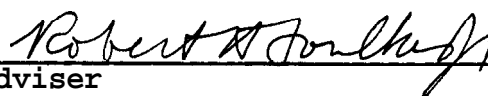


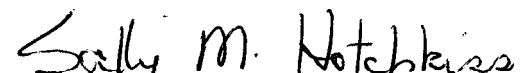
A DESIGN TECHNIQUE FOR MULTIVARIABLE  
SERVO-COMPENSATORS WITH AN APPLICATION  
TO FLIGHT CONTROL

by

ALI MOTAKEF

Submitted in Partial Fulfillment of the Requirements  
for the Degree of  
Master of Science in Engineering  
in the  
Electrical Engineering  
Program

  
Adviser 3-25-87  
Date

  
Dean of the Graduate School March 27, 1987  
Date

YOUNGSTOWN STATE UNIVERSITY

March, 1987

## ABSTRACT

A DESIGN TECHNIQUE FOR MULTIVARIABLE  
SERVO-COMPENSATORS WITH AN APPLICATION  
TO FLIGHT CONTROL

ALI MOTAKEF

MASTER OF SCIENCE IN ENGINEERING

YOUNGSTOWN STATE UNIVERSITY, 1987

A servo design technique is developed and applied. The technique is applied to a certain B-737 (aircraft) flight control parameters in order to turn the airplane. The system is multivariable with two inputs and two outputs. It is then divided to two single-input single-output (SISO) subsystems without any interactions between them and each subsystem is studied separately. Later the interactions are put back into the system and their effects are studied. Another design is developed to reduce the interactions between the subsystems. The reduced system is then studied.

TABLE OF CONTENTS

	PAGE
ABSTRACT .....	ii
TABLE OF CONTENTS .....	iii
LIST OF SYMBOLS .....	v
LIST OF FIGURES .....	vi
CHAPTER	
I. INTRODUCTION .....	1
Servo-compensators .....	2
SISO classical PID controllers .....	2
State variable feedback designs .....	3
Multivariable design using i/o model .....	6
II. DEVELOPMENT OF A DESIGN TECHNIQUE .....	9
Decomposition into subsystems .....	9
A SISO servo design .....	11
Least squares design to reduce subsystem interaction .....	19
III. APPLICATION .....	21
Model development .....	21
Subsystem designs .....	23
Reduction of interaction between subsystems ...	34
IV. CONCLUSION .....	48
Brief summary of the project .....	48
Suggestion for further work .....	49
APPENDIX A. ....	51

	PAGE
Computer programs .....	52
REFERENCES .....	65

## LIST OF SYMBOLS

SYMBOLS	DEFINITION	UNITS OF REFERENCE
$\Sigma$	Summation of the mathematical terms that follow	none
$\alpha_1, \alpha_2$	Constants	none
$T$	Sampling period	sec
$\Phi$	Discrete equivalent of analog model A	none
$\Gamma$	Discrete equivalent of analog model B	none
$\beta$	Constant	
$\mathcal{L}_x$	Controllability matrix	
$G(z)$	Open-loop transfer function	
$H(z)$	Overall transfer function	
$D_H(z)$	The denominator of $H(z)$	
$D(z)$	Compensator transfer function	
$\phi$	An element of $\Phi$ matrix	
$\gamma$	An element of $\Gamma$ matrix	

LIST OF FIGURES

FIGURES	PAGES
1. A servo system	2
2. Classical compensation scheme	3
3. Feedback control	4
4. <b>Multivariable</b> PI control	6
5. The closed-loop system	8
6. A digital SISO plant	12
7. The plant with a controller	12
8. Closed-loop SISO servo subsystem	13
9. Overall system	24
10. Aileron control system	27
11. Bank angle(A) and Bank angle command(B) vs time [decoupled]	28
12. aileron(A) [decoupled]	29
13. Bank angle(A) and Bank angle command(B) vs time [ $k_d$ changed]	31
14. aileron(A) [ $k_d$ changed]	32
15. Rudder control system	34
16. Yaw rate(A) and Yaw rate command(B) vs time [decoupled]	35
17. rudder(A) [decoupled]	36

FIGURES

PAGES

18.	Yaw rate(A) and Yaw rate command(B) vs time [ $k_d$ changed]	37
19.	rudder(A) [ $k_d$ changed]	38
20.	Bank angle(A) and Bank angle command(B) vs time [coupling]	40
21.	aileron(A) [with coupling]	41
22.	Yaw rate(A) and Yaw rate command(B) vs time [coupling]	42
23.	rudder(A) [with coupling]	43
24.	Bank angle(A) and Bank angle command(B) vs time [reduction of interaction]	44
25.	Drud(A) and Dail(B) vs time [reduction of interaction]	45
26.	Drudl(A) and Dail1(B) vs time [reduction of interaction]	46
27.	Yaw rate(A) and Yaw rate command(B) vs time [reduction of interaction]	47

## CHAPTER I

### INTRODUCTION

The purpose of this thesis is to demonstrate a design technique for multivariable servo-compensators with an application to flight control.

As an introduction, the concept of servo-compensators and their use in single-input, single-output (SISO) classical PID controllers, state variable feedback (SVFB) design and **input/output** model for multivariable design are discussed. In chapter two, the theory of the project is developed as a general multivariable control design in three sections: decomposition into SISO subsystems, SISO design with PID controller plus SVFD, and a design for reduction of interaction. Chapter three discusses the application of the project to flight control. This chapter has three sections: model development, subsystem design, and reduction of interaction between subsystems. In subsystem design section, the designs for each subsystem are discussed separately. The final chapter is a conclusion with a brief summary and some suggestions for further work on the design.



## SERVO-COMPENSATORS [1]

---

A servo system is a feedback control system where the input varies and the output must be made to follow it as closely as possible. A block diagram of a servo system is shown in Figure 1.

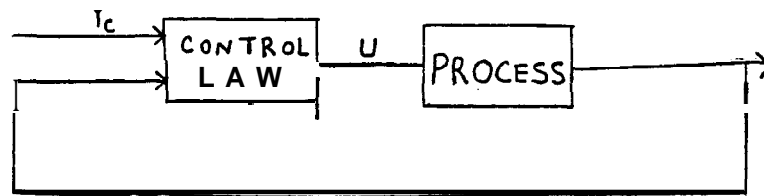


Figure 1. A Servo System.

Power steering is one example of a servo system, as are the systems for positioning control surfaces on aircraft. Automated manufacturing machinery, such as numerically controlled machine tools, use servos extensively for the control of positions or speeds. A brief overview of three different approaches to servo design is presented below.

## SISO CLASSICAL PID CONTROLLERS [2]

---

The transfer function, which is the fundamental concept of frequency-domain analysis, expresses the relationship between the Laplace transform  $Y(s)$  of the system output  $y(t)$  and the Laplace transform  $U(s)$  of the input  $u(t)$ :

$$Y(s) = H(s) * U(s) \quad (1)$$

where  $H(s)$  is the transfer function of the system.

This relationship is valid for any time-invariant linear system, even when the system can not be represented by sets of ordinary differential equations of finite order.

Frequency-domain analysis and design possess a wealth of graphical and semi-graphical techniques that can be applied to linear time-invariant control systems virtually of any complexity. Some of well-known methods are the Nyquist plot, the Bode diagram, the gain-phase plot, and the root locus method.

A block diagram of a SISO with a PID controller is shown in Figure 2.

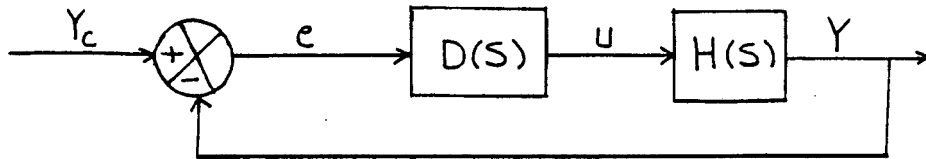


Figure 2. Classical Compensation Scheme.

Here,  $D(s)$  is a PID (proportional-integral-derivative) controller.

#### STATE VARIABLE FEEDBACK DESIGNS {2}

.....

Many systems of practical importance have many more than one input and output. Moreover, the behavior of internal

variables of a plant is often of vital interest, For instance, it may happen that while the plant output is stable some elements inside the plant are exceeding specified ratings. In such an event a simultaneous knowledge of the state of the variables at some predetermined points along the flow of signal (or information) would be of immense help. These have led to the evolution of what has come to be known as the state variable feedback approach to the analysis and design of control systems.

For system design, consider the feedback control configuration in Figure 3, The plant is described by its state-space model, and it is assumed that all state variables are available for use in feedback. For now, the external system inputs are taken to be zero, the state feedback control in Figure 3 is then  $u = -k*x$  and the system equation is

$$\dot{x} = A*x + B*u = (A - B*k) * x \quad (2)$$

$(A - B*k)$  is the closed-loop system matrix and must have stable eigenvalues for the response to tend to zero.

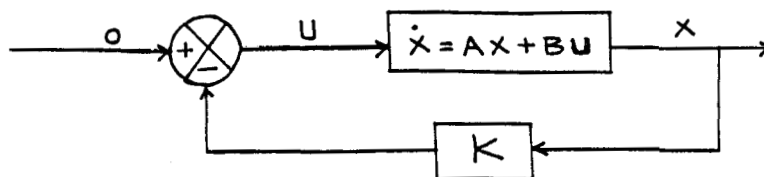


Figure 3. Feedback Control,

The following theorem is of fundamental importance:

Feedback control theorem [1]:

Any specified-set of closed-loop eigenvalues can be obtained by state feedback with  $K$  consisting of constant gains if and only if the pair  $(A,B)$  is controllable.

The pair  $(A,B)$  can be controlled by design of  $K$ . Constant gains can achieve this, while dynamic compensation was often found necessary just for adequate performance in classical single-input, single-output design, because there is feedback from all states instead of only from a single output.

Two major approaches to the design of  $K$  are the following:

- 1- Pole assignment: The closed-loop eigenvalues are placed in specified locations.
- 2- Optimal control: A specified mathematical performance criterion is minimized.

State variable feedback gives good control of transient response, but not of steady state response. Therefore the use of integral control is of great importance in achieving servo design. Integral control can provide a stable design (i.e.,  $x \rightarrow 0$  as  $t \rightarrow \infty$ ) with zero steady state error (i.e.,  $y \rightarrow y_{ref}$  as  $t \rightarrow \infty$ ). The configuration of integral and feedforward controls is illustrated in Figure 4.

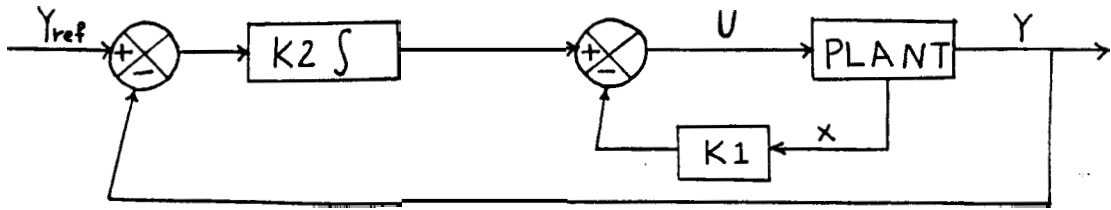


Figure 4. Multivariable PI Control.

It can be seen from the diagram that the control U equals

$$U = -k_1 * x + k_2 * \int_0^t (-y + y_{ref}) * dt \quad (3)$$

This consists of proportional state feedback and integral control of output error, and represents a multivariable generalization of PI control.

MULTIVARIABLE DESIGN USING 1/0 MODEL [3] (EXTENTION OF TRANSFER

FUNCTION TECHNIQUES)

The design considers a servo problem, which is formulated as a model following problem. Thus it is desired to find a control law such that the appropriate response to command inputs is obtained. This formulation includes pole-assignment design as a special case.

It is assumed that the process has inputs,  $U_k$ , and measured outputs,  $Y_k$ . The relation between  $U_k$  and  $Y_k$  is given by the pulse-transfer function

$$H(z) = B(z)/A(z) \quad (4)$$

where  $A(z)$  and  $B(z)$  are polynomials. Notice that the pulse transfer function represents the dynamics of the process, including hold circuit, actuator, sensor and antialiasing filter. It is assumed that the polynomials  $A$  and  $B$  do not have any common factors.

The servo specifications are expressed in terms of a model that gives the desired response to command signals. The desired closed-loop pulse-transfer function is given by

$$H_m(z) = B_m(z)/A_m(z) \quad (5)$$

where  $B_m$  and  $A_m$  do not have any common factors.

Now, the control law should be specified. The compensator has output  $U_k$ , and inputs  $Y$  and  $Y_c$ , where  $Y_c$  is the reference input. A general linear structure is

$$U_k(k) = [T_1(q)/R_1(q)] * Y_c(k) - [S_1(q)/R_2(q)] * Y_k(k) \quad (6)$$

where  $R_1$ ,  $R_2$ ,  $T_1$  and  $S_1$  are polynomials in the forward shift operator  $q$ . Then the control law can be written as

$$R(q) * U_k = T(q) * (Y_c)_k - S(q) * Y_k \quad (7)$$

where  $R = R_1 * R_2$ ,  $T = T_1 * R_2$ , and  $S = S_1 * R_1$ . It is assumed that the coefficient of the highest power in  $R$  is unity.

So, a process model specified by the input-output relation is

$$A(q) * Y(k) = B(q) * U(k) \quad (8)$$

A block diagram of the closed-loop system is shown in Figure 5. The input-output relationship for the closed-loop system are obtained by eliminating  $U_k$  between equations 7 and 8.

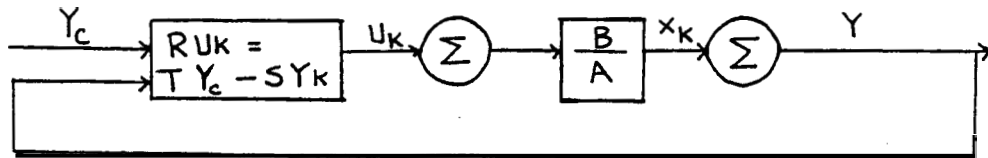


Figure 5. The Closed-loop System.

Hence

$$(A * R + B * S) * Y_k = B * T * Y_c \quad (9)$$

Requiring that this input-output relation is equivalent to equation 5 gives

$$(B * T) / (A * R + B * S) = B_m / A_m \quad (10)$$

By finding the polynomials  $R$ ,  $S$ , and  $T$  that satisfy equation 10, the design can be completed.

## CHAPTER II

### DEVELOPMENT OF A DESIGN TECHNIQUE

In this section a design technique is developed. This includes the decomposition of multivariable systems into SISO (single-input single-output) subsystems, a SISO servo design, and a least-squares approach to reduce the interaction between subsystems.

#### A: DECOMPOSITION INTO SISO SUBSYSTEMS [4]

---

The first step in any multivariable design may be an attempt to find an approximate model consisting of two or more single **input/output** models. An approximate model may be achieved by decoupling the system to SISO subsystems. The decoupling is done by block partitioning the system matrices and making the interaction blocks (coupling terms) be zero. This step may lead to a plant description which is substantially simpler for design purposes and yet yields no significant degradation from an analysis based on the full multivariable system.

Consider the state-variable description of a multivariable



plant:

$$x_{k+1} = \Phi * x_k + \Gamma * u_k \quad (11)$$

$$y = C * x \quad (12)$$

$x$ ,  $u$ , and  $y$  are the state, control, and output vectors respectively. The parameter matrices  $\Phi$ ,  $\Gamma$ , and  $C$  are  $n \times n$ ,  $n \times m$ , and  $m \times n$  constant matrices, respectively. Note that we assume that the number of outputs equals the number of inputs, which is less than number of states. Hence, we can identify  $m$  input-output pairs and, thus  $m$  SISO subsystems. We also assume that each component of  $y$  is a component of  $x$ , i.e., that the states are "sensor coordinates".

To identify the  $m$  subsystems, the state vector is partitioned into  $m$  subvectors:

$$x = \begin{bmatrix} x_1^T & x_2^T & \dots & x_m^T \end{bmatrix}.$$

each subvector is associated with one of the  $m$  input-output pairs. The matrices  $\Phi$  and  $\Gamma$  are then partitioned accordingly:

$$\Phi = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \dots & \Phi_{1m} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \Phi_{m1} & \Phi_{m2} & \dots & \Phi_{mm} \end{bmatrix}, \quad \Gamma = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & \dots & \Gamma_{1m} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \Gamma_{m1} & \Gamma_{m2} & \dots & \Gamma_{mm} \end{bmatrix}$$

where the  $\Phi_{ij}$  and  $\Gamma_{ij}$  are blocks of appropriate dimension. The on-diagonal blocks ( $\Phi_{ij}$  and  $\Gamma_{ij}$ ) are used to define the SISO subsystems, which the off-diagonal blocks determine the

interaction among the subsystems.

