ADAPTING BORROWED COMPUTING SUBPROGRAMS FOR A SPECIFIC COMPUTING INSTALLATION

by 7654

THOMAS MICHAEL RAWSON

B. S., Kansas State University, 1965
M. S., Kansas State University, 1967

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Statistics and Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1970

Approved by:

K. E. Kemp
Major Professor

ACKNOWLEDGEMENT

The author wishes to acknowledge the seemingly endless assistance of Charles Walker, Assistant Professor of Statistics and Computer Science at Kansas State University. Although Professor Walker's name does not appear on any of the official credits connected with this project, the author is deeply indebted to him for his valuable help.

INTRODUCTION

A common misconception that continues to exist in the computing world is that if an installation does not have its own version of a computational program then the user need only consult some program library, make the appropriate selection, and the problem will be solved. The fallacy of this solution is usually understood by most professional computing center personnel, but the average user is probably unaware of the time, patience and understanding that it takes to convert a borrowed program into a smooth-running, accurate scientific tool. The purpose of this report is to point out all of the major difficulties encountered when an average computer user is unable to find a local program and does not desire to write one to solve a particular problem which he faces. This report will not concern itself with the theory involved in the calculations mentioned in the following pages.

METHOD

In undertaking this study the author chose a problem which purposely might appear to the reader as atypical of the average computing user's needs. The problem, the computation of eigenvalues for real and complex matrices is indeed not a simple one. This particular problem was chosen primarily because the associated mathematical theories made the author feel that the simplest way to solve the problem was to borrow a "canned" program. The Kansas State University Computing Center, henceforth called the Computing Center or the KSU Computing Center, did not have suitable subprograms to solve problems of this type. The advice from the local

computing center librarian was to consult the latest SHARE index. For the readers unfamiliar with SHARE, it is a worldwide cooperative organization of computing installations whose primary purpose is the sharing of "standardized" computing programs. Appendix I of this report contains a more extensive explanation of the SHARE organization. A review of the index and the SHARE Secretary Distributions (SSD's) revealed that three subprograms were available which would solve the problem which the author proposed. These three subprograms, SDA 3099, SDA 3219, and SDA 3441, were ordered by the local SHARE secretary. A period of three weeks elapsed between the time these subprograms were ordered and the time they were received. Two of them, SDA 3099 and SDA 3441, were in card form; the third was available only on microfilm since it was in such little demand. The decision to borrow another subprogram from a similar cooperative library program organization was made after concluding that SDA 3219 would be too difficult to adapt. The subprogram F4 UTEX MATSUB, also called EIVAL, was obtained as a substitute for SDA 3219. EIVAL was obtained from CO-OP, an organization similar to SHARE sponsored by CDC, the Control Data Corporation. Thus the problem was to make these three subprograms operational on the computing system available at Kansas State University.

Because of the diagnostic value and compilation speed of Waterloo FORTRAN (WATFOR), which is similar to IBM 360 FORTRAN IV, the programs were tested and made operational using that compiler. The programs were then compiled and executed on IBM 360 FORTRAN IV Level G for the purpose of comparison. All of the keypunching mentioned was done by the author because of the nature of this project. It was felt that this would make the project as realistic as possible. The following three sections of this

report will deal with the problems encountered in making these routines operational. The fourth section will describe comparison tests which were made for the purpose of selecting the best of the three borrowed routines. The fifth section will mention some of the logistical problems which the author encountered during the project. The final section will summarize the project and will contain concluding remarks.

SUBPROGRAM ONE: ALLMAT

The Arbitrary Matrix Eigensystem Solver, originally shortened to AMAT, but now known as ALLMAT, was developed by R. E. Funderlic and J. Rinzel. This subprogram was submitted to SHARE in March, 1966. The version now available from SHARE is the 1968 revision, designated SDA 3441-01. It was written in FORTRAN IV for compilation and execution on an IBM 7090. It is designed to calculate both eigenvalues and a set of eigenvectors for arbitrary real or complex matrices.

ALLMAT utilizes the QR algorithm in its calculations. The Weilandt inverse power method is used to calculate the eigenvectors.[1] A complete discussion of the theory and algorithm may be found in The Algebraic Eigenvalue Problem by J. H. Wilkinson. It does not contain other subroutines which were written specifically for the routine. ALLMAT has five arguments, A, EIGVAL, N, IA, and NCAL, which are defined as follows:

---

[1]R. E. Funderlic and J. Rinzel, "Eigenvalues and Eigenvectors of Complex Matrices" (Oak Ridge: Union Carbide Corporation, 1966), p. 3. (Mimeographed.)

1. A is the complex array which contains the input matrix upon entry to ALLMAT and contains the complex eigenvectors upon return. The maximum size allowed by ALLMAT is limited only by the needs of the user.
2. EIGVAL is a complex vector which contains the eigenvalues of the appropriate eigenvector upon return from ALLMAT.
3. N is the order of the input matrix A.
4. IA is the first dimension of the input matrix in the calling program.
5. NCAL is the number of eigenvalues with corresponding eigenvectors which ALLMAT returns to the calling program.

ALLMAT allows ten iterations per eigenvalue before termination occurs. In the case of equal eigenvalues, identical eigenvectors are returned even though linearly independent eigenvectors may exist.

The original ALLMAT source deck consisted of 234 cards which had been punched in Binary Coded Decimal (BCD) characters. The first task to be undertaken was the conversion from BCD to Extended Binary Coded Decimal Interchange Code (EBCDIC). This conversion was accomplished using an assembler language program available at the KSU Computing Center. Although this conversion was not absolutely necessary, it was thought that such a conversion at the start of the project could save later problems. The subprogram ALLMAT was a "called" program and contained no input-output provisions, so an input-output "calling" program had to be supplied by the user. ALLMAT itself was not well-documented, but accompanying literature allowed the construction of an input-output program. This input-output

was written in IBM 360 FORTRAN IV and added to ALLMAT prior to the first attempt at compilation.

The first compilation revealed several unexpected problems. The FORTRAN IV compiler used on the IBM 7090 allowed the first member of the vector dimensioned SHIFT(3) to be referenced by SHIFT. The WATFOR compiler would not allow this type of coding. As a result, all references to SHIFT had to be changed to SHIFT(1). The initialization of SHIFT(1) to zero necessitated the addition of two source cards. The first run also revealed that the original program used variables names SIN, COS, and INT. All of these are identifiers for IBM 360 FORTRAN IV library subroutines. The variable named SIN was changed to SIN1, COS was changed to COS1 and the variable INT was changed to JNT.

Another error was one which was characteristic of the WATFOR compiler. The library subroutines square root and conjugate function (CSQRT and CONJG) had to be declared complex at the beginning of the ALLMAT subprogram

A pair of complex matrices of order two were added as test data prior to the next run.[1] An error which resulted from not equating the subprogram argument IA to the first dimension of A in the calling program caused it to yield incorrect results. The correction of this problem proved successful and the routine ALLMAT performed satisfactorily for both sets of test data.

---

[1]R. Johnston, "Numerical Analysis Project Program Review, SDA 3441" (Ontario: Department of Highways, 1967), p. 8. (Mimeographed.)

The total time involved in making ALLMAT operational was approximately four hours of programming, re-programming, keypunching and re-keypunching. The total computing cost to this point was $3.01. However, the elapsed time since the first submission of the program was more than four days. Since the conversion from BCD to EBCDIC, 59 of the 265 source cards (22 per cent) required to execute ALLMAT were supplied by the user or required alteration.

SUBPROGRAM TWO: EIG4

The SHARE subprogram, NU EIG4, SDA 3099, was written by B. N. Parlett in April, 1963. The routine now being distributed was revised in June, 1963. It was programmed in FORTRAN II for use on an IBM 7090. The subprogram calculates the eigenvalues of complex matrices or real matrices in complex form.

EIG4 reduces the input matrix to Hessenberg form, evaluates the characteristic polynomial and its derivatives using an extension of Hyman's method, and computes the eigenvalues using a modification of Laguerre's method.[1] The main subprogram EIG4 makes use of two specially written FORTRAN subroutines, designated CXTRI and CXLAG. Two machine language subroutines were also included in SDA 3099. The arguments for subprogram EIG4 are not specially defined in the program review which SHARE includes with the routine. Neither are these arguments defined in the program documentation itself. However, the program user is able to define the

---

[1] B. N. Parlett, "Laguerre's Method Applied to the Matrix Eigenvalue Problem", Mathematics of Computation, XVIII January, 1964), 464.

arguments after careful reading of the program review. The arguments, A, N, M, RT, Z, are defined as follows:

1. A is the doubly-dimensioned input matrix in complex form. EIG4 will find the eigenvalues for an input matrix of order less than or equal to seventy.
2. N is the order of the input matrix.
3. M is the number of eigenvalues which EIG4 computes.
4. RT is a singly-dimensioned complex vector which will contain the calculated eigenvalues upon return from EIG4.
5. Z is a starting value for the iteration of the first eigenvalue.

