BELLCOMM, INC.
955 L'ENFANT PLAZA NORTH, S.W. WASHINGTON, D. C. 20024

B70 06073

SUBJECT: Evaluation of Routines for Numerical Solution of the Matrix Equation $AX+XA^T+B=0$ Case 620

DATE: June 25, 1970

FROM: P. G. Smith
P. R. Dowling

ABSTRACT

It is shown that accurate solutions of $AX+XA^T+B=0$ can be obtained for dimension up to 146 x 146 by using Fortran subroutine MTXEQN (listing attached), which employs a matrix recursion formula for X. This conclusion is based on the solution of trial problems in which mean square response is computed for a lightly damped structure of variable dimension excited by white noise.

With MTXEQN one can obtain the response to random excitation of constant coefficient linear systems that are both larger and have less energy dissipation than is practical with other methods. In addition to structural equations, such systems might include the dynamics of control systems and other hardware.

BELLCOMM, INC.
955 L'ENFANT PLAZA NORTH, S.W. WASHINGTON, D.C. 20024

B70 06073

SUBJECT: Evaluation of Routines for Numerical Solution of the Matrix Equation $AX+XA^T+B=0$ Case 620

DATE: June 25, 1970

FROM: P. G. Smith
P. R. Dowling

MEMORANDUM FOR FILE

INTRODUCTION

Several problems of interest at Bellcomm require finding the mean square response of a linear system excited by random inputs. The system might include, for example, the dynamics of structures, control systems, experiment mounting devices, and instruments. Two approaches have been used for solving such problems: numerical integration of a system of differential equations driven by the filtered output of a random number generator [1] and methods in which mean square response is obtained from power spectral density [2]. Both approaches have computational shortcomings, however, when applied to large linear systems that include lightly damped structures: the former is computationally costly relative to power spectral density methods, and the latter, which relies on integration in the frequency domain, is subject to error when little damping exists.

Bar-Itzhack and Hou [3] have brought to light a third approach* that requires in the case of a time-invariant linear system the solution of

$$AX + XA^T = -B \tag{1}$$

for the symmetric N x N matrix X; A is the N x N coefficient matrix of the linear system, B is a symmetric N x N matrix related to the random input, and T denotes transposition. Unique solutions of (1) exist if and only if A and -A have no eigenvalues in common [4], so (1) has a unique solution whenever A represents an asymptotically stable system.

---

*This approach is summarized in Appendix A.

P. G. Smith [5] has suggested for further evaluation two means of solving equation (1) numerically,* one being a matrix recursion formula due to R. A. Smith [6] that has since been programmed as subroutine MTXEQN, and the other being an existing program called GASP [7] that generates X from the eigenvectors of a 2N x 2N Hamiltonian matrix. Our objective here is to determine how well each of these programs solves equation (1) for a trial problem involving a lightly damped structure of variable dimension.

TRIAL PROBLEM AND TEST PROGRAM

As shown in Figure 1, the trial problem represents $N/_2$ particles having masses $m_1,\ldots,m_{N/_2}$ connected by $N/_2$ springs having constants $k_1,\ldots,k_{N/_2}$ and by $N/_2$ dampers having constants $d_1,\ldots,d_{N/_2}$, respectively; displacements of the masses are denoted $y_1,\ldots,y_{N/_2}$, respectively. The system is an extension of the single degree of freedom system of Reference 7 to $N/_2$ degrees of freedom.

For simplicity take P(t) to be white noise, and let $k_i = m_i = 1$, $i = 1,\ldots,N/_2$.** Let damping be proportional to stiffness, and choose the constant of proportionality such that the first mode of vibration has a damping factor δ.

The test program generates A and B (B = diag (0,..., 0,1) for the case at hand), solves equation (1), and computes the trace of $\hat{X}$

$$\text{tr } \hat{X} = \sum_{i=1}^{N} \hat{x}_{ii}$$

---

*Reference 5 actually deals with the so-called Lyapunov equation, $XA + A^TX = -B$, which differs from equation (1) only in the transposition of A. The change is made to avoid the extra step of transposing A before solving the matrix equation.

**For this choice, the ratio of the lowest to highest natural frequencies is 0.18 when N = 50; this ratio, when small, is thought by some to increase the difficulty of solving the matrix equation.