The completely-decomposed model is obtained by setting all off-diagonal blocks to zero. The state equation, then, for the decoupled model is

$$x_{k+1} = \hat{\Phi} * x_k + \hat{\Gamma} * u_k$$

where

$$\hat{\Phi} = \begin{bmatrix} \Phi_{11} & 0 & 0 & 0 & 0 \\ 0 & \Phi_{22} & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & \Phi_{mm} \end{bmatrix}$$

and

$$\hat{\Gamma} = \begin{bmatrix} \Gamma_{11} & 0 & 0 & 0 & 0 \\ 0 & \Gamma_{22} & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & 0 \\ 0 & 0 & 0 & 0 & \Gamma_{mm} \end{bmatrix}$$

## B: A SISO servo design

---

In this section, a technique for **SISO** servo design is discussed. The technique is a mixture of classical and state-space methods. The intended application is for a system whose output is to track a reference input, which consists of manually (*i.e.*, slowly) adjusted set points.

A digital **SISO** plant is shown as block diagram in Figure 6. The dynamic equations of the plant are

$$x_{k+1} = \Phi * x_k + \Gamma * u_k \quad (13)$$

$$y = C * x_k \quad (14)$$

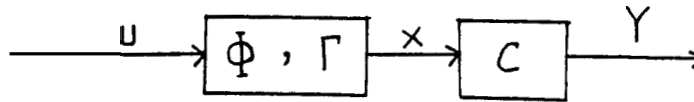


Figure 6. A Digital SISO Plant.

The open-loop transfer function,  $G(z)$ , of the plant

$$G(z) = y(z)/u(z) = C * (zI - \Phi)^{-1} * \Gamma$$

$$= \left( \sum_{i=0}^{n-1} b_i * z^i \right) / \left( z^n + \sum_{i=0}^{n-1} a_i * z^i \right) \quad (15)$$

Compensation for the plant consists of SVFB for transient control and a feedforward PI controller for steady state control. A block diagram of with the controller added to the plant is shown in Figure 7.

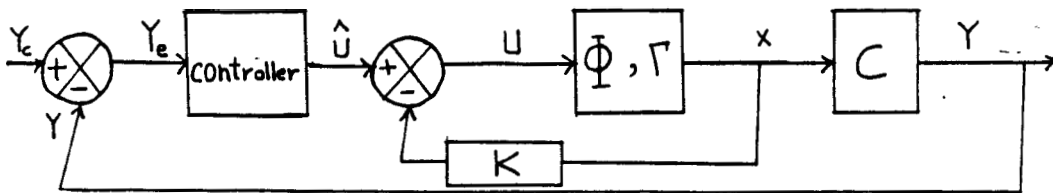


Figure 7. The Plant with a Controller.

The integral control is used for robust tracking of the reference signal  $Y_c$ . A disadvantage of integral control is a

sluggish transient response. Thus a proportional control term is also used.

An additional term,  $k_d * y$ , is subtracted from PI controller in order to minimize integrator build up during the transient portion of the response to a change in set points. Selection of  $k_d$  will be discussed later.

A block diagram of the overall system is shown in Figure 8.

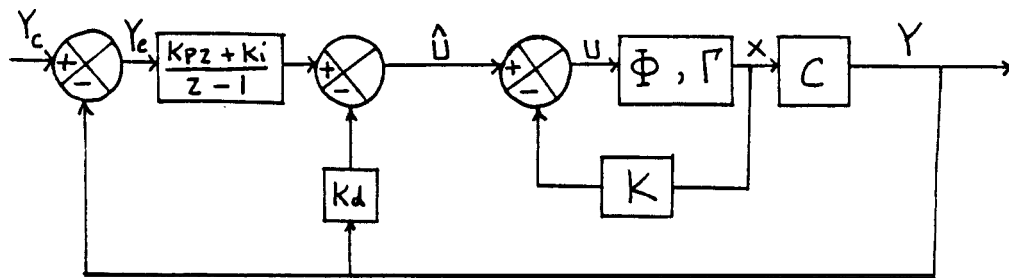


Figure 8. Closed-loop SISO Servo System.

The unity feedback loop in Figure 8 is analyzed in terms of the transfer function from  $u$  to  $y$ . The inner loop transfer function is

$$\begin{aligned} \hat{G}(z) &= y(z) / \hat{u}(z) = C * (Z * I - \Phi + \Gamma * K)^{-1} * \Gamma \\ &= \left( \sum_{i=0}^{n-1} b_i * z^i \right) / \left( z^n + \sum_{i=0}^{n-1} \hat{a}_i * z^i \right) \end{aligned} \quad (16)$$

Let

$$D(z) = (k_p z + k_i) / z - 1, \quad (17)$$

$$D_1(z) = [(k_p z + k_i) / z - 1] + k_d \quad (18)$$

and the overall transfer function be

$$H(z) = Y(z)/Y_C(z) = \left( \sum_{i=0}^n \beta_i * z^i \right) / \left( z^{n+1} + \sum_{i=0}^n p_i * z \right) \quad (19)$$

To achieve  $H(z)$ , from Figure 8, we have

$$\begin{aligned} \hat{u}(z) &= D(z) * y_e^{-k_d} * y = [(k_p z + k_i) / z - 1] * (y_c - y) - k_d * y \\ &= D(z) * y_c - D_1(z) * y \end{aligned} \quad (20)$$

and

$$\begin{aligned} y &= G(z) * u = G(z) * D(z) * y_c - G(z) * D_1(z) * y \\ &= [G(z) * D(z) / 1 + G(z) * D_1(z)] * y_c = H(z) * y_c. \end{aligned} \quad (21)$$

Therefore

$$\begin{aligned} H(z) &= \left\{ \left[ \left( \sum_{i=0}^{n-1} b_i * z^i \right) / \left( z^n + \sum_{i=0}^{n-1} \hat{a}_i * z^i \right) \right] * \left[ (k_p z + k_i) / z - 1 \right] \right\} \\ &/ \left\{ \left[ \left( 1 + \sum_{i=0}^{n-1} b_i * z^i \right) / \left( z^n + \sum_{i=0}^{n-1} \hat{a}_i * z^i \right) \right] * \left[ k_p z + k_i + k_d * (z - 1) \right] \right\} \\ &/ (z - 1) \end{aligned} \quad (22)$$

or

$$\begin{aligned} H(z) &= \left[ (k_p z + k_i) * \left( \sum_{i=0}^{n-1} b_i * z^i \right) \right] \\ &/ \left\{ \left[ \left( z^n + \sum_{i=0}^{n-1} \hat{a}_i * z^i \right) * (z - 1) \right] \right. \\ &\left. + \left[ \left( \sum_{i=0}^{n-1} b_i * z^i \right) * (k_p z + k_i + k_d z - k_d) \right] \right\}. \end{aligned} \quad (23)$$

Then the denominator of  $H(z)$  is

$$\begin{aligned}
D_H(z) &= z^{n+1} - z^n + \sum_{i=0}^{n-1} \hat{a}_i * z^{i+1} - \sum_{i=0}^{n-1} \hat{a}_i * z^i \\
&+ (k_p + k_d) * \sum_{i=0}^{n-1} b_i * z^{i+1} + (k_i - k_d) * \sum_{i=0}^{n-1} b_i * z^i \\
&= z^{n+1} + \sum_{i=1}^n [\hat{a}_{i-1} + (k_p + k_d) * b_{i-1}] * z^i \\
&+ \sum_{i=0}^{n-1} [-\hat{a}_i + (k_i - k_d) * b_i] * z^i \tag{24}
\end{aligned}$$

or

$$\begin{aligned}
D_H(z) &= z^{n+1} + z^n * [-1 + \hat{a}_{n-1} + (k_p + k_d) * b_{n-1}] \\
&+ z^{n-1} * [\hat{a}_{n-2} + (k_i + k_d) * b_{n-2} - \hat{a}_{n-1} + (k_i - k_d) * b_{n-1}] \\
&+ z^{n-2} * [\hat{a}_{n-3} + (k_p + k_d) * b_{n-3} - \hat{a}_{n-2} + (k_i - k_d) * b_{n-2}] \\
&+ z^{n-3} * [\hat{a}_{n-4} + (k_p + k_d) * b_{n-4} - \hat{a}_{n-3} + (k_i - k_d) * b_{n-3}] \\
&+ \dots + [-\hat{a}_0 + (k_i - k_d) * b_0] \tag{25}
\end{aligned}$$

Now, a pole placement approach [4] is used to find the gains by forcing the system (closed-loop) characteristic polynomial to be identical to the desired control characteristic equation. This is done when the elements of K are picked from the control law design in such a way that the roots of the closed-loop characteristic equation of the system are in desirable locations. Given the desired root locations, say

$$z_i = p_1, p_2, \dots, p_n \tag{26}$$

the desired control characteristic equation is

$$c(z) = (z - p_1) * (z - p_2) * \dots * (z - p_n) = z^{n+1} + \sum_{i=0}^n c_i * z^i \tag{27}$$

So, in this design,  $D_H(z)$  is the closed-loop characteristic

equation of the system and the desired pole locations are matched to the coefficient of  $D_H(z)$  as following:

if  $d_1 = k_p + k_d$  and  $d_0 = k_i - k_d$  then

$$a_{n-1} + b_{n-1} * d_1 = c_n + 1 \quad (28)$$

$$-a_{n-1} + a_{n-2} + b_{n-2} * d_1 + b_{n-1} * d_0 = c_{n-1} \quad (29)$$

$$-a_{n-2} + a_{n-3} + b_{n-3} * d_1 + b_{n-2} * d_0 = c_{n-2} \quad (30)$$

$$-a_{n-3} + a_{n-4} + b_{n-4} * d_1 + b_{n-3} * d_0 = c_{n-3} \quad (31)$$

.....  
 .....

$$-a_0 + b_0 * d_0 = c_0 \quad (32)$$

Then

$$a_{n-1} + b_{n-1} * d_1 = c_n + 1 \quad (33)$$

$$-a_{n-2} + (b_{n-1} + b_{n-2}) * d_1 + b_{n-1} * d_0 = 1 + c_n + c_{n-1} \quad (34)$$

$$-a_{n-3} + d_1 * \sum_{i=n-3}^{n-1} b_i + d_0 * \sum_{i=n-2}^{n-1} b_i = 1 + \sum_{i=n-2}^n c_i \quad (35)$$

.....  
 .....

$$d_1 * \sum_{i=0}^{n-1} b_i + d_0 * \sum_{i=0}^{n-1} b_i = 1 + \sum_{i=0}^{n-1} c_i \quad (36)$$

From equation 36:

$$\begin{aligned}
& \left( \sum_{i=0}^{n-1} b_i \right) * (d_1 + d_0) \quad \dots \\
& = \left( \sum_{i=0}^{n-1} b_i \right) * (k_i + k_p) = N_G(1) * (k_i + k_p) = D_H(1) \quad (37)
\end{aligned}$$

where  $N_G$  is the numerator of  $G(z)$ . Then we can solve for  $k_i + k_p = d_1 + d_0$  if  $N_G \neq 0$ , i.e., if  $z=1$  is not a zero of the original open-loop transfer function  $G(z)$ .

Now there are  $n$  equations and  $n+2$  unknowns. Thus, two additional constraints must be specified. If the system is to be type one with velocity constant  $K_V$  [4], we must have  $1/K_V$  error to a unit ramp.  $K_V$  is given by

$$K_V = \lim_{z \rightarrow 1} (z-1)^2 * [1/(z-1) * T * z] * G_f(z), \quad (38)$$

where  $G_f(z)$  is the feedforward transfer function of the unity feedback loop. Then

$$H(z) = G_f(z) / [1 + G_f(z)], \quad \text{so} \quad G_f(z) = H(z) / [1 - H(z)].$$

Also, since  $d_1 = k_p + k_d$  and  $d_0 = k_i - k_d$ , in order to add one additional constraint, the value of the velocity error constant  $K_V$  will be specified as

$$\begin{aligned}
K_V &= \lim_{z \rightarrow 1} [(z-1)/(T * z)] * [H(z)/(1-H(z))] \\
&= \lim_{z \rightarrow 1} [(z-1)/(T * z)] * [(N_H/D_H)/(1-N_H/D_H)] \\
&= \lim_{z \rightarrow 1} [(z-1)/(T * z)] * [(N_H/D_H - N_H) / (D_H - N_H)] = \lim_{z \rightarrow 1} [(z-1)/(T * z)] \\
&\quad * \{ [(k_p z + k_i) * N_G(z)] / [(z^n + \sum_{i=0}^{n-1} \hat{a}_i z^i) * (z-1) + k_d * (z-1)] \} \\
&= \{ [1 * (k_p + k_i) * \sum_{i=0}^{n-1} b_i] * [(T * z) * (1 + \sum_{i=0}^{n-1} \hat{a}_i + k_d)] \}
\end{aligned}$$



or

$$(1 + \sum_{i=0}^{n-1} \hat{a}_i + k_d) * T * K_v - (k_p + k_i) * \sum_{i=0}^{n-1} b_i = 0 \quad (39)$$

A second constraint is added by selecting  $k_d$  to minimize integrator build up. Without  $k_d$  the integrator input is  $(k_i + k_p) / T * e$ , where  $e = y_c - y$  and  $\dot{y}$  are similar in shape for a good design. Thus, integrator input can be reduced by selecting  $k_d$  so that

$$[(k_i + k_p) / T] * e - k_d * \dot{y} = 0 \quad (40)$$

a nominal value of  $k_d$  is chosen from the steady-state condition

$$[(k_i + k_p) / T] * e_{ss} - k_d * \dot{y}_{ss} = 0, \quad (41)$$

where

$$e_{ss} = R / K_v \quad (42)$$

and

$$\dot{y}_{ss} = R \quad (43)$$

when  $y_c(t)$  is a ramp of shape  $R$ . Then

$$k_d = (k_i + k_p) / (T * K_v) \quad (44)$$

The above equations are solved for  $k_i$ ,  $k_d$ ,  $k_p$ , and the coefficients  $\hat{a}_i$  of  $G(z)$ . A BASIC language program for the solution is listed on p. 53 of appendix A.

Next, the SVFB gains,  $K = [k_1, k_2, \dots, k_n]$ , are found by

using Ackermann's formula [4] for pole-placement design.

$$K = e_n^t * e_x^{-1} * \alpha_c(\Phi) \quad (45)$$

where  $\alpha_c(\Phi)$  is the closed-loop characteristic polynomial with the open-loop matrix  $\Phi$  substituted for the complex variable  $z$ :

$$\alpha_c(\Phi) = \Phi^n + \alpha_1 * \Phi^{n-1} + \dots + \alpha_n * I. \quad (46)$$

$e_n$  is the  $n$ th unit vector

$$e_n^t = [0 \ 0 \ \dots \ 0 \ 1] \quad (47)$$

and  $e_x$  is the controllability matrix [4]

$$e_x = [\Gamma * \Phi * \Gamma \ \dots \ \Phi^{n-1} * \Gamma] \quad (48)$$

At this point all the gains are calculated and can be used for simulation.

C: Least square design to reduce subsystem interaction

In a multivariable system, there are interactions between the subsystems discussed in section A of this chapter. These interactions can be reduced as discussed below.

Suppose the actual plant dynamics are:

$$x_{k+1} = \Phi * x_k + \Gamma * u_k \quad (49)$$

We want to force the actual plant dynamics to be equal to the assumed plant dynamics used in the SISO designs given in section B. These dynamics are obtained from the decoupled equations of section A, and may be represented as

$$x_{k+1} = \hat{\Phi} * x_k + \hat{\Gamma} * u_k \quad (50)$$

where  $\hat{\Phi}$  and  $\hat{\Gamma}$  are block diagonal matrices from  $\Phi$  and  $\Gamma$ , and  $\hat{u}_k$  is the control vector from the SISO compensators designed in section B. From equations 49 and 50 we have:

$$x_{k+1} = \Phi * x_k + \Gamma * u_k = \hat{\Phi} * x_k + \hat{\Gamma} * u_k \quad (51)$$

or

$$\Gamma * u_k = \hat{\Gamma} * \hat{u}_k - (\Phi - \hat{\Phi}) * x_k \quad (52)$$

An exact solution for  $u_k$  almost never exists, so a least-squares approximation is used. The result is [5]

$$u_k = [\Gamma^* * \Gamma] * \hat{u}_k - [\Gamma^* * (\Phi - \hat{\Phi})] * x_k \quad (53)$$

where

$$\Gamma^* = (\Gamma^t * Q * \Gamma)^{-1} * \Gamma^t * Q \quad (54)$$

is the generalized inverse of  $\Gamma$ , and  $Q$  is a weighting matrix which can be chosen arbitrarily.

