RESEARCH ON AN EXPERT SYSTEM FOR DATABASE OPERATION OF SIMULATION/EMULATION MATH MODELS

Phase I Results

VOLUME I

Prepared by:

K. Kawamura, G.O. Beale, J.D. Schaffer,
B.-J. Hsieh, S. Padalkar, J.J. Rodriguez-Moscoso

Center for Intelligent Systems
Vanderbilt University
P.O. Box 1804, Station B
Nashville, Tennessee 37235

This work was performed for NASA's George C. Marshall Space Flight Center under contract NAS8-36285.

AUGUST 9, 1985

RESEARCH ON AN EXPERT SYSTEM FOR DATABASE OPERATION OF SIMULATION/EMULATION MATH MODELS

Phase I Results

VOLUME I

Prepared by:

K. Kawamura, G.O. Beale, J.D. Schaffer,
B.-J. Hsieh, S. Padalkar, J.J. Rodriguez-Moscoso

Center for Intelligent Systems
Vanderbilt University
P.O. Box 1804, Station B
Nashville, Tennessee 37235

This work was performed for NASA's George C. Marshall Space Flight Center under contract NAS8-36285.

AUGUST 9, 1985

VOLUME I

TABLE OF CONTENTS

	Page
SUMMARY	i
INTRODUCTION	ii
REPORT ORGANIZATION	iii
PART I. BACKGROUND	
1. Spacecraft Attitude Control Problem	1.1
2. Coupling Symbolic Processing and Numerical Computation	2.1
PART II. EXPERT SIMULATION SYSTEM	
3. System Definition	3.1
3.1 Operating System Environment	
3.2 Inference Engine (GENIE)	
3.3 Expert System (NESS)	
4. NESS Implementation	4.1
4.1 NESS Overview	
4.2 NESS Architecture	
5. NESS Knowledge Description	5.1
5.1 Knowledge Acquisition	
5.2 Knowledge Overview	
PART III. GENERIC SIMULATION MODEL	
6. Model Definition	6.1
6.1 Model Overview	
6.2 Prototype 0	
6.3 Prototype I	
7. Model Simulation	7.1
7.1 Time-Domain Analysis	
7.2 Frequency-Domain Analysis	
Pseudo Open-Loop Response	
PART IV. SYSTEM RUNS, VERIFICATION & EVALUATION	
8. System Runs	8.1
9. System Verification	9.1
10. System Evaluation	10.1
PART V. CONCLUSIONS	
11. Conclusions	11.1

SUMMARY

SUMMARY

This report describes the results of the first phase of a project, "Research on an Expert System for Database Operation of Simulation/Emulation Math Models," for NASA's George C. Marshall Space Flight Center. The project was designed to bring techniques from artificial intelligence (AI) to bear on task domains of interest to NASA Marshall Space Flight Center. One such domain is simulation of spacecraft attitude control systems.

During the 7 months of this phase, two related software systems were developed and delivered to NASA. One was a generic simulation model for spacecraft attitude control, written in FORTRAN. The second was an expert system which understands the usage of a class of spacecraft attitude control simulation software and can assist the user in running the software. This "NASA Expert Simulation System" (NESS), written in LISP, contains general knowledge about digital simulation, specific knowledge about the simulation software, and self-knowledge.

These two prototype software systems constitute a "proof of principle" demonstration of the applicability of AI technology to a task of interest to NASA. They also represent software tools which can be incrementally extended until they can perform genuinely useful work.

INTRODUCTION

INTRODUCTION

Current NASA planning to begin in 1986 the design and development of a low earth-orbit Space Station poses a unique opportunity for development of an expert system for coupling symbolic processing and numerical computation.

Development of the Space Station System will require close coordination between system designers from NASA, the aerospace industry and others. An intelligent system can provide design coordination for the diverse teams expected to participate in the development of the Space Station by handling the access and data requirements to and from various simulation models and by assisting the user in running them.

Most current expert systems focus on the symbolic processing and inference mechanisms of artificial intelligence (AI) and are not well suited to deal with engineering problems requiring dynamic simulation models. Therefore, a need exists for coupling symbolic reasoning with conventional mathematical algorithms to provide a basis for multilevel expert systems.

Recognizing such a need, NASA's George C. Marshall Space Flight Center awarded a contract to Vanderbilt University Center for Intelligent Systems to develop an expert system to run a class of spacecraft simulation programs. This contract has the following long-range objectives:

1. To create an expert system that can assist the user in running a variety of the simulation programs employed in the development of the Space Station.

2. To create an expert system that understands the usage of a NASA-supplied database management system and that can assist the user in the operation of various features of the database system.

As a first step toward development of such an intelligent system, an expert system was developed, called NESS (NASA Expert Simulation System), which understands the usage of a prototype spacecraft attitude control simulation model and can assist the user in running the simulation model. NESS was built using a general inference engine called GENIE (GENeric Inference Engine), written in Franzlisp. The simulation software, written in Fortran-77, implements a generic spacecraft attitude control system and includes modules such as Controller, Body Dynamics, Quaternion Transformation, and Integrator.

REPORT ORGANIZATION

REPORT ORGANIZATION

This document details the work performed on Phase I of Contract NAS8-36285, "Research on an Expert System for Data Base Operation of Simulation/Emulation Math Models," for the Marshall Space Flight Center of the National Aeronautics and Space Administration. It was prepared by the Vanderbilt University Center for Intelligent Systems in accordance with the contract's Statement of Work.

This document consists of Summary, Introduction and Report Organization sections, five main parts, references and three appendices. Of the five main parts, Part I provides background information on the spacecraft attitude control problem and the coupling of symbolic processing and numerical computation. Part II gives an overview of the expert simulation system NESS. Part III describes the generic simulation model and offers a discussion of the time- and frequency-domain analyses it can perform. Part IV contains the results of sample runs, system verification and system evaluation. Part V provides conclusions. Part I through V make up Volume I of the Phase I Report.

Appendix A is a user's manual for NESS developed under this contract and is contained in Volume II. Appendix B is a programmer's manual for NESS. Appendix C contains a discussion on Quaternions. Appendix D provides a listing for NESS and a simulation model listing.

PART I. BACKGROUND

1. SPACECRAFT ATTITUDE CONTROL PROBLEM

The function of a spacecraft attitude control system is to maneuver a space vehicle into a certain orientation, defined by a reference vector, and to maintain that orientation over an extended period of time. As an example of this attitude control, consider the pointing control system for the Space Telescope (Nurre and Dougherty). The control system must maneuver the telescope through a 90 degree arc in less than 20 minutes, and then maintain a stable line-of-sight to within 0.007 arc-seconds for 24 hours. Thus, the control system must be designed to maneuver through a large change in direction and then track the vehicle's position about a constant direction. The vehicle's dynamics would be represented by non-linear differential equations during the maneuvering mode; in the tracking mode, the equations could be linearized about the desired operating point. In Phase I of this research project, only the simulation of the vehicle and control system during the tracking mode is considered.

The commanded inputs to the control system would generally be angular position. Both angular position and angular rate would be measured by star trackers and rate gyros, and these measurements would be available to the control system. The torque required to accomplish the maneuvers would be provided by a set of control moment gyros (CMG's). Generally, redundancy in sensors and actuators would be a design feature of the control system. For example, four sensors could be positioned to measure some variable in three-dimensional space such that any three of the sensors would provide linearly independent measurements. With this type of configuration, all four sensors could be used and consistency checks made on the measurements. If any one sensor failed, the remaining three could provide complete coverage of the desired variable. Figure 1.1 is a simplified illustration of the pointing control system for the Space Telescope. The controller, control moment gyros, and rate and position sensors mentioned above can easily be identified in the figure. The Fine Guidance Sensor block is used in different ways for the different modes of searching for a new target, coarse tracking of the target, and finally maintaining an attitude locked onto the target.

One factor which makes the control of a space vehicle more involved than the traditional position control problem of classical control theory is the need for several coordinate frames, and the fact that these coordinate frames move with respect to time. Some of these various coordinate frames specify the actual vehicle orientation, orientation of the target with respect to the vehicle position, and alignment of the guidance sensors. Other coordinate frames define the orbital and magnetic vertical orientations of the vehicle and the plane of the vehicle orbit. Some of these coordinate frames have their origin attached to the vehicle, others have their origin on the Earth. However, all of the coordinate frames are specified relative to a single inertial coordinate frame having its origin at the center of the Earth. Often, the coordinate frames are defined such that pairs of frames have one common axis. In the case of the Space Telescope, examples of coordinate frames which have common axes are the orbital and magnetic local vertical coordinate frames, the orbital local vertical and orbital coordinate frames, and the equatorial inertial and equatorial earth-fixed coordinate frames. Figures 1.2 and 1.3 illustrate some of the coordinate frames used on the Space Telescope (Kennel, 1976).