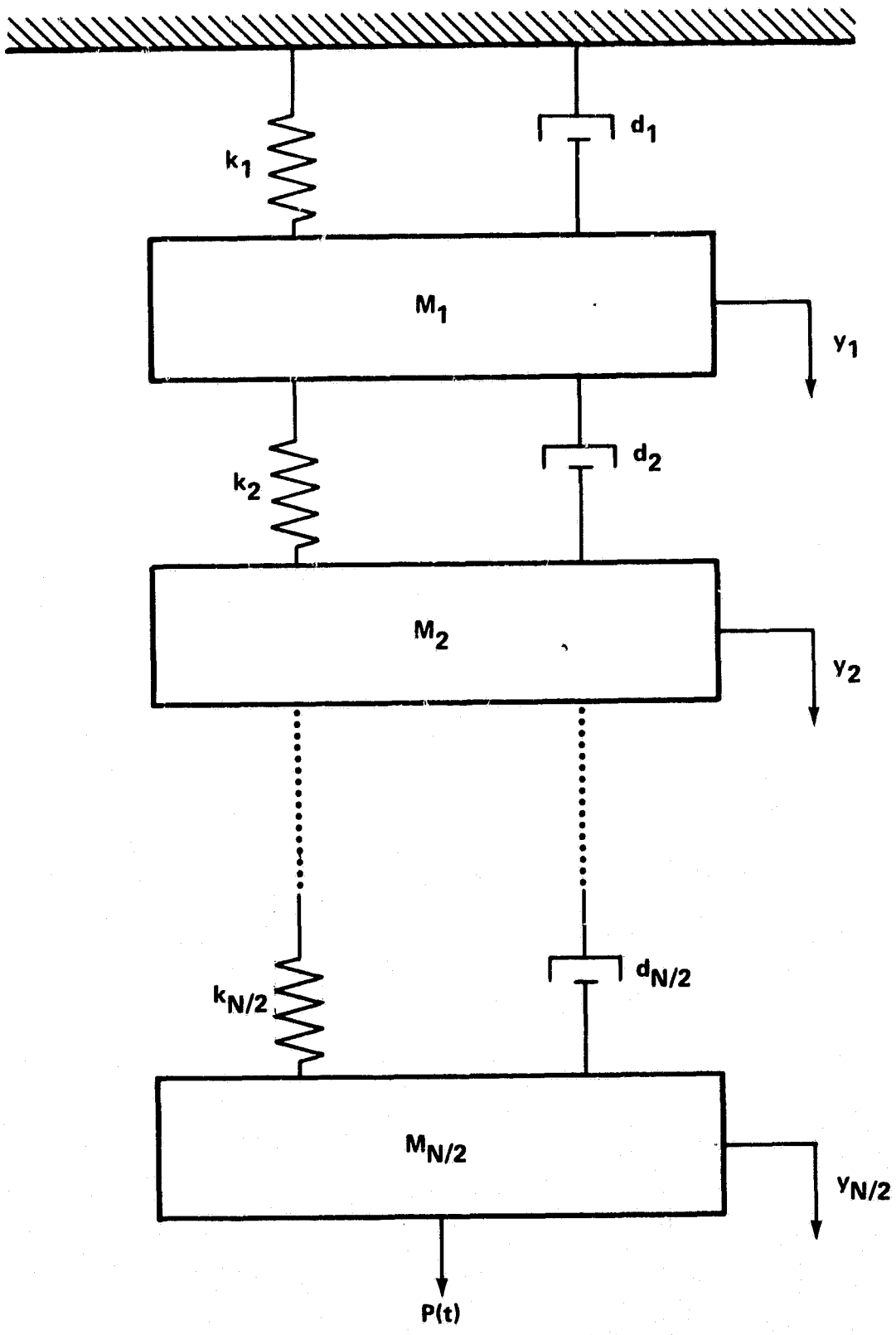

Figure 1
Physical System Represented

where $\hat{X}$ is the computed solution (as opposed to the exact solution X). The remainder, R, is computed as

$$R = A\hat{X} + \hat{X}A^T + B$$

Then the error $E = \hat{X} - X$ is computed by solving the matrix equation

$$AE + EA^T = -(-R)$$

and tr $E = \sum_{i=1}^{N} e_{ii}$ is obtained.

Because only the diagonal elements of X are of interest for the application discussed in the introduction and because it is known a priori that these elements are non-negative,

tr E/tr $\hat{X}$ gives a good indication of accuracy of the computed solution.

RESULTS FOR MTXEQN

The Fortran subroutine MTXEQN, written by J. Pascher, solves equation (1) using a recursive procedure with quadratic rate of convergence. The method converges whenever all eigenvalues of A have negative real parts, as will hold for the trial problem whenever $\delta>0$. MTXEQN is listed in Appendix B.

Two parameters must be specified before using MTXEQN. One is a positive scalar Q that appears in the original derivation [6], and the other is a relative error bound, ERROR, that terminates the iteration.

Core storage in the Bellcomm Univac 1108 can accommodate MTXEQN for $N \leq 146$.* Solution of a maximum-size problem requires

---

*More elaborate programming of MTXEQN may permit some increase in this value.

about 232 charges;* this number accounts just for the time MTXEQN takes to solve the matrix equation. For this problem tr E/tr $\hat{X}$ is 0.515×10^{-3}, which demonstrates that even though N is large, accuracy is satisfactory.

Figures 2 and 3 show how accuracy and the number of charges, respectively, vary with N. Surprisingly, rather than increasing with N, the number of iterations remains almost constant at 12.

