N88-17223

DEVELOPMENT OF A COUPLED EXPERT SYSTEM FOR THE SPACECRAFT ATTITUDE CONTROL PROBLEM

K. Kawamura, G. Beale, J. Schaffer, B.-J. Hsieh, S. Padalkar
and J. Rodriguez-Moscoso
Center for Intelligent Systems
Vanderbilt University
Nashville, TN 37235

F. Vinz and K. Fernandez
National Aeronautics and Space Administration
Huntsville, Alabama 35812

Abstract

A majority of the current expert systems focus on the symbolic-oriented logic and inference mechanisms of artificial intelligence (AI). Common rule-based systems employ empirical associations and are not well suited to deal with problems often arising in engineering. This paper describes a prototype expert system which combines both symbolic and numeric computing. The expert system's configuration is described and its application to a space craft attitude control problem is presented.

Introduction

Current NASA planning to develop a low earth-orbit Space Station poses a unique opportunity for the development of an expert system for coupling symbolic processing and numerical computations.

Computer simulations are used extensively by NASA to verify system design. These simulations are developed by highly skilled simulation specialists and the complexity of these simulations require that the specialist be involved in the operational phase as well as in development. This results in a poor utilization of personnel. An expert system coupling symbolic processing and numerical computations may solve this problem by permitting detailed experiments and studies to be performed without the investigator's need to have a detailed knowledge of the model implementation.

Development of the Space Station will also require close coordination among system designers from NASA, the aerospace industry and other participants. An intelligent system with enough knowledge of system design may be able to assist in this coordination. Such a system could interact with each system designer in an intelligent way, allowing for the exploration of alternative designs, pointing out potential problems, catching forgotten details, etc. (De Jong (1983)). This system could also inform the other members of the design team of critical decisions made.

Most current expert systems focus on symbolic reasoning and inference mechanisms and traditionally have not been concerned with the numerical processes frequently used on engineering problems (e.g., the simulation of dynamic systems) (Kawamura (1985a)). However, the intelligent use of these numerical methods involves the kinds of expertise with which AI has dealt, and which is frequently in short supply.

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

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

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

As an initial step toward development of such an intelligent system, an expert system called NESS (NASA Expert Simulation System) was developed, which understands the usage of a class of spacecraft attitude control simulation software and can assist the user in running the software. NESS was build using a knowledge-engineering tool called GENIE (GENeric Inference Engine) (Sandell (1984)), developed at Vanderbilt University. The simulation software represents a simplified model of a typical spacecraft. It has many of the same functions which appear in the simulation software of an actual spacecraft. The purpose of the generic simulation model is to serve as a test-bed simulation during the development of NESS. Since it was developed at Vanderbilt, the generic simulation model is well understood and is easily modified. Its use made the understanding of how to interface expert systems to simulation programs much easier than if an actual simulation program had been used.

COUPLED EXPERT SYSTEM

Design Principle

One of the major design decisions of this project was to maintain a clear separation between the generic simulation model, which performs the numeric computations, and the expert system, which performs symbolic processing. This parallels the situation in which a human expert sets out to perform a numerical simulation experiment. The human expert, using his or her knowledge of the system to be modeled and the characteristics of the simulation software, makes decisions about how to run the 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 the simulation model and switches which inform the program of the options selected by the user. The user then issues a command to the operating system to run the simulation program. If it runs without error, the user then examines the output files and interprets the results.

Following this approach, the expert system (NESS) was designed as a software system separate from the generic simulation model. Figure 1 shows the interaction of these two systems schematically. Each system was written in the language most natural to it. The generic simulation model was written in FORTRAN following years of traditional engineering practice, and NESS was build using a general inference engine (GENIE) written in FRANZ LISP, following current AI practice.

The knowledge-base of NESS contains three types of knowledge: general knowledge about spacecraft attitude control simulation experiments; specific knowledge of the input parameters and their formats required by the generic simulation program; and self-knowledge which is used to prevent foolish behavior, such as attempting to examine results before a simulation run has been executed.

The user-interaction scenario is envisioned as follows. The users invokes NESS which queries him or her about the system to be modeled and the experiment to be performed. 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 your questions. The setting of the low-level parameters should be inferred and performed by NESS, using its knowledge. After gaining sufficient information to specify a complete experiment, NESS runs the simulation and checks for run-time error messages from the operating system.

NESS should then examines the output files created by the simulation model and interpret them for the user in light of his or her major questions. This involves exhibiting plots of model responses and comments on the stability of the proposed system design.

