Office of Naval Research

Contract N00014-67-A-0298-0006

NR - 372 - 012

National Aeronautics and Space Administration

Grant NGR-22-007-068

TWO CONVERSATIONAL LANGUAGES FOR CONTROL-THEORETICAL COMPUTATIONS IN THE TIME-SHARING MODE

By

P. M. Newbold and A. K. Agrawala

Technical Report No. 546

November 1967

The research reported in this document was made possible through support extended the Division of Engineering and Applied Physics, Harvard University by the U. S. Army Research Office, the U. S. Air Force Office of Scientific Research and the U. S. Office of Naval Research under the Joint Services Electronics Program by Contracts N00014-67-A-0298-0006, 0005, and 0008 and by the National Aeronautics and Space Administration under Grant NGR-22-007-068.

Division of Engineering and Applied Physics

Harvard University Cambridge, Massachusetts

TWO CONVERSATIONAL LANGUAGES FOR CONTROL-THEORETICAL COMPUTATIONS IN THE TIME-SHARING MODE

By

P. M. Newbold and A. K. Agrawala

Division of Engineering and Applied Physics

Harvard University Cambridge, Massachusetts

ABSTRACT

The paper presents in the form of users' manuals, a description of two conversational languages for use on a direct-access time-sharing computer. The languages are designed for control-theoretic applications. The first language is a matrix manipulation language. The second is designed to simulate linear dynamic systems and to solve the associated Riccati equations. Further applications are also possible. Each manual includes detailed examples of the usage of the languages.

INTRODUCTION.

Since the advent of the high-level scientific programming languages, much attention has been devoted to devising languages which are to an ever-increasing degree user- and problem-oriented. If we limit our attention to languages which are useful from the control theory point of view, it is obvious from the amount of work already accomplished that what we might call the 'basic' languages, Fortran, Algol, Mad, and so on, are insufficiently specialized for many purposes. Generally speaking the more restricted the class of problems to be solved, the requirements expected of a language are increased ease of use, and increased speed of problem specification (with the assumption of course of more detailed theoretical knowledge). As a penalty the user must be prepared to accept longer times for translation into machine language, and in some cases longer execution time.

To describe the operation of a specialized language it is convenient to define the following hierarchy:

Level I: basic machine instructions

Level II: symbolic assembly : MAP, TAP, etc.

Level III: 'basic' scientific : Fortran, Algol, etc.

Level IV: specialized scientific.

Most of the processors for the specialized, or Level IV languages are written in Level III or mixed Level II / Level III languages, to which the former appear as data. This data determines sets of options on computational procedures contained in the processor which perform the tasks required by the user. Because of this structure it is possible to use the processor in a version precompiled to a Level I stage, and

run the program under the standard Operating System for the user's particular machine.

A number of languages exist for system simulation. Some of these [1, 2] are designed to present the digital computer to the user as a pseudo-analogue computer. Programming consists mainly of the tabulation of analogue computation blocks and their interconnexions. Other systems simulators consist more of Fortran-like statements [3].

Some languages are directly oriented towards the solution of differential equations, though most of these can barely merit the title of language in spite of having the processor structure outlined above.

GPSS III [4] is a language oriented towards queueing theory. Probably one of the most important languages from the control point of view is ASP [5], which is a language for the manipulation of matrices and the solution of matrix equations.

Recently the introduction of direct access time-sharing computer systems has made the usage of specialized languages more attractive because they can take on a simple question-answer or command-response form. Little or no software of this type is available at the moment: the motivation for the development of the languages described in this report was the partial filling of this software gap.

The remainder of this paper is divided into two parts, each part being in effect a user's manual for the language described. The languages are intended to be used both as research and as educational tools. The processors were written in Fortran II for the SDS 940 Time-sharing System at the Harvard University Computation Laboratory. Their object language versions operate normally under the Fortran

Operating System. The processors are to some extent machine dependent, but little labour would be involved in adapting them to other systems of a similar type.

PART I describes a language for the manipulation of matrices. Most of the usual matrix operations can be carried out, rather in the same way as one might carry out scalar operations on a desk calculator. Operations are carried out one at a time, the intermediate results being stored for later re-use.

PART II describes a language to be used for the simulation of discrete linear systems; for the solution of the Riccati equations for such systems and the calculation of feedback gains; and for the discretization of continuous linear systems. The simulation of linear stochastic systems and the solution of variance equations are also possible; by trickery even more may be feasible. The language is dignified by the name CLOSE.

The two languages have been designed to be mutually compatible in the sense that that matrices stored on disk file from one language may be used directly in the other. Any shortcomings in the languages are mainly due to lack of computer storage space which has severely limited the sophistication of the processors.

DISCLAIMER.

Copies of the processor are obtainable from the authors. Whilst every effort has been made to eliminate errors and keep documentation of modifications up to date, the authors give no guarantee that the current versions of the processors will run correctly according to the manual under every eventuality. At the same time the authors would be glad to receive notice of errors and suggestions for improvement.

REFERENCES

[1] F. J. Sansom and H. E. Petersen, "Mimic Programming Manual". SEG-TR-67-31, Systems Engineering Group, Wright-Patterson AFB (1967).

[2] A. E. Weinert and B. Hausner, "Simscript - A Simulation Language". IBM Share Library, SDA3347 (1965).

[3] W. M. Syn and D. G. Wyman, "DSL90 Digital Simulation Language". Users' Guide, IBM Share Library, SDA 3358 (1965).

[4] "General Purpose Systems Simulator III - Introduction". IBM Technical Publications (1965).

[5] R. E. Kalman and T. S. Englar, "A Users' Manual for the Automatic Synthesis Program". NASA CR-475 (1966).

PART I

A MATRIX MANIPULATION LANGUAGE

MATRIX MANIPULATOR

Version (110-2)

1. INTRODUCTION.

The Matrix Manipulator, abbreviated to MM, is a Fortran II program written for the SDS 940 system to operate under the Executor FOS. It allows various matrix operations on up to 60 matrices; the maximum size of a matrix is limited to 10 by 10. No knowledge of Fortran is required. Any number of operations can be performed in any order; up to 6 matrices can be stored on files for later use in the same or different programs.

2. DESCRIPTION.

The program is available on the Library file MAT. which can be loaded directly under FOS. The Manipulator also requires the loading of the Library file FTL. The command ;G then passes control to MM. The sequence of instructions is as follows:

```
@FOS.®
LOAD MAIN PROGRAM®
FROM FILE MAT. (LIBRARY)®
LOAD SUBPROGRAMS®
FROM FILE FTL. (LIBRARY)®
LOADING COMPLETE®
659 WORDS OF STORAGE UNUSED®
+;G®
```

The standard convention is followed that portions typed out by the computer are underlined. The symbol ® represents a carriage return. The user himself types out the portions not underlined. MM then types:

MATRIX MANIPULATOR (110-2)

The symbol < typed immediately after a carriage return, and followed by two dings on the bell indicates that MM is waiting for a command. At

present one can converse with MM using any one of about twenty different commands described in the following sections. Matrices and scalars used in the program are defined by a two-character name. The name ". . " is reserved for a special matrix where results of operations are stored. Apart from this there is no restriction on the names of the variables.

3. COMMANDS.

There are two types of commands; defining commands which set up the names of the variables and their dimensions, or manipulate the list of existing names; and executable commands which carry out matrix operations or handle input and output. Only the first three letters of a command need be typed out, the others being optional. In the description given below these three letters are capitalized.

3.1. Defining Commands.

The general format for a defining command, with one exception, is

< {command}®

There are four different commands:

(i) < MATrices ®

This command is used to name and define the sizes of a list of matrices of any length up to the capacity of MM. After the carriage return MM signals by a double bell that it is waiting for the list to be typed in. Typing ALL instead of a list entry terminates the command and the list. The general format for a list entry is

{s} = m, n,®

where {s} is any valid two-character name and m, n are the row and column dimensions of the matrix written as integers. Special care must be taken not to omit the final comma of each list entry.

Example:

```
< MATRICES ®
AA=2, 2, ®
C  =2, 3, ®
* D=1, 6, ®
ALL ®
<
```

(ii) < SCAlars ®

This command operates in the same way as the previous one, and is used to define scalars. The general format of a list entry is:

{s} ,®

Note again the terminal comma. MM treats scalars as 1 by 1 matrices, and the user himself can treat them so by alternatively defining them as such in the list of matrices.

Example:

```
< SCALARS ®
DF, ®
G&, ®
ALL®
<
```

If duplication of variable names occurs, both will be given space in the list of variables, but only the first occurrence in the list will be effective for executable commands.

On first entry into MM, after typing the heading the two above commands are issued successively by MM, so avoiding any forgetfulness on the part of the user. Thereafter at any stage in execution the user may add to his list of variables. When these two commands have been completed at first entry, MM issues the invitation

```
START !!!!®
<
```

(iii) < LISt ®

If the user wishes to check the lists of variables he has defined, use

of this command will result in MM printing out the complete lists of all matrices and scalars.

The last command of this type has a slightly different format.

(iv) $\leq$ {s}, OMIt ®

This command is used to delete the variable name {s} from the list in which it appears. If the command is used to delete a duplicate of a name from a list, then it results in the first occurrence of that name being deleted from the list. The matrix ". . " cannot be deleted.

3.2. Executable Commands.

The general format of the executable commands (except for the STOP command) is as follows:

$\leq$ {s}, {command} ® or

$\leq$ {s}, {t}, {command} ® where {s}, {t} are valid variable names.

The STOP command signals that no more operations are to be carried out. The format is:

(v) $\leq$ STOp ®

The words

STOP ®
+

are printed out and control is relinquished to FOS.

The remainder of the commands can be divided into two groups; those which use ". . " as the standard output variable; and other commands. The sizes of ". . " take on the required values dictated by the particular matrix operation.

3.2.1. Miscellaneous Commands.

(vi) $\leq$ {s}, REAd ®

This command is used to read in the values of a matrix from the typewriter

console. MM waits for the values to be typed in after giving a double bell. Input format is standard and is described in the following section.

Example:

```
< AA, READ ®          ( AA is a 2 by 2 matrix)
2.1, 3.5, ®
7.5, 0.3, ®
<
```

(vii) < {s}, PRInt ®

On receipt of this command MM prints out the values of the specified matrix in the standard format.

(viii) < {s}, STOre®

This command is used to store the values of matrix {s} on disk file. The exact sequence is:

```
< {s}, STOre®          where {s} is a variable name
FILE NO = n, ®         and n is a file number between
<                      1 and 6. See Section 6.
```

Only one matrix can be stored on each file.