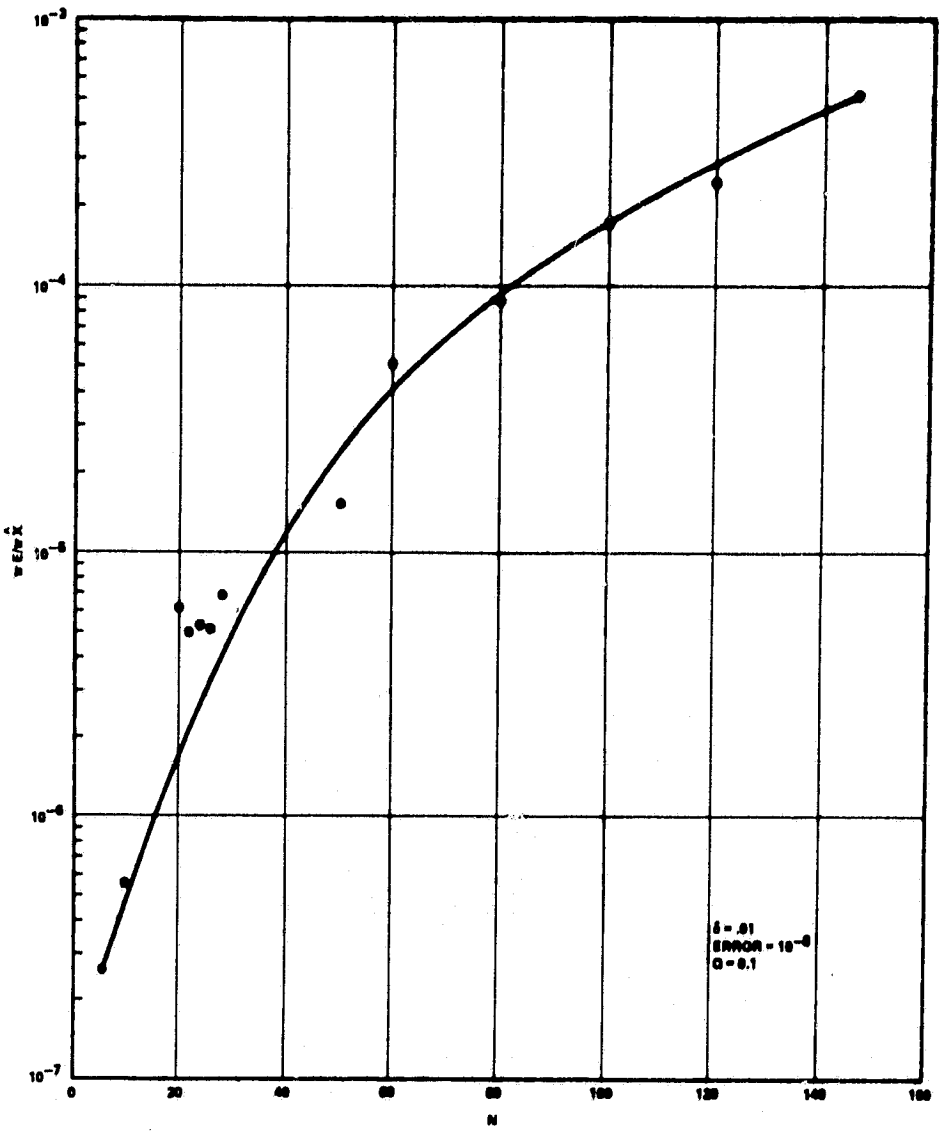

Figure 2

Accuracy of MTXEQN

*Charges for the 1108 are explained in Appendix C.