Originally, EIG4 allowed fifteen iterations per eigenvalue before assuming non-convergence. As an important factor in the converging process, the EIG4 author stressed in the program summary that the selection of an adequate starting value, Z, is very important in the accuracy of the calculated eigenvalues.

The EIG4 deck supplied by SHARE was also punched in BCD characters. Since the KSU Computing Center has more EBCDIC keypunches, the default option on the WATFOR compiler specifies EBCDIC characters, and the line printer does not convert the BCD characters to EBCDIC which makes the listings hard to read, it was decided to convert the deck to EBCDIC. This conversion was accomplished using the previously mentioned routine. EIG4 did not include input provisions, so the input routine had to be supplied by the program user.

The original EIG4 deck contained 342 source cards. A total of 279 of these cards comprised the three FORTRAN subprograms previously mentioned. The remaining sixty-three cards were the assembler language subroutines.

After the subprogram was converted to EBCDIC several major problems occured. EIG4 was written in FORTRAN II, which handled complex arithmetic in a vastly different manner than IBM 360 FORTRAN IV. Instead of using explicit statements to declare variables to be complex, FORTRAN II treated a complex number as two parts of a singly-dimensioned vector. The letter I in the first column of a source card indicated to the compiler that complex arithmetic was to be performed in the expression which appeared on that card. This method of accomplishing complex arithmetic created many of the problems which follow.

The most obvious task was to create the appropriate complex declarations for each of the complex variables used in the three FORTRAN subprograms. The next job was the re-keypunching of the seventy-nine source cards containing the letter I in card column one. It was determined that the two machine language subroutines were included to process accumulator overflow conditions. The user decided to eliminate these two subroutines and their call statements since the WATFOR compiler contained its own floating point overflow routine.

The FORMAT statements supplied for output in EIG4 had to be revised because of the changes involved in the output of complex numbers. The associated output statements, written in the cumbersome FORTRAN II style, also had to be altered The square root function had to be declared complex (SQRTF was changed to CSQRT) in the appropriate subroutines. Statements containing FORTRAN II function names such as XMINOF, ABSF, MAX1F, MINOF had to be changed to the equivalent IBM 360 forms of MINO, ABS, MAX1, and MINO.

Certain statements which found the magnitude of the real part plus the magnitude of the imaginary part of a complex number had to be changed. These changes involved the introduction of the IBM 360 FORTRAN IV functions REAL and AIMAGS. After making these alterations, another attempt at compilation was made.

The next attempt detected another serious problem with EIG4. The FORTRAN II compiler used for the original version of EIG4 differed from our IBM 360 FORTRAN IV compiler in its testing of the parameters during the execution of DO statements. Values which would fall through and allow completion of the loop on the original version were not allowable on the IBM 360 FORTRAN IV compilers. The correction of this error necessitated the addition of several IF statements to check the DO parameters.

A pair of complex test matrices of order two were supplied prior to the next attempt at compilation. The compilation was successful but EIG4 failed to provide acceptable results. The results of the computations indicated that the iteration of many of the eigenvalues had been halted after the fifteenth attempt. The user was forced to alter the routine in such a manner to allow thirty iterations per eigenvalue. This modification produced results which were initially acceptable to the user.

The user spent approximately ten hours making EIG4 operational. This time was spent mainly on re-programming and keypunching and was spread over a period of two weeks. The operational version of EIG4 contained only 168 of the original source cards supplied by SHARE. Thus more than 49 per cent of the original source deck was either eliminated or modified by the user. More than 18 per cent of the operational version of EIG4

had to be programmed by the user. The total computing costs required to make EIG4 operational was $12.63.

SUBPROGRAM THREE: EIVAL

The subprogram known as F4 UTEX MATSUB or EIVAL was written in 1961, by L. H. Ehrlich. This subprogram was written in FORTRAN II for use on either a Control Data Corporation Model 1604 or 3600. It calculates all eigenvalues of a real or complex matrix, as well as the eigenvector for each eigenvalue. EIVAL utilizes the direct and inverse power methods and matrix deflation in its calculations.[1] The source deck consisted of one main subprogram. EIVAL consists of twelve arguments, M, IEG, IVEC, ALRS, ALIS, GBR, GBI, IDET, MIT, MITS, EP1, and EP2, which are defined as follows:

1. M is the order of the input matrix A. The order of A must be less than or equal to eighteen.
2. IEG is a dummy variable which will allow the eigenvalue iterants to be printed when initialized to one but will not allow printing when set to zero.
3. IVEC is a summy variable which when initialized to one will allow the subroutine to calculate a set of eigenvectors. When IVEC is set to zero EIVAL will bypass the calculation of eigenvectors.
4. ALRS is the real part of the starting value Alpha.

---

[1] L. H. Ehrlich, "F4 UTEX MATSUB Summary" (Austin: The University of Texas, 1961), p. 1. (Mimeographed.)

5. ALIS is the imaginary part of the starting value, Alpha.
6. GBR is the real part of beta.
7. GBI is the imaginary part of beta.
8. IDET is a dummy variable which will allow the subroutine to calculate $|C-\lambda I|$ when initialized to one. When IDET is set to zero these values are not calculated.
9. MIT is the maximum number of iterations allowed for the power method.
10. MITS is the maximum number of iterations allowed for the inverse method.
11. EP1 is the constant used to test the success of the first iteration method. EP1 is normally initialized to 10^{-4}.
12. EP2 is the constant used to test the success of the inverse power method. This constant is normally initialized to 10^{-14}.

It must be noted that unlike the two subprograms previously discussed, EIVAL does not pass the values of the input matrix to subprogram as arguments. The subprogram utilizes COMMON statements to pass these values. Another major difference is the manner in which the subprogram handles complex arithmetic. EIVAL treats the complex matrix as two separate and distinct matrices. Matrix AR is composed of the real portions of the complex numbers while matrix AI is composed of their imaginary components. This method of performing complex arithmetic created many problems which were difficult to visualize because of the physical separation of the two elements of the complex matrix.

The subprogram EIVAL consisted of 353 BCD source cards. As was the case with ALLMAT and EIG4, the original deck was converted to EBCDIC

characters prior to compilation at the cost of $1.24. A suitable input-output program consisting of thirty-seven source cards was added by the user prior to the first attempt at compilation.

This first attempt revealed several expected errors. All of the output statements provided in the original subprogram had to be revised. The associated FORMAT statements also had to be altered by the user. It was also determined that source cards containing the square root function and the modular function had to be changed to comply with IBM 360 FORTRAN IV. Thus SQRTF was changed to SQRT and XMODF became MOD.

Another attempt at compilation revealed additional errors in EIVAL. Many of the variables used by EIVAL were not initialized to zero by the original routine. The "background" is not set to zero on the IBM 360, so an initial value of zero cannot be assumed. This failure caused the variables to be undefined on our compiler and caused the generation of numerous error messages. The initialization of the necessary variables caused the addition of seven source cards. The next attempt at compilation revealed an error in the input program as well as the location of several keypunch errors.

The program author suggested the EP2, the constant used to determine the success of the convergence of the inverse power method, be initialized to 10^{-14}. Because of the lack of precision of the IBM 360 single precision floating point arithmetic, the user changed this constant to 10^{-7}. The same test data used for the initial testing of the other routines yielded satisfactory results with EIVAL.

A total of six hours of keypunching and re-programming was spent by the user in making EIVAL operational. Approximately 20 per cent of

the final version of EIVAL was added or altered by the user. The computing cost amounted to $6.33. A period of eight days elapsed between the conversion from BCD to EBCDIC and the successful execution of the program.

COMPARISON OF RESULTS

After the three subprograms had been compiled successfully, the user switched from WATFOR to IBM 360 FORTRAN IV Level G. The programs were modified slightly by adding the INTIME function to determine the comparative execution times of the subprograms. Test data were extremely difficult to obtain, especially complex matrix data. The eigenvalues calculated by all three programs were compared with the hand calculated results of the test data.

Real test matrices of order two, three, four, and eight were supplied by the user. It was found that ALLMAT calculated the eigenvectors and eigenvalues of the real matrices approximately 20 per cent faster than EIVAL. Using single-precision arithmetic, ALLMAT gave eigenvalues which agreed with the suggested results to at least the six significant digits. The EIVAL routine was not as accurate as ALLMAT, often yielding errors in the fifth significant digit. Since the built-in complex routines weren't used in EIVAL, the user was able to convert the routine to double precision. The resulting answers were accurate to at least seven significant digits.