(ix) < {s}, LOAd ®

This command is used to read in the values for matrix {s} from a specified file. The exact sequence is:

```
< {s}, LOAd ®
FILE NO = n, ®         See Section 6.
<
```

(x) < {s}, {t}, EQUate ®

This command is used to copy the values of {s} into {t} . Except where the second matrix is ". . " the two matrices must be of the same size.

(xi) < {s}, NULl ®

This command computes {s} equal to zero.

(xii) < {s}, EIGenvalue ®

This command results in the computation of eigenvalues and eigenvectors for matrix {s}. Their numerical values are printed out as calculated,

but are not stored. However ".." is used and contains junk at the end of the calculation. If MM cannot find an eigenvalue the message

COMPUTATION FAILURE ®
<

is printed out and the command terminated.

3.2.2. Commands having ".." as Output Matrix.

(xiii) < {s}, {t}, ADD ® : computes .. = {s} + {t}

(xiv) < {s}, {t}, SUBtract ® : computes .. = {s} - {t}

(xv) < {s}, TRAnspose ® : computes .. = $\{s\}^T$

(xvi) < {s}, NEGate ® : computes .. = -{s}

(xvii) < {s}, {t}, MULtiply ® : computes .. = {s} · {t}

(xviii) < {s}, INVert ® : computes .. = $\{s\}^{-1}$

(xix) < {s}, {t}, SCAlar multiply ® : computes .. = {s} · {t} where {t} is a scalar.

(xx) < {s}, DIAgonal sum ® : computes .. = Trace {s}

(xxi) < {s}, DETerminant ® : computes .. = Det {s}

Two points are especially to be noted.

(a) If the user attempts to invert a singular matrix, ".." will be set to zero, and the following message will appear:

MATRIX SINGULAR, RANK = # ®
<

(b) If the user attempts to find the determinant of a singular matrix, ".." is set to zero, and the following message will appear:

DETERMINANT ZERO ®
<

A complete directory of commands is given in Section 7.

4. FORMATS.

MM reads and writes all matrices and scalars in a fixed standard format, both on the typewriter console and on file. While typing in

values under the READ command, the user should type in the matrix row by row. Due to some unfortunate Fortran restrictions, no more than 4 entries are permitted on any line. Hence when larger matrices than 4 by 4 are used, each row of the matrix is typed in order, 4 entries per line up to three lines where necessary. Each row must start on a new line; as an aid, a double bell rings when each row of the matrix is completed. Each individual entry must take up fifteen spaces or less and terminate in a comma.

An example showing the input of values to a 2 by 7 matrix is given immediately below:

Example:

```
< AA, READ ®          AA is a 2 by 7 matrix.
1.0, 2.0, 3.0, 4.0, ®
5.0, 6.0, 7.0, ®      End of row 1
7.0, 6.0, 5.0, 4.0, ®
3.0, 2.0, 1.0, ®      End of row 2
<
```

Although it is unnecessary to know the specific output formats, a complete listing is provided for interested users in Appendix I. Standard Fortran notation is employed, both in typing in the numbers, and in the Appendix.

5. ERROR MESSAGES.

If the user commits an error MM will either drop out entirely due to a corresponding Fortran run-time error in the MM executor; more hopefully an error message will be put out by MM.

(i)
```
WHAT ? ®
<
```

This message is given if MM cannot recognize the command being input. The command should be repeated correctly. If the error message is given during input of a variable list, the current entry only is ignored, not the whole list.

(ii) TOO BIG ®
<

This message is given if the matrix just defined in the list is oversize.

(iii) TOO MANY ®
<

This message is given either when more than 60 variables have been defined, or when the total storage required by the defined variables exceeds that available (1500 locations).

(iv) VARIABLE {s} NOT SET ®
<

This message is given when MM cannot find the variable {s} just used in an executable command in its list of defined variables.

(v) MATRICES INCOMPATIBLE ®
<

The compatibility of the matrices used in the executable commands is checked out before the command is executed. If they are found to be incompatible, execution of the command is stopped and the above message is given.

6. FILE HANDLING.

The commands LOAD and STORE require the creation of files before they can be used. Only files actually used in MM operation need be defined. If LOAD and STORE are never used this section may be ignored. The files required must be created separately for each user; they remain separate for each user, even though the users may take turns using MM under their own user numbers. The files need only be created once (unless of course by some mischance they are erased).

The file names MUST take the following form:

/FILEn/

where the integer n corresponds to the integer file number in the LOAD or STORE command.

The easiest way to create the files is to type the following before entering FOS:

@COPY TEL. TO /FILEn/ NEW FILE.@ †
@

If uncreated files are requested, MM will drop out.

7. DIRECTORY OF COMMANDS

Defining commands:

MATRICES

SCALARS

LIST

OMIT

Executable commands:

STOP

(i) Miscellaneous

READ

PRINT

LOAD

STORE

EQUATE

NULL

EIGENVALUE

(ii) Commands with .. as Output

ADD

SUBTRACT

TRANSPOSE

---

† The @ not underlined implies that the user should hit the escape key once, not that he should hit the symbol "@".

(ii) Commands with . . as Output (con't.)

NEGATE

MULTIPLY

INVERT

SCALAR MULTIPLY

DIAGONAL SUM

DETERMINANT

APPENDIX I

FORMATS

The program reads and writes all the matrices in fixed standard format. This format depends on the number of columns the matrix has. The following formats are used.

No. of Columns	Format
1	(E16.8/)
2	(2E16.8/)
3	(3E16.8/)
4	(4E16.8/)
5	(4E16.8/E20.8/)
6	(4E16.8/E20.8, E16.8/)
7	(4E16.8/E20.8, 2E16.8/)
8	(4E16.8/E20.8, 3E16.8/)
9	(4E16.8/E20.8, 3E16.8/E24.8/)
10	(4E16.8/E20.8, 3E16.8/E24.8, E16.8/)

Standard Fortran notation is used in the above list.

For example the format E16.8 means that the number at input and output appears with an explicit exponent, and 8 digits to the right of the decimal point, while the complete number is right-justified in a field of 16 spaces.

As is obvious from previous examples on input several relaxations are usually employed:

(i) Early termination of a field by a comma.

(ii) Omission of the exponent.

(iii) Shift of the decimal point.

APPENDIX II

EXAMPLES OF USAGE.

```
@FOS.
LOAD MAIN PROGRAM
FROM FILE MAT. (LIBRARY)
LOAD SUBPROGRAMS
FROM FILE FTL. (LIBRARY)
LOADING COMPLETE
470 WORDS OF STORAGE UNUSED
#;G
MATRIX MANIPULATOR  (110-1)

< MATRICES
AA=2,2,
AB=2,2,
AC=4,4                    .............
TOO BIG
AC=4,4,
ALL
< SCALARS
 D,
 E,
ALL
START !!!!
< AA,READ
1.,2.,
3.,4.,
<AA,AB,EQUATE
<AB,PRINT
    .10000000E  1    .20000000E  1
    .30000000E  1    .40000000E  1
< AA,AB,MULTIPLY
< .., PRINT
    .70000000E  1    .10000000E  2
    .15000000E  2    .22000000E  2
< LIST
NAME          SIZE
 ..          2   BY    2
 AA          2   BY    2
 AB          2   BY    2
 AC          4   BY    4
  D          1   BY    1
  E          1   BY    1
< BB,TRANSPOSE                .........
VARIABLE  BB  NOT SET
< MATRICES
BB=1,4,
ALL
< READ                    ..................
WHAT?
< BB,REED                 ..................
WHAT?
< BB,READ
```

Loading of MM under FOS

Terminal comma omitted

BB not yet defined in a variable list.

Unintelligible command.

READ misspelled.

```
1.,2.,3.,4.,
<BB,TRANSPOSE
<..,PRINT
    .10000000E  1
    .20000000E  1
    .30000000E  1
    .40000000E  1
< AC,OMIT
< LIST
NAME          SIZE
 ..          4   BY    1
 AA          2   BY    2
 AB          2   BY    2
  D          1   BY    1
  E          1   BY    1
 BB          1   BY    4
< AA,INVERT
< ..,PRINT
   -.20000000E  1    .10000000E  1
    .15000000E  1   -.50000000E  0
< AA,DETERMINANT
< ..,PRINT
   -.20000000E  1
< AA,EIGENVALUE
EIGENVALUE  1             .53722813E  1
            .45742711E  0

            .10000000E  1
EIGENVALUE  2            -.37228138E  0
            .76111655E  0

           -.52223300E  0
< AB,NULL
< AB,PRINT
    .00000000E  0    .00000000E  0
    .00000000E  0    .00000000E  0
< AA,STORE
FILE NO =2,
< AA,NULL
< AB,LOAD
FILE NO =2,
< AA,PRINT
    .00000000E  0    .00000000E  0
    .00000000E  0    .00000000E  0
< AB,PRINT
    .10000000E  1    .20000000E  1
    .30000000E  1    .40000000E  1
< AB,INVERT
< ..,AB,MULTIPLY
< ..,PRINT
    .10000000E  1    .00000000E  0
    .00000000E  0    .10000000E  1
< AB,BB,ADD
MATRICES INCOMPATIBLE      .....
< STOP

*STOP*

+
```

AC has been deleted from the list of matrices.

Value of AA has been stored on file and retrieved again.

Operation attempted on incompatible matrices.

PART II

"CLOSE"

CONVERSATIONAL LINEAR OPTIMAL SYSTEM EXPERIMENTER

A CONTROL SYSTEM SIMULATOR

1. INTRODUCTION.

CLOSE is a Fortran II program written for SDS 940 time sharing system to operate under FOS. At present it deals with linear time invariant systems and can handle the following problems:

1. Discretization - Given a continuous linear system and a step size it can compute the coefficient matrices for the equivalent discrete system.

2. Riccati Equation Solution - For the discrete linear system and a quadratic cost function, it can compute the solution of the discrete version of the corresponding Riccati equation.

3. Simulation - A linear discrete feedback system can be simulated with or without Gaussian white noise.

4. Filtering Problem - The optimal gains for a linear system with Gauss Markov noise sequence can be computed.

The program operates in a conversational mode, i.e. the execution of the program proceeds in a question-answer form. No knowledge of Fortran is necessary on the part of the user to run this program to solve his problem even though the program operates under FOS. Presently the program can handle problems with at most 10 state, control and noise variables.

In this manual the standard convention is used that the underlined parts of question-answer sequences are those typed by CLOSE, the rest by the user. Ⓡ is used to indicate the carriage return that the user must give.

2. COMPUTATIONS.

The problems accepted by CLOSE perform the computations described in this section.