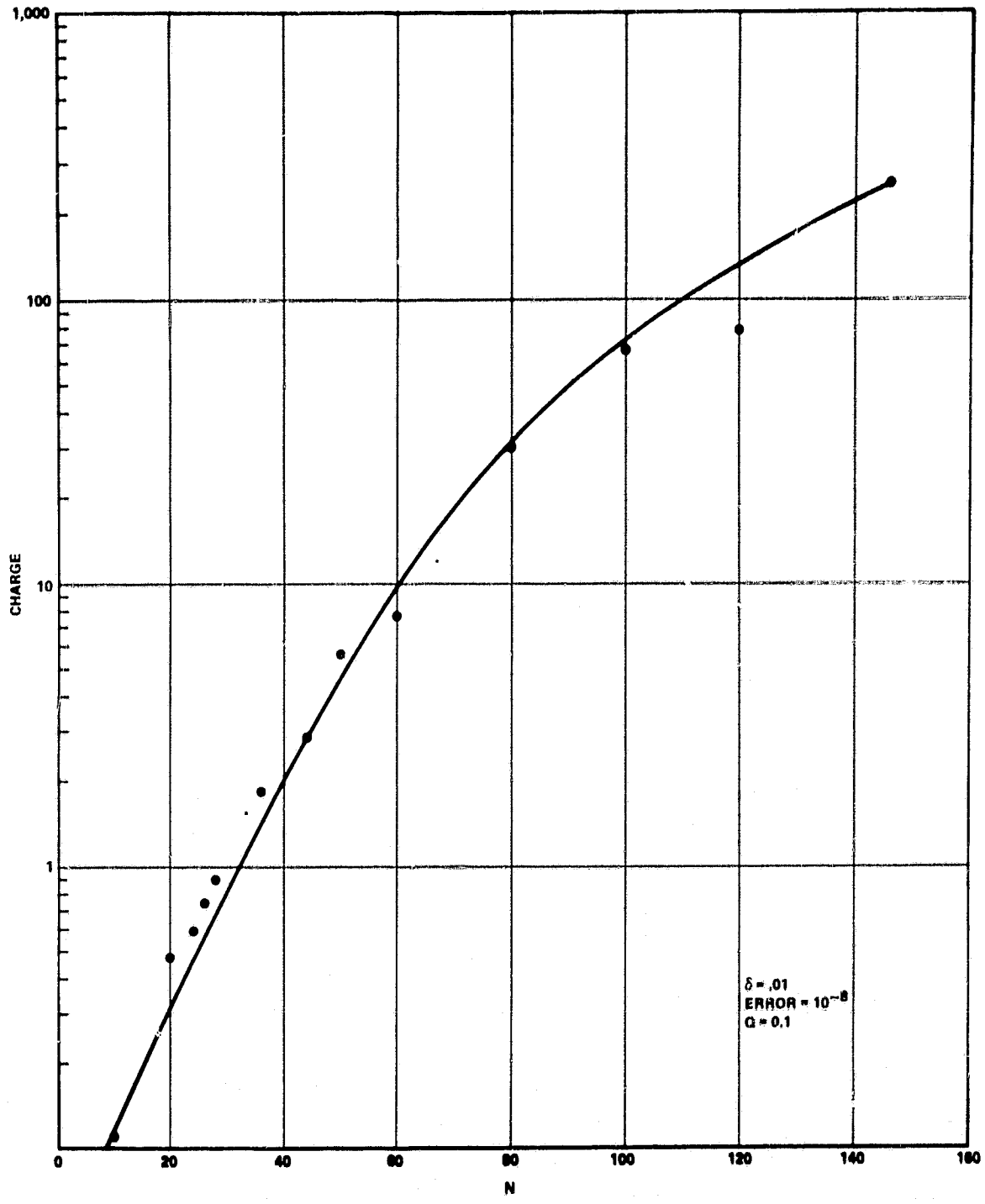

Figure 3

Charges Required by MTXEQN

A basic case is taken with N = 50, δ = .01, Q = 0.1, and ERROR = 10^{-8}; it is felt that this case is typical in size of future problems. The basic case consumes about 6 charges, requires 11 iterations, and yields a tr E/tr $\hat{X}$ of 1.5 x 10^{-5}.

Table 1 shows the effect of varying δ (the damping factor). As δ is decreased the number of iterations generally increases and accuracy becomes poorer. For $\delta \geq 10^{-3}$, accuracy is satisfactory.

Varying ERROR over the range $2. \times 10^{-13}$ to $2. \times 10^{-3}$ gives no significant change in the number of iterations. For ERROR greater than $2. \times 10^{-3}$, accuracy is not maintained, as shown in Table 2.

Data in Table 3 indicate that an optimal value of Q exists for which both the charge and the error are small. For the basic case this value is about 0.1.

RESULTS FOR GASP

GASP[7] is a group of Fortran programs written at Martin Marietta Corporation to solve optimal control problems. The programs have been given to Bellcomm, and one of them, PBSOL, which obtains steady-state solutions to the matrix Riccati equation,

$$\dot{X} = XA + A^T X + B + XCX$$

by computing the eigenvectors of the 2N x 2N (nonsymmetric) Hamiltonian matrix

$$H = \begin{bmatrix} -A & -C \\ B & A^T \end{bmatrix}$$

may be used to solve equation (1). However, for the trial problem, it was found to be slower and less accurate than MTXEQN. GASP, as now written, is constrained to $N \leq 50$, although we found that accuracy is unsatisfactory when $N > 40$.

CONCLUSIONS

It has been shown that an accurate solution to the matrix equation (1) may be obtained by using MTXEQN with input

Q = 0.1 and ERROR = 10^{-8}. This conclusion holds for all N up to the capacity of the computer (N = 146) and for all $\delta \geq 10^{-3}$.

The parameter Q appearing in MTXEQN has an optimum value of about 0.1 for the problem considered here, and this can serve as a starting value for the other problems as well. Parameter ERROR has little effect on the results as long as its value is smaller than 10^{-2}.

For handling the present task, MTXEQN is better than GASP, but GASP can handle equations more general than(1). We hope to obtain the Boeing program [9], which is similar to but more refined than GASP.

ACKNOWLEDGEMENT

The writers are grateful to Dr. Tom Hendricks (Martin Marietta) for supplying GASP and advice on using it. Acknowledgement is also due Jerry Pascher, who implemented GASP at Bellcomm and who also wrote MTXEQN.