The EIG4 program provided the least accurate eigenvalues. EIG4 was found to be of little value in calculating the eigenvalues of real symmetric matrices. The new limit of thirty iterations per eigenvalue

was not satisfactory in some cases. This limit was raised to fifty by the user in hopes of improving the routine's accuracy. The value of EPS in subroutine CXLAG was changed from 10^{-4} to 10^{-8}. These two changes yielded the desired effect but increased the execution time of the routine considerably. These changes increased the accuracy of EIG4 for both symmetric and non-symmetric real matrices. The results were generally accurate to six significant digits.

The user was not able to make a direct conversion to double precision arithmetic because of the lack of double precision functions comparable to REAL and AIMAGJ. EIVAL was easily converted because of the manner in which it handles complex arithmetic.

The user compared the speed of the single precision form of EIVAL with that of EIG4 for the calculation of eigenvalues for the real test matrices. The differences in execution time depended on the particular input matrix. However, EIG4 was found to be considerably slower than EIVAL on all of the real matrices tested.

Complex test data were much more difficult to obtain. Several complex matrices of order two were used to compare the three routines. It was determined that ALLMAT was approximately 30 per cent faster in the calculation of eigenvectors and eigenvalues than the single precision version of EIVAL. The accuracy obtained for the eigenvalues and eigenvectors was comparable to that obtained for the real test matrices. The EIG4 routine did not give as accurate results for the symmetric complex matrices. Table I compares the accuracy and speed obtained for selected test matrices.

LOGISTICAL PROBLEMS

The problems mentioned in this section of the report are illustrative of the barriers encountered by an average computing center user who does not make his permanent residence in the center. These remarks are not to be taken as malicious, destructive criticisms of our local computing facility. The problems mentioned in this section are particular barriers encountered by the user. The reader should realize that the author did not intend to imply that these problems are universal in nature.

The granting of an unsponsored account number for this project was the first major logistical problem encountered by the user. This application resulted in the granting of $1250 of "free" computing time, required the signatures of several people, and caused a delay of three days.

The next problem consisted of scanning the volumes of SHARE minutes in search of clues to applicable subprograms. Computing Center rules stated that these volumes were not to be removed from the Computing Center Library. The local SHARE secretary was most helpful in the ordering of the necessary subprograms. The delay of approximately three weeks was unavoidable.

The original EIVAL deck was not in good enough shape to be reproduced on the IBM card reproducer owned by the center. Having decided the best way to reproduce the deck was to write a program, the user accidentally found out that this center had a "reproduction" program. The program operated satisfactorily and the routine was reproduced.

TABLE I

Comparison of Accuracy and Speed
for Selected Test Data

TEST MATRIX	HAND-CALCULATED EIGENVALUES	ALLMAT EIGENVALUES	EIG4 EIGENVALUES	EIVAL EIGENVALUES (S.P.)
5 2 1 2 4 2 1 2 3	7.504664 3.135359 1.359977	7.504664 3.135353 1.359976	7.504664 3.135358 1.359976	7.504660 3.135360 1.359977
	Time 1*		5.77	0.97
	Time 2	0.96		1.25
1 0 0 0 1 0 0 0 1	1.000000 1.000000 1.000000	1.000000 1.000000 1.000000	1.000000 1.000000 1.000000	1.000000 1.000000 1.000000
	Time 1		0.43	0.65
	Time 2	0.62		0.95
10 9 7 5 9 10 8 6 7 8 10 7 5 6 7 5	30.288683 3.858057 0.843107 0.010150	30.288559 3.858056 0.843102 0.010149	30.288666 3.858054 0.854598 0.010152	30.288670 3.858060 0.843121 0.010149
	Time 1		16.64	1.60
	Time 2	1.68		2.03
6 4 4 1 4 6 1 4 4 1 6 4 1 4 4 6	15.000000 5.000000 5.000000 -1.000000	15.999977 5.000007 5.000000 -0.999998	15.000003 5.000000 5.000000 -1.000000	14.999999 5.000010 5.000000 -1.000000
	Time 1		1.90	1.38
	Time 2	1.37		1.73
3 1+1 -1+1 2	2.5 + 1.322876i 2.5 - 1.422876i	2.5 + 1.322875i 2.5 - 1.322875i	2.5 + 1.322880i 2.5 - 1.322876i	2.5 + 1.353790i 2.5 - 1.052736i
	Time 1		0.42	0.63
	Time 2	0.40		0.76

*Time 1 indicates the time in seconds required to calculate the eigenvalues only.
Time 2 indicates the time in seconds required to calculate both eigenvalues and a set of eigenvectors.

The machine language program which the computing center maintains for the conversion from BCD to EBCDIC worked perfectly for the user. The card interpreter owned by the center, an ancient model from IBM, is almost worthless.

The computing center hours were found to be adequate by this user. The attitudes of the dispatchers seemed to improve during the test period to a point where this user felt that they almost understood the problems of the user. The turn-around time during the testing of the routines ranged from as little as two hours to as much as seven hours. These routines were two-minute batch WATFOR jobs. During the majority of the testing phase of the project the user found that the best time schedule was program submission early in the morning, mid-afternoon and late evening. This schedule allowing three turn-arounds a day was maintained during most of the project. The turn-around time for the five-minute FORTRAN IV Level G comparison runs was only slightly longer than for the batch WATFOR jobs. It should be mentioned that most of the computing was done in late August and early September, a period when the computing center processes an abnormally high percentage of long jobs.

The university library also presented many problems to the author. The book The Algebraic Eigenvalue Problem by J. A. Wilkinson was listed as available in the library but the book could not be located by the library staff. An attempt to obtain the book on inter-library loan from the University of Kansas was unsuccessful. The user was unable to locate many of the references mentioned in many of the mimeographed reports.

CONCLUSION

The adaptation of a borrowed or "canned" program for use at a particular computing installation may be a very tedious, time-consuming process. The particular problem chosen by this user was more difficult than the typical computing user would face, however, the fact still remains that when a user needs a particular program to solve a given problem then he has no other choice than to write his own program or try to use someone else's program. In this particular case, the adaptation of someone else's program seemed to be the best solution. The variance in FORTRAN compilers and in the configuration of computing installations makes the establishment of an all-purpose program library nearly impossible. The problems which arose in attempting to convert one version of FORTRAN to another have been discussed in this report. The difficulties in converting from one programming language to another are undoubtedly even more time-consuming. Since the user did choose three subprograms which were not specifically written in IBM FORTRAN IV Level G for execution on an IBM 360/50, the problems mentioned in this report are not identical to those faced by all users who wish to borrow programs.

This user was able to make several conclusions based on this project. The user should make every attempt to borrow a routine which has been written in as nearly the same language version as he plans to use. The user should seek copies of program reviews prior to making his choice. He should be prepared to spend considerable time in obtaining the routine and making it operational. The user should be aware of particular aspects of the

computing hardware which he will utilize. Such aspects include "background" variable initialization and floating-point accuracy. He must either find reliable test data or generate his own.

Two specific improvements are suggested to cooperative program libraries such as SHARE. Improvements in the program documentation and the accompanying program summaries would greatly facilitate the use of borrowed routines. At least one set of test data with accurate results should accompany all library routines. These two improvements would eliminate many of the problems which the author has mentioned in this report.

BIBLIOGRAPHY

BIBLIOGRAPHY

Anderson, Decima M. Computer Programming. New York: Appleton-Century Crofts, 1966.

Ashton, T. "Eigenvalues and Eigenvectors of a Real Matrix." Ontario: Department of Highways, 1966. Mimeographed.)

Carnahan, Brice, H. A. Luther, and James O. Wilkes. Applied Numerical Methods. New York: John Wiley and Sons, Inc., 1964.

Conte, S. D. Elementary Numerical Analysis. New York: McGraw-Hill Book Company, 1965.

Ehrlich, Louis W. "F4 UTEX MATSUB Summary." Austin: The University of Texas, 1961. (Mimeographed.)

Funderlic, R. E. and J. Rinzel. "SDA 3441 Program Package Submittal." Oak Ridge: Union Carbide Corporation, 1966. (Mimeographed.)

Galligani, Ilio. "Review of the SHARE Program SDA 3219." Varese, Italy: EURATOM, 1967. (Mimeographed.)

Galligani, Ilio. "Review of the SHARE Program SDA 3441." Varese, Italy: EURATOM, 1968. (Mimeographed.)

Jennings, Walter. First Course in Numerical Methods. New York: The Macmillan Company, 1964.

Johnston, R. "Numerical Analysis Project Program Review." Ontario: Department of Highways, 1967. (Mimeographed.)