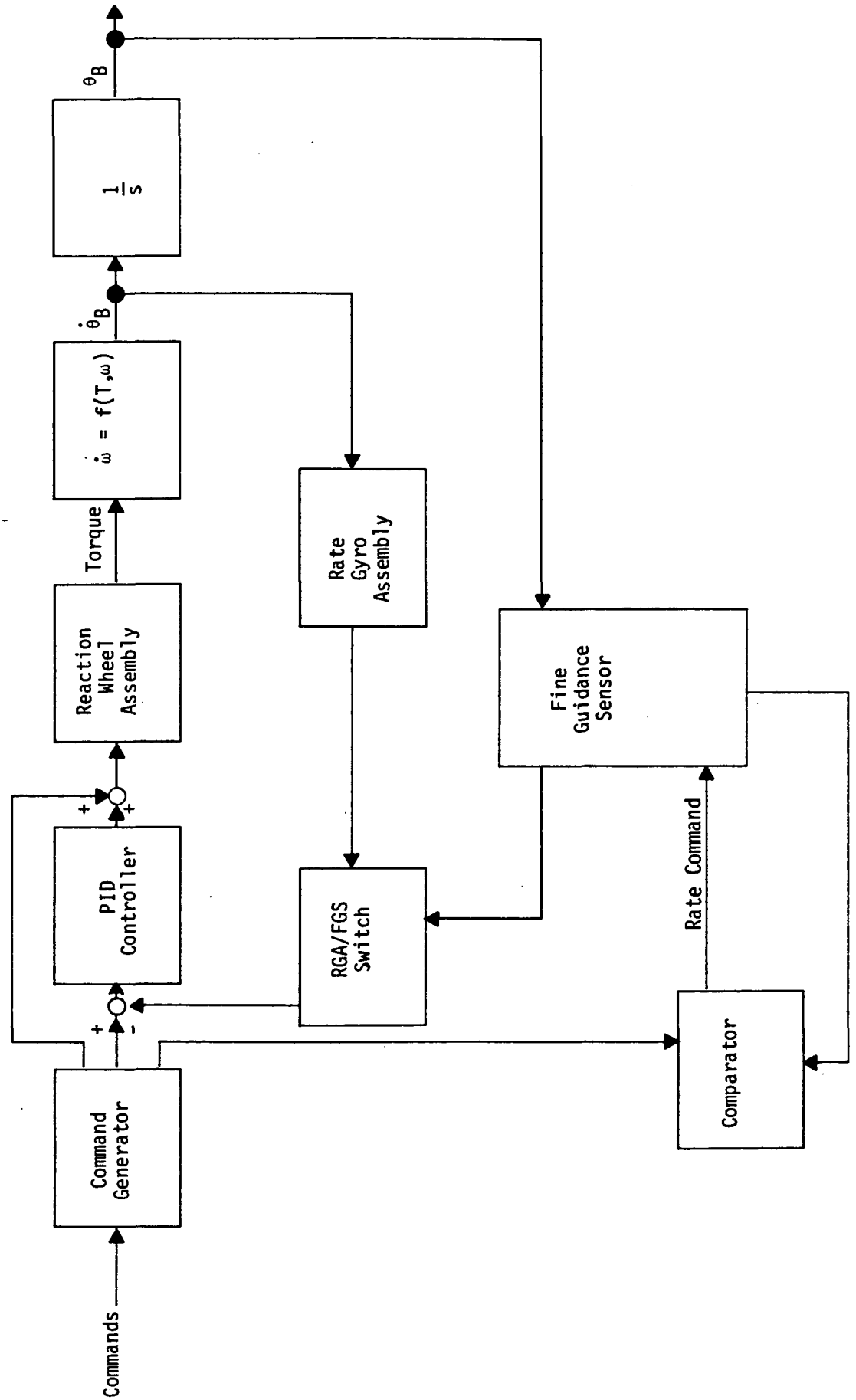


Figure 1.1. Simplified Attitude Control System

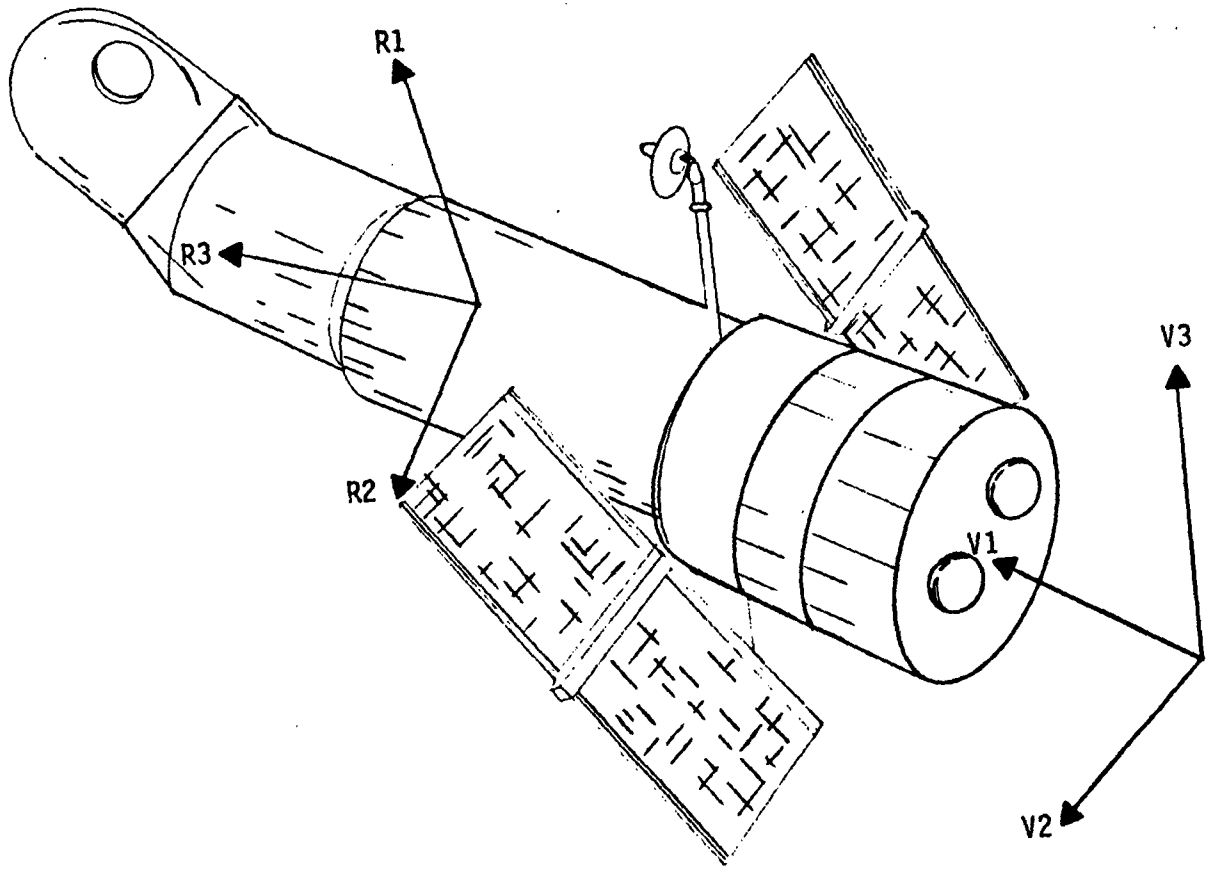


Figure 1.2: Vehicle Control Axes and Attitude Reference Coordinate Frames

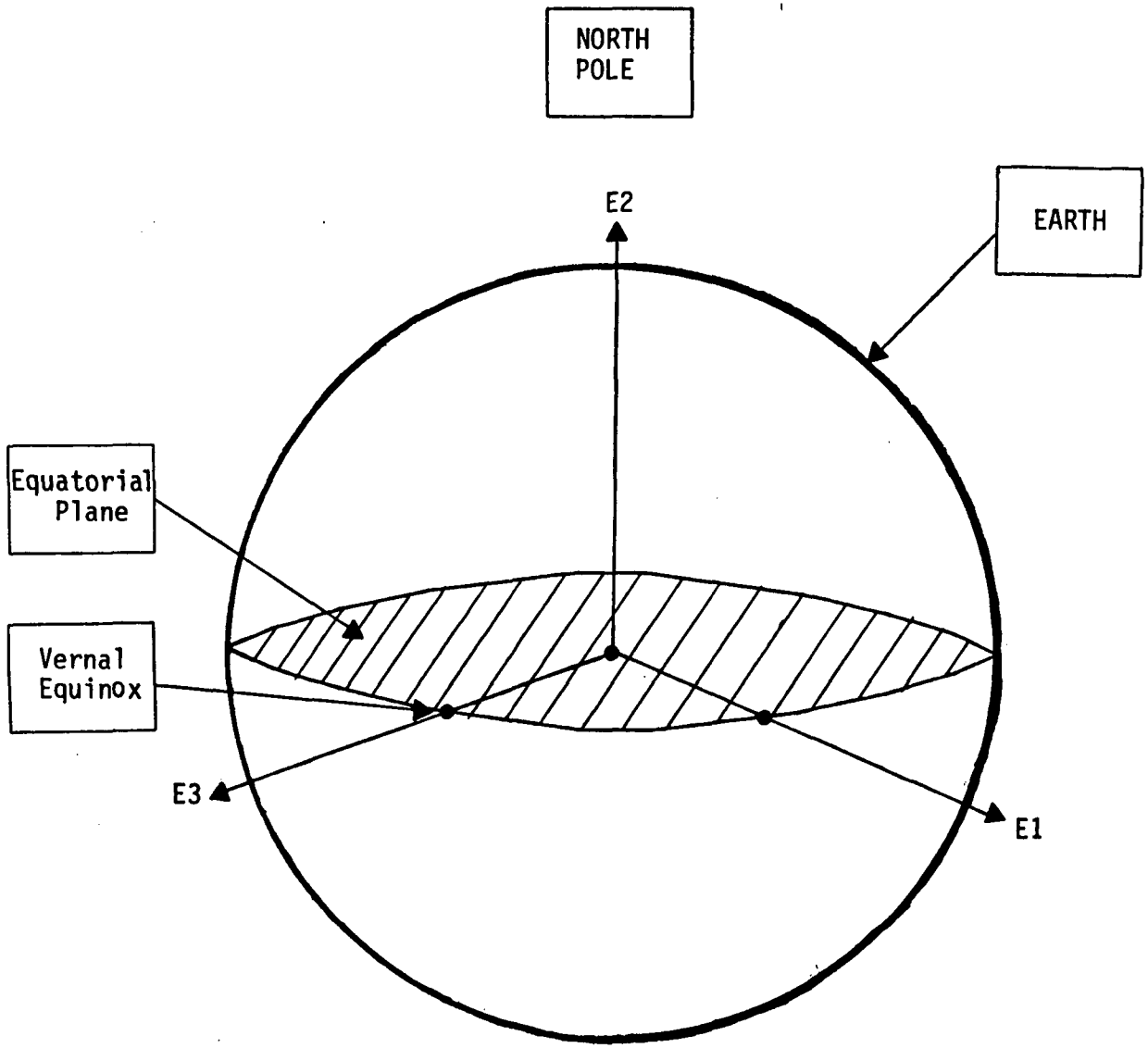


Figure 1.3: Equatorial Inertial Coordinate Frame

Figure 1.2 shows the vehicle control axes frame and attitude reference coordinate frame for arbitrary vehicle and reference directions. Figure 1.3 shows the equatorial inertial coordinate frame.

Transformations are possible between these various coordinate frames (Brady et al., 1982; Paul, 1981). A matrix can be defined which can multiply a vector in one coordinate frame to convert it into the equivalent vector in a second coordinate frame. Premultiplication of the vector by the matrix corresponds to having all rotations relative to the original coordinate frame, while postmultiplication corresponds to having the rotations specified in the current coordinate frame. The elements of the matrix can be functions of the Euler angle rotations necessary to align the axes of the two coordinate frames. Transformation from one coordinate frame to another can be done in one step or as a series of operations involving intermediate coordinate frames. Transformation matrices can also be defined in terms of Roll-Pitch-Yaw angles between the coordinate frames or in terms of a rotation about a specified vector. This last approach is normally applied to spacecraft attitude control problems using the concept of Quaternions. A Quaternion is a four element array; three of the elements define the vector about which the rotation is made, and the fourth element is the angle of rotation. The Quaternion is described in detail in Appendix C, and its use in this research project is discussed in the section on the Generic Simulation.

A simple space attitude control system would have a minimum of three (3) coordinate frames. These coordinate frames would represent the inertial coordinate system, the actual vehicle orientation, and the target reference direction. Zero position error is achieved when the reference and vehicle coordinate frames are parallel to each other. These three frames are used in the generic simulation for this research project. The function of the attitude control system is to maneuver the space vehicle until its coordinate frame becomes parallel to the reference coordinate frame, and then to maintain that orientation until new reference direction commands are given. The differences between actual and commanded angular positions and actual and commanded angular rates would be used by the control system as the error signals to the actuators. These signals would command torque from the actuators about an axis to force the errors to zero. This amounts to determining the transformation matrix between the current vehicle orientation and that of the reference vector, and determining the control signals necessary to physically implement that transformation matrix.

REFERENCES

Brady, et al., Robot Motion: Planning and Control, MIT Press, Cambridge, 1982.

Kennel, H.F., "Space Telescope Coordinate Systems, Symbols, and Nomenclature Definitions," NASA Technical Memorandum, TMX-73343, September, 1976.

Nurre, G.S. and Dougherty, H.J., "The Pointing System for Space Telescope."

Nurre, G.S., Dougherty, H., and Tompetrini, K., "Space Telescope Pointing Control System."

Paul, R.P., Robot Manipulators: Mathematics, Programming, and Control, MIT Press, 1981.

2. COUPLING SYMBOLIC AND NUMERIC COMPUTATION

One of the major design decisions of this project was to maintain a clear separation between the generic simulation software system, which performs the numeric computations, and the expert system which performs symbolic reasoning computations. This parallels the situation in which a human expert sets out to perform a numeric simulation experiment. The human expert, using his knowledge of the system to be modeled and the characteristics of the numerical software at his disposal, makes decisions about how to instantiate his experiment. These decisions are then frequently implemented by creating an input file to be read by the general purpose simulation software. This file contains parameters describing his system and switches which inform the simulation system of the options selected by the expert user. The user then issues a command to the operating system to run the simulation program. If it runs without error, he then examines the output file(s) and interprets the results.

Following this model, an expert system (NESS) was designed as a software system separate from the generic simulation software. Figure 2.1 shows the interaction of these two systems schematically. Each system was written in the programming language most natural to it. The generic simulation software was written in FORTRAN, following years of traditional engineering practice, and NESS was built using a general inference engine (GENIE [Sandell, 1984]) written in LISP, following current AI practice.

The knowledge-base of NESS contains three types of knowledge: general knowledge of spacecraft attitude control simulation; knowledge of the input parameters and their formats required by the generic simulation software; and self (or internal control) knowledge.

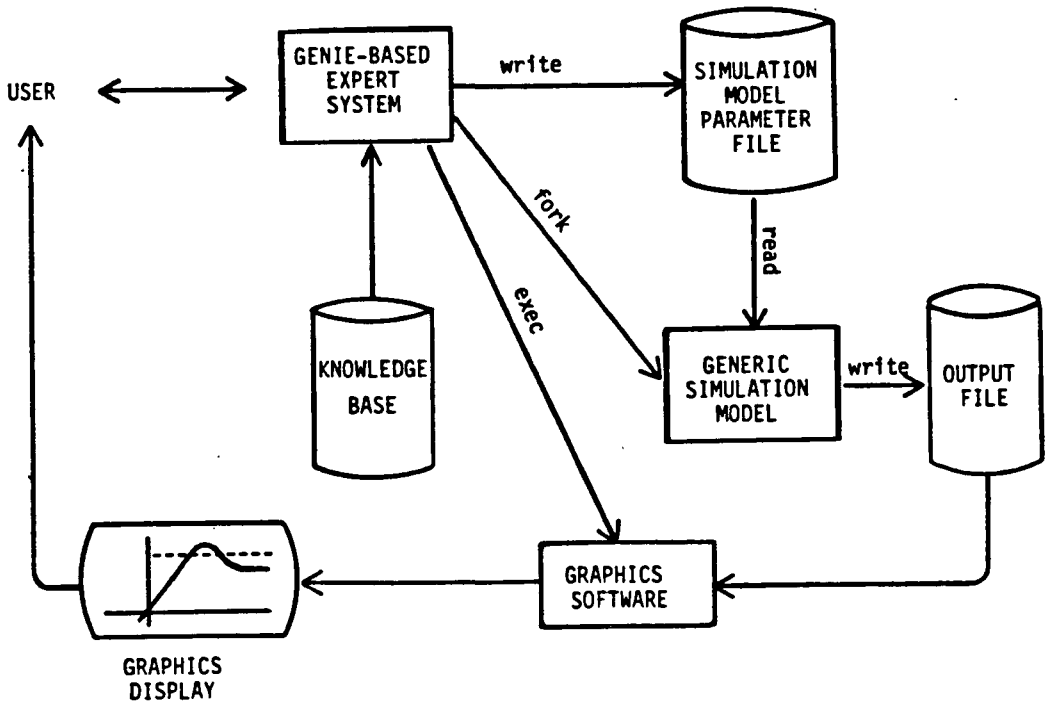


Figure 2.1.

The user-interaction scenario is envisioned as follows. The user invokes NESS, which queries him about his system and the experiment he wishes to perform. This interaction should avoid requiring the user to specify all the low-level parameters. Rather, it should concentrate on the major engineering decisions required to get an answer to his major questions. The setting of the low-level parameters should be inferred and performed by NESS using its knowledge. However, for an expert user, these low level details must remain accessible so that NESS's recommendations can be overridden, if desired. After gaining sufficient information to specify a complete experiment, NESS runs the simulation and checks for run-time error messages from the operating system. This process uses the FRANZ LISP ability to call UNIX systems functions to fork descendent processes and is described in more detail in Appendix B. Failures should be rare if the knowledge is sufficient to avoid setting up experiments which can be predicted to fail.

NESS would then examine the output file(s) created by the simulation program and interpret them for the user in light of his major questions. This may involve exhibiting plots of system responses and comments on the stability of the proposed system design.

The prototype implementation of NESS described in this report cannot be considered a complete realization of this vision. Rather, it should be considered a "proof of principle" demonstration and a foundation upon which to build the full capabilities. True to the theory of good expert system design, extensions to the existing knowledge-base should be relatively easy to implement. Indeed, the current system was developed from an initial knowledge-base of only two rules. All of the capabilities described in the rest of this report were added to that original system, one by one, thereby demonstrating the soundness of the overall approach and its consistency with the theory of good expert system design. Some future expansions will be described in Part V of this report.

One additional point should be made concerning the coupling of numeric and symbolic computation in this system. Occasionally, a simulation expert may need to perform some quick calculations before he is ready to run an experiment, the calculation of the eigenvalues of the proposed system matrix is an example. Some system behaviors may be anticipated from this information, and so NESS is able to make those calculations. Since numeric computation is not a strong aspect of the LISP programming language and well-tested FORTRAN routines for computing eigenvalues are available, this operation was effected by directly linking a compiled FORTRAN subroutine with a LISP demon which NESS calls when needed. This design continues the parallel with human practice. Engineers needing eigenvalues or other complex calculations, usually use a pocket calculator (often programmable) or an easily accessible computer.

REFERENCES

Sandell, H.S.H., A Knowledge Engineering Tool for Creating Frame- and Rule-Based Expert Systems, Ph.D. Dissertation, Vanderbilt University, Nashville, TN, 1984.

PART II. EXPERT SIMULATION SYSTEM

3. SYSTEM DEFINITION

3.1 OPERATING SYSTEM ENVIRONMENT

The prototype systems developed for this project use several operating system features specific to the Vanderbilt VAX system. The expert system (NESS) was designed using the GENIE inference engine (Sandell), which in turn was written in FRANZ LISP (Foderaro). NESS contains LISP code, which makes calls to functions provided by the EUNICE version 3.2 operating system. EUNICE, in turn, operates under version 3.5 of the VAX/VMS operating system. The generic simulation model was written in FORTRAN and runs under the supervision of EUNICE. The various system environments and their relationships are illustrated in Figure 3.1.

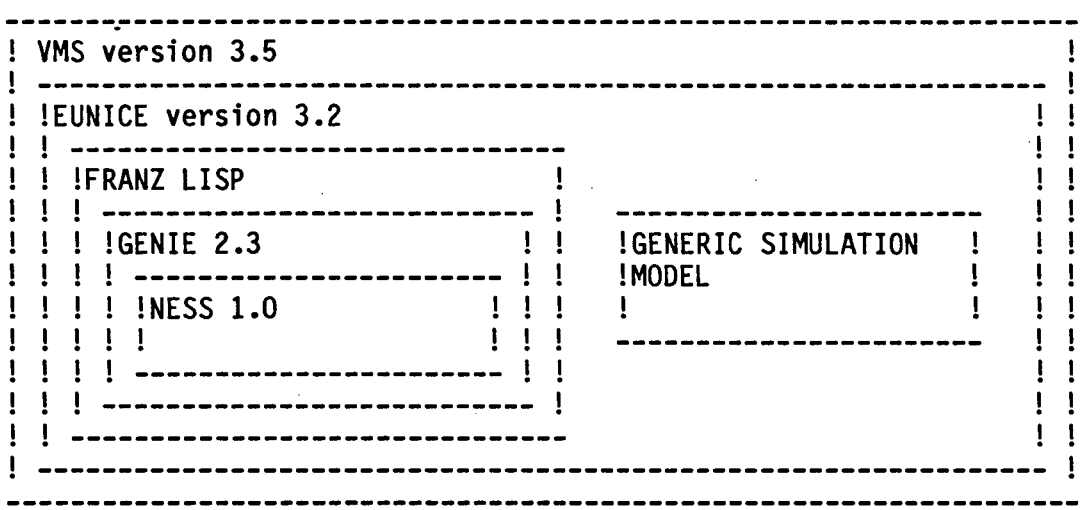


Figure 3.1

3.2 GENERAL INFERENCE ENGINE

The expert system was designed using GENIE, a general inference engine developed at Vanderbilt University. GENIE allows a user to create an expert system with the help of an interactive process. GENIE also acts as an interpreter for the user-created expert system. The main features of GENIE are summarized below.