System Architecture

NESS was designed using frames, agendas, menu-inputs and rule-bases, all of which ar facilities provided by GENIE. Frames are one of the basic data structures currently used in AI. Agendas provide the control information necessary for running an expert system. A menu-input stage is used to gather information from the user. Rule-bases containing individual rules store knowledge obtained from a domain expert. The architecture of NESS is illustrated in Figure 2.

As can be seen from Figure 2, NESS consists of five specific functional modules controlled by a top-level Manager. The Model Instantiator obtains initial parameter values from the user, the Simulation Executor runs the simulation model, the Librarian stores and retrieves parameter values from disk files, the Parameter Editor allows the user to edit parameter values, and the Graphics module displays the simulation results. The top-level Manager controls the firing of each of the five modules by means of a forward-chained rule-base. This architecture along with the rule-base control results in a modular, flexible and expandable expert system.

System Implementation

FRANZ LISP provides a number of ways in which a LISP process such as NESS can effect operating system calls. These calls allow NESS to do things like write a disk file containing the parameters that the simulation program needs, cause its execution and read the output files it creates (as illustrated in Figure 1).

ORIGINAL PAGE IS
OF POOR QUALITY

The most straightforward utilization of a system call is simply to include a FRANZ LISP function "exec" (Foderaro (1983)) to cause the execution of a standard UNIX commands, directly in a rule clause. For example, output_display_rb_rule6 checks the precondition that insures that the simulation program has run (self knowledge) and that the user wants to see a plot of theta, which the simulation program would have deposited in a disk file called "theta0plt.stp." The 'then' side of the rule looks as follows:

```
($then (exec cat theta0plt.stp)).
```

This causes the UNIX "cat" command to execute, which simply copies the named file (theta0plt.stp) to the user's terminal.

A slightly more involved method is to write a demon (i.e., a special purpose LISP function) to perform some specific operation which may involve one or more calls to UNIX system functions. For example, a demon named "setup_init_val_in_simula.inp" calls the system function "fileopen," "close," and "cprintf," which performs formatted file write operations. This demon is called by run_rb_rule1.

This rule also calls the demon "start_sim," which uses the FRANZ LISP function "process" to fork a child process, which is the actual execution of the simulation.

Currently NESS and the generic simulation model reside on a VAX 11/785 running under the VMS/EUNICE operating system.

GENERIC SIMULATION MODEL

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 attitude control, consider the pointing control system for the Space Telescope (Dougherty (1982)). The control system must maneuver the telescope through a 90 degree arc in less than 20 minutes, and then maintain a stable line-of-site 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 vehicles's position about a constant direction. The vehicle's dynamics could be represented by nonlinear differential equation during the maneuvering mode; in the tracking mode, the equation could be linearized about the desired operating point. In the initial phase of our project, only the simulation of the vehicle and control system during the tracking mode was 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 (CMGs). Generally, redundancy in sensors and actuators could be a design feature of the control system. For example, four sensors could be positioned to measure the variable in three-dimensional space such that any three of the sensors would provide linearly independent measurement. 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 3. is a simplified illustration of the pointing control system for the Space Telescope. The controller, reaction wheel assemblies, 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, course 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 is the need to use several coordinate frames in defining the vehicles's location and orientation. It is common practice for the vehicles's attitude to be specified by a series of transformations from an inertial frame to frames that are geocentric, defined in the orbital plane, and dependent on orbital shape. In addition, the vehicle's orientation is defined relative to a local vertical frame defined at the vehicle's center of mass and oriented with respect to the orbit normal and local vertical directions. Other reference frames are defined fixed within the vehicle at the location of sensors, actuators and bending modes; these internal frames relate sensor data, generalized forces, and bending deformations, respectively, to the vehicle's dynamic equations.

Transformations are possible between those various coordinate frames (Brady (1982) and 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. Transformation matrices can be defined in terms of roll-pitch-yaw angles between the coordinate frames or in terms applied to spacecraft attitude control problems through the concept of Quaternions (Ickes (1970) and Grubin (1970)). A Quaternion is a four parameter system composed of a vector about which the rotation is to be made and a scalar which is a measure of the angle of rotation.

Model Overview

The generic simulation model shown in Figure 4 represents the simulation of the spacecraft and control system during the tracking mode.