Miller, Robert W. "SHARE Numerical Analysis Project Evaluation of EIG4-SDA 3099." Bell Helicopter Company, 1967. (Mimeographed.)

Parlett, B. N. "NU EIG4." New York: New York University, 1963. (Mimeographed.)

Parlett, B. N. "Laquerre's Method Applied to the Matrix Eigenvalue Problem", Mathematics of Computation, XVIII (January, 1964), 464-75.

Pennington, Ralph H. Introductory Computer Methods and Numerical Analysis. New York: The Macmillan Company, 1965.

Ralston, Anthony and Herbert S. Wilf. Mathematical Methods for Digital Computers. New York: John Wiley and Sons, Inc., 1967.

APPENDIX A

SHARE

SHARE was founded in August, 1955, in a seminar held by IBM for Rand, Lockheed and North American to discuss the standardization of computing programs for the IBM 704. Seventeen corporations and universities made up the original SHARE organization. The name SHARE was chosen as representing the common aims of the organization. Today a substantial portion of the major users of the larger IBM computing equipment is represented in SHARE. The benefits of the organization SSD's include written documentary information (SSD's) as well as informal conversation at meetings. SHARE membership is entirely voluntary, members are in no way forced to contribute or participate although such failure defeats the purpose of the organization. To be eligible for membership the system must be composed of IBM 704, 709, 7044, 7090, 7094, or 360/50 or higher. Other systems may be approved for membership by the SHARE Executive Board. SHARE membership now numbers several hundred corporations and universities from throughout the world.

```
C
C                    SUBPROGRAM ONE: SDA 3441, ALLMAT
C
C
C
C
C
C
C INPUT-OUTPUT PROGRAM
C
      COMPLEX A(9,9),EIGVAL(9)
  999 FORMAT(2I2)
  998 FORMAT(5(F10.6,F10.6))
  997 FORMAT(1H1,'THE INPUT MATRIX WAS AS FOLLOWS:')
  996 FORMAT(1H ,4((F16.8,F12.8),' I'))
  995 FORMAT(1H-,'THE COMPUTED EIGENVALUES ARE:')
  994 FORMAT(1H-,'THE NUMBER OF EIGENVALUES WITH CORRESPONDING EIGENVECT
     1ORS IS:',5X,I3)
  993 FORMAT(1H-,'THE COMPUTED EIGENVECTOR IS:')
  992 FORMAT(1H-,'EXECUTION TIME =',5X,F10.6)
    4 CALL INTIME(ITIME)
      READ(1,999,END=5) N,IA
      DO 1 I=1,N
    1 READ(1,998) (A(I,J),J=1,N)
      NCAL=0
      WRITE(3,997)
      DO 2 I=1,N
    2 WRITE(3,996) (A(I,J),J=1,N)
      CALL ALLMAT(A,EIGVAL,N,IA,NCAL)
      WRITE(3,995)
      WRITE(3,996) (EIGVAL(I),I=1,N)
      WRITE(3,993)
      DO 3 I=1,N
    3 WRITE(3,996) (A(I,J),J=1,N)
      WRITE(3,994) NCAL
      CALL INTIME(ITIME1)
      TIME=(ITIME1-ITIME)/100.
      WRITE(3,992) TIME
      GO TO 4
    5 CONTINUE
      STOP
      END
C
C
C
C
C
      SUBROUTINE ALLMAT(A,LAMBDA,M,IA,NCAL)
C
C PROG.AUTHORS JOHN RINZEL,R.E.FUNDERLIC,UNION CARBIDE CORP.
C NUCLEAR DIVISION,CENTRAL DATA PROCESSING FACILITY,
C OAK RIDGE TENNESSEE
C
      COMPLEX A(IA,1),H(30,30),HL(30,30),LAMBDA(1),VECT(30),
     1MULT(30),SHIFT(3),TEMP,        TEMP1,TEMP2,SIN1,COS1,CONJG,CSQRT
      LOGICAL INTH(30),TWICE
      INTEGER JNT(30),R,RP1,RP2
      N=M
      NCAL=N
```

```
      IF(N.NE.1)GO TO 1
      LAMBDA(1)=A(1,1)
      A(1,1)=1.
      GO TO 58
    1 ICOUNT=0
      DO 73 I=1,3
   73 SHIFT(I)=0.
      IF(N.NE.2)GO TO 4
    2 TEMP=(A(1,1)+A(2,2)+CSQRT((A(1,1)+A(2,2))**2-
     14.*(A(2,2)*A(1,1)-A(2,1)*A(1,2))))/2.
      IF(REAL(TEMP).NE.0..OR.AIMAG(TEMP).NE.0.)GO TO 3
      LAMBDA(M)=SHIFT(1)
      LAMBDA(M-1)=A(1,1)+A(2,2)+SHIFT(1)
      GO TO 38
    3 LAMBDA(M)=TEMP+SHIFT(1)
      LAMBDA(M-1)=(A(2,2)*A(1,1)-A(2,1)*A(1,2))/(LAMBDA(M)-SHIFT(1))+
     1SHIFT(1)
      GO TO 38
C
C     REDUCE MATRIX A TO HESSENBERG FORM
C
    4 NM2=N-2
      DO 15 R=1,NM2
      RP1=R+1
      RP2=R+2
      ABIG=0.
      JNT(R)=RP1
      DO 5 I=RP1,N
      ABSSQ=REAL(A(I,R))**2+AIMAG(A(I,R))**2
      IF(ABSSQ.LE.ABIG)GO TO 5
      JNT(R)=I
      ABIG=ABSSQ
    5 CONTINUE
      INTER=JNT(R)
      IF(ABIG.EQ.0.)GO TO 15
      IF(INTER.EQ.RP1)GO TO 8
      DO 6 I=R,N
      TEMP=A(RP1,I)
      A(RP1,I)=A(INTER,I)
    6 A(INTER,I)=TEMP
      DO 7 I=1,N
      TEMP=A(I,RP1)
      A(I,RP1)=A(I,INTER)
    7 A(I,INTER)=TEMP
    8 DO 9 I=RP2,N
      MULT(I)=A(I,R)/A(RP1,R)
    9 A(I,R)=MULT(I)
      DO 11 I=1,RP1
      TEMP=0.
      DO 10 J=RP2,N
   10 TEMP=TEMP+A(I,J)*MULT(J)
   11 A(I,RP1)=A(I,RP1)+TEMP
      DO 13 I=RP2,N
      TEMP=0.
      DO 12 J=RP2,N