Frames

GENIE stores all facts about the domain in a data structure called a frame. A frame can be tailored to behave like a variable, a matrix, a set of variables, facts about the domain, or almost anything else the user has in mind. A frame in GENIE is represented as follows:

```
(frame_name
          (slot1_name (value1))
          (slot2_name (value2))
          :
          :
          (slotn_name (valuen)))
```

Note that a slot may itself be a frame, and this may be repeated to any depth.

Rules

Rules are used by GENIE to store knowledge from the domain. Rules are of the form

IF condition THEN action

where the condition as well as the action may be highly complex in nature. A rule, when evaluated, behaves as follows: If the condition is true, then the action is performed; if not, nothing is done.

Rule-Bases

A collection of rules is called a rule-base. The GENIE rule interpreter can apply a rule-base with either of two control strategies, forward-chaining or backward-chaining.

Forward-chaining

In this mode all rules in a rule-base are evaluated in a sequential manner until no rule's condition holds true.

Backward-chaining

In this mode we first make a hypothesis about our domain or set a goal to be satisfied. GENIE then evaluates all rules referring to our goal in order to satisfy it.

Menu-Inputs

GENIE provides a menu-input facility to gather information from the user. The user examines the a menu of various choices that appears on the screen and then indicates his choice.

Agendas

Control information about driving the expert system is in special frames called agendas. GENIE uses the information in these frames to run the expert system.

NESS can utilize all the facilities mentioned above. Please refer to the GENIE User's Manual for more detailed information about them.

A description of the salient portions of the expert system follows.

3.3 NESS (NASA EXPERT SIMULATION SYSTEM)

The main function of NESS is to gather initial values of parameters necessary for the simulation experiment, organize them in the correct format in a file, and run the FORTRAN-based simulation program, which gets its input data from the file created by NESS. In addition, NESS allows the user to look at the parameter values and change them if necessary before running the simulation program. After the simulation has been run, NESS allows the user to observe the outputs generated by the simulation program. NESS also has the capability of loading in initial values of parameters from a file and is able to store current initial values of parameters in a file for future reference.

The primary knowledge in NESS is concerned with gathering input data for the simulation experiment. NESS first asks the user for initial values of some parameters. To obtain other necessary values, it then asks the user questions from which it can infer those particular values. This is done in a systematic manner as follows:

a) NESS gathers all the data required to define the system to be simulated. This includes getting values for the inertial and controller matrices, initializing the Quaternion module and selecting a method of integration.

b) NESS asks for the type of response to be obtained from the system. It does so by asking the user to choose either STEP or FREQUENCY response.

c) NESS then completes the set of parameters that are required to run the simulation experiment.

This process is illustrated in Figure 3.2.

After all parameter values have been obtained, the simulation program can be executed. The user may observe the results of simulation following its execution.

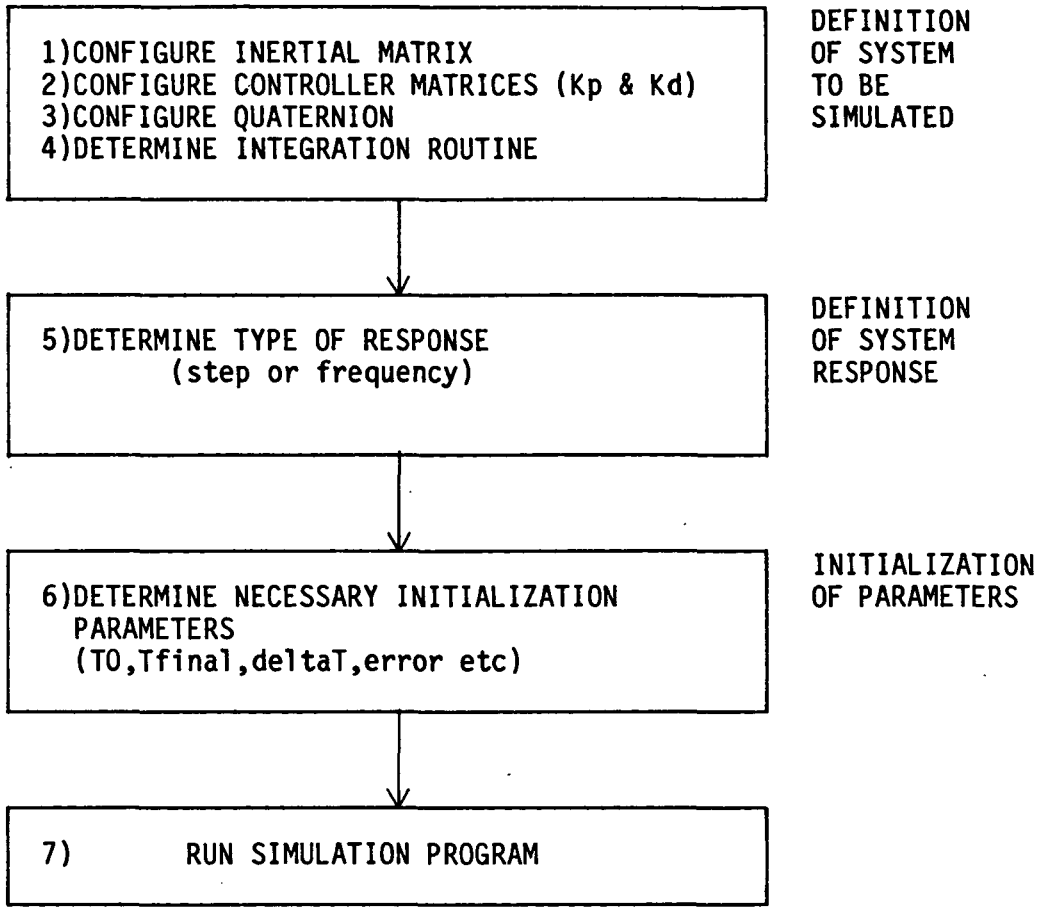

1)CONFIGURE INERTIAL MATRIX
2)CONFIGURE CONTROLLER MATRICES (Kp & Kd)
3)CONFIGURE QUATERNION
4)DETERMINE INTEGRATION ROUTINE
DEFINITION OF SYSTEM TO BE SIMULATED
5)DETERMINE TYPE OF RESPONSE
(step or frequency)
DEFINITION OF SYSTEM RESPONSE
6)DETERMINE NECESSARY INITIALIZATION PARAMETERS
(TO,Tfinal,deltaT,error etc)
INITIALIZATION OF PARAMETERS
7) RUN SIMULATION PROGRAM

Figure 3.2

4. NESS IMPLEMENTATION

4.1 NESS OVERVIEW

When the user starts NESS, the screen displays the following menu.

1) Exit to GENIE
2) Set up initial parameter values
3) Run simulation program
4) Display current parameter values required for simulation
5) Display outputs generated by simulation
6) Change initial parameter values required for simulation
7) Set up initial parameter values to default values
8) Store current parameter values in a disk file.

These represent the functions that NESS can perform during a session. The user is asked to enter a choice and NESS executes the function. After the function has been executed, NESS returns to the menu, thus allowing the user to choose another function. This is known as the TOP LEVEL of NESS.

Explanation of the Above Functions

1) Exit to GENIE

This function allows the user to halt NESS and go to the GENIE environment.

2) Set up initial parameter values

This is the main function of NESS, i.e., collecting initial values of parameters necessary for the simulation experiment. This function is performed through the use of a rule-base named value_input_rb, which is driven in the backward-chained mode.

3) Run simulation program

This function allows the user to run the FORTRAN-based simulation program provided the initial values of the necessary parameters have already been specified. Using a backward-chained rule-base called run_rb, this function creates the data file required by the simulation program and runs it using a FRANZ LISP function called "process," which allows a program written in a foreign language like FORTRAN to be executed in the EUNICE environment.

4) Display current parameter values required for simulation

This function displays a menu of all parameters known to NESS. The user can choose the parameters of interest with a forward-chained rule-base disp_init_val_rb, which displays the values of the chosen parameters. In addition, the user can go back to the TOP LEVEL by means of an option present on the menu at this level.

5) Display outputs generated by simulation

This function displays a menu of the outputs generated by the simulation program. The user can see any of the outputs simply by indicating a choice from the menu. Another choice provided allows the user to return to the TOP LEVEL of NESS.

6) Change initial parameter values required for simulation

This function displays the same menu displayed by Function 4, i.e., a menu of all parameters known to NESS. From this, the user can choose the parameters whose values he wants changed. This is done using a forward-chained rule-base named change_param_rb. Once again, using an option present on the menu, the user can go back to the TOP LEVEL.

7) Set initial parameter values to default values

In this function the user is prompted for the file containing the default values of parameters; NESS then loads in these values into its database. At present, the file containing the default values of parameters has to be one that has been stored, using Function 8 (store current parameter values in a disk file).

8) Store current parameter values in a disk file

This function prompts the user for the name of a file in which the current values of parameters should be stored.

At this point it should be remembered that certain functions of the TOP LEVEL menu, such as 'set up initial parameter values,' 'run simulation program,' 'set up default values of all parameters,' and 'store current parameter values in a disk file,' return to the TOP LEVEL immediately after their work is finished. The other three functions (excluding Function 1 'Exit to GENIE') remain in their particular mode until the user chooses to return to the TOP LEVEL.

4.2 NESS ARCHITECTURE

As noted earlier, the main constituents of GENIE are data frames, agenda frames, menu-input and rule-bases. The agenda frames provide a control strategy for NESS, the menu-input provides a means of entering a user's choice from a menu of options, and the data frames are used to store data. The rule-bases form a primary source of knowledge and can draw inferences and perform other useful work such as gathering information from the user.

The two basic structures that occur throughout NESS are:

1) An agenda frame followed by a menu-input, followed by one or more rule-bases. This structure is used when some input is required from the user. Input is through menu-input. The rule-base then uses the input given by the user to perform the necessary functions.

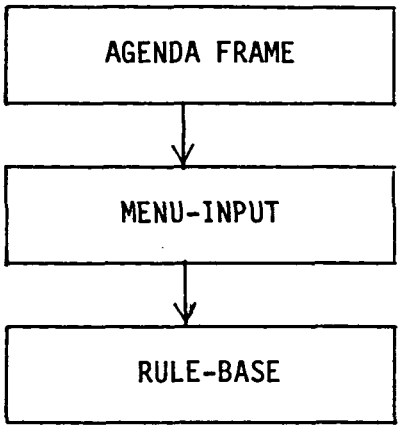


Figure 4.1

2) An agenda frame followed by a rule-base. In this structure inputs may or may not be required from the user. If any inputs are required, they can be obtained through the rule-base. Rules may ask the user for information they require, if it is not already known.

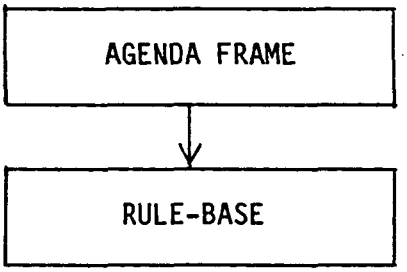


Figure 4.2

The difference between the two structures lies in the way they obtain inputs from the user. The first structure uses a menu-input stage in which the user must indicate his choice from a menu of options displayed on the screen. In both the structures, a backward-chained rule-base may be employed to gather various types of inputs, e.g., input from a menu, a numerical input, or a literal input. In NESS, the first structure utilizes only a forward-chained rule-base, while the second utilizes only a backward-chained rule-base. Figure 4.3 diagrams the architecture of NESS.

The start_agenda frame initiates NESS and loads in necessary files. Then it calls top_level_agenda, which keeps NESS at its TOP LEVEL MENU until the user decides to return to the GENIE environment. The top_level_agenda frame displays the TOP LEVEL MENU of functions described earlier, and after the user has made a choice it drives the stage_1_rb to carry out the function selected. Stage_1_rb performs the functions given in the TOP LEVEL MENU either by calling an agenda frame that deals with a specific function, or by performing other functions itself.

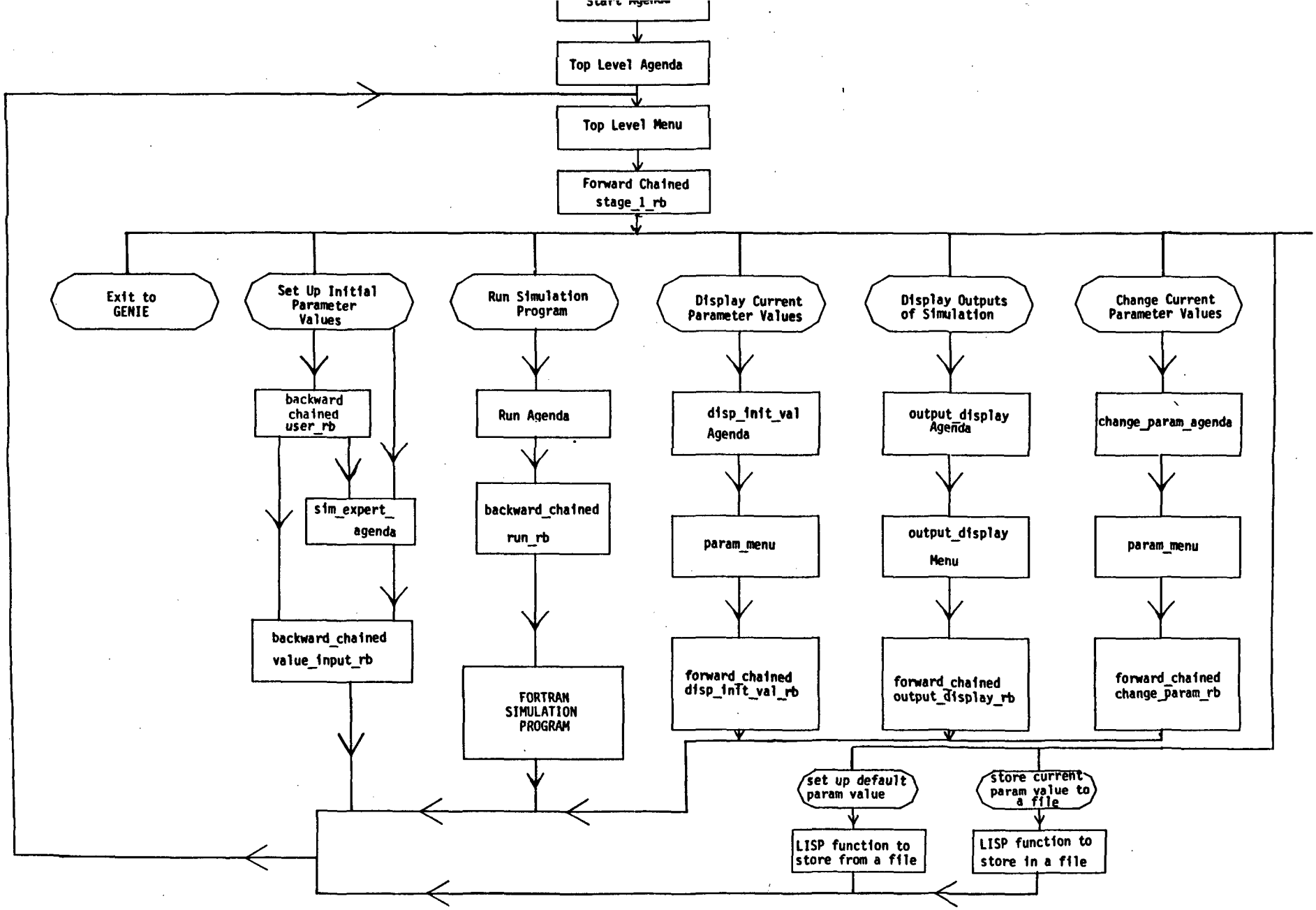


FIGURE 4.3: ARCHITECTURE OF NESS

A description of all rule-bases follows.

a) Stage_1_rb

The majority of the rules in this forward-chained rule-base are of this form.

