1989
NASA/ASEE SUMMER FACULTY RESEARCH FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

YAW RATE CONTROL OF AN AIR BEARING VEHICLE

Prepared by:	Bruce L. Walcott, Ph.D
Academic Rank:	Assistant Professor
University and Department:	University of Kentucky Deparment of Electrical Engineering
NASA/MSFC:	
Laboratory:	Flat Floor Facility
Division:	Information and Electronic Systems
Branch:	Control Electronics
MSFC Colleagues:	E.C. Smith and Frank Nola
Date:	August 11, 1989
Contract No.	The University of Alabama in Huntsville NGT-01-008-021

YAW RATE CONTROL OF AN AIR BEARING VEHICLE

by

Bruce L. Walcott
Assistant Professor of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046

ABSTRACT

This report summarizes the results of a 6 week project which focused on the problem of controlling the yaw (rotational) rate the air bearing vehicle used on NASA's flat floor facility. Contained within is a listing of the equipment available for task completion and an evaluation of the suitability of this equipment. This report also details the identification (modeling) process of the air bearing vehicle as well as the subsequent closed-loop control strategy. The effectiveness of the solution is discussed and further reccomendations are included.

ACKNOWLEDGEMENTS

The author expresses his deepest appreciation and gratitude for the warmth and hospitality extended by his adopted family, the friendly people of EB24. In particular, he wishes to thank his principal colleagues, E.C. Smith Frank Nola for their advice, supervision, encouragement and mentoring. The author also wishes to acknowledge Bill Jacobs for his assistance and expertise concerning the air bearing vehicle and Tom Bryan for his advice and midnight-oil-burning chats. A special thanks is extended to Cindy Coker and Harry Reid for their discussions and advice on the two projects the author worked on during the four weeks following the completion of the project outlined in this report. The author is grateful to Ricky Howard, Mike Book, Charles Oliver and Tom Sutherland for the use of their equipment (and for challenging homework problems!). Of course, Andy Gamble, Zack Barnett, Jon McInroy and Kirk Smith are to be awarded the highest commendations for making exiled life in 4619 bearable. Lastly, Dr. Gerald Karr and Dr. Frank Six are to be congratulated on their outstanding seminars and superb supervision of the 1989 ASEE/NASA MSFC Summer Faculty Fellowship Program. Great job guys!

1 INTRODUCTION

The Marshall Space Flight Center in Huntsville, Alabama is home to the largest precision flat floor facility in the world. This 4200 square foot floor is constructed of self-leveling black epoxy and is flat to with 1/1000 of an inch over any given square yard and to within 3/1000 of an inch from corner to corner. The featured player on this ebony stage is a 4400 pound cubicle vehicle built from NASA's own design. This vehicle rides 6/1000 of an inch above the flat floor on a cushion of air supplied by three air bearings. Satellite and Orbital Maneuvering Vehicle mock-ups are mounted to the front of the vehicle in order to perform docking and rendezvous simulations. Lighting conditions can be completely controlled on the floor plus the vehicle can transmit video and telemetry via an RF link. Unstable satellites can be simulated using the eight degree of freedom dynamic overhead target simulator (DOTS). Thus, even the most difficult OMV docking problems can be simulated using the vehicle and the DOTS. A total of 24 air thrusters are mounted on the lower four corners of the vehicle to enable the vehicle to move in X, Y and Yaw directions while servo motors on the front of the vehicle give mock-ups the added capabilities of Z, Roll, and Pitch. Thus, a mock-up on the vehicle has a total of six degrees of freedom.

In the past, prestigious high-tech corporations such as General Electric, TRW, and Martin Marietta have utilized the flat floor and the air bearing vehicle to develop and validate docking and rendezvous strategies as well as to investigate contact dynamic problems. These companies have future commitments to the flat floor facility thereby making continual operation and functionality enhancement high priorities of NASA.

2 OBJECTIVES

The following is a list of objectives which must be attained in order to complete the project:

1. Develop an assembly language driver for the data acquisition board. This driver should be capable of interfacing with a C language program.

2. Mount a static inverter on the air bearing vehicle
3. Become familiar with Microsoft 5.1 C.
4. Mount CMGs and rate gyros on the vehicle.
5. Connect torquer motor servo input to D/A output of data acquisition board and rate gyro output to A/D input.
6. Model the open loop system consisting of the input to the torquer servo and the output of the rate gyros. Initially, this identification will be performed off-line. If an adaptive control is required, this identification will be done on-line as part of the control loop.
7. Develop a control law based on the results of the identification scheme. Initially, this control law will be developed off-line but should have the capability of being developed on-line should an adaptive control scheme be required.
8. Implement and test the control law.
9. Devise and implement a scheme for desaturating the CMGs.

3 PROBLEM STATEMENT

3.1 Motivation

The results of earlier research and experimentation demonstrated that the use of the air thrusters to control translational (X and Y) and rotational (Yaw) motion on the vehicle is inefficient. This is due to the high degree of coupling which exists between the thrusters dedicated to translational movement and the thrusters responsible for rotational motion. That is, a translational motion from the thrusters introduced a rotational component and vice-versa. Thus, the thrusters act opposedly rather than separately in controlling the vehicle. Consequently, finding a decoupled control law or suitable alternative source of motion for the vehicle is of urgence to NASA.

3.1.1 Control Moment Gyros

One possible solution to the coupled control problem detailed above is to replace the Yaw thrusters by control moment gyros (CMGs). The theory behind the use of CMGs to control Yaw can be summarized by Newton's second law of motion as applied to rotational systems. Newton's second law states that the rate of change of momentum of a body is equal to the sum of the external torques acting upon it. That is,

$$\sum T_{external} = \frac{dH}{dt}$$

where H is the angular momentum of the gyro and $T_{external}$ are the external torques. A CMG operates in the exact opposite manner as an inertial guidance gyro. In an inertial guidance gyro, a torque is applied to the gyro when the body changes direction. This torque produces a rate perpendicular to the gyro's spin axis, which is measured. Conversely, in a CMG a rate is produced perpendicular to the gyro's spin axis by a torquer motor. This rate causes a change in the angular momentum which produces a torque perpendicular to the gyro's spin axis.

To illustrate, consider Figure 1. Figure 1 shows a one degree of freedom

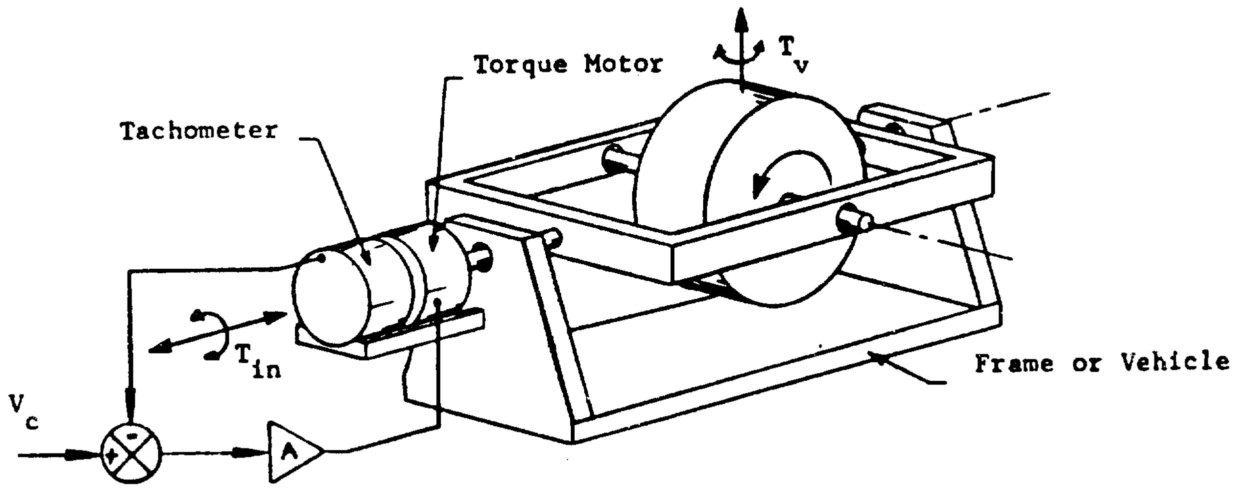


Figure 1 - Single Degree of Freedom CMG

CMG [1]. The servo torquer motor produces a rate ω as shown. This rate yields a torque, T_v given by

$$T_v = \omega \times H$$

If we assume that the magnitude of the angular momentum is constant (i.e., the gyro is spinning at a constant rate), then the component of torque produced in the horizontal plane is

$$T_{yaw} = H\omega cos\theta \tag{1}$$

where θ is the angle formed between the spin axis of the gyro and the horizontal plane in which the vehicle travels.

Therefore, as a solution to the coupling control problem, a single degree of freedom CMG could replace the yaw thrusters.

4 PROPOSED SOLUTION

4.1 Available Equipment and Specifications

Hardware which can be dedicated to the project includes:

- Two Sperry reaction wheels and brushless DC torquer motor.
- One Apollo Telescope Mount (ATM) rate gyro rated at 1 deg/sec
- Two Spring-driven rate gyros rated at 60 deg/sec
- One TELEX 1280 PC-AT clone operating at 12 MHz
- One IBM PC Data Acquisition and Control Adapter
- One Avionics Instruments Static Inverter

Software which is available for the project includes:

- Microsoft C 5.1
- Microsoft Assembler 5.0

- Matlab 3.13 and the Control and System Identification Toolbox

To determine if the hardware available is sufficient to solve the problem, let us consider the yaw specifications of the OMV mock-up which is the most popular mock-up on the vehicle. The proposed OMV will be able to achieve a maximum yaw rate of 2.3 deg/sec. If we consider the vehicle as a uniform rectangular solid with a mass of 2000 kg and the dimensions 1.5 meters wide by 2 meters long by 2 meters tall, then we may calculate the moment of inertia by

$$I_{vehicle} = \frac{2000[(1.5)^2 + (2.0)^2]}{12} = 1041 kgm^2 \qquad (2)$$

Next, let us note that the maximum rate of the available DC brushless torquer motor is approximately 1.0 rad/sec. Recall from equation (1) that the torque supplied by the CMG varies with $\cos\theta$ where θ is the angle formed between the gyro's spin axis and horizontal. To estimate the torque required to meet the OMV specifications, let us assume that the torquer motor turns the gyro at a constant rate equal to the motor's maximum rate (i.e., 1.0 rad/sec). We will further assume that θ is allowed to vary from 0 radians to $\frac{\pi}{2}$ radians. Neglecting the dynamics of the torquer, the time necessary to turn the gyro from 0 to $\frac{\pi}{2}$ radians is $\frac{\pi}{2}$ seconds. Therefore, the maximum torque required to achieve the OMV specification is estimated at

$$T_v = \frac{1041 \times 2.3 \times (\frac{\pi}{180})}{\frac{\pi}{2}} = 26.6 Nm$$

Thus, the angular momentum required of the gyro is

$$H = \frac{T_v}{\omega(cos\theta)_{ave}} = \frac{26.6}{1.0 \times \frac{2}{\pi}} = 41.8 kgm^2/s$$

where $cos\theta_{ave}$ is the average value of the $\cos\theta$ as the torquer turns the gyro from 0 to $\pi/2$ radians. The reaction wheels are rated at an angular momentum of 11.4 kgm^2/s at a speed of 1500 rpm. Recall that the angular momentum of a rotating cylinder is $mr^2\omega$. Thus, to satisfy the rate demanded by the OMV specification plus a safety margin, two Sperry reaction wheels running at 3000 rpm must be used in the CMG design. This configuration yields a total angular momentum of 45.6 kgm^2/s, more than the required 41.8 kgm^2/s. The reaction wheels were mounted at opposite ends of the torquer motor shaft with the two spin axes parallel to each other and perpendicular to the motor shaft. Thus, each reaction wheel behaves as a single degree of freedom gyro. Although the gyros can be coupled at any point on the vehicle and still transmit their maximum torque to the vehicle, they were mounted as close to the center of mass as possible to minimize any unbalance the gyros might introduce.