   12 TEMP=TEMP+A(I,J)*MULT(J)
   13 A(I,RP1)=A(I,RP1)+TEMP-MULT(I)*A(RP1,RP1)
      DO 14 I=RP2,N
      DO 14 J=RP2,N
   14 A(I,J)=A(I,J)-MULT(I)*A(RP1,J)
```

```
   15 CONTINUE
C
C      CALCULATE EPSILON
C
       EPS=0.
       DO 16 I=1,N
   16  EPS=EPS+CABS(A(1,I))
       DO 18 I=2,N
       SUM=0.
       IM1=I-1
       DO 17 J=IM1,N
   17  SUM=SUM+CABS(A(I,J))
   18  IF(SUM.GT.EPS)EPS=SUM
       EPS=SQRT(FLOAT(N))*EPS*1.E-12
       IF(EPS.EQ.0.)EPS=1.E-12
       DO 19 I=1,N
       DO 19 J=1,N
   19  H(I,J)=A(I,J)
   20  IF(N.NE.1)GO TO 21
       LAMBDA(M)=A(1,1)+SHIFT(1)
       GO TO 38
   21  IF(N.EQ.2)GO TO 2
   22  MN1=M-N+1
       IF(REAL(A(N,N)).NE.0..OR.AIMAG(A(N,N)).NE.0.)
      1 IF(ABS(REAL(A(N,N-1)/A(N,N)))+ABS(AIMAG(A(N,N-1)/A(N,N)))-1.E-9)
      2 24,24,23
   23  IF(ABS(REAL(A(N,N-1)))+ABS(AIMAG(A(N,N-1))).GE.EPS)GO TO 25
   24  LAMBDA(MN1)=A(N,N)+SHIFT(1)
       ICOUNT=0
       N=N-1
       GO TO 21
C
C      DETERMINE SHIFT
C
   25  SHIFT(2)=(A(N-1,N-1)+A(N,N)+CSQRT((A(N-1,N-1)+A(N,N))**2
      1 -4.*(A(N,N)*A(N-1,N-1)-A(N,N-1)*A(N-1,N))))/2.
       IF(REAL(SHIFT(2)).NE.0..OR.AIMAG(SHIFT(2)).NE.0.)GO TO 26
       SHIFT(3)=A(N-1,N-1)+A(N,N)
       GO TO 27
   26  SHIFT(3)=(A(N,N)*A(N-1,N-1)-A(N,N-1)*A(N-1,N))/SHIFT(2)
   27  IF(CABS(SHIFT(2)-A(N,N)).LT.CABS(SHIFT(3)-A(N,N)))GO TO 28
       INDEX=3
       GO TO 29
   28  INDEX=2
   29  IF(CABS(A(N-1,N-2)).GE.EPS)GO TO 30
       LAMBDA(MN1)=SHIFT(2)+SHIFT(1)
       LAMBDA(MN1+1)=SHIFT(3)+SHIFT(1)
       ICOUNT=0
       N=N-2
       GO TO 20
   30  SHIFT(1)=SHIFT(1)+SHIFT(INDEX)
       DO 31 I=1,N
   31  A(I,I)=A(I,I)-SHIFT(INDEX)
C
C      PERFORM GIVENS ROTATIONS, QR ITERATES
C
       IF(ICOUNT.LE.10)GO TO 32
       NCAL=M-N
       GO TO 38
   32  NM1=N-1
```

```
      TEMP1=A(1,1)
      TEMP2=A(2,1)
      DO 37 R=1,NM1
      RP1=R+1
      RHO=SQRT(REAL(TEMP1)**2+AIMAG(TEMP1)**2+
     1 REAL(TEMP2)**2+AIMAG(TEMP2)**2)
      IF(RHO.NE.0.) GO TO 33
      TEMP1=A(RP1,RP1)
      TEMP2=A(R+2,RP1)
      GO TO 37
   33 COS1=TEMP1/RHO
      SIN1=TEMP2/RHO
      INDEX=MAXO(R-1,1)
      DO 34 I=INDEX,N
      TEMP=CONJG(COS1)*A(R,I)+CONJG(SIN1)*A(RP1,I)
      A(RP1,I)=-SIN1*A(R,I)+COS1*A(RP1,I)
   34 A(R,I)=TEMP
      TEMP1=A(RP1,RP1)
      TEMP2=A(R+2,R+1)
      DO 35 I=1,R
      TEMP=COS1*A(I,R)+SIN1*A(I,RP1)
      A(I,RP1)=-CONJG(SIN1)*A(I,R)+CONJG(COS1)*A(I,RP1)
   35 A(I,R)=TEMP
      INDEX=MINO(R+2,N)
      DO 36 I=RP1,INDEX
      A(I,R)=SIN1*A(I,RP1)
   36 A(I,RP1)=CONJG(COS1)*A(I,RP1)
   37 CONTINUE
      ICOUNT=ICOUNT+1
      GO TO 22
C
C     CALCULATE VECTORS
C
   38 IF(NCAL.EQ.0)GO TO 58
      N=M
      NM1=N-1
      IF(N.NE.2)GO TO 39
      EPS=AMAX1(CABS(LAMBDA(1)),CABS(LAMBDA(2)))*1.E-8
      IF(EPS.EQ.0.)EPS=1.E-12
      H(1,1)=A(1,1)
      H(1,2)=A(1,2)
      H(2,1)=A(2,1)
      H(2,2)=A(2,2)
   39 DO 57 L=1,NCAL
      DO 41 I=1,N
      DO 40 J=1,N
   40 HL(I,J)=H(I,J)
   41 HL(I,I)=HL(I,I)-LAMBDA(L)
      DO 45 I=1,NM1
      MULT(I)=0.
      INTH(I)=.FALSE.
      IP1=I+1
      IF(CABS(HL(I+1,I)).LE.CABS(HL(I,I)))GO TO 43
      INTH(I)=.TRUE.
      DO 42 J=I,N
      TEMP=HL(I+1,J)
      HL(I+1,J)=HL(I,J)
   42 HL(I,J)=TEMP
   43 IF(REAL(HL(I,I)).EQ.0..AND.AIMAG(HL(I,I)).EQ.0.)GO TO 45
      MULT(I)=-HL(I+1,I)/HL(I,I)
```

```
      DO 44 J=IP1,N
   44 HL(I+1,J)=HL(I+1,J)+MULT(I)*HL(I,J)
   45 CONTINUE
      DO 46 I=1,N
   46 VECT(I)=1.
      TWICE=.FALSE.
   47 IF(REAL(HL(N,N)).EQ.0..AND.AIMAG(HL(N,N)).EQ.0.)HL(N,N)=EPS
      VECT(N)=VECT(N)/HL(N,N)
      DO 49 I=1,NM1
      K=N-I
      DO 48 J=K,NM1
   48 VECT(K)=VECT(K)-HL(K,J+1)*VECT(J+1)
      IF(REAL(HL(K,K)).EQ.0..AND.AIMAG(HL(K,K)).EQ.0.)HL(K,K)=EPS
   49 VECT(K)=VECT(K)/HL(K,K)
      BIG=0.
      DO 50 I=1,N
      SUM=ABS(REAL(VECT(I)))+ABS(AIMAG(VECT(I)))
   50 IF(SUM.GT.BIG)BIG=SUM
      DO 51 I=1,N
   51 VECT(I)=VECT(I)/BIG
      IF(TWICE)GO TO 53
      DO 52 I=1,NM1
      IF(.NOT.INTH(I))GO TO 52
      TEMP=VECT(I)
      VECT(I)=VECT(I+1)
      VECT(I+1)=TEMP
   52 VECT(I+1)=VECT(I+1)+MULT(I)*VECT(I)
      TWICE=.TRUE.
      GO TO 47
   53 IF(N.EQ.2)GO TO 56
      NM2=N-2
      DO 55 I=1,NM2
      N1I=N-1-I
      NI1=N-I+1
      DO 54 J=NI1,N
   54 VECT(J)=H(J,N1I)*VECT(N1I+1)+VECT(J)
      INDEX=JNT(N1I)
      TEMP=VECT(N1I+1)
      VECT(N1I+1)=VECT(INDEX)
   55 VECT(INDEX)=TEMP
   56 DO 57 I=1,N
   57 A(I,L)=VECT(I)
   58 RETURN
      END
```