IF the user desires to perform one of the functions listed on the TOP LEVEL MENU, THEN GENIE performs that function by calling the FRANZ LISP or GENIE functions that will accomplish the job.

In some cases these functions drive agenda frames that control some of the complex functions listed on the TOP LEVEL MENU.

Other rules prevent the user from running the simulation program or looking at the outputs of the simulation program if the necessary conditions for carrying out these functions are not fulfilled.

b) Value_input_rb

This is the backward-chained rule-base which gathers initial values of parameters necessary for the simulation experiment. The backward chain starts from rule_1, which says:

IF values have been found for all parameters necessary for simulation, THEN print message saying that the user can run the simulation program.

IF values have not been found for all parameters, THEN NESS evaluates rules concerning each of those parameters. Generally those rules are of two forms.

1) IF the user supplies the value of a parameter, THEN that parameter's value is found.

2) IF NESS can infer a parameter value from information provided by the user, THEN that parameter value is found.

Thus, when all required parameters have initial values, this rule-base stops execution and returns control to the TOP LEVEL.

At this juncture, some explanation of how frames are used in NESS should be provided. Frames are used to store parameter values. Different types of frames can store different kinds of information, e.g., scalar values, matrices, and specific attributes about parameters.

A frame which stores scalar values would look as follows:

```
(initial_value
          (T0 (0.0))
          (deltaT (0.001))
                :
                :
          (error (0.0001)))
```

Here initial_value is the name of the frame and T0, deltaT, ... error are the names of the parameters, which are scalar in nature. Following the parameters are their values as found by NESS. Initially the values would be "nil," signifying that they are not yet found.

A matrix frame would look as follows:

```
(Kp
   ((1 1) (1.0))
   ((1 2) (0.0))
          :
          :
   ((3 3) (1.0)))
```

Here Kp would be the name of the matrix, and slots (1 1) to (3 3) would be the elements of the matrix. Following the elements are their values.

A frame containing specific attributes about parameters would look as follows:

```
(Kp_matrix
          (matrix_type (diagonal))
          (matrix_full (true)))
```

3) Disp_init_val_rb

This forward-chained rule-base allows the user to look at parameter values. The rules in it are of the form:

IF the user wishes to see a particular parameter's value, THEN display that parameter's value using suitable FRANZ LISP or GENIE functions.

4) Change_param_rb

This forward-chained rule-base lets the user change parameter values before running the simulation program. Its rules are of the form:

IF the user wishes to change a particular parameter's value, THEN change that parameter's value by asking the user for a new value, and storing the new value in the slot reserved for that parameter.

But, if the user wishes to change values of a matrix, another forward-chained rule-base called change_matrix_rb is invoked.

5) Output_display_rb

This forward-chained rule-base allows the user to look at the results of the simulation experiment. The rules in it are of the form:

IF the user wishes to look at a particular result, THEN display it by

making an operating system call, which displays the contents of the file containing the result of interest onto the screen.

This is done by using the FRANZ LISP function "exec," which receives as its argument any valid operating system command.

6) Run_rb

This backward-chained rule-base runs the simulation program. First, eigenvalues of the system matrix are computed, and the results are used to infer the value of Tfinal. Special FRANZ LISP functions create the correctly formatted data file and execute the FORTRAN-based simulation program from the EUNICE environment.

5. NESS KNOWLEDGE DESCRIPTION

5.1 NESS KNOWLEDGE ACQUISITION

The knowledge residing in NESS was gathered from two primary sources, domain experts at NASA and members of the task force at Vanderbilt. The domain experts at NASA outlined the main purpose of the project and gave the task force suggestions regarding what they would like NESS to do. The task force at Vanderbilt was divided into two groups, the control group and the expert system group. The control group produced the FORTRAN-based simulation program and gave the expert system group instructions regarding the organization of NESS. The control group had frequent contacts with the experts at NASA, from whom they gained domain-specific knowledge about the project. They then developed the simulation model and generated the knowledge that was to be incorporated into NESS. The expert system group interacted with the control group and obtained all the knowledge necessary to design NESS, and developed the software for NESS as well.

5.2 NESS KNOWLEDGE OVERVIEW

NESS contains three basic types of knowledge in its knowledge-base.

1) Knowledge on FORTRAN program inputs

NESS recognizes the format of the data-file containing parameter values as required by the FORTRAN simulation model. This knowledge is contained in a special purpose FRANZ LISP function called 'setup_init_val_in_simula.inp'.

2) Knowledge on simulation

The simulation knowledge in NESS is limited to the parameters whose values are required by the simulation model. Most of the parameter values are provided by the user.

NESS contains some knowledge that enables it to infer certain parameter values. For example, to configure the Kp and Kd matrices, NESS knows that if either of them is not wanted by the user then all elements of that particular matrix are set to 0.0. If the user does not want any cross-coupling between control axes for a particular type of control, then NESS infers that the particular matrix is diagonal and sets the off-diagonal elements to 0.0. If a matrix is found to be diagonal, NESS finds the values of the diagonal elements from the user. In case the user wants all the diagonal elements to have the same value, NESS asks the user for that common value and sets all diagonal elements to that common value. If NESS finds a particular matrix to be non-diagonal then it asks whether the user wants that matrix to be symmetric or not. If the user wants a symmetric matrix, then NESS prompts the user for only six of its values and infers the other three. If a non-symmetric matrix is desired then NESS prompts the user for all nine of its values. To find the initial angular rate omega, NESS asks whether the user wants omega to be 0.0 for all axes. If the user so desires, NESS sets omega for all axes at 0.0. If the same non-zero values of omega are desired for all three axes, NESS prompts the user for that value and sets all three values to that common value. If the

user desires three different values for omega, then NESS prompts the user for three different values. The same kind of inference is used to initialize angular position theta in case the Quaternion is not wanted. If the Quaternion is wanted, then theta is initialized to its initial values. To find Tfinal (the time when the simulation program should stop computation), NESS finds the eigenvalues of the system matrix. It then finds the eigenvalue that is closest to 0.0. NESS takes the inverse of this value (TAU) and multiplies it by 5.0 (a heuristic constant) to get the value of Tfinal for step response. For frequency response NESS finds the eigenvalue real part closest to 0.0, inverts its real part and passes this value to the FORTRAN program. In case either Kp or Kd contain all 0.0 elements, the value of TAU becomes infinity. In this case a default value of 1.0 is chosen for TAU by NESS and hence Tfinal for step response becomes 5.0.

This type of knowledge is represented by rules in value_input_rb rule-base. A typical rule looks as follows:

```
IF
        [proportional control is not desired]

THEN
        [each element of Kp matrix = 0]
    and [Kp matrix is full]
    and [Kp_matrix_type is zero]
```

3) Self-knowledge (control)

NESS also contains knowledge about its own functions. This knowledge is used to prevent the user from running the simulation program or displaying, changing, or storing parameter values in a disk file when there are no parameter values in the database. NESS also prevents the user from displaying the outputs generated by the simulation program when the simulation program has not been run. NESS also contains knowledge which allows the user to display and change parameter values, display outputs generated by the simulation, store current parameter values in a disk file and set parameter values to default values.

This type of knowledge is represented in NESS by the stage_1_rb rule-base.

A typical rule looks as follows:

```
IF
        [user wants to run simulation program]
    and [all simulation parameters are not known]
THEN
        [display a message telling the user that the simulation
         program cannot be run]
```

REFERENCES

Sandell, H.S.H., The GENIE User's Manual, Vanderbilt University, 1984.

Foderaro, J.K. et al., The FRANZ LISP Manual, University of California, Berkeley, June, 1983.

Winston, P.H., Artificial Intelligence.

PART III. GENERIC SIMULATION MODEL

6. MODEL DEFINITION

6.1 MODEL OVERVIEW

The Generic Simulation Model shown in Figure 6.1 represents the generic spacecraft attitude control problem during the tracking mode. Two prototypes of the generic simulation model, Prototype 0 and Prototype I, are presently running under DEC VAX-VMS 11/780. A more detailed description of each prototype model will be discussed in the following sections.

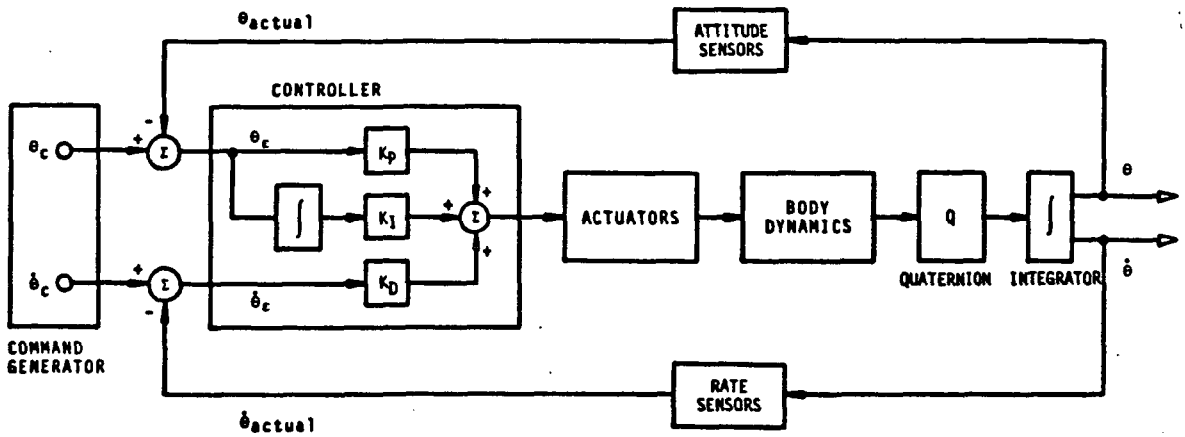


Figure 6.1. Block Diagram of the Generic Simulation Model.

The present software realization of the simulation models was done in a unique way in order to satisfy the modularity requirements. Each block in Figure 6.1 has been implemented as a module, giving the software engineer an easy way to make future changes.

Following is a discussion of the most relevant features of the generic simulation model.

Command Generator

Commanded angular attitude and/or rate are the inputs to the system. Thus, the Command Generator drives the system to meet the required input specifications in position and/or angular velocity.

Controller

Attitude error and rate error are the components of the controller presently implemented. While the attitude error may or may not drive the system according to the design requirements (Glaese et al., 1976), the rate error plays a more relevant role when performance of the system is tested under small position variations. In this way the control law of the attitude control system is considered proportional to attitude errors and/or rate errors depending on design requirements.

Actuators

The actuators will act as the modules for introducing the steering distribution laws; other torque laws, such as the magnetic torque law for a pointing system, are also included in this module for the Space Telescope (Nurre).

Body Dynamics

The body dynamics module represents the vehicle dynamics. Rate-change equations form the basis of the body dynamics. The corresponding equations for changes in angular position may or may not be included in this module.

Quaternion

Quaternion-rate equations are solved during the simulation to determine actual attitudes (Ickes 1970). The Euler angle convention is followed during the solution of such equations, and new angular positions are determined from the updated rotation matrix VR (Attitude Module Section 2).

Integrator

Three different integrators have been included in this module, giving the user the opportunity to select one of them. They are Euler, Runge-Kutta 4th order, and Predictor-software realization.

Sensor Units

Sensor units include rate and position sensors. They provide the actual values for angular attitude and rate, which will be compared with the input commands.

6.2 PROTOTYPE 0

Prototype 0 represents the simplest version of the Generic Simulation Model represented in Figure 6.1. This model was developed for the sole purpose of verifying the performance of the Generic Simulation Model without Quaternions.

Figure 6.2 illustrates the block diagram for Prototype 0. The various modules in this representation are discussed below.

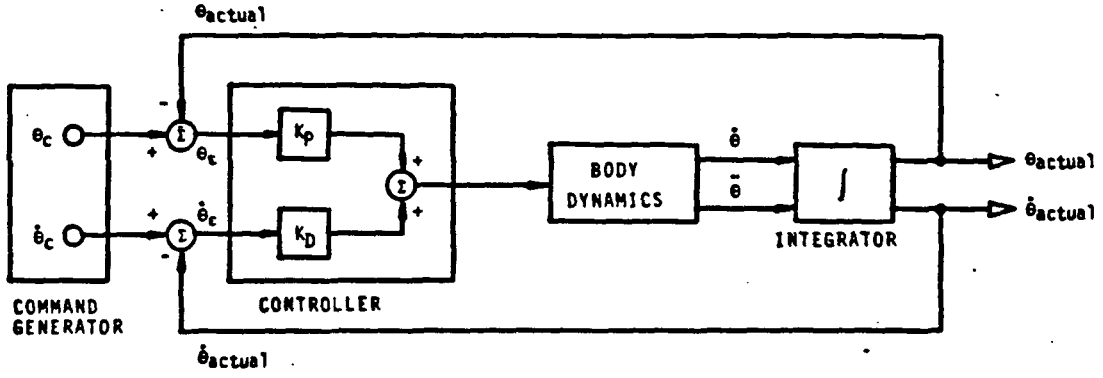


Figure 6.2. Block Diagram of Prototype 0.

Command Generator

Input commands for angular position and angular rate are considered. The input commands are therefore represented by θ_c and $\dot{\theta}_c$.

Controller

The control law is given by the torque command equation:

$$T_c = K_p\theta_e + K_d\omega_e$$

The K_p and K_d matrices are the proportional and derivative gain matrices of the system, respectively.

Body Dynamics

The dynamics of the vehicle to be controlled are represented by the set of differential equations

$$\dot{\theta} = \omega$$

$$\dot{\omega} = I^{-1}\,(T_c - \omega x I\omega)$$

where ω is the rate vector; I is the inertia matrix of the vehicle; T_c is the torque command vector; and x stands for the vectorial product between

two vectors.

Integrator

The two sets of differential vector equations are solved by either Euler, Runge-Kutta 4th order or fourth-order Predictor-Corrector routines. The selection of the type of integration routine can be made during the interaction with the expert system.

6.3 PROTOTYPE I

Prototype I more accurately models the spacecraft attitude control problem than Prototype System 0. With Prototype System I, the spacecraft can be in a moveable coordinate frame with respect to a reference direction. In the simulation of this vehicle, a Quaternion is used to transform the coordinates in one frame into equivalent coordinates in the other frame, an operation discussed in Section 1. Since the vehicle coordinate frame (V) is moving, a differential equation is a function of the vehicle angular rates. The Quaternion rate is integrated numerically, along with the differential equation representing the body dynamics, to produce the current value of the Quaternion. A vehicle-to-reference transformation matrix is thereby formed as a function of the Quaternion. The angular position of the vehicle can then be calculated by equating this specific transformation matrix with the solving for the angle of rotation (Kennel 1976). This angle is the angular position of the vehicle in the vehicle coordinate frame. In the simulation of Prototype I, it is assumed that the reference coordinate frame (R) has zero angular rate.