A BASIC language program for calculation of  $\Gamma^*$  is listed on pp. 54-6 of appendix A.

## CHAPTER III

### APPLICATION

This section consists of a lateral autopilot design for B-737 aircraft which starts with a model development. Then the design technique developed in the previous chapter is applied.

The lateral controls (aileron and rudder) on a conventional airplane have three principal functions (6):

- 1- to provide trim in the presence of asymmetric thrust associated with power plant failure;
- 2- to provide corrections for unwanted motions associated with atmospheric turbulence or other random events;
- 3- to provide for turning maneuvers, *i.e.*, rotation of the velocity vector in a horizontal plane.

These purposes are served by having the controls generate aerodynamic moments about the X and Z axes, *i.e.*, rolling and yawing moments.

#### MODEL DEVELOPMENT

The linearized equations of an aircraft's lateral motion

are fourth-order. They are given in equation 55 in analog form for a Boeing 737 in a-final approach to landing configuration.

$$\dot{x} = A * x + B * u \quad (55)$$

where

$$A = \begin{bmatrix} -.1380 & -.9940 & 0.1551 & 0.6655E-1 \\ 0.6385 & -.1399 & 0 & -.1305 \\ 0 & 0.5827E-1 & 0 & 1 \\ -3.474 & 0.8081 & 0 & -1.476 \end{bmatrix},$$

$$B = \begin{bmatrix} 0.6197E-3 & -.1537E-3 \\ -.1007E-1 & 0.4720E-2 \\ 0 & 0 \\ 0.8577E-2 & 0.4004E-1 \end{bmatrix}$$

$$u = \begin{bmatrix} u_r \\ u_a \end{bmatrix}, \quad x = \begin{bmatrix} v \\ r \\ \phi \\ p \end{bmatrix}$$

$u_r$  (degrees) is rudder control and  $u_a$  (degrees) is aileron control.  $v$  (rad) is sideslip angle,  $r$  (rad/sec) is yaw rate,  $\phi$  (rad) is bank angle, and  $p$  (rad/sec) is roll rate.

In digital compensation the system is studied as seen from a computer. The computer receives measurements from the process at discrete times and transmits new control signals at discrete times. The goal then is to describe the change in the signals from sample to sample and disregard the behavior between the

samples. The calculations required to sample a continuous-time system are the **evaluation** of a matrix exponential and integration of a matrix exponential. So in order to find the discrete-model open-loop parameters, the following formulas are used:

$$\Phi = e^{AT} \quad (56)$$

$$\Gamma = \left[ \int_0^T e^{A(T-t)} * dt \right] * B . \quad (57)$$

T is the sampling period and was chosen to produce eight samples per second.

Then the discrete-model open-loop parameters for the system were found to be

$$\Phi = \begin{bmatrix} 9.761E-1 & -1.213E-1 & 1.918E-2 & 9.583E-3 \\ 8.153E-2 & 9.769E-1 & 7.855E-4 & -1.438E-2 \\ -2.491E-2 & 1.417E-2 & 9.998E-1 & 1.140E-1 \\ -3.882E-1 & 1.162E-1 & -3.909E-3 & 8.289E-1 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 1.587E-4 & -3.225E-5 \\ -1.250E-3 & 5.452E-4 \\ 5.817E-5 & 2.975E-4 \\ 8.923E-4 & -4.603E-3 \end{bmatrix}$$

### SUBSYSTEM DESIGNS

A design is developed separately for each subsystem. The aileron is used to control the roll axis dynamics, and rudder the yaw axis dynamics. Then the coupling term between two subsystems

were put back into the  $\Phi$  and  $\Gamma$  matrices and their effects on the transient responses, pole locations, and controls (aileron and rudder) were examined.

The overall system is viewed from a block diagram as shown in Figure 9.

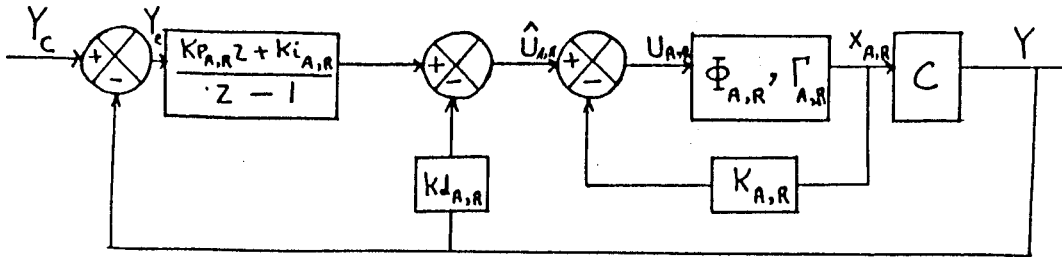


Figure 9. Overall System.

A control law of the form  $u = -k \cdot x$  shows that there are eight entries in the gain matrix to be selected, and specification of four closed-loop roots will clearly leave many possible values of  $k$  which will meet the specifications. This is an example of the incompleteness of the pole-placement approach to multivariable system design [4].

A decoupling approach which removes the ambiguity is to restrict the control law to

$$\begin{bmatrix} u_r \\ u_a \end{bmatrix} = - \begin{bmatrix} k_{11} & k_{12} & 0 & 0 \\ 0 & 0 & k_{23} & k_{24} \end{bmatrix} * \begin{bmatrix} v \\ r \\ \beta \\ p \end{bmatrix} \quad (58)$$

This makes good physical sense because the rudder primarily yaws the aircraft about a vertical axis ( $r$ -motion), thus directly

causing sideslip ( $v$ ) while the aileron primarily rolls the aircraft about an **axis** through the nose, thus causing changes in  $\phi$  and  $p$ . Given an achievable set of desired pole locations, there are unique values of the four non-zero components of  $K$ .

A further decoupling that would permit an easy gain calculation is to assume that the dynamic lateral equations are of the form

$$\begin{bmatrix} v \\ r \\ \phi \\ p \end{bmatrix}_{k+1} = \begin{bmatrix} \phi_{11} & \phi_{12} & 0 & 0 \\ \phi_{21} & \phi_{22} & 0 & 0 \\ 0 & 0 & \phi_{33} & \phi_{34} \\ 0 & 0 & \phi_{43} & \phi_{44} \end{bmatrix} * \begin{bmatrix} v \\ r \\ \phi \\ p \end{bmatrix}_k + \begin{bmatrix} \gamma_{11} & 0 \\ \gamma_{21} & 0 \\ 0 & \gamma_{32} \\ 0 & \gamma_{42} \end{bmatrix} * \begin{bmatrix} u_r \\ u_a \end{bmatrix} \quad (59)$$

and that the control law is given by equation 58.

This made some physical sense but ignored important coupling between the two aircraft modes. It decoupled the system into second-order systems for which the method of chapter II was easily applied to obtain the gains instead of using a fourth-order system, where the calculations are done with difficulty.

Aileron equations of motion are of the form

$$x_{k+1} = \Phi_A * x_k + \Gamma_A * u_k \quad (60)$$

or

$$\begin{bmatrix} \phi \\ p \end{bmatrix}_{k+1} = \begin{bmatrix} \phi_{33} & \phi_{34} \\ \phi_{43} & \phi_{44} \end{bmatrix} * \begin{bmatrix} \phi \\ p \end{bmatrix}_k + \begin{bmatrix} \gamma_{32} \\ \gamma_{42} \end{bmatrix} * [u_a] \quad (61)$$

so the control law is



$$[u_a] = -[k_{23} \ k_{24}] * \begin{bmatrix} \theta \\ p \end{bmatrix} \quad (62)$$

Ackermann's formula was used to find the gains. The process is as follows:

$$\Phi_A * \Gamma_A = \begin{bmatrix} 9.998E-1 & 1.140E-1 \\ -3.909E-3 & 8.289E-1 \end{bmatrix} * \begin{bmatrix} 2.975E-4 \\ 4.603E-3 \end{bmatrix} = \begin{bmatrix} 8.221E-4 \\ 3.814E-3 \end{bmatrix}$$

$$e_{Ax} = [\Gamma_A \ \Phi_A \ \Gamma_A] = \begin{bmatrix} 2.975E-4 & 8.221E-4 \\ 4.603E-3 & 3.814E-3 \end{bmatrix}$$

$$[\Gamma_A \ \Phi_A \ \Gamma_A]^{-1} = \begin{bmatrix} -1439.54 & 310.29 \\ 1737.67 & -112.39 \end{bmatrix}.$$

Also

$$\alpha_c(\Phi_A) = \Phi_A^2 + \alpha_1 \Phi_A + \alpha_2 I \quad (63)$$

$$\Phi_A^2 = \begin{bmatrix} 9.992E-1 & 2.085E-1 \\ -7.148E-3 & 6.867E-1 \end{bmatrix}, \quad e_{Ax}^T = [0 \ 1].$$

Then

$$K_A = e_{Ax}^T * e_{Ax}^{-1} * \alpha_c(\Phi_A) \quad (64)$$

$$K_A = [1735.94 + 1724.18 * \alpha_1 + 1737.33 * \alpha_2 \quad -89.54 - 99.89 * \alpha_1 - 112.29 * \alpha_2]$$

By adding two more equations to the BASIC program on p. 53 of appendix A another basic program was developed. This program solved for the second-order aileron system's gains

$k_i, k_p, k_d, k_{23}, k_{24}$  and  $a_i^A$ 's. The program is listed on p. 57 of appendix A.

The pole locations in continuous-time are given as a real pole  $p = -2$  and a complex pair with damping factor of .8 and natural frequency of 2 rad/sec. After converting to equivalent discrete-time poles, the closed-loop denominator for aileron is

$$z^3 - 2.3979z^2 + 1.9312z - .5220. \quad (65)$$

The coefficients of equation 65 were put in BASIC program of p. 57 of appendix A and  $K_v$  was chosen to be 1.50. The gains were calculated to be  $k_p = 2.0976, k_d = 1.8328, k_i = -1.7540, k_{23} = -7995,$  and  $k_{24} = 79.0870$ . The gains were checked using the BASIC program on pp. 60-4 of appendix A.

A block diagram for aileron control system with the obtained gains is shown in Figure 10.

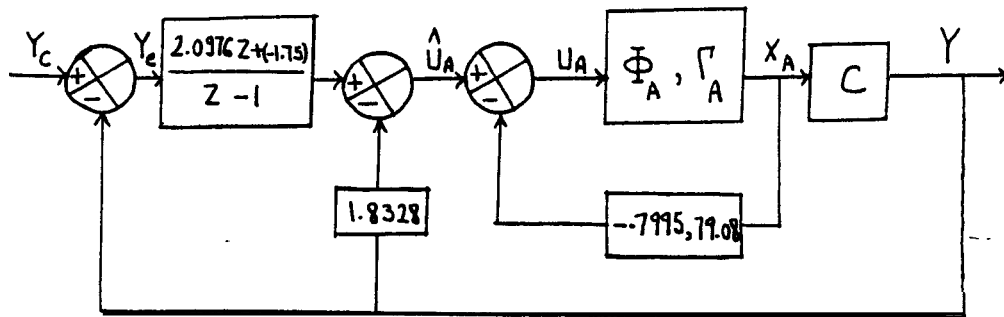


Figure 10. Aileron Control System.

At this point an ACSL program was developed for simulation purposes. The program is illustrated on p. 58 of appendix A. The simulation results are given in Figures 11 and 12. As it is shown in the plot of bank angle ( $\phi$ ), there is a slight overshoot in

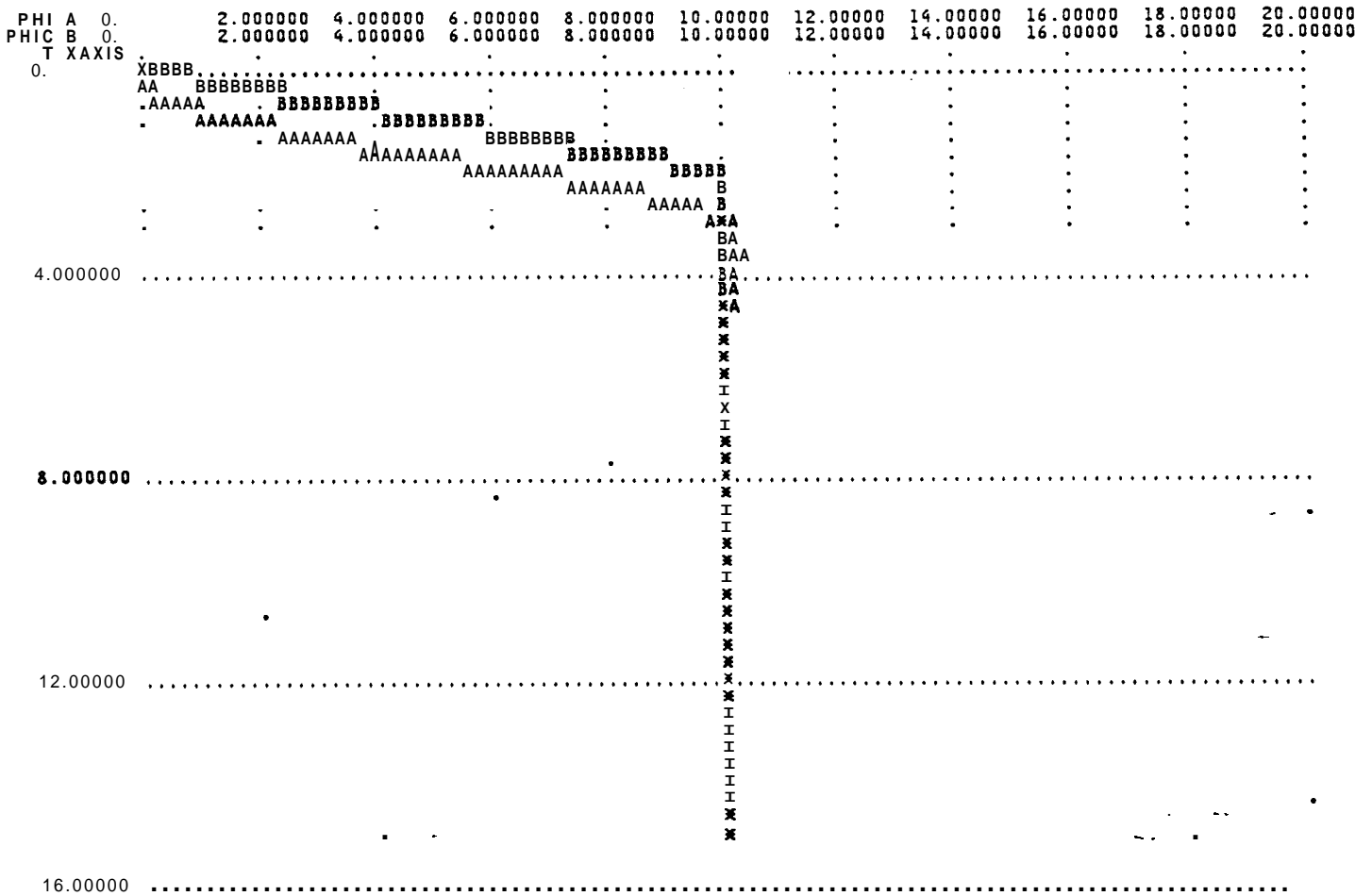


Figure 11: Bank angle (A) and bank angle command (B) (in degrees) vs time (in seconds) [decoupled]