A. Discretization:

Consider a linear system described by the Eq. 1. For a specified time step Δ an equivalent discrete linear system is given by Eq. 2.

$$\dot{x} = Fx + Gu \tag{1}$$

$$x\{(k+1)\Delta\} = \Phi x(k\Delta) + Du(k\Delta) \tag{2}$$

When F and G are constant coefficient matrices, Φ and D are also constant and have the analytic form given by Eq. 3 and 4.

$$\Phi = e^{F\Delta} \tag{3}$$

$$D = \int_0^{\Delta} e^{Ft} dt \cdot G \tag{4}$$

This part of the program computes Φ and D given F, G and Δ. The series expansion for the right-hand side of Eq. 3 and 4 is used, summing the first 35 terms of the series. If the trace of the coefficient matrices changes by less than 10^{-5}, the series is terminated earlier.

The method used for this part is very similar to the method used in ASP [5], for exponential routines.

B. Riccati Equation Solution:

Consider a linear system described by the difference Eq. 5. If we want to minimize a cost function J given by Eq. 6

$$x(i+1) = \Phi x(i) + Du(i) \tag{5}$$

$$J = x^T(N)S(N)x(N) + \sum_{i=0}^{N-1} x^T(i)Ax(i) + u^T(i)Bu(i) \tag{6}$$

the optimal control is given by

$$u_{opt}(i) = -\{D^T S(i+1)D + B\}^{-1} D^T S(i+1)\Phi x(i)$$
$$= K(i)x(i) \qquad (7)$$

where S(i) is the solution of the equivalent Riccati equation for this problem defined by the Eqs. 8, 9 and 10.

$$R(i+1) = S(i+1)D\{D^T S(i+1)D + B\}^{-1} \qquad (8)$$

$$P(i+1) = \{I - R(i+1)D^T\}S(i+1)\{I - R(i+1)D^T\}^T$$
$$+ R(i+1)BR^T(i+1) \qquad (9)$$

$$S(i) = \Phi^T P(i+1)\Phi + A \qquad (10)$$

This part of the program computes the values of S(i), starting backwards from S(N), and the feedback gain matrix K(i).

The feedback gains computed this way are the same for the discrete system and the equivalent continuous system but the values of S are out by the factor of the step size.

C. Simulation:

For a linear discrete system with Gaussian white noise the system equation may be written as

$$x(i+1) = \Phi x(i) + Du(i) + \Gamma w(i) \qquad (11)$$

This part of the program simulates this equation in a feedback environment, i.e.

$$u(i) = K x(i) \qquad (12)$$

where K may be the asymptotic value of K(i), or any other constant feedback gain matrix.

w is a Gaussian white noise sequence with mean α and a diagonal covariance matrix which is treated as a vector β by this program.

The names used by CLOSE for the matrices defined in this section are given in Table 1.

The only computations performed by CLOSE are those described in this section so far. The user may, however, use these computations for some other problems by discovering the correspondences. One such example is the Filtering Problem.

D. Filtering Problem:

Consider the system described by Eq. 13 and 14, where w(k) is a white noise sequence $N(\bar{w}, Q)$, z is a measurement vector, and v is a second white noise sequence $N(O, R)$:

$$x(i+1) = \Phi x(i) + \Gamma w(i) \tag{13}$$

$$z(i) = Hx(i) + v(i) \tag{14}$$

For the best estimate of x we have

$$\hat{x}(i) = \bar{x}(i) + P(i)H^T R(i)^{-1}\{z(i) - H\bar{x}(i)\} \tag{15}$$

$$\bar{x}(i+1) = \Phi\hat{x}(i) + \Gamma\bar{w}(i) \tag{16}$$

$$P(i) = M(i) - M(i)H^T\{HM(i)H^T + R\}^{-1}HM(i) \tag{17}$$

$$M(i+1) = \Phi P(i)\Phi^T + \Gamma Q\Gamma^T \tag{18}$$

A correspondence exists between equations 17, 18 and equations 8, 9, 10. Realizing this the user may use the computations done in Part B for computing the variance equation results for this problem.

3. PROGRAM DETAILS.

This program is to be loaded under FOS from file /CLOSE/. The details of the loading procedure are described in Appendix I. The program operates under FOS monitor and its execution for any problem may be divided into 4 phases. Figure 1 shows a very basic block diagram for the program with the phases of its execution.

Matrix Name	Name Used by CLOSE	File Name
Φ	PHI	/PHI/
D	D	/D/
F	F	/F/
G	G	/G/
x	X	/X/
K	K	/K/
Γ	GAM	/GAM/
α	AV	/AV/
β	VAR	/VAR/
S	S	/S/
A	A	/A/
B	B	/B/

Table 1. List of Matrix and File Names Used by CLOSE

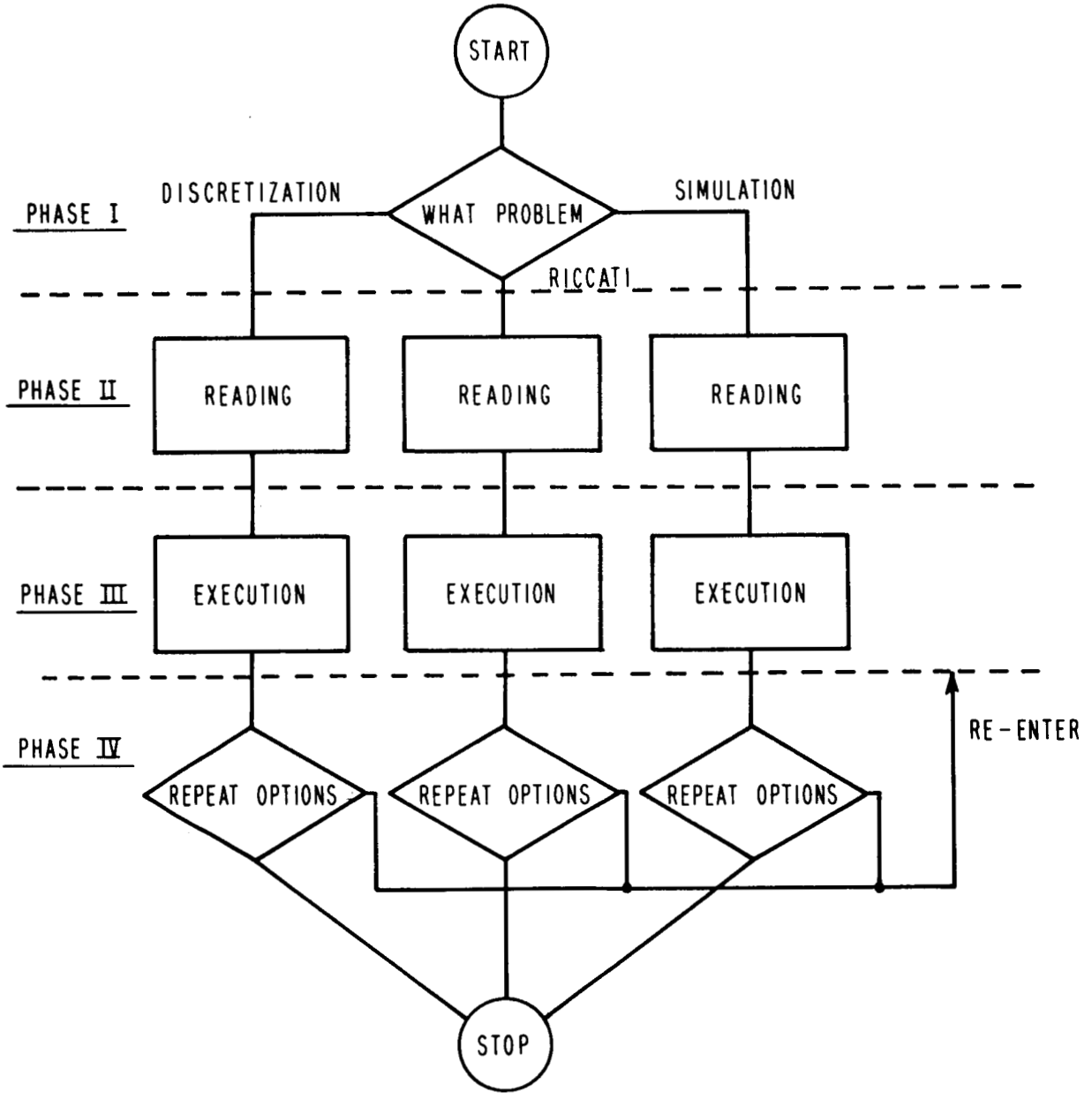

FIG. 1 FLOW DIAGRAM FOR CLOSE

3.1. PHASE I. Problem Selection Phase.

This is the first phase of the program, which is automatically entered at the beginning of a run. The information that CLOSE wants in this phase is the name of the problem the user intends to solve. The user identifies the problem by typing the name of the problem, i.e. DISCRETIZATION, RICCATI EQUATION or SIMULATION as the answer to the question WHAT PROBLEM ? which CLOSE asks first. An example follows:

WHAT PROBLEM ?
RICCATI,@

The sequence above is a typical example of the question-answer structure of the program. Generally, if CLOSE does not recognize an answer given by the user it gives an error message and repeats the question (for details of error messages see Section 4). Note that CLOSE uses only the first 3 characters of the name to identify the problem. Therefore the user need not type any more characters. However, THE USER MUST TERMINATE HIS ANSWERS BY A COMMA and this applies to all the answers he gives.

After receiving and identifying the name of the problem as acceptable, CLOSE selects the appropriate branch to take for the subsequent phases and proceeds to the next phase.

3.2. PHASE II. Data Acceptance Phase.

Having identified the problem, in this phase CLOSE demands all the relevant input data. The first information asked for is the dimensions of the system vectors. CLOSE initiates the following sequence:

DIMENSIONS:
STATE = 2,®
CONTROL = 2,®
NOISE = 1,®

On wanting the numerical values of dimension the computer waits after giving a double bell on the console, whereupon the user types the number in, not forgetting to follow it by a comma and a carriage return. Note that only the required vectors for the particular problem appear in the sequence.

Next CLOSE asks for the values for the relevant matrices for the problem. For each matrix in turn CLOSE initiates the following sequence,

READ {a} MATRIX ?
YES,®
TYPE ?
YES,®
GO ON !
1.0, 2.0,®
2.134, 4.606,®
DONE !

Here CLOSE first asks if the user wishes to input values for the matrix {a}, where {a} is one of the matrix names given in Section 2. A "NO" answer is only valid if CLOSE already has values in its working storage. Otherwise the user must, at this point, decide whether he wants to type the values in from the console or read them from a reserved file. If he answers "YES" to "TYPE ?", CLOSE returns "GO ON" and expects the values to be typed in; otherwise values are taken off the file. In either case, on completion of reading CLOSE types "DONE" and a new sequence follows.