The block diagram for Prototype I is shown in Figure 6.3. The various modules in this representation are discussed below.

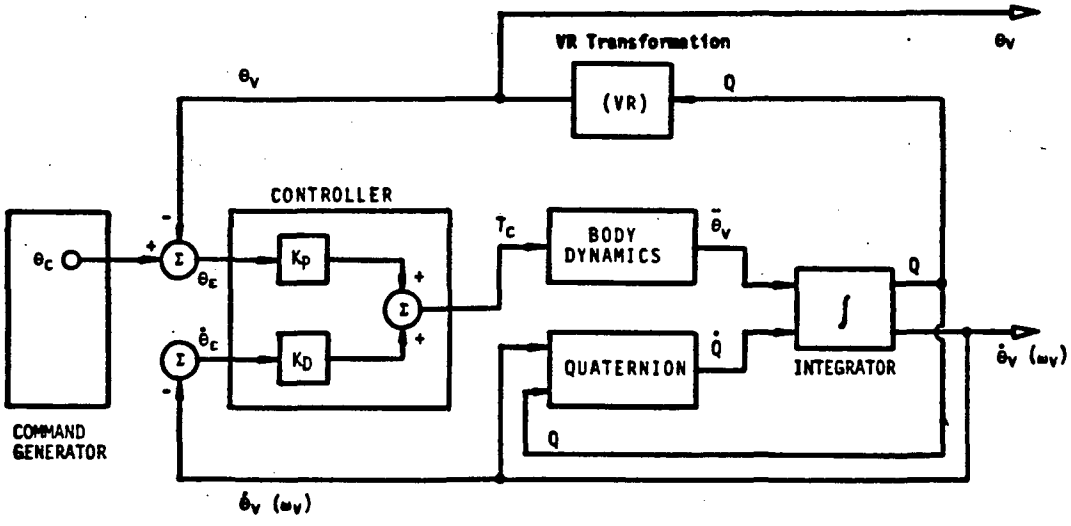


Figure 6.3. Block Diagram of Prototype I.

Command Generator

During simulation of the system, only commanded angular attitude is considered. The input command is therefore represented by θ_c.

Controller

The control algorithm is represented by the torque command equation

$$Tc = K_p \cdot \theta_{err} - K_d \cdot \omega_v$$

where θ_e represents the attitude error (the difference between the position command θ_c and angular position in V coordinates), K_p the proportional gain matrix, ω_v the body rate in terms of V coordinates, and K_d the derivative gain matrix. Since inertial hold conditions are assumed during the simulation, the angular rate of the R frame ω_r is assumed to be zero.

Body Dynamics

The dynamics of the vehicle for this prototype system are defined in terms of

$$\dot{\omega}_v = F(t,\omega_v,Tc)$$

where rate variations d/dt (ω_v) are defined in V coordinates and in terms of the F function, which is the same representation as the one in Prototype 0.

Quaternion

The Quaternion-rate equations (Klummp 1976) are defined by

$$d/dt(Q) = G(t, Q,W_v)$$

where Q represents the Quaternion, G is the vector function of states containing the information related to angular rates in V coordinates. (See Appendix C for a full discussion of Quaternions.)

Integrator

The numerical solution of the two sets of vector differential equations above is done in a similar way as the integration applied for Prototype 0. Euler, Runge-Kutta 4th order, and Predictor-Corrector routines can be chosen by the user during the interaction with the expert system. The current Quaternion values and angular rate in V coordinates are obtained from the integral solution of the aforementioned equations.

VR Transformations

Once the current Quaternion is known, a process of Quaternion normalization is performed to update these elements. After the normalized Quaternion is found, the rotation matrix VR is computed to determine the actual attitude of the vehicle.

A new set of actual angular positions with respect to each axis is computed from the current VR matrix. These new attitude values are in terms of Euler angles in V coordinates.

REFERENCES

Kennel, H.F., "Space Telescope Coordinate Systems, Symbols, and Nomenclature Definitions," NASA Technical Memorandum, TMX-73343, Sept. 1976

Glaese, J.R. et al., "Low-Cost Space Telescope Pointing Control System," Journal of Spacecraft, Vol. 13, No. 7, July 1976.

Space Station Project to the System Dynamics Lab, Marshall Space Flight Center, September, 1984.

Klummp, A.R., "Singularity-Free Extraction of a Quaternion from a Direction-Cosine Matrix," Journal of Spacecraft, Vol. 13, No. 12, Dec. 1976.

Ickes, B.P., "A New Method for Performing Digital Control System Attitude Computation Using Quaternions," AIAA Journal, Vol. 8, No. 1, Jan. 1970.

7. SYSTEM SIMULATION

The purpose of a spacecraft attitude control system is to maintain the spacecraft in a fixed orientation for a certain period of time. In order to test the performance of such a control system, time-domain and frequency-domain analyses were conducted. The details of simulation methods, analytical routines, and other useful information provided by the simulation are presented in the following sections.

7.1 TIME-DOMAIN ANALYSIS

Since the outputs of spacecraft attitude control systems are usually functions of time, the performance of the system in the time-domain must be evaluated.

Among time-domain analytical techniques, the step response method is often used. In general, a step waveform is chosen since it is easy to simulate and its instantaneous jump in amplitude shows how quickly the system will respond to a sudden input change (Kuo 1981).

The step input generated by the command generator is usually in an angular position, representing a sudden rotational requirement to the vehicle. The step response analysis will provide the following information:

a) Transient Response:

How fast the vehicle would respond to the input command. Transient response is characterized by delay time, peak time, and rising time, and maximum overshoot.

b) Steady-State Response:

The final accuracy of the vehicle in steady state. The steady state is characterized by the steady-state error and settling time.

Initial Parameter Values

Most initial parameter values are provided by the user during an interactive session with the expert system. The description of each parameter is summarized below:

. Initial time		[t0]*
. Time interval		[deltat]
. Integration allowance round error (for multistep integration methods)		[Bound(4)]**
. Steady-state error		[sse]
. Input command for:	X-axis	[Bound(13)]
	Y-axis	[Bound(14)]
	Z-axis	[Bound(15)]

. Initial conditions of:

		Prototype 0	Prototype I
- angular rate			
	Omega(x)	[Y(1)]	[Y(1)]
	Omega(y)	[Y(2)]	[Y(2)]
	Omega(z)	[Y(3)]	[Y(3)]
- angular position			
	Theta(x)	[Y(4)]	N/A
	Theta(y)	[Y(5)]	N/A
	Theta(z)	[Y(6)]	N/A
- Quaternion			
	Roll angle	N/A	[Y(7)]
	Pitch angle	N/A	[Y(8)]
	Yaw angle	N/A	[Y(9)]

. The three (3) by three (3) proportional gain matrix [Kp]

. The three (3) by three (3) derivative gain matrix [Kd]

. The three (3) by three (3) inertia matrix [INMAT]

* The variable names in the square brackets are used in the FORTRAN program.

** "Bound" is a vector which contain various input parameters; see the FORTRAN routine GENSIM in Appendix D for details.

Current Experimental Conditions:

. The inertia matrix INMAT is either identity or a multiple of identity.

. The initial conditions of angular rate are zero.

Response Analysis Routines

Three routines (see Appendix D for details), OUTPUT_17, ANALYS, and RESULTS, are used to analyze the step response.

OUTPUT_17 will be called in each time interval to write all the intermediate angular positions and rates into a file that will be used by RESULTS.

ANALYS obtains data from OUTPUT_17 to calculate the intermediate values:

. percentage of maximum overshoot [PMO]

. the starting point of rising time (10% of input) [RTini]

- the ending point of rising time (90% of input) [RTend]
- delay time (50% of input) [DT]
- settling time [ST]

RESULTS uses the outputs of ANALYSIS and provides options to display outputs in several formats:

- maximum overshoot, delay time, rising time, peak time, and settling time in numerical values
- numerical results of angular position and rate with respect to time in a screen-page format
- the graphical plotting of outputs

All of these outputs are stored in different files which can be examined in the expert system NESS by entering the proper choice in TOP LEVEL MENU of NESS.

Figure 7.1 illustrates the process-flow diagram of the Generic Simulation Model.

7.2 FREQUENCY-DOMAIN ANALYSIS

The basic feature of the frequency-domain analysis is that the performance description of a linear time-invarient control system is given in terms of its steady-state response to sinusoidally-varying input commands. Since the frequency-domain analysis lets us predict the time-domain characteristics of the performance of a system based on the sinusoidal-steady-state analysis information, it has constituted the core of classical control theory.

Stability is one important characteristic that can be derived from the frequency response. In general, the larger the gain margin and phase margin, the more stable the system becomes.

Initial Parameter Values

Most of the parameter values of frequency response are also provided by the user during an interactive session with the expert system. The description of each parameter is summarized below:

- Initial time [tO]
- Time interval [deltat]
- Integration allowance round error (for multistep integration methods) [Bound(4)]
- Input sinusoid amplitude [Amp]

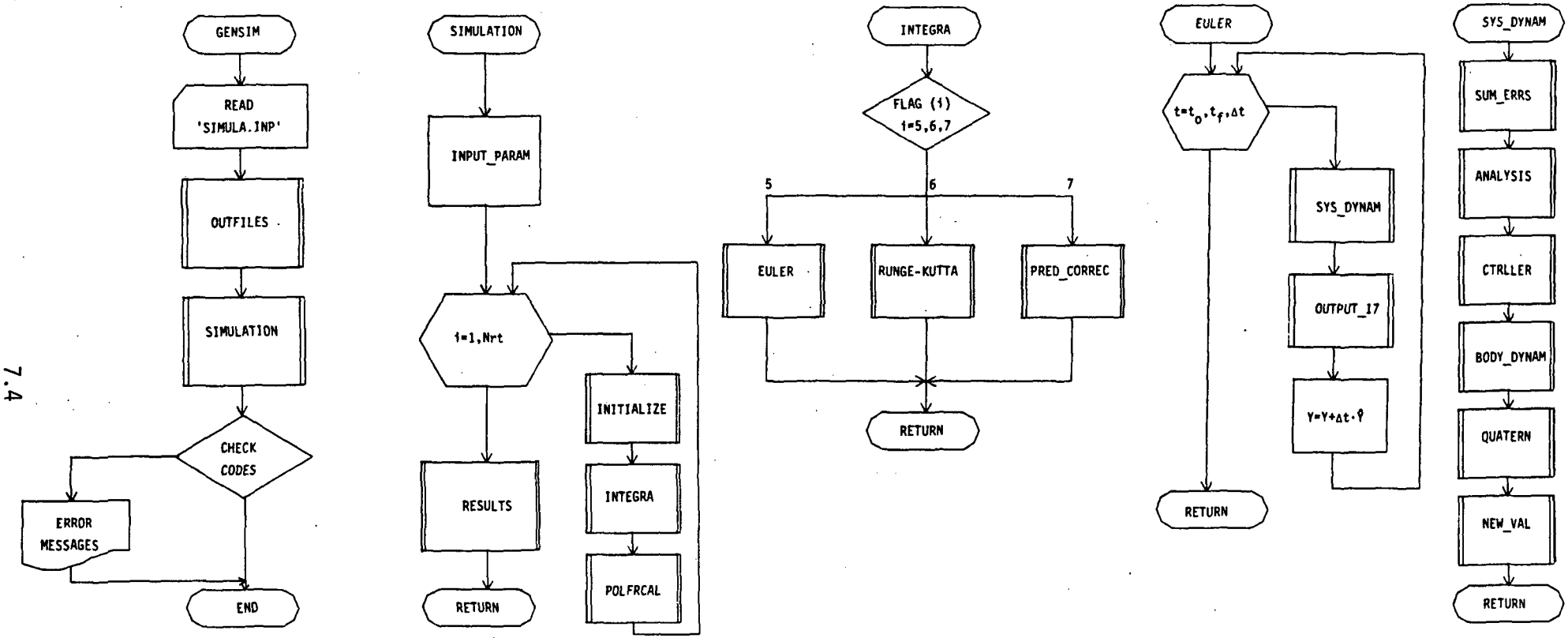


FIGURE 7.1: FLOWCHART OF GENERIC SIMULATION MODEL

. Input sinusoid lowest interest frequency [Freq]

. Input sinusoid phase [Phase]

. Number of frequency decade [Ndec]

. Number of sampling frequencies per decade [Nsd]

. Input command

X-axis	[Bound(13)]
Y-axis	[Bound(14)]
Z-axis	[Bound(15)]

. Initial conditions of:

		Prototype 0	Prototype I
- angular rate	Omega(x)	[Y(1)]	[Y(1)]
	Omega(y)	[Y(2)]	[Y(2)]
	Omega(z)	[Y(3)]	[Y(3)]
- angular position			
	Theta(x)	[Y(4)]	N/A
	Theta(y)	[Y(5)]	N/A
	Theta(z)	[Y(6)]	N/A
- Quaternion			
	Roll angle	N/A	[Y(7)]
	Pitch angle	N/A	[Y(8)]
	Yaw angle	N/A	[Y(9)]

. The three (3) by three (3) proportional controller matrix [Kp]

. The three (3) by three (3) differential controller matrix [Kd]

. The three (3) by three (3) inertial matrix [INMAT]

Current Experimental Conditions

. The inertial matrix INMAT is either identity or a multiple of identity.

. The input angular position commands are zeroes or

$$Amp \times \sin(Freq \times t + Phase)$$

Freq is the current input frequency

. The input angular rate commands are zeroes or

$$Amp \times Freq \times \cos(Freq \times t + Phase)$$

The following parameter values will be calculated before running simulation.

. Lowest frequency (Freq)
The frequencies of interest for spacecraft range from 1/60 Hz. to 20 Hz (or 0.1047197 rad./sec. to 104.7197 rad./sec).

. Number of sampling frequencies (Nsf)
The user is asked to provide the lowest frequency of interest, the number of frequency decades (Ndec), and number of sampling frequencies per decade (Nsd).

The following formulae are applied to obtain testing frequencies which are equally spaced on a log scale.

The total number of sampling frequencies (Nsf) are

$$Nsf = Nsd \times Ndec + 1 \qquad (1)$$

The constant multiplier (Const) is calculated by:

$$Const = EXP(LN(10.0)/Ndec) \qquad (2)$$

where
EXP is the exponential function
LN is the natural logarithm function

Then, the next frequency is the product of Const and the current frequency

$$Freq(next) = Freq(current) \times Const \qquad (3)$$

For example, if the user gives the lowest frequency as 0.1 rad./sec., Ndec is 3, and Nsd is also 3, from Eq.(1)

$$Nsf = 3 \times 3 + 1 = 10$$

The constant is

$$Const = EXP(LN(10.0)/3) = 2.1544347$$

From Eq.(3), ten sampling frequencies are

the first decade:
Freq(1) = 0.10000000 rad/sec
Freq(2) = 0.21544347 rad/sec
Freq(3) = 0.46415880 rad/sec

the second decade:
Freq(4) = 1.0000000 rad/sec
Freq(5) = 2.1544347 rad/sec
Freq(6) = 4.6415880 rad/sec

the third decade:
Freq(7) = 10.000000 rad/sec
Freq(8) = 21.544347 rad/sec

Freq(9) = 46.415880 rad/sec

the fourth decade:
Freq(10) = 100.00000 rad/sec

. Final time (tf):
This input parameter is calculated by the heuristic

$$tf = 6 \times Tau + T \tag{4}$$

Where T is the period of the input frequency.