Since a variety of mock-ups and payloads will be used on the air bearing vehicle, an adaptive control strategy may be required to adequately replace the yaw control. In addition, the CMGs supplying the torque to the vehicle will have to be desaturated intermittently. That is, when θ nears 90 deg, the amount of torque which the CMG's can supply is nearly zero and if θ exceeds 90 deg, the horizontal component of the torque will switch directions. Consequently, the problem solution should include a scheme whereby the CMGs can be returned to $\theta = 0$ without affecting the rate of the vehicle.

4.2 Restatement of Objectives

The following is a restatement of the objectives (tasks) which must be attained in order to complete the project:

1. Develop an assembly language driver for the data acquisition board. This driver should be capable of interfacing with a C language program.

2. Mount a static inverter on the air bearing vehicle.

3. Become familiar with Microsoft 5.1 C.

4. Mount CMGs and rate gyros on the vehicle.

5. Connect torquer motor servo input to D/A output of data acquisition board and rate gyro output to A/D input.

6. Model the open loop system consisting of the input to the torquer servo and the output of the rate gyros. Initially, this identification will be performed off-line. If an adaptive control is required, this identification will be done on-line as part of the control loop.

7. Develop a control law based on the results of the identification scheme. Initially, this control law will be developed off-line but should have the capability of being developed on-line should an adaptive control scheme be required.

8. Implement and test the control law.

9. Devise and implement a scheme for desaturating the CMGs.

Both of the above remarks indicate that the control solution should be completed on a microprocessor or micro-computer. Since a Telex PC-AT clone and data acquisition board are available for this project, the decision was made to design the control law around a micro-computer.

5 PROJECT SUMMARY

The first two weeks of the project were devoted to developing an assembly language driver for the IBM data acquisition board which could be interfaced with a C language control program. This time was also used to learn the peculiarities of Microsoft C 5.1 which differs significantly from Turbo C 2.0 used at the University of Kentucky.

Fortunately, NASA engineer Charles Oliver had already developed an assembly language driver for the IBM data acquisition board which had been modified by NASA engineer Bill Jacobs to interface to Microsoft Quick BASIC programs. However, as was painfully discovered during the initial two weeks of the project, calling and return conventions are entirely different in C programs than in BASIC programs. BASIC programs pass the long address (segment and offset) of pointers to the arguments of the subroutine on the stack while C programs pass the actual pointers themselves on the stack. Furthermore, BASIC programs expect the return values to be passed on the stack while C programs expect the return values of functions to be stored in the AX register.

The source listing for the assembly language driver is contained in Appendix A.

After the completion of the assembly language driver, the ensuing two weeks were dedicated to the installation of the static inverter and to obtaining a model for the vehicle with the input being the voltage to the torquer servo loop and the output being the voltage out of the rate gyros measuring the yaw of the vehicle. The static inverter was installed with the invaluable aid of Bill Jacobs. The inverter, which is of aircraft quality, converts the 28 volt DC power present on the vehicle to 115 volts 60 Hz AC signal thereby enabling the PC-AT clone with the data acquisition board to be mounted on the vehicle.

In the initial stages of modeling the system, it was discovered that the spring driven gyros rated at 60 deg/sec worked reasonably well in lieu of a tachometer in the torquer servo loop. Here, rates of up to 1 rad/sec (57 deg/sec) are being measured. The maximum yaw rate of the vehicle is 2.3 deg/sec, however, which is still within the noise level of the spring-driven gyros. Consequently, the decision was made to use the ATM rate gyro to measure the yaw of the vehicle. This gyro saturated at 1 deg/sec which corresponded to an output of 6.6 volts (after a voltage division circuit), but gave an extremely clean and linear signal up to that point. Yet, the relatively low saturation level inhibited testing of the vehicle at yaw rates above 1 deg/sec. The decision was made to continue with the modeling procedure using the ATM gyro's output, but also to devote some thought to how one might increase the saturation level on the ATM.

Since an adaptive control strategy may be required to control the yaw rate of the vehicle, the identification (modeling) scheme must possess the capability of being implemented on-line. Consequently, the identification procedure purported by Ljung [2], Eykhoff [3], Astrom [4], Soderstrom and Stoica [5] and others was selected. In brief, this identification scheme consists of three parts: selection, application, and recording of data; selection of possible candidate models; and determination of the best-suited model. Once the structure of the model is known, model parameters can be estimated using a criterion such as least-squares.

5.1 Data Collection

In sampling data using our PC-AT clone data acquisition system, we must select a sampling interval which will avoid the effects of aliasing. Furthermore, since our application involves not just identification but control as well, we do not want to sample too fast which might lead to a non-minimum phase model and/or a model with a delay of many sampling periods. Thus,

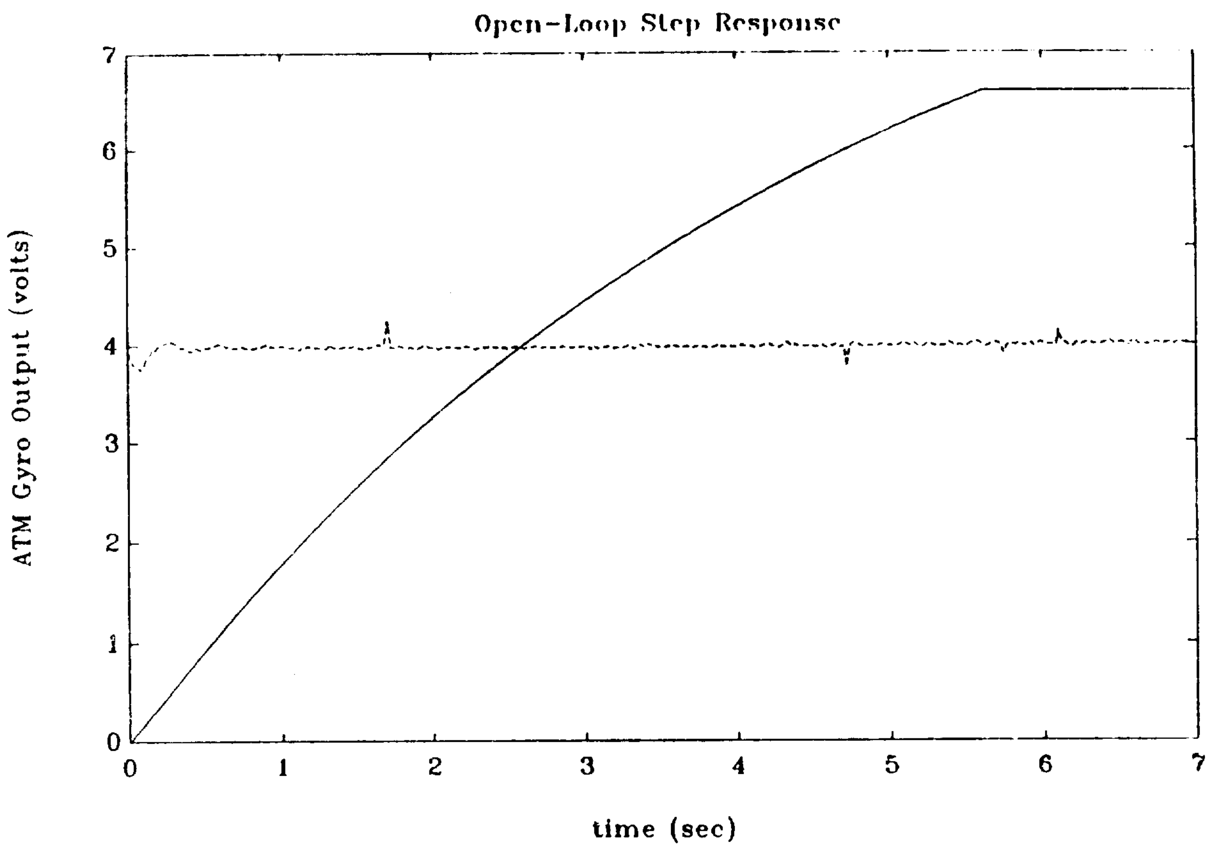


Figure 2 - Open-Loop Step Response of the Vehicle

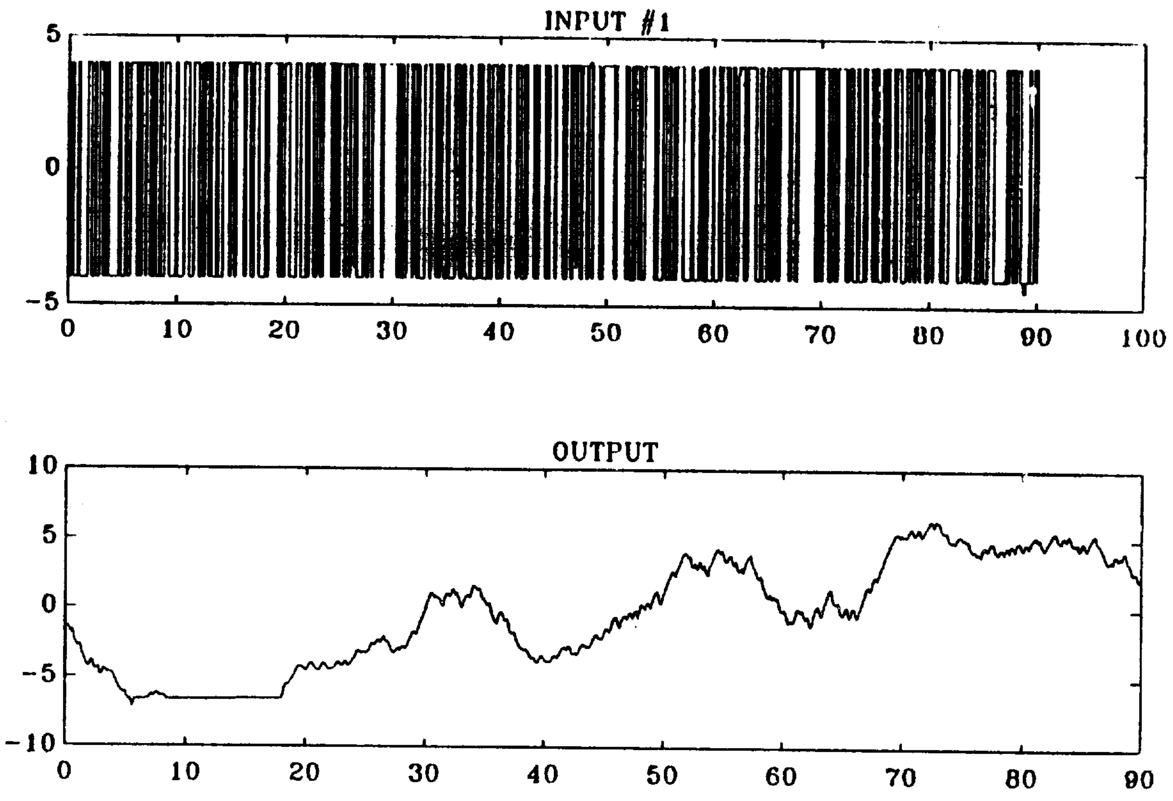


Figure 3 - Random Binary Input and Corresponding Output of ATM Rate Gyro

the choice of the sampling time, T, should be less than the dominant time constants of the system, but not significantly.