Comment 1. Whenever the user types the values of the elements of any matrix from the console, they are also stored in on the reserved file so that the user does not have to type the same numbers again.

Comment 2. When the user wants to read the values for S in RICCATI, or K and X in SIMULATION from the file he may be interested in using other than the first stored value. CLOSE provides this option by asking the following question,

```
USE N-TH STORED VALUE OF K FROM FILE
WHERE N= 5,®
```

Comment 3. When CLOSE asks for the values for S in RICCATI it is asking for the value at the final time, i.e. S(N) because it solves the equation backwards. The value of X asked in SIMULATION is the starting value.

For the details on typing format for the matrix values see Section 5.

3.3. PHASE III. Execution Phase.

In this phase CLOSE carries out the main body of the computation associated with the user's problem. In this phase too, the data resulting from the solution is output. Each of the problems has its own peculiarities so that the question-answer sequences do not follow each other in the same order.

For the RICCATI and SIMULATION problems only, CLOSE requires information on time indexing. For these two problems only, then, the following sequence is initiated,

```
SET INDEXING ?
YES,®
STARTING INDEX =10,®
NUMBER OF STEPS =10,®
NUMBER OF STEPS BETWEEN OUTPUTS =5,®
```

In control problems it is usual to solve the Riccati Equation backwards in time, so in this case the number of steps has a negative sign.

The next sequence initiated by CLOSE is the same for each problem,

STORE VALUES ?
YES,@
EXECUTE ?
YES,@

A "YES" answer to the first question of the sequence results in the storage on their appropriate file, of the values that are printed out as output during the execution. Note that the files for S, K and X will have many matrices stored on them. Otherwise no values are stored; they are only printed out on the console. A "NO" answer to the next question transfers CLOSE to a "REREAD" point in Phase IV described in 3.4; otherwise the actual computations follow. For the formats of the output see the example in Appendix II.

Additional information about the step size Δ is required for DISCRETIZATION. CLOSE asks for this before starting the computations in this case and then proceeds with the computations.

3.4. PHASE IV. Repeat Options.

Remaining within the control of CLOSE this phase allows the user to go back in any other phase at different levels. The following sequence is initiated by CLOSE in this phase for RICCATI and SIMULATION, being relevant only to these problems,

RE-EXECUTE ?
NO,@
RE-INDEX ?
NO,@

A "YES" answer to the first question takes CLOSE back to Phase III after EXECUTION ? and a new set of computations takes place. Note that this re-execution starts with the last computed value of S or X

(depending on RICCATI or SIMULATION being the problem) as the new starting value, indexing being the same (see example in Appendix II). When the user answers "YES" for "RE-INDEX", CLOSE goes back to the beginning of Phase III. Note that still the last value of S or X is used as the new starting value. A "NO" answer makes CLOSE initiate the next sequence.

The sequence described next follows Phase III for the DISCRETIZATION problem. It also follows a "NO" answer to "EXECUTE ?" in Phase III.

RE-READ ANY MATRICES ?
NO,®
START AGAIN ?
NO,®

A "YES" answer to the first question takes CLOSE back to the reading matrices level of Phase II. A "YES" answer to the next question takes CLOSE back to the beginning of Phase II and CLOSE asks for DIMENSIONS again; otherwise the next sequence follows,

ANY OTHER PROBLEM ?
NO,®
STOP
±

A "YES" answer to the question above takes CLOSE back to the beginning of Phase I and now the user has the option of selecting a different problem. A "NO" answer completes the run of CLOSE, the control is transferred back to FOS.

The user may find the flow diagrams in Figs. 2 and 3 of use in describing the flow through the program for the individual problems.

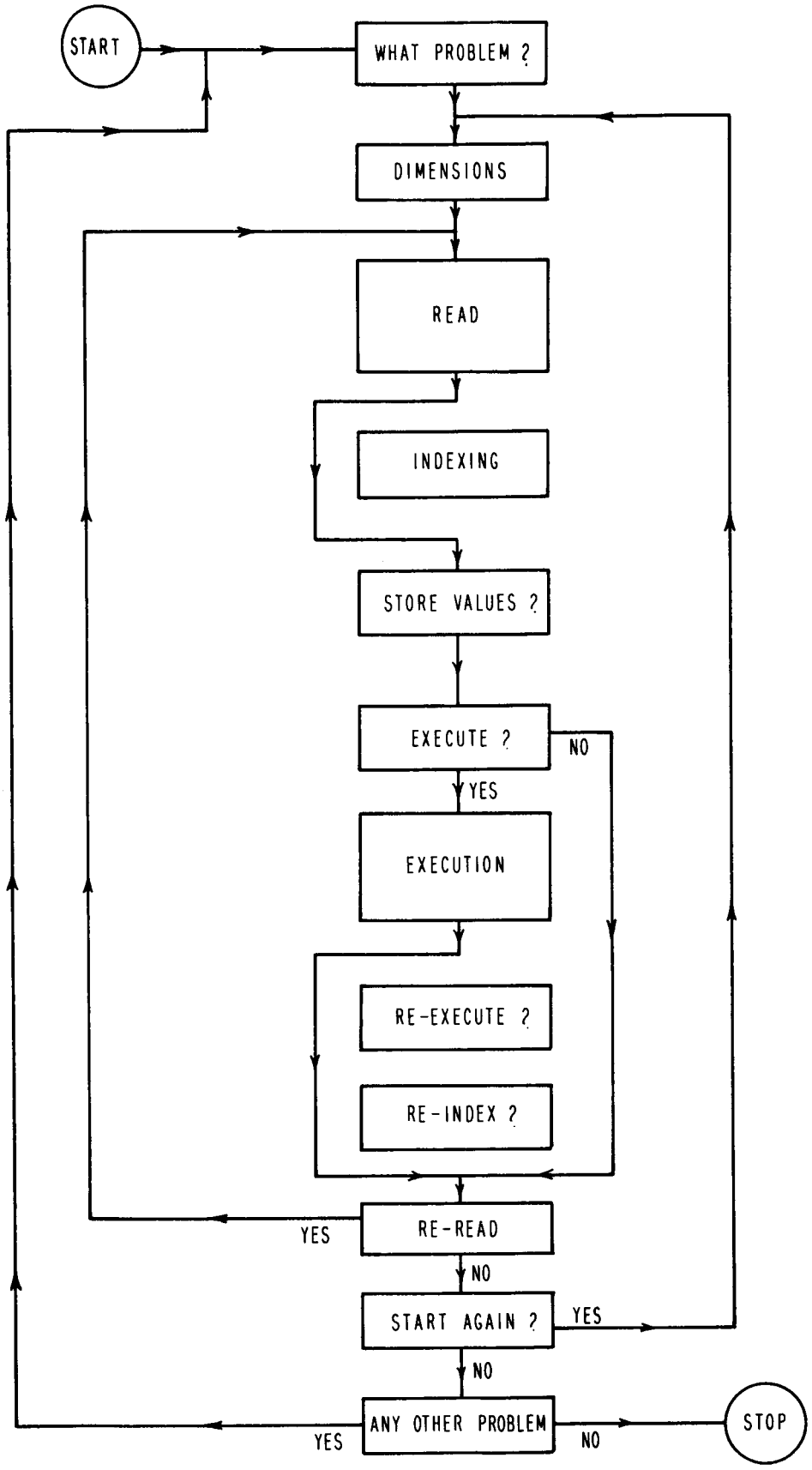

FIG. 2. FLOW DIAGRAM FOR DISCRETIZATION PROBLEM

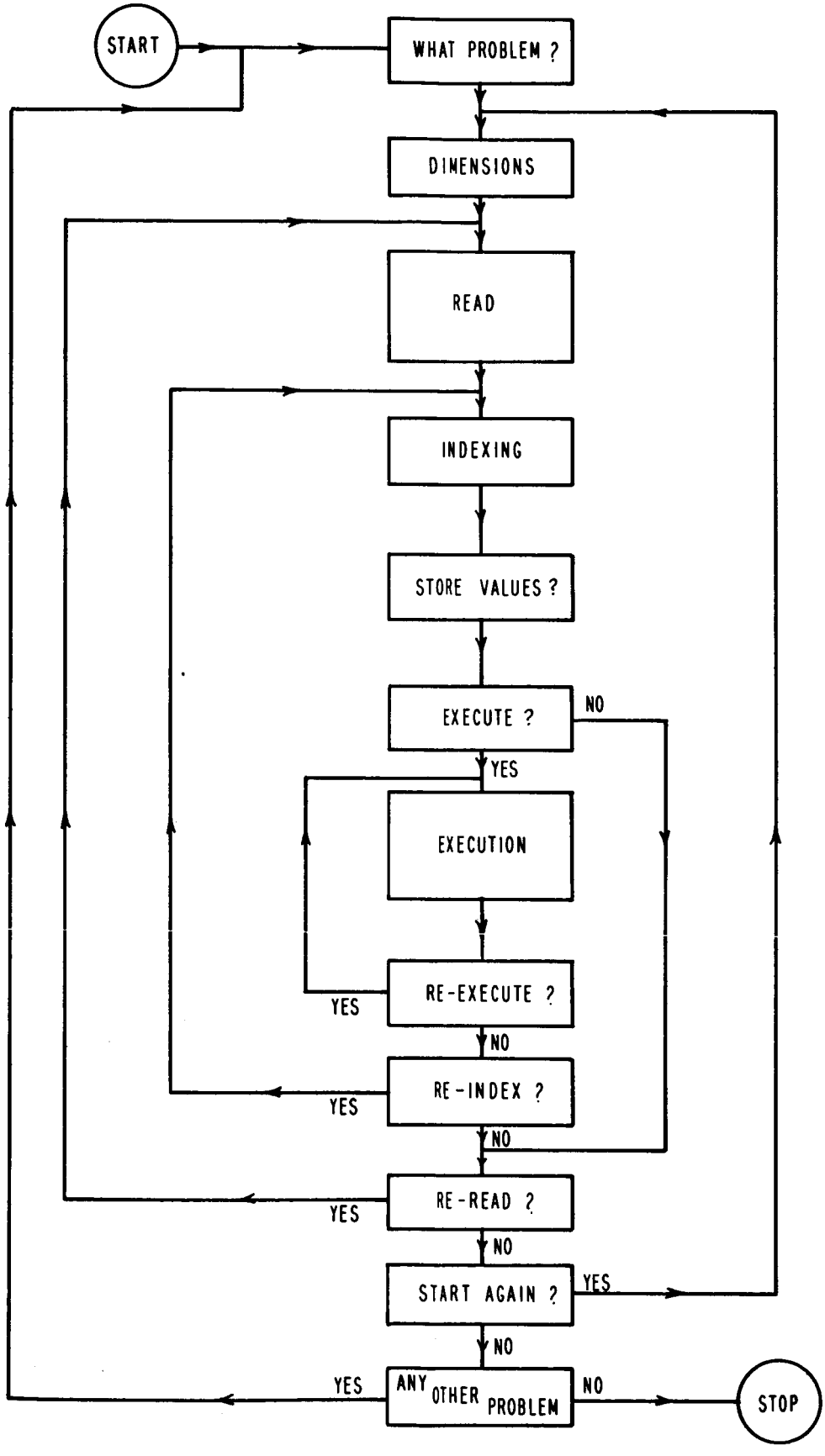

FIG. 3 FLOW DIAGRAM FOR RICCATI AND SIMULATION PROBLEMS.