Thus, tf will be different for each frequency.

Pseudo Open-Loop Frequency Response

While we are usually interested in the open-loop frequency response, spacecraft attitude control systems are closed-loop systems. In order to conduct the frequency-domain analysis, a pseudo open-loop frequency response analysis method was introduced from a utility program of the Simulation Lab at NASA Marshall Space Flight Center. For each frequency, by injecting a sinusoidal signal at a proper position in the closed-loop system, the pseudo open-loop amplitude and phase can be calculated once the system reaches steady state.

It is easily shown that the relationships of amplitude and phase between open-loop and close-loop systems can be expressed as:

$$AMP_{ol} = AMP_{cl}[AMP_{cl} + 2 \times AMP_{cl} \times \cos(PHA_{cl}) + 1] \tag{5}$$

$$PHA_{ol} = PHA_{cl} - \tan^{-1}[AMP_{cl} \times \sin(PHA_{cl}) + 1]^{1/2} \tag{6}$$

where

AMP = Amplitude
PHA = Phase
ol = open-loop system
cl = closed-loop system

For the system shown in Figure 7.2, the open-loop transfer function is

$$G(s) = Y(s)/X(s) \tag{7}$$

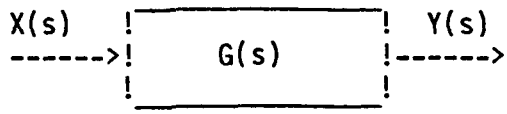


Figure 7.2

If X(s) is a sinusoidal function, then by applying the discrete

Fourier transform (Oppenheim & Schafer 1975), the discrete form of Y(s) can shown as

$$Y(n) = AMP_{cl} \times \sin[(2 \times \pi \times n/N) + PHA] \qquad (8)$$

where
n is an integer between 0 and N-1
N is the total number of samplings in T

Solving the closed-loop amplitude and phase from Eq. (8), we get

$$AMP_{cl} = 2 \times [C^2 + S^2]^{1/2}$$

$$PHA_{cl} = \tan^{-1}(C/S)$$

where

$$C = \sum_{n=1}^{N-1} \cos(2 \times \pi \times n/N) \times Y(n) \qquad (9)$$

$$S = \sum_{n=1}^{N-1} \sin(2 \times \pi \times n/N) \times Y(n) \qquad (10)$$

Substituting Eq. (9) and Eq. (10) into Eq. (5) and Eq. (6), the amplitude and phase of pseudo open-loop can be obtained.

Analysis Routines

Four routines (see Appendix D for details), INITIALIZE, ANALYS, POLFRCAL and RESULTS are used to analyze the frequency response.

INITIALIZE will be called in the beginning of each frequency response and performs the following initialization tasks:

- clears all the temporary storages
- computes input period [Period]
- computes angular frequency [Omega]
- computes integrate starting time [ts]
- computes integrate stop time [tf]

ANALYS performs the following tasks:

- computes the current angular position and rate errors to integrate sine and cosine series for the discrete Fourier summation [THS(I), THC(I), OMS(I), OMC(I)]

. computes the new angular position and rate errors
[Therr(I), Omerr(I)]

POLFRCAL will be called when the current frequency analysis is finished and uses the outputs of ANALYS.

. computes pseudo open-loop frequency amplitude and phase in decibels (dB) and degrees, respectively.
[THAOL(I), THPOL(I), OMAOL(I), OMPOL(I)]

RESULTS uses the outputs of the POLFRCAL routine and produces the frequency response outputs in several formats:

. Amplitudes and phases in numerical values for every frequency in the frequency range

. Bode plotting

REFERENCES

Kuo, B.C., Automatic Control Systems, 4th Edition, Prentice-Hall, 1981.

Oppenheim, A.V., and Schafer, R.W., Digital Signal Processing, Prentice-Hall, 1975.

PART IV. SYSTEM RUNS, VERIFICATION & EVALUATION

8. SYSTEM RUNS

The primary knowledge in NESS is concerned with gathering input data for the simulation experiment. NESS asks the user to provide initial values of some parameters and obtains others by asking questions from which it can infer them. This is done in a systematic manner as follows:

a) NESS gathers all the data required to define the system that is to be simulated. This includes getting values for the inertial and controller matrices, initializing the Quaternion module and selecting a method of integration.

b) NESS asks for the type of response desired from the system. This is done by giving the user a choice of selecting a type of response from STEP or FREQUENCY responses.

c) NESS then completes the set of parameters required to run the simulation experiment.

This is illustrated by the following sample NESS session logs for Prototype I.

```
+============================================================+
!        Welcome to NASA Expert Simulation System (NESS)     !
+============================================================+

Loading GENIE, and NESS.... (please be patient)

GENIE version 2.3 generated on Tue Apr  2 12:25:06 1985

 NESS version 1.0 generated on Thr Jun 27 11:45:05 1985

This Expert System provides an intelligent interface  to a generic
simulation program for spacecraft attitude control problems. Below
is a menu of  the functions  the system can perform.  Control will
repeatedly return to this menu after executing each user request.

          **  Please make only one choice at a time  **

top_level_choice
   top_level_menu

      1)  Exit to GENIE
      2)  Set up initial parameter values
      3)  Run simulation program
      4)  Display current parameter values required for simulation
      5)  Display outputs generated by simulation
      6)  Change initial parameter values required for simulation
      7)  Set up initial parameter values to default values
      8)  Store current parameter values in a disk file

  Please enter choice(s):
     # 2
For the present,  the Inertia matrix is assumed to be diagonal.
It is also assumed  all diagonal  elements to be 'equal'.  This
assumption makes the 'system' uncoupled in nature
Please enter the common element of the diagonal
Matrix.

Please, enter the following parameter value:

inertial_matrix
   (1 1)

Please enter value:
     >> 1
All values of the 'Inertia matrix' have been found.

Do you want a proportional (P) type control?
(For P and PD type control)

      1)  yes
      2)  no

  Please enter choice(s):
     # 1

For proportional control, do you want cross-coupling between control axes?
```

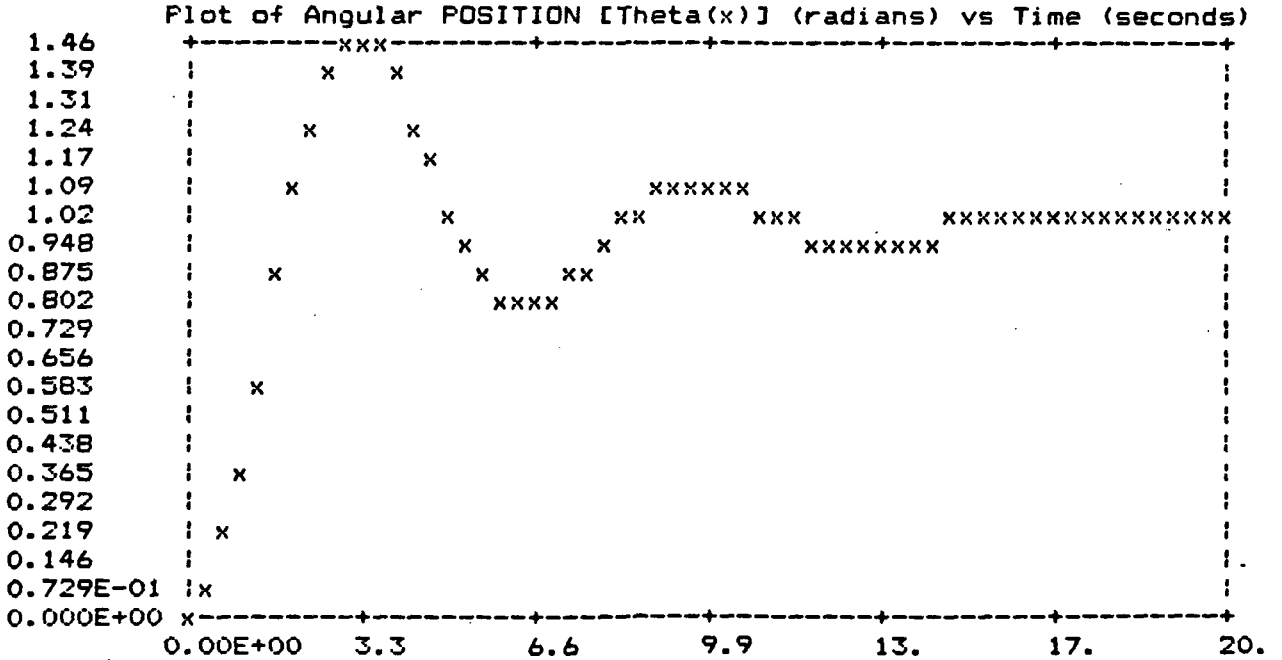
```
     1)  yes
     2)  no

  Please enter choice(s):
     # 2

Do you want the same proportional controller gain along
all axes ?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1
The common value of the Kp gain for all axes is required. It is
called Kp (1 1).

Please, enter the following parameter value:

Kp
    (1 1)

Please enter value:
     >> 1
All values of the Kp Matrix have been found.

Do you want a differential (D) type control?
(For D and PD type control)

     1)  yes
     2)  no

  Please enter choice(s):
     # 1

For differential control do you want cross-coupling between control axes?

     1)  yes
     2)  no

  Please enter choice(s):
     # 2

Do you want the same differential controller gain for
all axes ?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1
The value of the Kd Controller Gain Matrix common to all axes
is required.

Please, enter the following parameter value:
```

```
Kd
   (1 1)

Please enter value:
     >> 0.5
All values of the Kd Matrix have been found.

Do you want the Quaternion block to be included in the simulation?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1

Please enter initial roll angle (rotation about the X axis)
Enter your value in radians
Please enter value:
     >> 0

Please enter initial pitch angle (rotation about the Z axis)
Enter your value in radians
Please enter value:
     >> 0

Please enter initial Yaw angle (rotation about Y axis)
Enter your value in radians
Please enter value:
     >> 0
The Roll, Pitch, and Yaw angles have been initialized.

Please, select one method of integration:

     1)  Euler
     2)  Fourth-order Runge-Kutta
     3)  Predictor corrector

  Please enter choice(s):
     # 1

Please select from the following menu the type of System
response. Please make only one choice.

     1)  Step response
     2)  Frequency response

  Please enter choice(s):
     # 1

You can apply the input command to only one axis or none of them
Please select your choice (only one)

     1)  X
     2)  Y
     3)  Z
     4)  none
```

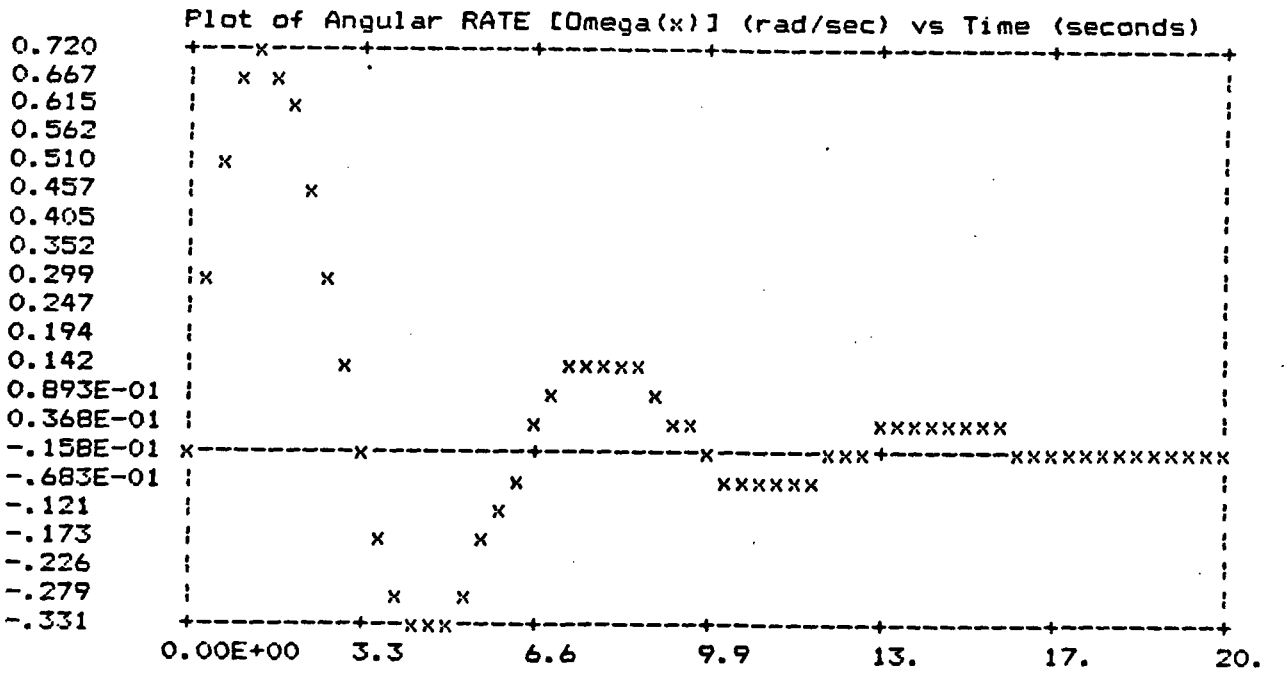
```
  Please enter choice(s):
     # 1

The time of application of the input signal, also known as
the initial time,  is required.  Usually its  value is 0.0
seconds.

initial_value
   T0

Please enter value:
     >> 0
Initial values for angular rate and/or angular position are required

The angular position for all axes have been initialized.

Do you want the initial value of the angular velocity to be 0.0
all axes?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1
The angular rate has been initialized.

Please, enter a value for delta_T. This is the value of the time
increment in the simulation.

Please enter value:
     >> 0.01

Please, enter the steady state error in percentage. For example,
type in 2.0 for a 2% steady state error.

Please enter value:
     >> 2

All parameter values needed for the simulation program have been found.
By using option 3 of the  TOP LEVEL MENU, the simulation program can be
run.

          **  Please make only one choice at a time  **

top_level_choice
   top_level_menu

     1)  Exit to GENIE
     2)  Set up initial parameter values
     3)  Run simulation program
     4)  Display current parameter values required for simulation
     5)  Display outputs generated by simulation
     6)  Change initial parameter values required for simulation
     7)  Set up initial parameter values to default values
     8)  Store current parameter values in a disk file
```

```
  Please enter choice(s):
     # 3

Eigenvalues computation has been succesful.

Tfinal
    20.0

Just above is the value of final time (Tfinal) as computed by
the system. Do you think that this value is what you want ?.
If you answer no then you can provide a new value for Tfinal.

     1)  yes
     2)  no

  Please enter choice(s):
     # 1

Do you want to run the Simulation Program?
  - If you say NO, you can go back and review parameter values.
  - If YES, you will be running the FORTRAN-77 simulation program.

     1)  yes
     2)  no

  Please enter choice(s):
     # 1
Initial parameter values are being put into 'input file'

The 'simulation program' is now being run...(Be patient)


     FORTRAN-77 'Generic Simulation Program' under Execution
     -------------------------------------------------------

      - Type of response analysis selected = STEP


          **  Please make only one choice at a time  **

top_level_choice
   top_level_menu

     1)  Exit to GENIE
     2)  Set up initial parameter values
     3)  Run simulation program
     4)  Display current parameter values required for simulation
     5)  Display outputs generated by simulation
     6)  Change initial parameter values required for simulation
     7)  Set up initial parameter values to default values
     8)  Store current parameter values in a disk file

  Please enter choice(s):
     # 5

          The simulation program generates the following outputs:
```

```
          - Options 2 and 3 are ASCII file plottings.
          - Options 4 and 5 are numerical outputs.
        Be sure to have run the simulation before observing the
        outputs.  The outputs shown will correspond to the out-
        puts generated when the simulation was last run.

output_display_choice
   output_display_menu

     1)  Return to TOP LEVEL MENU
     2)  Plot of omega
     3)  Plot of theta
     4)  Characteristics of the step response analysis
     5)  Numerical output generated by simulation

  Please enter choice(s):
     # 3
```



```
        The simulation program generates the following outputs:
              - Options 2 and 3 are ASCII file plottings.
              - Options 4 and 5 are numerical outputs.
        Be sure to have run the simulation before observing the
        outputs.  The outputs shown will correspond to the out-
        puts generated when the simulation was last run.

output_display_choice
   output_display_menu

     1)  Return to TOP LEVEL MENU
     2)  Plot of omega
     3)  Plot of theta
     4)  Characteristics of the step response analysis
     5)  Numerical output generated by simulation

  Please enter choice(s):
     # 2
```



```
          The simulation program generates the following outputs:
                - Options 2 and 3 are ASCII file plottings.
                - Options 4 and 5 are numerical outputs.
          Be sure to have run the simulation before observing the
          outputs.  The outputs shown will correspond to the out-
          puts generated when the simulation was last run.


output_display_choice
   output_display_menu

      1)  Return to TOP LEVEL MENU
      2)  Plot of omega
      3)  Plot of theta
      4)  Characteristics of the step response analysis
      5)  Numerical output generated by simulation

  Please enter choice(s):
      # 1

            **  Please make only one choice at a time  **

top_level_choice
   top_level_menu

      1)  Exit to GENIE
      2)  Set up initial parameter values
      3)  Run simulation program
      4)  Display current parameter values required for simulation
      5)  Display outputs generated by simulation
      6)  Change initial parameter values required for simulation
      7)  Set up initial parameter values to default values
      8)  Store current parameter values in a disk file

  Please enter choice(s):
      # 1
```

```
+============================================================+
!       Welcome to NASA Expert Simulation System (NESS)      !
+============================================================+

Loading GENIE, and NESS.... (please be patient)

GENIE version 2.3 generated on Tue Apr  2 12:25:06 1985

 NESS version 1.0 generated on Thu Jun 27 11:45:05 1985

This Expert System provides an intelligent interface  to a generic
simulation program for spacecraft attitude control problems. Below
is a menu of  the functions  the system can perform.  Control will
repeatedly return to this menu after executing each user request.

          **  Please make only one choice at a time  **


top_level_choice
   top_level_menu

     1)  Exit to GENIE
     2)  Set up initial parameter values
     3)  Run simulation program
     4)  Display current parameter values required for simulation
     5)  Display outputs generated by simulation
     6)  Change initial parameter values required for simulation
     7)  Set up initial parameter values to default values
     8)  Store current parameter values in a disk file

  Please enter choice(s):
     # 2
For the present,  the Inertia matrix is assumed to be diagonal.
It is also assumed  all diagonal  elements to be 'equal'.  This
assumption makes the 'system' uncoupled in nature.
Please enter the common element of the diagonal of the Inertial
Matrix.
Please, enter the following parameter value:

inertial_matrix
   (1 1)

Please enter value:
     >> 1
All values of the 'Inertia matrix' have been found.

Do you want a proportional  (P) type control?
(For P and PD type control)

     1)  yes
     2)  no

  Please enter choice(s):
     # 1

For proportional control, do you want cross-coupling between control axes?
```

```
     1)  yes
     2)  no

  Please enter choice(s):
     # 2

Do you want the same proportional controller gain along
all axes ?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1
The common value of the Kp gain for all axes is required. It is
called Kp (1 1).
Please, enter the following parameter value:

Kp
   (1 1)

Please enter value:
     >> 1
All values of the Kp Matrix have been found.

Do you want a differential (D) type control?
(For D and PD type control)

     1)  yes
     2)  no

  Please enter choice(s):
     # 1

For differential control do you want cross-                          axes?

     1)  yes
     2)  no

  Please enter choice(s):
     # 2

Do you want the same differential controller gain for
all axes ?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1
The value of the Kd Controller Gain Matrix common [illegible]
is required.
Please, enter the following parameter value:

Kd
   (1 1)
```