A simple spacecraft attitude control system would have a minimum number 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 identical to each other. These three frames are used in the generic simulation for this research project. The function of the attitude control spacecraft until its coordinate frame becomes identical to the reverence 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 used to compute command signals for the actuators. These signals would command torque from the actuators about an axis defined by the Quaternion Transformation 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.

Implementation Status

Currently we are running NESS with a simplified simulation model, i.e., there are no bending modes, the inertia matrix is diagonal, and the controller is of the PID (proportional-integral-differential) type. The input command is angular attitude. Runge-Kutta and linear multistep integration algorithms are available for performing the numerical integration of the equations of motion. The simplified model (Prototype I) is shown in Figure 5.

SAMPLE SESSION WITH NESS

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 other 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 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. A choice of STEP or FREQUENCY response is offered.

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

CONCLUSIONS AND FURTHER WORK

This paper has illustrated 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 demonstrated the coupling of symbolic processing and numerical computation. That was the purpose of Phase I of this research (Kawamura (1985b)).

In the subsequent phase, the capabilities of both the simulation model and the expert system will be extended. The simulation model will be extended to include actuator and steering distribution equations. Bending modes are being added in the body dynamic equations since they represented a significant concern to the control system designer. The expert system is being extended to assist the user in running a wide variety of simulation models. It will interpret the output data to determine system characteristics such as percent overshoot, settling time, gain margin and phase margin. Ness will also be extended to recommend a suitable series compensator to be added to the simulation model that is required to achieve the desired frequency or time response e.g., achieve a specified overshoot or phase and gain margins.

REFERENCES

[1] De Jong, K., Intelligent Control:

Integrating AI and Control Theory, Proc. Trends and Applications 1983, National Bureau of Standards (1983) 158-161

[2] Kawamura, K., Coupling Symbolic and Numerical Computations, Proc. 1985 IEEE International Conference on Systems, Man and Cybernetics (1985a) 507-510.

[3] Sandell, H., Bourne, J. and Shiavi, R., GENIE: A Generic Inference Engine for Medical Applications, Proc. Sixth Annl. Conf. IEEE Engr. Med. Biol. (1984) 66-69.

[4] DeCeraro, J.K. (ed.), et al., The FRANZ LISP Manual (University of California, Berkely, 1983).

[5] Dougherty, H., et al., Space Telescope Pointing Control System, Journal of Guidance, Control, and Dynamics 5(4) (1982) 403-409.

[6] Brady, M. (ed.), et al., Robot Motion: Planning and Control (MIT Press, Cambridge, 1982).

[7] Paul, R.P., Robot Manipulators: Mathematics, Programming and Control (MIT Press, 1981).

[8] Ickes, B.P., A New Method for Performing Digital Control System Attitude Computation using Quaternions, AIAA Journal 8(1) (1970) 13-17.

[9] Grubin, C., Derivation of the Quaternion Scheme via the Euler Axis and Angle, Journal of Spacecraft 7(10) (1970) 1261-1263.

[10] Kawamura, K., Beale, G., Schaffer, J., Hsieh, B.-J., Padalkar, S., and Rodriguez-Moscoso, J., Research on an Expert System for Database Operation of Simulation/Emulation Math Models, NASA Phase I Final Report Vol. I and II Contract #NAS8-36285, Center for Intelligent Systems, Vanderbilt University. (August, 1985b).

NOTE:

This edited paper describing NESS Phase I appears in its complete form as a chapter in Coupling Symbolic and Numerical Computing in Expert Systems edited by Janusz Kowalik and published by the North-Holland Company in 1986. Phase II of NESS was completed in May of 1986, and it resulted in the development of more general interface specifications allowing NESS to interact with a wider range of digital simulations. Emphasis was also placed on incorporating knowledge specific to the design of series control system compensation yielding a system that assists the control system designer in achieving desired system performance. The results of Phase II are documented in the Final Report to MSFC on Contract #NAS8-36285, Center for Intelligent Systems, Vanderbilt University (May 1986).