4. ERROR PROCEDURES.

The errors encountered in a run of CLOSE may be divided into two groups. As CLOSE operates under FOS a Fortran error may occur; in addition, there are some types of errors for which CLOSE checks by itself.

Fortran errors are most likely to occur during input-output. The user can recognize this type of an error by the + sign typed by FOS after the error message (indicating the return of control to FOS). When the user detects an error of this type, a new entry into FOS is recommended, after going back to the executive level by hitting the escape key twice.

The type of errors that the user may encounter more often fall in the second category. When, answering a question the user commits a typographical error, making the answer unintelligible for CLOSE, the message WHAT ? is typed by CLOSE followed by the repetition of the question.

If the user types in the dimensions of the system vectors as more than 10, a message TOO BIG is generated by CLOSE followed by the repetition of the questions about the dimensions. Note that this may happen when the user forgets to terminate the numerical value of dimension by a comma.

5. TYPING THE MATRICES FROM THE CONSOLE.

When the user is typing the values for the matrix elements for any matrix in Phase II of the program, he has to type the numbers row by row after CLOSE has indicated its readiness to accept the new row of values by sounding a double bell on the console. Each entry of the row is terminated by a comma. The vectors are also typed in as a row.

READ {a} MATRIX ?
YES,@
TYPE ?
YES,@
GO ON !
1.3, 1.41, 2.567, 3.45, @
2.67, @
0.0, 1.39, 2.67, 1.4, @
0.98, @
DONE !

Here {a} is a 2 by 5 matrix. CLOSE gave a double bell before the user typed 1.3 and 0.0 the first elements of the rows.

CLOSE does not accept more than 4 numbers to a line. Therefore when matrices with more than 4 columns occur, each row is typed in order, 4 entries per line using up to 3 lines where necessary. Each new row must, however, start on a new line and after the double bell.

Although it is not necessary for the user to know the exact formats used in the program for the input and output, a complete listing is provided for the interested users in Appendix III.

6. FILING SYSTEM.

CLOSE requires a set of 12 files defined as 'PRIVATE', on which matrices are stored as they are typed on or generated as the output in the computations, so that if the occasion arises to read a matrix again during a run it need not be typed again, but may be read from file. Also, these files will have the values stored on them when the user logs-out, which he may use next time he intends to execute some problem. Each matrix is assigned a reserved file name and Table 1 contains a complete list of the file names used.

The user must define the 12 files as 'PRI' before running CLOSE, otherwise a Fortran error will occur. Files are automatically created as 'PRIVATE'.

APPENDIX I

LOADING PROCEDURE

This section is for the benefit of those users who are unfamiliar with the SDS 940 system completely, and describes how to enter and load the program in FOS.*

After getting connected (by dialing 2 from an internal line), the system wants some information about the account number, pass-word, and name. If these are acceptable, the system types @ which is the trade mark of the executive mode. The sequence is shown below. Now the user has to type 'FOS.' and load the main program from /CLOSE/.

```
HARVARD TIME SHARING SYSTEM (DØØ-H16): 1Ø-6-67
NOV 1, 1967  11:Ø3 A M.

ACCOUNT: 11Ø
PASSWORD:
NAME: AGRAWALA.

@FOS.
LOAD MAIN PROGRAM
FROM FILE /CLOSE/
LOAD SUBPROGRAMS
FROM FILE FTL. (LIBRARY)
LOADING COMPLETE
998 WORDS OF STORAGE UNUSED
+;G
```

CLOSE also requires library subprograms to be loaded from FTL. When the system types back the + sign it is ready for a run and a ';G' will pass the control on to CLOSE and a run of CLOSE will follow.

---

* This applies only to the Operating System at Harvard University.

APPENDIX II

EXAMPLE - LATERAL AUTOPILOT PROBLEM

Let us consider a 10^5 lbs. aeroplane flying at 30,000 ft. with a velocity of 500 m.p.h. The equations of lateral motion for the aeroplane may be written as follows.

$$\dot{\beta} = -0.0297\beta - r + 0.0438\phi \tag{II-1}$$

$$\dot{r} = 0.331\beta - 0.0042r - 0.0461p - 0.0668\delta_a - 0.379\delta_r \tag{II-2}$$

$$\dot{p} = -1.135\beta + 0.1286r - 0.795p - 1.587\delta_a - 0.0404\delta_r \tag{II-3}$$

$$\dot{\phi} = p \tag{II-4}$$

$$\dot{\psi} = r \tag{II-5}$$

where (see Fig. 4)

β = sideslip angle

r = yaw angular velocity

p = roll angular velocity

ϕ = roll angle

ψ = yaw angle

δ_a = aileron deflection

δ_r = rudder deflection

The system equations may be written in matrix form as

$$\dot{x} = Fx + Gu \tag{II-6}$$

where

$$F = \begin{bmatrix} -0.0297 & -1. & 0. & 0.0438 & 0. \\ 0.331 & -0.0042 & -0.0461 & 0. & 0. \\ -1.135 & 0.1286 & -0.795 & 0. & 0. \\ 0. & 0. & 1. & 0. & 0. \\ 0. & 1. & 0. & 0. & 0. \end{bmatrix}$$

and

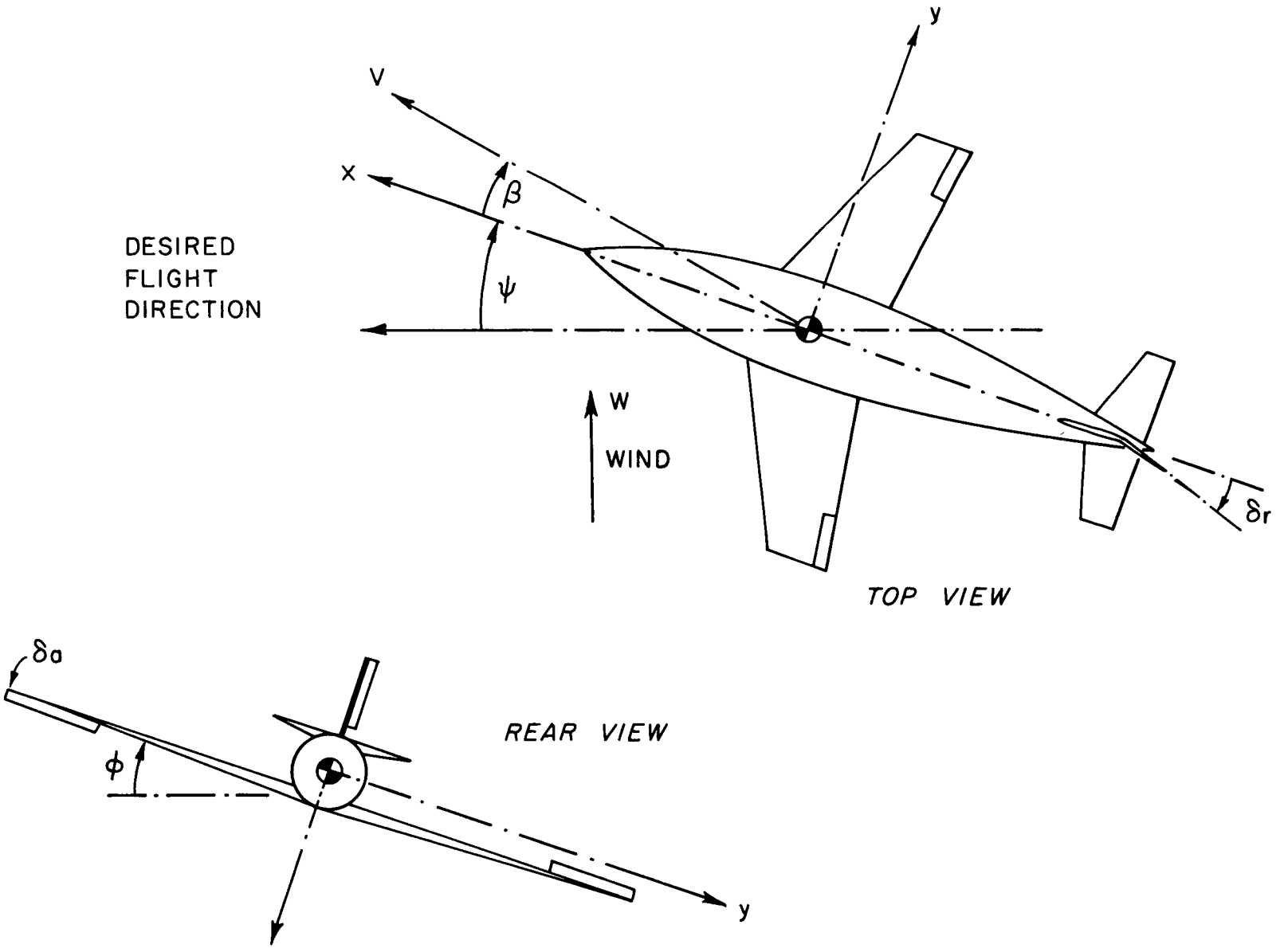

FIG. 4 LATERAL AUTOPILOT PROBLEM

$$G = \begin{bmatrix} 0. & 0. \\ -0.0668 & -0.379 \\ -1.587 & -0.0404 \\ 0. & 0. \\ 0. & 0. \end{bmatrix}$$

treating β, r, p, ϕ and ψ as state variables and δ_a, δ_r as control variables. We want to find the steady state feedback gain matrix K which minimizes the performance index:

$$J = \lim_{t_f - t_0 \to \infty} \int_{t_0}^{t_f} \{\delta_a^2 + \delta_r^2 + 144[(\beta + \psi)^2 + \phi^2]\}\, dt \qquad \text{(II-7)}$$

Note that we are treating this as a regulator problem.