P. G. Smith

P. R. Dowling

1072-PGS-PRD-mef

Attachments

Table 1

The Effect of Changing δ

δ	NUMIT (NO. OF ITERATIONS)	Charge	tr E/tr $\hat{X}$
10^{-6}	25	17.4	3.3×10^{-1}
10^{-5}	21	12.6	2.6×10^{-1}
10^{-4}	18	10.9	2.4×10^{-1}
10^{-3}	15	7.8	2.4×10^{-4}
10^{-2}	11	5.8	1.5×10^{-5}
10^{-1}	10	6.9*	1.0×10^{-6}
10^{-0}	13	6.5	1.1×10^{-6}

N = 50

Q = 0.1

ERROR = 10^{-8}

*The charge routine is not always consistent, due to the multi-programming system.

Table 2

The Effect of Changing ERROR

ERROR	NUMIT (NO. OF ITERATIONS)	Charge	tr E/tr $\hat{X}$
2×10^{-13}	12	8.0	1.5×10^{-5}
1×10^{-8}	11	5.8	1.5×10^{-5}
2×10^{-4}	10	4.6	1.5×10^{-5}
2×10^{-3}	10	2.8	1.5×10^{-5}
1×10^{-2}	9	2.8	9.0×10^{-5}
2×10^{-1}	8	4.9	1.0×10^{-2}
5×10^{-1}	3	1.7	3.1×10^{-1}
1×10^{-0}	1	.9	4.7×10^{-1}

$N = 50$

$\delta = 10^{-2}$

$Q = 0.1$

Table 3

The Effect of Changing Q

Q	NUMIT (NO. OF ITERATIONS)	Charge	tr E/tr $\hat{X}$
10.0	18	20.3	4.0×10^{-4}
1.0	14	7.2	6.2×10^{-5}
0.3	13	6.7	1.1×10^{-4}
0.1	11	5.8	1.5×10^{-5}
0.05	11	3.4	1.2×10^{-5}
0.01	13	5.1	1.0×10^{-5}
0.001	16	5.7	8.8×10^{-1}

$N = 50$

$\delta = 10^{-2}$

$ERROR = 10^{-8}$

APPENDIX A

Origin of the Covariance Matrix Equation

Given the linear system

$$\frac{dy}{dt} = Ay + b$$

in which y is the state vector and b is a white noise vector (other types of random excitation can be accommodated by including so-called shaping filters in A [3]), we desire the mean square value of y. Let X be the covariance of y,

$$X = E\{yy^T\}$$

Then the desired values, $E\{y_i^2\}$, are just the diagonal elements X_{ii} of X.

Let

$$E\{b(t_1)b^T(t_2)\} = B\,\delta(t_1-t_2)$$

where δ is the Dirac delta function. It is shown in Reference 3 that X satisfies the matrix differential equation

$$\frac{d}{dt}X = AX + XA^T + B$$

where matrices A and B may be time-varying. For the time-invariant case, steady-state solutions of this differential equation can be obtained by solving equation (1).