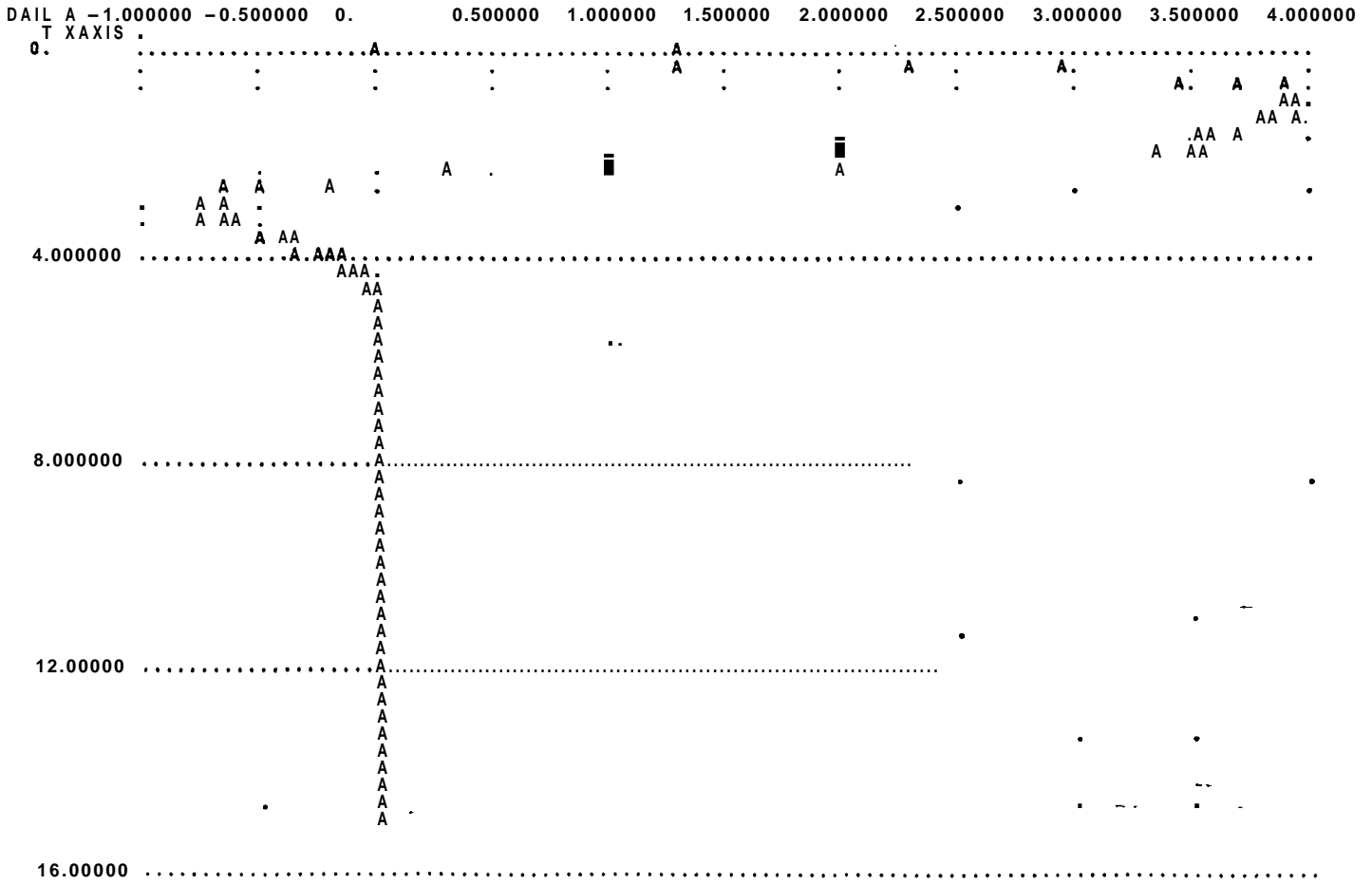


Figure 12: Aileron (A) [decoupled]

the response but the response settles nicely. In the control plot (dail), there was a little oscillation but it settled down quickly. To tune the response there are two options: one is to change the pole locations and the other is to change the value of  $k_d$ . It was felt that there was no need to change the pole locations because the response was good. So  $k_d$  was changed and its effects are illustrated in Figures 13 and 14.

Rudder equations of motion are of the form

$$x_{k+1} = \Phi_R * x_k + \Gamma_R * u_k \quad (66)$$

or

$$\begin{bmatrix} v \\ r \end{bmatrix}_{k+1} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} * \begin{bmatrix} v \\ r \end{bmatrix}_k + \begin{bmatrix} \gamma_{11} \\ \gamma_{21} \end{bmatrix} * [u_r] \quad (67)$$

so the control law is

$$[u_r] = -[k_{11} \quad k_{12}] * \begin{bmatrix} v \\ r \end{bmatrix} \quad (68)$$

The procedure is the same as the aileron design and is as follows:

$$\Phi_R * \Gamma_R = \begin{bmatrix} 9.761E-1 & -1.213E-1 \\ 8.153E-2 & 9.769E-1 \end{bmatrix} * \begin{bmatrix} 1.587E-4 \\ -1.250E-3 \end{bmatrix} = \begin{bmatrix} 3.065E-4 \\ -1.208E-3 \end{bmatrix}$$

$$C_{Rx} = [\Gamma_R \quad \Phi_R \Gamma_R] = \begin{bmatrix} 1.587E-4 & 3.065E-4 \\ -1.250E-3 & -1.208E-3 \end{bmatrix}$$



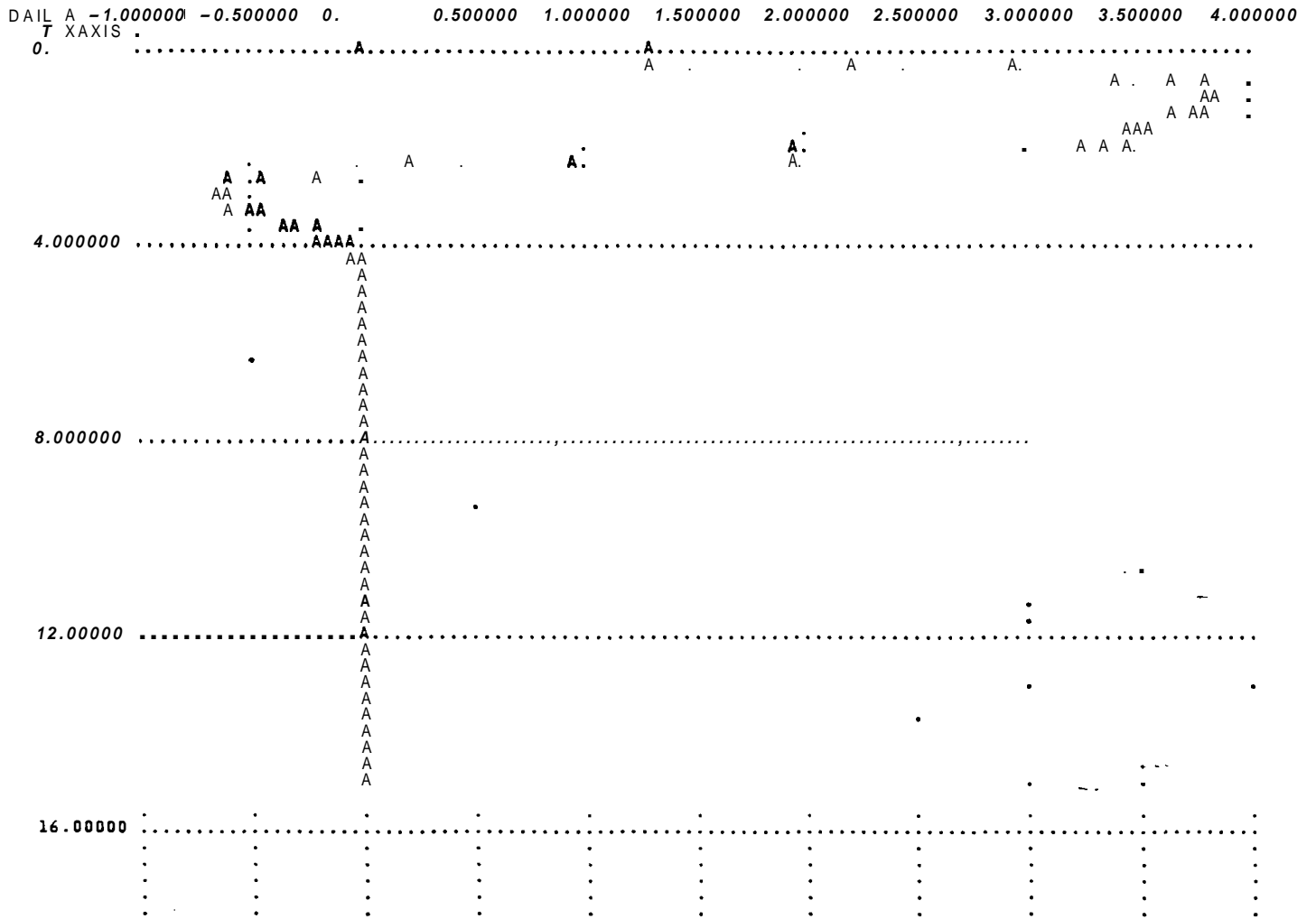


Figure 14: Aileron (A) [ $K_d$  changed]

$$e_{Rx}^{-1} = \begin{bmatrix} -6311.39 & -1601.36 \\ 6530.83 & 829.15 \end{bmatrix}$$

Also

$$\alpha_c(\Phi_R) = \Phi_R^2 + \alpha_1 * \Phi_R + \alpha_2 * I \quad (69)$$

$$\Phi_R^2 = \begin{bmatrix} 9.429E-1 & -2.369E-1 \\ 1.592E-1 & 9.444E-1 \end{bmatrix}, \quad e_2^T = [0 \quad 1].$$

Then

$$K_R = e_2^T * e_{Rx}^{-1} * \alpha_c(\Phi_R) \quad (70)$$

$$K_R = [6289.92 + 6442.34 * \alpha_1 + 6530.83 * \alpha_2 \quad -764.1 + 17.8 * \alpha_1 + 829.15 * \alpha_2]$$

By adding two more equations to the BASIC program on p. 53 of appendix A, another BASIC program was developed. This program solved for the second-order rudder system gains  $k_i, k_p, k_d, k_{11}, k_{12}$  and  $\hat{a}_i$ 's. The program is listed on p. 59 of appendix A.

The pole locations in continuous-time were given as a real pole  $p = -.132$  and a complex pair with damping factor of .9 and natural frequency of .8 rad/sec. The closed-loop denominator for the discrete-time rudder transfer function is then

$$z^3 - 2.8097 * z^2 + 2.6315 * z - .8216. \quad (71)$$

The coefficients of equation 71 were put in BASIC program of p. 59



of appendix A and  $K_v$  was chosen to be 1.25. The gains were calculated to be  $k_p = 0$ ,  $k_d = -.9859$ ,  $k_i = -.1550$ ,  $k_{11} = -74.615$ , and  $k_{12} = -67.629$ . The gains were checked using the BASIC program on pp. 60-4 of appendix A.

A block diagram for rudder control system with the obtained gains is illustrated in Figure 15.

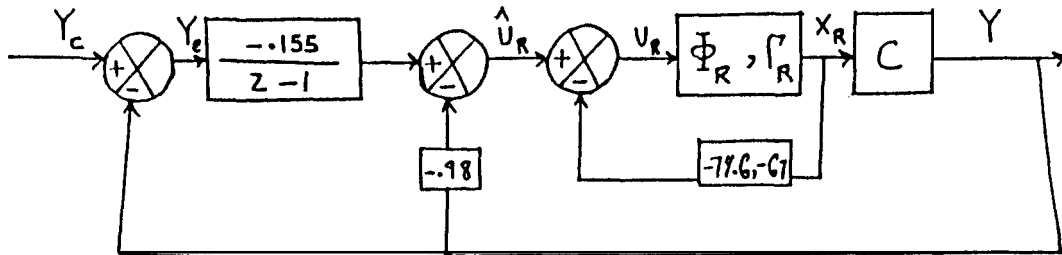


Figure 15. Rudder Control System.

The simulation results are given in Figures 16 and 17. As it is shown in the plot of yaw rate,  $r$ , the response has a little overshoot and is sluggish. But considering the nature of open-loop system, it is not a bad response. The control (drud) shows that only a nominal amount of rudder is used; the maximum rudder deflection is less than 1.3 degrees. The effect of tuning the response using  $k_d$  is shown in Figures 18 and 19.

#### REDUCTION OF INTERACTION BETWEEN SUBSYSTEMS

Now the gains for both systems are used together in the ACSL run time with the coupling terms. The simulation results are



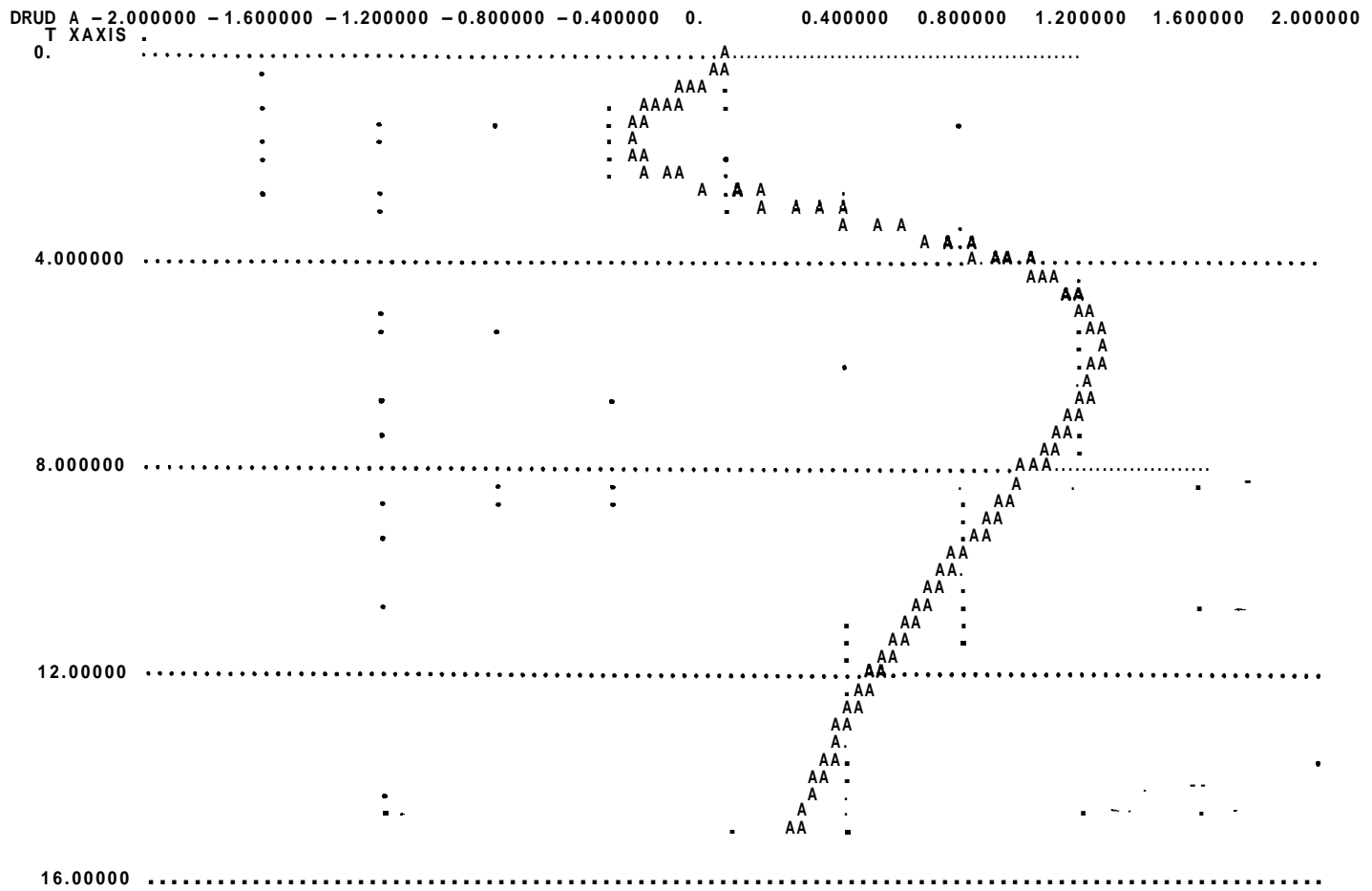


Figure 17: Rudder (A) [decoupled]



