6.459	-20.791	902.112
7.00	6.161	-20.804	900.000
7.20	4.244	-15.968	896.412
7.50	2.745	-12.734	892.178
7.80	1.906	-11.318	888.597
8.10	1.368	-10.632	885.316
8.40	0.998	-10.284	882.185
8.70	0.735	-10.103	879.130
9.00	0.543	-10.009	876.114
9.30	0.402	-9.959	873.120
9.60	0.299	-9.932	870.137
9.90	0.222	-9.919	867.159
10.20	0.165	-9.911	864.185
10.50	0.122	-9.907	861.212
10.80	0.091	-9.905	858.240
11.10	0.067	-9.904	855.269
11.40	0.050	-9.904	852.298
11.70	0.037	-9.903	849.327
12.00	0.028	-9.903	846.356
12.30	0.021	-9.903	843.385
12.60	0.015	-9.903	840.414
12.90	0.011	-9.903	837.443
13.20	0.008	-9.903	834.472
13.50	0.006	-9.903	831.501
13.80	0.005	-9.903	828.530
14.10	0.003	-9.903	825.559
14.40	0.003	-9.903	822.588
14.70	0.002	-9.903	819.618
15.00	0.001	-9.903	816.647
15.30	0.001	-9.903	813.676
15.60	0.001	-9.903	810.705
15.90	0.001	-9.903	807.734
16.20	0.000	-9.903	804.763
16.50	0.000	-9.903	801.792
16.80	0.000	-9.903	798.821
17.10	0.000	-9.903	795.850
17.40	0.000	-9.903	792.879
17.70	0.000	-9.903	789.908
18.00	0.000	-9.903	786.938
18.30	0.000	-9.903	783.967
18.60	0.000	-9.903	780.996
18.90	0.000	-9.903	778.025
19.20	0.000	-9.903	775.054
19.50	0.000	-9.903	772.083
19.80	0.000	-9.903	769.112
20.00	0.000	-9.903	767.132

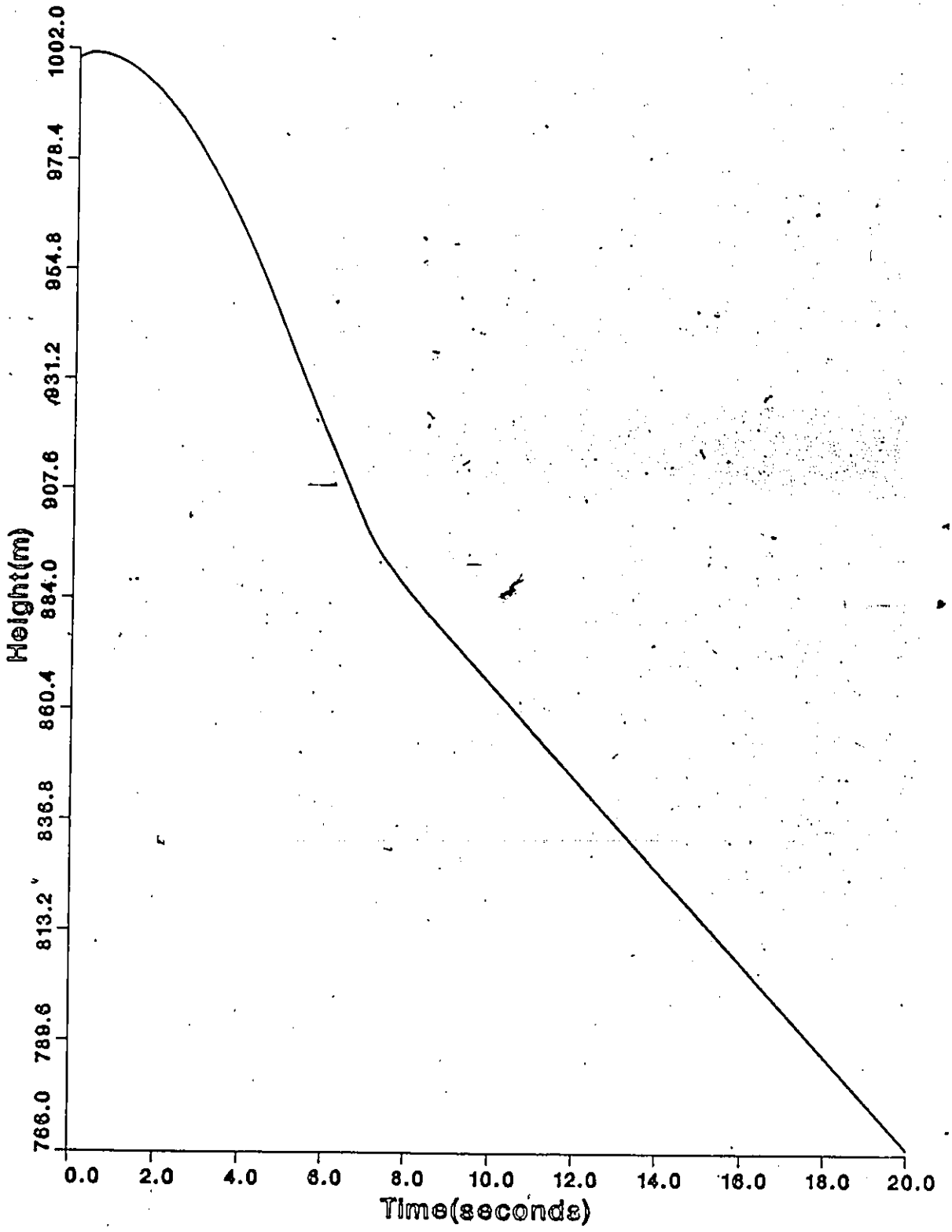
Study Of Safety Parachute Model
Discontinuity Processing Algorithm
Based On RK-Merson



Study Of Safety Parachute Model
Discontinuity Processing Algorithm
Based On RK-Merson



Study Of Safety Parachute Model
Discontinuity Processing Algorithm
Based On RK-Merson



Appendix. F THE TREE BIOMASS MODEL

F.1 Problem Description

This simulation is fully described by Goldstein and Harris (1973). It is a deterministic model to simulate tree biomass dynamics of a closed canopy, forested watershed on a year to year basis. The model is conceived on the basis of measurable forest processes influencing change in biomass. Trees are divided into three size classes: 1.3-8.8, 8.9-23.8 and >23.8 dbh (diameter at breast height) and the resulting coupling between the different compartments is as shown in Figure F.1. The live above ground biomasses are designated Y1, Y2 and Y3 respectively. There are three variables Y4, Y5 and Y6 for the three size classes of standing dead material and Y7 is the total live root biomass in all the classes. The equations defining the rate of change of these variables and all other model information is shown in the Gest specification in section F.2. The simulation is carried out over a 300 year period with the following initial conditions for each sector Yi: 10000, 30000, 60000, 2500, 2000 and 32000 (measured in kilograms per hectare). The results are similar to those shown in the article by Goldstein and Harris (1973).

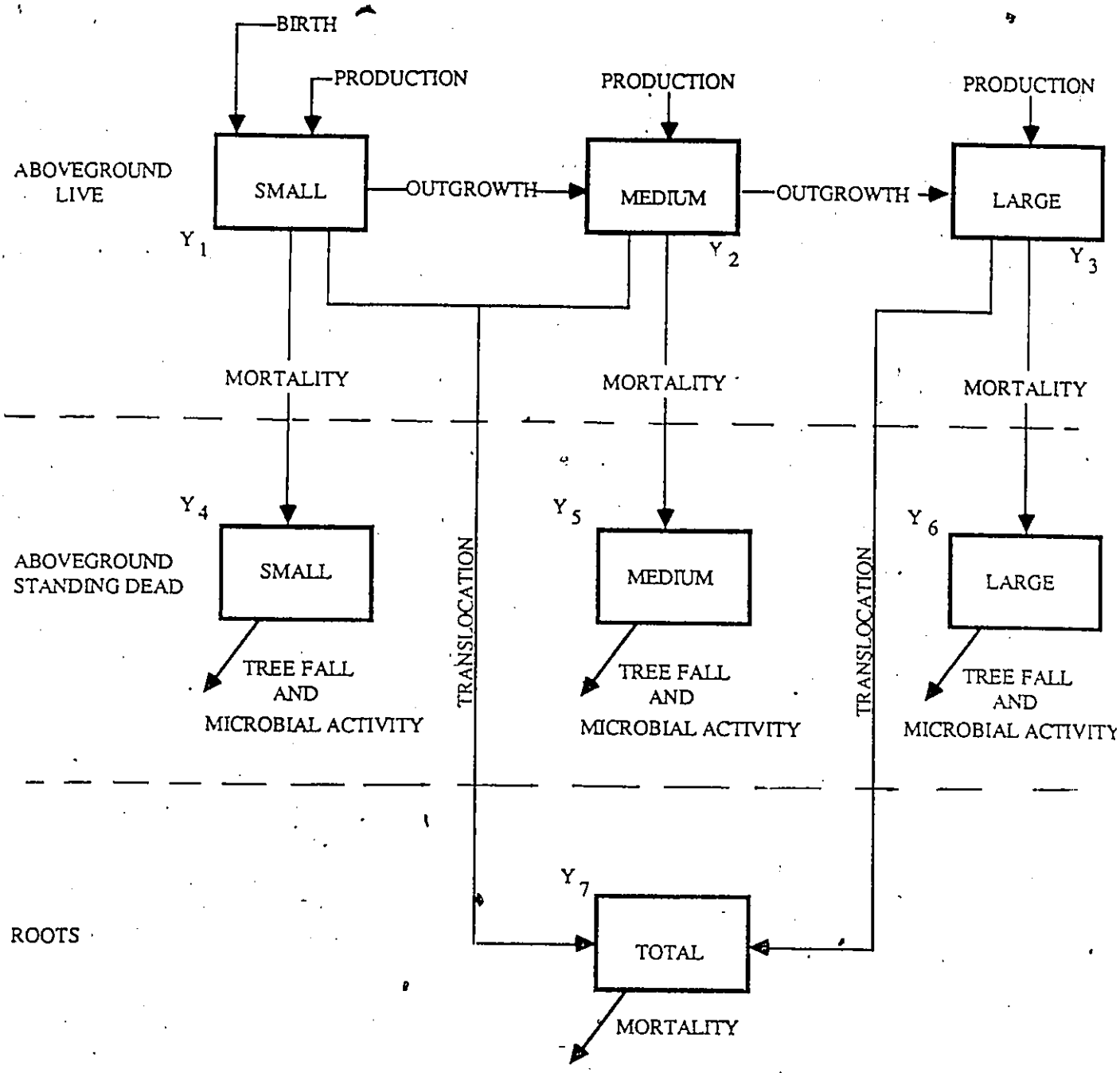


Figure F.1 Schematic diagram depicting compartments and transfers of nonfoliar biomass for SERENDIPITY

F.2 Gest Specification of The Tree Biomass Model

```
Program StudyOfTreeBiomassDynamics .

CoupledModel ForestBiomass

  ComponentModels
    LiveAbovegroundBiomass
    StandingDeadCompartments
    RootBiomass
  EndComponentModels

ContinuousModel LiveAbovegroundBiomass

    StaticStructure
      States
          Y1,                    (* Live Aboveground Biomass of the
                                    small size class                  *)
          Y2,                    (* Live Aboveground Biomass of the
                                    medium size class                 *)
          Y3                     (* Live Aboveground Biomass of the
                                    large size class                  *)
      Outputs
          Y1,
          Y2,
          Y3,
          BIOSUM                 (* Total Aboveground_Biomass         *)
      AuxiliaryVariables
          PY2[3],                (* Live Aboveground_Biomass of the
                                    medium size class in the last
                                    3 consecutive years               *)

          PY3[3]                 (* Live Aboveground_Biomass of the
                                    large size class in the last
                                    3 consecutive years               *)

      Parameters
          ProdS,                 (* Production rate in small size
                                    class *)
          ProdM,                 (* Production rate in medium size
                                    class *)
          ProdL,                 (* Production rate in large size
                                    class *)
          OutgS,                 (* Outgrowth from small size class   *)
          OutgM,                 (* Outgrowth from medium size class  *)
          OutgL,                 (* Outgrowth from large size class   *)
          MortS,                 (* Mortality from small size class   *)
          MortM,                 (* Mortality from medium size class  *)
          MortL,                 (* Mortality from large size class   *)
          YMAX,                  (* Maximum Aboveground_Live_Biomass
                                    Density                           *)
          I                      (* Ingrowth rate coefficient         *)
    EndStaticStructure
    DynamicStructure
      Derivatives
        Y1' = (ProdS - OutgS) * Y1 * (1 -(Y1 + Y2 +Y3)/YMAX)
              - MortS * Y1* (1 + (Y1 + Y2 + Y3)/YMAX) + I *
              (PY2[3] + PY3[3]) * (1 - (Y1 + Y2 + Y3)/YMAX)
        Y2' = (ProdM - OutgM) * Y2 * (1 -(Y1 + Y2 +Y3)/YMAX)
              - MortM * Y2* (1 + (Y1 + Y2 + Y3)/YMAX) + OutgS
              *Y1 * (1 - (Y1 + Y2 + Y3)/YMAX)
        Y3' = ProdS * Y3 * (1 -(Y1 + Y2 +Y3)/YMAX)
              - MortL * Y3* (1 + (Y1 + Y2 + Y3)/YMAX) + OutgM
```

```
            *Y2 * (1 - (Y1 + Y2 + Y3)/YMAX)

        PY2[3] = PY2[2]
        PY2[2] = PY2[1]
        PY2[1] = Y2
        PY3[3] = PY3[2]
        PY3[2] = PY3[1]
        PY3[1] = Y3

    EndDerivatives
    OutputFunctions
      BIOSUM = Y1 + Y2 + Y3
    EndOutputFunctions
  EndDynamicStructure
EndModel LiveAbovegroundBiomass

ContinuousModel StandingDeadCompartments

  StaticStructure
    Inputs
        Y1,
        Y2,
        Y3
    States
        Y4,                    (* Standing dead material for small
                                  size class                        *)
        Y5,                    (* Standing dead material for medium
                                  size class                        *)
        Y6                     (* Standing dead material for large
                                  size class                        *)
    Outputs
        Y4,
        Y5,
        Y6
    Parameters
        MortS,
        MortM,
        MortL,
        YMAX,
        MdS,                   (* Microbial decay rate of 'small'
                                  standing dead                     *)
        MdM,                   (* Microbial decay rate of 'medium'
                                  standing dead                     *)
        MdL,                   (* Microbial decay rate of 'large'
                                  standing dead                     *)

        FfmS,                  (* Movement small standing dead to
                                  forest floor                      *)
        FfmM,                  (* Movement medium standing dead to
                                  forest floor                      *)
        FfmL                   (* Movement large standing dead to
                                  forest floor                      *)
  EndStaticStructure
  DynamicStructure
    Derivatives
      Y4' = MortS * Y1 * (1 + (Y1 + Y2 + Y3)/YMAX) - (MdS + FfmS)
            * Y4
      Y5' = MortM * Y2 * (1 + (Y1 + Y2 + Y3)/YMAX) - (MdM + FfmM)
            * Y5
      Y6' = MortL * Y3 * (1 + (Y1 + Y2 + Y3)/YMAX) - (MdL + FfmL)
            * Y6
    EndDerivatives
  EndDynamicStructure
EndModel StandingDeadCompartments

ContinuousModel RootBiomass
```

```
    StaticStructure
      Inputs
          Y1,
          Y2,
          Y3,
          BIOSUM,
          PY2[3],
          PY3[3]
      States
          Y7                        (* Total live root biomass in all
                                       dbh classes                    *)
      Outputs
          Y7,
          Ratio                     (* Root/Shoot Ratio               *)
      Parameters
          MortS,
          MortM,
          MortL,
          YMAX,
          ProdS,
          ProdM,
          ProdL,
          I,
          RT,                       (* Root Turnover *)
          E                         (* Ratio of biomass translocated to
                                       the roots to aboveground prodn. *)
    EndStaticStructure
    DynamicStructure
      Derivatives
        Y7' = E * (ProdS*Y1 + ProdM*Y2 + ProdL*Y3 + I *
              (PY2[3] + PY3[3]) * (1-(Y1 + Y2 + Y3)/YMAX) -
              (MortS*Y1 + MortM*Y2 +  MortL*Y3) * (1 +
              (Y1 + Y2 + Y3)/YMAX) * (Y7/(Y1 + Y2 + Y3))) - RT*Y7
      EndDerivatives
      OutputFunctions
        Ratio = Y7/BIOSUM
      EndOutputFunctions
    EndDynamicStructure
  EndModel RootBiomass

InternalCoupling
  RootBiomass.Y1 <-- LiveAbovegroundBiomass.Y1
  RootBiomass.Y2 <-- LiveAbovegroundBiomass.Y2
  RootBiomass.Y3 <-- LiveAbovegroundBiomass.Y3
  RootBiomass.PY2[3] <-- LiveAbovegroundBiomass.PY2[3]
  RootBiomass.PY3[3] <-- LiveAbovegroundBiomass.PY3[3]
  StandingDeadCompartments.Y1 <-- LiveAbovegroundBiomass.Y1
  StandingDeadCompartments.Y2 <-- LiveAbovegroundBiomass.Y2
  StandingDeadCompartments.Y3 <-- LiveAbovegroundBiomass.Y3
EndInternalCoupling

ParameterSet 1
  Model LiveAbovegroundBiomass
    Parameters
      ProdS  = 0.12
      ProdM  = 0.05
      ProdL  = 0.03
      OutgS  = 0.10
      OutgM  = 0.025
      MortS  = 0.020
      MortM  = 0.010
      MortL  = 0.005
           I  = 0.001
      YMAX    = 400000
    EndParameters
```

```
    EndModel LiveAbovegroundBiomass

    Model StandingDeadCompartments
      Parameters
          MdS   = 0.0001
          MdM   = 0.0001
          MdL   = 0.0001
         FfmS   = 0.2000
         FfmM   = 0.2000
         FfmL   = 0.2000
        MortS   = 0.0200
        MortM   = 0.0100
        MortL   = 0.0050
        YMAX    = 400000
      EndParameters
    EndModel StandingDeadCompartments

    Model RootBiomass
      Parameters
        ProdS   = 0.12
        ProdM   = 0.05
        ProdL   = 0.03
        MortS   = 0.020
        MortM   = 0.010
        MortL   = 0.005
             I  = 0.001
            RT  = 0.266
             E  = 2.0
        YMAX    = 400000
      EndParameters
    EndModel RootBiomass
EndParameterSet 1

Experiment 1
    TimeUnitIs Year
      StartTime = 0.0
    IntegrateBy Runge-KuttaFehlberg
      Tol = 0.000001
    Communicate at every 3 Years
    SimulateUntil Time = 300.0
    Model LiveAbovegroundBiomass
      InitialStates
         Y1     = 10000.0
         Y2     = 30000.0
         Y3     = 60000.0
        PY2[1]  = 30000.0
        PY2[2]  = 30000.0
        PY2[3]  = 30000.0
        PY3[1]  = 60000.0
        PY3[2]  = 60000.0
        PY3[3]  = 60000.0
      EndInitialStates
      SaveOutputs Y1,Y2,Y3,BIOSUM
    EndModel LiveAbovegroundBiomass
    Model StandingDeadCompartments
      InitialStates
        Y4 = 2500.0
        Y5 = 2000.0
        Y6 = 2000.0
      EndInitialStates
      SaveOutputs Y4,Y5,Y6
    EndModel StandingDeadCompartments
    Model RootBiomass
      InitialStates
        Y7 = 32000.0
      EndInitialStates
```

```
      SaveOutputs Y7,RATIO
    EndModel RootBiomass
EndExperiment 1

Run 1
  ToObserve ForestBiomass
    WithParameterSet 1
    InExperiment 1
    WithPostRun
      OutputModule 1 On Printer
      OutputModule 2 On Printer
      OutputModule 3 On Printer
    EndPostRun
EndRun 1


OutputModule 1
  New Page
  Heading ForFirstPage 1 Line
   'Study of Aboveground_Live_Compartments'
  List Parameters And InitialConditions
  PlotandList Y1,Y2,Y3
EndOutputModule 1

OutputModule 2
  New Page
  Heading ForFirstPage 1 Line
    'Study of Aboveground_Standing_Dead_Compartments'
  PlotandList Y4,Y5,Y6
EndOutputModule 2
OutputModule 3
  New Page
  Heading ForFirstPage 1 Line
    'Study of Root_Biomass'
  List Y7,BIOSUM,RATIO
EndOutputModule 3
EndProgram StudyOfTreeBiomassDynamics
```

```
C  F.3     Equivalent Fortran Program With Results
C          For Tree Biomass Model
C
C                              Variable List
C
C Input  : N              --Dimension of the Vector of State Variables
C
C          Y(1)-Y(N)      --Initial Values of State Variables
C
C          T              --Computational Time
C
C          TINIT          --Initial Time of Simulation
C
C          TFIN           --End Time of Simulation
C
C          TCOM           --Communication Time
C
C          COMINT         --Communication Interval
C
C
C          H              --Current Integration Step Size
C
C          HINIT          --Starting Integration Step Size
C
C          HBOUND         --Upper Bound on Initial Step Size
C
C          HMIN           --Minimum step size integration is
C                            Variable List
C
C
C          TOL            --Tolerance or Accuracy specified by user
C
C
C Output : TIME           --Vector of times of Output Values
C
C          Y(1)-Y(7)      --Vector of values of Y at Output Points
C
C          BIOSUM         --Total Aboveground Production of Biomass
C
C          RATIO          --Biomass Transported to Biomass Production
C
C Program Variables :
C          Y(N)           --State Variables Vector
C
C          DY(N)          --Derivative Vector of State Variables
C
C        DY1(15)-DY4(15)--Working Vectors in Integration Routine
C                          capable of Integration systems with up
C                          to 15 state variables.
C
C        F1(15)-F3(15)  --Working Vectors in Integration Routine
C                          capable of Integrating systems with up
C                            to 15 state variables.
C
C         NFLAG          --Flag Which Signals Collection of Output
C                          Variables.
C
C          HPOS           --Intermediate variable used in computing HINIT
C
```

```
C          NPL          --Counter used for Graphics Routine.                RKB00630
C                                                                           RKB00640
C          FDER         --Maximum value of derivative functions at          RKB00650
C                         initial time                                      RKB00660
C                                                                           RKB00670
C       This Test Program Demonstrates the use of the Fourth                RKB00680
C       Order Runge-Kutta Integration Algorithm in the                      RKB00690
C       Simulation of Coupled Model Tree Biomass                            RKB00700
                                                                            RKB00710
      PROGRAM MAIN                                                          RKB00720
      IMPLICIT REAL*4 (A-H,O-Z)                                             RKB00730
      DIMENSION Y(10),DY(10),ERR(10)                                        RKB00740
      COMMON/DIFF/TIME(400),Y1(400),Y2(400),Y3(400),Y4(400),Y5(400)         RKB00750
      COMMON/DIFF1/Y6(400),Y7(400),BIO(400),RAT(400)                        RKB00760
      COMMON/PARS/DMORTS,DMORTM,DMORTL,YMAX,DMDS,DMDM,DMDL,FFMS,FFMM        RKB00770
      COMMON/PARS1/PRODS,PRODM,PRODL,DI,RT,E,OUTGS,OUTGM,FFML,H             RKB00780
      COMMON/COUNT/NK,PREVY2,PREVY3,P2(3),P3(3)                             RKB00790
      COMMON/GRA/NPL                                                        RKB00800
      COMMON/XMEN/PU(32)                                                    RKB00810
      CHARACTER*1 CHOICE                                                    RKB00820
      EXTERNAL DERIV                                                        RKB00830
                                                                            RKB00840
      PRINT*,'Welcome to the Tree Biomass Model'                            RKB00850
      PRINT*,'    '                                                         RKB00860
      PRINT*,'Would You Like To Perform a Simulation ? (Y/N) '              RKB00870
      READ*, CHOICE                                                         RKB00880
      IF (CHOICE .EQ. 'Y') THEN                                             RKB00890
         CALL INIT                                                          RKB00900
                                                                            RKB00910
                                                                            RKB00920
C                                  Initialization from Menu                 RKB00930
C                                                                           RKB00940
C                                     Initialize Parameters                 RKB00950
                                                                            RKB00960
199   DMORTS  = PU(1)                                                       RKB00970
      DMORTM  = PU(2)                                                       RKB00980
      DMORTL  = PU(3)                                                       RKB00990
      YMAX    = PU(4)                                                       RKB01000
      DMDS    = PU(5)                                                       RKB01010
      DMDM    = PU(6)                                                       RKB01020
      DMDL    = PU(7)                                                       RKB01030
      FFMS    = PU(8)                                                       RKB01040
      FFMM    = PU(9)                                                       RKB01050
      FFML    = PU(10)                                                      RKB01060
      PRODS   = PU(11)                                                      RKB01070
      PRODM   = PU(12)                                                      RKB01080
      PRODL   = PU(13)                                                      RKB01090
      DI      = PU(14)                                                      RKB01100
      RT      = PU(15)                                                      RKB01110
      E       = PU(16)                                                      RKB01120
      OUTGS   = PU(17)                                                      RKB01130
      OUTGM   = PU(18)                                                      RKB01140
                                                                            RKB01150
                                                                            RKB01160
C                                     Initialize State Variables            RKB01170
      N    = 7                                                              RKB01180
      Y(1) = PU(19)                                                         RKB01190
      Y(2) = PU(20)                                                         RKB01200
      Y(3) = PU(21)                                                         RKB01210
```

```
      Y(4) = PU(22)                                                    RKB01220
      Y(5) = PU(23)                                                    RKB01230
      Y(6) = PU(24)                                                    RKB01240
      Y(7) = PU(25)                                                    RKB01250
                                                                       RKB01260
C                                                                      RKB01270
C                                    Initialize Run Control            RKB01280
C                                          Variables                   RKB01290
      T      = PU(26)                                                  RKB01300
      TFIN   = PU(27)                                                  RKB01310
      COMINT= PU(28)                                                   RKB01320
      TOL    = PU(29)                                                  RKB01330
      HBOUND= PU(30)                                                   RKB01340
      HMIN   = PU(31)                                                  RKB01350
      HINIT = PU(32)                                                   RKB01360
                                                                       RKB01370
                                                                       RKB01380
      NPL    = 1                                                       RKB01390
      NK     = 1                                                       RKB01400
                                                                       RKB01410
      BIOSUM= Y(1)+Y(2)+Y(3)                                           RKB01420
      RATIO = BIOSUM/Y(7)                                              RKB01430
      DO 37 I = 1,3                                                    RKB01440
         P2(I)= 30000.0                                                RKB01450
         P3(I)= 60000.0                                                RKB01460
   37 CONTINUE                                                         RKB01470
                                                                       RKB01480
      PREVY2 = 30000.0                                                 RKB01490
      PREVY3 = 60000.0                                                 RKB01500
                                                                       RKB01510
                                                                       RKB01520
                                                                       RKB01530
C                                    Pre-Run Output :                  RKB01540
C                                          Set Up Output Format and OutRKB01550
C                                          put Initial Values of T and RKB01560
C     WRITE(8,17)                                                      RKB01570
C  17 FORMAT(1X,'Method:',10X,'Fourth Order Runge-Kutta '/10X/         RKB01580
C    + 1X,'Problem:',10X,'Coupled Model TreeBiomass'/10X/1X,           RKB01590
C    + 'TIME',6X,'Y1',6X,'Y2',6X,'Y3',6X,'Y4',6X,'Y5',6X,'Y6')         RKB01600
C                                                                      RKB01610
C     PRINT*,'    '                                                    RKB01620
                                                                       RKB01630
      WRITE(8,17)                                                      RKB01640
   17 FORMAT(1X,'METHOD:',10X,'Fourth Order Runge-Kutta '/10X/         RKB01650
     + 1X,'Problem:',10X,'Coupled Model TreeBiomass'/10X/1X,           RKB01660
     + 'TIME',9X,'Y7',9X,'BIOSUM',9X,'RATIO')                          RKB01670
                                                                       RKB01680
      PRINT*,'    '                                                    RKB01690
C     WRITE(8,21) T,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)                      RKB01700
C  21 FORMAT(F6.2,2X,F9.1,2X,F9.1,2X,F9.1,2X,F7.1,2X,F7.1,2X,F7.1)     RKB01710
                                                                       RKB01720
      WRITE(8,21) T,Y(7),BIOSUM,RATIO                                  RKB01730
   21 FORMAT(1X,F6.2,1X,F9.1,1X,F9.1,1X,F6.3)                          RKB01740
                                                                       RKB01750
      TIME(1) = T                                                      RKB01760
      Y1(1) = Y(1)                                                     RKB01770
      Y2(1) = Y(2)                                                     RKB01780
      Y3(1) = Y(3)                                                     RKB01790
      Y4(1) = Y(4)                                                     RKB01800
```