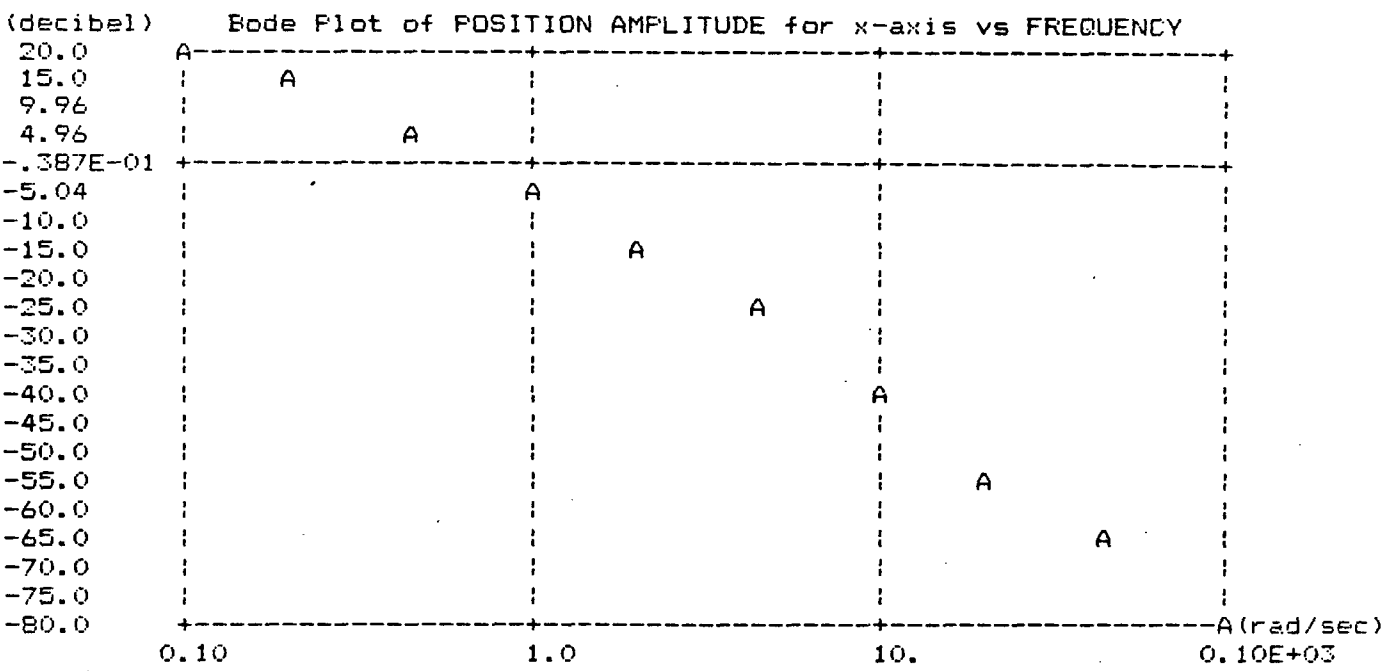
```
Please enter value:
     >> 1
All values of the Kd Matrix have been found.

Do you want the Quaternion block to be included in the simulation?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1

Please enter initial roll angle (rotation about the X axis)
Enter your value in radians
Please enter value:
     >> 0

Please enter initial pitch angle (rotation about the Z axis)
Enter your value in radians
Please enter value:
     >> 0

Please enter initial Yaw angle (rotation about Y axis)
Enter your value in radians
Please enter value:
     >> 0

The Roll, Pitch, and Yaw angles have been initialized.

Please, select one method of integration:

     1)  Euler
     2)  Fourth-order Runge-Kutta
     3)  Predictor corrector

  Please enter choice(s):
     # 1

Please select from the following menu the type of System
response. Please make only one choice.

     1)  Step response
     2)  Frequency response

  Please enter choice(s):
     # 2

You can apply the input command to only one axis or none of them
Please select your choice (only one)

     1)  X
     2)  Y
     3)  Z
     4)  none

  Please enter choice(s):
```

```
     # 1

The time of application of the input signal, also known as
the initial time,  is required.  Usually its  value is 0.0
seconds.

initial_value
   TO

Please enter value:
     >> 0

Initial values for angular rate and/or angular position are required
The angular position for all axes have been initialized.

Do you want the initial value of the angular velocity to be 0.0
all axes?

     1)  yes
     2)  no

  Please enter choice(s):
     # 1
The angular rate has been initialized.

Initial parameter values for FREQUENCY response analysis are required.

Enter the Amplitude of the input waveform for FREQUENCY response
Normal value is 1.0 radian
Please enter value:
     >> 1

Enter the lowest frequency of input wavwform to be applied to
the system for FREQUENCY response
Enter your value in Hertz
Normal value is 0.0159154 Hz corresponding to 0.1 rad/sec
Please enter value:
     >> 0.0159154

Please enter a value for deltaT for frequency response.
If you choose 512 from the menu below deltaT will be fixed
to 1/512 or 0.00195312 sec.
This value of 512 is recommended.

     1)  256
     2)  512
     3)  1024

  Please enter choice(s):
     # 2

Enter the number of decades of input signal for FREQUENCY response
Normal value is 3 decades
```

```
Please enter value:
     >> 3

Enter the number of sampling frequencies per decade
for FREQUENCY response
Normal value is 3 frequencies
Please enter value:
     >> 3

Enter the phase of the input waveform for FREQUENCY response
Normal value is 0.0 radians
Please enter value:
     >> 0

All parameter values for FREQUENCY response analysis have been
found.

All parameter values needed for the simulation program have been found.
By using option 3 of the  TOP LEVEL MENU, the simulation program can be
run.

          **  Please make only one choice at a time  **


top_level_choice
   top_level_menu

     1)  Exit to GENIE
     2)  Set up initial parameter values
     3)  Run simulation program
     4)  Display current parameter values required for simulation
     5)  Display outputs generated by simulation
     6)  Change initial parameter values required for simulation
     7)  Set up initial parameter values to default values
     8)  Store current parameter values in a disk file

  Please enter choice(s):
     # 3

Eigenvalues computation has been succesful.
2.0

Given above is the calculated value of TAU.


Do you want to run the Simulation Program?
  - If you say NO, you can go back and review parameter values.
  - If YES, you will be running the FORTRAN-77 simulation program.

     1)  yes
     2)  no

  Please enter choice(s):
     # 1

Initial parameter values are being put into 'input file'
```

```
The 'simulation program' is now being run...(Be patient)

     FORTRAN-77 'Generic Simulation Program' under Execution
     -------------------------------------------------------

      - Type of response analysis selected = FREQUENCY

                  [ Display of computation results ]
                  [          from FORTRAN-77       ]


            **  Please make only one choice at a time  **


top_level_choice
   top_level_menu

      1)  Exit to GENIE
      2)  Set up initial parameter values
      3)  Run simulation program
      4)  Display current parameter values required for simulation
      5)  Display outputs generated by simulation
      6)  Change initial parameter values required for simulation
      7)  Set up initial parameter values to default values
      8)  Store current parameter values in a disk file

  Please enter choice(s):
     # 5

           The simulation program generates the following outputs:
                 - Options 2 and 3 are ASCII file plottings.
                 - Options 4 and 5 are numerical outputs.
           Be sure to have run the simulation before observing the
           outputs.  The outputs shown will correspond to the out-
           puts generated when the simulation was last run.


output_display_choice
   output_display_menu

      1)  Return to TOP LEVEL MENU
      2)  Plot of omega
      3)  Plot of theta
      4)  Characteristics of the step response analysis
      5)  Numerical output generated by simulation

  Please enter choice(s):
     # 3

Do you want to display for theta
     1) Amplitude.
     2) Phase.
Please enter choice:
```

1



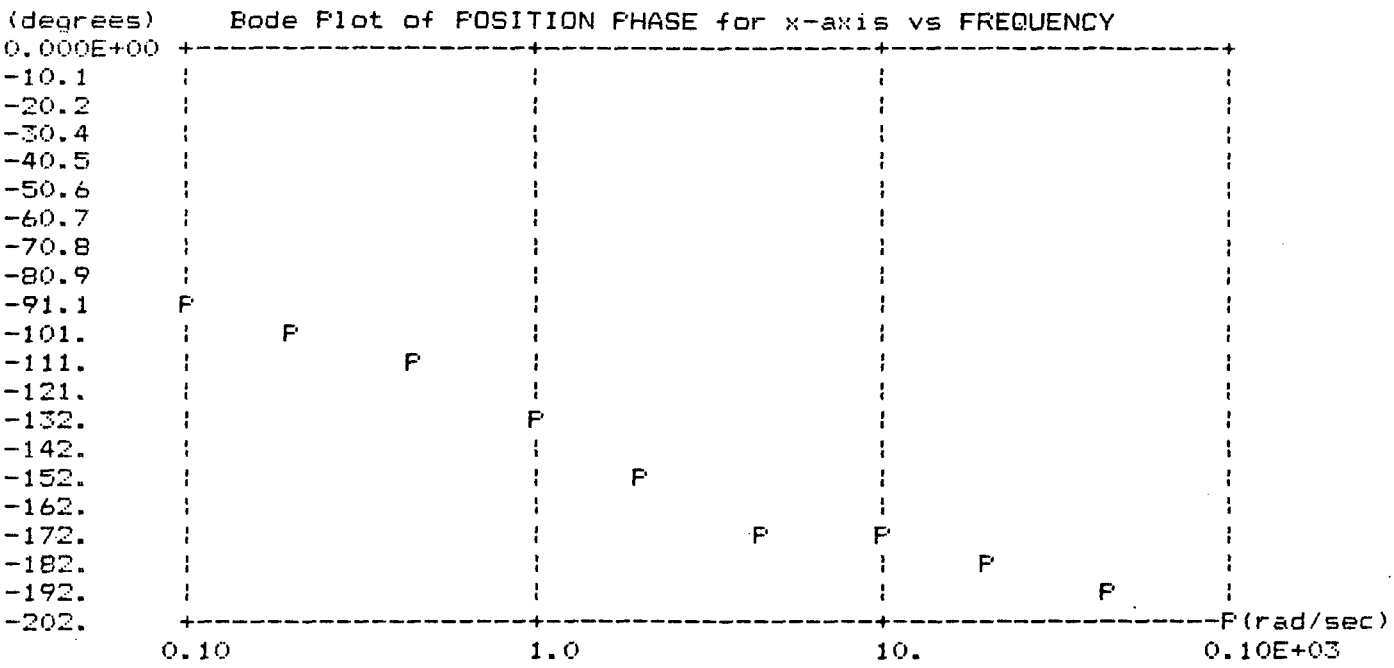
$

The simulation program generates the following outputs:
- Options 2 and 3 are ASCII file plottings.
- Options 4 and 5 are numerical outputs.