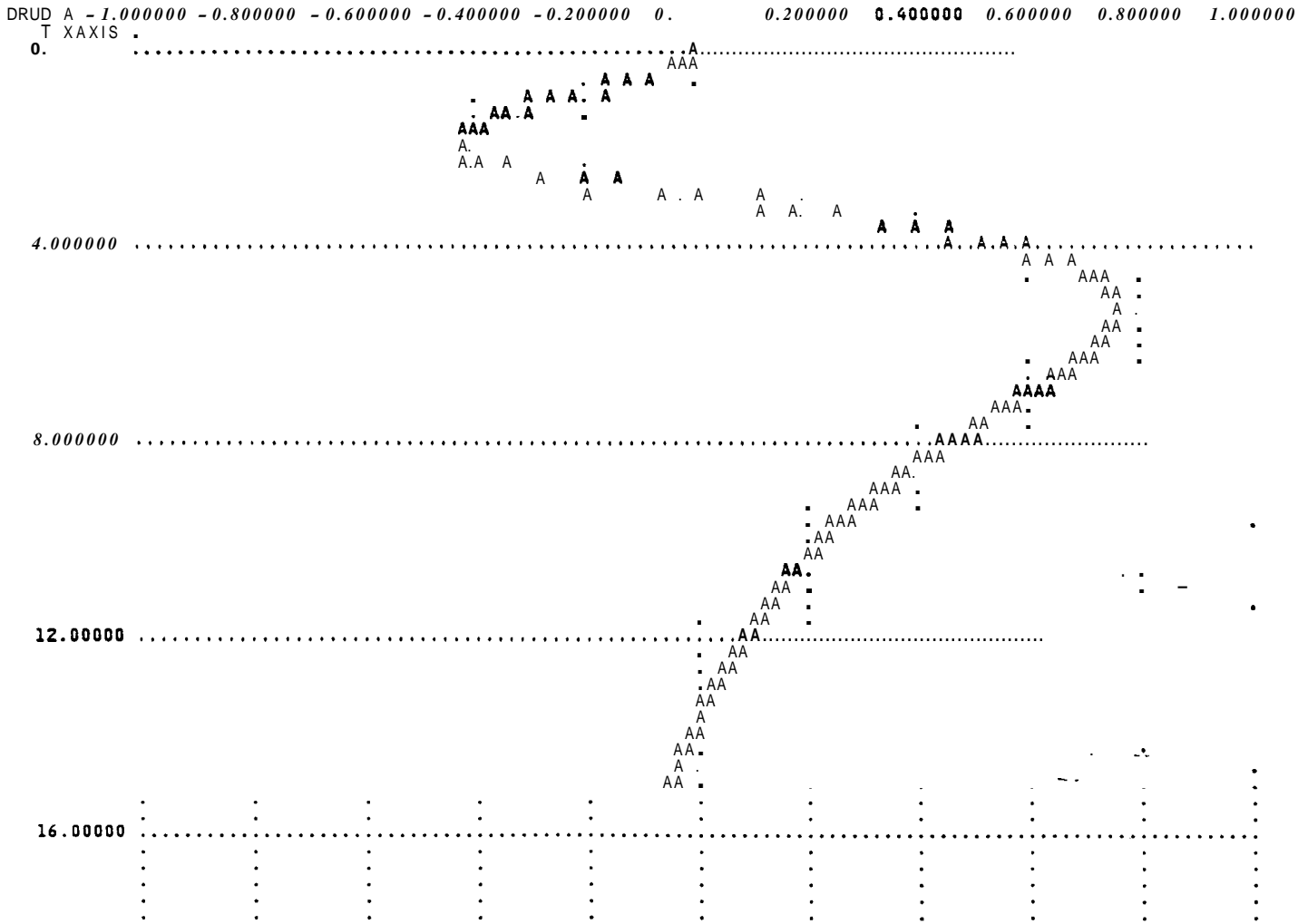


Figure 19: Rudder (A) [ $K_d$  changed]

given in Figures 20, 21, 22, and 23. As it is illustrated in the plot of the bank angle, the overshoot that was present in decoupled case is reduced; otherwise the response looks the same as before. The plot of yaw rate shows that oscillation is present but the response settles faster than the decoupled response. The plot for aileron shows that there is still oscillation as in the decoupled response and the response tends to settle late. The plot for rudder shows that the oscillation is present but the response settles better. However there is more rudder used than in the decoupled case.

By comparing the coupled and decoupled simulations, we realize that the decoupled responses were somewhat better than the coupled ones. In order to examine the effect of reducing the coupling terms, another simulation was run. First the BASIC program of pp. 54-6 of appendix A was used to calculate <sup>\*</sup>. The interaction reduction scheme discussed in chapter II was then included in the ACSL simulation. The simulation results are given in Figures 24, 25, 26, and 27. As it is shown from the plots the responses were very much the same as the decoupled cases for both subsystems.

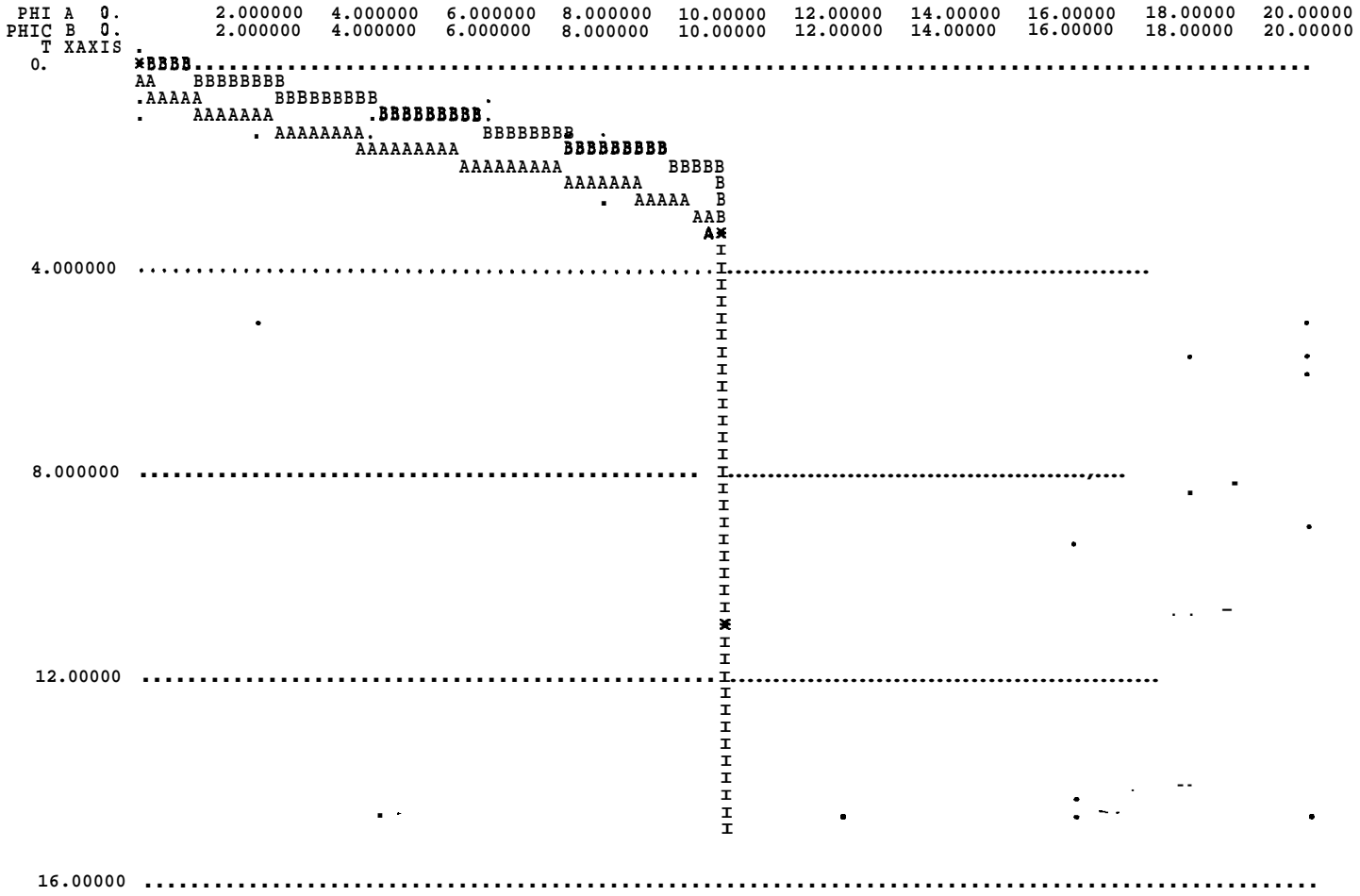


Figure 20: Bank angle (A) and bank angle command (B) (in degrees) vs time (in seconds) [coupling]

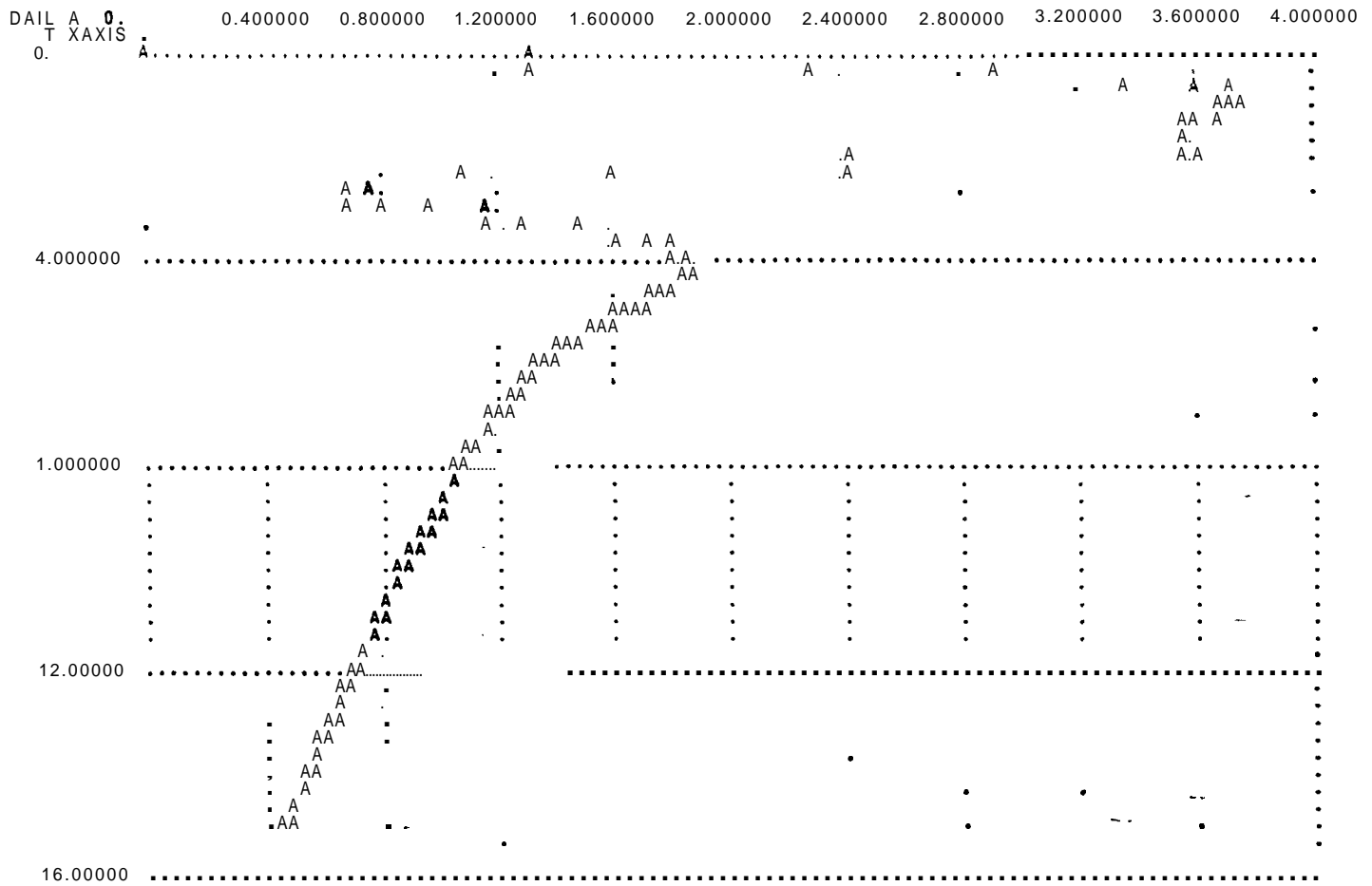


Figure 21: Aileron (A) (in degrees) vs time (in seconds) [coupling]



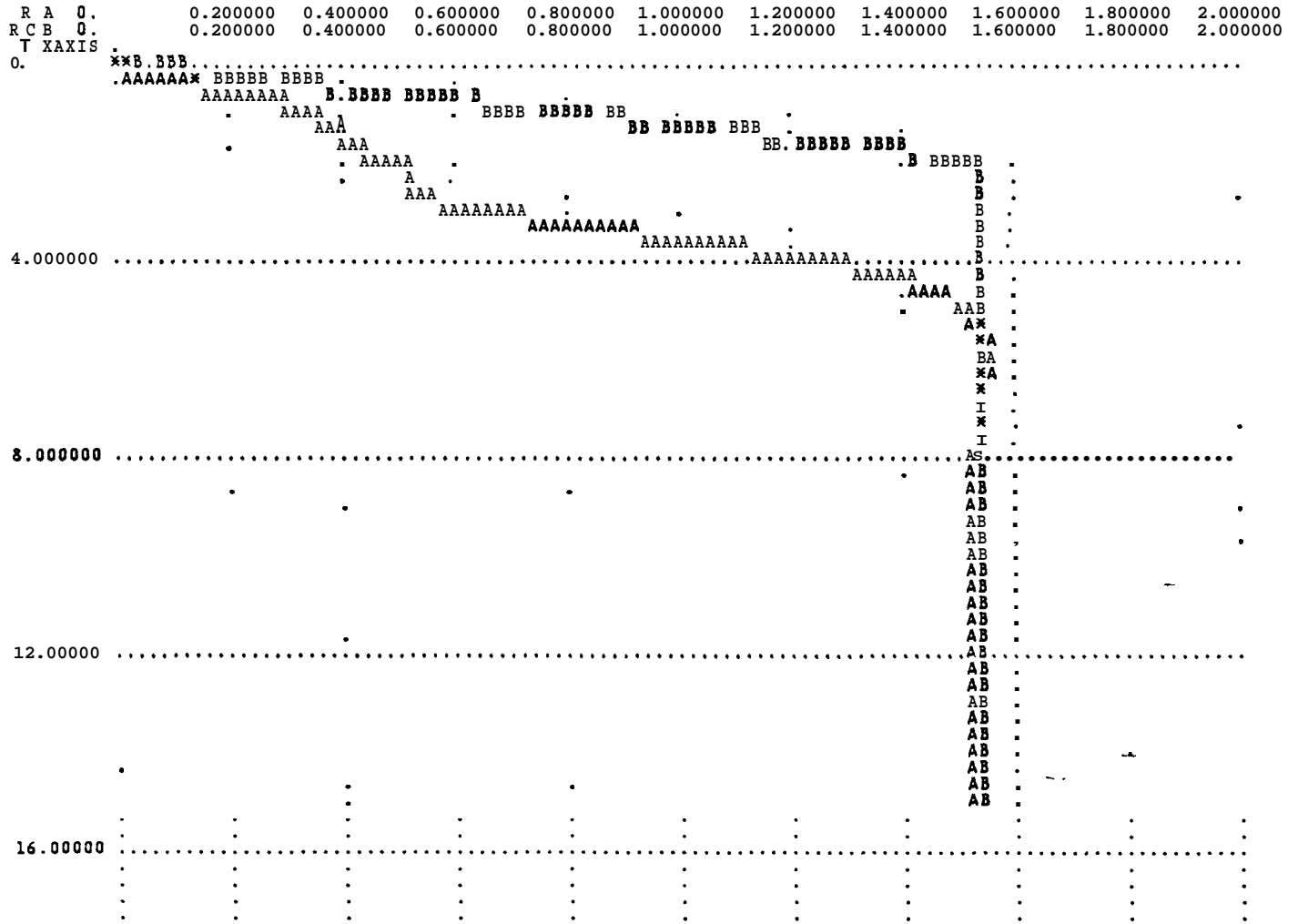


Figure 22: Yaw rate (A) and yaw rate command (B) (in degrees) vs time (in seconds) [coupling]