APPENDIX B

```
C
C                    SUBPROGRAM TWO: SDA 3099, EIG4
C
C
C
C
C
C
C INPUT PROGRAM
C
      COMPLEX A(70,70),RT(70),Z
  999 FORMAT(2I2)
  998 FORMAT(5(F10.6,F10.6))
  997 FORMAT(1H1,'ECHO CHECK OF INPUT MATRIX')
  996 FORMAT(1H ,'EXECUTION TIME =',10X,F10.6)
    4 CALL INTIME(ITIME)
      READ(1,999,END=5) N,M
      DO 1 I=1,N
    1 READ(1,998) (A(I,J),J=1,N)
      WRITE(3,997)
      DO 2 I=1,N
    2 WRITE(3,998) (A(I,J),J=1,N)
      DO 3 I=1,70
    3 RT(I)=(0.,0.)
      Z=(1.0E+36,0.)
      CALL EIG4(A,N,M,RT,Z)
      CALL INTIME(ITIME1)
      TIME=(ITIME1-ITIME)/100.
      WRITE(3,996) TIME
      GO TO 4
    5 CONTINUE
      STOP
      END
C
C
C
C
C
C
      SUBROUTINE EIG4(A,N,M,RT,Z)
      COMPLEX A(70,70),TRACE,RT(70),X,E,Z,CSQRT
      DIMENSION INT(70)
      DO 101 I=1,70
  101 INT(I)=0
      TRACE=A(1,1)
      DO 10 I=2,N
   10 TRACE=TRACE+A(I,I)
      WRITE(3,5) TRACE
      CALL CXTRI(A,1.E-8,N,INT)
      TRACE=A(1,1)
      DO 11 I=2,N
   11 TRACE=TRACE+A(I,I)
      WRITE(3,6) TRACE
      NU=0
      NV=0
   13 IF(NV-N)14,12,14
   14 NV=NV+1
      NU=NV
   16 IF(INT(NV))15,17,15
   15 NV=NV+1
```

```
      GO TO 16
 17   IF(NV-NU)19,18,19
 18   RT(NU)=A(NU,NU)
      X=RT(NU)
      WRITE(3,7) X
      GO TO 13
 19   IF(NV-NU-1)20,21,20
 20   NP= MINO(M,NV)
      CALL CXLAG(A,1.E-8,NP,NU,NV,RT,Z)
      GO TO 13
 21   R=(.5,.0)*(A(NU,NU)+A(NV,NV))
      E=R**2-A(NU,NU)*A(NV,NV)+A(NU,NV)*A(NV,NU)
      S=CSQRT(E)
      RT(NU)=R+S
      RT(NV)=R-S
      X=RT(NU)
      E=RT(NV)
      WRITE(3,7) X,E
      GO TO 13
 12   X=(.0,.0)
      DO 24 J=1,M
 24   X=X+RT(J)
      WRITE(3,8)X
      RETURN
  5   FORMAT(1H-,30X,'TRACE OF GIVEN MATRIX =',(F16.8,F16.8))
  6   FORMAT(1H-,27X,'TRACE OF HESSENBERG FORM =',(F16.8,F16.8))
  7   FORMAT(1H-,'EIGENVALUE',12X,(F16.8,F16.8))
  8   FORMAT(1H-,33X,'SUM OF EIGENVALUES',(F16.8,F16.8))
      END
C
C
C
C
C
      SUBROUTINE CXTRI(A,EPS,N,INT)
      COMPLEX A(70,70),C,U
      DIMENSION INT(70)
      R=0.
      N1=N-1
      N2=N-2
      DO 21 J=1,N1
      J1=J+1
      J2=J+2
      L=J1
      NJ1=N-J1
      C=A(J,J1)
      S=ABS(REAL(C))+ABS(AIMAG(C))
      IF(NJ1 .LE. 0) GO TO 15
      IF(J2 .GT. N) GO TO 312
 6    DO 12 K=J2,N
      C=A(J,K)
      T=ABS(REAL(C))+ABS(AIMAG(C))
      IF(T-S)12,12,11
 11   L=K
      S=T
 12   CONTINUE
 312  IF( L .EQ. J1) GO TO 15
 13   DO 131 K=1,N
      C=A(K,J+1)
      A(K,J+1)=A(K,L)
```

```
 131   A(K,L)=C
 14    DO 141 K=1,N
       C=A(J+1,K)
       A(J+1,K)=A(L,K)
 141   A(L,K)=C
       R=0.
 15    DO 151 K=1,J
       C=A(J,K)
       T=ABS(REAL(C))+ABS(AIMAG(C))
 151   R=AMAX1(R,T)
       IF( S .GT. EPS*R) GO TO 17
 16    L=0
       NJ1=0
       GO TO 181
 17    C=A(J,J+1)
       IF( J2 .GT. N) GO TO 181
       DO 18 K=J2,N
 18    A(J,K)=A(J,K)/C
 181   DO 20 I=1,N
       M=MINO(J,I-2)
       U=(0.,0.)
       IF( NJ1 .LE. 0) GO TO 19
       IF(J2 .GT. N) GO TO 19
 7     DO 8 K=J2,N
 8     U=U+A(K,I)*A(J,K)
 19    IF( M .LE. 0) GO TO 20
 9     DO 10 K=1,M
 10    U=U-A(K,I)*A(J+1,K+1)
 20    A(J+1,I)=A(J+1,I)+U
 21    INT(J)=L
 22    INT(N)=0
       RETURN
        END
C
C
C
C
C
       SUBROUTINE CXLAG(A,EPS,N1,NU,N,RT,Z)
       COMPLEX A(70,70),P(3,71),R,SPUR1,SPUR2,T,S1,S2,D,  F1,Z,
      1B1,B2,B3,Q1,Q2,E,DELZ,RT(70),CSQRT
       WRITE(3,1)
       BL1=1.
       NUQ=NU-1
       LLY=0
       DELOLD=1.
       ROLD=1.
       SUM1=0.
       SUM2=0.
       A(N,N+1)=(0.,0.)
       ZZ=0.
       X=0.
       P(1,N+1)=(0.,0.)
       P(2,N+1)=(0.,0.)
       P(3,N+1)=(0.,0.)
       P(1,NU)=(1.,.0)
       P(2,NU)=(.0,.0)
       P(3,NU)=(.0,.0)
       NU1=NU+1
       CUP=0.
```

```
      IF(NU1 .GT. N) GO TO 123
      DO 11 J=NU1,N
      R=A(J-1,J)
 11   CUP=CUP+ABS(REAL(R))+ABS(AIMAG(R))
 123  CUP=CUP/FLOAT(N-NU)
      CAP=0.
C
C     FIND THE TRACE OF A AND A**2
C
      SPUR1=A(NU,NU)
      SPUR2=SPUR1**2
      IF(NU1 .GT. N) GO TO 124
      DO 13 J=NU1,N
      T=A(J,J)
      SPUR1=SPUR1+T
 13   SPUR2=SPUR2+T**2+(2.,0.)*A(J-1,J)*A(J,J-1)
 124  S1=-SPUR1
      S2=SPUR2
C
C     INITIAL ITERATE (EITHER GIVEN OR FROM INFINITY)
C
 131  IF(REAL(Z) -1.E+35)23,14,14
 14   F1=N-NUQ
      IF(ABS(REAL(S1))+ABS(AIMAG(S1))+ABS(REAL(S2))+ABS(AIMAG(S2))-
     11.E-6*CAP)15,15,16
 15   Z=CUP
      GO TO 23
 16   D=(F1-(1.,0.))*(F1*S2-S1**2)
      Z=(CSQRT(D)-S1)/F1
C
C     EVALUATE POLYNOMIAL AND DERIVATIVES
C
 23   P(1,NU)=1.0
      IF (LLY+NUQ-NU) 24,24,235
 24   P(1,NU)=1.E-20
      IF(NU .GT. N  ) GO TO 331
 235  DO 33 K=NU,N
      T=-A(K,K+1)
 25   DO 33 L=1,3
      S=FLOAT(MOD(L-1,3))
      IF(L-1 .NE. 0) GO TO 251
      R=-(Z*P(L,K))
      GO TO 252
 251  R=-(Z*P(L,K)+S*P(L-1,K))
      IF( NU .GT. K) GO TO 125
 252  DO 28 J=NU,K
 28   R=R+P(L,J)*A(K,J)
 125  IF(X)29,34,29
 29   X=0
      P(1,NU)=P(1,NU)*1.E-15
      IF(REAL(P(1,NU)))30,30,235
 30   F=.9*FLOAT (K-NU)/FLOAT (N-NU+1)
      WRITE(3,4) Z
      Z=F*Z
      P(1,NU)=(1.,0.)
      GO TO 235
 34   IF (N-K) 31,31,32
 31   P(L,K+1)=R
      GO TO 33
 32   P(L,K+1)=R/T
```

```
 33   CONTINUE
 331  CONTINUE
C
C     SCALE DOWN THESE VALUES IF REQUIRED TO AVOID OVERFLOW
C
 35   B1=P(1,N+1)
      B2=P(2,N+1)
      B3=P(1,N+1)
      G1=ABS(REAL(B1))+ABS(AIMAG(B1))
      G2=ABS(REAL(B2))+ABS(AIMAG(B2))
      G3=ABS(REAL(B3))+ABS(AIMAG(B3))
      S=AMAX1(G1,G2)
      S=AMAX1(S,G3)
      IF(S - 1.E+19) 43,43,36
 36   B1=B1/(1.E+19,0.)
      B2=B2/(1.E+19,0.)
      WRITE(3,2) Z,G1,G2,G3
      B3=B3/(1.E+19,0.)
C
C     REMOVE EFFECT OF KNOWN ROOTS FROM S1,S2 THE LOG DERIVATIVES
C
 43   Q1=(.0,.0)
      Q2=(.0,.0)
      IF(NUQ-NU)19,21,21
 21   DO 44 J=NU,NUQ
      D=(1.,0.)/(RT(J)-Z)
      Q1=Q1+D
 44   Q2=Q2+D**2
      IF (G1) 41,41,19
 19   S1=Q1+B2/B1
      S2 = (B2/B1)**2 - B3/B1 - Q2
C
C     FIND NEXT ITERATE
C
      LLY=LLY+1
      IF(1.E+7-ZZ*(ABS(REAL(S1))+ABS(AIMAG(S1))))41,41,42
 41   MARK=1
      GO TO 100
 42   G=N-NUQ
 50   IF(BL1)65,65,66
 65   H=.5*(G-2.)
      GO TO 67
 66   H=G-1.
 67   D=H*(G*S2-S1**2)
      E=CSQRT(D)
      IF(REAL(S1)*REAL(E)+AIMAG(S1)*AIMAG(E))55,56,56
 55   E=-E
 56   DELZ=-G/(S1+E)
      Z=Z+DELZ
      DELNEW=ABS(REAL(DELZ))+ABS(AIMAG(DELZ))
      RNEW=DELNEW/DELOLD
      ZZ=ABS(REAL(Z))+ABS(AIMAG(Z))
C
C     TEST FOR CYCLING AT 57 AND FOR LINEAR CONVERGENCE AT 571
C
      IF(LLY-3)62,62,57
 57   IF(DELNEW-MAX1 (3.*DELOLD,.5*ZZ))571,571,570
 570  IF(BL1)571,571,572
 572  DELOLD=CAP+1.0
      ROLD=3.
```

```
      IF(LLY-15)14,14,100
  571 IF(RNEW-.7*ROLD)62,58,58
  58  MARK=3
      IF(DELNEW-.1*EPS*MAX1 (ZZ,.01*CAP)) 70,59,59
  59  IF(BL1)61,61,60
  60  Z=Z-DELZ
      BL1=0.
      GO TO 50
  61  BL1=1.
      GO TO 63
C
C     TEST FOR AN EIGENVALUE
C
  62  IF(DELNEW-10.*EPS*AMAX1(ZZ,.001*CAP))64,64,63
  63  DELOLD=DELNEW
      ROLD=RNEW
      IF(LLY-50)23,23,100
  64  MARK=2
  70  BL1=1.
C
C     WE ACCEPT Z AS A ROOT
C
 100  NUQ=NUQ+1
      RT(NUQ)=Z
      WRITE(3,3) Z,LLY,MARK
      LLY=0
      CAP=MAX1 (ZZ,CAP)
      DELOLD=1.
      ROLD=1.
      SUM1=SUM1+RT(NUQ)
      SUM2=SUM2+RT(NUQ)**2
      S1=SUM1-SPUR1
      S2=SUM2-SPUR2
      IF(NUQ-N1)84,101,101
C
C     A NEWTON STEP TOWARDS NEXT ROOT
C
  84  IF((ABS(REAL(Q1))+ABS(AIMAG(Q1)))*MAX1(ZZ,.001*CAP)-1.E+4)86,86,14
  86  Z=Z-B2/((.5,.0)*B3-B2*Q1)
      GO TO 23
 101  CONTINUE
      RETURN
  1   FORMAT(1HO50X,19HLAGUERRE ITERATIONS//31X,9HREAL PART10X,10H IMAG.
     1PART22X,1HP11X,7HP PRIME6X,11HP DBL PRIME)
  2   FORMAT(1H ,'ITERATE',20X,(F15.8,F15.8),8X,3F15.4)
  3   FORMAT(1H ,'EIGENVALUE',12X,(F20.8,F20.8),12X,I3,'  ITERATIONS,
     1TEST',I1,//)
  4   FORMAT(1H ,'ITERATE',20X,(F20.8,F20.8),12X,'OVERFLOW')
      END
```