APPENDIX B

Listing of Fortran Subroutine MTXEQN

```
C TITLE          MTXEQN, MATRIX RICCATI EQUATION
C
C AUTHOR         J. R. PASCHER
C
C SPONSOR        P. G. SMITH
C
C DATE           FEBRUARY 1970
C
C KEY WORDS      MATRIX RICCATI EQUATION, SYMMETRIC MATRIX,
C                MATRIX TRANSPOSE, MATRIX INVERSE
C
C PURPOSE        TO SOLVE A CLASS OF MATRIX RICCATI EQUATIONS GIVEN THE
C                COEFFICIENT MATRICES.
C                THE PROGRAM IS TO BE USED FOR LARGE MATRICES, AND
C                THEREFORE SHOULD BE PROGRAMMED TO ECONOMIZE ON CORE
C                STORAGE REQUIREMENTS.
C
C METHOD         SOLUTION IS OBTAINED IN THE FORM OF A MATRIX INFINITE
C                SEQUENCE. TERMS ARE COMPUTED ONE AT A TIME UNTIL THE
C                DIFFERENCE OF SUCCESSIVE TERMS IS CONSIDERED TO BE SMALL
C                ENOUGH. THE LAST TERM OF THE SEQUENCE WHICH IS COMPUTED,
C                IS THE COMPUTED SOLUTION OF THE MATRIX RICATTI EQUATION.
C
C REFERENCES     ' MATRIX EQUATION XA + BX = C ', R. A. SMITH
C                SIAM J. APPL. MATH.,VOL. 16, NO. 1, 1968 ( 198 - 201 )
C
C                ' LITERATURE SURVEY ON NUMERICAL SOLUTION OF MATRIX
C                EQUATIONS ' , P. G. SMITH
C                BELLCOMM, MEMORANDUM FOR FILE B70 01040,JAN 23,1970 (7-9)
C
C CALL           CALL MTXEQN( A,B,ND,NH,N,X,E,Q,D,ERR,IT,$,NCASE )
C
C INPUT          A, B                    THE COEFFICIENT MATRICES OF THE
C                                        MATRIX RICCATI EQUATION TO BE SOLVED.
C
C                ND                      THE MAXIMUM DIMENSION OF THE MATRICES
C                                        A,B,X OF THE MATRIX EQUATION.
C
C                NH                      THE MAXIMUM DIMENSION FOR THE
C                                        SYMMETRIC MATRICES B AND X (THE LOWER
C                                        TRIANGULAR PARTS OF B AND X ARE EACH
C                                        STORED IN MTXEQN AS A ONE DIMENSIONAL
C                                        ARRAY OF MAXIMUM LENGTH NH, TO SAVE
C                                        STORAGE. )
```

```
C
C          N                    THE ACTUAL DIMENSION OF THE MATRICES
C                               A, B, X FOR THE CURRENT PROBLEM.
C
C          E, D                 ARRAYS NEEDED IN THE COURSE OF
C                               COMPUTATION
C
C          Q                    AN ARBITRARY POSITIVE CONSTANT
C                               ( E.G.  Q = 0.1 )
C
C          ERR                  A CONVERGENCE CONSTANT. COMPUTE THE
C                               RELATIVE ERROR OBTAINED FROM THE
C                               DIFFERENCE OF THE PREVIOUS TERM OF
C                               THE SEQUENCE AND THE PRESENT TERM.
C                               IF THIS ERROR IS SMALLER THAN ERR,
C                               STOP COMPUTING NEW TERMS AND CALL THE
C                               CURRENT TERM THE SOLUTION.
C
C          IT                   THE MAXIMUM NUMBER OF TERMS TO USE IN
C                               THE SEQUENCE. IF THE SEQUENCE DOES
C                               NOT CONVERGE BY THEN, ASSUME IT DOES
C                               NOT CONVERGE AND STOP.
C
C               $               AN ERROR RETURN STATEMENT NUMBER
C
C               NCASE           DETERMINES WHICH FORM OF THE MATRIX
C                               RICCATI EQUATION IS TO BE SOLVED.
C
C OUTPUT        X               THE SOLUTION MATRIX
C
C ROUTINES USED    GJR          MATHPACK ROUTINE FOR MATRIX INVERSION
C
C NOTE B AND X ARE IN REALITY SQUARE MATRICES OF ORDER N. SINCE THEY ARE
C      BOTH SYMMETRIC, TO SAVE CORE STORAGE THEY ARE STORED AS ONE
C      DIMENSIONAL ARRAYS( LOWER TRIANGULAR PART OF MATRIX, COLUMNWISE )
C      THEREFORE, INPUT OF B TO MTXEQN AND OUTPUT OF X FROM MTXEQN
C      SHOULD BE IN THIS FORM.
C
C NOTE D AND B SHOULD BE EQUIVALENCED IN THE CALLING PROGRAM.
C      THE PURPOSE OF THIS IS ALSO TO SAVE CORE STORAGE.
C
C      THE SUBROUTINE MTXEQN SOLVES ONE OF THE FOLLOWING TWO MATRIX
C      RICCATI EQUATIONS, DEPENDING ON THE VALUE OF NCASE
C      X * A + AT * X + B = 0       ( IF NCASE = 1 )
C      X * AT + A * X + B = 0       ( IF NCASE = 2 )
C      WHERE AT MEANS A TRANSPOSED.
C      A, B, X ARE ALL NXN MATRICES. B AND X ARE SYMMETRIC.
C
C      STEP 1            CALCULATE X( 0 ) = AT * B * A
C      DO THIS USING ONLY TWO FULL NXN MATRICES.
C
C      STEP 2    THE EXACT SOLUTION X IS THE LIMIT OF THE SEQUENCE X(N).
C      HERE THE SUBSCRIPT N REFERS TO THE N-TH TERM OF THE SEQUENCE.
C      COMPUTE EACH TERM X(N+1) RECURSIVELY, BASED ON X(N) AS FOLLOWS.
```

```
C       X( N + 1 ) = X( N ) + U( N ) * X( N ) * V( N ).
C       STOP COMPUTING TERMS OF THE SEQUENCE WHEN SUCCESSIVE TERMS
C       BECOME VERY CLOSE.
C       U( N ), V( N ) DEPEND ON A AS FOLLOWS,
C       U( 0 ) = (  ( Q * I - AT ) INVERSE )  *  ( Q * I + AT )
C       WHERE I IS THE IDENTITY MATRIX.
C       U( N ) = U( 0 ) ** ( 2 ** N )
C       V( N ) = U( N ) TRANSPOSED, FOR ALL N
C       THIS STEP REQUIRES 2.5 * N * N LOCATIONS
C
C
              SUBROUTINE MTXEQN(A,B,ND,NH,N,X,E,Q,D,ERR,IT,$,NCASE)
C
C
              DIMENSION A(ND,ND),B(NH),X(NH),E(ND),D(ND,ND),P(2)
C
C       D AND B ARE EQUIVALENCED SO THAT THEY ARE ALLOCATED A TOTAL OF
C       N TIMES N LOCATIONS.
C
              INDEX(I,J)=I+((J-1)*(2*N-J))/2
              N2=N*(N+1)/2
              DO 1 I = 1 , N
C
C       A = A - Q * I
C
 1                A( I , I ) = A( I , I ) - Q
                  Q2 = 2.0 * Q
                  P( 1 ) = 1
C
C       OPTIONAL PRINT STATEMENTS
C
C                 WRITE( 6 , 900 )
C                 WRITE( 6 , 920 ) (( A( I , J ), J = 1,N ),I = 1,N )