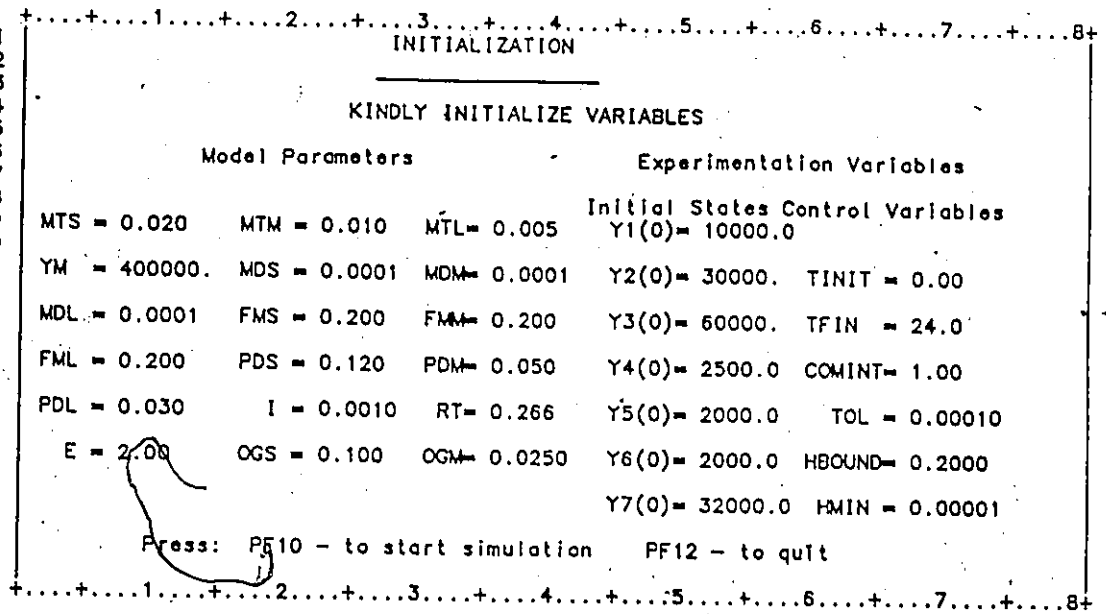
```
      Y5(1) = Y(5)                                                        RKB01810
      Y6(1) = Y(6)                                                        RKB01820
      Y7(1) = Y(7)                                                        RKB01830
      BIO(1)= BIOSUM                                                      RKB01840
      RAT(1)= RATIO                                                       RKB01850
                                                                          RKB01860
      CALL RK4(T,Y,N,TFIN,H,HINIT,HBOUND,HMIN,TOL,COMINT,DERIV)           RKB01870
                                                                          RKB01880
C                              Post-Run Output:                           RKB01890
C                                        Tabulate Values of State          RKB01900
C                                        Variables and Time at             RKB01910
C                                        Output Points                     RKB01920
C     DO 99 I = 2,NPL                                                     RKB01930
C        WRITE(8,87)TIME(I),Y1(I),Y2(I),Y3(I),Y4(I),Y5(I),Y6(I)           RKB01940
C  87    FORMAT(F6.2,2X,F9.1,2X,F9.1,2X,F9.1,2X,F6.1,2X,F6.1,2X,F8.1)      RKB01950
C        PRINT*,'  '                                                      RKB01960
C  99 CONTINUE                                                           RKB01970
                                                                          RKB01980
                                                                          RKB01990
      DO 97 I = 2,NPL                                                     RKB02000
         WRITE(8,67)TIME(I),Y7(I),BIO(I),RAT(I)                           RKB02010
   67    FORMAT(F6.2,3X,F9.1,3X,F9.1,3X,F6.3)                             RKB02020
         PRINT*,'  '                                                     RKB02030
   97 CONTINUE                                                            RKB02040
C                                        Plot Values of State              RKB02050
C                                        Variables with Time               RKB02060
      CALL ANYDEV                                                         RKB02070
      CALL GRAPH(Y1,TIME,NPL)                                             RKB02080
      CALL GRAPH(Y2,TIME,NPL)                                             RKB02090
      CALL GRAPH(Y3,TIME,NPL)                                             RKB02100
      CALL GRAPH(Y4,TIME,NPL)                                             RKB02110
      CALL GRAPH(Y5,TIME,NPL)                                             RKB02120
      CALL GRAPH(Y6,TIME,NPL)                                             RKB02130
         PRINT*,'Do you wish to carry out another simulation (Y/N) ?'     RKB02140
         READ*,CHOICE                                                     RKB02150
         IF (CHOICE.EQ.'Y') THEN                                          RKB02160
            CALL INIT                                                     RKB02170
            GOTO 199                                                      RKB02180
          ENDIF                                                           RKB02190
      ENDIF                                                               RKB02200
      STOP                                                                RKB02210
      END                                                                 RKB02220
                                                                          RKB02230
                                                                          RKB02240
                                                                          RKB02250
      SUBROUTINE GRAPH(Y1,TIME,NPL)                                       RKB03270
C         This Subroutine Provides A Graphical Display Of The             RKB03280
C         Outputs Of The Component Model With Time                        RKB03290
                                                                          RKB03300
      DIMENSION Y1(NPL),TIME (NPL)                                        RKB03310
      MINT = TIME(1)                                                      RKB03320
      MAXT = TIME(NPL)                                                    RKB03330
      VALMAX = Y1 (NPL)                                                   RKB03340
      VALMIN = Y1 (1)                                                     RKB03350
      DO 14 I = 1,NPL                                                     RKB03360
         IF (Y1 (I).GT.VALMAX) THEN                                       RKB03370
            VALMAX = Y1 (I)                                               RKB03380
         ELSE IF (Y1 (I).LT.VALMIN) THEN                                  RKB03390
            VALMIN = Y1 (I)                                               RKB03400
```

```
      END IF                                                            RKB03410
   14 CONTINUE                                                          RKB03420
      MAXVAL = VALMAX + 1                                               RKB03430
      MINVAL = VALMIN - 1                                               RKB03440
      SPAC1 = (MAXT - MINT)/10.0                                        RKB03450
      SPAC2 = (MAXVAL - MINVAL)/10.0                                    RKB03460
      CALL HWROT('MOVIE')                                               RKB03470
      CALL PAGE(8.5,11.0)                                               RKB03480
      CALL AREA2D(6.0,8.0)                                              RKB03490
      CALL SWISSM                                                       RKB03500
      CALL HEADIN('TreeBiomass Model$',100,1.5,2)                       RKB03510
      CALL HEADIN('Fourth-Order Runge-Kutta Method$',100,-1.25,2)       RKB03520
      CALL XNAME('Time(years)$',100)                                    RKB03530
      CALL YNAME('Large Trees (kg/ha)$',100)                            RKB03540
      CALL GRAF(MINT,SPAC1,MAXT,MINVAL,SPAC2,MAXVAL)                    RKB03550
C  Draw Curve with Rounded and Thickened Line and                       RKB03560
C  Rational Spline Interpolation                                        RKB03570
      CALL RASPLN (1.5)                                                 RKB03580
      CALL THKCRV (.015)                                                RKB03590
      CALL THKRND (0.)                                                  RKB03600
      CALL CURVE(TIME,Y1,NPL,0)                                         RKB03610
      CALL ENDPL(0)                                                     RKB03620
      RETURN                                                            RKB03630
      END                                                               RKB03640
                                                                        RKB03650
                                                                        RKB03660
      SUBROUTINE OUTPUT(Y,N,T)                                          RKB03670
C        This Subroutine stores values of Output Variables              RKB03680
C        at communication times. Additionally, it provides              RKB03690
C        an interactive tabular display of the results.                 RKB03700
                                                                        RKB03710
      IMPLICIT REAL*4 (A-H,O-Z)                                         RKB03720
      DIMENSION Y(N)                                                    RKB03730
      COMMON/DIFF/TIME(400),Y1(400),Y2(400),Y3(400),Y4(400),Y5(400)     RKB03740
      COMMON/DIFF1/Y6(400),Y7(400),BIO(400),RAT(400)                    RKB03750
      COMMON/GRA/NPL                                                    RKB03760
      COMMON/OUTP/BIOSUM,RATIO                                          RKB03770
                                                                        RKB03780
      WRITE(6,23) T,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7)                  RKB03790
   23 FORMAT(1X,F6.2,1X,F9.2,1X,F9.2,1X,F9.2,1X,F8.2,1X,F8.2,1X,F8.2,   RKB03800
     +       1X,F8.2)                                                   RKB03810
      NPL = NPL + 1                                                     RKB03820
      TIME(NPL) = T                                                     RKB03830
      Y1(NPL) = Y(1)                                                    RKB03840
      Y2(NPL) = Y(2)                                                    RKB03850
      Y3(NPL) = Y(3)                                                    RKB03860
      Y4(NPL) = Y(4)                                                    RKB03870
      Y5(NPL) = Y(5)                                                    RKB03880
      Y6(NPL) = Y(6)                                                    RKB03890
      Y7(NPL) = Y(7)                                                    RKB03900
      BIO(NPL)= BIOSUM                                                  RKB03910
      RAT(NPL)= RATIO                                                   RKB03920
      RETURN                                                            RKB03930
      END                                                               RKB03940
                                                                        RKB03950
                                                                        RKB03960
      SUBROUTINE DERIV(T,Y,DY,N)                                        RKB03970
      IMPLICIT REAL*4 (A-H,O-Z)                                         RKB03980
      DIMENSION Y(N),DY(N)                                              RKB03990
```

```
      COMMON/PARS/DMORTS,DMORTM,DMORTL,YMAX,DMDS,DMDM,DMDL,FFMS,FFMM    RKB04000
      COMMON/PARS1/PRODS,PRODM,PRODL,DI,RT,E,OUTGS,OUTGM,FFML,H        RKB04010
      COMMON/GRA/NPL                                                   RKB04020
      COMMON/COUNT/NK,PREVY2,PREVY3,P2(3),P3(3)                        RKB04030
      COMMON/OUTP/BIOSUM,RATIO                                         RKB04040
                                                                       RKB04050
C                                Subroutine consists of Slow System    RKB04060
C                                Derivatives which are specified by    RKB04070
C                                the user for his particular model.    RKB04080
                                                                       RKB04090
      BIOSUM = Y(1) + Y(2) + Y(3)                                      RKB04100
                                                                       RKB04110
      RATIO  = Y(7)/BIOSUM                                             RKB04120
                                                                       RKB04130
                                                                       RKB04140
                                                                       RKB04150
                                                                       RKB04160
                                                                       RKB04170
      BOUND  = ((100 * H * NK) + 0.000000001)                          RKB04180
      BOUND1 = ((100 * H * NK) - 0.000000001)                          RKB04190
                                                                       RKB04200
      IF ((T.GT.BOUND1).AND.(T.LT.BOUND)) THEN                         RKB04210
                                                                       RKB04220
         P2(1)  = Y(2)                                                 RKB04230
         P2(2)  = P2(1)                                                RKB04240
         P2(3)  = P2(2)                                                RKB04250
         PREVY2 = P2(3)                                                RKB04260
         P3(1)  = Y(3)                                                 RKB04270
         P3(2)  = P3(1)                                                RKB04280
         P3(3)  = P3(2)                                                RKB04290
         PREVY3 = P3(3)                                                RKB04300
         NK = NK + 1                                                   RKB04310
         NC = NC + 1                                                   RKB04320
      ENDIF                                                            RKB04330
                                                                       RKB04340
                                                                       RKB04350
                                                                       RKB04360
                                                                       RKB04370
                                                                       RKB04380
      DY(1) = (PRODS - OUTGS) * Y(1) * (1-(Y(1)+Y(2)+Y(3))/YMAX)       RKB04390
     +           - DMORTS * Y(1) * (1+(Y(1)+Y(2)+Y(3))/YMAX)           RKB04400
     +        + DI * (PREVY2+PREVY3) * (1-(Y(1)+Y(2)+Y(3))/YMAX)       RKB04410
                                                                       RKB04420
      DY(2) = (PRODM - OUTGM) * Y(2) * (1-(Y(1)+Y(2)+Y(3))/YMAX)       RKB04430
     +           - DMORTM * Y(2) * (1+(Y(1)+Y(2)+Y(3))/YMAX)           RKB04440
     +        + OUTGS * Y(1) * (1-(Y(1)+Y(2)+Y(3))/YMAX)               RKB04450
                                                                       RKB04460
      DY(3) = PRODS * Y(3) * (1-(Y(1)+Y(2)+Y(3))/YMAX)                 RKB04470
     +           - DMORTL * Y(3) * (1+(Y(1)+Y(2)+Y(3))/YMAX)           RKB04480
     +        + OUTGM * Y(2) * (1-(Y(1)+Y(2)+Y(3))/YMAX)               RKB04490
                                                                       RKB04500
      DY(4) = DMORTS * Y(1)*(1+(Y(1)+Y(2)+Y(3))/YMAX)-(DMDS+FFMS)*Y(4) RKB04510
      DY(5) = DMORTM * Y(2)*(1+(Y(1)+Y(2)+Y(3))/YMAX)-(DMDM+FFMM)*Y(5) RKB04520
      DY(6) = DMORTL * Y(3)*(1+(Y(1)+Y(2)+Y(3))/YMAX)-(DMDL+FFML)*Y(6) RKB04530
                                                                       RKB04540
      DY(7) = E * (PRODS * Y(1)+PRODM * Y(2)+ PRODL*Y(3) + DI *        RKB04550
     +        (PREVY2+PREVY3)*(1-(Y(1)+Y(2)+Y(3))/YMAX))-(DMORTS*      RKB04560
     +        Y(1)+MORTM*Y(2)+DMORTL*Y(3))*(1+(Y(1)+Y(2)+Y(3))/YMAX)   RKB04570
     +        *(Y(7)/(Y(1)+Y(2)+Y(3)))-RT*Y(7)                         RKB04580
```

```
      RETURN                                                            RKB04590
      END                                                               RKB04600
                                                                        RKB04610
                                                                        RKB04620
      SUBROUTINE INIT                                                   RKB04630
                                                                        RKB04640
      INTEGER NUM,RC,RLEN                                               RKB04650
      CHARACTER*8 KEY,MENUN                                             RKB04660
      CHARACTER*8 FLDN(32)                                              RKB04670
      CHARACTER*4 CC                                                    RKB04680
      COMMON/XMEN/PU(32)                                                RKB04690
                                                                        RKB04700
C        Xmenu used for Parameters and  Initial Conditions              RKB04710
                                                                        RKB04720
      FLDN(1) = 'MORTS'                                                 RKB04730
      FLDN(2) = 'MORTM'                                                 RKB04740
      FLDN(3) = 'MORTL'                                                 RKB04750
      FLDN(4) = 'YMAX'                                                  RKB04760
      FLDN(5) = 'MDS'                                                   RKB04770
      FLDN(6) = 'MDM'                                                   RKB04780
      FLDN(7) = 'MDL'                                                   RKB04790
      FLDN(8) = 'FFMS'                                                  RKB04800
      FLDN(9) = 'FFMM'                                                  RKB04810
      FLDN(10) = 'FFML'                                                 RKB04820
      FLDN(11) = 'PRODS'                                                RKB04830
      FLDN(12) = 'PRODM'                                                RKB04840
      FLDN(13) = 'PRODL'                                                RKB04850
      FLDN(14) = 'I'                                                    RKB04860
      FLDN(15) = 'RT'                                                   RKB04870
      FLDN(16) = 'E'                                                    RKB04880
      FLDN(17) = 'OUTGS'                                                RKB04890
      FLDN(18) = 'OUTGM'                                                RKB04900
      FLDN(19) = 'Y1'                                                   RKB04910
      FLDN(20) = 'Y2'                                                   RKB04920
      FLDN(21) = 'Y3'                                                   RKB04930
      FLDN(22) = 'Y4'                                                   RKB04940
      FLDN(23) = 'Y5'                                                   RKB04950
      FLDN(24) = 'Y6'                                                   RKB04960
      FLDN(25) = 'Y7'                                                   RKB04970
      FLDN(26) = 'TINIT'                                                RKB04980
      FLDN(27) = 'TFIN'                                                 RKB04990
      FLDN(28) = 'COMINT'                                               RKB05000
      FLDN(29) = 'TOL'                                                  RKB05010
      FLDN(30) = 'HBOUND'                                               RKB05020
      FLDN(31) = 'HMIN'                                                 RKB05030
      FLDN(32) = 'HINIT'                                                RKB05040
                                                                        RKB05050
      MENUN = 'BIOMASS'                                                 RKB05060
      CALL MCLSCR(NUM,RC)                                               RKB05070
      CALL MLOAD(NUM,RC,MENUN)                                          RKB05080
      CALL MPFMAP(NUM,RC,1)                                             RKB05090
      CALL MALARM(NUM,RC,1)                                             RKB05100
      CALL MPRINT(NUM,RC,2,PROUT)                                       RKB05110
      CALL MDSPRD(NUM,RC,KEY)                                           RKB05120
                                                                        RKB05130
                                                                        RKB05140
                                                                        RKB05150
                                                                        RKB05160
                                                                        RKB05170
```

```
C     CALL MEXIT(NUM,RC)                                              RKB05180
      IF (KEY .EQ. 'PF10') GOTO 25                                    RKB05190
      IF (KEY .EQ. 'PF12') STOP                                       RKB05200
   25 DO 23 I = 1,32                                                  RKB05210
         CALL MSCR2D(NUM,RC,FLDN(I),CC,7,RLEN)                        RKB05220
         CALL MCVTCF(PU(I),RC,CC,7,1)                                 RKB05230
   23 CONTINUE                                                        RKB05240
      RETURN                                                          RKB05250
      END                                                             RKB05260
                                                                      RKB05270
                                                                      RKB05280
```



Menu For Tree Biomass Model

METHOD: Fourth Order Runge-Kutta

Problem: Coupled Model TreeBiomass

TIME	Y1	Y2	Y3	Y4	Y5	Y6
0.00	10000.0	30000.0	60000.0	2500.0	2000.0	2000.0
3.00	9888.3	32664.3	78320.3	1944.7	2005.4	2104.5
6.00	9743.5	34980.5	99567.7	1657.3	2122.5	2506.4
9.00	9556.9	36877.9	123168.4	1516.3	2299.6	3144.9
12.00	9320.9	38304.9	148201.9	1453.0	2503.4	3980.5
15.00	9031.1	39238.6	173532.9	1427.9	2710.7	4974.7
18.00	8687.6	39688.7	198020.6	1417.3	2903.7	6083.2
21.00	8295.7	39694.7	220699.1	1407.3	3069.1	7252.4
24.00	7865.5	39317.5	240922.1	1390.0	3198.3	8426.3
27.00	7409.3	38629.5	258400.5	1361.5	3286.8	9554.1
30.00	6940.6	37703.7	273144.7	1321.2	3334.5	10597.0
33.00	6471.6	36606.8	285369.9	1270.3	3344.3	11530.1
36.00	6012.4	35395.3	295400.6	1211.1	3321.1	12343.1
39.00	5570.8	34114.4	303594.9	1146.1	3270.8	13036.5
42.00	5152.2	32798.9	310295.2	1078.0	3199.4	13618.9
45.00	4759.9	31474.5	315804.4	1009.0	3112.4	14103.0
48.00	4395.5	30159.6	320375.1	940.9	3014.5	14503.2
51.00	4059.4	28867.2	324211.3	875.1	2909.5	14833.8
54.00	3751.5	27606.0	327474.1	812.5	2800.5	15107.5
57.00	3470.2	26381.9	330287.8	753.6	2689.9	15335.6
60.00	3214.0	25198.3	332747.8	698.9	2579.4	15527.3
63.00	2981.3	24057.3	334926.9	648.3	2470.2	15690.3
66.00	2770.4	22959.8	336879.8	601.9	2363.3	15830.5
69.00	2579.5	21906.0	338648.4	559.4	2259.2	15952.6
72.00	2407.0	20895.4	340264.7	520.7	2158.4	16060.3
75.00	2251.1	19927.3	341752.0	485.5	2061.0	16156.4
78.00	2110.4	19000.8	343129.7	453.5	1967.3	16243.1
81.00	1983.5	18114.7	344411.8	424.6	1877.2	16321.9
84.00	1869.0	17267.7	345610.1	398.5	1790.7	16394.2
87.00	1765.8	16458.6	346732.9	374.8	1707.9	16461.0
90.00	1672.7	15686.0	347788.9	353.5	1628.6	16522.9
93.00	1588.8	14948.5	348782.6	334.2	1552.7	16580.7
96.00	1513.2	14244.7	349719.9	316.9	1480.2	16634.8
99.00	1445.1	13573.4	350603.8	301.2	1411.0	16685.6
102.00	1383.6	12933.2	351440.2	287.0	1344.9	16733.4
105.00	1328.2	12322.9	352230.5	274.3	1281.7	16778.3
108.00	1278.3	11741.1	352978.9	262.8	1221.5	16820.7
111.00	1233.3	11186.7	353687.5	252.4	1164.1	16860.8
114.00	1192.6	10658.4	354358.2	243.1	1109.3	16898.7
117.00	1156.0	10155.2	354993.7	234.7	1057.1	16934.5
120.00	1122.9	9675.9	355596.6	227.1	1007.3	16968.4
123.00	1093.0	9219.5	356168.0	220.2	959.9	17000.6
126.00	1066.1	8784.9	356708.7	214.0	914.7	17031.0
129.00	1041.7	8371.2	357222.8	208.4	871.7	17059.8
132.00	1019.7	7977.4	357709.3	203.3	830.7	17087.2
135.00	999.8	7602.5	358170.6	198.8	791.7	17113.2
138.00	981.8	7245.8	358608.9	194.7	754.8	17137.7
141.00	965.6	6906.3	359025.3	190.9	719.1	17161.0
144.00	950.8	6583.3	359419.9	187.6	685.5	17183.1
147.00	937.5	6276.1	359793.6	184.5	653.4	17204.1
150.00	925.4	5983.7	360148.6	181.7	622.9	17224.1
153.00	914.5	5705.7	360485.9	179.2	593.9	17242.8

METHOD: Fourth Order Runge-Kutta

Problem: Coupled Model TreeBiomass

TIME	Y1	Y2	Y3	Y4	Y5	Y6
156.00	904.6	5441.2	360804.8	177.0	566.3	17260.6
159.00	895.6	5189.6	361107.0	174.9	540.0	17277.6
162.00	887.4	4950.4	361394.1	173.1	515.0	17293.7
165.00	880.0	4722.8	361667.7	171.4	491.3	17309.0
168.00	873.3	4506.5	361929.4	169.9	468.7	17323.4
171.00	867.2	4300.8	362173.3	168.5	447.2	17337.3
174.00	861.6	4105.2	362406.5	167.2	426.7	17350.2
177.00	856.5	3919.7	362631.5	166.1	407.3	17362.7
180.00	851.9	3743.4	362838.2	165.0	388.8	17374.5
183.00	847.7	3575.7	363039.9	164.1	371.3	17385.8
186.00	843.9	3416.4	363227.4	163.2	354.6	17396.3
189.00	840.4	3264.9	363408.9	162.4	338.8	17406.6
192.00	837.2	3120.9	363577.6	161.7	323.7	17415.9
195.00	834.2	2984.0	363743.0	161.0	309.4	17425.3
198.00	831.6	2853.9	363893.0	160.4	295.7	17433.5
201.00	829.1	2730.2	364043.0	159.9	282.8	17441.7
204.00	826.9	2612.7	364178.5	159.4	270.5	17449.6
207.00	824.8	2501.0	364309.7	158.9	258.8	17456.6
210.00	822.9	2394.8	364440.7	158.5	247.7	17463.7
213.00	821.2	2293.9	364553.2	158.1	237.1	17470.6
216.00	819.6	2198.1	364665.7	157.8	227.1	17476.4
219.00	818.1	2106.9	364778.2	157.5	217.6	17482.3
222.00	816.8	2020.4	364875.9	157.2	208.5	17488.1
225.00	815.6	1938.1	364969.7	156.9	199.9	17493.7
228.00	814.4	1859.9	365063.4	156.6	191.7	17498.5
231.00	813.4	1785.6	365157.2	156.4	183.9	17503.5
234.00	812.4	1715.0	365233.6	156.2	176.5	17508.2
237.00	811.5	1647.9	365308.6	156.0	169.5	17512.9
240.00	810.7	1584.2	365383.6	155.8	162.8	17516.8
243.00	809.9	1523.6	365458.6	155.6	156.4	17520.7
246.00	809.2	1466.0	365531.9	155.5	150.4	17524.6
249.00	808.5	1411.4	365588.1	155.3	144.7	17528.1
252.00	807.9	1359.4	365644.4	155.2	139.2	17531.7
255.00	807.3	1310.0	365700.6	155.1	134.1	17534.8
258.00	806.8	1262.4	365757.7	155.0	129.1	17537.8
261.00	806.3	1216.9	365815.3	154.9	124.3	17540.8
264.00	805.8	1173.7	365872.9	154.8	119.8	17543.9
267.00	805.4	1132.6	365917.9	154.7	115.5	17546.5
270.00	805.0	1093.7	365956.8	154.6	111.4	17548.9
273.00	804.6	1056.8	365995.2	154.5	107.5	17551.3
276.00	804.3	1021.8	366033.6	154.4	103.8	17553.7
279.00	804.0	988.5	366071.9	154.4	100.4	17555.7
282.00	803.7	957.0	366110.3	154.3	97.0	17557.7
285.00	803.4	927.0	366148.7	154.2	93.9	17559.7
288.00	803.1	898.6	366187.1	154.2	90.9	17561.7
291.00	802.8	871.6	366225.4	154.1	88.1	17563.8
294.00	802.5	846.0	366257.1	154.1	85.4	17565.5
297.00	802.3	821.7	366281.2	154.0	82.9	17566.7
300.00	802.1	798.7	366304.1	154.0	80.5	17567.9