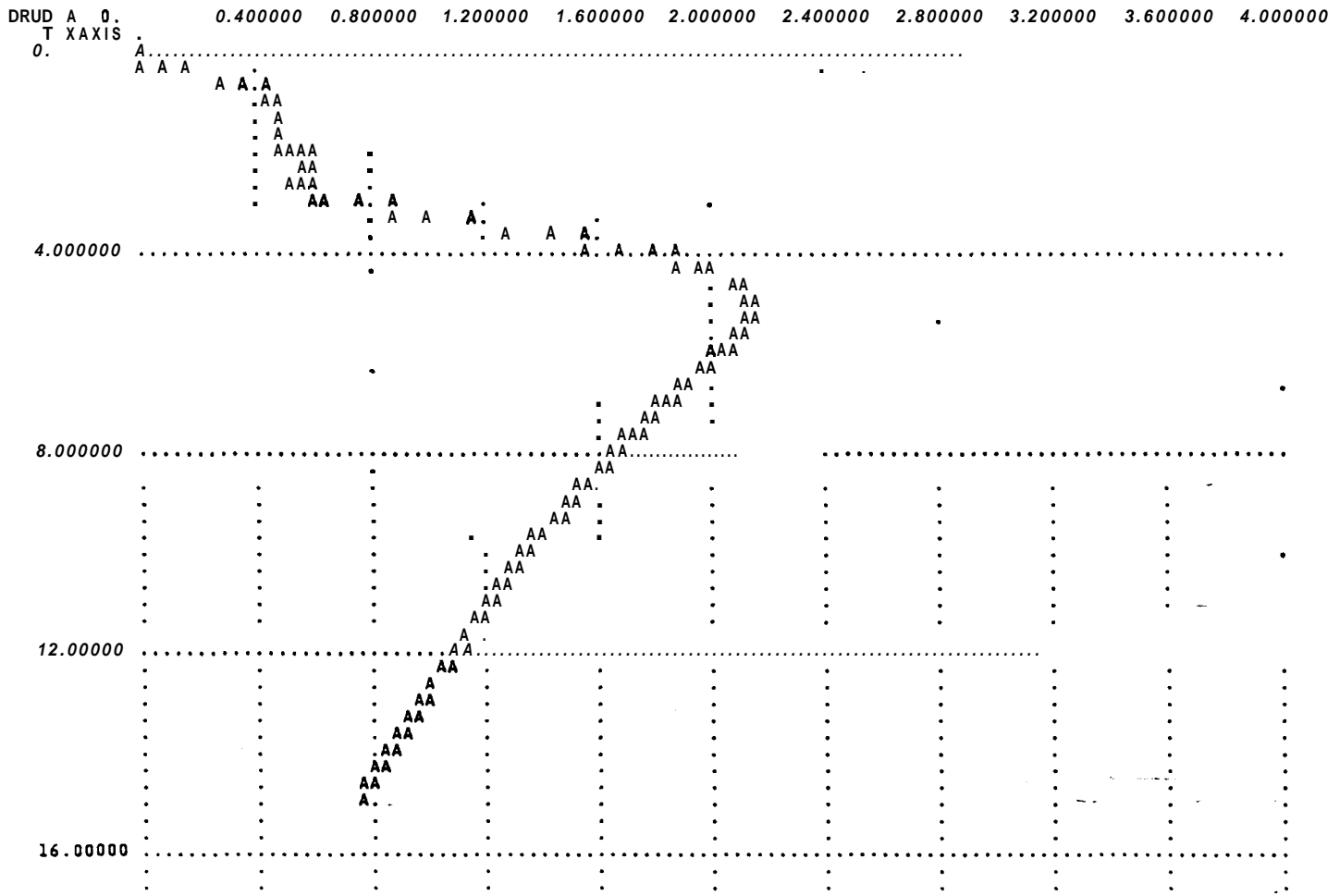


Figure 23: Rudder (A) (in degrees) vs time (in seconds) [coupling]

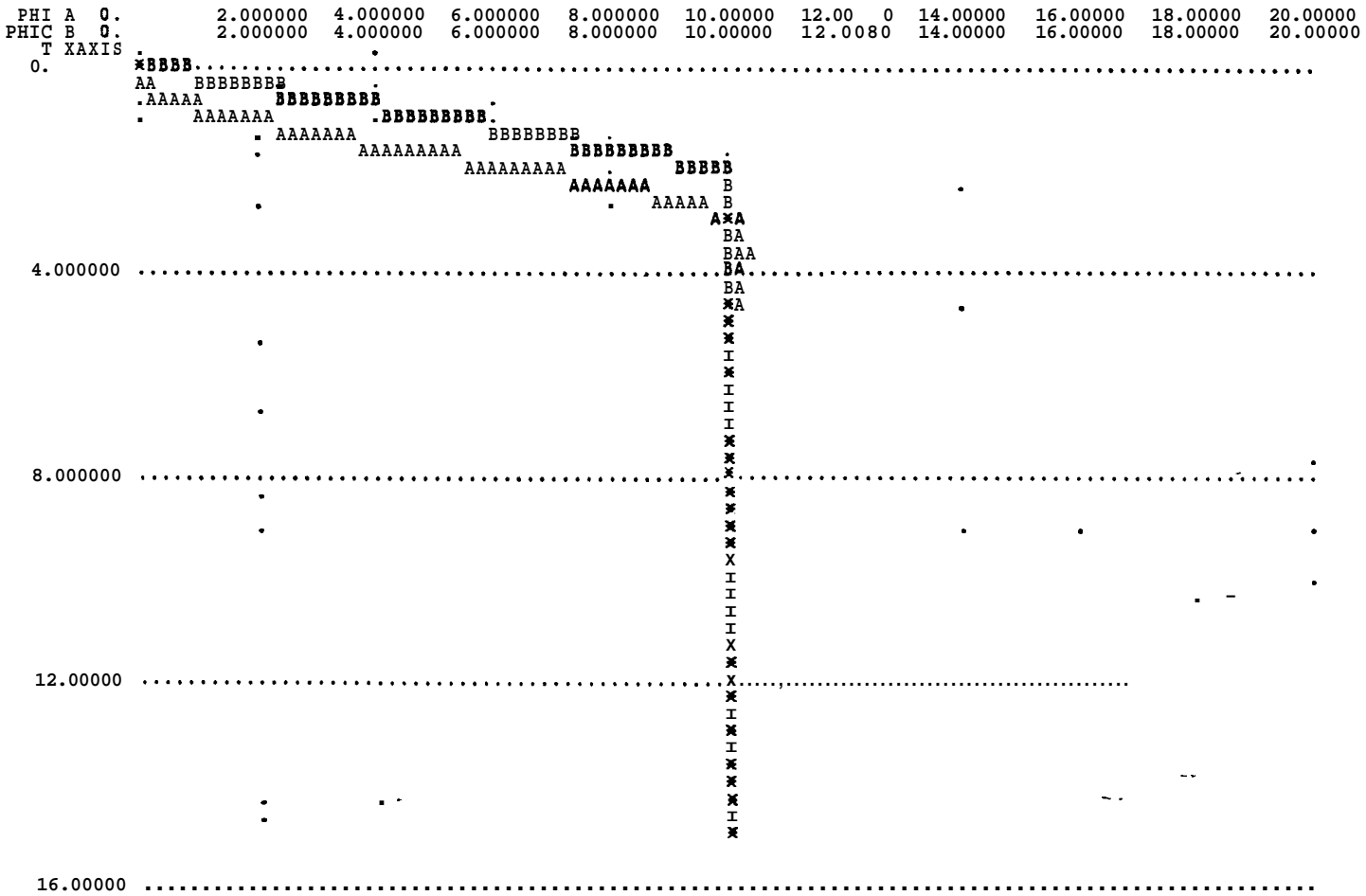


Figure 24: Bank angle (A) and bank angle command (B) (in degrees) vs time (in seconds) [reduction of interaction] -

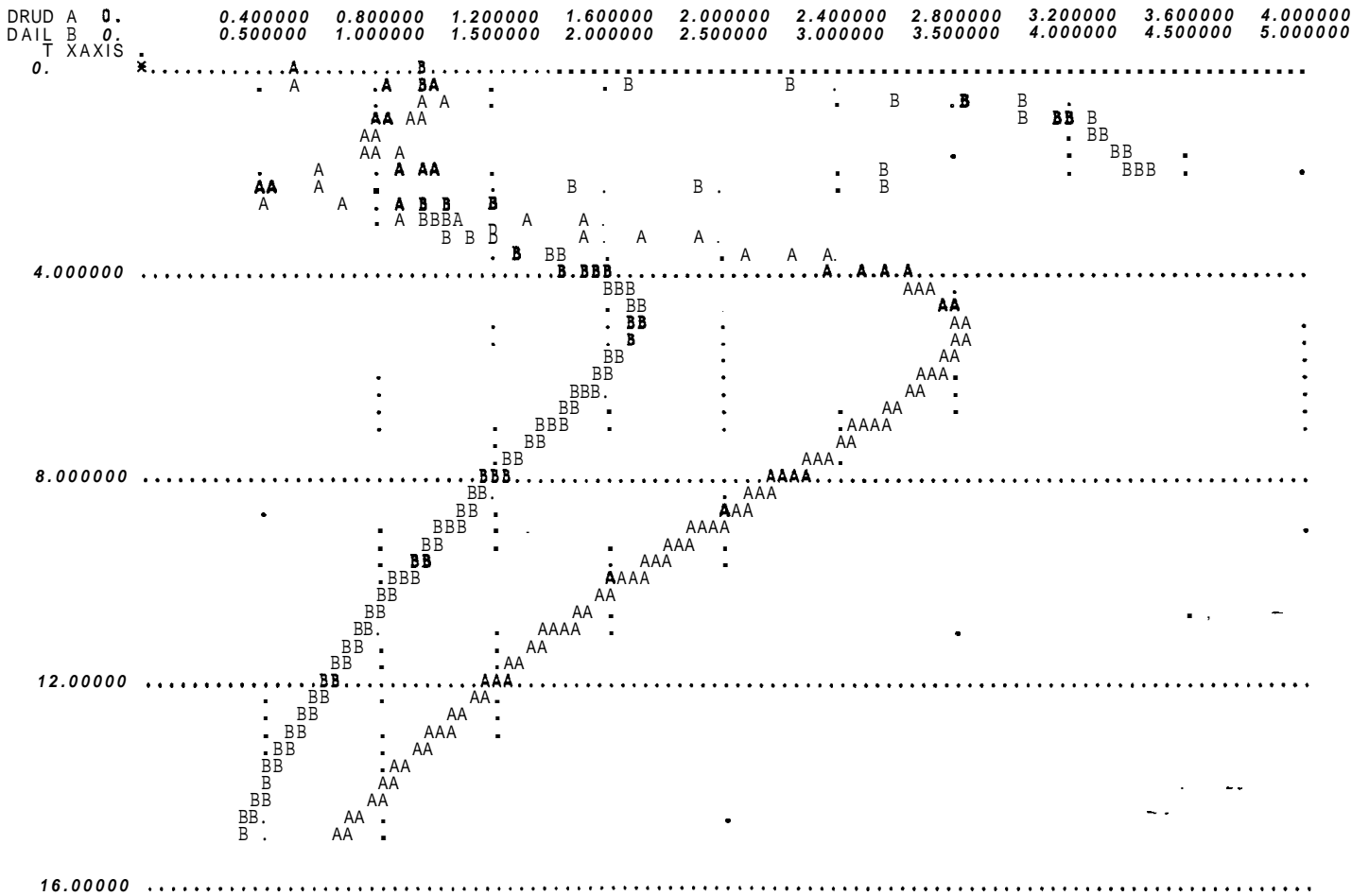


Figure 25: Drud (A and Dail (B) (in degrees) vs time (in seconds)  
[reduction of interaction]

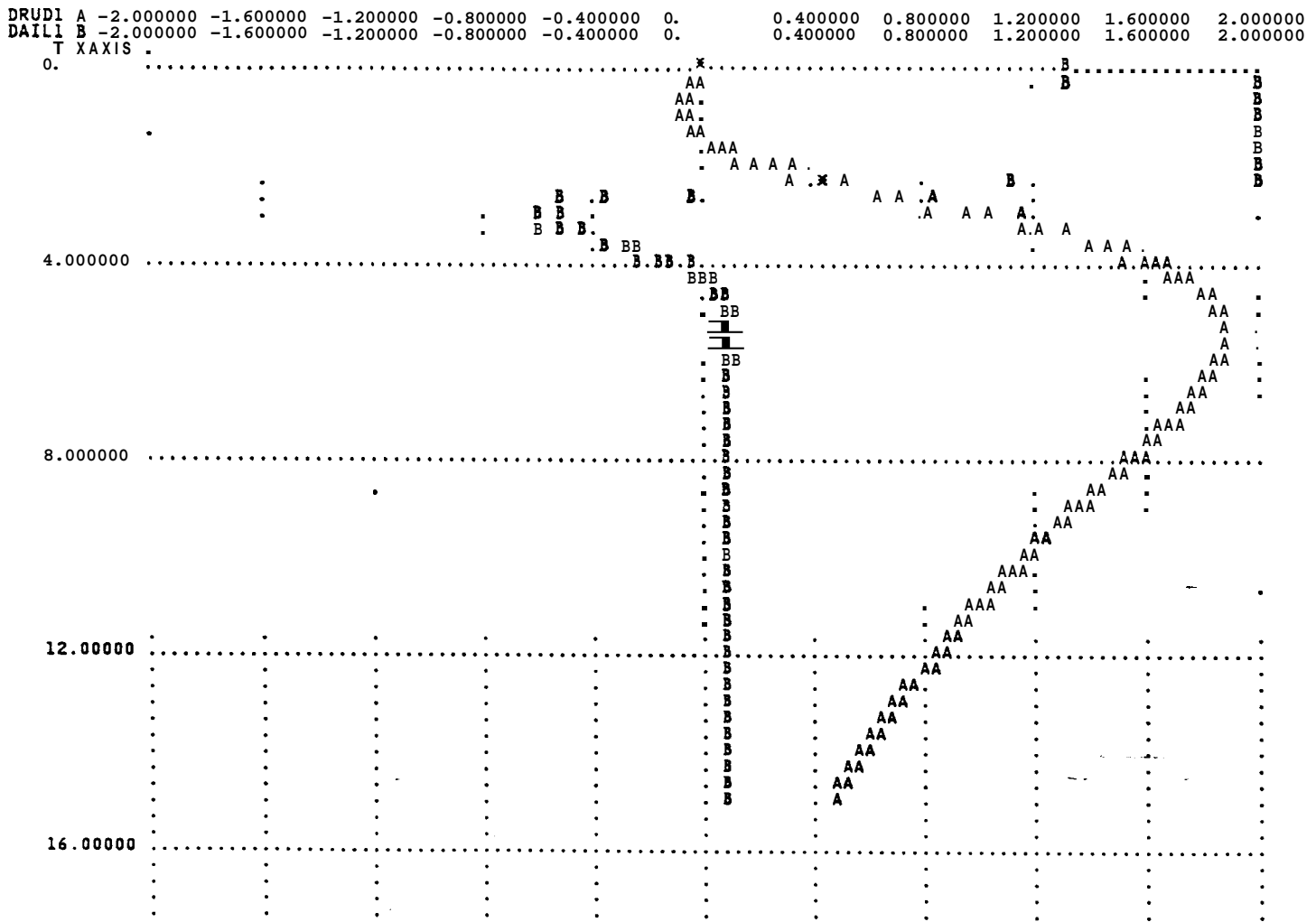


Figure 26: Drudl (A) and Daill (B) (in degrees) vs time (in seconds) [reduction of interaction]

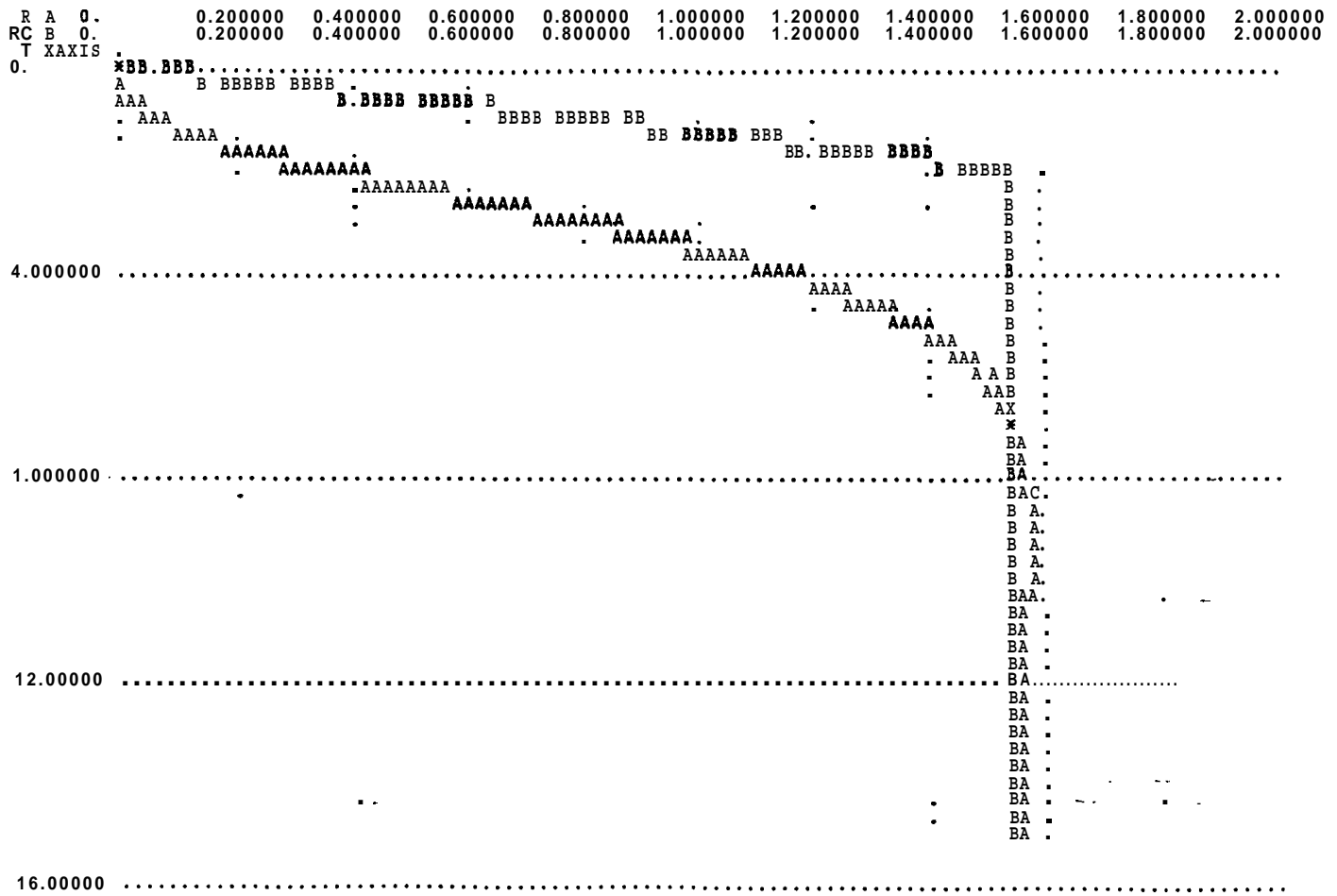


Figure 27: Yaw rate and yaw rate command (in degrees) vs time (in seconds) [reduction of interaction]

## CHAPTER IV

## CONCLUSION

In this section a brief summary of the project is discussed and some suggestions for further work on the design are given.