C                 WRITE( 6 , 905 )
C                 WRITE( 6 , 920 ) ( B( I ) , I = 1 , N2 )
C                 WRITE( 6 , 905 )
C
C       USE MATHPACK ROUTINE GJR TO OBTAIN THE INVERSE OF MATRIX A.
C       ( OR ANY OTHER MATRIX INVERSION ROUTINE WHICH STORES THE ANSWER
C       A INVERSE, BACK INTO A )
C
C       A = ( A ) INVERSE
C
              CALL GJR(A,ND,ND,N,N,$295,E,P)
C
              DO 20 K=1,N
C
C       FORM B * A AND STORE IN E
C
              DO 10 I=1,N
              E(I)=0.0
              IF( NCASE .EQ. 2 ) GO TO 3
              DO 2 J=1,I
              IJ=INDEX(I,J)
```

```
 2              E(I)=E(I)+B(IJ)*A(J,K)
                GO TO 5
 3              DO 4 J = 1 , I
                IJ=INDEX(I,J)
 4              E(I)=E(I)+B(IJ)*A(K,J)
 5              JJ=I+1
                IF(JJ.GT.N)GO TO 10
                IF( NCASE .EQ. 2 ) GO TO 7
                DO 6 J=JJ,N
                IJ=INDEX(J,I)
 6              E(I)=E(I)+B(IJ)*A(J,K)
                GO TO 10
 7              DO 9  J = JJ , N
                IJ=INDEX(J,I)
 8              E(I)=E(I)+B(IJ)*A(K,J)
 9              CONTINUE
 10             CONTINUE
C
C       COMPUTE X = 2 * Q * AT * B * A
C
                DO 15 IKT=K,N
                IJ=INDEX(IKT,K)
                X( IJ ) = 0.0
                IF( NCASE .EQ. 2 ) GO TO 13
                DO 12 J=1,N
 12             X(IJ ) = X( IJ ) + A( J , IKT ) * E( J ) * Q2
                GO TO 15
 13             DO 14 J = 1 , N
 14             X(IJ ) = X( IJ ) + A( IKT , J ) * E( J ) * Q2
 15             CONTINUE
 20             CONTINUE
C               WRITE( 6 , 905 )
C               WRITE(6,920)(X(I),I=1,N2)
C
C       COMPUTE A = - 2 * Q * A
C
                DO 25 I = 1 , N
                DO 25 J = 1 , N
 25             A( I , J ) = - Q2 * A( I , J )
C
C       COMPUTE A = A - I
C
                DO 30 I = 1 , N
 30             A( I , I ) = A( I , I ) - 1
C
C       OPTIONAL PRINT STATEMENTS
C
C               WRITE( 6 , 900 )
C               WRITE( 6 , 920 ) (( A( I , J ), J = 1,N ),I = 1,N )
C               WRITE( 6 , 905 )
C               WRITE( 6 , 920 ) ( B( I ) , I = 1 , N2 )
C               WRITE( 6 , 905 )
C
C       ITERATIVE PART OF THE PROGRAM
```

```
C       START COMPUTING TERMS OF THE SEQUENCE, AND TEST THE SEQUENCE FOR
C       CONVERGENCE.
C       NUMIT = NUMBER OF ITERATIONS = NUMBER OF TERMS OF THE SEQUENCE
C       CALCULATED.
C
                DO 260 NUMIT = 1 , 100
C
C       STORE X IN B
C
                DO 40 M = 1 , N2
 40             B( M ) = X( M )
                DO 200 K = 1 , N
                DO 100 I = 1 , N
                E(I)=0.0
                IF( NCASE .EQ. 2 ) GO TO 60
C
C       FORM B * A AND STORE IN E
C
                DO 50 J = 1 , I
                IJ=INDEX(I,J)
 50             E(I)=E(I)+B(IJ)*A(J,K)
                GO TO 75
 60             DO 65 J = 1 , I
                IJ=INDEX(I,J)
 65             E(I)=E(I)+B(IJ)*A(K,J)
 75             JJ = I + 1
                IF(JJ.GT.N)GO TO 100
                IF( NCASE .EQ. 2 ) GO TO 90
                DO 80 J = JJ , N
                IJ=INDEX(J,I)
 80             E(I)=E(I)+B(IJ)*A(J,K)
                GO TO 100
 90             DO 95 J = JJ , N
                IJ=INDEX(J,I)
 95             E(I)=E(I)+B(IJ)*A(K,J)
 100            CONTINUE
C
C       COMPUTE X = AT * B * A
C
                DO 150 IKT = K , N
                IJ=INDEX(IKT,K)
                X( IJ ) = 0.0
                IF( NCASE .EQ. 2 ) GO TO 130
                DO 120 J = 1 , N
 120            X(IJ ) = X( IJ ) + A( J , IKT ) * E( J )
                GO TO 150
 130            DO 140 J = 1 , N
 140            X(IJ ) = X( IJ ) + A( IKT , J ) * E( J )
 150            CONTINUE
 200            CONTINUE
C
C       ADD X TO B, THE PREVIOUS TERM OF THE SEQUENCE, TO GET THE NEXT
C       TERM OF THE SEQUENCE.
C       X = X + B
```

```
C
                DO 210 I = 1 , N2
 210            X( I ) = B( I ) + X( I )
C               WRITE( 6 , 905 )
C               WRITE(6,920)(X(I),I=1,N2)
C
C       COMPARE THIS ANSWER TO THE ONE OBTAINED IN THE PREVIOUS ITERATION
C       IF THE RELATIVE ERROR IS LESS THAN ERR, RETURN.
C       IF NOT GO ON TO THE NEXT ITERATION.
C
                SUM = 0.0
                DO 220 I = 1 , N
                II = INDEX( I , I )
 220            SUM = SUM + ABS( ( X(II ) - B( II ) ) / X( II ) )
                IF(SUM.LT.ERR*N) GO TO 290
C
C       IFTHE NUMBER OF ITERATIONS IS GREATER THAN ( IT ) , RETURN.
C
                IF( NUMIT .GT. IT ) GO TO 280
C
C
C       THE SEQUENCE HAS NOT CONVERGED.
C       SQUARE THE MATRIX A, AND GO BACK AND COMPUTE ANOTHER TERM IN THE
C       SEQUENCE.
C       A = A * A
C
                DO 230 I = 1 , N
                DO 230 J = 1 , N
                D( I , J ) = 0.0
                DO 230 K = 1 , N
 230            D( I , J ) = D( I , J ) + A( I , K ) * A( K , J )
                DO 240 I = 1 , N
                DO 240 J = 1 , N
 240            A( I , J ) = D( I , J )
 260            CONTINUE
 280            WRITE( 6 , 960 )
                GO TO 300
 290            WRITE( 6 , 970 ) NUMIT
C               WRITE( 6 , 920 ) ( X( I ) , I = 1 , N2 )
                RETURN
 295            WRITE( 6 , 980 )
 300            RETURN 12
C
C       FORMAT STATEMENTS
C
 900            FORMAT( 1H1 )
 905            FORMAT( //// )
 920            FORMAT(1X,3(E15.8,2X))
 960            FORMAT( //' STOP, LACK OF CONVERGENCE'
     1          ' NUMBER OF ITERATIONS IS GREATER THAN IT' // )
 970            FORMAT( ///' NUMIT = NUMBER OF ITERATIONS REQUIRED FOR'
     1          ' CONVERGENCE IS = ', I6 /// )
 980            FORMAT( // ' ERROR IN GJR- MATRIX INVERSION - OVERFLOW'/)
                END
```