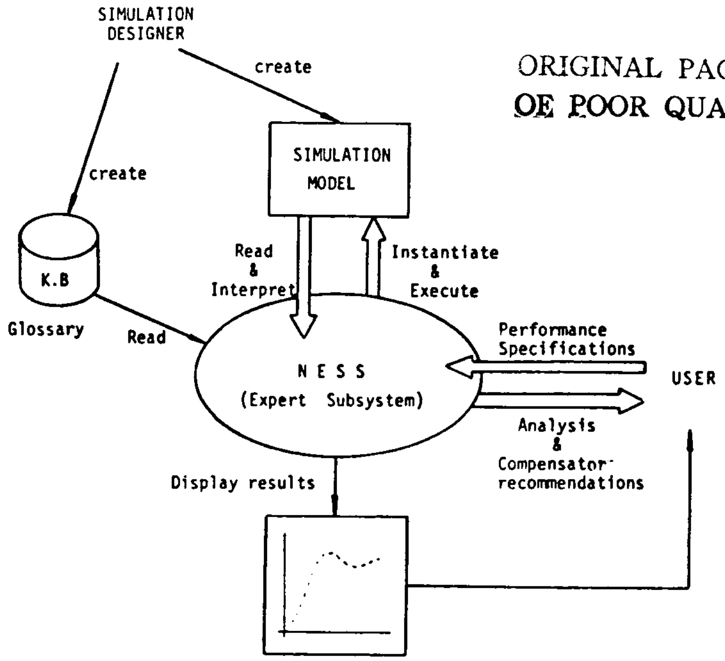

Figure 1 User Interaction with the Coupled Expert System

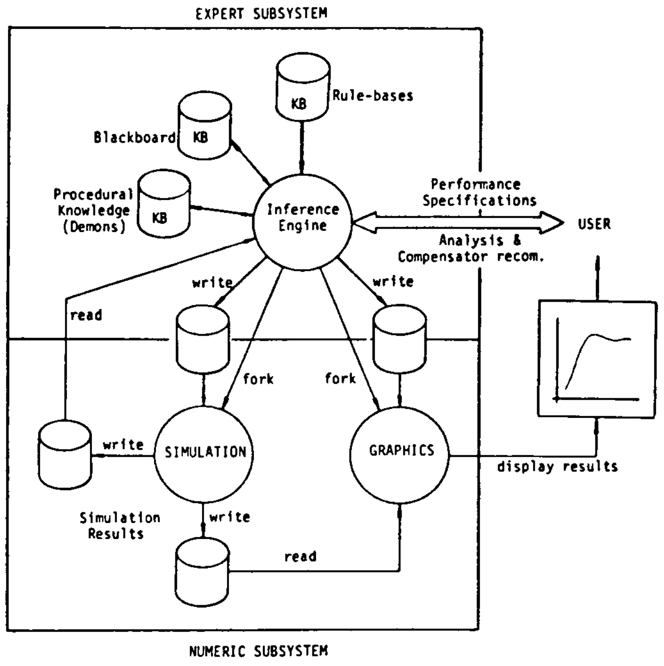

Figure 2 NESS Architecture

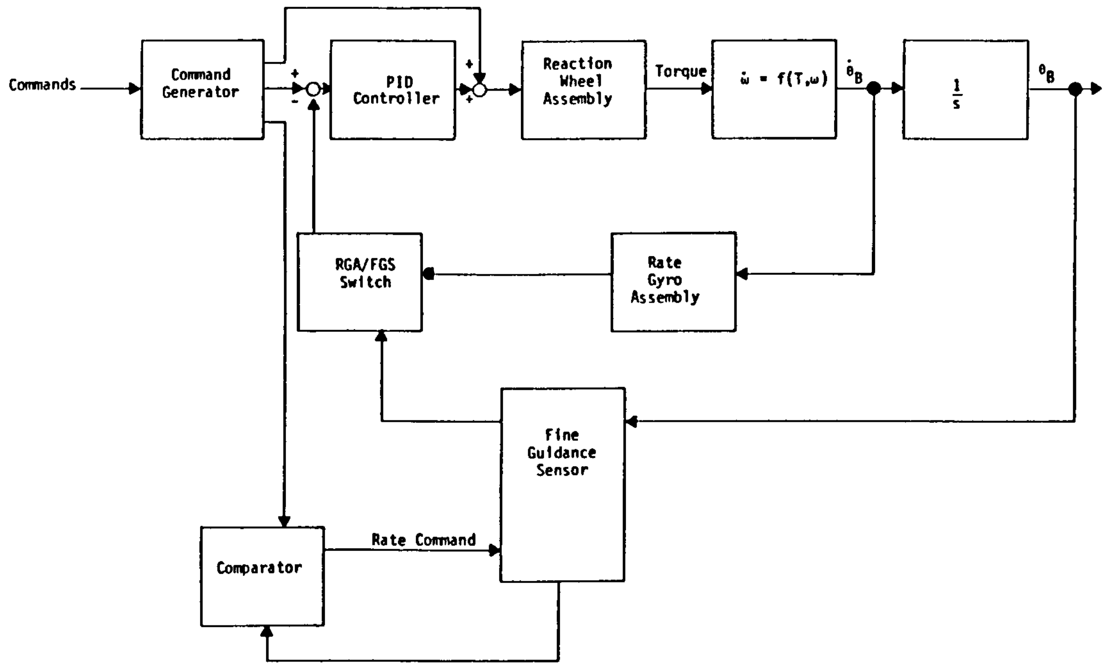