APPENDIX C

```
C
C                SUBPROGRAM THREE: F4 UTEX MATSUB, EIVAL
C
C
C
C
C
C INPUT PROGRAM
C
      COMMON AR(18,18),AI(18,18)
  999 FORMAT(6I2,F4.3,F8.4)
  998 FORMAT(1H1,'ECHO CHECK OF INPUT PARAMETERS')
  997 FORMAT(1H, 6I6,F10.5,F18.8)
  996 FORMAT(10F10.6)
  995 FORMAT(1H-,'ECHO CHECK OF REAL PART OF INPUT MATRIX')
  994 FORMAT(1H-,'ECHO CHECK OF IMAGINARY PART OF INPUT MATRIX')
  993 FORMAT(1H ,T7,'M',T11,'IEG',T16,'IVEC',T22,'IDET',T29,'MIT',T34,
     1'MITS',T42,'EP1',T60,'EP2')
  992 FORMAT(1H-,'EXECUTION TIME=',10X,F10.6)
    5 CALL INTIME(ITIME)
      READ(1,999,END=5) M,IEG,IVEC,IDET,MIT,MITS,EP1,EP2
      WRITE(3,998)
      WRITE(3,993)
      WRITE(3,997) M,IEG,IVEC,IDET,MIT,MITS,EP1,EP2
      DO 1 I=1,M
    1 READ(1,996) (AR(I,J),J=1,M)
      DO 2 I=1,M
    2 READ(1,996) (AI(I,J),J=1,M)
      WRITE(3,995)
      DO 3 I=1,M
    3 WRITE(3,996) (AR(I,J),J=1,M)
      WRITE(3,994)
      DO 4 I=1,M
    4 WRITE(3,996) (AI(I,J),J=1,M)
      ALRS=0.
      ALIS=0.
      GBR=0.
      GBI=0.
      CALL EIVAL(M,IEG,IVEC,ALRS,ALIS,GBR,GBI,IDET,MIT,MITS,EP1,EP2)
      CALL INTIME(ITIME1)
      TIME=(ITIME1-ITIME)/100.
      WRITE(3,992) TIME
      GO TO 5
    6 CONTINUE
      STOP
      END
C
C
C
C
C
      SUBROUTINE EIVAL (M,IEG,IVEC,ALRS,ALIS,GBR,GBI,IDET,MIT,MITS,EP1,
     1EP2)
      DIMENSION AR(18,18),AI(18,18),BR(18,18),BI(18,18),CR(18,18),CI(18,
     118),XR(18   ),XI(18   ),YR(18),YI(18),ZR(18),ZI(18)
      COMMON CR,CI
      ITWO=2
      IONE=1
      N=M
```

```
      SUMR=0.0
      SUMI=0.0
      PRDR=1.0
      PRDI=0.0
      TRACER=0.0
      TRACEI=0.0
      DO 1499 I=1,18
      XR(I)=0.
      XI(I)=0.
      YR(I)=0.
      YI(I)=0.
      ZR(I)=0.
 1499 ZI(I)=0.
      DO 450 I=1,N
      TRACER=TRACER+CR(I,I)
  450 TRACEI=TRACEI+CI(I,I)
C
C     SET UP MATRICES
C
      DO 519 I=1,N
      DO 519 J=1,N
      BR(I,J)=CR(I,J)
      AR(I,J)=CR(I,J)
      BI(I,J)=CI(I,J)
  519 AI(I,J)=CI(I,J)
C
C     EVALUATE DETERMINANT
C
      ASSIGN 520 TO IA
      ASSIGN 811 TO ID
      MM=M
      INTER=0
      GO TO 535
  520 DETR=1.0
      DETI=0.0
      DO 522 K=1,M
      T1=DETR*AR(K,K)-DETI*AI(K,K)
      DETI=DETR*AI(K,K)+DETI*AR(K,K)
  522 DETR=T1
      INTER=MOD  (INTER,2)
      IF (INTER) 1000,917,810
 1000 WRITE(3,1001)
 1001 FORMAT(1H ,'STOP')
      RETURN
  810 DETR=-DETR
      DETI=-DETI
  917 GO TO ID, (811,912)
  811 WRITE(3,557) TRACER,TRACEI,DETR,DETI
  557 FORMAT(1H-,'TRACE OF MATRIX =',2F18.8,5X,'DETERMINANT OF MATRIX ='
     1,2F18.8)
      ASSIGN 912 TO ID
      ASSIGN 530 TO IA
      ASSIGN 40 TO IB
      ASSIGN 523 TO IC
      ISL=-1
      GO TO 92
  523 ISL=0
C
C     EIGENVALUE GUESS OR ORIGIN TRANSLATION
C
```

```
    9 ALR=ALRS
      ALI=ALIS
      IT=1
C
C     EIGENVECTOR GUESS
C
  403 DO 504 I=1,N
      XR(I)=1.0
  504 XI(I)=0.0
    4 DO 5 I=1,N
      AR(I,I)=AR(I,I)-ALR
    5 AI(I,I)=AI(I,I)-ALI
C
C     FIRST ITERATION - POWER METHOD
C
      IJ=1
   10 BIG=0.
C
C     COMPUTE Y=(A-ALPHA)*X
C
      DO 13 I=1,N
      YR(I)=0.
      YI(I)=0.
      DO 11 J=1,N
      YR(I)=YR(I)+AR(I,J)*XR(J)-AI(I,J)*XI(J)
   11 YI(I)=YI(I)+AI(I,J)*XR(J)+AR(I,J)*XI(J)
      AM=YR(I)**2+YI(I)**2
      IF (AM-BIG) 13,13,12
   12 BIG=AM
      JJ=I
   13 CONTINUE
      IF (BIG) 109,106,109
C
C     EXACT EIGENVALUE AND EIGENVECTOR - Y=0. FLAG=1000
C
  106 ICT=1000
      DO 108 I=1,N
      JJ=I
      IF (XR(I)-1.0) 108,118,108
  118 ISL=1
      GO TO 92
  108 CONTINUE
      WRITE(3,650)
  650 FORMAT(1H ,'ERROR.  EIGENVECTOR NOT NORMALIZED IN METHOD 1.')
      GO TO 990
C
C     MU RAYLEIGH QUOTIENT - (Y.X)/(X,X)=MU
C
  109 RQNR=0.
      RQNI=0.
      RQD=0.
      DO 14 I=1,N
      RQNR=RQNR+XR(I)*YR(I)+XI(I)*YI(I)
      RQNI=RQNI+XR(I)*YI(I)-XI(I)*YR(I)
   14 RQD=RQD+XR(I)**2+XI(I)**2
      AMUR=RQNR/RQD
      AMUI=RQNI/RQD
      AMM=AMUR**2+AMUI**2
      IF (IEG) 1000,81,80
   80 ALRC=AMUR+ALR
```

```
      ALIC=AMUI+ALI
      WRITE(3,300) IONE,IJ,ALRC,ALIC
  300 FORMAT(1H ,2I4,2F20.8)
C
C     TEST FIRST ITERATION
C     MAGNITUDE OF (Y-MU*X)=TS
C
   81 TS=0.
      DO 15 I=1,N
   15 TS=TS+(YR(I)-AMUR*XR(I)+AMUI*XI(I))**2+
     1(YI(I)-AMUR*XI(I)-AMUI*XR(I))**2
C
C     NORMALIZATION
C