The following pages show the results of an actual run of CLOSE to solve this. The system equations are first discretized with a time step of 0.03 and the Riccati equation is solved to compute the gain matrix. This gain matrix is then used for a simulation with no noise terms.

PROGRAM LISTING

```
+;G
WHAT PROBLEM?
DISCRETIZATION,
DIMENSIONS:
          STATE   = 5,
        CONTROL   = 2,
READ  F  MATRIX ?
YES,
TYPE?
YES,
GO ON!
-0.0297,-1.,0.,0.0438,
0.,
0.331,-0.0042,-0.0461,0.,
0.,
-1.135,0.1286,-0.795,0.,
0.,
0.,0.,1.,0.,
0.,
0.,1.,0.,0.,
0.,
DONE!
READ  G  MATRIX ?
YES,
TYPE?
YES,
GO ON!
0.,0.,
-0.0668,-0.379,
-1.587,-0.0404,
0.,0.,
0.,0.,
DONE!
STORE VALUES?
YES,
EXECUTE?
YES,
**START**
STEP SIZE=0.03,
  PHI
   .99895996E  0  -.29983208E- 1   .40120480E- 4   .13133492E- 2
       .00000000E  0
   .99477994E- 2   .99972211E  0  -.13664173E- 2   .65321088E- 5
       .00000000E  0
  -.33611431E- 1   .43184378E- 2   .97642895E  0  -.22177977E- 4
       .00000000E  0
  -.50634638E- 3   .62514940E- 4   .29645060E- 1   .99999978E  0
       .00000000E  0
   .14913486E- 3   .29996536E- 1  -.20579203E- 4   .65240050E- 7
       .10000000E  1
   D
   .29411011E- 4   .17047160E- 3
  -.19711134E- 2  -.11367876E- 1
  -.47050854E- 1  -.12213413E- 2
  -.70854678E- 3  -.18268848E- 4
  -.29730720E- 4  -.17053011E- 3
DONE!
REREAD ANY MATRICES?
NO,
START AGAIN?
NO,
ANY OTHER PROBLEM?
YES,
```

```
WHAT PROBLEM?
RICCATI
DIMENSIONS:
          STATE   = 5,
        CONTROL   = 2,
READ  A  MATRIX ?
YES,
TYPE?
YES,
GO ON!
144.,0.,0.,0.,
144.,
0.,0.0,0.,0.,
0.,
0.,0.,0.,0.,
0.,
0.,0.,0.,144.,
0.,
144.,0.,0.,0.,
144.,
DONE!
READ  B  MATRIX ?
YES,
TYPE?
YES,
GO ON!
1.,0.,
0.,1.,
DONE!
READ  S  MATRIX ?
YES,
TYPE?
YES,
GO ON!
0.,0.,0.,0.,0.,
0.,
0.,0.,0.,0.,0.,
0.,
0.,0.,0.,0.,
0.,
0.,0.,0.,0.,
0.,
0.,0.,0.,0.,
0.,
DONE!
READ PHI MATRIX ?
YES,
TYPE?
NO,
DONE!
READ  D  MATRIX ?
YES,
TYPE?
NO,
DONE!
SET INDEXING ?
YES,
STARTING INDEX = 500,
NUMBER OF STEPS = -500,
NUMBER OF STEPS BETWEEN OUTPUTS = 250,
STORE VALUES?
YES,
EXECUTE?
YES,
```

```
**START**

S( 500)
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
       .00000000E  0
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
       .00000000E  0
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
       .00000000E  0
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
       .00000000E  0
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
       .00000000E  0

 K ( 500)
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
       .00000000E  0
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
       .00000000E  0


S( 250)
   .19765817E  5   .93222289E  3  -.16507317E  2   .25768488E  3
       .20345639E  5
   .93227613E  3   .15082037E  3  -.25117821E  1   .56679358E  1
       .98354642E  3
  -.16509862E  2  -.25119123E  1   .72034082E  2   .25219949E  3
      -.24110830E  1
   .25768711E  3   .56672545E  1   .25219959E  3   .16468659E  4
       .27872463E  3
   .20345611E  5   .98349360E  3  -.24085421E  1   .27872190E  3
       .20986893E  5

 K ( 250)
   .90707681E  0   .18090235E  0   .32834168E  1   .11079835E  2
       .16739632E  1
   .10477964E  2   .17031543E  1   .51364068E- 1   .33789315E  0
       .11067146E  2

S(   0)
   .21399978E  5   .98392360E  3  -.15120493E  2   .27904573E  3
       .22056647E  5
   .98396181E  3   .15811561E  3  -.27388108E  1   .63898331E  1
       .10393001E  4
  -.15122281E  2  -.27388854E  1   .72048183E  2   .25221522E  3
      -.10395783E  1
   .27904626E  3   .63894415E  1   .25221533E  3   .16471519E  4
       .30095553E  3
   .22056668E  5   .10392597E  4  -.10376360E  1   .30095549E  3
       .22781539E  5

 K (   0)
   .10620361E  1   .18419973E  0   .32836240E  1   .11081842E  2
       .18356291E  1
   .11054133E  2   .17851298E  1   .48802693E- 1   .34597157E  0
       .11688354E  2
 DONE!

 RE-EXECUTE?
 NO,
 RE-INDEX?
 YES,
 SET INDEXING ?
 YES,
```

```
STARTING INDEX = 0,
NUMBER OF STEPS =-10,
NUMBER OF STEPS BETWEEN OUTPUTS = 10,
STORE VALUES?
YES,
EXECUTE?
YES,
**START**

S(   0)
   .21399978E  5   .98392360E  3  -.15120493E  2   .27904573E  3
       .22056647E  5
   .98396181E  3   .15811561E  3  -.27388108E  1   .63898331E  1
       .10393001E  4
  -.15122281E  2  -.27388854E  1   .72048183E  2   .25221522E  3
      -.10395783E  1
   .27904626E  3   .63894415E  1   .25221533E  3   .16471519E  4
       .30095553E  3
   .22056668E  5   .10392597E  4  -.10376360E  1   .30095549E  3
       .22781539E  5

 K (   0)
   .10620361E  1   .18419973E  0   .32836240E  1   .11081842E  2
       .18356291E  1
   .11054133E  2   .17851298E  1   .48802693E- 1   .34597157E  0
       .11688354E  2

S( -10)
   .21402198E  5   .98418041E  3  -.15127301E  2   .27908013E  3
       .22058995E  5
   .98421698E  3   .15815377E  3  -.27400750E  1   .63938086E  1
       .10395630E  4
  -.15128981E  2  -.27401698E  1   .72048269E  2   .25221559E  3
      -.10466839E  1
   .27908025E  3   .63933246E  1   .25221565E  3   .16471559E  4
       .30099129E  3
   .22059012E  5   .10395242E  4  -.10447995E  1   .30099147E  3
       .22783972E  5

 K ( -10)
   .10622028E  1   .18421360E  0   .32836254E  1   .11081859E  2
       .18357953E  1
   .11057023E  2   .17855608E  1   .48788291E- 1   .34601569E  0
       .11691341E  2
DONE!

RE-EXECUTE?
NO,
RE-INDEX?
NO,
REREAD ANY MATRICES?
NO,
START AGAIN?
NO,
ANY OTHER PROBLEM?
YES,
WHAT PROBLEM?
SIMULATION
DIMENSIONS:
          STATE   = 5,
        CONTROL   = 2,
        NOISE     = 1,
```

```
READ PHI MATRIX ?
YES,
TYPE?
NO,
DONE!
READ  D  MATRIX ?
YES,
TYPE?
NO,
DONE!
READ GAM MATRIX ?
NO,
READ  X  MATRIX ?
YES,
TYPE?
YES,
GO ON!
0.,0.,0.,0.,
1.,
DONE!
READ  K  MATRIX ?
YES,
TYPE?
NO,
USE N-TH STORED VALUE OF K FROM FILE
WHERE N= 2,
DONE!
READ AV  MATRIX ?
NO,
READ VAR MATRIX ?

SET INDEXING ?
YES,
STARTING INDEX = 0,
NUMBER OF STEPS =200,
NUMBER OF STEPS BETWEEN OUTPUTS = 10,
STORE VALUES?
YES,
EXECUTE?

**START**

X(   0)
   .00000000E  0   .00000000E  0   .00000000E  0   .00000000E  0
      .10000000E  1

X(  10)
   .18835798E  0  -.12058103E  1  -.35309251E  0  -.82960647E- 1
      .81062908E  0

X(  20)
   .68587141E  0  -.20989124E  1  -.13214357E  0  -.15794831E  0
      .30775456E  0

X(  30)
   .14031124E  1  -.27141081E  1   .25705458E- 1  -.17028470E  0
     -.42087364E  0

X(  40)
   .22583128E  1  -.30679838E  1   .64627447E- 1  -.15460570E  0
     -.12944667E  1
```

```
X(  50)
   .31751839E  1  -.31807425E  1   .56345921E- 1  -.13594767E  0
     -.22374426E  1

X(  60)
   .40853047E  1  -.30815799E  1   .43798033E- 1  -.12095898E  0
     -.31816352E  1

X(  70)
   .49306388E  1  -.28063523E  1   .36480707E- 1  -.10896748E  0
     -.40687248E  1

X(  80)
   .56649415E  1  -.23941589E  1   .32362656E- 1  -.98624515E- 1
     -.48517149E  1

X(  90)
   .62542322E  1  -.18847437E  1   .29053514E- 1  -.89371196E- 1
     -.54954835E  1

X( 100)
   .66765530E  1  -.13166057E  1   .25769257E- 1  -.81116473E- 1
     -.59766889E  1

X( 110)
   .69212751E  1  -.72553260E  0   .22539354E- 1  -.73858170E- 1
     -.62831637E  1

X( 120)
   .69879852E  1  -.14341823E  0   .19530063E- 1  -.67554075E- 1
     -.64129211E  1

X( 130)
   .68851073E  1   .40261382E  0   .16838075E- 1  -.62120458E- 1
     -.63728508E  1

X( 140)
   .66283111E  1   .89075600E  0   .14488594E- 1  -.57455492E- 1
     -.61771765E  1

X( 150)
   .62388130E  1   .13047629E  1   .12479472E- 1  -.53454609E- 1
     -.58458402E  1

X( 160)
   .57416531E  1   .16339428E  1   .10806043E- 1  -.50014259E- 1
     -.54028294E  1

X( 170)
   .51640527E  1   .18728939E  1   .94518406E- 2  -.47033276E- 1
     -.48745258E  1

X( 180)
   .45338717E  1   .20210219E  1   .83876319E- 2  -.44418194E- 1
     -.42882023E  1

X( 190)
   .38782542E  1   .20819461E  1   .75840040E- 2  -.42083659E- 1
     -.36706557E  1

X( 200)
   .32224799E  1   .20627708E  1   .70057245E- 2  -.39954885E- 1
     -.30470588E  1
DONE!
```

```
RE-EXECUTE?
NO,
RE-INDEX?
YES,
SET INDEXING ?

STARTING INDEX = 200,
NUMBER OF STEPS = 300,
NUMBER OF STEPS BETWEEN OUTPUTS = 50,
STORE VALUES?
NO,
EXECUTE?
YES,
**START**

X( 200)
   .32224799E  1   .20627708E  1   .70057245E- 2  -.39954885E- 1
     -.30470588E  1

X( 250)
   .60135410E  0   .11605460E  1   .61871073E- 2  -.30565864E- 1
     -.50784415E  0

X( 300)
  -.24049607E  0   .27505055E- 1   .59672983E- 2  -.21410862E- 1
      .33079682E  0

X( 350)
   .15476664E  0  -.41437521E  0   .46189466E- 2  -.13291632E- 1
     -.61303637E- 1

X( 400)
   .65875864E  0  -.21740733E  0   .26674914E- 2  -.77849283E- 2
     -.58533625E  0

X( 450)
   .69087664E  0   .11437881E  0   .11862807E- 2  -.49999449E- 2
     -.64982442E  0

X( 500)
   .37021062E  0   .23610332E  0   .58424451E- 3  -.38007454E- 2
     -.35371136E  0
DONE!

RE-EXECUTE?
NO,
RE-INDEX?

REREAD ANY MATRICES?
YES,
READ PHI MATRIX ?
NO,
READ  D  MATRIX ?

READ GAM MATRIX ?

READ  X  MATRIX ?
YES,
TYPE?
YES,
GO ON!
0.,0.,0.,1.,
0.,
DONE!
```

```
READ  K  MATRIX ?
NO,
READ AV  MATRIX ?

READ VAR MATRIX ?

SET INDEXING ?
YES,
STARTING INDEX = 0,
NUMBER OF STEPS =100,
NUMBER OF STEPS BETWEEN OUTPUTS = 10,
STORE VALUES?
NO,
EXECUTE?
YES,
**START**

X(   0)
   .00000000E  0   .00000000E  0   .00000000E  0   .10000000E  1
      .00000000E  0

X(  10)
   .32164932E- 1  -.92761340E- 1  -.19508347E  1   .55620067E  0
     -.21472685E- 1

X(  20)
   .56577653E- 1  -.42362356E- 1  -.93152568E  0   .10969500E  0
     -.42272089E- 1

X(  30)
   .62977369E- 1  -.82378571E- 2  -.14347106E  0  -.34981078E- 1
     -.48982101E- 1

X(  40)
   .63051131E- 1  -.26790383E- 2   .75651191E- 1  -.35051837E- 1
     -.50148722E- 1

X(  50)
   .63578302E- 1  -.70368086E- 2   .58666209E- 1  -.12636030E- 1
     -.51545726E- 1

X(  60)
   .65641578E- 1  -.10631131E- 1   .16700100E- 1  -.18066260E- 2
     -.54264992E- 1

X(  70)
   .68405207E- 1  -.11393112E- 1  -.75611213E- 3   .78220670E- 5
     -.57632097E- 1

X(  80)
   .71047990E- 1  -.10155628E- 1  -.26461770E- 2  -.70473778E- 3
     -.60900755E- 1

X(  90)
   .73107220E- 1  -.78170176E- 2  -.88173976E- 3  -.12413881E- 2
     -.63616420E- 1

X( 100)
   .74345068E- 1  -.48582633E- 2   .22733802E- 3  -.13118957E- 2
     -.65529299E- 1
```

```
    DONE!

RE-EXECUTE?
NO,
RE-INDEX?
NO,
REREAD ANY MATRICES?
NO,
START AGAIN?
NO,
ANY OTHER PROBLEM?
NO,

*STOP*
+
```