To obtain an estimate of a reasonable sampling time, the open-loop step response was found with the input being the voltage into the torquer-servo loop and the output being the voltage from the ATM gyro which is proportional to the rate of the vehicle until saturation. If the input were a step torque supplied from the gyro (i.e., if T_v were a step), we would expect the step response to produce some sort of ramp (assuming the vehicle is riding on a frictionless surface). However, since the input is a step voltage to the torquer servo, we would expect such an input to produce a constant precession rate, ω, in the gyros (neglecting the dynamics of the servo loop). The torque applied to the vehicle will not be constant, since equation (1) shows us that this torque varies with the cosine of θ. The results of the step response are displayed in Figure 2. All experiments on the vehicle were conducted at a setting of 50 psi on the air bearings. As can be seen from Figure 2, the step response is a cross between a type 0 step response and a type 1 step response. Although no actual dominant time constant can be measured, from the overall slow response depicted in Figure 2, a sampling time of about 150 milliseconds should suffice.

Upon selection of a reasonable sampling time, the system must be excited by a random signal which excites all of the modes of the system. Such an input is white noise. Since we are using a discrete magnitude-limited input, we cannot generate a true white noise signal. However, a binary input which switches values with a probability of 0.5 has the same effect as white noise (see Ljung (1987)). A magnitude of 4 volts was selected for the random binary signal. The results of the application of the random binary input as well as the input itself can be seen in Figure 3. A total of 600 data points were taken at a sampling interval of 0.15 sec.

Following the data collection, the next step in the identification process is to develop a list of possible model structure candidates and to pose a criterion for selecting the best-fitting structure. From the step response shown in Figure 2 and from the equations of motion used to estimate the torque required of the CMG, a linear model would be a reasonable choice to represent the vehicle. The particular linear model structure candidate was chosen to be an Auto-Regressive Exogeneous structure (ARX) with unknown parameters. The notation ARX(m,n,p) represents the following

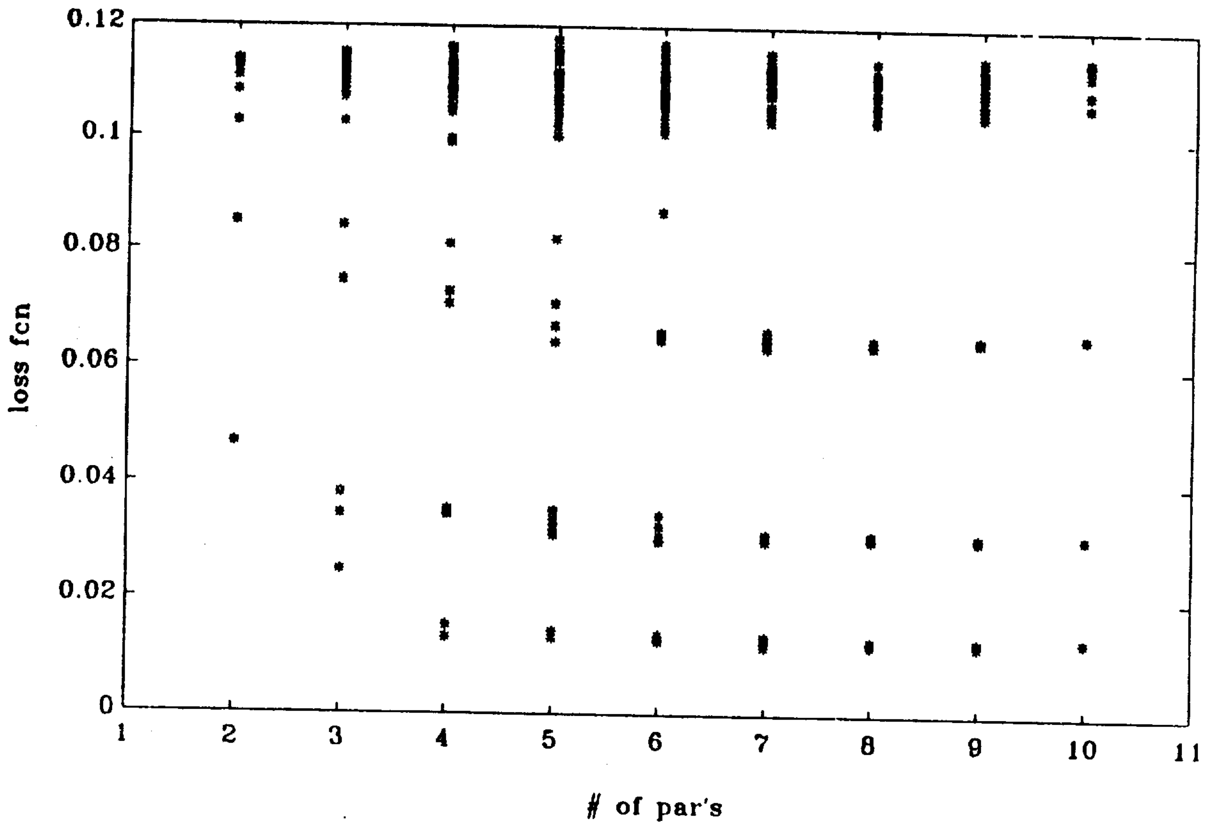


Figure 4 - Loss Function V for 250 ARX Candidates

input-output relationship:

$$y(t)+a_1y(t-T_s)+...+a_ny(t-nT_s) = b_1x(t-pT_s)+...+b_mx(t-(p+m-1)T_s) \tag{3}$$

where T_s is the sampling time. That is, the transfer function $\frac{Y(q)}{X(q)}$ has n poles, m-1 zeros and a time delay of pT_s. The parameters a_i and b_i are the parameters to be estimated. The ARX structure was chosen over a ARMAX (auto-regressive with a moving average and exogeneous term) because the disturbances on the system are not prominent. Furthermore, calculating the control law is simpler with an ARX model than an ARMAX model.

Once the model structure has been decided upon, a criterion for determining the best fitting ARX structure is needed. That is, the best combination of number of delays, the order of the numerator, and the order of the denominator needs to be evaluated. Also, the data must be divided into two categories: data for identification and data for validation. In other words, a portion of the data must be used to identify the best fitting ARX structure and then the remainder of the data should be used for cross verification purposes. To this end, the data was divided into two equal portions: the first three hundred points were used for estimation purposes while the final three hundred data points were employed for verification purposes.

In trying to identify a criterion for selecting the best-fitting ARX structure, two closely-related strategies predominate. The first is the Akaike Final Prediction Error (FPE) criterion where the following entity is desired to be minimized:

$$FPE = \frac{1+n/N}{1-n/N}V$$

where n is the total number of parameters to be estimated, N is the number of data points, and V is the quadratic loss function for the particular structure under scrutiny (see Ljung (1987)). The second criterion, called Akaike's Information Theoretic Criterion (AIC), is closely related to the FPE criterion. In the AIC, the following quantity is minimized:

$$AIC \approx log[(1+2n/N)V]$$

The FPE criterion is selected for our identification problem.

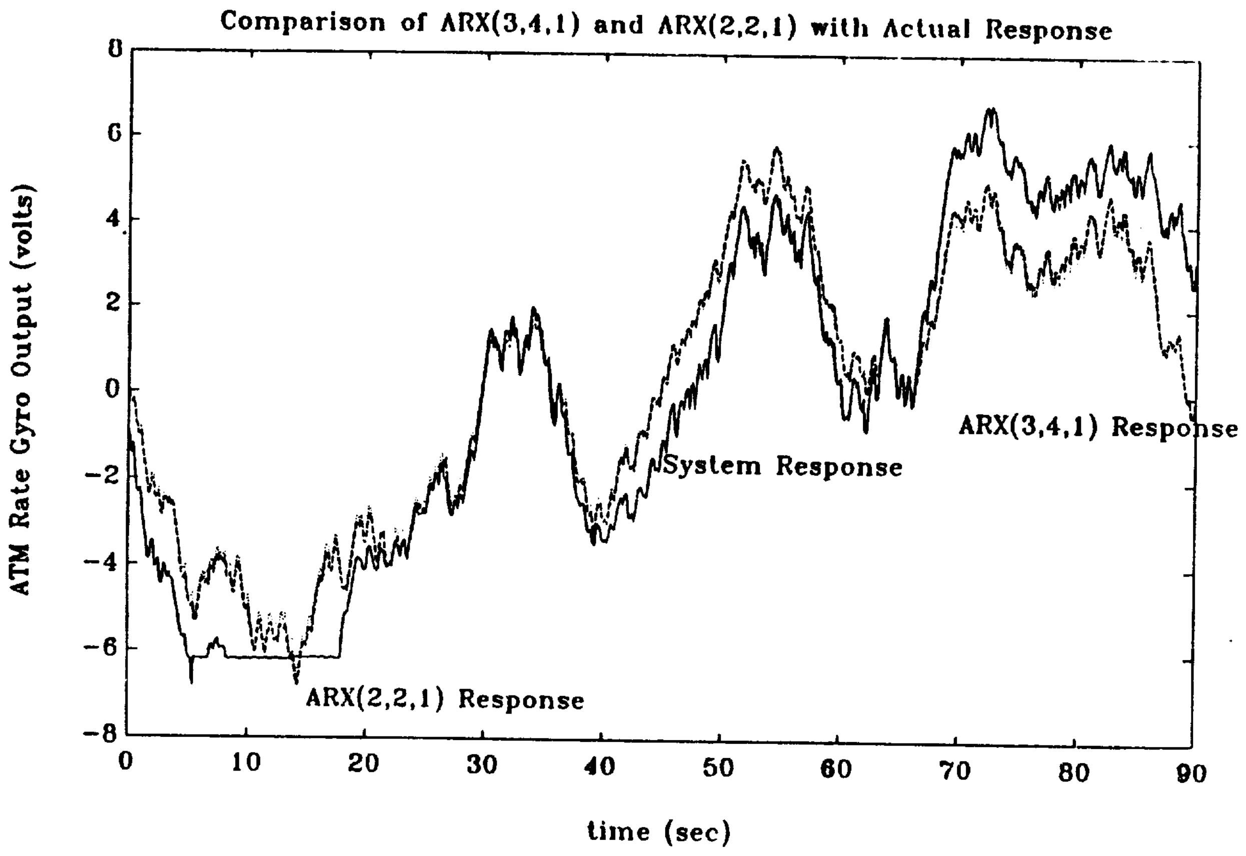


Figure 5 - Comparison of Response of the Two ARX Models to the Actual Response

Figure 3 depicts the binary input which was applied to the vehicle's torquer servo and the corresponding response. In the structure selection ARX(m,n,p) candidates were considered where m was varied from 1 to 5, n was varied from 1 to 5, and p was varied from 1 to 10. That is, a total of 250 structures were considered by calculating the FPE for each structure. The results of this examination are shown in Figure 4. The structure candidate which achieved the lowest overall FPE was an ARX(3,4,1) model with an FPE = 0.0242 and a loss function of V = 0.0231. This implies that the best model for the system (out of the 250 models considered) is

$$y(t)+a_1y(t-T_s)+\ldots+a_5y(t-5T_s) = b_1x(t-T_s)+b_2x(t-2T_s)+b_3x(t-3T_s)$$

Using a least-squares criterion, the 7 unknown parameters were estimated to be

$$a_1 = -1.0280, a_2 = 0.1515, a_3 = -0.2828, a_4 = 0.1633$$

and

$$b_1 = 0.0319, b_2 = 0.0548, b_3 = -0.0167$$

If we were concerned solely with identification, we would stop our search with the ARX(3,4,1) model. However, we are concerned with control as well as identification and a fourth-order model does not readily lend itself to control design, especially on-line computation. Thus, we should look for a lower order model which has an FPE comparable to the minimum of 0.0242. To this end we found that an ARX(2,2,1) structure has an FPE of 0.029 and a loss function of V = 0.0283 (compare to 0.0231). This second-order ARX structure is far preferable to the best-fitting ARX(3,4,1) model from a control perspective. An ARX(2,2,1) model has the input-output relationship:

$$y(t) + a_1y(t - T_s) + a_2y(t - 2T_s) = b_1x(t - T_s) + b_2x(t - 2T_s)$$

Using a least-squares criterion, the 4 unknown parameters were estimated to be

$$a_1 = -0.8025, a_2 = -0.1925, b_1 = 0.0320, b_2 = 0.0617 \qquad (4)$$

To further illustrate that the ARX(2,2,1) model adequately represents the response of the vehicle, consider Figure 5 which depicts the actual response of the system to the binary input as well the responses of the