Be sure to have run the simulation before observing the outputs. The outputs shown will correspond to the outputs generated when the simulation was last run.

```
output_display_choice
   output_display_menu

     1)  Return to TOP LEVEL MENU
     2)  Plot of omega
     3)  Plot of theta
     4)  Characteristics of the step response analysis
     5)  Numerical output generated by simulation

  Please enter choice(s):
     # 3

Do you want to display for theta
     1) Amplitude.
     2) Phase.
Please enter choice:
```

2



$

The simulation program generates the following outputs:
- Options 2 and 3 are ASCII file plottings.
- Options 4 and 5 are numerical outputs.

Be sure to have run the simulation before [illegible]
outputs. The outputs shown will [illegible]
puts generated when the simulation [illegible]

```
output_display_choice
   output_display_menu

      1)  Return to TOP LEVEL MENU
      2)  Plot of omega
      3)  Plot of theta
      4)  Characteristics of the step response analysis
      5)  Numerical output generated by simulation

  Please enter choice(s):
      # 1

           **  Please make only one choice at a time  **

top_level_choice
   top_level_menu
```

```
    1)  Exit to GENIE
    2)  Set up initial parameter values
    3)  Run simulation program
    4)  Display current parameter values required for simulation
    5)  Display outputs generated by simulation
    6)  Change initial parameter values required for simulation
    7)  Set up initial parameter values to default values
    8)  Store current parameter values in a disk file

Please enter choice(s):
    # 1
```

9. SYSTEM VERIFICATION

In the design of any software, a major consideration is the verification of the design. It is extremely important to determine whether or not the software correctly implements the specified design. As software designs become larger and more complex, this task becomes increasingly difficult. Methods of verification include comparison with analytic solutions and comparison with known and accepted solutions from other designs. Each of these methods is used in the verification of the software developed during Phase I of this research.

Generally two approaches are taken in verifying a software design. First, individual routines are tested to determine that they correctly perform their operation. Once all (or as many as possible) of the routines are tested individually, the overall collection of routines must perform satisfactorily. This step is necessary as a final check of the design and to verify that the main program properly initializes variables and calls the various routines in the proper order.

The system represented by the generic simulation is sixth-order. The system description includes three matrices, each of dimension three-by-three. These matrices are the inertia matrix, the proportional gain matrix, and the derivative gain matrix. If the inertia matrix is a multiple of the identity matrix and each of the two gain matrices is diagonal, the sixth-order system is broken up into three completely independent second-order systems. These three systems are then also linear for the body dynamics chosen for the generic simulation. Since analytic solutions for second-order linear systems are readily available for responses to step and sinusoidal inputs, a convenient basis of comparison exists for this situation.

In order to verify the software in the uncoupled configuration, the simulation is run for step and sinusoidal inputs for commanded position. The sinusoidal input is used to produce the system's frequency response, and the step input produces a time domain response. Results of the simulation are compared to solutions generated analytically for the same parameter values. Close agreement between the solutions obtained analytically and through simulation verifies that the simulation software is accurately modeling the given system and accurately solving those equations of motion for the specified input. This approach has been taken to verify the software for Prototype 0. Table 9.1 shows the analytic and simulated position outputs for a step input, and Table 9.2 presents a list of analytic and simulated characteristics, such as percent overshoot, for the same input. Table 9.3 and Table 9.4 present frequency domain results. In each table, the position command is a sinusoid of specified frequency, and both analytic and simulated results are shown. In Table 9.3, the rate command is zero, and in Table 9.4, the rate command is the time derivative of the position command. These Tables show excellent agreement between the simulation results and those obtained analytically.

Tables 9.5 and 9.6 include step response results for Prototype I; the former table shows the actual output position values, and the latter table gives the summary characteristics of the step response. There is close agreement between the results of Prototype 0 and Prototype I, as would be

expected. Since the results of Prototype 0 agree closely with those of the analytic solution, the performance of Prototype I is also verified. Table 9.7 presents the frequency domain results for Prototype I. The rate command is zero in this table. Again, there is close agreement between Prototype 0 and Prototype I, providing verification of the accuracy of the simulation software. Tables 9.8 and 9.9 list the input parameters used during time-domain and frequency domain analysis, respectively.

One of the computations that the software must perform is the determination of the eigenvalues of the system. In the uncoupled case where only second-order systems are considered, this is trivial. When the two controller matrices have non-zero off-diagonal elements, motion along the three coordinate axes will be coupled; however, as long as the inertia matrix remains a multiple of the identity matrix, the system is still linear. Determination of the eigenvalues then requires the manipulation of a six-by-six matrix. The eigenvalue routine was evaluated by testing it on a seven-by-seven matrix with known eigenvalues. The matrix has two sets of complex conjugate eigenvalues, one repeated real eigenvalue of multiplicity two, and one real distinct eigenvalue. Most of the eigenvalues are represented by irrational numbers. The software routine used to determine the eigenvalues completed the calculation of these eigenvalues to well within the expected round-off error.

Time (sec)	Theta Analy. (rad)	Theta NESS (rad)
0.00000	0.000000	0.000000
0.50000	0.104405	0.103096
1.00000	0.340300	0.339653
1.50000	0.610493	0.611507
2.00000	0.849426	0.852124
2.50000	1.023360	1.027101
3.00000	1.124350	1.128246
3.50000	1.161650	1.164889
4.00000	1.153120	1.155205
4.50000	1.118450	1.119230
5.00000	1.074590	1.072416
5.50000	1.033620	1.032493
6.00000	1.002290	1.000809
6.50000	0.982846	0.981385
7.00000	0.974359	0.973183
7.50000	0.974152	0.973394
8.00000	0.979007	0.978680
8.50000	0.985996	0.986027
9.00000	0.992934	0.993203
9.50000	0.998504	0.998885
10.0000	1.002170	1.002555

TABLE 9.1
Comparisons of Analytical and Computational Results
for Theta of Step Response (Prototype 0).

Characteristics of Step Response	Analy.	NESS
PMO (%)	16.3033	16.6023
PT (sec)	3.6200	3.6100
RT (sec)	1.6400	1.6300
DT (sec)	1.2900	1.2900
ST (sec)	8.0800	8.1000

PMO - PERCENTAGE OF MAXIMUM OVERSHOOT
PT - PEAK TIME
RT - RISING TIME
DT - DELAY TIME
ST - SETTLING TIME

TABLE 9.2
Comparisons of Analytical and Computational Characteristics Results for Theta of Step Response (Prototype 0).

Frequency (rad/sec)	Amplitude Analy. (dB)	Amplitude NESS	Phase Analy. (degree)	Phase NESS
0.1000	19.96	19.96	- 95.71	- 95.75
0.2154	13.14	13.13	-102.16	-102.19
0.4642	5.82	5.82	-114.90	-114.95
1.0000	-3.01	- 3.00	-135.00	-135.10
2.5140	-14.18	-14.16	-155.10	-155.42
4.6240	-26.86	-26.83	-167.84	-168.38
10.0000	-40.04	-40.03	-174.29	-175.31
21.5400	-53.54	-53.25	-177.34	-179.71
46.4200	-66.67	-66.72	-178.77	-183.48
100.0000	-80.00	-80.06	-179.73	-190.50

TABLE 9.3
Comparisons of Analytical and Computational Results for Frequency Response of Prototype 0 (Rate command = 0 & Amplitude = 1.0).

Frequency (rad/sec)	Amplitude Analy. (dB)	Amplitude NESS	Phase Analy. (degree)	Phase NESS
0.1000	40.04	39.98	-174.29	-174.47
0.2152	26.86	26.84	-167.84	-167.77
0.4642	14.18	14.18	-155.10	-155.16
1.0000	3.01	3.01	-135.00	-135.15
2.1520	- 5.82	- 5.83	-114.90	-115.22
4.6420	-13.14	-13.13	-102.16	-102.65
10.0000	-19.96	-19.97	- 95.71	- 97.02
21.5200	-26.66	-26.68	- 92.66	- 95.17
46.4200	-33.33	-33.13	- 91.23	- 96.77
100.0000	-40.00	-39.51	- 90.57	-101.19

TABLE 9.4
Comparisons of Analytical and Computational Results for Frequency Response of Prototype 0 (Rate command = Derivative of Step Command & Amplitude = 1.0).

Time (sec)	Theta Analy.	SYS. 0 (rad)	SYS. I
0.00000	0.000000	0.000000	0.000000
0.50000	0.104405	0.103096	0.103240
1.00000	0.340300	0.339653	0.340560
1.50000	0.610493	0.611507	0.613719
2.00000	0.849426	0.852124	0.855714
2.50000	1.023360	1.027101	1.031648
3.00000	1.124350	1.128246	1.133025
3.50000	1.161650	1.164889	1.169142
4.00000	1.153120	1.155205	1.158355
4.50000	1.118450	1.119230	1.120997
5.00000	1.074590	1.072416	1.074658
5.50000	1.033620	1.032493	1.031832
6.00000	1.002290	1.000809	0.999473
6.50000	0.982846	0.981385	0.979799
7.00000	0.974359	0.973183	0.971699
7.50000	0.974152	0.973394	0.972247
8.00000	0.979007	0.978680	0.977973
8.50000	0.985996	0.986027	0.985753
9.00000	0.992934	0.993203	0.993275
9.50000	0.998504	0.998885	0.999179
10.0000	1.002170	1.002555	1.002947

TABLE 9.5
Comparisons of Analytical and Computational Results
for Theta of Step Responses (Prototype 0 & I).

Characteristics of Step Response	Analy.	SYS. 0	SYS. I
PMO (%)	16.3033	16.6023	17.0077
PT (sec)	3.6200	3.6100	3.6100
RT (sec)	1.6400	1.6300	1.6200
DT (sec)	1.2900	1.2900	1.3000
ST (sec)	8.0800	8.1000	8.1500

PMO - PERCENTAGE OF MAXIMUM OVERSHOOT
PT - PEAK TIME
RT - RISING TIME
DT - DELAY TIME
ST - SETTLING TIME

TABLE 9.6
Comparisons of Analytical and Computational Characteristics Results for Theta of Step Responses (Prototype 0 & I).

Frequency (rad/sec)	Amplitude Analy. (dB)	Amplitude SYS. 0	Amplitude SYS. I	Phase Analy. (degree)	Phase SYS. 0	Phase SYS. I
0.1000	19.96	19.96	19.96	- 95.71	- 95.75	- 95.76
0.2154	13.14	13.13	13.13	-102.16	-102.19	-102.22
0.4642	5.82	5.82	5.82	-114.90	-114.95	-115.00
1.0000	-3.01	- 3.00	- 3.00	-135.00	-135.10	-135.21
2.5140	-14.18	-14.16	-14.16	-155.10	-155.42	-155.67
4.6240	-26.86	-26.83	-26.83	-167.84	-168.38	-168.90
10.0000	-40.04	-40.03	-40.03	-174.29	-175.31	-176.43
21.5400	-53.54	-53.25	-53.25	-177.34	-179.71	-182.10
46.4200	-66.67	-66.72	-66.73	-178.77	-183.48	-188.84
100.0000	-80.00	-80.06	-80.03	-179.73	-190.50	-202.37

TABLE 9.7
Frequency Response (Rate command = 0) Comparisons for Analytical (Prototype 0) and Computational (Prototype 0 & I) Results (Amplitude = 1.0).

Input Parameter	unit	value
- Initial time	(sec)	0.00
- Final time	(sec)	10.0
- Time interval	(sec)	0.01
- Steady-state-error	(%)	2.00
- Input commands		
angular position	(rad)	
theta(x)		1.0
theta(y)		0.0
theta(z)		0.0
angular rate	(rad/sec)	
omega(x)		0.0
omega(y)		0.0
omega(z)		0.0
- Initial conditions		
SYS. 0		
angular position	(rad)	
theta(x)		0.0
theta(y)		0.0
theta(z)		0.0
angular rate	(rad/sec)	
omega(x)		0.0
omega(y)		0.0
omega(z)		0.0
SYS. I		
Quaternion	(rad)	
angular rate	(rad/sec)	
omega(x)		0.0
omega(y)		0.0
omega(z)		0.0
- Kp, Kd, INMAT		identity matrices
- Integration method		Euler method

TABLE 9.8
Input Parameters for Time-Domain Analysis.

Input Parameter	unit	value
- Initial time	(sec)	0.0
- Starting time for computing open-loop freq. response	(sec)	ts = 6 x Tau
- Final time	(sec)	ts + T
- Time interval	(sec)	1/512
- Lowest frequency	(rad)	0.1
- Number of frequency decade		3
- Number of sampling frequencies per decade		3
- Input commands		
angular position	(rad)	
theta(x)		1.0
theta(y)		0.0
theta(z)		0.0
angular rate	(rad/sec)	
omega(x)		0.0
omega(y)		0.0
omega(z)		0.0
- Initial conditions		
SYS. 0		
angular position	(rad)	
theta(x)		0.0
theta(y)		0.0
theta(z)		0.0
angular rate	(rad/sec)	
omega(x)		0.0
omega(y)		0.0
omega(z)		0.0
SYS. I		
Quaternion	(rad)	
angular rate	(rad/sec)	
omega(x)		0.0
omega(y)		0.0
omega(z)		0.0
- Kp, Kd, INMAT		identity matrices
- Integration method		Euler method

TABLE 9.9 Input Parameters for Frequency-Domain Analysis.

10. SYSTEM EVALUATION

The specific goals of this project evolved in the project's early weeks and did not result in concrete performance targets. The major goal of Phase I became to produce a prototype system to demonstrate the feasibility of applying AI technology to this task domain and further to serve as a foundation for a more intelligent system. With this goal in mind, many interactive evaluations were performed with NASA personnel while the prototype was taking shape. While these informal evaluations resulted in many of the features embodied in the prototype system, we now feel that a more formal evaluation is in order.

We envision the evaluation as a cooperative activity between the NASA personnel, who will specify what they would like the system to do, and the Vanderbilt project team, who will try to specify what can be achieved. We propose that this evaluation be the first activity of Phase II and that from it a set of clear performance objectives should be formulated for the system at the end of Phase II.

The following thoughts are suggestions which may be helpful in framing this evaluation.

a) After the Phase I prototype system is demonstrated, it might be useful to draft a statement of the desired performance envelope for the ultimate expert simulation system now envisioned by NASA. This statement should describe the ideal user interaction format, the practical limits on the generality of the generic simulation software (i.e., when special-purpose routines would be expected) and a clear statement of the inferences (expert knowledge) NESS would have to be able to perform to be useful. The suggestions offered in the Phase II proposal might be helpful here, and a statement of the prior experience expected of the user community would also be useful.

b) With a clear statement of the desired performance envelope in hand, the performance of the prototype can be evaluated and a clear list of work to be done can be formulated and prioritized. This, then, would become part of the Phase II agenda.

PART V

CONCLUSIONS

The current phase of development (Phase I) has produced an expert system that can assist the user in running a class of spacecraft attitude control simulations. Although the knowledge-base and the simulation model are relatively simple and limited, we have successfully demonstrated the coupling of symbolic processing and numerical computation. Future work (Phase II) will include enhancement and expansion of the capabilities of both the expert system and the simulation model.

In Phase II, the expert system will be extended to interpret the output data and determine system characteristics such as percent overshoot, settling time, gain margin and phase margin. NESS will also be extended to recommend parameter values or subroutines that should be used to get the desired results.

The simulation model will be extended to include actuator and steering distribution equations. It will also be extended to include bending modes in the body dynamics equations. Finally, it will be modularized in such a way that NESS will allow the user to choose a combination of submodules in the simulation model and run experiments.