APPENDIX III

FORMATS

The program CLOSE reads and writes all the matrices in fixed standard formats depending on the number of columns of the matrix. The following formats are used:

No. of Columns	Format
1	(E16.8/)
2	(2E16.8/)
3	(3E16.8/)
4	(4E16.8/)
5	(4E16.8/E20.8/)
6	(4E16.8/E20.8, E16.8/)
7	(4E16.8/E20.8, 2E16.8/)
8	(4E16.8/E20.8, 3E16.8/)
9	(4E16.8/E20.8, 3E16.8/E24.8/)
10	(4E16.8/E20.8, 3E16.8/E24.8, E16.8/)

Standard Fortran notation is used in the above list.

For example the format E16.8 means that the number at input and output appears with an explicit exponent, and 8 digits to the right of the decimal point, while the complete number is right-justified in a field of 16 spaces.

As is obvious from previous examples on input several relaxations are usually employed:

(i) Early termination of a field by a comma.

(ii) Omission of the exponent.

(iii) Shift of the decimal point.

Academy Library (DFSLB)
U. S. Air Force Academy
Colorado Springs, Colorado 80912

AEDC (ARO, INC)
Attn: Library/Documents
Arnold AFB, Tenn. 37389

Aeronautics Library
Graduate Aeronautical Laboratories
California Institute of Technology
1201 E. California Blvd.
Pasadena, California 91109

Aerospace Corporation
P. O. Box 95085
Los Angeles, Calif. 90045
Attn: Library Acquisitions Group

Airborne Instruments Laboratory
Deerpark, New York 11729

AFAL (AVTE/R. D. Larson)
Wright-Patterson AFB
Ohio 45433

AFCRL (CRMXLR)
ARCRL Research Library, Stop 29
L. G. Hanscom Field
Bedford, Mass. 01731

AFETR (ETLIG - 1)
STINFO Officer (for library)
Patrick AFB, Florida 32925

AFETR Technical Library
(ETV, MU-135)
Patrick AFB, Florida 32925

AFFTC (FRBPP-2)
Technical Library
Edwards AFB, Calif. 93523

APGC (PHBPS-12)
Eglin AFB
Florida 32542

ARL (ARIY)
Wright-Patterson AFB
Ohio 45433

AUL3T-9663
Maxwell AFB
Alabama 36112

Mr. Henry L. Bachman
Assistant Chief Engineer
Wheeler Laboratories
122 Cuttermill Road
Great Neck, N. Y. 11021

Bendix Pacific Division
11600 Sherman Way
North Hollywood, Calif. 91605

Colonel A. D. Blue
RTD (RTTL)
Bolling AFB
Washington, D. C. 20332

California Institute of Technology
Pasadena, California 91109
Attn: Documents Library

Carnegie Institute of Technology
Electron al Engineering Dept.
Pittsburg, Pa. 15213

Central Intelligence Agency
Attn: OCR/DD Publications
Washington, D. C. 20505

Chief of Naval Operations
OP-07
Washington, D. C. 20350 [2]

Chief of Naval Research
Department of the Navy
Washington, D. C. 20360
Attn: Code 427 [3]

Commandant
U. S. Army and General Staff College
Attn: Secretary
Fort Leavenworth, Kansas 66370

Commander
Naval Air Development and
Material Center
Johnsville, Pennsylvania 18974

Commanding General
Frankford Arsenal
Attn: SMUFA-L6000 (Dr. Sidney Ross)
Philadelphia, Pa. 19137

Commandant
U. S. Army Air Defense School
Attn: Missile Sciences Div. C and S Dept.
P. O. Box 9390
Fort Bliss, Texas 79916

Commander
U. S. Naval Air Missile Test Center
Point Magu, California 93041

Commanding General
Attn: STEWS-WS-VT
White Sands Missile Range
New Mexico 88002 [2]

Commanding General
U. S. Army Electronics Command
Fort Monmouth, N. J. 07703
Attn: AMSEL-SC
RD-D
RD-G
RD-GF
RD-MAT
XL-D
XL-E
XL-C
XL-S
HL-D
HL-CT-R
HL-CT-P
HL-CT-L
HL-CT-O
HL-CT-I
HL-CT-A
NL-D
NL-A
NL-P
NL-R
NL-S
KL-D
KL-E
KL-S
KL-T
VL-D
WL-D

Commanding General
U. S. Army Material Command
Attn: AMCRD RS DE-E
Washington, D. C. 20315

Commanding General
U. S. Army Missile Command
Attn: Technical Library
Redstone Arsenal, Alabama 35809

Commanding Officer
Naval Avionics Facility
Indianapolis, Indiana 46241

Commanding Officer
U. S. Army Limited War Laboratory
Attn: Technical Director
Aberdeen Proving Ground
Aberdeen, Maryland 21005

Commanding Officer
U. S. Army Materials Research Agency
Watertown Arsenal
Watertown, Massachusetts 02172

Commanding Officer
U. S. Army Security Agency
Arlington Hall
Arlington, Virginia 22212

Commanding Officer and Director
U. S. Naval Underwater Sound Lab.
Fort Trumbull
New London, Conn. 06840

Defense Documentation Center
Attn: TISIA
Cameron Station, Bldg. 5
Alexandria, Virginia 22314 [20]

Det No. 6, OAR (LODAR)
Air Force Unit Post Office
Los Angeles, Calif. 90045

Director
Advanced Research Projects Agency
Department of Defense
Washington, D. C. 20301

Director for Materials Sciences
Advanced Research Projects Agency
Department of Defense
Washington, D. C. 20301

Director
Columbia Radiation Laboratory
Columbia University
538 West 120th Street
New York, New York 10027

Director
Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61803

Director
Electronics Research Laboratory
University of California
Berkeley, California 94720

Director
Electronic Sciences Laboratory
University of Southern California
Los Angeles, California 90007

Director
Microwave Laboratory
Stanford University
Stanford, California 94305

Director - Inst. for Exploratory
Research
U. S. Army Electronics Command
Attn: Mr. Robert G. Parker
Executive Secretary, JSTAC
(AMSEL-XL-D)
Fort Monmouth, N. J. 07703

Director
National Security Agency
Fort George G. Meade
Maryland 20755
Attn: James T. Tippett

Director, Naval Research Laboratory
Technical Information Officer
Washington, D. C.
Attn: Code 2000 [6]

Director
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, Mass. 02139

Director
Stanford Electronics Laboratories
Stanford University
Stanford, California 94305