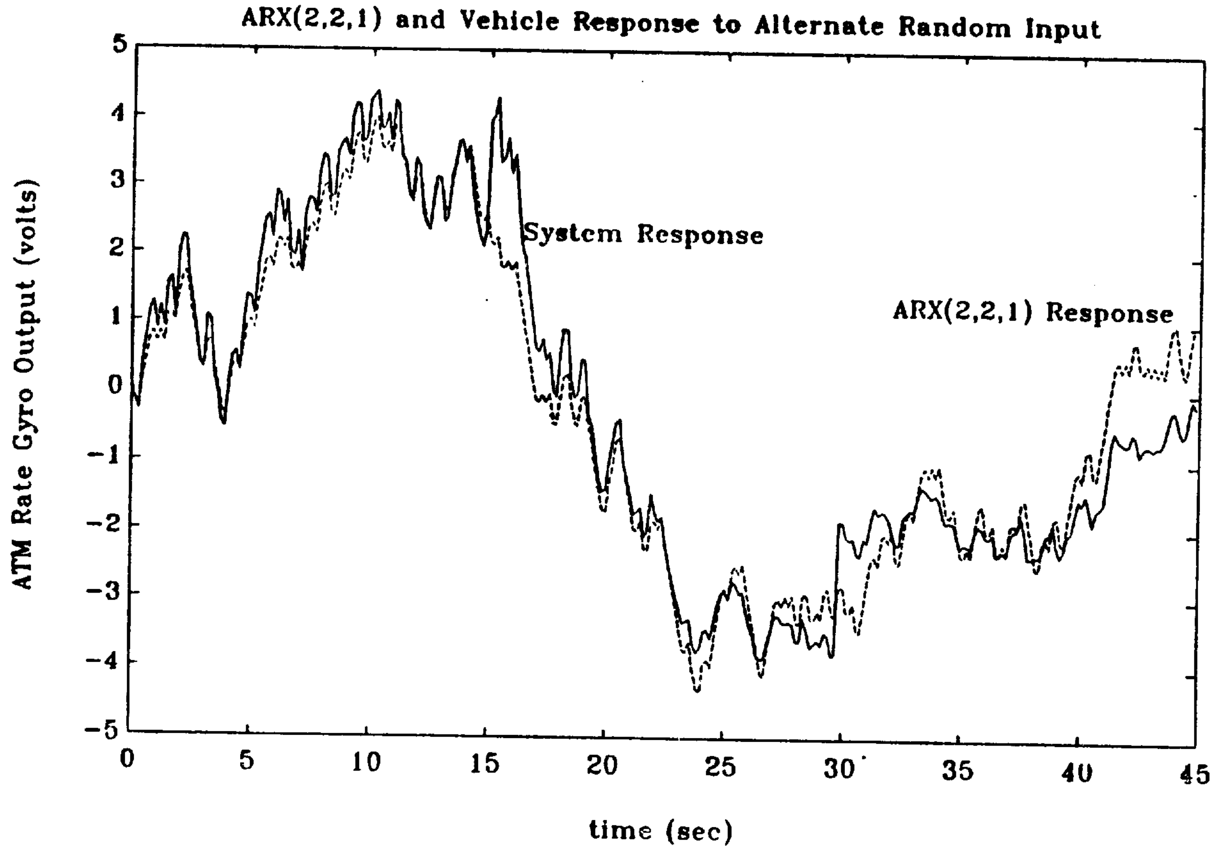


Figure 6 - Response of ARX(2,2,1) Model and Vehicle to Alternate Binary Input

best-fitting ARX(3,4,1) model and the ARX(2,2,1). As can be seen from Figure 5, there is very little difference between the two ARX responses and both closely follow the actual response of the vehicle. Hence, by selecting the ARX(2,2,1) model over the ARX(3,4,1) structure, we do not sacrifice much fidelity, yet gain a reduction in model order of one-half.

Figure 6 contains a final testament to the accuracy of the ARX(2,2,1) representation of the vehicle. Figure 6 shows the response of the vehicle to a random binary input different from that which was used to identify the ARX model. Also contained in Figure 6 is the response of the ARX(2,2,1) model to the same input. Note how closely the model response follows the vehicle response. Not only do the results displayed in Figure 6 indicate that the ARX(2,2,1) model adequately represents the system, but since these results were recorded on a date different from the data used to identify the system, they also suggest that an on-line identification scheme may not be necessary. That is, the dynamics of the system appear not to vary greatly with time (although no payload was attached), consequently the ARX parameters given in equation (3) accurately represent the system for all time t. This hypothesis will be true especially if our control law is sufficiently robust.

5.2 Control Design Considerations

Once the proper model for the vehicle has been selected, a closed-loop control law must be designed. The specifications for such a control are

1. Near zero steady-state error due to a step command
2. Settling time of less than 3 seconds
3. No overshoot
4. To be sufficiently robust so that an adaptive scheme is not necessary or to have the ability to be computed on-line

The settling time criterion stems from the 3 second time delay which is inherent in the OMV communication link when the OMV is piloted from the ground.

ORIGINAL PAGE IS
OF POOR QUALITY

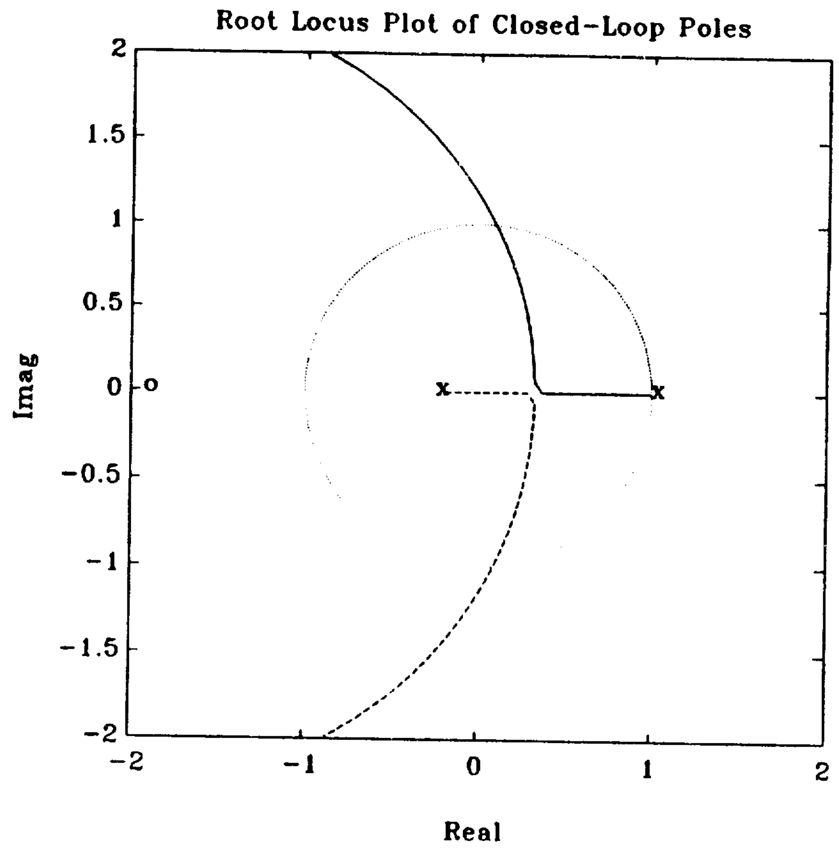


Figure 7 - Root Locus Plot of the Closed-Loop Poles

To address the first design specification, let us return our attention to Figure 2 which shows the open-loop step response of the vehicle. Note that the response does not attain a steady-state value and appears to ramp at a constant rate as t becomes large. This information suggests that the system type is greater than zero and that the closed-loop uncompensated step response would result in zero steady-state error. Therefore, no integrator is necessary in the feedback loop to satisfy the steady-state error specification.

To meet the settling time criterion, the poles in the s-plane must be to the left of $\sigma = -3/4$. Therefore, in the Z-plane the magnitude of the closed-loop poles must obey

$$|poles| <= |e^{-3/4T_s}| = 0.8936$$

A root locus plot of the ARX(2,2,1) model is contained in Figure 7. To make the control design sufficiently robust without adding overshoot to the response, the value of the feedback gain is chosen such that the system is critically damp. That is, both poles are located at $\sigma = 0.3229$, $j\omega = 0$ which is well within the magnitude limit of 0.8936 imposed by the settling time specification. These closed-loop poles yield a theoretical settling time of

$$t_s = T_s \frac{ln(0.3229)}{ln(0.02)} = 0.5191 sec$$

Such a large margin may appear at first to be overly conservative but if we drive our D/A output into saturation (± 10 volts), the settling time will probably be longer than this theoretical value.

The value of the feedback gain which yields a critically-damped system can be computed on-line from the values of the ARX parameters, a_1, a_2, b_1, b_2. One can readily show that

$$k = -[\frac{(2a_1b_1 - 4b_2)^2 + [(2a_1b_1 - 4b_2)^2 - 4b_1^2(a_1 - 4a_2)]^{1/2}}{2b_1^2}] \qquad (5)$$

For the values given in equation (4), equation (5) evaluates to

$$k = -4.9642$$

Although the above feedback gain causes our ARX off-line system model to become critically damped, if we choose to use an on-line parameter identification scheme, then equation (5) can easily be used on-line to calculate the necessary feedback gain.