APPENDIX C

Charge Routine for Bellcomm Univac 1108 Computer

Charges are computed as follows:

$$1 \text{ Charge} = \frac{\text{Core Sec}}{60} + \frac{\text{CPU Sec}}{15} + \frac{\text{I/O}}{425}$$

where

1 Core Sec = 8196 words occupied for 1 second

1 CPU Sec = use of Central Processing Unit for 1 second

1 I/O = 1 read or write request

REFERENCES

1. Kirkendall, N. I., "Digital Simulations of Random Crew Motion Disturbances," Bellcomm Memorandum for File B68-12107, December 31, 1968.

2. James, H. M. and others, Theory of Servomechanisms, Boston Technical Publishers, 1964, pp. 268, 278.

3. Bar-Itzhack, I. Y. and Hou, S. N., "Statistical Analysis of Nonstationary Structural Response Under Feedback Conditions," Bellcomm Technical Memorandum No. 70-2031-2, April 6, 1970.

4. Gantmacher, F. R., Theory of Matrices, Chelsea, 1959, Vol. 1, p. 225.

5. Smith, P. G., "Literature Survey on Numerical Solution of Matrix Equations," Bellcomm Memorandum for File B70-01040, January 23, 1970.

6. Smith, R. A., "Matrix Equation XA + BX = C," SIAM J. Appl. Math., Vol. 16, pp. 198-201 (January 1968).

7. Freested, W. C. and others, "The 'GASP' Computer Program - on Integrated Tool for Optimal Control and Filter Design," Proc. 1968 Joint Automatic Control Conf., pp. 198-202.

8. Flügge, W., Handbook of Engineering Mechanics, McGraw-Hill, 1962, Chapter 56.

9. Fath, A. F., "Computational Aspects of the Linear Optimal Regulator Problem," Proc. 1969 Joint Automatic Control Conf., pp. 44-49; See also IEEE Trans. Automatic Control, Vol. AC-14, pp. 547-550 (October 1969).