Commanding Officer
Naval Ordnance Laboratory
Corona, California 91720

Commanding Officer
Naval Ordnance Laboratory
White Oak, Maryland 21502 [2]

Commanding Officer
Naval Ordnance Test Station
China Lake, Calif. 93555

Commanding Officer
Naval Training Device Center
Orlando, Florida 32811

Commanding Officer
Office of Naval Research Branch Office
1030 East Green Street
Pasadena, California

Commanding Officer
Office of Naval Research Branch Office
219 South Dearborn Street
Chicago, Illinois 60604

Commanding Officer
Office of Naval Research Branch Office
495 Summer Street
Boston, Massachusetts 02210

Commanding Officer
Office of Naval Research Branch Office
207 West 24th Street
New York, New York 10011

Commanding Officer
Office of Naval Research Branch Office
Box 39, Fleet Post Office
New York 09510

Commanding Officer
U. S. Army Electronics R & D Activity
White Sands Missile Range
New Mexico 88002

Commanding Officer
U. S. Army Engineer R & D Laboratory
Attn: STINFO Branch
Fort Belvoir, Virginia 22060

Commanding Officer
U. S. Army Research Office (Durham)
Attn: CRD-AA-IP (Richard O. Ulsh)
Box CM, Duke Station
Durham, North Carolina 27706

Commanding General
USASTRATCOM
Technical Information Center
Fort Huachuca, Arizona 85613

Commanding Officer
Harry Diamond
Attn: Dr. Berthold Altman (AMXDO-TI)
Connecticut Ave. & Van Ness St. NW
Washington, D. C. 20438

Commanding Officer
Human Engineering Laboratories
Aberdeen Proving Ground
Maryland 21005

Commanding Officer
U. S. Army Ballistics Research Lab.
Attn: V. W. Richards
Aberdeen Proving Ground
Maryland 21005

Director, USAF Project RAND
Via: Air Force Liaison Office
The RAND Corporation
1700 Main Street
Santa Monica, Calif. 90406
Attn: Library

Director
U. S. Army Engineer Geodesy,
Intelligence and Mapping
Research and Development Agency
Fort Belvoir, Virginia 22060

Director
U. S. Naval Observatory
Washington, D. C. 20390

Director, U. S. Naval Security Group
Attn: G43
3801 Nebraska Avenue
Washington, D. C. 20390

Division of Engineering and Applied
Physics
130 Pierce Hall
Harvard University
Cambridge, Massachusetts 02138

Professor A. A. Dougal, Director
Laboratories for Electronics and
Related Science Research
University of Texas
Austin, Texas 78712

ESD (ESTI)
L. G. Hanscom Field
Bedford, Mass. 01731 [2]

European Office of Aerospace Research
Shell Building
47 Rue Cantersteen
Brussels, Belgium [2]

Colonel Robert E. Fontana
Dept. of Electrical Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

General Electric Company
Research Laboratories
Schenectady, New York 12301

Professor Nicholas George
California Institute of Technology
Pasadena, California 91109

Goddard Space Flight Center
National Aeronautics and Space Admin.
Attn: Library, Documents Section
Code 252
Green Belt, Maryland 20771

Dr. John C. Hancock, Director
Electronic Systems Research Laboratory
Purdue University
Lafayette, Indiana 47907

Dr. H. Harrison, Code RRE
Chief, Electrophysics Branch
National Aeronautics and Space Admin.
Washington, D. C. 20546

Head, Technical Division
U. S. Naval Counter Intelligence
Support Center
Fairmont Building
4420 North Fairfax Drive
Arlington, Virginia 22203

Headquarters
Defense Communications Agency
The Pentagon
Washington, D. C. 20305

Dr. L. M. Hollingsworth
AFCRL (CRN)
L. G. Hanscom Field
Bedford, Massachusetts 01731

Hunt Library
Carnegie Institute of Technology
Schenely Park
Pittsburgh, Pa. 15213

The Johns Hopkins University
Applied Physics Laboratory
8621 Georgia Avenue
Silver Spring, Maryland 20910
Attn: Boris W. Kuvshinoff
Document Librarian

Lt. Col. Robert B. Kalisch
Chief, Electronics Division
Directorate of Engineering Sciences
Air Force Office of Scientific Research
Arlington, Virginia 22209 [5]

Colonel Kee
AFRSTE
Hqs. USAF
Room ID-429, The Pentagon
Washington, D. C. 20330

Dr. S. Benedict Levin, Director
Institute for Exploratory Research
U. S. Army Electronics Command
Fort Monmouth, New Jersey 07703

Los Alamos Scientific Laboratory
Attn: Reports Library
P. O. Box 1663
Los Alamos, New Mexico 87544

Librarian
U. S. Naval Electronics Laboratory
San Diego, California 95152 [2]

Lockheed Aircraft Corp.
P. O. Box 504
Sunnyvale, California 94088

Dr. I. R. Mirman
AFSC (SCT)
Andrews Air Force Base, Maryland

Lt. Col. Bernard S. Morgan
Frank J. Seiler Research Laboratory
U. S. Air Force Academy
Colorado Springs, Colorado 80912

Dr. G. J. Murphy
The Technological Institute
Northwestern University
Evanston, Illinois 60201

Mr. Peter Murray
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

NASA Lewis Research Center
Attn: Library
21000 Brookpark Road
Cleveland, Ohio 44135

NASA Scientific & Technical
Information Facility
Attn: Acquisitions Branch (S/AK/DL)
P. O. Box 33
College Park, Maryland 20740 [2]

National Science Foundation
Attn: Dr. John R. Lehmann
Division of Engineering
1800 G Street, NW
Washington, D. C. 20550

National Security Agency
Attn: R4 - James Tippet
Office of Research
Fort George C. Meade, Maryland 20755

Naval Air Systems Command
AIR 03
Washington, D. C. 20360 [2]

Naval Electronics Systems Command
ELEX 03
Falls Church, Virginia 22046 [2]

Naval Ordnance Systems Command
ORD 32
Washington, D. C. 20360 [2]

Naval Ordnance Systems Command
SHIP 035
Washington, D. C. 20360

Naval Ship Systems Command
SHIP 031
Washington, D. C. 20360

New York University
College of Engineering
New York, New York 10019

Dr. H. V. Noble
Air Force Avionics Laboratory
Wright-Patterson AFB, Ohio 45433

Office of Deputy Director
(Research and Information Rm. 3D1037)
Department of Defense
The Pentagon
Washington, D. C. 20301

Polytechnic Institute of Brooklyn
55 Johnson Street
Brooklyn, New York 11201
Attn: Mr. Jerome Fox
Research Coordination

RAD (EMLAL-1)
Griffiss AFB, New York 13442
Attn: Documents Library

Raytheon Company
Bedford, Mass. 01730
Attn: Librarian

Lt. Col. J. L. Reeves
AFSC (SCBB)
Andrews Air Force Base, Md. 20331

Dr. A. A. Dougal
Asst. Director of Research
Office of Defense Res. and Eng.
Department of Defense
Washington, D. C. 20301

Research Plans Office
U. S. Army Research Office
3045 Columbia Pike
Arlington, Virginia 22204

Dr. H. Robl, Deputy Chief Scientist
U. S. Army Research Office (Durham)
Durham, North Carolina 27706

Emil Schafer, Head
Electronics Properties Info. Center
Hughes Aircraft Company
Culver City, California 90230

School of Engineering Sciences
Arizona State University
Tempe, Arizona 85281

SAMSO (SMSDI-STINFO)
AF Unit Post Office
Los Angeles, California 90045

SSD (SSTRT/Lt. Starbuck)
AFUPO
Los Angeles, California 90045

Superintendent
U. S. Army Military Academy
West Point, New York 10996

Colonel A. Swan
Aerospace Medical Division
AMD (AMRXI)
Brooks AFB, Texas 78235

Syracuse University
Dept. of Electrical Engineering
Syracuse, New York 13210

University of California
Santa Barbara, California 93106
Attn: Library

University of Calif. at Los Angeles
Dept. of Engineering
Los Angeles, California 90024

University of Michigan
Electrical Engineering Dept.
Ann Arbor, Michigan 48104

U. S. Army Munitions Command
Attn: Technical Information Branch
Picatinney Arsenal
Dover, New Jersey 07801

U. S. Army Research Office
Attn: Physical Sciences Division
3045 Columbia Pike
Arlington, Virginia 22204

U. S. Atomic Energy Commission
Division of Technical Information Ext.
P. O. Box 62
Oak Ridge, Tenn. 37831

Dept. of Electrical Engineering
Texas Technological College
Lubbock, Texas 79409

U. S. Naval Weapons Laboratory
Dahlgren, Virginia 22448

Major Charles Waespy
Technical Division
Deputy for Technology
Space Systems Division, AFSC
Los Angeles, California 90045

The Walter Reed Institute of Research
Walter Reed Medical Center
Washington, D. C. 20012

AFSC (SCTR)
Andrews Air Force Base
Maryland 20331

Weapons Systems Test Division
Naval Air Test Center
Patuxtent River, Maryland 20670
Attn: Library

Weapons Systems Evaluation Group
Attn: Col. Daniel W. McElwee
Department of Defense
Washington, D. C. 20305

Yale University
Engineering Department
New Haven, Connecticut 06720

Mr. Charles F. Yost
Special Asst. to the Director of Research
NASA
Washington, D. C. 20546

Dr. Leo Young
Stanford Research Institute
Menlo Park, California 94025

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Division of Engineering and Applied Physics
Harvard University
Cambridge, Massachusetts 02138

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
TWO CONVERSATIONAL LANGUAGES FOR CONTROL-THEORY COMPUTATIONS IN THE TIME-SHARING MODE

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Interim technical report

5. AUTHOR(S) (First name, middle initial, last name)
P. M. Newbold and A. K. Agrawala

6. REPORT DATE
November 1967

7a. TOTAL NO. OF PAGES
50

7b. NO. OF REFS
5

8a. CONTRACT OR GRANT NO.
N00014-67-A-0298-0006 and

b. PROJECT NO. NGR-22-007-068

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
Technical Report No. 546

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
Reproduction in whole or in part is permitted for any purpose of the United States Government.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Office of Naval Research

13. ABSTRACT

The paper presents in the form of users' manuals, a description of two conversational languages for use on a direct-access time-sharing computer. The languages are designed for control-theoretic applications. The first language is a matrix manipulation language. The second is designed to simulate linear dynamic systems and to solve the associated Riccati equations. Further applications are also possible. Each manual includes detailed examples of the usage of the languages.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
time-sharing computer programs matrix algebra dynamic system simulation solution of matrix equations computer languages for control theory						