5.3 Control Validation

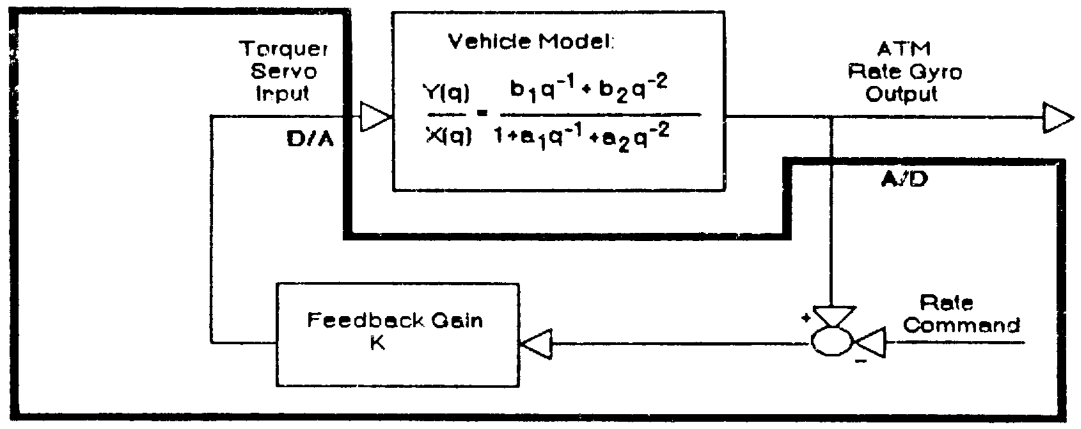
To solve the problem of the ATM saturating at 1 deg/sec NASA engineers Hugo Berry and Bill Jacobs suggested that we use equation (1) to our advantage. That is, if the spin axis of the ATM gyro is tilted by an angle ϕ then the amount of yaw rate measured by the gyro in the plane of the vehicle will decrease by the $cos\phi$. Thus, letting $\phi = 75°$, the ATM gyro saturates at a value of $1 \times \frac{1}{cos 75} = 3.864$ deg/sec rather than 1 deg/sec and the OMV maximum rate of 2.3 deg/sec can be easily realized. However, we have effectively decreased the forward loop gain by a factor of 3.864. Therefore, the critically-damped feedback gain listed in equation (5) must be multiplied by 3.864 to compensate which yields

$$k = -19.1802$$

Furthermore, spin axis of the ATM rate gyro is no longer parallel to the Earth's spin axis which implies that some variable rate disturbance will be introduced into the system depending upon the alignment of the vehicle on the flat floor. Although tilting the spin axis of the ATM gyro offers an inexpensive short-term remedy to the problem of low rate saturation, it is recommended that a more permanent solution be found either in the form of a different rate gyro, or in the form of a more effective modification of the current ATM rate gyro.

Appendix B contains two programs developed by Dr. Walcott during his 1989 tenure as an ASEE/NASA Summer Faculty Fellow. The first program closes the loop on the vehicle using the feedback control given in equation (5) and the computer control configuration shown in Figure 8. The desired output of the ATM gyro in volts (which is directly proportional to the yaw rate of the vehicle) is entered by the user via keyboard. This serves as the demand signal and is combined with the inverted ATM gyro output to form the error signal which is the input to the proportional control block. The output of the proportional control block is the actuating signal to the vehicle. The second program utilizes the same control strategy but the input demand can be varied via the user by typing a '1' to increase the demand signal by 1/8th of a volt and '2' to decrease the demand signal by the same amount.

The results of the first program are shown in Figure 9. For purposes



Micro-Computer Control

Figure 8 - Micro-Computer Control Block Diagram

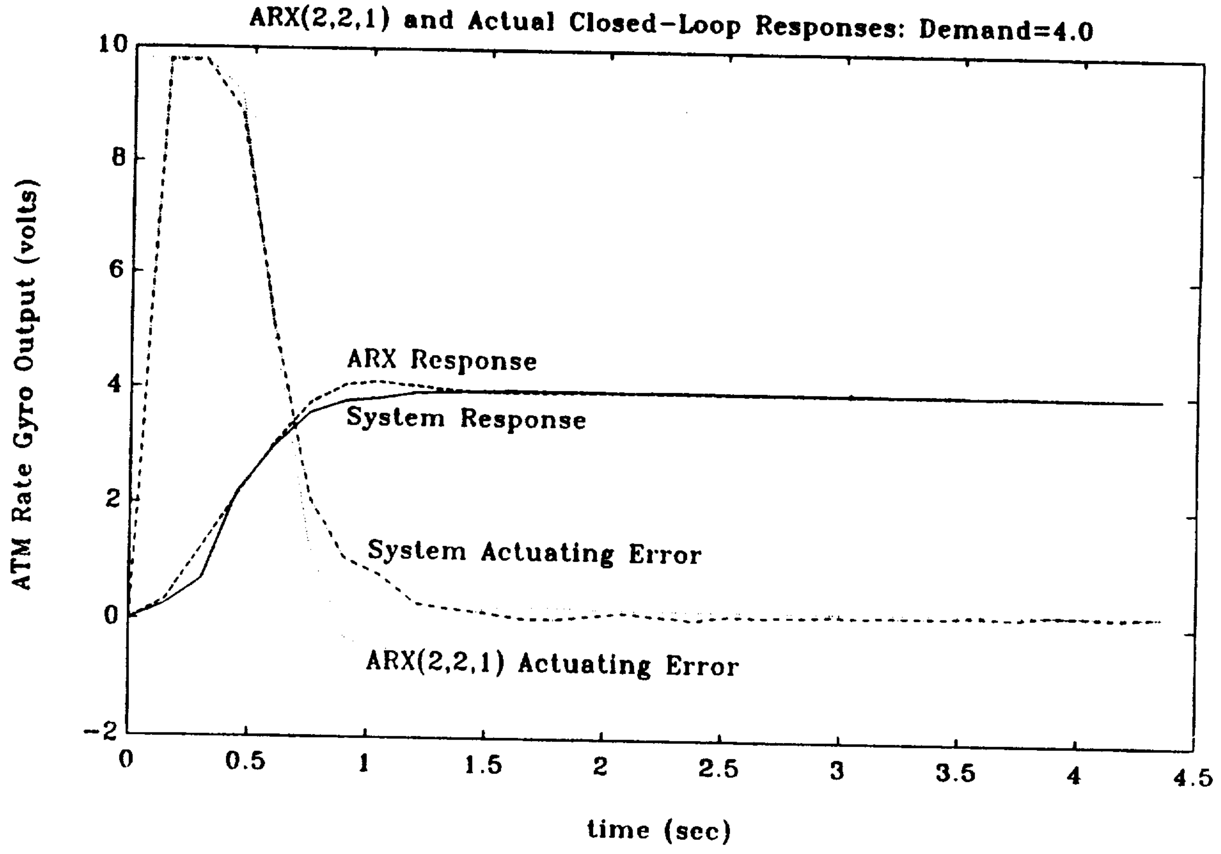


Figure 9 - Closed-Loop Response to a Rate Command of 4 Volts

of comparison, the ideal ARX(2,2,1) closed-loop response was generated by limiting the actuating error signal to ± 10 volts (which is the maximum available output of the D/A converter). Recall that a limiter acts as a low pass filter which causes the settling time to be much larger than the anticipated value of 0.5191 sec. The settling time of both the ideal and actual responses in Figure 9 is about twice this value, yet still well within the design specification of less than 3 seconds. In Figure 9 we also see that the ideal response overshoots every so slightly while the actual response does not. The slight overshoot of the ideal response is caused by the presence of a closed-loop zero at q=-1.9313. Apparently, this zero is not as prominent in the actual system.

Figure 9 also reveals that the steady-state error for both the ideal and actual response is negligible and that the two responses are virtually identical except for the small difference in overshoot. This indicates that our off-line identification scheme has resulted in a high-fidelity model of the system and that on-line parameter identification is not necessary. Thus, we have met the first three control design specifications and the last specification, robustness, will be tested in the final phase of the project – solving the CMG desaturation problem.

5.4 CMG Desaturation

The benefits for utilizing the CMGs rather than the thrusters to control the yaw rate of the vehicle are that the CMGs deliver a pure torque which produces a rotational motion completely decoupled from translational motion. However, a disadvantage of the CMGs is that when the θ - the angle formed between the spin axis of the CMGs and horizontal – nears 90 degrees, very little torque is available in the plane of the vehicle. Moreover, if θ exceeds 90 degrees, the torque produced in the plane of the vehicle is in the opposite direction of the torque that was produced for θ less than 90 degrees. Therefore, we have a positive feedback situation and the system will go unstable. The identical problem occurs when θ nears -90 degrees so for purposes of discussion we will restrict our attention to the situation of θ increasing positively towards 90 degrees. These aforementioned problems illustrate the necessity of periodically desaturating the CMGs (that is, returning θ to 0 degrees).

The most obvious method for desaturating the CMGs is to sense the angle θ and fire the yaw thrusters in harmony with the torque produced by the CMGs as θ nears 90 degrees. That is, use the yaw thrusters to produce more torque than is demanded of the CMGs. This will act as a torque disturbance to the closed-loop control system and thereby produce an actuating error signal which will cause the torquer motor to turn the CMGs back towards decreasing values of θ. If the yaw thrusters continue to fire in harmony with the CMGs, the torquer will continue to desaturate the CMGs until $\theta = 0$ at which time the thrusters would cease fire.

This method may initially appear to re-introduce the problem of coupling between the translational thrusters and the yaw thruster but closer scrutiny bears that this is not the case. The yaw thrusters are no longer part of the closed-loop control and function merely as a bang-bang torque disturbance. Of course firing the yaw thrusters introduces a translational force disturbance in addition to the torque disturbance. This translational force disturbance can be compensated for by the translational thrusters and any additional torque disturbance produced by this compensation can be lumped with the original torque disturbance and eliminated by the CMGs, not the yaw thrusters. In other words, by virtue of the CMGs producing a pure torque, the torque disturbances produced by the translational thrusters are not coupled back to translational disturbances when counteracted by the CMGs.

This desaturation method assumes that the implemented control law is sufficiently robust to compensate for external torque disturbances without a significant deterioration in the error between the demand rate and the actual rate of the vehicle. Experiments with the control indicated that small disturbances such as convection currents and imperfections in the flat floor could easily be compensated. Whether the control was tight enough to counteract a large torque disturbance, remained to be seen.

Prior to testing the desaturation scheme, a few hardware modifications were necessary. First, the angle θ needed to be measured. Initially, this information was obtained by integrating the command to the torquer rate gyro. This solution worked amazingly well and would have sufficed if the torquer servo loop had been tighter. But the torquer rate gyro used spring driven operation which had a tendency to drift thus making accurate integration impossible for long periods of time. One solution would be to replace the torquer rate gyro with a tachometer but a more inexpensive solution would be to mount a position sensor on the torquer axis. NASA engineer Bill Jacobs and NASA technician Zack Barnett came to the rescue by mounting a rotational potentiometer to the outside casing of one of the CMGs along the torquer spin axis. The output from the wiper arm of the pot was connected to an A/D channel of the adapter board with 0 volts corresponding to $\theta = 0$ and ± 8.3 volts corresponding to $\theta = \pm 90$ degrees.

Another hardware modification is to interface the yaw thrusters to the PC. The vehicle has two sets of yaw thrusters (clockwise and counterclockwise) and each set requires a 5 volt, low-current input to fire and an open circuit to cease fire. The IBM control adapter board has two D/A outputs and one of these has been consumed by the actuating signal output to the torquer servo. This implies that we must use a single D/A output to control both sets of thrusters.