## BRIEF SUMMARY OF THE PROJECT

.....

In this paper we have presented a technique which is a mixture of classical and state-space methods to solve a MIMO servo problem. In chapter I we overviewed the design of servo-compensators in SISO and **multivariable** systems. In chapter **II** we developed a 3-step multivariable system design technique: (1) decompose into several interacting SISO subsystems; (2) **design** separate SISO servo systems ignoring the interactions; and (3) reduce the coupling between the subsystems. In chapter **III** an application of the technique to the design of a lateral autopilot was presented. A discrete model of the plant dynamics was developed and separated (decoupled) into two SISO subsystems in

order to use our design technique. Separate SISO designs were implemented and the interaction between them examined via simulation. Finally some reduction in interaction was obtained by following the procedure in chapter II. In conclusion, the simulation of decoupled Aileron and Rudder designs showed that the design technique was successful. Tuning was done in order to improve the decoupled responses by changing the value of  $k_d$  for each subsystem. It was shown that the changes in  $k_d$  made the responses settle down with less overshoot. The interaction reduction design between subsystems proved its effectiveness; the responses with the reduction terms present were very close to the decoupled responses.

#### SUGGESTIONS FOR FURTHER WORK

-----

In the design, it was assumed that all the states of the system were available for direct measurement. If some state variables are not available, estimates of these variables would be needed for the control law. The use of separately-designed observers for the SISO subsystems provides an area for further work.

In part A of chapter II, decomposition of a multivariable system into SISO subsystems was discussed. A designer may extend the design technique by decomposing the system into MISO subsystems. A MISO servo design technique similar to the SISO design which was used could be developed.



Another area for further work is the inclusion of adaptive control techniques in order to handle open-loop parameter changes. For example, the gains used in the interaction-reduction scheme depend directly on the open-loop parameters. It might be possible to design against parameter changes by using some form of self-tuning control or model-following control.

APPENDIX A

**COMPUTER PROGRAMS**

```

5 REM program to calculate KI,KP,KD and Ahat's
10 READ N,KV,T
20 LET NM = N-1 : LET NP = N+1
30 DIM B(NM),BB(NM),A(N),AA(NP),AH(NM)
40 FOR I=0 TO NM : READ B(I) : NEXT I
50 FOR I=0 TO N : READ A(I) : NEXT I
100, BB(0) = B(0) : AA(0) = A(0)
110 FOR I=1 TO NM : BB(I) = BB(I-1) + B(I) : NEXT I
120 FOR I=1 TO N : AA(I) = AA(I-1) + A(I) : NEXT I
130 AA(NP) = AA(N) + 1
200 REM
210 LET SA = AA(0) : LET SB = BB(0)
220 FOR I=1 TO NM : SB = SB + BB(I) : NEXT I
230 FOR I=1 TO N : SA = SA + AA(I) : NEXT I
300 REM COMPUTE GAINS
310 KI = (AA(NP)/(KV*T) + SA - AA(NP)*SB/BB(NM))/BB(NM)
320 KP = AA(NP)/BB(NM) - KI
330 KD = (KI+KP) / KV*T
340 FOR I=1 TO NM : AH(I)=KI*BB(I)+KP*BB(I-1)-KD*B(I)-AA(I):NEXT I
350 AH(0) = KI*BB(0) - KD*B(0) - AA(0)
400 REM PRINT RESULTS
410 PRINT "KI IS " ;:PRINT USING "###.###^";KI : PRINT
420 PRINT "KP IS " ;:PRINT USING "###.###^";KP :PRINT
430 PRINT "KD IS " ;:PRINT USING "###.###^";KD :PRINT
440 PRINT "AHAT IS "
450 FOR I=0 TO NM : PRINT USING "###.###^";AH(I), : NEXT I
455 PRINT
460 GOTO 1000
1000 DATA 2,1,.125
1010 DATA 1.7046e-2,7.0654e-2
1020 DATA -.602,2.13,-2.521,1

```

```

10 REM PROGRAM LSTSQR
100 REM READ DIM AND INPUT DATA
110 READ M,N,Z0
120 M1 = M+1 : MN = M + N
130 DIM G(N,M),Q(N),WA(M1,MN),GI(M,N),E(N,N),P(N,N)
140 FOR J=1 TO M
150 FOR I=1 TO N : READ G(I,J) : NEXT I
160 NEXT J
170 FOR I=1 TO N : READ Q(I) : NEXT I
180 FOR I=1 TO N
181 FOR J=1 TO N : READ P(I,J) : NEXT J
182 NEXT I
190 AS = "THIS IS THE INITIAL SET-UP"
200 REM LOAD WORK ARRAY
210 GOTO 4000
250 FOR J=1 TO M : WA(M1,J) = J : NEXT J
260 FOR J=M1 TO MN : WA(M1,J) = Q(J-M) : NEXT J
270 GOSUB 1500
300 REM WORK LOOP
310 FOR IP=1 TO M
320 REM FIND PIVOT
330 GOSUB 2000
340 REM DO PIVOTING
350 GOSUB 3000
360 REM ROW OPERATIONS
370 GOSUB 3500
380 REM INCREMENT INDEX POSTION
390 NEXT IP
400 REM PRINT RESULTS
410 LPRINT "THE WORK ARRAY IS" : LPRINT
420 FOR I=1 TO M1
430 FOR J=1 TO M
431 LPRINT " ";
432 LPRINT USING "##.###^";WA(I,J);
433 NEXT J
440 LPRINT : LPRINT
450 NEXT I
460 FOR I=1 TO M1
470 FOR J=M1 TO MN
471 LPRINT " ";
472 LPRINT USING "##.###^";WA(I,J);
473 NEXT J
480 LPRINT : LPRINT
490 NEXT I
495 LPRINT
500 REM SORT SOLUTION
510 FOR I=1 TO M
520 IX = WA(M1,I)
530 FOR J=1 TO N
531 JM = J + M
532 GI(IX,J) = WA(I,JM)
533 NEXT J
540 NEXT I

```

```

550 LPRINT "THE SORTED SOLUTION IS" : LPRINT
555 FOR I=1 TO M
560 FOR J=1 TO N
565 LPRINT " ";
570 LPRINT USING "##.###^";GI(I,J);
575 NEXT J
580 LPRINT : LPRINT
585 NEXT I
600 REM COMPUTE AND PRINT LEAST SQUARES ERROR MATRIX
605 LPRINT " THE LEAST SQUARES ERROR MATRIX IS": LPRINT
610 FOR I=1 TO N
620 FOR J=1 TO N
630 E(I,J) = 0! : IF I=J THEN E(I,J) = -1!
640 FOR K=1 TO M;E(I,J) = E(I,J) + G(I,K)*GI(K,J):NEXT K
650 LPRINT " ";
660 LPRINT USING "##.###^";E(I,J);
670 NEXT J
680 LPRINT : LPRINT
690 NEXT I
800 REM N=2 DATA
810 REM DATA 2,2,1.E-30
820 REM DATA -1.25015E-3,8.92337E-4
830 REM DATA 5.45204E-4,4.60281E-3
840 REM DATA 1.,1.
900 REM INPUT DATA
910 DATA 2,4,1.E-30
920 DATA 1.58668E-4,-1.25015E-3,5.81741E-5,8.92337E-4
930 DATA -3.22461E-5,5.45204E-4,2.97546E-4,4.60281E-3
940 DATA 1.E-7,1.,1.E-7,1.
950 REM DAT FOR PHI - PHIHAT ROW-BY-ROW
951 DATA 0.,0.,0.,9.583E-3
952 DATA 0.,0.,0.,-1.438E-2
953 DATA -2.491E-2,1.417E-2,0.,0.
954 DATA -3.882E-1,1.162E-1,0.,0.
999 GOTO 4200
1500 REM PRINT INITIAL WORK ARRAY
1510 LPRINT AS : LPRINT
1520 FOR I=1 TO M1
1530 FOR J=1 TO MN
1535 LPRINT " ";
1540 LPRINT USING "##.###^";WA(I,J);
1550 NEXT J
1560 LPRINT : LPRINT
1570 NEXT I
1580 RETURN
2000 REM ROUTINE TO FIND PIVOT
2010 MM=0 : IO=IP : JO=IP
2020 FOR I=IP TO M
2030 FOR J=IP TO M
2040 IF ABS(WA(I,J)) <= MM THEN GOTO 2110
2050 MM = ABS(WA(I,J))
2060 IO = I : JO = J
2110 NEXT J
2120 NEXT I
2200 REM CHECK MM = SIZE OF PIVOT ELEMENT
2210 IF MM >= Z0 THEN GOTO 2240
2220 PRINT : PRINT "SMALL PIVOT ENCOUNTERED AT POSITION ";IP;" WITH VALUE
2230 STOP
2240 RETURN

```

```

3000 REM ROUTINE TO PIVOT ROWS AND COLUMNS
3005 IF I0=IP THEN GOTO 3045
3010 REM INTERCHANGE ROWS
3020 FOR J=IP TO MN
3030 X = WA(IP,J) : WA(IP,J) = WA(I0,J) : WA(I0,J) = X
3040 NEXT J
3045 IF J0=IP THEN GOTO 3100
3050 REM INTERCHANGE COLUMNS
3060 FOR I=1 TO M1
3070 X = WA(I,IP) : WA(I,IP) = WA(I,J0) : WA(I,J0) = X
3080 NEXT I
3100 RETURN
3500 REM ROW OPERATIONS
3510 X = 1!/WA(IP,IP)
3520 FOR J=1 TO MN : WA(IP,J) = WA(IP,J)*X : NEXT J
3530 FOR I=1 TO M
3540 IF I=IP THEN GOTO 3590
3550 X = WA(I,IP)
3560 FOR J=IP TO MN
3570 WA(I,J) = WA(I,J) - X * WA(IP,J)
3580 NEXT J
3590 NEXT I
3600 RETURN
4000 REM COMPUTE GAM-TRANS * Q
4010 FOR K=1 TO N
4020 KM = K + M
4030 FOR I=1 TO M: WA(I,KM) = Q(K)*G(K,I) : NEXT I
4040 NEXT K
4050 REM GAM-TRANS * Q * GAM
4060 FOR I=1 TO M
4070 FOR J=1 TO M
4080 WA(I,J) = 0!
4090 FOR K=1 TO N
4100 KM = K + M
4110 WA(I,J) = WA(I,J) + WA(I,KM)*G(K,J)
4120 NEXT K
4130 NEXT J
4140 NEXT I
4150 GOTO 250
4200 REM COMPUTE THE GAINS
4210 G(3,1)=0! : G(4,1)=0! : G(1,2)=0! : G(2,2) = 0!
4220 FOR I=1 TO M
4230 FOR J=1 TO M
4240 WA(I,J) = 0!
4250 FOR K=1 TO N : WA(I,J)=WA(I,J)+GI(I,K)*G(K,J) : NEXT K
4260 NEXT J
4270 FOR J=M1 TO MN
4280 WA(I,J) = 0!
4290 FOR K=1 TO N : WA(I,J)=WA(I,J)+GI(I,K)*P(K,J-M) : NEXT K
4300 NEXT J
4310 NEXT I
4320 AS = "THE GAINS ARE" : M1 = M
4330 GOSUB 1500
6789 STOP

```

```

5 REM aileron system
10 READ N,KV,T
20 LET NM = N-1 : LET NP = N+1
30 DIM B(NM),BB(NM),A(N),AA(NP),AH(NM)
40 FOR I=0 TO NM : READ B(I) : NEXT I
50 FOR I=0 TO N : READ A(I) : NEXT I
100 BB(0) = B(0) : AA(0) = A(0)
110 FOR I=1 TO NM : BB(I) = BB(I-1) + B(I) : NEXT I
120 FOR I=1 TO N : AA(I) = AA(I-1) + A(I) : NEXT I
130 AA(NP) = AA(N) + 1
200 REM
210 LET SA = AA(0) : LET SB = BB(0)
220 FOR I=1 TO NM : SB = SB + BB(I) : NEXT I
230 FOR I=1 TO N : SA = SA + AA(I) : NEXT I
300 REM COMPUTE GAINS
310 KI = (AA(NP)/(KV*T) + SA - AA(NP)*SB/BB(NM))/BB(NM)
320 KP = AA(NP)/BB(NM) - KI
330 KD = (KI+KP) / (KV*T)
340 FOR I=1 TO NM : AH(I)=KI*BB(I)+KP*BB(I-1)-KD*B(I)-AA(I):NEXT I
350 AH(0) = KI*BB(0) - KD*B(0) - AA(0)
400 REM PRINT RESULTS
410 LPRINT "KI IS " ;:LPRINT USING "##.###^" ;KI :LPRINT
420 LPRINT "KP IS " ;:LPRINT USING "##.###^" ;KP :LPRINT
430 LPRINT "KD IS " ;:LPRINT USING "##.###^" ;KD :LPRINT
440 LPRINT "AHAT IS "
450 FOR I=0 TO NM :LPRINT USING "##.###^" ;AH(I), : NEXT I
455 LPRINT
460 GOTO 470
470 K1=1735.94+1724.18*AH(1)+1737.33*AH(0)
475 LPRINT " k1 is " ;:LPRINT USING "##.###^" ;K1 :LPRINT
480 K2=-89.54-99.89*AH(1)-112.29*AH(0)
490 LPRINT " k2 is " ;:LPRINT USING "##.###^" ;K2 :LPRINT
1000 DATA 2,1.50,.125
1010 DATA 1.7046E-2,1.5939E-2
1020 DATA -.5220,1.9312,-2.3978,1