Figure 3 Simplified Attitude Control System

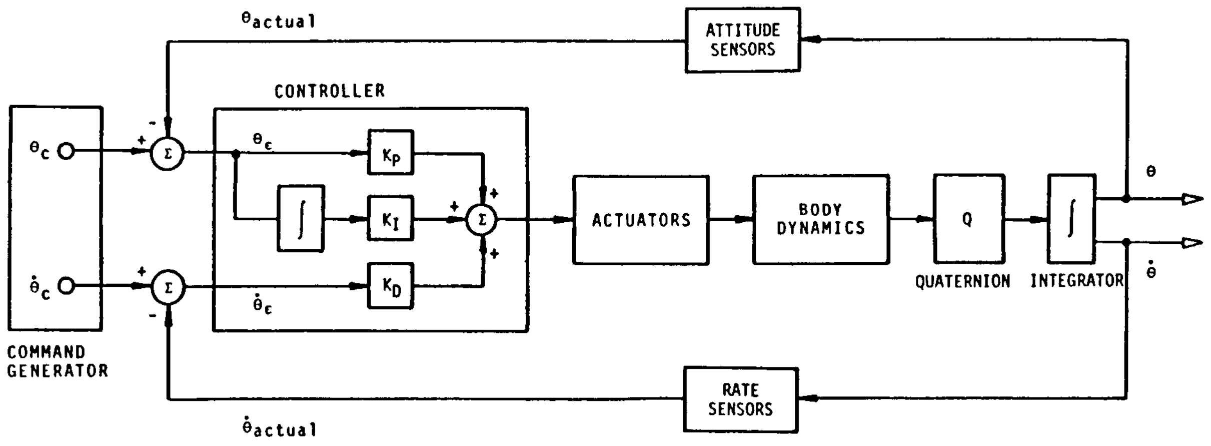

Figure 4 Block Diagram of the Generic Simulation Model

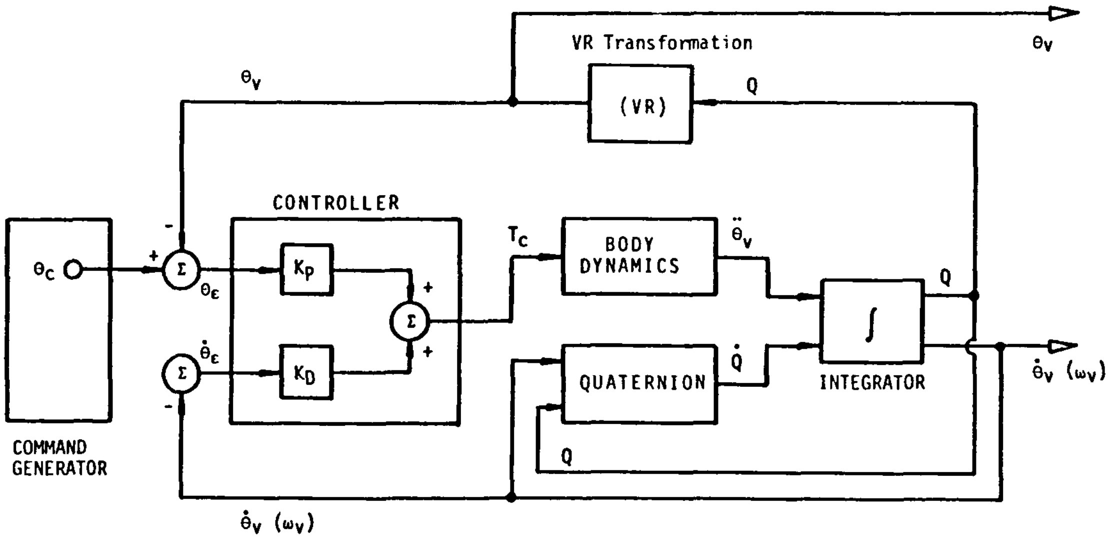

Figure 5 Simplified Diagram of the Generic Simulation Model