The following circuit enables both sets of thrusters to be controlled from a single D/A output:

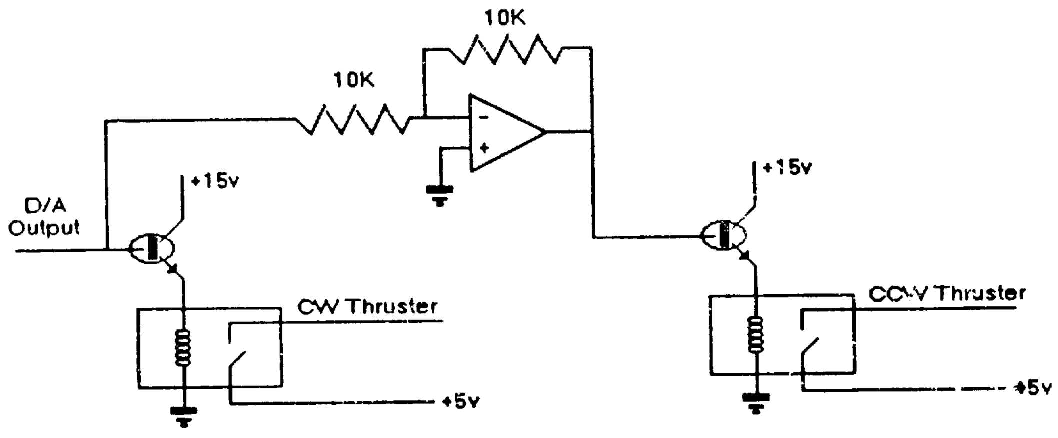


Figure 10 - Circuit to Control Yaw Thrusters

ORIGINAL PAGE IS
OF POOR QUALITY

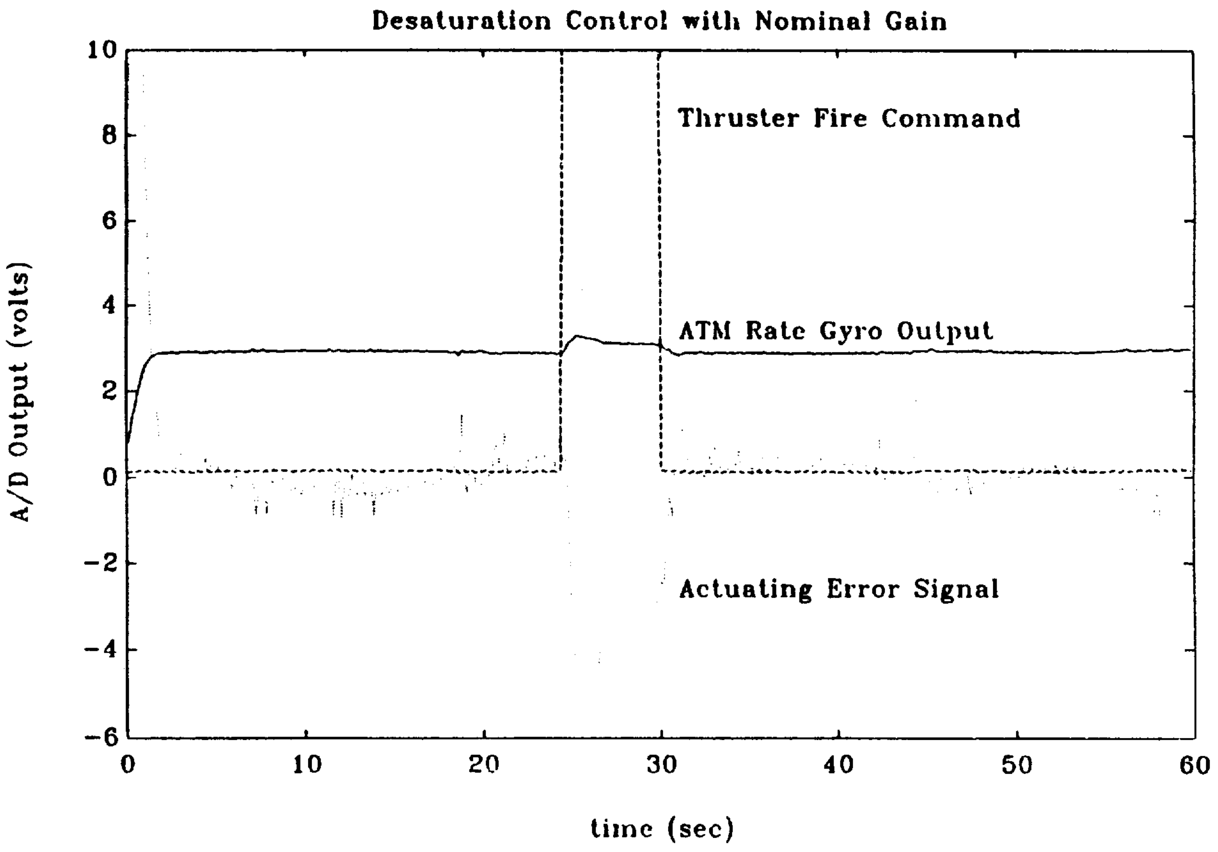


Figure 11 - Response of Desaturation Process with Nominal K

ORIGINAL PAGE IS
OF POOR QUALITY

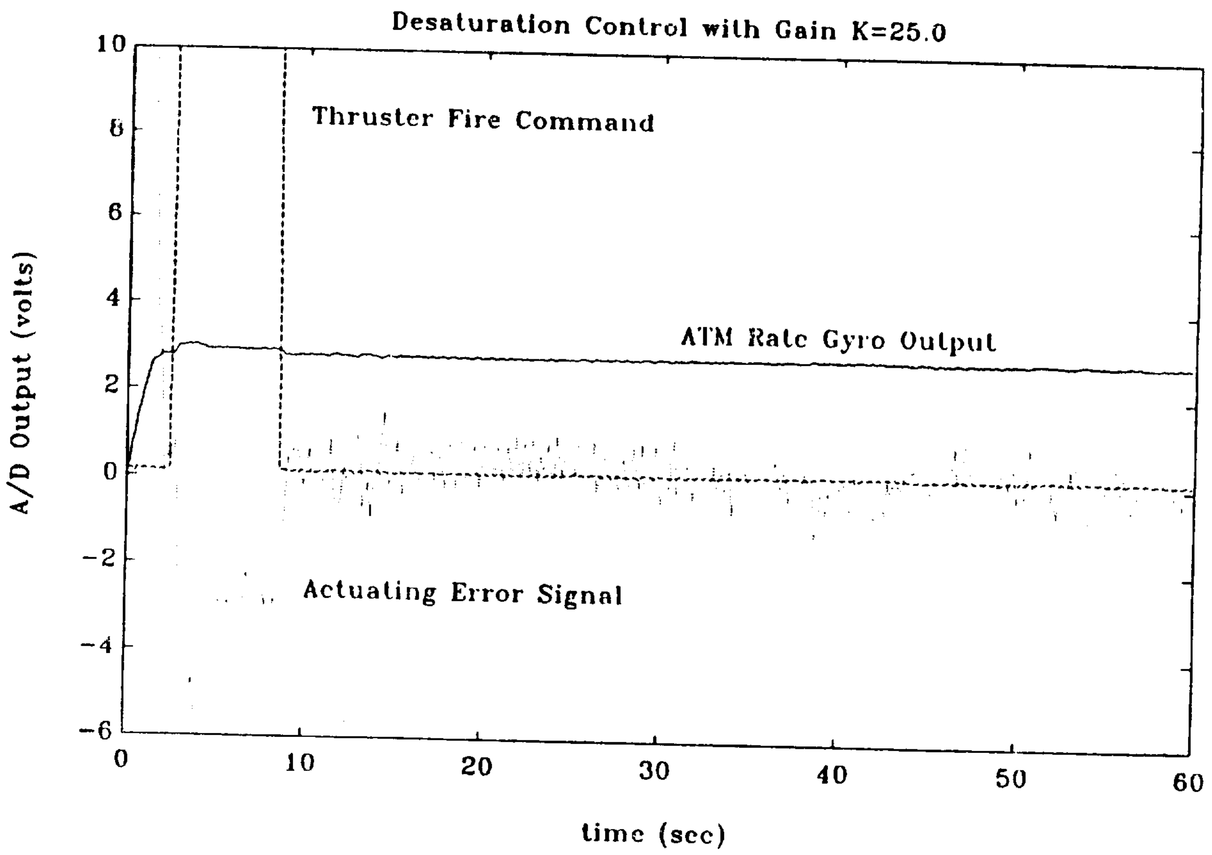


Figure 12 - Response of Desaturation Process with K = 25

Note that an output of 0 from the D/A does not affect a change in the relays while an output of +10 (-10) will turn on the left (right) transistor which in turn will close the left (right) relay thereby transmitting the necessary +5 volts to the clockwise (counter clockwise) thrusters.

After some experimentation, a value of $|\theta| = 65$ degrees was selected to trigger the desaturation process, at which point the available torque is still 42.3% of the maximum available torque. A value of $|\theta|$ closer to 90 degrees resulted in an insufficient amount of torque to counteract the disturbance while a value of $|\theta|$ closer to 0 resulted in desaturation occurring too frequently.

The results of the desaturation process are shown in Figures 11 and 12. Both trials were run with a setting of 50 psi on the air thrusters and on the air bearings. Figure 11 depicts the desaturation process for the nominal value of the feedback gain K. Here, a small increase in rate can be detected when the thrusters fire to desaturate the CMGs. If the application were teleoperation of the vehicle, this error could be tolerated since a pilot is in the loop making adjustments according to his display screen. If the application is autonomous, this disturbance may be outside of tolerance. The amount of rate error could be decreased, however, by increasing the value of the feedback gain. Recall that increasing the feedback gain increases the robustness of the system. Yet theoretically, our system is critically damped and any increase in the feedback gain will manifest itself in terms of overshoot. However, our ARX(2,2,1) model does not accurately represent the effects of limiting our input to ±10 volts. Limiting our input acts much like a low pass filter in cascade with our control block. Therefore, we may be able to increase the gain without the response acquiring overshoot. Figure 12 shows the desaturation process when K is increased to 25. Notice that there is no overshoot and that the change in the rate of the vehicle during desaturation is imperceptible. Conclusion: if maintaining a constant rate is critical, then a value of 25 should be employed in the feedback gain.

6 Conclusions and Recommendation

This report has presented a summary of a six week project concerning the yaw rate control of NASA's air bearing vehicle using CMGs. From the information presented, the following conclusions/recommendations are drawn:

- Two of the Sperry reaction wheels spinning at 3000 rpm are required to meet the the torque stipulations derived from the maximum yaw rate of the OMV.

- Of the 250 possible structures considered, an ARX(3,4,1) was the best fit according to the Akaike Final Prediction Error Criterion (FPE). However, an ARX(2,2,1) structure had a similar FPE and was chosen over the ARX(3,4,1) model for ease of control law design.

- A least squares estimation was employed off-line to compute the best parameter values for the ARX(2,2,1) structure. Although this estimation could be performed on-line, experimental data indicated that the parameters did not vary significantly. Thus, no adaptive estimation is necessary

- The open-loop step response and the results of the parameter estimation indicated that the vehicle behaved as a type 1 system. This implies that no integration is necessary in the feedback control loop to meet the steady-state error. A proportional feedback control (given by equation (5)) was designed so that the closed-loop was critically damped. This control can be computed on-line in a self-tuning scheme if necessary. Furthermore, due to the saturation effect of the D/A outputs, this feedback gain may be increased above this nominal value if more robustness is required during desaturation.