```



```

PROGRAM LATCHS
LOGICAL $*1
* OPEN LOOP PARAMETERS *
CONSTANT A11=-1.380E-1, A12=-9.940E-1, A13= 1.551E-1, A14= 6.655E-2, ...
A21= 6.395E-1, A22=-1.399E-1, A23= 0.000E-0, A24=-1.305E-1, ...
A31= 0.000E-0, A32= 5.827E-1, A33= 0.000E-0, A34= 1.000E-0, ...
A41=-3.474E+0, A42= 8.081E-1, A43= 0.000E-0, A44=-1.476E+0, ...
CONSTANT B11= 6.177E-4, B21=-1.007E-3, B31= 0.000E-0, B41= 8.577E-3, ...
S12=-1.537E-4, S22= 4.720E-3, S32= 0.000E-0, S42= 4.004E-2, ...
CONSTANT S11= 0.000E-0, S13= 0.000E-0, S14= 0.000E-0, ...
S21= 0.000E-0, S22= 0.000E-0, S23= 0.000E-0, S24= 0.000E-0, ...
CONSTANT G11= 1.587E-4, G12= 0.000E-0, G13= 0.000E-0, G14= 0.000E-0, ...
G21= 0.000E-0, G22= 0.000E-0, G23= 2.975E-4, G24= 4.603E-3, ...
CONSTANT H11= 0.000E-0, H12= 0.000E-0, H13= 0.000E-0, H14= 9.583E-3, ...
H21= 0.000E-0, H22= 0.000E-0, H23= 0.000E-0, H24=-1.438E-3, ...
H31=-2.491E-2, H32= 1.417E-2, H33= 0.000E-0, H34= 0.000E-0, ...
H41=-3.882E-1, H42= 1.162E-1, H43= 0.000E-0, H44= 0.000E-0
* CONTROL LAW CONSTANTS
CONSTANT KDH=-1.802E-0, KIR=-2.631E-0, KPR= 2.225E-0, KR1=-9.571E+1, ...
KR2=-2.312E+2, KA1= 2.980E+2, KA2= 11.30E+2, KIA=-6.570E-0, ...
KPA= 6.792E-0, KDA=-4.482E-0, SW1= TRUE.
* SIMULATION PARAMETERS
CONSTANT TFNL=5., CU=57.3, G=32.2, V=207
CONSTANT BETA=0., RRO=0., PHIRO=0., PO=0., PHICO=0.
* COMMAND CONSTANTS
CONSTANT T0=0., T1=2., PHIDCA=5., T2=3., T3=4., PHIDCB=0.
INITIAL
CINTERVAL CINT=0.03125
WA=0. $ WR=0. $ DAIL=0. $ DRUD=0.
BETA=BETA0 $ RR=RRO $ PHIR=PHIRO $ P=PO
GOVERN=G/V
L11= S11*G11 + S12*G21 + S13*G31 + S14*G41
L12= S11*G12 + S12*G22 + S13*G32 + S14*G42
L21= S21*G11 + S22*G21 + S23*G31 + S24*G41
L22= S21*G12 + S22*G22 + S23*G32 + S24*G42
Z11= S11*H11 + S12*H21 + S13*H31 + S14*H41
Z12= S11*H12 + S12*H22 + S13*H32 + S14*H42
Z13= S11*H13 + S12*H23 + S13*H33 + S14*H43
Z14= S11*H14 + S12*H24 + S13*H34 + S14*H44
Z21= S21*H11 + S22*H21 + S23*H31 + S24*H41
Z22= S21*H12 + S22*H22 + S23*H32 + S24*H42
Z23= S21*H13 + S22*H23 + S23*H33 + S24*H43
Z24= S21*H14 + S22*H24 + S23*H34 + S24*H44
END $ "OF INITIAL"
DYNAMIC
* THE ANALOG MODEL FOLLOWS *
DERIVATIVE ANALOG
ALGORITHM IALG=5
NSTEPS NSTP=1
* LATERAL 737 DYNAMICS *
BETA = A11*BETA + A12*RR + A13*PHIR + A14*P + B11*DRUD + B12*DAIL
RR = A21*BETA + A22*RR + A23*PHIR + A24*P + B21*DRUD + B22*DAIL
PHIR = A31*BETA + A32*RR + A33*PHIR + A34*P + B31*DRUD + B32*DAIL
P = A41*BETA + A42*RR + A43*PHIR + A44*P + B41*DRUD + B42*DAIL
* COMMAND EQUATIONS
PHIDC = PHIDCA*(STEP(T0)-STEP(T1)) + PHIDCB*(STEP(T2)-STEP(T3))
PHIC = INTEG(PHIDC, PHICO)
RC = GOVERN*SIN(PHIC/CU)*CU
* OUTPUT EQUATION
PHI = PHIR * CU
R = RR * CU
EA = PHIC - PHI
ER = RC - R
END $ "OF ANALOG"
* THE DISCRETE CONTROLLER SECTION IS NEXT *
DISCRETE DIGITAL
INTERVAL HSAMP=0.125
PROCEDURAL (DRUD, DAIL = ER, BETA, RR, R, EA, PHI, PHIR, P)
DRUD1 = -KR1*BETA - KR2*RR - KDR*H + (KIR+KPR)*WR + KPR*ER
DAIL1 = -KA1*PHIR - KA2*P - KDA*PHI + (KIA+KPA)*WA + KPA*EA
DRUD2 = L11*DRUD1 + L12*DAIL1 - Z11*BETA - Z12*RR - Z13*PHIR - Z14*P
DAIL2 = L21*DRUD1 + L22*DAIL1 - Z21*BETA - Z22*RR - Z23*PHIR - Z24*P
DRUD = RSW(SW1, DRUD1, DRUD2)
DAIL = RSW(SW1, DAIL1, DAIL2)
WR = WR + ER
WA = WA + EA
END $ "OF PROCEDURAL"
END $ "OF DIGITAL"
TERMT(T.GE.TFNL)
END $ "OF DYNAMIC"
TERMINAL
END $ "OF TERMINAL"
END $ "OF PROGRAM"

```

```

5 REM rudder system
10 READ N,KV,T
20 LET NM = N-1 : LET NP = N+1
30 DIM B(NM),BB(NM),A(N),AA(NP),AH(NM)
40 FOR I=0 TO NM : READ B(I) : NEXT I
50 FOR I=0 TO N : READ A(I) : NEXT I
100 BB(0) = B(0) : AA(0) = A(0)
110 FOR I=1 TO NM : BB(I) = BB(I-1) + B(I) : NEXT I
120 FOR I=1 TO N : AA(I) = AA(I-1) + A(I) : NEXT I
130 AA(NP) = AA(N) + 1
200 REM
210 LET SA = AA(0) : LET SB = BB(0)
220 FOR I=1 TO NM : SB = SB + BB(I) : NEXT I
230 FOR I=1 TO N : SA = SA + AA(I) : NEXT I
300 REM COMPUTE GAINS
310 KI = (AA(NP)/(KV*T) + SA - AA(NP)*SB/BB(NM))/BB(NM)
320 KP = AA(NP)/BB(NM) - KI
330 KD = (KI+KP) / (KV*T)
340 FOR I=1 TO NM : AH(I)=KI*BB(I)+KP*BB(I-1)-KD*B(I)-AA(I):NEXT I
350 AH(0) = KI*BB(0) - KD*B(0) - AA(0)
400 REM PRINT RESULTS
410 PRINT "KI IS " ;:PRINT USING "###.###^ ^ ^ ^";KI : PRINT
420 PRINT "KP IS " ;:PRINT USING "###.###^ ^ ^ ^";KP : PRINT
430 PRINT "KD IS " ;:PRINT USING "###.###^ ^ ^ ^";KD : PRINT
440 PRINT "AHAT IS "
450 FOR I=0 TO NM : PRINT USING "###.###^ ^ ^ ^";AH(I), : NEXT I
455 PRINT
460 GOTO 470
470 K1=6289.92+6442.34*AH(1)+6530.83*AH(0)
475 PRINT " k1 is " ;:PRINT USING "###.###^ ^ ^ ^";K1 : PRINT
480 K2=-764.1+17.8*AH(1)+829.15*AH(0)
490 PRINT " k2 is " ;:PRINT USING "###.###^ ^ ^ ^";K2 : PRINT
1000 DATA 2,1.25,.125
1010 DATA -7.1624E-2,7.0654E-2
1020 DATA -.8216,2.6315,-2.8098,1

```

```

10 REM PROGRAM ACKERE
20 READ N,Z0,KT
30 N1 = N + 1
40 DIM FI(N,N),B(N),AH(N),A(N,N),WA(N1,N1),X(N),E(N)
50 FOR I=1 TO N
60 FOR J=1 TO N : READ FI(I,J) : NEXT J
70 NEXT I
80 FOR I=1 TO N : READ B(I) : NEXT I
90 FOR I=1 TO N : READ AH(I) : NEXT I
100 REM INPUT DATA
101 DATA 2,1.E-30,0
102 DATA 0.,1.,-0.905,1.905
103 DATA -4.38E-3,1.39E-2
104 DATA -1.1,0.3
110 DATA 3,1.E-30,1
120 DATA 9.124E-1,1.159E-1,3.761E-4
130 DATA -1.421E-1,9.307E-1,1.568E-4
140 DATA -9.112E-3,1.208E-1,1.000E+0
150 DATA -2.322E-4,-2.361E-3,-1.495E-4
160 DATA -2.267,1.712,-0.4327
180 GOSUB 4700
190 GOTO 4500
200 REM LOAD WORK ARRAY
210 FOR I=1 TO N
220 FOR J=1 TO N : WA(I,J) = A(I,J) : NEXT J
230 NEXT I
240 FOR I=1 TO N : WA(I,N1) = B(I) : NEXT I
250 FOR J=1 TO N : WA(N1,J) = J : NEXT J
260 WA(N1,N1) = 0
270 GOSUB 1500
300 REM WORK LOOP
310 FOR IP=1 TO N
320 REM FIND PIVOT
330 GOSUB 2000
340 REM DO PIVOTING
350 GOSUB 3000

```

```

360 REM ROW OPERATIONS
370 GOSUB 3500
380 REM INCREMENT INDEX POSTION
390 NEXT IP
400 REM PRINT RESULTS
110 LPRINT "THE WORK ARRAY IS" : LPRINT
420 FOR I=1 TO N
430 FOR J=1 TO N
431 LPRINT "  ";
132 LPRINT USING "##.###^~^~";WA(I,J);
433 NEXT J
440 LPRINT : LPRINT
150 NEXT I
460 FOR J=1 TO N : LPRINT USING "##.";WA(N1,J);:NEXT J
465 LPRINT
170 FOR I=1 TO N : LPRINT USING "##.###^~^~";WA(I,N1);:NEXT I
475 LPRINT
500 REM SORT SOLUTION
310 FOR I=1 TO N
320 IX = WA(N1,I)
530 X(IX) = WA(I,N1)
540 NEXT I
350 LPRINT "THE SORTED SOLUTION IS" : LPRINT
360 FOR I=1 TO N : LPRINT USING "##.###^~^~";X(I);:NEXT I
570 LPRINT
500 REM CHECK SOLUTION
510 FOR I=1 TO N
520 E(I) = B(I)
530 FOR J=1 TO N
540 E(I) = E(I) - A(I,J)*X(J)
550 NEXT J
560 NEXT I
700 REM PRINT ERROR VECTOR
705 LPRINT
710 LPRINT "THE ERROR E = B - A*X IS" : LPRINT
720 FOR I=1 TO N : LPRINT USING "##.###^~^~";E(I);:NEXT I
730 LPRINT
300 REM INPUT DATA
399 GOTO 5000
1500 REM PRINT INITIAL WORK ARRAY

```

```

1510 LPRINT "THIS IS THE INITIAL SET-UP" : LPRINT
1520 FOR I=1 TO N1
1530 FOR J=1 TO N1
1535 LPRINT " "
1540 LPRINT ~ S I "###.###^~^~";WA(I,J);
1550 NEXT J
1560 LPRINT : LPRINT
1570 NEXT I
1580 RETURN
2000 REM ROUTINE TO FIND PIVOT
2010 MM=0 : I0=IP : J0=IP
2020 FOR I=IP TO N
2030 FOR J=IP TO N
2040 IF ABS(WA(I,J)) <= MM THEN GOTO 2110
2050 MM = ABS(WA(I,J))
2060 I0 = I : J0 = J
2110 NEXT J
2120 NEXT I
2200 REM CHECK MM = SIZE OF PIVOT ELEMENT
2210 IF MM >= Z0 THEN GOTO 2240
2220 PRINT : PRINT "SMALL PIVOT ENCOUNTERED AT POSITION ";IP;" WITH VALUE
2230 STOP
2240 RETURN
3000 REM ROUTINE TO PIVOT ROWS AND COLUMNS
3005 IF I0=IP THEN GOTO 3045
3010 REM INTERCHANGE ROWS
3020 FOR J=IP TO N1
3030 X = WA(IP,J) : WA(IP,J) = WA(I0,J) : WA(I0,J) = X
3040 NEXT J
3045 IF J0=IP THEN GOTO 3100
3050 REM INTERCHANGE COLUMNS
3060 FOR I=1 TO N1
3070 X = WA(I,IP) : WA(I,IP) = WA(I,J0) : WA(I,J0) = X
3080 NEXT I
3100 RETURN
3500 REM ROW OPERATIONS
3510 X = 1!/WA(IP,IP)

```

```

3520 FOR J=1 TO N1 : WA(IP,J) = WA(IP,J)*X : NEXT J
3530 FOR I=1 TO N
3540 IF I=IP THEN GOTO 3590
3550 X = WA(I,IP)
3560 FOR J=IP TO N1
3570 WA(I,J) = WA(I,J) - X * WA(IP,J)
3580 NEXT J
3590 NEXT I
3600 RETURN
4000 REM FORM OBSERVABILITY MATRIX
4010 FOR J=1 TO N : A(1,J) = B(J) : NEXT J
4020 FOR I=2 TO N
4025 I1 = I - 1
4030 FOR J=1 TO N
4040 A(I,J) = 0!
4050 FOR K=1 TO N
4060 A(I,J) = A(I,J) + A(I1,K)*FI(K,J)
4070 NEXT K
4080 NEXT J
4090 NEXT I
4100 REM STORE EN IN B
4110 FOR I=1 TO N
4120 B(I) = 0!
4130 NEXT I
4140 B(N) = 1!
4200 REM SOLVE A*X = B
4210 GOTO 200
4500 REM TRANSPOSE PHI (FI) IF KT = 1
4510 IF KT<>1 THEN GOTO 4000
4520 FOR I=1 TO N
4530 I1 = I + 1
4540 FOR J=I1 TO N
4550 X=FI(I,J) : FI(I,J) = FI(J,I) : FI(J,I) = X
4560 NEXT J
4570 NEXT I
4580 REM RETURN TO FORM OBSERVABILITY MATRIX
4590 GOTO 4000
4700 REM ROUTINE TO PRINT INITIAL DATA
4710 LPRINT "THE INPUT DATA IS": LPRINT
4720 FOR I=1 TO N
4730 FOR J=1 TO N
4740 LPRINT "  ";
4750 LPRINT USING "##.###^ ^ ^";FI(I,J);
4760 NEXT J
4770 LPRINT : LPRINT

```

```

4780 NEXT I
4790 FOR I=1 TO N
4800 LPRINT " "; : LPRINT USING "##.###^~^~";B(I);
4810 NEXT I
4820 LPRINT : LPRINT
4830 FOR I=1 TO N
4840 LPRINT " "; : LPRINT USING "##.###^~^~";AH(I);
4850 NEXT I
4860 LPRINT : LPRINT
4870 RETURN
5000 REM MATRIX POLYNOMIAL
5010 FOR I=1 TO N
5020 FOR J=1 TO N
5030 WA(I,J) = FI(I,J)
5040 NEXT J
5050 WA(I,I) = WA(I,I) + AH(I)
5060 NEXT I
5100 FOR I=2 TO N
5110 FOR J=1 TO N
5120 REM STORE COL J OF WA IN COL N1 OF WA
5130 FOR K=1 TO N : WA(K,N1) = WA(K,J) : NEXT K
5160 REM MULTIPLY FI * COL N1 OF WA AND PUT IN COL J OF WA
5170 FOR K=1 TO N
5180 WA(K,J) = 0
5190 FOR L=1 TO N
5200 WA(K,J) = WA(K,J) + FI(K,L)*WA(L,N1)
5210 NEXT L
5220 NEXT K
5230 NEXT J
5300 REM ADD AH(I) TO DIAGONAL
5310 FOR J=1 TO N
5320 WA(J,J) = WA(J,J) + AH(I)
5330 NEXT J
5400 NEXT I
6000 REM COMPUTE GAINS AND STORE IN B
6010 FOR I=1 TO N
6020 B(I) = 0
6030 FOR J=1 TO N
6040 B(I) = B(I) + WA(I,J)*X(J)
6050 NEXT J
6060 NEXT I
6100 REM PRINT GAINS
6110 LPRINT
6120 LPRINT "THE GAINS ARE" : LPRINT
6130 FOR I=1 TO N
6140 LPRINT " "; : LPRINT USING "##.###^~^~";B(I);
6150 NEXT I
6160 LPRINT
6789 STOP

```

## REFERENCES

- [1]. John Van De Vegte. Feedback Control Systems (Englewoods cliffs, N.Y. : Simon & Schuster, 1986).
- [2]. Someshwar C. Gupta, Lawrence Hasdorff. Fundamentals of Automatic Control (New York : John Wiley & Sons, 1970).
- [3]. Karl J. Astrom, Bjorn Wittenmark. Computer Controlled Systems Theory and Design (Englewoods Cliffs, N.J. : Prentice-Hall Inc, 1984).
- [4]. Gene F. Franklin and J. David Powell. Digital Control of Dynamic Systems (New York : Addison-wesley Publishing Company Inc., 1980).
- [5]. William L. Brogan. Modern Control Theory (Englewoods Cliffs, N.J. : Prentice-Hall, Inc., 1985).
- [6]. Bernard Etkin. Dynamics of Flight - Stability and control (Canada, John Wiley & sons, Inc., 1982).