METHOD: Fourth Order Runge-Kutta

Problem: Coupled Model TreeBiomass

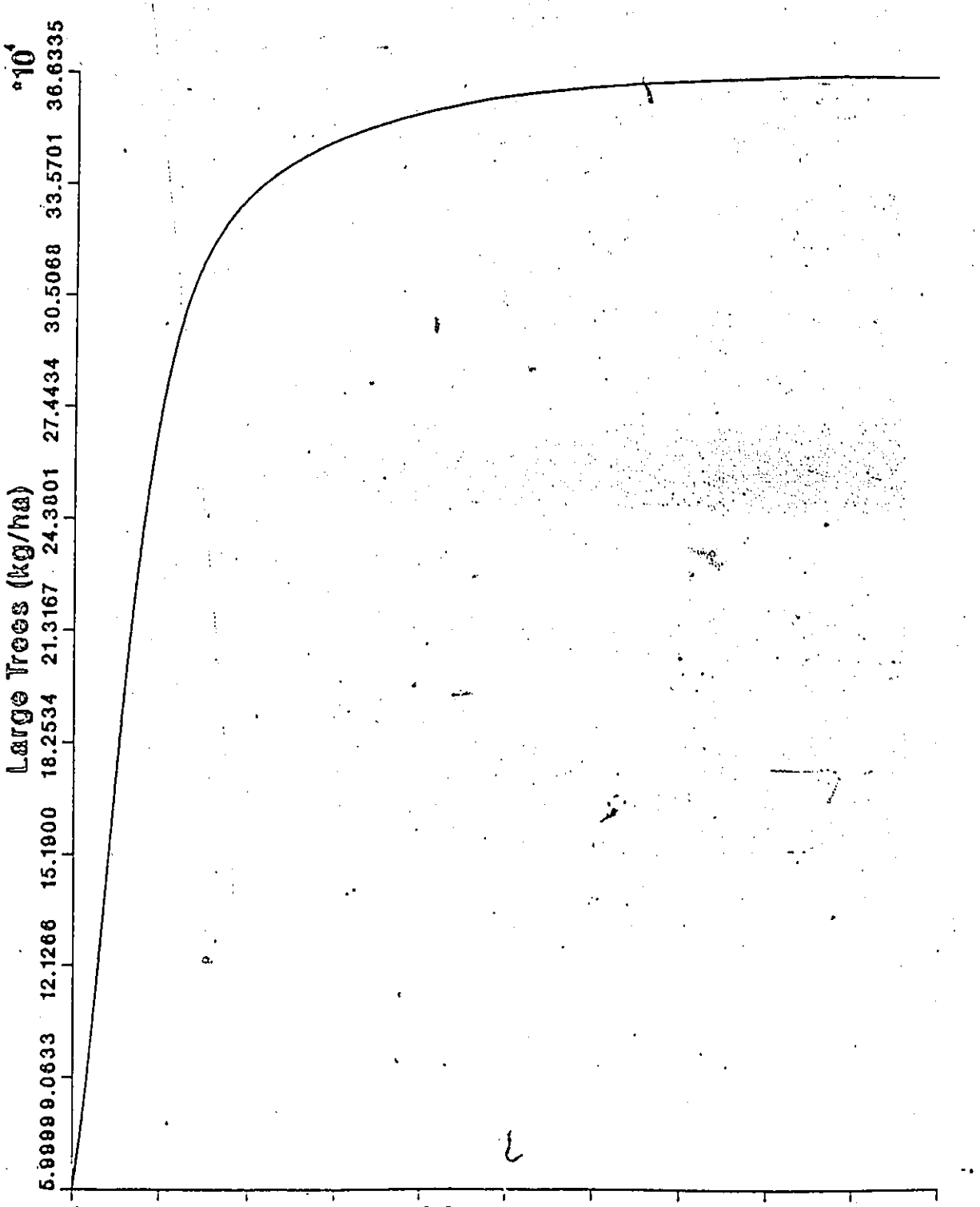
TIME	Y7	BIOSUM	RATIO
0.00	32000.0	100000.0	3.125
3.00	34419.3	120872.9	0.285
6.00	38419.5	144291.7	0.266
9.00	43326.7	169603.2	0.255
12.00	48726.8	195827.7	0.249
15.00	54295.1	221802.6	0.245
18.00	59747.4	246396.8	0.242
21.00	64839.2	268689.4	0.241
24.00	69381.4	288105.0	0.241
27.00	73273.7	304439.4	0.241
30.00	76480.9	317789.1	0.241
33.00	79027.5	328448.2	0.241
36.00	80978.8	336808.2	0.240
39.00	82420.8	343280.1	0.240
42.00	83445.2	348246.3	0.240
45.00	84139.2	352038.8	0.239
48.00	84578.4	354930.1	0.238
51.00	84826.6	357137.9	0.238
54.00	84935.1	358831.6	0.237
57.00	84944.3	360139.8	0.236
60.00	84883.5	361160.1	0.235
63.00	84775.4	361965.5	0.234
66.00	84635.9	362610.0	0.233
69.00	84479.2	363133.9	0.233
72.00	84310.4	363567.0	0.232
75.00	84141.5	363930.4	0.231
78.00	83972.6	364240.9	0.231
81.00	83807.3	364509.9	0.230
84.00	83654.6	364746.7	0.229
87.00	83504.4	364957.2	0.229
90.00	83354.3	365147.5	0.228
93.00	83217.7	365319.8	0.228
96.00	83086.3	365477.8	0.227
99.00	82958.5	365622.2	0.227
102.00	82845.9	365757.0	0.227
105.00	82733.2	365881.6	0.226
108.00	82628.4	365998.3	0.226
111.00	82534.6	366107.4	0.225
114.00	82440.7	366209.2	0.225
117.00	82349.8	366304.9	0.225
120.00	82273.9	366395.4	0.225
123.00	82198.8	366480.5	0.224
126.00	82123.7	366559.6	0.224
129.00	82050.9	366635.7	0.224
132.00	81991.6	366706.4	0.224
135.00	81934.9	366772.9	0.223
138.00	81878.6	366836.6	0.223
141.00	81822.2	366897.2	0.223
144.00	81765.8	366954.1	0.223
147.00	81719.4	367007.1	0.223
150.00	81677.7	367057.7	0.223
153.00	81638.2	367106.1	0.222
156.00	81600.1	367150.6	0.222
159.00	81562.4	367192.2	0.222
162.00	81524.8	367231.9	0.222
165.00	81487.2	367270.5	0.222
168.00	81449.6	367309.2	0.222
171.00	81419.9	367341.2	0.222

METHOD: Fourth Order Runge-Kutta

Problem: Coupled Model TreeBiomass

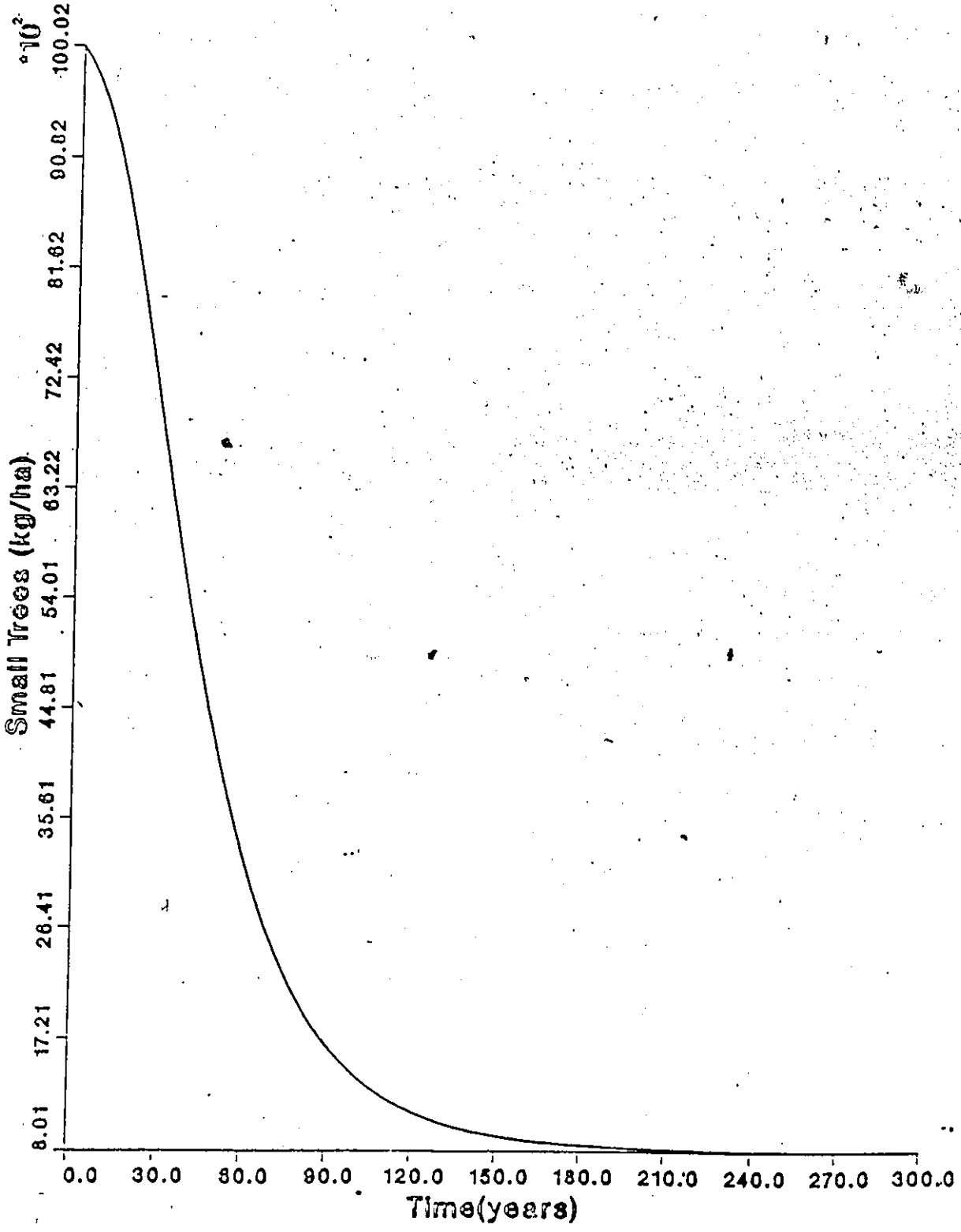
TIME	Y7	BIOSUM	RATIO
174.00	81393.2	367373.3	0.222
177.00	81368.8	367407.7	0.221
180.00	81344.3	367433.4	0.221
183.00	81322.2	367463.2	0.221
186.00	81300.5	367487.6	0.221
189.00	81280.6	367514.1	0.221
192.00	81261.0	367535.6	0.221
195.00	81242.1	367561.2	0.221
198.00	81223.2	367578.4	0.221
201.00	81204.4	367602.3	0.221
204.00	81185.6	367618.0	0.221
207.00	81166.9	367635.5	0.221
210.00	81148.1	367658.4	0.221
213.00	81129.7	367668.3	0.221
216.00	81117.2	367683.4	0.221
219.00	81106.4	367703.3	0.221
222.00	81094.4	367713.1	0.221
225.00	81083.2	367723.2	0.221
228.00	81073.5	367737.7	0.220
231.00	81065.3	367756.1	0.220
234.00	81054.9	367760.9	0.220
237.00	81045.6	367767.9	0.220
240.00	81037.4	367778.4	0.220
243.00	81030.6	367792.0	0.220
246.00	81024.4	367807.1	0.220
249.00	81015.8	367807.9	0.220
252.00	81008.3	367811.6	0.220
255.00	81001.7	367817.9	0.220
258.00	80996.0	367826.9	0.220
261.00	80991.1	367838.4	0.220
264.00	80987.1	367852.4	0.220
267.00	80981.3	367855.9	0.220
270.00	80975.0	367855.5	0.220
273.00	80969.3	367856.6	0.220
276.00	80964.4	367859.6	0.220
279.00	80960.1	367864.4	0.220
282.00	80956.4	367870.9	0.220
285.00	80953.4	367879.1	0.220
288.00	80950.9	367888.7	0.220
291.00	80948.9	367899.8	0.220
294.00	80946.0	367905.6	0.220
297.00	80942.1	367905.2	0.220
300.00	80938.3	367904.8	0.220

TreeBiomass Model
Fourth-Order Runge-Kutta Method



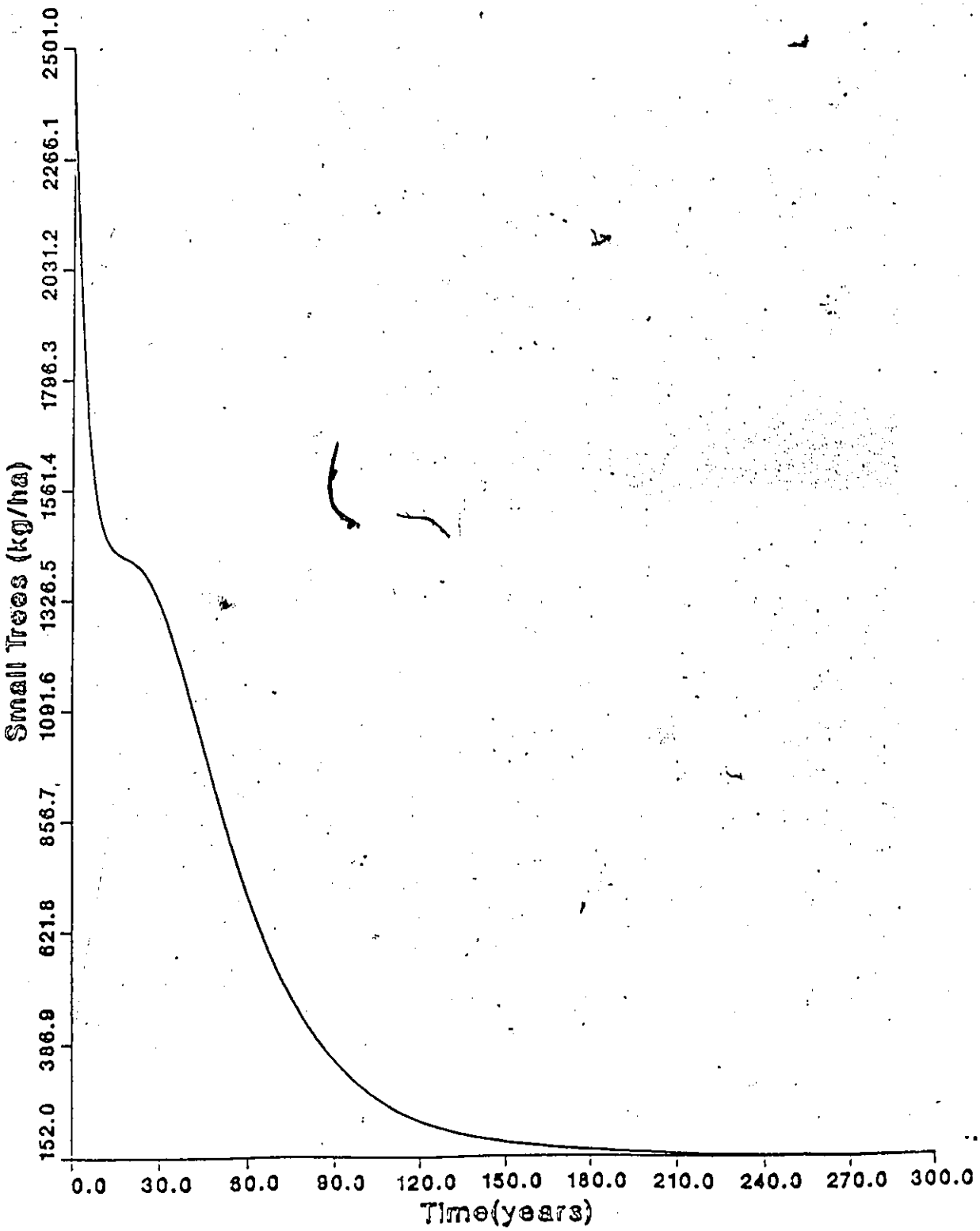
Live Aboveground Biomass

TreeBiomass Model
Fourth-Order Runge-Kutta Method



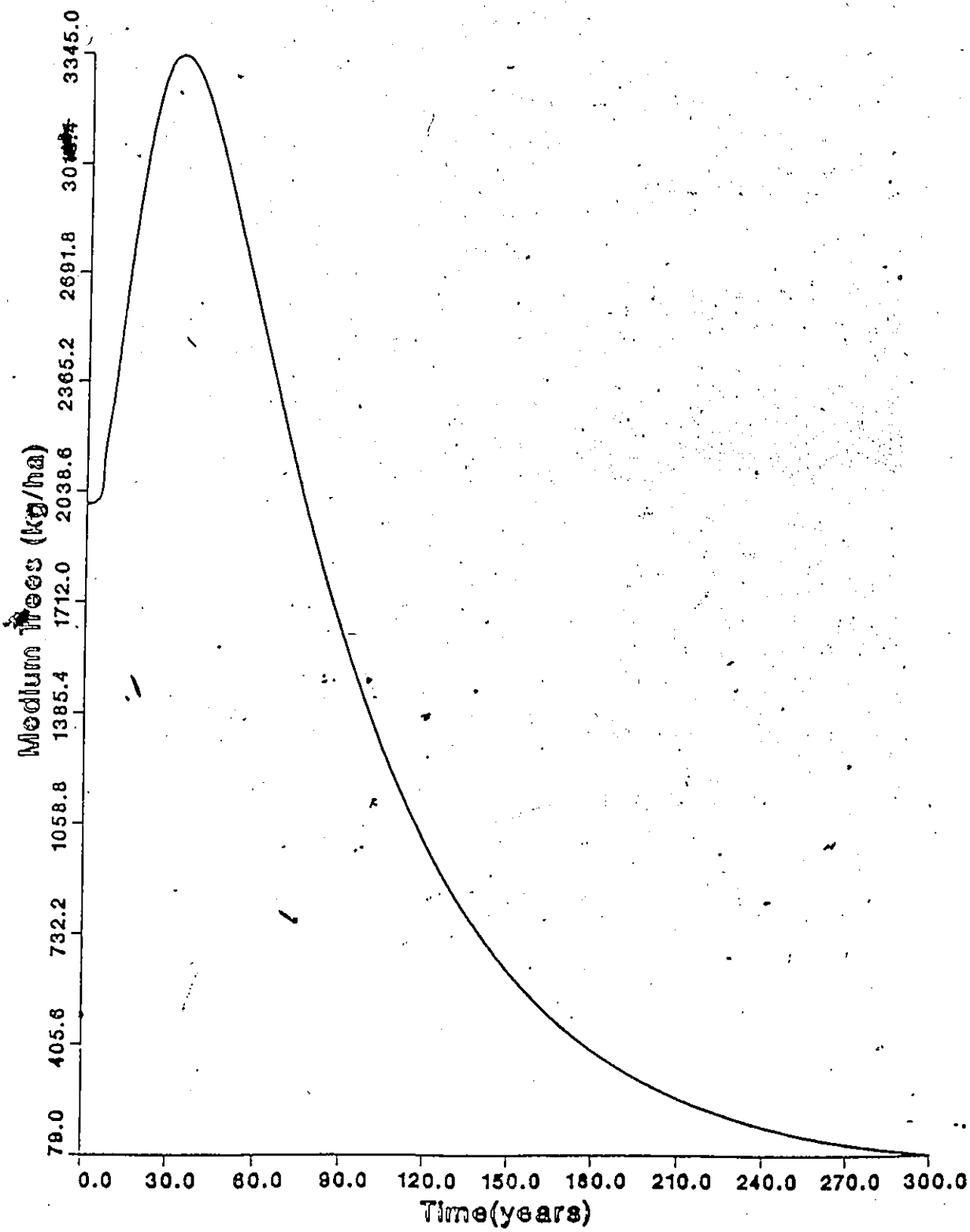
Live Aboveground Biomass

TreeBiomass Model
Fourth-Order Runge-Kutta Method



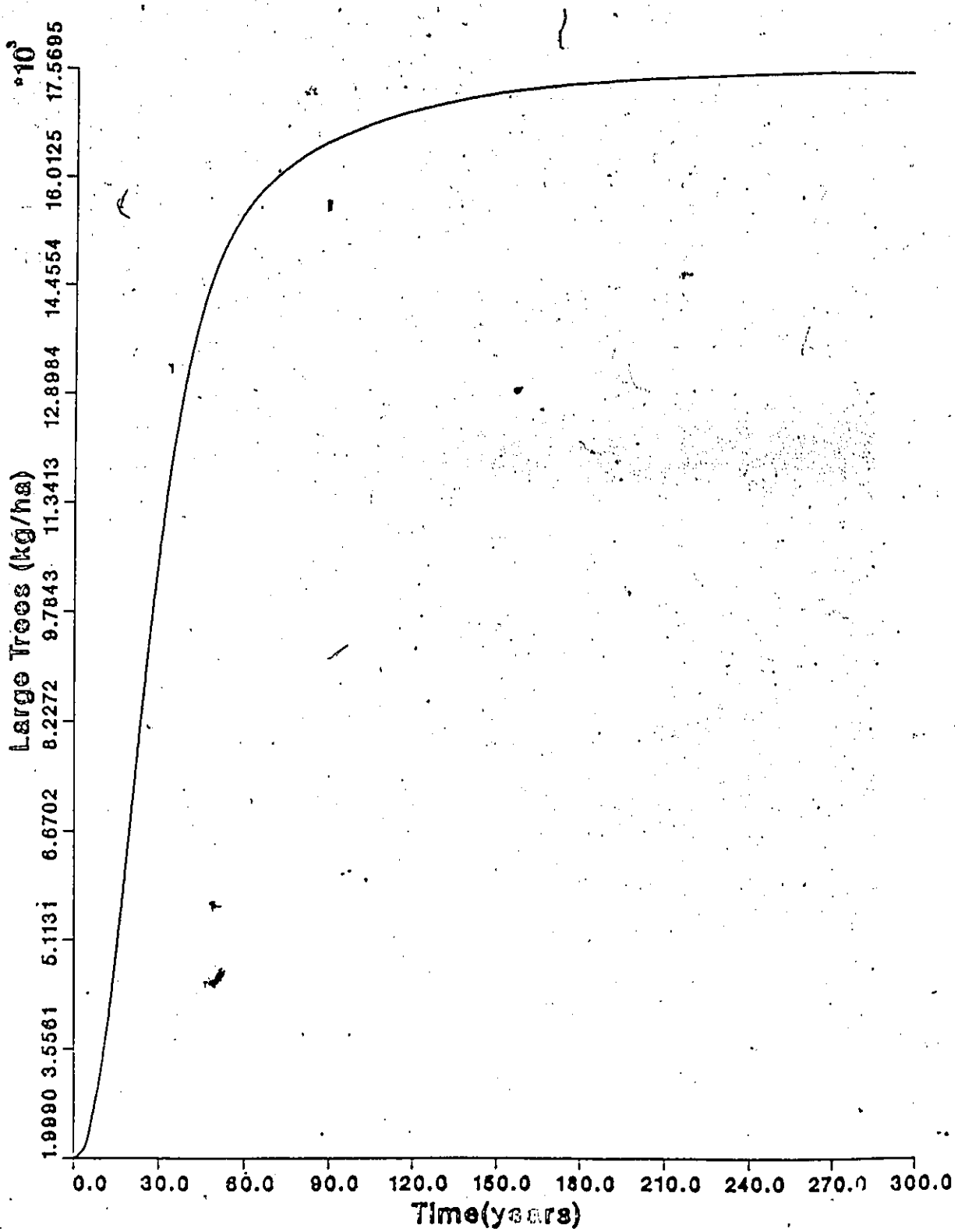
Standing Dead Biomass

TreeBiomass Model
Fourth-Order Runge-Kutta Method



Standing Dead Biomass

TreeBiomass Model
Fourth-Order Runge-Kutta Method



Standing Dead Biomass

Appendix G. THE ELECTRONIC OSCILLATOR MODEL (Revisited)

G.1 Problem Definition

This test example utilizes the multi-rate integration algorithm of the run-time library in the revisitation of the electronic oscillator model as described in Appendix B. A block diagram of the partitioned model for use of the multi-rate algorithm is shown in Figure G.1 . The full details of the specification of this multi-rate model is given in Section G.2. As can be seen, the results follow closely to those in Appendix B the algorithm requiring significantly lower CPU-time.