- Although tilting the ATM rate gyro's spin axis provided an excellent temporary solution to the problem of insufficient dynamic range, the vehicle is now sensitive to the Earth's rotation. A more permanent solution would be to rescale the electronics of the ATM or find an alternate rate gyro to devote to the project which has a dynamic range of approximately 3 degrees per second.

- The end product of this project is a control system – both hardware and software. Plans are currently underway to install an AT-bus 80386 computer on the vehicle. If and when this envent transpires, the control adapter can be installed in and the software ported to the new 80386 box.

- Using the thrusters to desaturate the CMGs did introduce some translational disturbance, but because the thrusters were not part of the closed-loop yaw control, this translational disturbance will not be coupled back into a rotational disturbance.

7 References

1. Clyde Jones, "Simplified Description of Control Moment Gyros," **NASA Technical Report R-ASTR-G-WP-12-67**, MSFC, July 11, 1967.

2. Lenart Ljung, **System Identification: Theory for the User.** Prentice-Hall, New Jersey, 1987.

3. P. Eykhoff, **Trends and Progress in System Identification.** Pergamon Press, Helmsford, New York, 1981.

4. K.J. Astrom, "Theory and applications of adaptive control -A survey", *Automatica*, vol. 19, pp. 471-486, 1983.

5. T. Soderstrom and P. Stoica, **Instrumental Variable Methods for System Identification.** Lecture Notes in Control and Information Sciences, Springer-Verlag, New York, 1983.

APPENDIX A

```
page    ,132
title   CONTROL MOMENT GYRO ASSEMBLY LANGUAGE DRIVER
subttl  SUPPORT DRIVER FOR IBM DATA ACQUISITION AND CONTROL ADAPTER

;source code filename:  ADAPTDRV.ASM

;programmer: Charles Oliver , EB24, NASA MSFC     APR 1, 1987

;    modified by Bruce L. Walcott, EB24, NASA MSFC  JUNE 5, 1989


; This program contains functions which are intended to be
; used by C programs compiled under MICROSOFT C 5.1 or higher
;
;
; analog_input function:
; MICROSOFT C STATEMENT:
;
; extern int far analog_input(int,int);
; int voltin, int channel, int adapter;
;       .
;       .
;       .
;voltin= analog_input(channel,adapter);
;
;returns an integer representation of the voltage on "channel"
;where channel is either 0, 1, 2, or 3 and adapter is 0 or 1 depending
;upon which board is desired.  If only one board is present, ther
;adapter must be set to 0.
; The ADC's have 12 bit resolution and have been set to a -10 to 10 volt range.
; For an input voltage of -10 volts, voltin will equal 0.  For an input volt
; of +10 volts, voltin will equal 4095.

; analog_output function
; MICROSOFT C STATEMENT:
;
; extern void far analog_output(int, int, int);
; int adapter, channel, voltout;
;       .
;       .
;       .
;analog_output(voltout,channel,adapter)
; The subroutine sets an analog output channel.  The three arguments must be
; integer variables.  adapter must be 0 or 1.  channel must be 0 or 1.
; The DAC's have 12 bit resolution and have been set to a -10 to +10 volt
; range.  voltout equals 0 produces a -10 volt output and voltout equals 4095
; produces a +10 volt output.
;________________________________________________________________________
;
;equates for accessing IBM data acquisition and control adapter
;________________________________________________________________________

.MODEL MEDIUM                           ;tells MS to use medium model
.CODE
device_number_reg0 equu  0c2e2h     ;< adapter 0 device number register >
device_number_reg1 equ  0c6e2h      ;< adapter 1 device number register >

analog_device      equ  9           ;code to select analog device
binary_device      equ  8           ;code to select binary device
```

```
AI_control_reg0     equ  02e2h     ;< adapter 0 analog input control register >
AI_status_reg0      equ  02e2h     ;< adapter 0 analog input status register >
AI_data_reg0        equ  22e2h     ;< adapter 0 analog input data register >
  _control_reg1     equ  06e2h     ;< adapter 1 analog input control register >
AI_status_reg1      equ  06e2h     ;< adapter 1 analog input status register >
AI_data_reg1        equ  26e2h     ;< adapter 1 analog input data register >

ad0_channel         equ  0000h     ;code to select A/D channel 0
ad1_channel         equ  0100h     ;code to select A/D channel 1
ad2_channel         equ  0200h     ;code to select A/D channel 2
ad3_channel         equ  0300h     ;code to select A/D channel 3

convert_start       equ  0001h     ;convert start bit in AI_control
busy_state          equ  0001h     ;busy state bit in AI_status
count4_multiplexer_settling  equ  0010h    ;count for delay while analog
                                           ;multiplexer settles

AO_control_reg0     equ  12e2h     ;< adapter 0 analog output control reg >
AO_data_reg0        equ  32e2h     ;< adapter 0 analog output data register >
AO_control_reg1     equ  16e2h     ;< adapter 1 analog output control reg >
AO_data_reg1        equ  36e2h     ;< adapter 1 analog output data register >

da0_channel         equ  0000h     ;code to select D/A channel 0
da1_channel         equ  0100h     ;code to select D/A channel 1

BI_status_reg0      equ  02e2h     ;< adapter 0 binary status register >
BI_control_reg0     equ  02e2h     ;< adapter 0 binary control register >
BI_input_reg0       equ  22e2h     ;< adapter 0 binary input register >
BI_output_reg0      equ  22e2h     ;< adapter 0 binary output register >
  _status_reg1      equ  06e2h     ;< adapter 1 binary status register >
BI_control_reg1     equ  06e2h     ;< adapter 1 binary control register >
BI_input_reg1       equ  26e2h     ;< adapter 1 binary input register >
BI_output_reg1      equ  26e2h     ;< adapter 1 binary output register >

cntr_control_reg0   equ  0b2e2h   ;< adapter 0 counter control register >
cntr_control_reg1   equ  0b6e2h   ;< adapter 1 counter control register >
counter0_reg0       equ  82e2h    ;< adapter 0 counter 0 read/write register >
counter0_reg1       equ  86e2h    ;< adapter 1 counter 0 read/write register >
counter1_reg0       equ  92e2h    ;< adapter 0 counter 1 read/write register >
counter1_reg1       equ  96e2h    ;< adapter 1 counter 1 read/write register >
counter2_reg0       equ  0a2e2h   ;< adapter 0 counter 2 read/write register >
counter2_reg1       equ  0a6e2h   ;< adapter 1 counter 2 read/write register >

set_counter0_mode3 equ  36h       ;code to set counter 0 to mode 3
set_counter1_mode3 equ  76h       ;code to set counter 1 to mode 3
set_counter2_mode3 equ  0b6h      ;code to set counter 2 to mode 3
latch_counter2     equ  80h       ;code to latch counter 2

int_control_reg0   equ  0d2e2h     ;< adapter 0 interrupt control register >
int_status_reg0    equ  0d2e2h     ;< adapter 0 interrupt status register >
int_control_reg1   equ  0d6e2h     ;< adapter 1 interrupt control register >
int_status_reg1    equ  0d6e2h     ;< adapter 1 interrupt status register >
                                           ;
irq_stat           equ  01000000b         ;external interrupt status, IRQ

out_ax        macro      io_port, value
              mov        dx,io_port
              mov        ax,value
              out        dx,ax
              jmp        $+2
```

```
                jmp         $+2
                endm

out_al          macro       io_port, value
                mov         dx,io_port
                mov         al,value
                out         dx,al
                jmp         $+2
                jmp         $+2
                endm


;______________________________________________________________________
;
;main program
;______________________________________________________________________

prog            SEGMENT     para public 'CODE'
                ASSUME      cs:prog

;device driver header
                DD          -1              ;pointer to another driver
                DW          8000h           ;character device attribute
                DB          "stxdriv "
;end of header



device_number dw            0     ;locations used by binary and analog
BI_input        dw          0     ;input/output subroutines
BI_output       dw          0
channel         dw          0
AI_control      dw          0
AI_status       dw          0
AI_data         dw          0
AO_control      dw          0
AO_data         dw          0

bit15_flag      dw          0     ;used for triggering adscan with data bit 15
bit15_tlow      dw          0     ;timeout counter low word
bit15_thigh     dw          0     ;timeout counter high word


;______________________________________________________________________
;
;analog_input function (called from C)
;______________________________________________________________________
                PUBLIC _analog_input
_analog_input   proc         far

                push bp
                mov         bp,sp
                mov         bx,[bp+8]
                cmp         bx,0  ;compare ADAPTER argument with 0
                jne         17                  ;if not zero, setup for adapter
                mov         device_number,device_number_reg0   ;else, setup for
                mov         AI_control,AI_control_reg0         ;adapter 0
                mov         AI_status,AI_status_reg0
                mov         AI_data,AI_data_reg0
                jmp         18
```

```
17:            mov       device_number,device_number_reg1
               mov       AI_control,AI_control_reg1
               mov       AI_status,AI_status_reg1
               mov       AI_data,AI_data_reg1

18:            mov       bx,[bp+6]
               cmp       bx,0        ;compare CHANNEL arg with 0
               jne       19
               mov       channel,ad0_channel       ;set for channel 0
               jmp       112
19:            cmp       bx,1        ;compare CHANNEL arg with 1
               jne       110
               mov       channel,ad1_channel       ;set for channel 1
               jmp       112
110:           cmp       bx,2        ;compare CHANNEL arg with 2
               jne       111
               mov       channel,ad2_channel       ;set for channel 2
               jmp       112
111:           mov       channel,ad3_channel       ;set for channel 3

112:           out_al    device_number,analog_device   ;select adapter

               out_ax    AI_control,channel            ;select channel

               mov       cx,count4_multiplexer_settling    ;wait for analog
wait4:         loop      wait4                         ;multiplexer to settle

               mov       ax,channel          ;start analog conversiom
               or        ax,convert_start
               out_ax    AI_control,ax

               mov       dx,AI_status        ;wait for A/D conversion to finish
wait5:         in        ax,dx
               and       ax,busy_state
               jnz       wait5

               out_ax    AI_control,channel

               mov       dx,AI_data          ;input analog value
               in        ax,dx

               pop bp
               ret                         ;return to BASIC

_analog_input  endp

;____________________________________________________________________
;
;analog_output function (called from C)
;____________________________________________________________________
                PUBLIC _analog_output
_analog_output proc      far

               push      bp
               mov       bp,sp
               mov       bx,[bp+10]
               cmp       bx,0  ;compare ADAPTER argument with 0
               jne       113               ;if not zero, setup for adapter 1
               mov       device_number,device_number_reg0   ;else, setup for
               mov       AO_control,AO_control_reg0         ;adapter 0
```

```
            mov       AO_data,AO_data_reg0
            jmp       114
113:        mov       device_number,device_number_reg1
            mov       AO_control,AO_control_reg1
            mov       AO_data,AO_data_reg1

114:        mov       bx,[bp+8]
            cmp       bx,0    ;compare CHANNEL arg with 0
            jne       115                 ;if not zero, setup for channel 1
            mov       channel,ad0_channel ;else, setup for channel 0
            jmp       116
115:        mov       channel,ad1_channel

116:        out_al    device_number, analog_device  ;select analog device

            out_ax    AO_control,channel            ;select channel

            mov       bx,[bp+6]          ;send word from BASIC to analog
            out_ax    AO_data,bx     ;output channel

            pop       bp
            ret                       ;return to basic

_analog_output endp

;_____________________________________________________________________
;



            DB        64 dup ('s','t','a','c','k',' ',' ',' ')
stack:      DW        0

;_____________________________________________________________________
;
;end of program
;_____________________________________________________________________

lastword:   DW        0

prog        ENDS

            END
```