Fast Routine

$$Y1' = Y2$$

$$Y2' = -64Y1 + 7.5(2 + w1 - Y1^2)Y2$$

$$X1' = \frac{2}{R1} - Y1Y2 + \frac{1}{R1C1}(X2 - X1)$$

$$X2' = \frac{1}{R2C1} X1 - \left(\frac{1}{R2C1} + \frac{1}{R2C2}\right) X2 + \frac{1}{R1C1} W2$$

X2

W1 = s

W2 = X3

Slow Routine

$$X3' = \frac{1}{R3C2} X2 - \left(\frac{1}{R3C2} + \frac{1}{R3C3}\right) X2 + \frac{1}{R3C3} X4$$

$$X4' = \frac{1}{R4C3} X3 - \left(\frac{1}{R4C3} + \frac{1}{R4C4}\right) X4 + \frac{1}{R4C4} X5$$

$$X5' = \frac{1}{R5C4} X4 - \left(\frac{1}{R5C4} + \frac{1}{R5C5}\right) X5 + \frac{1}{R5C5} X6$$

$$X6' = \frac{1}{R6C5} X5 - \left(\frac{1}{R6C5} + \frac{1}{R6C6}\right) X6$$

$$V' = \frac{1}{C6} X6$$

$$S' = 0.02\left|36 - 5.29\left(|V|\right)^{.5}\right|^{1.2} \text{sign}\left|36 - 5.29\left(|V|\right)^{.5}\right| - 0.02s$$

Figure G.1 Partitioned Model Of Electronic Oscillator

G.2 Gest Specification of The Electronic Oscillator Model (Revisited)

```
Program StudyOfElectronicOscillator

MutirateModel ElectronicOscillator      (* A Coupled Model with 2 Continuous
  ComponentModels                       (* Component Models which  has been
    SlowContinuousModel  Slow           (* Carefully partitioned  to utilize
    FastContinuousModel  Fast           (* Multi-Rate  Integration          *)
  EndComponentModels

  ContinuousModel  Slow
    StaticStructure
      Inputs
          X2
      States
          X3,
          X4,
          X5,
          X6,
          V,
          S
      Outputs
          X3,
          S
      Parameters
          R3,              (* Resistance in kOhms                          *)
          R4,              (* Resistance in kOhms                          *)
          R5,              (* Resistance in kOhms                          *)
          R6,              (* Resistance in kOhms                          *)
          C2,              (* Capacitance in microFarads                   *)
          C3,              (* Capacitance in microFarads                   *)
          C4,              (* Capacitance in microFarads                   *)
          C5,              (* Capacitance in microFarads                   *)
          C6               (* Capacitance in microFarads                   *)
    EndStaticStructure
    DynamicStructure
      Derivatives
        X3' = X2/R3*C2 - (1/R3*C2 + 1/R3*C3)*X3 + X4/R3*C3
        X4' = X3/R4*C3 - (1/R4*C3 + 1/R4*C4)*X4 + X5/R4*C4
        X5' = X4/R5*C4 - (1/R5*C4 + 1/R5*C5)*X5 + X6/R5*C5
        X6' = X5/R6*C5 - (1/R6*C5 + 1/R6*C6)*X6
         V' = X6/C6
         S' = 0.02*Abs(36 - 5.29 * Sqrt(Abs(V))) ** 1.2 *
              Sign(Abs(36 - 5.29 * Sqrt(Abs(V))) - 0.02*S)
      EndDerivatives
    EndDynamicStructure
  EndModel  Slow

ContinuousModel  Fast
    StaticStructure
      Inputs
          S,
          X3
      States
          Y1,
          Y2,
          X1,
          X2
      Outputs
          X2
      Parameters
          R1,              (* Resistance in kOhms                          *)
```

```
        R2,                 (* Resistance in kOhms                    *)
        C1,                 (* Capacitance in microFarads              *)
        C2                  (* Capacitance in microFarads              *)
    EndStaticStructure
    DynamicStructure
      Derivatives
        Y1' = Y2
        Y2' = -64*Y1 + 7.5*(2 + S - Y1**2.0) * Y2
        X1' = 2*Y1*Y2/R1 + (X2-X1)/R1*C1
        X2' = X1/R2*C1 - (1/R2*C1 + 1/R2*C2)*X2 + X3/R1*C1
      EndDerivatives
    EndDynamicStructure
  EndModel  Fast

  InternalCoupling
    Slow.X2   <--  Fast.X2
    Fast.S    <--  Slow.S
    Fast.X3   <--  Slow.X3
  EndInternalCoupling
EndModel ElectronicOscillator


ParameterSet 1
  Model  Slow
    Parameters
      R3 = 8.0
      R4 = 8.0
      R5 = 8.0
      R6 = 8.0
      C2 = 4.25
      C3 = 1.25
      C4 = 1.25
      C5 = 1.25
      C6 = 1.25
    EndParameters
  EndModel

  Model  Fast
    Parameters
      R1 = 8.0
      R2 = 8.0
      C1 = 6.25
      C2 = 4.25
    EndParameters
  EndModel
EndParameterSet 1


Experiment 1
  Global
    TimeUnitIs Second
      StartTime = 0.0
    IntegrateBy Multi-Rate Integration
    Communicate AtEvery 0.2 Second
    SimulateUntil Time = 20.0
  EndGlobal

  Model  Slow
    InitialStates
      X3 = 40.0
      X4 = 45.0
      X5 = 50.0
      X6 = 55.0
       V = 60.0
       S = 65.0
    EndInitialStates
```

```
    Tolers = 0.50
    SaveOutputs X3,S
  EndModel  Slow

  Model  Fast
    InitialStates
      Y1 = 60.0
      Y2 = 70.0
      X1 = 80.0
      X2 = 85.0
    EndInitialStates
    Tolerf = 0.01
  EndModel  Fast
EndExperiment 1


Run 1
  ToObserve ElectronicOscillator
    WithParameterSet 1
   •InExperiment 1
    WithPostRun
      OutputModule 1 On Printer
    EndPostRun
EndRun 1


OutputModule 1
  New Page
  Heading ForFirstPage 1 Line
    'Study of ElectronicOscillator'
  List Parameters And InitialConditions
  PlotandList X3,S
EndOutputModule 1


EndProgram StudyOfElectronicOscillator
```

```
C  G.3     Equivalent Fortran Program With Results For                    MUL00020
C          Electronic Oscillator Model:Revisited                         MUL00030
                                                                         MUL00040
                                                                         MUL00050
                                                                         MUL00060
                                                                         MUL00070
C                          Variable List                                 MUL00080
C                                                                        MUL00090
CINPUT: NS-- The Dimension of the Vector of Slow System State Variables  MUL00100
C                                                                        MUL00110
C       NF-- The Dimension of the Vector of Fast System State Variables  MUL00120
C                                                                        MUL00130
C       S(1)..S(N)-- Initial values of Slow System State Variables       MUL00140
C                                                                        MUL00150
C       F(1)..F(M)-- Initial values of Fast System State Variables       MUL00160
C                                                                        MUL00170
C       W1..W4-- Dummy Arrays                                            MUL00180
C                                                                        MUL00190
C       DF1..DF5   Required For RK-Merson Integration                    MUL00200
C                                                                        MUL00210
C       HS-- Slow System Integration Step Size                           MUL00220
C                                                                        MUL00230
C       HSMIN -- Minimum Allowable Slow System Integration Step Size     MUL00240
C                                                                        MUL00250
C       HBOUND-- Upper Bound On Initial Step Size                        MUL00260
C                                                                        MUL00270
C       HF    -- Fast System Integration Step Size                       MUL00280
C                                                                        MUL00290
C       HFMIN -- Minimum Allowable Fast System Integration Step Size     MUL00300
C                                                                        MUL00310
C       TOLERS-- User Tolerance for Slow System                          MUL00320
C                                                                        MUL00330
C       TOLERF-- User Tolerance for Fast System                          MUL00340
C                                                                        MUL00350
C       T     --Computational Time                                       MUL00360
C                                                                        MUL00370
C       TINIT -- Initial Value of Time                                   MUL00380
C                                                                        MUL00390
C       TFIN  -- End Time of Simulation                                  MUL00400
C                                                                        MUL00410
C       TCOM  -- Communication Time                                      MUL00420
C                                                                        MUL00430
C       COMINT-- Communication Interval                                  MUL00440
C                                                                        MUL00450
COutput:T-- Time at Output Points                                        MUL00460
C                                                                        MUL00470
C       S(K)--Values of Slow System Variable at Output points            MUL00480
C                                                                        MUL00490
CProgram Variables :                                                     MUL00500
C       S(NS) --State Variables Vector                                   MUL00510
C                                                                        MUL00520
C       DS(NS)--Vector of Derivatives of Slow System State Variables     MUL00530
C                                                                        MUL00540
C       PS(NS)--Vector of Predicted Slow System State Variables at       MUL00550
C               TN+HS/3                                                  MUL00560
C                                                                        MUL00570
C       DPS(NS)--Vector of Derivatives of Slow System State Variables    MUL00580
C               at TN+HS/3 using Predicted Values                        MUL00590
C                                                                        MUL00600
```

```
C       P1S(NS)--Vector of Corrected Slow System State Variables at      MUL00610
C              TN+HS/3                                                   MUL00620
C                                                                        MUL00630
C       DP1S(NS)--Vector of Derivatives of Slow System State Variables   MUL00640
C              at TN+HS/3                                                MUL00650
C                                                                        MUL00660
C       P2S(NS)--Vector of Estimates of State Variable Values at         MUL00670
C              TN + HS/2                                                 MUL00680
C                                                                        MUL00690
C       DP2S(NS)--Vector of Derivatives of Slow System State Variables   MUL00700
C              at TN+HS/2                                                MUL00710
C                                                                        MUL00720
C       P3S(NS)--Vector of Estimates of State Variable Values at         MUL00730
C              TN + HS                                                   MUL00740
C                                                                        MUL00750
C       DP3S(NS)--Vector of Derivatives of Slow System State Variables   MUL00760
C              at TN+HS                                                  MUL00770
C                                                                        MUL00780
C       INTS(2)--Vector of Interpolated/Extrapolated Values of Slow      MUL00790
C              System Variables involved in coupling between TN          MUL00800
C              and TN + HS                                               MUL00810
C                                                                        MUL00820
C                                                                        MUL00830
C                                                                        MUL00840
C                                                                        MUL00850
C                                                                        MUL00860
C       ERR(NS)--Vector of Local Error Estimates                         MUL00870
C              For Slow System Variables                                 MUL00880
C                                                                        MUL00890
C       ERRMAX--Variable containing Maximum Error                        MUL00900
C              For Slow System Variables                                 MUL00910
C                                                                        MUL00920
C       ER(NF)--Vector of Local Error Estimates                          MUL00930
C              For Fast System Variables                                 MUL00940
C                                                                        MUL00950
C       ERMAX--Variable containing Maximum Error                         MUL00960
C                                                                        MUL00970
C       HPOS --Intermediate Variable Used in determining HINIT           MUL00980
C                                                                        MUL00990
C       NPL  --Counter used for Graphics Routine                         MUL01000
C                                                                        MUL01010
C       FDER --Maximum value of Derivative Functions                     MUL01020
C              at Initial Time                                           MUL01030
C                                                                        MUL01040
C       NFLAG--Signals The Detection Of A Communication Time             MUL01050
C                                                                        MUL01060
C                                                                        MUL01070
C                                                                        MUL01080
C                              Main Program                              MUL01090
      PROGRAM MAIN                                                       MUL01100
C                                                                        MUL01110
C                             This program utilizes a Multi-             MUL01120
C                             Rate Integration Algorithm based           MUL01130
C                             on Runge-Kutta Merson's to simulate        MUL01140
C                             a particular electronic oscillator         MUL01150
C                             model.                                     MUL01160
C                                                                        MUL01170
C                              Variable Declaration                      MUL01180
                                                                         MUL01190
```

```
      IMPLICIT REAL(A-H,O-Z)                                          MUL01200
      REAL S(8),F(6)                                                  MUL01210
      CHARACTER*1 CHOICE                                              MUL01220
      COMMON/DIFF/PU(29)                                              MUL01230
      COMMON/PARS/C1,C2,C3,C4,C5,C6,R1,R2,R3,R4,R5,R6                 MUL01240
      COMMON/FAST/TOLERF,HFMAX,HFMIN                                  MUL01250
      COMMON/SLOW/TOLERS,HSMAX,HSMIN,COMINT                           MUL01260
      COMMON/OUT/NPL,SVAL(400), TIME(400), SVAL1(400)                 MUL01270
      EXTERNAL DERIVS                                                 MUL01280
                                                                      MUL01290
      PRINT*,'Welcome to the Electronic Oscillator Model'             MUL01300
      PRINT*,'    '                                                   MUL01310
      PRINT*,'Would You Like To Perform a Simulation ? (Y/N) '        MUL01320
      READ*, CHOICE                                                   MUL01330
      IF (CHOICE .EQ. 'Y') THEN                                       MUL01340
         CALL INIT                                                    MUL01350
                                                                      MUL01360
                                                                      MUL01370
C                              Initialization from Menu               MUL01380
C                                                                     MUL01390
C                                                                     MUL01400
C                                    Initialize Parameters            MUL01410
                                                                      MUL01420
199      R1 = PU(1)                                                   MUL01430
         R2 = PU(2)                                                   MUL01440
         R3 = PU(3)                                                   MUL01450
         R4 = PU(4)                                                   MUL01460
         R5 = PU(5)                                                   MUL01470
         R6 = PU(6)                                                   MUL01480
         C1 = PU(7)                                                   MUL01490
         C2 = PU(8)                                                   MUL01500
         C3 = PU(9)                                                   MUL01510
         C4 = PU(10)                                                  MUL01520
         C5 = PU(11)                                                  MUL01530
         C6 = PU(12)                                                  MUL01540
C                              Initialize State Variables             MUL01550
C                                                                     MUL01560
         S(1)  = PU(13)                                               MUL01570
         S(2)  = PU(14)                                               MUL01580
         S(3)  = PU(15)                                               MUL01590
         S(4)  = PU(16)                                               MUL01600
         S(5)  = PU(17)                                               MUL01610
         S(6)  = PU(18)                                               MUL01620
         F(1)  = PU(19)                                               MUL01630
         F(2)  = PU(20)                                               MUL01640
         F(3)  = PU(21)                                               MUL01650
         F(4)  = PU(22)                                               MUL01660
C                              Initialize Time                        MUL01670
         T     = PU(23)                                               MUL01680
         TFIN  = PU(24)                                               MUL01690
         TOLERF= PU(25)                                               MUL01700
         TOLERS= PU(26)                                               MUL01710
         HSMIN = PU(27)                                               MUL01720
         HFMIN = PU(28)                                               MUL01730
         COMINT= PU(29)                                               MUL01740
         ORD   = 5                                                    MUL01750
C                              Add Dimensions of Fast System          MUL01760
C                              and Slow System                        MUL01770
         NF = 4                                                       MUL01780
```

```
      NS = 6                                                          MUL01790
C                                 Add Graphics Counter                MUL01800
      NPL = 1                                                         MUL01810
                                                                      MUL01820
C                                 Results For Slow Routine at         MUL01830
C                                 the end of the simulation.          MUL01840
                                                                      MUL01850
                                                                      MUL01860
      WRITE(8,17)                                                     MUL01870
   17 FORMAT(1X,'Method:',19X,'Multi-Rate Algorithm Based on '        MUL01880
     + /28X,'Runge-Kutta Merson'/10X/1X,'Study Of Electronic          MUL01890
     + Oscillator',14X,'Slow Routine'/10X/1X,'Time',17X,              MUL01900
     + 'Input To Fast Routine'/3X,'T',16X,'X3',14X,'S')               MUL01910
      PRINT*,'    '                                                   MUL01920
      WRITE(8,3)T,S(1),S(6)                                           MUL01930
    3 FORMAT(1X,F5.1,10X,F7.3,10X,F7.3)                               MUL01940
      TIME  (NPL)= T                                                  MUL01950
      SVAL  (NPL)= S(1)                                               MUL01960
      SVAL1 (NPL) = S(6)                                              MUL01970
      CALL  SRATE(NS,NF,S,F,HS,DERIVS,T,TFIN)                         MUL01980
C                                 Post-Run Output:                    MUL01990
C                                       Tabulate Values of State      MUL02000
C                                       Variables and Time at         MUL02010
C                                       Output Points                 MUL02020
      DO 99 I = 2,NPL                                                 MUL02030
      WRITE(8,87)TIME(I),SVAL(I),SVAL1(I)                             MUL02040
   87 FORMAT(1X,F5.1,5X,F8.3,5X,F8.3)                                 MUL02050
      PRINT*,'  '                                                    MUL02060
   99 CONTINUE                                                        MUL02070
C                                 Plot results of outputs from        MUL02080
C                                 the two routines                    MUL02090
      CALL ANYDEV                                                     MUL02100
      CALL GRAPH(TIME,SVAL,NPL)                                       MUL02110
      CALL GRAPH(TIME,SVAL1,NPL)                                      MUL02120
      PRINT*,'Do you wish to carry out another simulation (Y/N) ?'    MUL02130
      READ*,CHOICE                                                    MUL02140
      IF (CHOICE.EQ.'Y') THEN                                         MUL02150
         CALL INIT                                                    MUL02160
         GOTO 199                                                     MUL02170
      ENDIF                                                           MUL02180
   ENDIF                                                              MUL02190
   END                                                                MUL02200
                                                                      MUL02210
                                                                      MUL05370
                                                                      MUL05380
   SUBROUTINE GRAPH(TVAL,SVALU,NPL)                                   MUL05390
C     This Subroutine Provides A Graphical Display of Component       MUL05400
C     Model Master Routine. Output X3 is plotted against Time.        MUL05410
   DIMENSION TVAL(NPL),SVALU(NPL)                                     MUL05420
   MINT = TVAL(1)                                                     MUL05430
   MAXT = TVAL(NPL) + 1                                               MUL05440
   VALMAX = SVALU(NPL)                                                MUL05450
   VALMIN = SVALU(1)                                                  MUL05460
   DO 13 I = 1,NPL                                                    MUL05470
      IF (SVALU(I).GT.VALMAX) THEN                                    MUL05480
         VALMAX = SVALU(I)                                            MUL05490
      ELSE IF(SVALU(I).LT.VALMIN) THEN                                MUL05500
         VALMIN = SVALU(I)                                            MUL05510
      ENDIF                                                           MUL05520
```

```
   13 CONTINUE                                                          MUL05530
      MAXVAL = VALMAX + 1                                               MUL05540
      MINVAL = VALMIN - 1                                               MUL05550
      SPAC1 = (MAXT - MINT)/10.0                                        MUL05560
      SPAC2 = (MAXVAL - MINVAL)/10.0                                    MUL05570
      CALL HWROT('MOVIE')                                               MUL05580
      CALL PAGE(8.5,11.0)                                               MUL05590
      CALL SWISSM                                                       MUL05600
      CALL AREA2D(6.0,8.0)                                              MUL05610
      CALL HEADIN('STUDY OF ELECTRONIC OSCILLATOR MODEL$',100,1.5,2)    MUL05620
      CALL HEADIN('MULTI-RATE INTEGRATION METHOD$',100,-1.15,2)         MUL05630
      CALL XNAME('TIME(SECS)$',100)                                     MUL05640
      CALL YNAME('VALUE OF X3$',100)                                    MUL05650
      CALL GRAF(MINT,SPAC1,MAXT,MINVAL,SPAC2,MAXVAL)                    MUL05660
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                       MUL05670
C  RATIONAL SPLINE INTERPOLATION                                        MUL05680
      CALL RASPLN (1.5)                                                 MUL05690
      CALL THKCRV (.015)                                                MUL05700
      CALL THKRND (0.)                                                  MUL05710
      CALL CURVE(TVAL,SVALU,NPL,1)                                      MUL05720
      CALL ENDPL(0)                                                     MUL05730
       RETURN                                                           MUL05740
      END                                                               MUL05750
                                                                        MUL05760
      SUBROUTINE GRAPH1(TVAL,SVALU,NPL)                                 MUL05770
C        THIS SUBROUTINE PROVIDES A GRAPHICAL DISPLAY OF COMPONENT      MUL05780
C        MODEL MASTER ROUTINE. OUTPUT S IS PLOTTED AGAINST TIME.        MUL05790
                                                                        MUL05800
      DIMENSION TVAL(NPL),SVALU(NPL)                                    MUL05810
      MINT = TVAL(1)                                                    MUL05820
      MAXT = TVAL(NPL) + 1                                              MUL05830
      VALMAX = SVALU(NPL)                                               MUL05840
      VALMIN = SVALU(1)                                                 MUL05850
      DO 13 I = 1,NPL                                                   MUL05860
         IF (SVALU(I).GT.VALMAX) THEN                                   MUL05870
            VALMAX = SVALU(I)                                           MUL05880
         ELSE IF(SVALU(I).LT.VALMIN) THEN                               MUL05890
            VALMIN = SVALU(I)                                           MUL05900
         ENDIF                                                          MUL05910
   13 CONTINUE                                                          MUL05920
      MAXVAL = VALMAX + 1                                               MUL05930
      MINVAL = VALMIN - 1                                               MUL05940
      SPAC1 = (MAXT - MINT)/10.0                                        MUL05950
      SPAC2 = (MAXVAL - MINVAL)/10.0                                    MUL05960
      CALL HWROT('MOVIE')                                               MUL05970
                                                                        MUL05980
      CALL PAGE(8.5,11.0)                                               MUL05990
      CALL AREA2D(6.0,8.0)                                              MUL06000
      CALL SWISSM                                                       MUL06010
      CALL HEADIN('STUDY OF ELECTRONIC OSCILLATOR MODEL$',100,1.5,2)    MUL06020
      CALL HEADIN('MULTI-RATE INTEGRATION METHOD$',100,-1.15,2)         MUL06030
      CALL XNAME('TIME(SECS)$',100)                                     MUL06040
      CALL YNAME('VALUES OF S$',100)                                    MUL06050
      CALL GRAF(MINT,SPAC1,MAXT,MINVAL,SPAC2,MAXVAL)                    MUL06060
C  DRAW CURVE WITH ROUNDED AND THICKENED LINE AND                       MUL06070
C  RATIONAL SPLINE INTERPOLATION                                        MUL06080
      CALL RASPLN (1.5)                                                 MUL06090
      CALL THKCRV (.015)                                                MUL06100
      CALL THKRND (0.)                                                  MUL06110
```

```
      CALL CURVE(TVAL,SVALU,NPL,1)                                          MUL06120
      CALL ENDPL(0)                                                         MUL06130
      RETURN                                                                MUL06140
      END                                                                   MUL06150
                                                                            MUL06160
      SUBROUTINE DERIVS(S,DS,NF,NS,F)                                       MUL06170
      IMPLICIT REAL(A-H,O-Z)                                                MUL06180
      REAL S(NS),DS(NS),F(NF)                                               MUL06190
      COMMON/PARS/C1,C2,C3,C4,C5,C6,R1,R2,R3,R4,R5,R6                       MUL06200
C        Subroutine consists of Slow System Derivatives which are           MUL06210
C        specified by the user for his particular model.                    MUL06220
                                                                            MUL06230
                                                                            MUL06240
                                                                            MUL06250
      DS(1) = F(4)/(R3*C2)-(1/(R3*C2)+1/(R3*C3))*S(1) + S(2)/(R3*C3)        MUL06260
                                                                            MUL06270
      DS(2) = S(1)/(R4*C3)-(1/(R4*C3)+1.0/(R4*C4))*S(2) + S(3)/(R4*C4)      MUL06280
                                                                            MUL06290
      DS(3) = S(2)/(R5*C4)-(1/(R5*C4)+1/(R5*C5))*S(3) + S(4)/(R5*C5)        MUL06300
                                                                            MUL06310
      DS(4) = S(3)/(R6*C5) - (1.0/(R6*C5) + 1.0/(R6*C6))*S(4)               MUL06320
                                                                            MUL06330
      DS(5) = S(4)/C6                                                       MUL06340
                                                                            MUL06350
      DS(6) = 0.02 * ABS(36.0 - 5.29*SQRT(ABS(S(5))))**1.2*                 MUL06360
     +          SIGN(1.0,ABS(36.0-5.29*SQRT(ABS(S(5)))) - 0.02*S(6)         MUL06370
      RETURN                                                                MUL06380
      END                                                                   MUL06390
                                                                            MUL06400
                                                                            MUL06410
      SUBROUTINE DERIVF(F,DF,NF,NS,INTS)                                    MUL06420
      IMPLICIT REAL(A-H,O-Z)                                                MUL06430
      REAL F(NF),DF(NF),INTS(2)                                             MUL06440
      COMMON/PARS/C1,C2,C3,C4,C5,C6,R1,R2,R3,R4,R5,R6                       MUL06450
                                                                            MUL06460
C        Subroutine consists of Fast System Derivatives which are           MUL06470
C        specified by the user for his particular model.                    MUL06480
                                                                            MUL06490
                                                                            MUL06500
      DF(1) = F(2)                                                          MUL06510
                                                                            MUL06520
      DF(2) = -64.0*F(1) + 7.5*(2.0+INTS(1)-(F(1)*F(1)))*F(2)               MUL06530
                                                                            MUL06540
      DF(3) = 2.0*F(1)*F(2)/R1 + (F(4)-F(3))/(R1*C1)                        MUL06550
                                                                            MUL06560
      DF(4)=F(3)/(R2*C1)-F(4)*(1.0/(R2*C1)+1.0/(R2*C2))+INTS(2)/(R1*C1)     MUL06570
                                                                            MUL06580
      RETURN                                                                MUL06590
      END                                                                   MUL06600
                                                                            MUL06610
                                                                            MUL06630
      SUBROUTINE INIT                                                       MUL07050
                                                                            MUL07060
      INTEGER NUM,RC,RLEN                                                   MUL07070
C     REAL RES1                                                             MUL07080
      CHARACTER*8 KEY,MENUN                                                 MUL07090
      CHARACTER*8 FLDN(29)                                                  MUL07100
      CHARACTER*4 CC                                                        MUL07110
                                                                            MUL07120
```

```
      COMMON/DIFF/PU(29)                                                MUL07130
                                                                        MUL07140
C        Xmenu used for Parameters and  Initial Conditions              MUL07150
                                                                        MUL07160
      FLDN(1) = 'R1'                                                    MUL07170
      FLDN(2) = 'R2'                                                    MUL07180
      FLDN(3) = 'R3'                                                    MUL07190
      FLDN(4) = 'R4'                                                    MUL07200
      FLDN(5) = 'R5'                                                    MUL07210
      FLDN(6) = 'R6'                                                    MUL07220
      FLDN(7) = 'C1'                                                    MUL07230
      FLDN(8) = 'C2'                                                    MUL07240
      FLDN(9) = 'C3'                                                    MUL07250
      FLDN(10) ='C4'                                                    MUL07260
      FLDN(11) ='C5'                                                    MUL07270
      FLDN(12) ='C6'                                                    MUL07280
      FLDN(13) ='X3'                                                    MUL07290
      FLDN(14) ='X4'                                                    MUL07300
      FLDN(15) ='X5'                                                    MUL07310
      FLDN(16) ='X6'                                                    MUL07320
      FLDN(17) ='V'                                                     MUL07330
      FLDN(18) ='S'                                                     MUL07340
      FLDN(19) ='Y1'                                                    MUL07350
      FLDN(20) ='Y2'                                                    MUL07360
      FLDN(21) ='X1'                                                    MUL07370
      FLDN(22) ='X2'                                                    MUL07380
      FLDN(23) ='TINIT'                                                 MUL07390
      FLDN(24) ='TFIN'                                                  MUL07400
      FLDN(25) ='TOLERF'                                                MUL07410
      FLDN(26) ='TOLERS'                                                MUL07420
      FLDN(27) ='HSMIN'                                                 MUL07430
      FLDN(28) ='HFMIN'                                                 MUL07440
      FLDN(29) ='COMINT'                                                MUL07450
                                                                        MUL07460
      MENUN = 'OSCIL5'                                                  MUL07470
      CALL MCLSCR(NUM,RC)                                               MUL07480
      CALL MLOAD(NUM,RC,MENUN)                                          MUL07490
      CALL MPFMAP(NUM,RC,1)                                             MUL07500
      CALL MALARM(NUM,RC,1)                                             MUL07510
      CALL MPRINT(NUM,RC,2,PROUT)                                       MUL07520
      CALL MDSPRD(NUM,RC,KEY)                                           MUL07530
C     CALL MEXIT(NUM,RC)                                                MUL07540
      IF (KEY .EQ. 'PF10') GOTO 25                                      MUL07550
      IF (KEY .EQ. 'PF12') STOP                                         MUL07560
   25 DO 23 I = 1,22                                                    MUL07570
         CALL MSCR2D(NUM,RC,FLDN(I),CC,4,RLEN)                          MUL07580
         CALL MCVTCF(PU(I),RC,CC,4,2)                                   MUL07590
   23 CONTINUE                                                          MUL07600
      DO 28 I = 23,29                                                   MUL07610
         CALL MSCR2D(NUM,RC,FLDN(I),CC,7,RLEN)                          MUL07620
         CALL MCVTCF(PU(I),RC,CC,7,2)                                   MUL07630
   28 CONTINUE                                                          MUL07640
      RETURN                                                            MUL07650
      END                                                               MUL07660
                                                                        MUL07670
                                                                        MUL07680
      SUBROUTINE OUTPUT(T,S,NS)                                         MUL07690
      IMPLICIT REAL(A-H,O-Z)                                            MUL07700
      REAL S(NS)                                                        MUL07710
```

```
      COMMON/OUT/NPL,SVAL(400), TIME(400), SVAL1(400)                    MUL07720
C        Display values of Slow System Outputs in Tabular Form           MUL07730
      NPL = NPL + 1                                                     MUL07740
      WRITE(6,98)T,S(1),S(6)                                            MUL07750
   98 FORMAT(1X,F5.2,10X,F7.3,10X,F7.3)                                 MUL07760
      SVAL (NPL) = S(1)                                                 MUL07770
      SVAL1(NPL)  = S(6)                                                MUL07780
      TIME(NPL)  = T                                                    MUL07790
      RETURN                                                            MUL07800
      END                                                               MUL07810
```

```
                        INITIALIZATION

                   KINDLY INITIALIZE VARIABLES

     Model Parameters                          Experimentation Variables

                                  Initial States
                                S.System   F.System          Control Variables

C1 = 6.25 mF  R1 = 8.00 kOhms X3(0) = 40.0 Y1(0) = 20.0   TINIT  = 0.00

C2 = 4.25 mF  R2 = 8.00 kOhms X4(0) = 45.0 Y2(0) = 70.0    TFIN  = 20.0

C3 = 1.25 mF  R3 = 8.00 kOhms X5(0) = 50.0 X1(0) = 80.0   TOLERF = 0.01

C4 = 1.25 mF  R4 = 8.00 kOhms X6(0) = 55.0 X2(0) = 85.0   TOLERS = 0.50

C5 = 1.25 mF  R5 = 8.00 kOhms V(0) = 60.0                  HSMIN  = 0.0001

C6 = 1.25 mF  R6 = 8.00 kOhms S(0) = 65.0                  HFMIN  = 0.00001
                                                          HBOUND  = 0.10
                                                           COMINT = 0.20

        Press:  PF10 - to start simulation    PF12 - to quit
```

Menu For Electronic Oscillator
Model: Multi-Rate Integration

Problem: Study Of Electronic Oscillator

Method: Multi-Rate Algorithm Based on
Runge-Kutta Merson

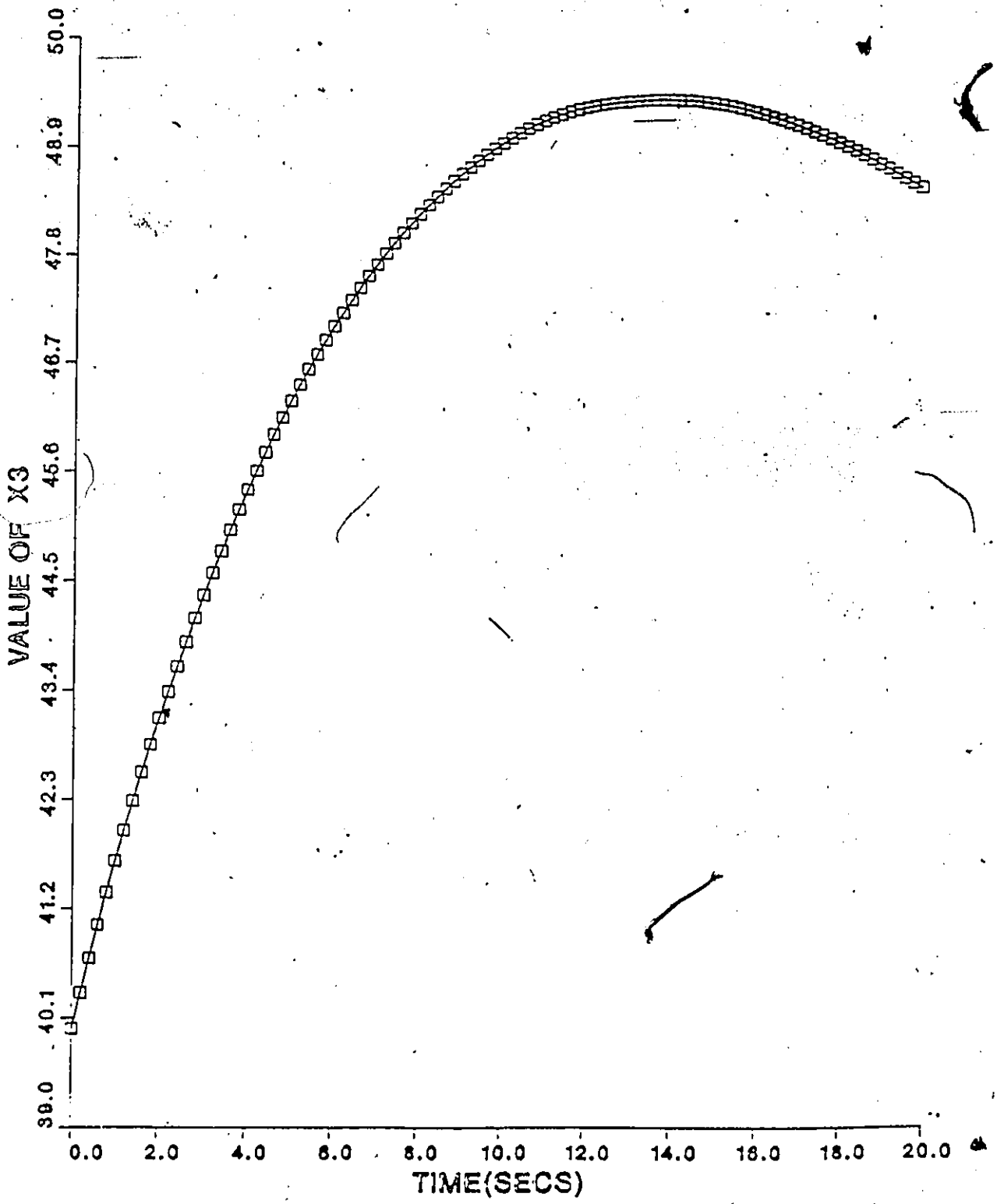
Time T	Inputs To Fast Routine X3	S
0.0	40.000	65.000
0.2	40.469	64.713
0.4	40.707	64.576
0.6	41.178	64.327
0.8	41.371	64.234
1.0	41.688	64.091
1.2	41.995	63.966
1.4	42.293	63.859
1.6	42.582	63.769
1.8	42.862	63.695
2.0	43.222	63.621
2.2	43.398	63.594
2.4	43.653	63.566
2.6	43.901	63.551
2.8	44.142	63.551
3.0	44.448	63.570
3.2	44.601	63.589
3.4	44.819	63.626
3.6	45.032	63.676
3.8	45.237	63.737
4.0	45.436	63.810
4.2	45.692	63.924
4.4	45.815	63.988
4.6	46.069	64.139
4.8	46.170	64.207
5.0	46.338	64.332
5.2	46.501	64.466
5.4	46.715	64.663
5.6	46.811	64.761
5.8	47.019	64.995
6.0	47.099	65.092
6.2	47.238	65.270
6.4	47.372	65.457
6.6	47.502	65.651
6.8	47.668	65.919
7.0	47.751	66.063
7.2	47.870	66.280
7.4	47.985	66.504
7.6	48.136	66.819
7.8	48.205	66.974
8.0	48.353	67.326
8.2	48.411	67.471
8.4	48.508	67.729
8.6	48.627	68.066
8.8	48.694	68.264
9.0	48.814	68.649
9.2	48.866	68.824
9.4	48.947	69.112
9.6	49.025	69.407
9.8	49.126	69.816
10.0	49.172	70.012
10.3	49.270	70.457
10.4	49.307	70.638
10.6	49.398	71.110
10.8	49.430	71.285
11.0	49.488	71.616
11.2	49.542	71.951
11.5	49.609	72.392
11.6	49.644	72.637
11.8	49.690	72.986
12.0	49.747	73.448
12.2	49.776	73.699
12.4	49.815	74.062
12.7	49.862	74.539
12.8	49.885	74.800
13.0	49.929	75.326
13.2	49.946	75.554

Problem: Study Of Electronic Oscillator

Method: Multi-Rate Algorithm Based on Runge-Kutta Merson

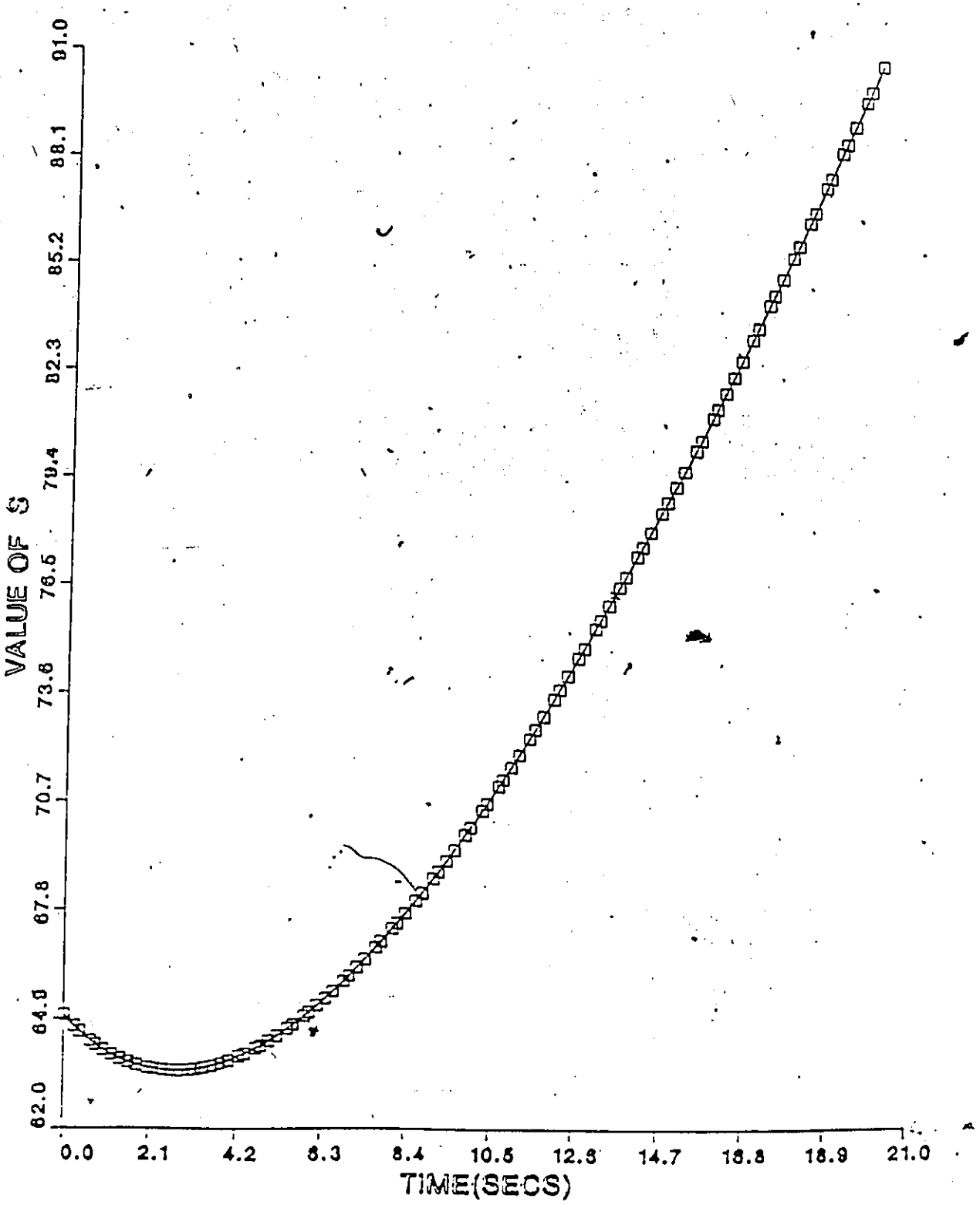
Time T	Inputs To Fast Routine X3	S
13.4	49.973	75.937
13.6	50.004	76.431
13.8	50.020	76.715
14.0	50.048	77.263
14.2	50.059	77.507
14.4	50.075	77.909
14.6	50.092	78.427
14.8	50.101	78.723
15.0	50.110	79.134
15.2	50.118	79.550
15.5	50.126	80.114
15.6	50.128	80.390
15.9	50.131	80.996
16.0	50.131	81.243
16.2	50.129	81.674
16.4	50.126	82.108
16.6	50.121	82.545
16.9	50.111	83.125
17.0	50.105	83.427
17.2	50.089	84.056
17.4	50.082	84.321
17.6	50.068	84.772
17.9	50.047	85.350
18.0	50.035	85.682
18.2	50.008	86.319
18.4	49.995	86.602
18.6	49.963	87.272
18.8	49.950	87.532
19.1	49.912	88.222
19.2	49.898	88.472
19.4	49.870	88.945
19.6	49.830	89.604
19.8	49.811	89.898
20.0	49.764	90.591

ELECTRONIC OSCILLATOR MODEL
MULTI-RATE INTEGRATION METHOD



STUDY OF ELECTRONIC OSCILLATOR MODEL

MULTI-RATE INTEGRATION METHOD



Appendix H. INTEGRATION ROUTINES FOR THE GEST RUN TIME LIBRARY

H.1 The Fourth Order Runge Kutta Algorithm

```
      SUBROUTINE RK4(T,Y,N,TFIN,H,HINIT,HBOUND,HMIN,TOL,COMINT,DERIV)    DIA01830
C         Fourth - Order Runge Kutta Method Used To                      DIA01840
C         Integrate Systems Having Frequent                              DIA01850
C         Output Points But Not Requiring A                              DIA01860
C         High Level Of Accuracy.                                        DIA01870
C                                                                        DIA01880
C                                  Variable Declaration                  DIA01890
C                                                                        DIA01900
      IMPLICIT REAL (A-H,O-Z)                                            DIA01910
      DIMENSION Y(N),DY1(15),DY2(15),DY3(15),DY4(15),F1(15),F2(15),      DIA01920
     +F3(15),DY(15)                                                      DIA01930
      EXTERNAL DERIV                                                     DIA01940
      CHARACTER*1 CHOICE                                                 DIA01950
C                                  Calculating Starting Step Size        DIA01960
      CALL DERIV(T,Y,DY,N)                                               DIA01970
      FDER = 0.0                                                         DIA01980
      DO 29 I = 1,N                                                      DIA01990
         IF (ABS(DY(I)).GT.FDER) THEN                                    DIA02000
            FDER = ABS(DY(I))                                            DIA02010
         ENDIF                                                           DIA02020
   29 CONTINUE                                                           DIA02030
      IF (FDER.NE.0.0) THEN                                              DIA02040
         HPOS = SQRT(0.5 * TOL/FDER)                                     DIA02050
         IF (HBOUND.LE.0.0) THEN                                         DIA02060
            HBOUND = 0.1                                                 DIA02070
         ENDIF                                                           DIA02080
         IF (HPOS.GT.HBOUND) THEN                                        DIA02090
            HSTART = HBOUND                                              DIA02100
         ELSE                                                            DIA02110
            HSTART = HPOS                                                DIA02120
         ENDIF                                                           DIA02130
      ELSE                                                               DIA02140
         IF (HBOUND.LE.0.0) THEN                                         DIA02150
            HBOUND = 0.1                                                 DIA02160
         ENDIF                                                           DIA02170
         HSTART = HBOUND                                                 DIA02180
      ENDIF                                                              DIA02190
      IF (HINIT.NE.0.0) THEN                                             DIA02200
         H = HINIT                                                       DIA02210
      ELSE                                                               DIA02220
         HINIT = HSTART                                                  DIA02230
         H     = HINIT                                                   DIA02240
      ENDIF                                                              DIA02250
                                                                         DIA02260
      IF (H.GT.HBOUND) THEN                                              DIA02270
         PRINT*,'HINIT > HBOUND. DO YOU WISH TO CARRY OUT INTEGRATION    DIA02280
     + WITH THIS HINIT (Y) OR HBOUND (N)?'                               DIA02290
         READ*, CHOICE                                                   DIA02300
         IF (CHOICE.EQ.'Y') THEN                                         DIA02310
            H = HINIT                                                    DIA02320
         ELSE                                                            DIA02330
            H     = HBOUND                                               DIA02340
            HINIT = HBOUND                                               DIA02350
         ENDIF                                                           DIA02360
      ENDIF                                                              DIA02370
```

```
      IF (COMINT.LE.0) THEN                                              DIA02380
         COMINT = (TFIN - T)/100                                         DIA02390
      ENDIF                                                              DIA02400
      IF (HMIN.LE.0.0) THEN                                              DIA02410
         HMIN = HINIT/10000                                              DIA02420
      ENDIF                                                              DIA02430
      NFLAG  = 0                                                         DIA02440
      TCOM  = T + COMINT                                                 DIA02450
C                                  Integrate system                      DIA02460
   99 IF (T.LT.TFIN) THEN                                                DIA02470
   88    IF (T.LT.TCOM) THEN                                             DIA02480
            IF (((TCOM - T).LE.(HINIT+ HMIN)).AND.                       DIA02490
     +          (NFLAG.EQ. 0)) THEN                                      DIA02500
               H = (TCOM - T)/2.0                                        DIA02510
               NFLAG= 1                                                  DIA02520
            ENDIF                                                        DIA02530
            H1 = H * 0.5                                                 DIA02540
            CALL DERIV(T,Y,DY1,N)                                        DIA02550
            DO 6 I = 1,N                                                 DIA02560
               F1(I) = DY1(I) * H/2.0 + Y(I)                             DIA02570
    6       CONTINUE                                                     DIA02580
            CALL DERIV(H1 + T,F1,DY2,N)                                  DIA02590
            DO 8 I = 1,N                                                 DIA02600
               F2(I) = H * DY2(I)/2.0 + Y(I)                             DIA02610
    8       CONTINUE                                                     DIA02620
            CALL DERIV(H1 + T,F2,DY3,N)                                  DIA02630
            DO 10 I = 1,N                                                DIA02640
               F3(I) = H*DY3(I) + Y(I)                                   DIA02650
   10       CONTINUE                                                     DIA02660
            CALL DERIV(H + T,F3,DY4,N)                                   DIA02670
            DO 16 I =1,N                                                 DIA02680
               Y(I)=Y(I)+(DY1(I)+2.0*DY2(I)+2.0*DY3(I)+DY4(I))*H/6.0     DIA02690
   16       CONTINUE                                                     DIA02700
            T = T +H                                                     DIA02710
C                                  Output Values Of Y and T at           DIA02720
C                                  each point of TOUT                    DIA02730
         ELSE                                                            DIA02740
            CALL OUTPUT(Y,N,T)                                           DIA02750
            TCOM = TCOM + COMINT                                         DIA02760
            IF (TCOM.GE.TFIN) THEN                                       DIA02770
               TCOM = TFIN                                               DIA02780
            ENDIF                                                        DIA02790
            NFLAG = 0                                                    DIA02800
            H     = HINIT                                                DIA02810
            GOTO 88                                                      DIA02820
         ENDIF                                                           DIA02830
         GOTO 99                                                         DIA02840
      ENDIF                                                              DIA02850
      CALL OUTPUT(Y,N,T)                                                 DIA02860
      RETURN                                                             DIA02870
      END                                                                DIA02880
```

H.2 The Runge Kutta Fehlberg Algorithm

```
      SUBROUTINE RKFEL(T,Y,N,TFIN,H,ORD,HBOUND,HMIN,COMINT,DERIV,TOL)   FEL01680
C         Sixth Order Variable Step R-K Felberg                        FEL01690
C         used to integrate systems with                               FEL01700
C         infrequent output to a high level of                         FEL01710
C         accuracy.                                                    FEL01720
C                                                                      FEL01730
C                                  Variable Declaration                FEL01740
      IMPLICIT REAL*4 (A-H,O-Z)                                        FEL01750
      DIMENSION Y(10),DY(10),DY1(10),DY2(10),DY3(10),DY4(10),DY5(10),  FEL01760
     +F(10),F1(10),F2(10),F3(10),F4(10),ERR(10)                        FEL01770
      CHARACTER*1 CHOICE                                               FEL01780
      REAL NUVAL                                                       FEL01790
      IF (COMINT.EQ.0) COMINT = (TFIN-T)/70                            FEL01800
      NFLAG= 0                                                         FEL01810
      TCOM = T + COMINT                                                FEL01820
C                                Calculate Starting Step Size          FEL01830
      CALL DERIV(T,Y,DY,N)                                             FEL01840
      FDER  = 0.0                                                      FEL01850
      DO 29 I = 1,N                                                    FEL01860
         IF (ABS(DY(I)).GT.FDER) THEN                                  FEL01870
           FDER = ABS(DY(I))                                           FEL01880
         ENDIF                                                         FEL01890
   29 CONTINUE                                                         FEL01900
      IF (FDER.NE.0.0) THEN                                            FEL01910
         HPOS  = SQRT(0.5 * TOL/FDER)                                  FEL01920
         IF (HBOUND.LE.0.0) THEN                                       FEL01930
            HBOUND = 0.1                                               FEL01940
         ENDIF                                                         FEL01950
         IF (HPOS.GT.HBOUND) THEN                                      FEL01960
            HSTART = HBOUND                                            FEL01970
         ELSE                                                          FEL01980
            HSTART = HPOS                                              FEL01990
         ENDIF                                                         FEL02000
      ELSE                                                             FEL02010
         IF (HBOUND.LE.0.0) THEN                                       FEL02020
            HBOUND = 0.1                                               FEL02030
         ENDIF                                                         FEL02040
         HSTART = HBOUND                                               FEL02050
      ENDIF                                                            FEL02060
      IF (HINIT.NE.0.0) THEN                                           FEL02070
         H = HINIT                                                     FEL02080
      ELSE                                                             FEL02090
         HINIT = HSTART                                                FEL02100
         H     = HINIT                                                 FEL02110
      ENDIF                                                            FEL02120
                                                                       FEL02130
      IF (H.GT.HBOUND) THEN                                            FEL02140
         PRINT*,'HINIT > HBOUND. DO YOU WISH TO CARRY OUT INTEGRATION  FEL02150
     + WITH THIS HINIT (Y) OR HBOUND (N) ?'                            FEL02160
         READ*, CHOICE                                                 FEL02170
         IF (CHOICE.EQ.'Y') THEN                                       FEL02180
            H = HINIT                                                  FEL02190
         ELSE                                                          FEL02200
            H     = HBOUND                                             FEL02210
            HINIT = HBOUND                                             FEL02220
         ENDIF                                                         FEL02230
```

```
      ENDIF                                                              FEL02240
      IF (COMINT.LE.0.0) THEN                                            FEL02250
         COMINT = (TFIN - T)/100                                         FEL02260
      ENDIF                                                              FEL02270
      IF (HMIN.LE.0.0) THEN                                              FEL02280
         HMIN = HINIT /1000                                              FEL02290
      ENDIF                                                              FEL02300
                                                                         FEL02310
                                                                         FEL02320
                                                                         FEL02330
C                              Integrate system                          FEL02340
   88 IF (T.LT.TFIN) THEN                                                FEL02350
                                                                         FEL02360
  188    IF (T.LT.TCOM) THEN                                             FEL02370
            IF (((TCOM - T).LE.(H + HMIN)) .AND. (NFLAG.EQ.0)) THEN      FEL02380
               H = (TCOM - T)/2                                          FEL02390
               NFLAG = 1                                                 FEL02400
            ENDIF                                                        FEL02410
            CALL DERIV(T,Y,DY,N)                                         FEL02420
            DO 6 I = 1,N                                                 FEL02430
               F(I) = DY(I) * H/4.0 + Y(I)                               FEL02440
    6       CONTINUE                                                     FEL02450
            CALL DERIV(H/4.0 + T,F,DY1,N)                                FEL02460
            DO 8 I = 1,N                                                 FEL02470
               F1(I) = (3.0*DY(I) + 9.0 * DY1(I))*H/32.0 + Y(I)          FEL02480
    8       CONTINUE                                                     FEL02490
            CALL DERIV(3.0*H/8.0 + T,F1,DY2,N)                           FEL02500
            DO 10 I = 1,N                                                FEL02510
               F2(I) = (1932.0*DY(I) + 7296.0*DY2(I) - 7200.0*DY1(I))    FEL02520
     +         *H/2197.0 + Y(I)                                          FEL02530
   10        CONTINUE                                                    FEL02540
             CALL DERIV(12*H/13.0 + T,F2,DY3,N)                          FEL02550
             DO 12 I = 1,N                                               FEL02560
                F3(I)=H*(439.0*DY(I)/216.0 - 8.0*DY1(I) + 3680.0*DY2(I)  FEL02570
     +          /513.0 - 845.0*DY3(I)/4104.0) + Y(I)                     FEL02580
   12        CONTINUE                                                    FEL02590
             CALL DERIV(T + H,F3,DY4,N)                                  FEL02600
             DO 14 I  = 1,N                                              FEL02610
                F4(I)=H*(-8.0*DY(I)/27.0 + 2.0*DY1(I)-3544.0 * DY2(I)/   FEL02620
     +          2565.0 + 1859.0*DY3(I)/4104.0-11.0*DY4(I)/40.0) + Y(I)   FEL02630
   14        CONTINUE                                                    FEL02640
             CALL DERIV(T+H/2.0,F4,DY5,N)                                FEL02650
             ERRMAX = 0.0                                                FEL02660
             DO 40 I = 1,N                                               FEL02670
                ERR(I)=ABS(DY(I)/360.0 - 128.0*DY2(I)/4275.0 - 2197.0    FEL02680
     +          *DY3(I)/75240.0 + DY4(I)/50.0 + 2.0*DY5(I)/55.0)         FEL02690
                IF (ERR(I).GT.ERRMAX) THEN                               FEL02700
                   ERRMAX = ERR(I)                                       FEL02710
                ENDIF                                                    FEL02720
   40        CONTINUE                                                    FEL02730
             IF (ERRMAX.LE.TOL) THEN                                     FEL02740
                T = T + H                                                FEL02750
                DO 16 I =1,N                                             FEL02760
                   Y(I) = Y(I) + H*(25.0*DY(I)/216.0 + 1408.0*DY2(I)/    FEL02770
     +             2565.0 + 2197.0 * DY3(I)/4104.0 - DY4(I)/5.0)         FEL02780
   16           CONTINUE                                                 FEL02790
             ENDIF                                                       FEL02800
C                                                                        FEL02810
             IF (NFLAG.EQ.0) THEN                                        FEL02820
```

```
            H = 0.9 * H * (TOL/ERRMAX) ** (1/ORD)                          FEL02830
            IF (H.LT.HMIN) THEN                                            FEL02840
               PRINT*,'HMIN Exceeded. Do you wish to change value of       FEL02850
     + HMIN (Y) or would you rather halt the integration and further       FEL02860
     +investigate the model (N) ?'                                         FEL02870
               READ*,CHOICE                                                FEL02880
               IF (CHOICE.EQ.'Y') THEN                                     FEL02890
                  PRINT*,'Please enter new value of HMIN. Note that        FEL02900
     +for integration to proceed,it should be less than',H                 FEL02910
                  READ*, NUVAL                                             FEL02920
                  HMIN = NUVAL                                             FEL02930
               ELSE                                                        FEL02940
                  STOP                                                     FEL02950
               ENDIF                                                       FEL02960
            ENDIF                                                          FEL02970
C           IF (H.GT.HMAX) THEN                                            FEL02980
C              H = HMAX                                                    FEL02990
C           ENDIF                                                          FEL03000
         ENDIF                                                             FEL03010
      ELSE                                                                 FEL03020
         CALL OUTPUT(Y,N,T)                                                FEL03030
         TCOM = TCOM + COMINT                                              FEL03040
         IF ((TCOM.GE.TFIN) .AND. (T.NE.TFIN)) THEN                        FEL03050
            TCOM = TFIN                                                    FEL03060
         ENDIF                                                             FEL03070
         NFLAG = 0                                                         FEL03080
         GOTO 88                                                           FEL03090
      ENDIF                                                                FEL03100
      GOTO 188                                                             FEL03110
   ENDIF                                                                   FEL03120
   CALL OUTPUT(Y,N,T)                                                      FEL03130
   RETURN                                                                  FEL03140
   END                                                                     FEL03150
```