APPENDIX B

```
/*****
*  stepdata.c
*
*  Programmer: Bruce L. Walcott
*
*  written: June 19, 1989
*
*  This program collects data points from the airbearing vehicle in closed-loop
*  given a setpoint
*
******/

/*
*  External connections are:  D/A channel 0 to torquer input
*                             D/A channel 1 to thruster circuit input
*                             A/D channel 0 to D/A channel 1
*                             A/D channel 1 to D/A channel 0
*                             A/D channel 2 to wiper arm of pot
*                             A/D channel 3 to ATM rate gyro output
*/

/*  Includes and external declarations  */

#include <graph.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
extern void far analog_output ( int, int, int);
extern int far analog_input ( int, int);

/*****
* Main Program
*****/

main()
{

/* Internal declarations   */

FILE *f;
int analgout,i, x, j, sampletime, k, flag,number;
long color;
static float voltin[8000];
float voltout, offset, diff, posit1, posit2, epsilon, k0, u, y[3], saturate;

/* Initialize variables  */

color=9;              /* 9 = blue */
number=0.;
k0 = 19.1802;         /* Feedback gain for critical damping */
epsilon=.005;         /* Tolerance for eliminating drift */
offset=0;             /* Initial value of DC offset to eliminate drift */

_settextwindow(1,1,25,80);
_setbkcolor(color);            /* Set-up graphics and clear screen */
_clearscreen(_GCLEARSCREEN);

_settextposition(6,20);        /* Print messages for gyro drift elimination */
_settextcolor(7);
_outtext("Eliminating rate gyro offset.");
```

```
_settextposition(6,20);
_outtext("      Please wait.     ");

 :le(1){

        posit1=(analog_input(2,0)/2048.)*10.-10.;    /* Take one pot reading */
        for(k=0 ; k < 500; k++){
                for(j=0; j < 300; j++) continue;     /* Wait */
                }
        posit2=(analog_input(2,0)/2048.)*10.-10.;    /* Take another */

        diff=posit2 - posit1;                       /* If difference not within
        if( fabs(diff) > epsilon){                  /* tolerance, change offset
                offset = offset - .05* fabs(diff)/diff;
                analog_output(2048+(int)((offset)*204.8),0,0);  /* Output offset
                }
        else break;                          /* If difference is within toleranc
        }

epsilon=0.01;                            /* Now move gyros back to zero position *

while(1){
        posit1=(analog_input(2,0)/2048.)*10.-10.;
        if( fabs(posit1) > epsilon){
                analog_output(2048+(int)((offset-.5*fabs(posit1)/posit1)*204.8),
                }
        else {
                analog_output(2048+(int)((offset)*204.8),0,0);
                break;
              }
        }

_settextposition(7,20);                  /* Put sampling time in sampletime */
_outtext("Enter the desired sampling time in miliseconds:  ");
scanf("%d",&sampletime);

_settextposition(8,20);                  /* Put number of points in number */
_outtext("Enter the number of data points to capture:   ");
scanf("%d",&number);

_settextposition(9,20);                  /* Put setpoint in voltout */
_outtext("Enter the value for step on D/A output on channel 0:  ");
scanf("%f",&voltout);

f = fopen("data\\stepdata.dat","w+");   /* Open data file */
if (f == NULL)
    {
    printf("          Cannot open stepdata.dat");
    return;
    }

/* Write header information  */

fprintf(f,"sampletime=%d, amplitude=%2.3f, offset=%2.3f, number=%d\n",sampletime

saturate=6.;                   /* Desaturate at +- 6 volts */
flag=0;                        /* flag = 1 when desaturating CW, 2 when desaturating
x=2048;                        /* x is command for thrusters */
```

```
/* Main control loop */


for (i=0;i < (4*number);i += 4)
     {
     analog_output(x,1,0);                               /* Command thrusters */

     voltin[0+i]=(analog_input(0,0)/2048.)*10.-10.;  /* Get inputs */
     voltin[1+i]=(analog_input(1,0)/2048.)*10.-10.;
     voltin[2+i]=(analog_input(2,0)/2048.)*10.-10.;
     voltin[3+i]=(analog_input(3,0)/2048.)*10.-10.;

     y[0]=voltin[3+i]-(voltout+offset);                  /* Calculate control */
     u=-k0*y[0];

     if(fabs(u) > 10) u=10*u/fabs(u);                    /* Limit value to +-10 volt
     analog_output((int)((u+10)*4095/20),0,0);           /* Output control */

     if( voltin[2+i] > saturate  ) {                     /* Check for CW desaturatio
     flag=1;                                             /* Set flag */
     x=4095;                                             /* Command CW thrusters off
     }

     if(voltin[2+i] < -saturate ) {                      /* Check for CCW desaturati
     flag=2;                                             /* Set flag */
     x= 0;                                               /* Command CCW thrusters on
     }

     if (flag == 1 && voltin[2+i]  < 0) {             /* Check if back to zero */
     x=2048;                                          /* Command CW thrusters off
     flag=0;                                          /* Reset flag */
     }

     if (flag == 2 && voltin[2+i]  > 0) {             /* Check if back to zero */
     x=2048;                                          /* Command CCW thrusters off
     flag=0;                                          /* Reset flag */
     }

     for(k=0; k < 58; k++) continue;                  /* Wait sampling time */
     for(k=0; k < sampletime -1; k++)
     for(j=0; j < 300; j++) continue;
     }

analog_output(2048+(int)((offset)*204.8),0,0);      /* Exit loop and turn off */
analog_output(2048,1,0);

for(i=0;i < (4*number);i += 4)                      /* Write data */
    {
    fprintf(f,"%2.3f %2.3f %2.3f %2.3f\n",voltin[i],voltin[i+1],voltin[i+2]-offs
    }
}
```

```
/*****
 *  Posit.c
 *
    Programmer: Bruce L. Walcott
 *
 *  written: July 14, 1989
 *
 *  This program closes the loop on the vehicle and desaturates the gyros
 *  using a potentiometer connected to the torquer motor spin axis.  The
 *  user can command desired rate from the keyboard.
 *
 ******/

/*
 *  External connections are:  D/A channel 0 to torquer input
 *                             D/A channel 1 to thruster circuit input
 *                             A/D channel 0 to D/A channel 1
 *                             A/D channel 1 to D/A channel 0
 *                             A/D channel 2 to wiper arm of pot
 *                             A/D channel 3 to ATM rate gyro output
 */

/*  Includes and external declarations  */

#include <graph.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
extern void far analog_output ( int, int, int);
extern int far analog_input ( int, int);

/*****
 * Main Program
 *****/

main()
{

/* Internal declarations   */

FILE *f;
int analgout,i, x, j, sampletime, k, flag;
long color;
static float voltin[8000];
float voltout, offset, zero1, zero2, diff, posit1, posit2, epsilon, k0, u, y[3],
char chr;

/* Initialize variables  */

color=9;              /* 9 = blue */
chr = '0';
k0 = 19.1802;         /* Feedback gain for critical damping */
epsilon=.005;         /* Tolerance for eliminating drift */
offset=0;             /* Initial value of DC offset to eliminate drift */

_settextwindow(1,1,25,80);
_setbkcolor(color);                 /* Set-up graphics and clear screen */
_clearscreen(_GCLEARSCREEN);
```

```
_settextposition(6,20);          /* Print messages for gyro drift elimination */
_settextcolor(7);
_outtext("Eliminating rate gyro offset.");
_settextposition(6,20);
_outtext("      Please wait.     ");

while(1){

        posit1=(analog_input(2,0)/2048.)*10.-10.;    /* Take one pot reading */
        for(k=0 ; k < 500; k++){
                for(j=0; j < 300; j++) continue;     /* Wait */
                }
        posit2=(analog_input(2,0)/2048.)*10.-10.;    /* Take another */

        diff=posit2 - posit1;                       /* If difference not within
        if( fabs(diff) > epsilon){                  /* tolerance, change offset
                offset = offset - .05* fabs(diff)/diff;
                analog_output(2048+(int)((offset)*204.8),0,0);  /* Output offset
                }
        else break;                          /* If difference is within toleranc
        }

epsilon=0.01;                          /* Now move gyros back to zero position *

while(1){
        posit1=(analog_input(2,0)/2048.)*10.-10.;
        if( fabs(posit1) > epsilon){
                analog_output(2048+(int)((offset-.5*fabs(posit1)/posit1)*204.8),
                }
        else {
                analog_output(2048+(int)((offset)*204.8),0,0);
                break;
              }
        }

_settextposition(7,20);
_outtext("Type 0 to stop, 1 to go cw, 2 to go ccw, b to break ");

voltout=0.;                  /* Initialize setpoint to 0 */
sampletime=150;              /* Sampling time is 150 msec */
saturate=6.;                 /* Desaturate at +- 6 volts */
flag=0;                      /* flag = 1 when desaturating CW, 2 when desaturating
x=2048;                      /* x is command for thrusters */

while(chr != 'b') {          /* Break when 'b' is entered */

     while(!kbhit())         /* Loop until key is struck */
     {
     analog_output(x,1,0);                             /* Command thrusters */

     voltin[0]=(analog_input(0,0)/2048.)*10.-10.;  /* Get inputs */
     voltin[1]=(analog_input(1,0)/2048.)*10.-10.;
     voltin[2]=(analog_input(2,0)/2048.)*10.-10.;
     voltin[3]=(analog_input(3,0)/2048.)*10.-10.;

     y[0]=voltin[3]-(voltout+offset);              /* Calculate control */
     u=-k0*y[0];

     if(fabs(u) > 10) u=10*u/fabs(u);               /* Limit value to +-10 volt
     analog_output((int)((u+10)*4095/20),0,0);      /* Output control */
```

```
      if( voltin[2] > saturate  ) {                         /* Check for CW desaturation
      flag=1;                                                 /* Set flag */
      x=4095;                                                 /* Command CW thrusters off
      }

      if(voltin[2] < -saturate ) {                          /* Check for CCW desaturation
      flag=2;                                                 /* Set flag */
      x= 0;                                                   /* Command CCW thrusters on
      }

      if (flag == 1 && voltin[2]  < 0) {                  /* Check if back to zero */
      x=2048;                                               /* Command CW thrusters off *
      flag=0;                                               /* Reset flag */
      }

      if (flag == 2 && voltin[2]  > 0) {                  /* Check if back to zero */
      x=2048;                                               /* Command CCW thrusters off
      flag=0;                                               /* Reset flag */
      }

      for(k=0; k < 58; k++) continue;                       /* Wait sampling time */
      for(k=0; k < sampletime -1; k++)
      for(j=0; j < 300; j++) continue;
      }

      chr=getch();                                          /* If is struck, get characte
      if(chr == '1') voltout = voltout+.125;                /* If '1', increase setpoint
      if(chr == '2') voltout = voltout - .125;              /* If '2', decrease setpoint
      if(chr == '0') voltout = 0.;                          /* If '0', reset setpoint to
      }

analog_output(2048+(int)((offset)*204.8),0,0);              /* Exit loop and turn off */
analog_output(2048,1,0);

}
```