H.3 The Fourth Order Adams Algorithm

```
      SUBROUTINE ADAMS(T,Y,N,TFIN,H,NORD,HBOUND,HMIN,DERIV,COMINT,TOL,K)ADA01860
C         FourthOrder Variable Step Adams Predictor Corrector Method    ADA01870
C         used to integrate systems to a high level of accuracy.        ADA01880
                                                                        ADA01890
C                                    Variable Declaration               ADA01900
      IMPLICIT REAL*4(A-H,O-Z)                                          ADA01910
      DIMENSION T(6000),Y(6000,5),Y1(5),Y2(5),Y3(5),                    ADA01920
     +      Y4(5),VC(6000,5),TIME(300),YVAL1(300),YVAL2(300),          ADA01930
     +     DY(6000,5)                                                 ADA01940
                                                                        ADA01950
      EXTERNAL DERIV                                                    ADA01960
      COMMON/STIFF/NCHECK                                               ADA01970
      CHARACTER*1 CHOICE                                                ADA01980
      REAL NUVAL                                                        ADA01990
                                                                        ADA02000
      K = 1                                                             ADA02010
C                                  CAECULATING STARTING STEP SIZE       ADA02020
      CALL DERIV(T,Y,DY,N,K)                                            ADA02030
      FDER = 0.0                                                        ADA02040
      DO 29 I = 1,N                                                     ADA02050
         IF (ABS(DY(1,I)).GT.FDER) THEN                                 ADA02060
            FDER = ABS(DY(1,I))                                         ADA02070
         ENDIF                                                          ADA02080
   29 CONTINUE                                                          ADA02090
      IF (FDER.NE.0.0) THEN                                             ADA02100
         HPOS = SQRT(0.5 * TOL/FDER)                                    ADA02110
         IF (HBOUND.LE.0.0) THEN                                        ADA02120
            HBOUND = 0.1                                                ADA02130
         ENDIF                                                          ADA02140
         IF (HPOS:GT.HBOUND) THEN                                       ADA02150
            HSTART = HBOUND                                             ADA02160
         ELSE                                                           ADA02170
            HSTART = HPOS                                               ADA02180
         ENDIF                                                          ADA02190
      ELSE                                                              ADA02200
         IF (HBOUND.LE.0.0) THEN                                        ADA02210
            HBOUND = 0.1                                                ADA02220
         ENDIF                                                          ADA02230
         HSTART = HBOUND                                                ADA02240
      ENDIF                                                             ADA02250
      IF (HINIT.NE.0.0) THEN                                            ADA02260
         H = HINIT                                                      ADA02270
      ELSE                                                              ADA02280
         HINIT = HSTART                                                 ADA02290
         H     = HINIT                                                  ADA02300
      ENDIF                                                             ADA02310
                                                                        ADA02320
      IF (H.GT.HBOUND) THEN                                             ADA02330
         PRINT*,'HINIT > HBOUND. DO YOU WISH TO CARRY OUT INTEGRATION   ADA02340
     + WITH THIS HINIT (Y) OR HBOUND (N)?'                              ADA02350
         READ*, CHOICE                                                  ADA02360
         IF (CHOICE.EQ.'Y') THEN                                        ADA02370
            H = HINIT                                                   ADA02380
         ELSE                                                           ADA02390
            H     = HBOUND                                              ADA02400
            HINIT = HBOUND                                              ADA02410
```

```
      ENDIF                                                              ADA02420
   ENDIF                                                                 ADA02430
   IF (COMINT.LE.0) THEN                                                 ADA02440
      COMINT = (TFIN - T(1))/30                                          ADA02450
   ENDIF                                                                 ADA02460
   IF (HMIN.LE.0.0) THEN                                                 ADA02470
      HMIN = HINIT/10000                                                 ADA02480
   ENDIF                                                                 ADA02490
   TCOM  = TCOM + COMINT                                                 ADA02500
   T1 = T(1)                                                             ADA02510
C                                Compute steps for stiffness check       ADA02520
   NSTEPS = (TFIN-T1)/HINIT * 10                                         ADA02530
C                                                                        ADA02540
C                                Compute the first three                 ADA02550
C                                values by Fourth-Order                  ADA02560
C                                Runge Kutta                             ADA02570
   CALL RK4(H,Y,K,T1,DERIV,Y2,T2,Y3,T3,Y4,T4,N)                          ADA02580
   NFLG = 1                                                              ADA02590
   MFLG = 1                                                              ADA02600
   NFLAG= 0                                                              ADA02610
   K  = 5                                                                ADA02620
   DO 62 I = 1,N                                                         ADA02630
      Y(K-1,I) = Y4(I)                                                   ADA02640
      Y(K-2,I) = Y3(I)                                                   ADA02650
      Y(K-3,I) = Y2(I)                                                   ADA02660
62 CONTINUE                                                              ADA02670
   T(K-1) = T4                                                           ADA02680
   T(K-2) = T3                                                           ADA02690
   T(K-3) = T2                                                           ADA02700
   TNOW   = T(K-1) + H                                                   ADA02710
99 IF (T(K-1).LE.TFIN) THEN                                              ADA02720
88 IF ((T(K-1).LE.TCOM) .OR. (MFLG.EQ.1)) THEN                           ADA02730
      IF (((TCOM - T(K-1)).LE.(H+ HMIN)).AND.(NFLAG.EQ.0)) THEN          ADA02740
         H = (TCOM - T(K-1))/2.0                                         ADA02750
         NFLAG= 1                                                        ADA02760
      ENDIF                                                              ADA02770
      CALL ADCOMP(H,Y,T,N,K,ERRMAX,VC)                                   ADA02780
      NCHECK = NCHECK + 1                                                ADA02790
      IF (ERRMAX.LE.TOL) THEN                                            ADA02800
         DO 101 I = 1,N                                                  ADA02810
            Y(K,I) = VC(K,I)                                             ADA02820
101      CONTINUE                                                        ADA02830
         T(K) = TNOW                                                     ADA02840
         MFLG = 0                                                        ADA02850
         NFLG = 0                                                        ADA02860
         K = K + 1                                                       ADA02870
      ENDIF                                                              ADA02880
      IF (ERRMAX .GT. 0.0) THEN                                          ADA02890
      IF ((ERRMAX.LE.(0.9*TOL)) .AND. (NFLAG.EQ.0)) THEN                 ADA02900
         HOLD = H                                                        ADA02910
         H = 0.9 * H * (TOL/ERRMAX) ** (1/ORD)                           ADA02920
         IF ((T(K-1) + 3*H).GE.TCOM) THEN                                ADA02930
            HOLD = TCOM - T(K-1)                                         ADA02940
            H    = HOLD                                                  ADA02950
         ENDIF                                                           ADA02960
         NFLG = 1                                                        ADA02970
         DO 59 I = 1,N                                                   ADA02980
            Y(K,I) = Y(K-1,I)                                            ADA02990
59       CONTINUE                                                        ADA03000
         T1 = T(K-1)
```

```
            CALL RK4(H,Y,K,T1,DERIV,Y2,T2,Y3,T3,Y4,T4,N)                    ADA03010
            DO 65 I = 1,N                                                   ADA03020
               Y(K+2,I) = Y4(I)                                             ADA03030
               Y(K+1,I) = Y3(I)                                             ADA03040
               Y(K,I)   = Y2(I)                                             ADA03050
 65         CONTINUE                                                        ADA03060
            T(K+2) = T4                                                     ADA03070
            T(K+1) = T3                                                     ADA03080
            T(K)   = T2                                                     ADA03090
            K = K + 3                                                       ADA03100
         ENDIF                                                              ADA03110
         ENDIF                                                              ADA03120
         IF (ERRMAX.GT.TOL) THEN                                            ADA03130
            MFLG = 1                                                        ADA03140
               H = 0.9 * H * (TOL/ERRMAX) ** (1/NORD)                       ADA03150
            IF (H.LT.HMIN) THEN                                             ADA03160
                  PRINT*,'HMIN Exceeded. Do you wish to change value of     ADA03170
     + HMIN (Y) or would you rather halt the integration and further         ADA03180
     +investigate the model (N) ?'                                          ADA03190
                  READ*,CHOICE                                              ADA03200
                  IF (CHOICE.EQ.'Y') THEN                                   ADA03210
                     PRINT*,'Please enter new value of HMIN. Note that      ADA03220
     +for integration to proceed,it should be less than',H                  ADA03230
                     READ*, NUVAL                                           ADA03240
                     HMIN = NUVAL                                           ADA03250
                  ELSE                                                      ADA03260
                     STOP                                                   ADA03270
                  ENDIF                                                     ADA03280
            ENDIF                                                           ADA03290
            IF (NFLG.EQ.1) K = K-3                                          ADA03300
            DO 150 I = 1,N                                                  ADA03310
               Y(K,I)=Y(K-1,I)                                              ADA03320
 150        CONTINUE                                                        ADA03330
            T1 = T(K-1)                                                     ADA03340
            CALL RK4(H,Y,K,T1,DERIV,Y2,T2,Y3,T3,Y4,T4,N)                    ADA03350
            DO 67 I = 1,N                                                   ADA03360
               Y(K+2,I) = Y4(I)                                             ADA03370
               Y(K+1,I) = Y3(I)                                             ADA03380
               Y(K,I)   = Y2(I)                                             ADA03390
 67         CONTINUE                                                        ADA03400
            T(K+2) = T4                                                     ADA03410
            T(K+1) = T3                                                     ADA03420
            T(K)   = T2                                                     ADA03430
            K = K + 3                                                       ADA03440
            NFLG = 1                                                        ADA03450
         ENDIF                                                              ADA03460
         TNOW = T(K-1) + H                                                  ADA03470
         T(K) = T(K-1) + H                                                  ADA03480
C        PRINT*,'TNOW = ',TNOW                                              ADA03490
C        PRINT*,'Y(K,1) = ',Y(K,1)                                          ADA03500
C        PRINT*,'Y(K,2) = ',Y(K,2)                                          ADA03510
C        PRINT*,'H = ',H                                                    ADA03520
                                                                            ADA03530
      ELSE                                                                  ADA03540
         CALL OUTPUT(Y,N,TNOW,K)                                            ADA03550
         TCOM = TCOM + COMINT                                               ADA03560
         IF ((TCOM.GE.TFIN) .AND. (TNOW.NE.TFIN)) THEN                      ADA03570
            TCOM = TFIN                                                     ADA03580
      ENDIF                                                                 ADA03590
```

```
      NFLAG = 0                                                          ADA03600
      GOTO 88                                                            ADA03610
    ENDIF                                                                ADA03620
    GOTO 99                                                              ADA03630
    ENDIF                                                                ADA03640
    CALL OUTPUT(Y,N,TNOW,K)                                              ADA03650
    IF (NCHECK.GT.NSTEPS) THEN                                           ADA03660
      PRINT*,'Test suspects Model to be stiff with time constants        ADA03670
     +varying greatly'                                                   ADA03680
    ENDIF                                                                ADA03690
    RETURN                                                               ADA03700
    END                                                                  ADA03710
                                                                         ADA03720
                                                                         ADA03730
                                                                         ADA03740
    SUBROUTINE RK4(H,Y,K,T1,DERIV,Y2,T2,Y3,T3,Y4,T4,N)                   ADA03750
                                                                         ADA03760
C     The Fourth-Order Runge-Kutta Method                                ADA03770
C     is used as a starter to determine                                  ADA03780
C     the four previous function evaluations                             ADA03790
C     needed by the Adams Method. It is                                  ADA03800
C     invoked at the beginning of the simul-                             ADA03810
C     ation and at each step size change.                                ADA03820
                                                                         ADA03830
C                              Variable Declaration                      ADA03840
                                                                         ADA03850
    IMPLICIT REAL*4 (A-H,O-Z)                                            ADA03860
    DIMENSION Y(6000,5),DY1(6000,5),DY2(6000,5),DY3(6000,5),             ADA03870
   + DY4(6000,5),P1(6000,5),P2(6000,5),P3(6000,5),Y2(5),Y3(5),Y4(5)      ADA03880
    COMMON/STIFF/NCHECK                                                  ADA03890
    EXTERNAL DERIV                                                       ADA03900
                                                                         ADA03910
    H1 = H * 0.5                                                         ADA03920
    KC = 1                                                               ADA03930
    DO 5 KC = 1,3                                                        ADA03940
      CALL DERIV(T1,Y,DY1,N,K)                                           ADA03950
      DO 6 I = 1,N                                                       ADA03960
        P1(K,I) =(DY1(K,I)*H/2.0)    + Y(K,I)                          ADA03970
  6   CONTINUE                                                           ADA03980
      CALL DERIV(H1 + T1,P1,DY2,N,K)                                     ADA03990
      DO 8 I = 1,N                                                       ADA04000
        P2(K,I) = H * DY2(K,I)/2.0 + Y(K,I)                              ADA04010
  8   CONTINUE                                                           ADA04020
      CALL DERIV(H1 + T1,P2,DY3,N,K)                                     ADA04030
      DO 10 I = 1,N                                                      ADA04040
        P3(K,I) = H*DY3(K,I) + Y(K,I)                                    ADA04050
 10   CONTINUE                                                           ADA04060
      CALL DERIV(H + T1,P3,DY4,N,K)                                      ADA04070
      DO 16 I =1,N                                                       ADA04080
        Y(K,I) = Y(K,I)+(DY1(K,I)+2.0*DY2(K,I)                           ADA04090
     +    + 2.0*DY3(K,I) + DY4(K,I)) *H/6.0                             ADA04100
 16   CONTINUE                                                           ADA04110
      IF (KC.EQ.1) THEN                                                  ADA04120
        DO 11 I = 1,N                                                    ADA04130
          Y2(I) = Y(K,I)                                                 ADA04140
 11     CONTINUE                                                         ADA04150
        T2 = T1 + H                                                      ADA04160
      ELSEIF (KC.EQ.2) THEN                                              ADA04170
        DO 12 I = 1,N                                                    ADA04180
```

```
            Y3(I) = Y(K,I)                                                    ADA04190
 12         CONTINUE                                                          ADA04200
            T3 = T1 + H                                                       ADA04210
          ELSEIF (KC.EQ.3) THEN                                               ADA04220
            DO 13 I = 1,N                                                     ADA04230
              Y4(I) = Y(K,I)                                                  ADA04240
 13         CONTINUE                                                          ADA04250
            T4 = T1                                                           ADA04260
          ENDIF                                                               ADA04270
          T1 = T1 + (KC) * H                                                  ADA04280
 5      CONTINUE                                                             ADA04290
        NCHECK = NCHECK + 3                                                   ADA04300
        RETURN                                                                ADA04310
        END                                                                   ADA04320
                                                                              ADA04330
                                                                              ADA04570
        SUBROUTINE ADCOMP(H,Y,T,N,K,ERRMAX,VC)                                ADA04580
        IMPLICIT REAL*4 (A-H,O-Z)                                             ADA04590
        DIMENSION Y(6000,5),T(6000),VP(6000,5),VC(6000,5),ERR(5),             ADA04600
       +   DY(6000,5),DVP(6000,5)                                             ADA04610
                                                                              ADA04620
C          This subroutine performs the calculations                          ADA04630
C          of the predictor and corrector for the                             ADA04640
C          Adams Method.                                                      ADA04650
                                                                              ADA04660
        CALL DERIV(T,Y,DY,N,K-1)                                              ADA04670
        CALL DERIV(T,Y,DY,N,K-2)                                              ADA04680
        CALL DERIV(T,Y,DY,N,K-3)                                              ADA04690
        CALL DERIV(T,Y,DY,N,K-4)                                              ADA04700
                                                                              ADA04710
        DO 50 I = 1,N                                                         ADA04720
           VP(K,I) = Y(K-1,I) + H/24.0 *(55.0*DY(K-1,I) - 59.0*DY(K-2,I)      ADA04730
     +       +37.0*DY(K-3,I)- 9.0*DY(K-4,I))                                  ADA04740
   50 CONTINUE                                                                ADA04750
        CALL DERIV(T,VP,DVP,N,K)                                              ADA04760
        DO 51 I = 1,N                                                         ADA04770
           VC(K,I) = Y(K-1,I) + H/24.0*(9.0*DVP(K,I) + 19.0                   ADA04780
     +    * DY(K-1,I) - 5.0*DY(K-2,I) + DY(K-3,I))                           ADA04790
   51 CONTINUE                                                                ADA04800
        ERRMAX= 0.0                                                           ADA04810
        DO 100 I = 1,N                                                        ADA04820
           ERR(I) = ABS(VC(K,I)-VP(K,I))/(10.0*H)                             ADA04830
           IF (ERR(I).GT.ERRMAX) THEN                                         ADA04840
              ERRMAX = ERR(I)                                                 ADA04850
           ENDIF                                                              ADA04860
  100 CONTINUE                                                                ADA04870
        RETURN                                                                ADA04880
        END                                                                   ADA04890
```

H.4 The Multi-Rate Integration Algorithm

```
      SUBROUTINE  SRATE(NS,NF,S,F,HS,DERIVS,T,TFIN)                    MUL02230
      IMPLICIT REAL(A-H,O-Z)                                           MUL02240
      REAL  F(NF),S(NS),ERR(10),PS(10),P2S(10),P3S(10),DP3S(10)        MUL02250
      COMMON/SAME/DS(10),P1S(10),DPS(10),DP1S(10),DP2S(10)             MUL02260
      COMMON/SLOW/TOLERS,HSMAX,HSMIN,COMINT                            MUL02270
      EXTERNAL DERIVF,DERIVS                                           MUL02280
      CHARACTER*1 CHANGE, CHOICE                                       MUL02290
                                                                       MUL02300
C                              Integration of Slow System              MUL02310
C                              Utilizing Integration Algorithm         MUL02320
C                              Based on Runge-Kutta Merson             MUL02330
C                              Variable Step Method.                   MUL02340
C                              HS- Slow System   HF- Fast System       MUL02350
      ORD     = 5                                                      MUL02360
      IF (COMINT.LE.0.0) THEN                                          MUL02370
         COMINT = (TFIN - T)/100                                       MUL02380
      ENDIF                                                            MUL02390
      TCOM = T + COMINT                                                MUL02400
                                                                       MUL02410
C                              Calculate Starting Step Size            MUL02420
      CALL DERIVS(S,DS,NF,NS,F)                                        MUL02430
      FDER = 0.0                                                       MUL02440
      DO 39 I = 1,N                                                    MUL02450
         IF (ABS(DS(I)).GT.FDER) THEN                                  MUL02460
            FDER = ABS(DS(I))                                          MUL02470
         ENDIF                                                         MUL02480
   39 CONTINUE                                                         MUL02490
      IF (FDER.NE.0.0) THEN                                            MUL02500
         HPOS = SQRT(0.5 * TOL/FDER)                                   MUL02510
         IF (HBOUND.LE.0.0) THEN                                       MUL02520
            HBOUND = 0.1                                               MUL02530
         ENDIF                                                         MUL02540
         IF (HPOS.GT.HBOUND) THEN                                      MUL02550
            HSTART = HBOUND                                            MUL02560
         ELSE                                                          MUL02570
            HSTART = HPOS                                              MUL02580
         ENDIF                                                         MUL02590
      ELSE                                                             MUL02600
         IF (HBOUND.LE.0.0) THEN                                       MUL02610
            HBOUND = 0.1                                               MUL02620
         ENDIF                                                         MUL02630
         HSTART = HBOUND                                               MUL02640
      ENDIF                                                            MUL02650
      IF (HINIT.NE.0.0) THEN                                           MUL02660
         HS= HINIT                                                     MUL02670
      ELSE                                                             MUL02680
         HINIT = HSTART                                                MUL02690
         HS    = HINIT                                                 MUL02700
      ENDIF                                                            MUL02710
                                                                       MUL02720
      IF (HS.GT.HBOUND) THEN                                           MUL02730
         PRINT*,'HINIT > HBOUND. DO YOU WISH TO CARRY OUT INTEGRATION  MUL02740
     + WITH THIS HINIT (Y) OR HBOUND (N)?'                             MUL02750
         READ*, CHOICE                                                 MUL02760
         IF (CHOICE.EQ.'Y') THEN                                       MUL02770
            HS= HINIT                                                  MUL02780
```

```
          ELSE                                                          MUL02790
             HS    = HBOUND                                           MUL02800
             HINIT = HBOUND                                           MUL02810
          ENDIF                                                       MUL02820
       ENDIF                                                          MUL02830
       IF (COMINT.LE.0) THEN                                          MUL02840
          COMINT = (TFIN - T)/100                                     MUL02850
       ENDIF                                                          MUL02860
       IF (HMIN.LE.0.0) THEN                                          MUL02870
          HMIN = HINIT/10000                                          MUL02880
       ENDIF                                                          MUL02890
                                                                      MUL02900
 99    IF (T.LT. TFIN)  THEN                                          MUL02910
                                                                      MUL02920
 188      IF (T.LT.TCOM) THEN                                         MUL02930
             IF (((TCOM - T).LE.(HS + HSMIN)) .AND. (NFLAG.EQ.0)) THEN MUL02940
                HS = (TCOM - T)/2                                     MUL02950
                NFLAG = 1                                             MUL02960
             ENDIF                                                    MUL02970
          TBEG = T                                                    MUL02980
                                                                      MUL02990
C                                  Determine K1 for components of     MUL03000
C                                  Slow System Represented By DS Below MUL03010
                                                                      MUL03020
          CALL DERIVS(S,DS,NF,NS,F)                                   MUL03030
                                                                      MUL03040
                                                                      MUL03050
C                                  Find Predicted Value of Slow       MUL03060
C                                  System Vector Y for all its        MUL03070
C                                  Components.                        MUL03080
                                                                      MUL03090
          DO 12 I = 1,NS                                              MUL03100
             PS(I) = S(I) + HS/3.0 * DS(I)                            MUL03110
 12       CONTINUE                                                    MUL03120
C                                Integrate Fast System between        MUL03130
C                                TN and TN + H/3 to determine         MUL03140
C                                Components of X involved in          MUL03150
C                                Coupling to Slow System              MUL03160
          TF = T + HS/3.0                                             MUL03170
          CALL FRATE(NS,NF,S,F,HS,DERIVF,TF,T,TBEG)                   MUL03180
                                                                      MUL03190
C                                Determine K2 for components of       MUL03200
C                                Slow System.                         MUL03210
                                                                      MUL03220
                                                                      MUL03230
          CALL DERIVS(PS,DPS,NF,NS,F)                                 MUL03240
                                                                      MUL03250
                                                                      MUL03260
C                                  Determine K3 for components of     MUL03270
C                                  Slow System.                       MUL03280
          DO 14 I =1,NS                                               MUL03290
             P1S(I) = S(I) + HS/6.0 * (DS(I) + DPS(I))                MUL03300
 14       CONTINUE                                                    MUL03310
          CALL DERIVS(P1S,DP1S,NF,NS,F)                               MUL03320
C                                Integrate Fast System Between        MUL03330
C                                TN + H/3 and TN + H/2 to determine   MUL03340
C                                Components of X involved in          MUL03350
C                                Coupling to Slow System              MUL03360
C                                                                     MUL03370
```

```
      TF = TBEG + HS/2.0                                                MUL03380
      CALL FRATE(NS,NF,S,F,HS,DERIVF,TF,T,TBEG)                         MUL03390
C                                                                       MUL03400
      DO 16 I =1,NS                                                     MUL03410
         P2S(I) = S(I) + HS/8.0 * (DS(I) + DP1S(I) * 3.0)               MUL03420
 16   CONTINUE                                                          MUL03430
C                                                                       MUL03440
C                              Determine K4 for components of           MUL03450
C                              Slow System.                             MUL03460
      CALL DERIVS(P2S,DP2S,NF,NS,F)                                     MUL03470
C                             Integrate Fast System between             MUL03480
C                             TN + H/2 and TN + H to determine          MUL03490
C                             Components of X involved in               MUL03500
C                             Coupling to Slow System                   MUL03510
      TF = TBEG + HS                                                    MUL03520
      CALL FRATE(NS,NF,S,F,HS,DERIVF,TF,T,TBEG)                         MUL03530
      DO 18 I =1,NS                                                     MUL03540
        P3S(I)=S(I)+HS/2.0*DS(I)-3.0*HS*DP1S(I)/2.0+2.0*HS*DP2S(I)      MUL03550
   18 CONTINUE                                                          MUL03560
C                              Determine K5 for components of           MUL03570
C                              Slow System.                             MUL03580
      CALL DERIVS(P3S,DP3S,NF,NS,F)                                     MUL03590
C                              Determine Largest Error                  MUL03600
      ERRMAX = 0.0                                                      MUL03610
      DO 40 I=1,NS                                                      MUL03620
         ERR(I) = HS * (DS(I)/3.0 - 3.0*DP1S(I)/2.0 + 4.0 * DP2S(I)/    MUL03630
     +      3.0 - DP3S(I)/6.0)                                          MUL03640
         IF (ERR(I) .GT. ERRMAX)  THEN                                  MUL03650
            ERRMAX = ERR(I)                                             MUL03660
         ENDIF                                                          MUL03670
   40 CONTINUE                                                          MUL03680
C                              Tolerance Check                          MUL03690
      IF (ERRMAX.LT.TOLERS) THEN                                        MUL03700
C                              Compute values of Slow System            MUL03710
C                              Components at the end of the time stepMUL03720
                                                                        MUL03730
         DO 29 I =1,NS                                                  MUL03740
            S(I) = S(I) + HS/6.0*(DS(I) + 4.0*DP2S(I) + DP3S(I))        MUL03750
   29    CONTINUE                                                       MUL03760
      ELSE                                                              MUL03770
         T = T - HS                                                     MUL03780
      ENDIF                                                             MUL03790
C              T = T-HS,Then Change Value Of  HS                        MUL03800
                                                                        MUL03810
C                              Adjust Step Size If Necessary            MUL03820
                                                                        MUL03830
         IF (NFLAG.EQ.0) THEN                                           MUL03840
            HS = 0.9 * HS * (TOLERS/ERRMAX) ** (1/ORD)                  MUL03850
            IF (HS.LT.HSMIN) THEN                                       MUL03860
               PRINT*,'HSMIN Exceeded. Do you wish to change value ofMUL03870
     + HSMIN (Y) or would you rather halt the integration and further   MUL03880
     +investigate the model (N) ?'                                      MUL03890
               READ*,CHANGE                                             MUL03900
               IF (CHANGE.EQ.'Y') THEN                                  MUL03910
                  PRINT*,'Please enter new value of HSMIN. Note that MUL03920
     +For Integration To Proceed,it should be less than',HS             MUL03930
                  READ*, NUVAL                                          MUL03940
                  HSMIN = NUVAL                                         MUL03950
               ELSE                                                     MUL03960
```

```
                  STOP                                                  MUL03970
               ENDIF                                                    MUL03980
            ENDIF                                                       MUL03990
            IF (HS.GT.HSMAX) THEN                                       MUL04000
               HS = HSMAX                                               MUL04010
            ENDIF                                                       MUL04020
         ENDIF                                                          MUL04030
      ELSE                                                              MUL04040
         CALL OUTPUT(T,S,NS)                                            MUL04050
         TCOM = TCOM + COMINT                                           MUL04060
         IF ((TCOM.GE.TFIN) .AND. (T.NE.TFIN)) THEN                     MUL04070
            TCOM = TFIN                                                 MUL04080
         ENDIF                                                          MUL04090
         NFLAG = 0                                                      MUL04100
         GOTO 99                                                        MUL04110
      ENDIF                                                             MUL04120
      GOTO 188                                                          MUL04130
   ENDIF                                                                MUL04140
   CALL OUTPUT(T,S,NS)                                                  MUL04150
   RETURN                                                               MUL04160
   END                                                                  MUL04170
                                                                        MUL04180
                                                                        MUL04190
                                                                        MUL04200
   SUBROUTINE FRATE(NS,NF,S,F,HS,DERIVF,TF,T,TBEG)                      MUL04210
   IMPLICIT REAL(A-H,O-Z)                                               MUL04220
   REAL  F(NF),S(NS),W1(10),W2(10),W3(10),W4(10),ERR(10),               MUL04230
  +INTS(4),DF1(10),DF2(10),DF3(10),DF4(10),DF5(10),DF(10)               MUL04240
   COMMON/SAME/DS(10),P1S(10),DPS(10),DP1S(10),DP2S(10)                 MUL04250
   COMMON/FAST/TOLERF,HFMAX,HFMIN                                       MUL04260
   CHARACTER*1 CHECK                                                    MUL04270
                                                                        MUL04280
C                                Subroutine For Fast Subsystem          MUL04290
C                                Integration. The Runge-Kutta           MUL04300
C                                Merson Method Is Utilized.             MUL04310
   HFMIN = 0.0000000000000001                                           MUL04320
   ORD   = 5                                                            MUL04330
   HF    = HINIT                                                        MUL04340
                                                                        MUL04350
   TNOW = T                                                             MUL04360
                                                                        MUL04370
 5 IF (T.LT.TF) THEN                                                    MUL04380
                                                                        MUL04390
C                                Depending on the time interpolate      MUL04400
C                                or extrapolate for values of Slow      MUL04410
C                                System Components involved in couplingMUL04420
                                                                        MUL04430
                                                                        MUL04440
      IF (T .LT. TBEG + HS/3.0) THEN                                    MUL04450
         CALL INTER1 (T,TNOW,S,DS,NS,INTS)                              MUL04460
         IF (NFIRST.EQ.0) THEN                                          MUL04470
                                                                        MUL04480
C                                Calculate Starting Step Size           MUL04490
            CALL DERIVF(F,DF,NF,NS,INTS)                                MUL04500
            FDER  = 0.0                                                 MUL04510
            DO 29 I = 1,NF                                              MUL04520
            IF (ABS(DF(I)).GT.FDER) THEN                                MUL04530
               FDER = ABS(DF(I))                                        MUL04540
            ENDIF                                                       MUL04550
```

```
29          CONTINUE                                                    MUL04560
            HPOS  = SQRT(0.5 * TOLERF/FDER)                             MUL04570
            IF (HFMAX.LE.0.0) THEN                                      MUL04580
               HFMAX = 0.1                                              MUL04590
            ENDIF                                                       MUL04600
            IF (HPOS.GT.HFMAX) THEN                                     MUL04610
               HINIT  = HFMAX                                           MUL04620
            ELSE                                                        MUL04630
               HINIT  = HPOS                                            MUL04640
            ENDIF                                                       MUL04650
            IF (HFMIN.LE.0.0) THEN                                      MUL04660
               HFMIN = HINIT/10000                                      MUL04670
            ENDIF                                                       MUL04680
            HF = HINIT                                                  MUL04690
         ENDIF                                                          MUL04700
         NFIRST = 1                                                     MUL04710
      ELSE IF (T .LT. TBEG + HS/2.0) THEN                               MUL04720
         CALL INTER2 (T,TNOW,P1S,DS,DPS,DP1S,NS,INTS)                   MUL04730
      ELSE IF (T .LT. TBEG + HS) THEN                                   MUL04740
         CALL EXTRAP(T,TNOW,P1S,DP2S,NS,INTS)                           MUL04750
      ENDIF                                                             MUL04760
      CALL DERIVF(F,DF1,NF,NS,INTS)                                     MUL04770
      DO 31 I =1,NF                                                     MUL04780
         W1(I) = DF1(I) * HF/3.0 + F(I)                                 MUL04790
31    CONTINUE                                                          MUL04800
      CALL DERIVF(W1,DF2,NF,NS,INTS)                                    MUL04810
      DO 33 I=1,NF                                                      MUL04820
         W2(I) = (DF1(I)+DF2(I))*HF/6.0 + F(I)                          MUL04830
33    CONTINUE                                                          MUL04840
      CALL DERIVF(W2,DF3,NF,NS,INTS)                                    MUL04850
      DO 35 I=1,NF                                                      MUL04860
         W3(I) = (DF1(I) + 3.0*DF3(I)) * HF*0.125 +F(I)                 MUL04870
35    CONTINUE                                                          MUL04880
      CALL DERIVF(W3,DF4,NF,NS,INTS)                                    MUL04890
      DO 37 I=1,NF                                                      MUL04900
         W4(I) = DF1(I)*HF*0.5-DF3(I)*HF*1.5+DF4(I)*HF*2.0 +F(I)        MUL04910
37    CONTINUE                                                          MUL04920
      CALL DERIVF(W4,DF5,NF,NS,INTS)                                    MUL04930
C                              Determine Largest Error                  MUL04940
      ERRMAX = 0.0                                                      MUL04950
      DO 41 I =1,NF                                                     MUL04960
         ERR(I) = HF*(DF1(I) *HF/3.0-1.5*HF*DF3(I) +4.0* DF4(I)*HF      MUL04970
     +   /3.0 -HF*DF5(I)/6.0)                                           MUL04980
         IF(ERR(I).GT.ERRMAX) THEN                                      MUL04990
           ERRMAX = ERR(I)                                              MUL05000
         ENDIF                                                          MUL05010
41    CONTINUE                                                          MUL05020
C                              Tolerance Check                          MUL05030
                                                                        MUL05040
      IF (ERRMAX .LE. TOLERF)THEN                                       MUL05050
         T = T + HF                                                     MUL05060
         DO 39 I=1,NF                                                   MUL05070
            F(I) = (DF1(I) + 4.0*DF4(I)+DF5(I))*HF/6.0+F(I)             MUL05080
39       CONTINUE                                                       MUL05090
      ENDIF                                                             MUL05100
C                              Adjust Step Size If Necessary            MUL05110
      IF (ERRMAX .GT. 0.0)THEN                                          MUL05120
         HF= 0.9 * HF* (TOLERF/ERRMAX) ** (1/ORD)                       MUL05130
         IF (HF.LT.HFMIN) THEN                                          MUL05140
```

```
                PRINT*,'HFMIN EXCEEDED. DO YOU WISH TO CHANGE VALUE OFMUL05150
     + HFMIN (Y) OR WOULD YOU RATHER HALT THE INTEGRATION AND FURTHER   MUL05160
     +INVESTIGATE THE MODEL (N) ?'                                      MUL05170
                READ*,CHECK                                             MUL05180
                IF (CHECK  .EQ.'Y') THEN                                MUL05190
                   PRINT*,'PLEASE ENTER NEW VALUE OF HFMIN. NOTE THAT MUL05200
     +FOR INTEGRATION TO PROCEED,IT SHOULD BE LESS THAN',HF             MUL05210
                   READ*, NUVAL                                         MUL05220
                   HFMIN = NUVAL                                        MUL05230
                ELSE                                                    MUL05240
                   STOP                                                 MUL05250
                ENDIF                                                   MUL05260
             ENDIF                                                      MUL05270
             IF (HF.GT.HFMAX) THEN                                      MUL05280
                HF = HFMAX                                              MUL05290
             ENDIF                                                      MUL05300
          ENDIF                                                         MUL05310
          IF (HF.GE. TF - T) HF = TF-T                                  MUL05320
          GOTO 5                                                        MUL05330
       ENDIF                                                            MUL05340
       RETURN                                                           MUL05350
       END                                                              MUL05360
                                                                        MUL05370
                                                                        MUL06160
                                                                        MUL06620
                                                                        MUL06630
       SUBROUTINE INTER1(T,TNOW,S,DS,NS,INTS)                           MUL06640
       IMPLICIT REAL(A-H,O-Z)                                           MUL06650
       REAL S(NS),DS(NS),INTS(2)                                        MUL06660
                                                                        MUL06670
C         Subroutine that returns values of Slow System components     MUL06680
C         involved in coupling between TN and TN + H/3                 MUL06690
                                                                        MUL06700
       INTS(1) = S(6) + (T-TNOW)* DS(6)                                 MUL06710
                                                                        MUL06720
       INTS(2) = S(1) + (TF-T)*DS(1)                                    MUL06730
                                                                        MUL06740
       RETURN                                                           MUL06750
       END                                                              MUL06760
                                                                        MUL06770
       SUBROUTINE INTER2(T,TNOW,P1S,DS,DPS,DP1S,NS,INTS)                MUL06780
       IMPLICIT REAL(A-H,O-Z)                                           MUL06790
       REAL P1S(NS),DS(NS),DPS(NS),DP1S(NS),INTS(2)                     MUL06800
                                                                        MUL06810
C         Subroutine that returns values of Slow System components     MUL06820
C         involved in coupling between TN + H/3 and TN + H/2           MUL06830
                                                                        MUL06840
       INTS(1) = P1S(6)+(T-TNOW)*(-DS(6)/4.0-DPS(6) + 9.0*DP1S(6)/4.0)  MUL06850
                                                                        MUL06860
       INTS(2) = P1S(1) + (T-TNOW)*(-DS(1)/4.0-DPS(1) + 9.0*DP1S(1)/4.0) MUL06870
                                                                        MUL06880
       RETURN                                                           MUL06890
       END                                                              MUL06900
                                                                        MUL06910
       SUBROUTINE EXTRAP(T,TNOW,P1S,DP2S,NS,INTS)                       MUL06920
       IMPLICIT REAL(A-H,O-Z)                                           MUL06930
       REAL P1S(NS),DP2S(NS),INTS(2)                                    MUL06940
                                                                        MUL06950
C         Subroutine that returns values of Slow System components     MUL06960
```

```
C       involved in coupling between TN + H/2 and TN + H                MUL06970
                                                                        MUL06980
      INTS(1) = P1S(6) + (T-TNOW)*DP2S(6)                               MUL06990
                                                                        MUL07000
      INTS(2) = P1S(1) + (T-TNOW)*DP2S(1)                               MUL07010
                                                                        MUL07020
      RETURN                                                            MUL07030
      END                                                               MUL07040
                                                                        MUL07050
```

H.5 The Discontinuity Processing Algorithm

```
      SUBROUTINE DISCON(T,Y,TE,N,TFIN,HMIN,DERIV,TOL,NDF)                BAL01840
                                                                         BAL01850
      IMPLICIT REAL*8 (A-H,O-Z)                                          BAL01860
      DIMENSION DY(5),F(10,10),TS(10,10),W4(9),ERR(9),ERD(9),DF(5)       BAL01870
     +,DY1(9),DY2(9),DY3(9),DY4(9),W1(9),W2(9),W3(9),Y(5),PDF(5),TE(5)   BAL01880
      COMMON/HOPE/YLAST(5),HINIT                                         BAL01890
      COMMON/TIMVAL/THOR                                                 BAL01900
      COMMON/WHY/COMINT,NOUT,NTIMUP                                      BAL01910
      COMMON/FLAG/NUMTE,NTIMEV,NFIRST,NTEALL                             BAL01920
      COMMON/DIFF/YVAL(500),TIME(500),YVAL1(500),NPL                     BAL01930
      EXTERNAL DERIV                                                     BAL01940
      CHARACTER*1 CHOICE                                                 BAL01950
                                                                         BAL01960
      NOUT = 0                                                           BAL01970
C                                  Calculate Starting Step Size          BAL01980
      CALL DERIV(T,Y,DY,N)                                               BAL01990
      FDER = 0.0                                                         BAL02000
      DO 29 I = 1,N                                                      BAL02010
         IF (ABS(DY(I)).GT.FDER) THEN                                    BAL02020
            FDER = ABS(DY(I))                                            BAL02030
         ENDIF                                                           BAL02040
   29 CONTINUE                                                           BAL02050
      IF (FDER.NE.0.0) THEN                                              BAL02060
         HPOS = SQRT(0.5 * TOL/FDER)                                     BAL02070
         IF (HBOUND.LE.0.0) THEN                                         BAL02080
            HBOUND = 0.1                                                 BAL02090
         ENDIF                                                           BAL02100
         IF (HPOS.GT.HBOUND) THEN                                        BAL02110
            HSTART = HBOUND                                              BAL02120
         ELSE                                                            BAL02130
            HSTART = HPOS                                                BAL02140
         ENDIF                                                           BAL02150
      ELSE                                                               BAL02160
         IF (HBOUND.LE.0.0) THEN                                         BAL02170
            HBOUND = 0.1                                                 BAL02180
         ENDIF                                                           BAL02190
         HSTART = HBOUND                                                 BAL02200
      ENDIF                                                              BAL02210
      IF (HINIT.NE.0.0) THEN                                             BAL02220
         H = HINIT                                                       BAL02230
      ELSE                                                               BAL02240
         HINIT = HSTART                                                  BAL02250
         H     = HINIT                                                   BAL02260
      ENDIF                                                              BAL02270
                                                                         BAL02280
      IF (H.GT.HBOUND) THEN                                              BAL02290
         PRINT*,'HINIT > HBOUND. DO YOU WISH TO CARRY OUT INTEGRATION    BAL02300
     + WITH THIS HINIT (Y) OR HBOUND (N)?'                               BAL02310
         READ*, CHOICE                                                   BAL02320
         IF (CHOICE.EQ.'Y') THEN                                         BAL02330
            H = HINIT                                                    BAL02340
         ELSE                                                            BAL02350
            H     = HBOUND                                               BAL02360
            HINIT = HBOUND                                               BAL02370
         ENDIF                                                           BAL02380
      ENDIF                                                              BAL02390
```

```
      IF (COMINT.LE.0) THEN                                              BAL02400
         COMINT = (TFIN - T)/100                                         BAL02410
      ENDIF                                                              BAL02420
      IF (HMIN.LE.0.0) THEN                                              BAL02430
         HMIN = HINIT/10000                                              BAL02440
      ENDIF                                                              BAL02450
                                                                         BAL02460
      IT    = 1                                                          BAL02470
      NC    = 1                                                          BAL02480
      ITER  = 0                                                          BAL02490
      IF (COMINT.EQ.0.0) THEN                                            BAL02500
         COMINT = (TFIN - T)/100                                         BAL02510
      ENDIF                                                              BAL02520
      TCOM = T + COMINT                                                  BAL02530
C                                  Determine the current time horizon    BAL02540
    7 IF (T.LT.TFIN) THEN                                                BAL02550
         IF ((IT.LE.NUMTE).AND. (NTIMUP.EQ.1)) THEN                      BAL02560
            IF (TE(IT).LT.TCOM) THEN                                     BAL02570
               THOR = TE(IT)                                             BAL02580
               NTIMUP= 0                                                 BAL02590
            ELSE                                                         BAL02600
               THOR = TCOM                                               BAL02610
               NTIMUP= 0                                                 BAL02620
            ENDIF                                                        BAL02630
         ENDIF                                                           BAL02640
         IF ((IT.GT.NUMTE).AND. (NTIMUP.EQ.1)) THEN                      BAL02650
            IF (TFIN.GT.TCOM) THEN                                       BAL02660
               THOR = TCOM                                               BAL02670
               NTIMUP=0                                                  BAL02680
            ELSE                                                         BAL02690
               THOR = TFIN                                               BAL02700
               NTIMUP= 0                                                 BAL02710
            ENDIF                                                        BAL02720
         ENDIF                                                           BAL02730
    8    IF ((T + H.LT.THOR) .OR. (NOUT.EQ.1))THEN                       BAL02740
           IF (NTEALL.EQ.1) THEN                                         BAL02750
              IF (NTIMEV.EQ.1) THEN                                      BAL02760
C                                Model Update for Time Events            BAL02770
                 CALL MODUPD(DF,NDF,T,Y,N,TOL)                           BAL02780
                 CALL OUTPUT(Y,N,T)                                      BAL02790
                      NTIMEV = 0                                         BAL02800
                      NOUT   = 0                                         BAL02810
                      IT = IT + 1                                        BAL02820
                      NTIMUP = 1                                         BAL02830
              ENDIF                                                      BAL02840
              CALL NORMIN(Y,H,T,TOL,DF,PDF,TLAST,TCOM,N,NDF,TFIN)        BAL02850
           ELSE                                                          BAL02860
              CALL DISFUN(Y,T,N,NDF,DF)                                  BAL02870
              NUM   = 0                                                  BAL02880
              NOT   = 0                                                  BAL02890
              IF (NFIRST.EQ.0) THEN                                      BAL02900
                 NFIRST = 1                                              BAL02910
                 CALL NORMIN(Y,H,T,TOL,DF,PDF,TLAST,TCOM,N,NDF,TFIN)     BAL02920
                 CALL DISFUN(Y,T,N,NDF,DF)                               BAL02930
              ENDIF                                                      BAL02940
C                                Detection of Discontinuities            BAL02950
              I     = 1                                                  BAL02960
    2         IF (I.LE.NDF)THEN                                          BAL02970
                 IF (PDF(I)*DF(I).LT.0.0)THEN                            BAL02980
```

```
                NUM=I                                                   BAL02990
                IF (DABS(DF(I)).GT.TOL)THEN                             BAL03000
                   NOT=1                                                BAL03010
                END IF                                                  BAL03020
             END IF                                                     BAL03030
             I=I+1                                                      BAL03040
             GOTO 2                                                     BAL03050
          ENDIF                                                         BAL03060
          IF (ITER.NE.1) THEN                                           BAL03070
             IF (NUM.EQ.0) THEN                                         BAL03080
                IF (NTIMEV.EQ.1) THEN                                   BAL03090
C                          Model Update As Specified By Time EventBAL03100
C                          No State Event In Last Step                  BAL03110
                   CALL MODUPD(DF,NDF,T,Y,N,TOL)                        BAL03120
                   CALL OUTPUT(Y,N,T)                                   BAL03130
                   NTIMEV = 0                                           BAL03140
                   NOUT   = 0                                           BAL03150
                   IT = IT + 1                                          BAL03160
                   NTIMUP = 1                                           BAL03170
                ELSE                                                    BAL03180
                CALL NORMIN(Y,H,T,TOL,DF,PDF,TLAST,TCOM,N,NDF,TFIN)BAL03190
                ENDIF                                                   BAL03200
             ELSE                                                       BAL03210
                I=1                                                     BAL03220
  83            IF (I.LE.NDF)THEN                                       BAL03230
                   F(1,I)=PDF(I)                                        BAL03240
                   F(2,I)=DF(I)                                         BAL03250
                   I=I+1                                                BAL03260
                   GOTO 83                                              BAL03270
                ENDIF                                                   BAL03280
                TS(2,1)=T                                               BAL03290
                TS(1,1)=TLAST                                           BAL03300
                NC     = 2                                              BAL03310
                J      = 2                                              BAL03320
                NDIS   = NUM                                            BAL03330
                IIME   = 0                                              BAL03340
                CALL AITKEN(NC,J,TS,F,PDF,NDIS)                         BAL03350
                HDIS   = TS(NC,NC)-TLAST                                BAL03360
                ITER   = 1                                              BAL03370
  16            T      = TLAST                                          BAL03380
                I      = 1                                              BAL03390
  90            IF (I.LE.N)THEN                                         BAL03400
                   Y(I)=YLAST(I)                                        BAL03410
                   I=I+1                                                BAL03420
                   GOTO 90                                              BAL03430
                ENDIF                                                   BAL03440
                IIME=IIME+1                                             BAL03450
  19            CALL RKD(N,T,Y,HDIS,DERIV,TOL)                          BAL03460
             ENDIF                                                      BAL03470
          ELSEIF(NOT.EQ.0.AND.NUM.NE.0) THEN                            BAL03480
C                              Discontinuity present and                BAL03490
C                              within Tolerance.No interpolationBAL03500
C                              required. Model Update done.             BAL03510
C                              Model Update by State Event              BAL03520
             CALL MODUPD(DF,NDF,T,Y,N,TOL)                              BAL03530
             CALL OUTPUT(Y,N,T)                                         BAL03540
             NOUT   = 0                                                 BAL03550
             NC     = 1                                                 BAL03560
             ITER   = 0                                                 BAL03570
```

```
                IF (NTIMEV.EQ.0) THEN                                   BAL03580
                   CALL NORMIN(Y,H,T,TOL,DF,PDF,TLAST,TCOM,N,NDF,TFIN)  BAL03590
                ELSE                                                    BAL03600
                   HDIS = THOR - T                                      BAL03610
                   YVAL(NPL) = Y(1)                                     BAL03620
                   YVAL1(NPL) = Y(2)                                    BAL03630
                   TIME(NPL) = T                                        BAL03640
                   NPL = NPL + 1                                        BAL03650
                   CALL RKD(N,T,Y,HDIS,DERIV,TOL)                       BAL03660
                   I=1                                                  BAL03670
  46               IF (I.LE.NDF)THEN                                    BAL03680
                      PDF(I) = DF(I)                                    BAL03690
                      I      = I+1                                      BAL03700
                      GOTO 46                                           BAL03710
                   ENDIF                                                BAL03720
                ENDIF                                                   BAL03730
             ELSE                                                       BAL03740
C                                  ITER = 1, Discontinuity present,     BAL03750
C                                  value of DY not within tolerance     BAL03760
C                                  Need to interpolate.                 BAL03770
                I = 1                                                   BAL03780
   4            IF (I.LE.NDF)THEN                                       BAL03790
                   F(NC+1,I)=DF(I)                                      BAL03800
                   I = I + 1                                            BAL03810
                   GOTO 4                                               BAL03820
                ENDIF                                                   BAL03830
                TS(NC+1,1)= T                                           BAL03840
                NC        = NC + 1                                      BAL03850
C                             Sort in ascending order of D.F. from      BAL03860
C                             0,the D.F.'s and their corresponding      BAL03870
C                             Times.                                    BAL03880
  11            CALL SORT(F,TS,NC,NDF)                                  BAL03890
                CALL SELECT(F,TS,NC,NDIS,PDF,J,NDF)                     BAL03900
C                             CARRY OUT INVERSE AITKEN INTERPOLATION    BAL03910
                CALL AITKEN(NC,J,TS,F,PDF,NDIS)                         BAL03920
                HDIS   = TS(NC,NC)-TLAST                                BAL03930
                ITER   = 1                                              BAL03940
                T      = TLAST                                          BAL03950
                I      = 1                                              BAL03960
   94           IF (I.LE.N)THEN                                         BAL03970
                   Y(I)=YLAST(I)                                        BAL03980
                   I=I+1                                                BAL03990
                   GOTO 94                                              BAL04000
                ENDIF                                                   BAL04010
                IIME=IIME+1                                             BAL04020
                CALL RKD(N,T,Y,HDIS,DERIV,TOL)                          BAL04030
             ENDIF                                                      BAL04040
          ENDIF                                                         BAL04050
          GOTO 8                                                        BAL04060
       ENDIF                                                            BAL04070
C                               Determine The Step Size needed to       BAL04080
C                               reach the Time Event Discontinuity      BAL04090
C                               or the end time of the Simulation.      BAL04100
C                               Use the Modified R-K Merson to reach    BAL04110
C                               that point exactly                      BAL04120
                                                                        BAL04130
       IF (NTIMUP.EQ.0) THEN                                            BAL04140
          HDIS = THOR - T                                               BAL04150
          CALL RKD(N,T,Y,HDIS,DERIV,TOL)                                BAL04160
```

```
            I=1                                                          BAL04170
 96         IF (I.LE.NDF)THEN                                            BAL04180
               PDF(I) = DF(I)                                            BAL04190
               I      = I+1                                              BAL04200
               GOTO 96                                                   BAL04210
            ENDIF                                                        BAL04220
            NOUT = 1                                                     BAL04230
            IF (THOR.EQ.TE(IT)) NTIMEV = 1                               BAL04240
         ENDIF                                                           BAL04250
         IF (THOR.EQ.TFIN) THEN                                          BAL04260
            CALL OUTPUT(Y,N,T)                                           BAL04270
         ENDIF                                                           BAL04280
         GOTO 7                                                          BAL04290
      ENDIF                                                              BAL04300
      RETURN                                                             BAL04310
      END                                                                BAL04320
                                                                         BAL04330
      SUBROUTINE AITKEN(NC,J,TS,F,PDF,NDIS)                              BAL04340
      IMPLICIT REAL*8 (A-H,O-Z)                                          BAL04350
      DIMENSION TS(10,10),F(10,5),PDF(5)                                 BAL04360
                                                                         BAL04370
      TOL  = 0.00001                                                     BAL04380
  12  DO 13 K=2,NC                                                       BAL04390
      DO 13 I=K,NC                                                       BAL04400
      TS(I,K)=(TS(K-1,K-1)*(F(I,NDIS)+DSIGN(TOL,PDF(NDIS))*0.5)          BAL04410
     +-TS(I,K-1)*(F(K-1,NDIS)+0.5*DSIGN(TOL,PDF(NDIS))))                 BAL04420
     +/(F(I,NDIS)-F(K-1,NDIS))                                           BAL04430
  13  CONTINUE                                                           BAL04440
C                                Check Validity of Interpolation Table   BAL04450
C                                                                        BAL04460
      IF (TS(NC,NC).GT.TS(J,1).OR.TS(NC,NC).LT.TS(J-1,1))THEN            BAL04470
         WRITE(6,120)TS(NC,NC)                                           BAL04480
 120     FORMAT(4X,'OUT OF RANGE',F10.8)                                 BAL04490
         IF (J.EQ.NC)THEN                                                BAL04500
            I =1                                                         BAL04510
 60         IF (I.LT.NC)THEN                                            BAL04520
               TS(I,1)=TS(I+1,1)                                         BAL04530
               N1=1                                                      BAL04540
 70            IF (N1.LE.NDF)THEN                                        BAL04550
                  F(I,N1)=F(I+1,N1)                                      BAL04560
                  N1=N1+1                                                BAL04570
                  GOTO 70                                                BAL04580
               ENDIF                                                     BAL04590
               I=I+1                                                     BAL04600
               GOTO 60                                                   BAL04610
            ENDIF                                                        BAL04620
            J=J-1                                                        BAL04630
            NC=NC-1                                                      BAL04640
            WRITE(6,130)                                                 BAL04650
130         FORMAT(10X,'ERASE FIRST ROW')                                BAL04660
            GOTO 12                                                      BAL04670
         END IF                                                          BAL04680
         IF (J.EQ.2)THEN                                                 BAL04690
            NC=NC-1                                                      BAL04700
            WRITE(6,140)                                                 BAL04710
140         FORMAT(10X,'ERASE LAST ROW')                                 BAL04720
            GOTO 12                                                      BAL04730
         END IF                                                          BAL04740
         TS(1,1)=TS(J-1,1)                                               BAL04750
```

```
      TS(2,1)=TS(J,1)                                                  BAL04760
      I=1                                                              BAL04770
80    IF (I.LE.DDF)THEN                                                BAL04780
         F(1,I)=F(J-1,I)                                               BAL04790
         F(2,I)=F(J,I)                                                 BAL04800
         I=I+1                                                         BAL04810
         GOTO 80                                                       BAL04820
      ENDIF                                                            BAL04830
      NC=2                                                             BAL04840
      J=2                                                              BAL04850
      GOTO 12                                                          BAL04860
   END IF                                                              BAL04870
   RETURN                                                              BAL04880
   END                                                                 BAL04890
                                                                       BAL04900
                                                                       BAL04910
                                                                       BAL04920
   SUBROUTINE NORMIN(Y,H,T,TOL,DF,PDF,TLAST,TCOM,N,NDF,TFIN)           BAL04930
   IMPLICIT REAL*8 (A-H,O-Z)                                           BAL04940
   DIMENSION Y(5),DF(5),PDF(5)                                         BAL04950
   COMMON/HOPE/YLAST(5),HINIT                                          BAL04960
   COMMON/WHY/COMINT,NOUT,NTIMUP                                       BAL04970
   COMMON/TIMVAL/THOR                                                  BAL04980
   EXTERNAL DERIV                                                      BAL04990
   HMAX  = 0.1                                                         BAL05000
   HMIN  = 0.0000000001                                                BAL05010
24 TLAST = T                                                           BAL05020
   I     = 1                                                           BAL05030
C                                To prevent discontinuity detection   BAL05040
C                                hazards                              BAL05050
95 IF (I.LE.N)THEN                                                     BAL05060
      IF (YLAST(2).LT.0.0) YLAST(2) = 0.0                              BAL05070
      YLAST(I) = Y(I)                                                  BAL05080
      I        = I+1                                                   BAL05090
      GOTO 95                                                          BAL05100
   ENDIF                                                               BAL05110
   I=1                                                                 BAL05120
96 IF (I.LE.NDF)THEN                                                   BAL05130
      IF (DF(I).LT.0.0)  DF(I) = 0.0                                   BAL05140
      PDF(I) = DF(I)                                                   BAL05150
      I      = I+1                                                     BAL05160
      GOTO 96                                                          BAL05170
   ENDIF                                                               BAL05180
C                                Output Values of Time with           BAL05190
C                                Corresponding State Values.          BAL05200
C                                Collect Output Values for plotting   BAL05210
   IF (NOUT.NE.1)  THEN                                                BAL05220
      CALL RKN(N,T,Y,H,DERIV,TOL,HMAX,HMIN)                            BAL05230
   ELSE                                                                BAL05240
        CALL OUTPUT(Y,N,T)                                             BAL05250
        TCOM = T + COMINT                                              BAL05260
        NOUT = 0                                                       BAL05270
        NTIMUP = 1                                                     BAL05280
   ENDIF                                                               BAL05290
   RETURN                                                              BAL05300
   END                                                                 BAL05310
                                                                       BAL05320
                                                                       BAL05560
   SUBROUTINE RKD(N,T,Y,H,DERIV,TOL)                                   BAL05570
```

```
      IMPLICIT REAL*8 (A-H,O-Z)                                          BAL05580
      DIMENSION Y(N),W1(10),W2(10),W3(10),W4(10),ERD(10),                BAL05590
     +DY(10),DY1(10),DY2(10),DY3(10),DY4(10)                             BAL05600
      C=1                                                                BAL05610
    2 HD=H/C                                                             BAL05620
      C1=0                                                               BAL05630
    4 CALL DERIV(T,Y,DY,N)                                               BAL05640
      DO 6 I=1,N                                                         BAL05650
    6 W1(I)=DY(I)*HD/3.0+Y(I)                                            BAL05660
      CALL DERIV(HD/3.0+T,W1,DY1,N)                                      BAL05670
      DO 8 I=1,N                                                         BAL05680
    8 W2(I)=(DY(I)+DY1(I))*HD/6.0+Y(I)                                   BAL05690
      CALL DERIV(HD/3.0+T,W2,DY2,N)                                      BAL05700
      DO 10 I=1,N                                                        BAL05710
   10 W3(I)=(DY(I)+3.0*DY2(I))*HD*0.125+Y(I)                             BAL05720
      CALL DERIV(HD*0.5+T,W3,DY3,N)                                      BAL05730
      DO 12 I=1,N                                                        BAL05740
   12 W4(I)=DY(I)*HD*0.5-DY2(I)*HD*1.5+DY3(I)*HD*2.0+Y(I)                BAL05750
      CALL DERIV(HC+T,W4,DY4,N)                                          BAL05760
                                                                         BAL05770
      ERDMAX = 0.0                                                       BAL05780
      DO 40 I=1,N                                                        BAL05790
      ERD(I)=HD*DABS(DY(I)*HD/3.0-1.5*HD*DY2(I)+4.0*DY3(I)*HD/3.0        BAL05800
     +-HD*DY4(I)/6.0)                                                    BAL05810
      IF(ERD(I).GT.MAXERD) THEN                                          BAL05820
        ERDMAX = ERD(I)                                                  BAL05830
      ENDIF                                                              BAL05840
   40 CONTINUE                                                           BAL05850
                                                                         BAL05860
      IF(ERDMAX.LE.TOL)GO TO 18                                          BAL05870
      HD=HD*0.5                                                          BAL05880
      C=C+C                                                              BAL05890
      C1=C1+C1                                                           BAL05900
      GO TO 4                                                            BAL05910
   18 T=T+HD                                                             BAL05920
      DO 14 I=1,N                                                        BAL05930
   14 Y(I)=(DY(I)+4.0*DY3(I)+DY4(I))*HD/6.0+Y(I)                         BAL05940
      C1=C1+1                                                            BAL05950
      IF(C1.EQ.C)RETURN                                                  BAL05960
      GO TO 4                                                            BAL05970
      END                                                                BAL05980
                                                                         BAL05990
                                                                         BAL06000
      SUBROUTINE RKN(N,T,Y,H,DERIV,TOL,HMAX,HMIN)                        BAL06010
      IMPLICIT REAL*8 (A-H,O-Z)                                          BAL06020
      DIMENSION Y(N),W1(10),W2(10),W3(10),W4(10),ERR(10),                BAL06030
     +DY(10),DY1(10),DY2(10),DY3(10),DY4(10)                             BAL06040
      COMMON/INTEG/ORD                                                   BAL06050
      CHARACTER*1 CHOICE                                                 BAL06060
                                                                         BAL06070
    4 CALL DERIV(T,Y,DY,N)                                               BAL06080
      DO 6 I=1,N                                                         BAL06090
    6 W1(I)=DY(I)*H/3.0+Y(I)                                             BAL06100
      CALL DERIV(H/3.0+T,W1,DY1,N)                                       BAL06110
      DO 8 I=1,N                                                         BAL06120
    8 W2(I)=(DY(I)+DY1(I))*H/6.0+Y(I)                                    BAL06130
      CALL DERIV(H/3.0+T,W2,DY2,N)                                       BAL06140
      DO 10 I=1,N                                                        BAL06150
   10 W3(I)=(DY(I)+3.0*DY2(I))*H*0.125+Y(I)                              BAL06160
```

```
      CALL DERIV(H*0.5+T,W3,DY3,N)                                      BAL06170
      DO 12 I=1,N                                                       BAL06180
   12 W4(I)=DY(I)*H*0.5-DY2(I)*H*1.5+DY3(I)*H*2.0+Y(I)                  BAL06190
      CALL DERIV(HC+T,W4,DY4,N)                                         BAL06200
                                                                        BAL06210
      ERRMAX = 0.0                                                      BAL06220
      DO 40 I=1,N                                                       BAL06230
      ERR(I)=H*DABS(DY(I)*H/3.0-1.5*H*DY2(I)+4.0*DY3(I)*H/3.0           BAL06240
     +-H*DY4(I)/6.0)                                                    BAL06250
      IF(ERR(I).GT.ERRMAX) THEN                                         BAL06260
        ERRMAX = ERR(I)                                                 BAL06270
      ENDIF                                                             BAL06280
   40 CONTINUE                                                          BAL06290
                                                                        BAL06300
      IF ((ERRMAX.GT.TOL).AND.(ERRMAX.NE.0.0))    THEN                  BAL06310
         H = 0.9 * H * (TOL/ERRMAX) ** (1/ORD)                          BAL06320
            IF (H.LT.HMIN) THEN                                         BAL06330
                  PRINT*,'HMIN Exceeded. Do you wish to change value of BAL06340
     + HMIN (Y) or would you rather halt the integration and further    BAL06350
     +investigate the model (N) ?'                                      BAL06360
                  READ*,CHOICE                                          BAL06370
                  IF (CHOICE.EQ.'Y') THEN                               BAL06380
                     PRINT*,'Please enter new value of HMIN. Note that  BAL06390
     +for integration to proceed,it should be less than',H              BAL06400
                     READ*, NUVAL                                       BAL06410
                     HMIN = NUVAL                                       BAL06420
                  ELSE                                                  BAL06430
                     STOP                                               BAL06440
                  ENDIF                                                 BAL06450
            ENDIF                                                       BAL06460
         GO TO 4                                                        BAL06470
      ENDIF                                                             BAL06480
      T=T+H                                                             BAL06490
      DO 14 I=1,N                                                       BAL06500
   14 Y(I)=(DY(I)+4.0*DY3(I)+DY4(I))*H/6.0+Y(I)                         BAL06510
      IF ((ERRMAX.LT.TOL).AND.(ERRMAX.NE.0.0)) THEN                     BAL06520
         H = 0.9 * H * (TOL/ERRMAX) ** (1/ORD)                          BAL06530
         IF (H.GT.HMAX) H = HMAX                                        BAL06540
      ENDIF                                                             BAL06550
      RETURN                                                            BAL06560
      END                                                               BAL06570
                                                                        BAL06580
                                                                        BAL06590
                                                                        BAL06600
      SUBROUTINE SORT(F,TS,NC,NDF)                                      BAL06610
      IMPLICIT REAL*8 (A-H,O-Z)                                         BAL06620
      DIMENSION F(10,10),TS(10,10),TSMIN(5)                             BAL06630
C                              This subprogram is model independent.    BAL06640
C                              It is responsible for rearranging        BAL06650
C                              the interpolation table in descending    BAL06660
C                              order of Ts after each iterative inter-  BAL06670
C                              polation.                                BAL06680
                                                                        BAL06690
      I=1                                                               BAL06700
   52 IF (I.LE.NC-1)THEN                                                BAL06710
         I1=1                                                           BAL06720
   54    IF (I1.LE.NDF)THEN                                            BAL06730
            TSMIN(I1)=TS(I,I1)                                          BAL06740
            I1=I1+1                                                     BAL06750
```

```
            GOTO 54                                                    BAL06760
          ENDIF                                                        BAL06770
          FMIN=F(I,1)                                                  BAL06780
          JMIN=I                                                       BAL06790
          J=I+1                                                        BAL06800
 56       IF (J.LE.NC)THEN                                             BAL06810
             IF (F(J,1).LT.FMIN)THEN                                   BAL06820
                I1=1                                                   BAL06830
 58             IF (I1.LE.NDF)THEN                                     BAL06840
                   TSMIN(I1)=TS(J,I1)                                  BAL06850
                   I1=I1+1                                             BAL06860
                   GOTO 58                                             BAL06870
                ENDIF                                                  BAL06880
                JMIN=J                                                 BAL06890
                FMIN=F(J,1)                                            BAL06900
             END IF                                                    BAL06910
             J=J+1                                                     BAL06920
             GOTO 56                                                   BAL06930
          ENDIF                                                        BAL06940
          I1=1                                                         BAL06950
 62       IF (I1.LE.NDF)THEN                                           BAL06960
             TS(JMIN,I1)=TS(I,I1)                                      BAL06970
             I1=I1+1                                                   BAL06980
             GOTO 62                                                   BAL06990
          ENDIF                                                        BAL07000
          F(JMIN,1)=F(I,1)                                             BAL07010
          I1=1                                                         BAL07020
 64       IF (I1.LE.NDF)THEN                                           BAL07030
             TS(I,I1)=TSMIN(I1)                                        BAL07040
             I1=I1+1                                                   BAL07050
             GOTO 64                                                   BAL07060
          ENDIF                                                        BAL07070
          F(I,1)=FMIN                                                  BAL07080
          I=I+1                                                        BAL07090
          GOTO 52                                                      BAL07100
       ENDIF                                                           BAL07110
       RETURN                                                          BAL07120
       END                                                             BAL07130
                                                                       BAL07140
                                                                       BAL07150
       SUBROUTINE SELECT(F,TS,NC,NDIS,PDF,J,NDF)                       BAL07160
       IMPLICIT REAL*8 (A-H,O-Z)                                       BAL07170
       DIMENSION F(10,5),TS(10,10),PDF(5),FF(10,5),TTS(10,10)          BAL07180
C         The subprogram selects the useful points which will          BAL07190
C         increase the convergence rate and discards those that        BAL07200
C         will not. It also performs a monotonic check.                BAL07210
C                                                                      BAL07220
       I=1                                                             BAL07230
 72    IF (I.LE.NC)THEN                                                BAL07240
          I1=1                                                         BAL07250
 74       IF (I1.LE.NDF)THEN                                           BAL07260
             IF (PDF(I1)*F(I,I1).LT.0.0)THEN                           BAL07270
                J=I                                                    BAL07280
                K=I1                                                   BAL07290
                GOTO 2                                                 BAL07300
             END IF                                                    BAL07310
             I1=I1+1                                                   BAL07320
             GOTO 74                                                   BAL07330
          ENDIF                                                        BAL07340
```

```
        I=I+1                                                          BAL07350
        GOTO 72                                                        BAL07360
     ENDIF                                                             BAL07370
   2 NDIS=K                                                            BAL07380
     I=1                                                               BAL07390
  76 IF (I.LE.NC-2)THEN                                                BAL07400
        IF ((F(I+2,K)-F(I+1,K))*(F(I+1,K)-F(I,K)).LT.0.0)THEN          BAL07410
           IF (I.LE.J-2)THEN                                           BAL07420
              K=1                                                      BAL07430
  78          IF (K.LE.NC-I)THEN                                       BAL07440
                 I1=1                                                  BAL07450
  82             IF (I1.LE.NDF)THEN                                    BAL07460
                    F(K,I1)=F(I+K,I1)                                  BAL07470
                    I1=I1+1                                            BAL07480
                    GOTO 82                                            BAL07490
                 ENDIF                                                 BAL07500
                 TS(K,1)=TS(I+K,1)                                     BAL07510
                 K=K+1                                                 BAL07520
                 GOTO 78                                               BAL07530
              ENDIF                                                    BAL07540
              NC=NC-I                                                  BAL07550
              J=J-I                                                    BAL07560
              WRITE(6,180)                                             BAL07570
 180          FORMAT(40X,'CUT OFF HEAD FOR NON-MONOTONY')              BAL07580
              GOTO 4                                                   BAL07590
             ELSE                                                      BAL07600
                 NC=NC-1                                               BAL07610
                 WRITE(6,190)                                          BAL07620
 190             FORMAT(40X,'NON-MONOTONY ,CUT OFF TAIL')              BAL07630
                 GOTO 4                                                BAL07640
             END IF                                                    BAL07650
        END IF                                                         BAL07660
        I=I+1                                                          BAL07670
        GOTO 76                                                        BAL07680
     ENDIF                                                             BAL07690
   4 IF (NC.EQ.5)THEN                                                  BAL07700
        IF (J.LE.3)THEN                                                BAL07710
           NC=NC-1                                                     BAL07720
           WRITE(8,200)                                                BAL07730
 200       FORMAT(40X,'NC=5,CUT OFF TAIL')                             BAL07740
        ELSE                                                           BAL07750
           I=1                                                         BAL07760
  84       IF (I.LT.NC)THEN                                           BAL07770
              I1=1                                                     BAL07780
  86          IF (I1.LE.NDF)THEN                                       BAL07790
                 F(I,I1)=F(I+1,I1)                                     BAL07800
                 I1=I1+1                                               BAL07810
                 GOTO 86                                               BAL07820
              ENDIF                                                    BAL07830
              TS(I,1)=TS(I+1,1)                                        BAL07840
              I=I+1                                                    BAL07850
              GOTO 84                                                  BAL07860
           ENDIF                                                       BAL07870
           NC=NC-1                                                     BAL07880
           J=J-1                                                       BAL07890
           WRITE(8,210)                                                BAL07900
 210       FORMAT(20X,'NC=5, CUT OFF HEAD')                            BAL07910
        END IF                                                         BAL07920
     END IF                                                            BAL07930
   9 RETURN                                                            BAL07940
     END                                                               BAL07950
```

References

Aytac, Z.K., Ören, T.I. (1986). Magest: A Model-Based Advisor and Certifier for Gest Programs. In: Modelling and Simulation Methodology in the Artificial Intelligence Era, M.S. Elzas, T.I. Oren, B.P. Zeigler (Eds.) ,North Holland, pp. 299-307.

Bergsten, J.R., Fung, M.G. (1984). XMENU/E User's Guide, Kolinar Corporation, Santa Clara, California.

Birta, L. (1980). A Quasi-Parallel Method For The Simulation of Loosely Coupled Continuous Subsystems. Mathematics and Computers in Simulation XXII, 189-199.

Birta, L.G., Ören, T.I., Kettenis, D.L. (1985). A Robust Procedure For Discontinuity Handling In Continuous System Simulation. Transactions of the Society For Computer Simulation, 2:3, 189-205.

Birtwistle, G.M. (1985). In: AI, Graphics, and Simulation. ScS, San Diego, CA.

Brosilow, C.B. (1985). Modular Integration Methods for Simulation of Large Scale Dynamics Systems. In: International Seminar on Modern Methods in Dynamic Simulation of Industrial Processes, The Norwegian Institute of Technology, Trondheim, Norway.

Carver, M.B. (1978). Efficient Integration over Discontinuities in Ordinary Differential Equation Simulations. Mathematics and Computers in Simulation, XX, 190-196.

Cellier, F.E. (1979). Combined Continuous/Discrete System Simulation By Use of Digital Computers: Techniques And Tools. Doctoral Dissertation ETH No. 6483, Swiss Federal Institute of Technology, Zurich.

Davies, M.J. (1976) A Differential Model of Diabetes Mellitus. In : Mathematical Modelling, J.G. Andrews, R.R. McLone (Eds.), Butterworths & Co. (Publishers) Ltd., 1976.

Ellison, D. (1981). Efficient Automatic Integration of Ordinary Differential Equations With Discontinuities. Mathematics and Computers in Simulation, XXIII, 12-22.

Elzas, M.S. (1979). What is Needed For Robust Simulation. In: Methodology in Systems Modelling and Simulation, B.P. Zeigler, G.J. Klir, M.S. Elzas, T.I. Oren (Eds.), North Holland, Amsterdam, pp. 57-91.

Elzas, M.S. (1986). The Kinship between Artificial

Intelligence, Modelling and Simulation: An Appraisal. In: Modelling and Simulation Methodology in the Artificial Intelligence Era, M.S. Elzas, T.I. Oren, B.P.Zeigler (Eds.), North-Holland, Amsterdam, 3-13.

Elzas, M.S., Oren, T.I., Zeigler, B.P. (Eds.) (1986). Modelling and simulation methodology in the artificial intelligence era, North Holland, Amsterdam.

Fikes, R., Kehler, T. (1985). The Role of Frame-Based Representation in Reasoning. Communications of the ACM, 28:9.

Gear, C.W. (1971). The Automtic Integration of Ordinary Differential Equations. Communictions of the ACM, 14:3, 176-190.

Gear, C.W. (1981). Numerical Solution of Ordinary Differential Equations: Is there Anything Left to do? SIAM Review, 23:1, 10-24.

Gear, C.W., Osterby, O. (1984). Solving Ordinary Differential Equations With Discontinuities. ACM TOMS 10, pp. 23-44.

Gerald, C.F., Wheatley, P.O. (1984). Applied Numerical Analysis, Addison Wesley Publishing Company.

Goldstein, R.A., Harris, W.F. (1973) Serendipity - A Watershed Level Simulation Model of Tree Biomass Dynamics. In : Summer Computer Simulation Conference, 1973, pp. 691-696.

Guariso, G., Hitz, M., Werthner, H. (1987). An Intelligent Simulation Model Generator. In: Proceedings of The Europeon Simulation Multi-Conference Vienna, Austria.

Harmon, P., King, D. (1985). Expert Systems: - Artificial Intelligence in Business. John Wiley & Sons, Inc., New York.

Henricksen, J. (1983). The Integrated Simulation Environment, Operations Research. Operations Research, 31:6.

Holmes, W.M. (Eds.) (1985). Artificial Intelligence and Simulation, SCS, San Diego, CA.

Hornbeck, R.W. (1975). Numerical Methods. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Hull, T.E. (1980). Comparison of Algorithms For Initial Value Problems. In: Computational Techniques For Ordinary Differential Equations, I. Gladwell, D.K.

Sayers (Eds.) Academic Press, London, New York, pp. 143-163.

Hull, T.E. et al. (1972). Comparing Numerical Methods for Ordinary Differential Equations. SIAM Journal of Numerical Analysis, 9, 603-607.

I.S.S.C. (1982). Disspla User's Manual. San Diego, CA 92121-1698.

Jackson, P. (1985). Introduction To Expert Systems. Addison-Wesley Publishing Company, Inc.

Kerchoffs, E.J.H., Vansteenkiste, G., Zeigler, B.P. (Eds.)(1986). Artificial Intelligence applied to simulation, SCS, San Diego, CA.

Kettenis, D.L. (1986). Knowledge-Based Model Storage and Retrieval: Problems and Possibilities. In: Modelling and Simulation Methodology in the Artificial Intelligence Era, M.S. Elzas, T.I. Oren, B.P. Zeigler (Eds.), North-Holland, Amsterdam, pp. 101-111.

Khoshnevis, B., Austin, W.M. (1986). An intelligent interface for systems dynamics modelling. In: Simulation And AI, P.A. Luker, G. Birtwistle (Eds.), SCS, San Diego, CA, 18:3, 81-87.

Lambert, J.D. (1973). Computational Methods in Ordinary Diferential Equations. John Wiley, New York.

Langer, K., Schmidt, B. (1987). The Simulation System SIMPLEX-II: A Model Construction And Experimentation Environment. In: Proceedings of The European Simulation Multi-Conference, Vienna, Austria.

Lehmann, A. (1987). Taxonomy and Applications of Expert Systems in Simulation. International Symposium on AI, Expert Systems and Languages in Modelling and Simulation, Barcelona, Spain, June 1987.

Luker, R.A., Adelsberger, H.H. (Eds.) (1986). Intelligent simulation environments. SCS, San Diego, CA.

Luker, R.A., Birtwistle, G. (Eds.) (1987). Simulation and AI, SCS, San Diego, CA.

Ma, J. Lui, W. (1986). Software Requirements of the Simulation of Discontinuous Systems. In: Proceedings of Bejing System Simulation Symposium, pp. 122-125.

Nielsen, N.R. (1986). The impact of using AI-based techniques in a control system simulator. In: Simulation And AI, P.A. Luker, G. Birtwistle (Eds.),

SCS. 18:3, 72-81.

Ören, T.I. (1984a). "Quality Assurance in Modelling and Simulation: A Taxonomy." In: Simulation and Model-Based Methodologies: An Integrative View, T.I. Ören, B.P. Zeigler, M.S. Elzas (Eds.), Springer Verlag, Heidelberg, W. Germany, pp. 477-517.

Ören, T.I. (1984b). Gest: A Modelling and Simulation Language Based on System Theoretic Concepts. In: Simulation and Model Based Methodologies: An Integrative View, T.I. Ören, B.P. Zeigler, M.S. Elzas (Eds.), Springer Verlag, Heidelberg, W. Germany, pp. 281-335.

Ören, T.I. (1984c). Model-Based Activities: A Paradigm Shift. In: Simulation and Model Based Methodologies: An Integrative View, T.I. Ören, B.P. Zeigler, M.S. Elzas (Eds.), Springer Verlag, Heidelberg, W. Germany, pp. 3-40.

Ören, T.I. (1986a). Artificial Intelligence and Simulation. Artificial Intelligence applied to Simulation, SCS, San Diego, CA.

Ören, T.I. (1986b). Knowledge bases for an advanced simulation environment. In: Intelligent Simulation Environments, P.A. Luker, H.H. Adelsberger (Eds.) SCS, San Diego, CA, pp. 16-22.

Ören, T.I. (1986c). Artificial Intelligence in Quality Assurance of Simulation Studies. In: Modelling and Simulation Methodology in the Artificial Intelligence Era, M.S. Elzas, T.I. Ören, B.P. Zeigler (Eds.), North-Holland, Amsterdam.

Ören, T.I., Ma, J. (1986). An Adaptive Order Discontinuity Algorithm For Simulation of Ordinary Differential Equations. Proceedings JSST Conference on Recent Advances in Simulation of Complex Systems, Tokyo, Japan, pp. 35-38.

Ören, T.I. (1987a). Quality Assurance Paradigms For Artificial Intelligence In Modelling and Simulation. Simulation, 48:4, 149-151.

Ören, T.I. (1987b). Artificial Intelligence In Simulation. In: Proceedings of Measurement, Modelling, and Evaluation of Computer Systems, Erlanger, West Germany, 1987, pp. 375-388.

Ören, T.I. (1987c). Model Update: A Model Specification Formalism With A Generalized View of Discontinuity. Summer Computer Simulation Conference, Montreal, Canada,

July 1987.

Ören, T.I., Sheng, G. (1988 In Press). Semantic Rules and Facts for an Expert Modelling and Simulation System. In: Preprints of IMACS Congress, Paris, France, July 1988.

Palusinski, O.A. (1985). Simulation of Dynamic Systems Using Multi-Rate Integration Techniques. Transactions of the Society For Computer Simulation, 2:4, 257-273.

Reddy, R., Fox, M.S. (1986). The Knowledge-Based Simulation System. IEEE Software, 3, pp. 26-37.

Reddy, R. (1987). Epistemology of Knowledge-Based Simulation. Simulation, 48:4, 162-166.

Rich, E., (1984). Natural Language Interfaces, Computer, pp. 39-47.

Shampine, L.F. (1980). What Everyone Solving Differential Equations Numerically Should Know. In: Computational Techniques For Ordinary Differential Equations, I. Gladwell, D.K. Sayers, (Eds.) Academic Press, London, New York, pp. 1-19.

Shampine, L.F. Gordon, M.K. (1975). Computer Solution of Ordinary Differential Equations: the initial value problem. W.H. Freeman & Co., San Francisco.

Shampine, L.F. Gordon, M.K. Wisniewski, J.A. (1980). Variable Order Runge-Kutta Codes. In: Computational Techniques For Ordinary Differential Equations, I. Gladwell, D.K. Sayers, Academic Press, London, New York, 83-103.

Shannon, R.E. Mayer, R., Adelsberger, H.H. (1985). Expert Systems and Simulation. Simulation, 44:6, 275-284.

Wagner, H.M. (1975). Principles of Operations Research. Prentice-Hall, NJ.

Widman, L.A., Loparo, K.A., Helman, d.H. (Eds) (1987). Artificial Intelligence, Simulation, and Modelling, D. Reidel publishing company.

Wymore, A.W. (1967). A Mathematical Theory of Systems Engineering: The Elements. John Wiley, New York.

Xiong, G., Song, A. (1986). An Expert System For Dynamic Systems Simulation. In: AI Applied To Simulation, E.G.H. Kerkhoffs, G.C. Vansteenkiste, B.P. Zeigler (Eds.), Simulation Series, 18:1(Feb.), SCS, San Diego,

CA, pp. 106-110.