      DO 16 I=1,N
      XR(I)=(YR(JJ)*YR(I)+YI(JJ)*YI(I))/BIG
   16 XI(I)=(YR(JJ)*YI(I)-YI(JJ)*YR(I))/BIG
      XR(JJ)=1.0
      XI(JJ)=0.0
  111 IF (TS/RQD-EP1) 20,20,18
   18 IF (IJ-MIT) 19,20,20
   19 IJ=IJ+1
      GO TO 10
C
C     SECOND ITERATION - INVERSE POWER METHOD
C
   20 ICT=IJ
      MIT2=MITS+IJ
      ALR=AMUR+ALR
      ALI=AMUI+ALI
      MM=N
      DO 310 I=1,N
      AR(I,I)=AR(I,I)-AMUR
  310 AI(I,I)=AI(I,I)-AMUI
      GO TO 29
   99 DO 100 I=1,N
      AR(I,I)=AR(I,I)-ALR
  100 AI(I,I)=AI(I,I)-ALI
   29 IJ=IJ+1
C
C     GAUSSIAN ELIMINATION - (A-ALPHA)*Y=X
C
  535 DO 27 I=2,MM
      IM1=I-1
      DO 27 J=1,IM1
   21 FM=AR(I,J)**2+AI(I,J)**2
      SM=AR(J,J)**2+AI(J,J)**2
      IF (FM-SM) 24,24,22
C
C     ROW INTERCHANGE - IF NECESSARY
C
   22 DO 23 K=J,MM
      T1=AR(J,K)
      T2=AI(J,K)
      AR(J,K)=AR(I,K)
      AI(J,K)=AI(I,K)
      AR(I,K)=T1
   23 AI(I,K)=T2
      T1=XR(J)
      T2=XI(J)
```

```
      XR(J)=XR(I)
      XI(J)=XI(I)
      XR(I)=T1
      XI(I)=T2
      T1=FM
      FM=SM
      SM=T1
      INTER=INTER+1
   24 IF (SM) 25,27,25
   25 IF (FM) 90,27,90
C
C     TRIANGULARIZATION
C
   90 RR=(AR(I,J)*AR(J,J)+AI(I,J)*AI(J,J))/SM
      RI=(AR(J,J)*AI(I,J)-AR(I,J)*AI(J,J))/SM
      DO 26 K=J,MM
      AR(I,K)=AR(I,K)-RR*AR(J,K)+RI*AI(J,K)
   26 AI(I,K)=AI(I,K)-RR*AI(J,K)-RI*AR(J,K)
      AR(I,J)=0.
      AI(I,J)=0.
      XR(I)=XR(I)-RR*XR(J)+RI*XI(J)
      XI(I)=XI(I)-RR*XI(J)-RI*XR(J)
   27 CONTINUE
      GO TO IA, (520,530,911,530)
  530 SMALL=1000.
      DO 28 K=1,MM
      IKK=K
      T1=AR(K,K)**2+AI(K,K)**2
      IF (T1) 750,752,750
  750 IF (T1-SMALL) 751,28,28
  751 SMALL=T1
      IZ=K
   28 CONTINUE
      GO TO IB, (40,753,40)
  752 IZ=IKK
      IF (ISL) 753,30,30
C
C     EXACT EIGENVALUE - (A-ALPHA) SINGULAR. FLAG=2000
C
   30 ISL=1
      ICT=2000
      DO 974 I=1,MM
      XR(I)=0.0
  974 XI(I)=0.0
  753 YR(IZ)=1.0
      YI(IZ)=0.0
      JJ=IZ
      BIG=1.0
      IF (IZ-MM) 33,32,33
   32 IZZ=2
      GO TO 95
   33 IZZ=IZ+1
      DO 31 I=IZZ,MM
      YR(I)=0.
   31 YI(I)=0.
      IZZ=MM-IZ+2
      IF (IZ-1) 95,49,95
C
C     BACKWARD SUBSTITUTION
C
```

```
   40 IZZ=1
   41 BIG=0.
   95 DO 46 I=IZZ,MM
      II=MM-I+1
      KK=II+1
      SR=0.
      SI=0.
      IF (I-1) 42,44,42
   42 DO 43 K=KK,MM
      SR=SR+AR(II,K)*YR(K)-AI(II,K)*YI(K)
   43 SI=SI+AR(II,K)*YI(K)+AI(II,K)*YR(K)
   44 T1=AR(II,II)**2+AI(II,II)**2
      YR(II)=(AR(II,II)*(XR(II)-SR)+AI(II,II)*(XI(II)-SI))/T1
      YI(II)=(AR(II,II)*(XI(II)-SI)-AI(II,II)*(XR(II)-SR))/T1
      AM=YR(II)**2+YI(II)**2
      IF (AM-BIG) 46,46,45
   45 JJ=II
      BIG=AM
   46 CONTINUE
C
C     NORMALIZATION - X=NORMALIZED Y
C
   49 DO 47 I=1,MM
      XR(I)=(YR(JJ)*YR(I)+YI(JJ)*YI(I))/BIG
   47 XI(I)=(YR(JJ)*YI(I)-YI(JJ)*YR(I))/BIG
      XR(JJ)=1.0
      XI(JJ)=0.0
   92 IF(N) 116,116,9921
 9921 DO 601 I=1,N
      DO 601 J=1,N
      AR(I,J)=BR(I,J)
  601 AI(I,J)=BI(I,J)
  116 IF (ISL) 755,50,60
  755 GO TO IC, (523,704,525)
C
C     ALPHA RAYLEIGH QUOTIENT - (AX,X)/(X,X)=ALPHA
C
   50 ALR=0.
      ALI=0.
      SUM=0.0
   55 DO 52 I=1,N
      YR(I)=0.
      YI(I)=0.
      DO 51 K=1,N
      YR(I)=YR(I)+AR(I,K)*XR(K)-AI(I,K)*XI(K)
   51 YI(I)=YI(I)+AR(I,K)*XI(K)+AI(I,K)*XR(K)
      ALR=ALR+XR(I)*YR(I)+XI(I)*YI(I)
      ALI=ALI+XR(I)*YI(I)-XI(I)*YR(I)
   52 SUM=SUM+XR(I)**2+XI(I)**2
      ALR=ALR/SUM
      ALI=ALI/SUM
      AM=ALR**2+ALI**2
      IF (IEG) 1000,83,82
   82 WRITE(3,300) ITWO,IJ,ALR,ALI
C
C     TEST SECOND ITERATION
C
   83 TS=0.
      DO 53 I=1,N
      T1=YR(I)-ALR*XR(I)+ALI*XI(I)
```

```
      T2=YI(I)-ALR*XI(I)-ALI*XR(I)
   53 TS=TS+T1**2+T2**2
   93 IF (TS/SUM-EP2)60,60,301
  301 IF (IJ-MIT2) 99,400,400
  400 WRITE(3,401) IT
  401 FORMAT(1H ,'INVERSE POWER METHOD NOT CONVERGED ON TRY NUMBER',I5)
      IF (IT-3) 402,990,402
  402 ALR=ALR+GBR
      ALI=ALI+GBI
      IT=IT+1
      WRITE(3,820) ALR,ALI
  820 FORMAT(1H ,'ALPHA=',2F20.8)
      GO TO 4
   60 ISL=0
   63 WRITE(3,64) ALR,ALI
   64 FORMAT(1H-,'EIGENVALUE=',2F18.8)
      ZR(N)=ALR
      ZI(N)=ALI
      SUMR=SUMR+ALR
      SUMI=SUMI+ALI
      T1=PRDR*ALR-PRDI*ALI
      PRDI=PRDR*ALI+PRDI*ALR
      PRDR=T1
C
C     DEFLATICN OF MATRIX
C
      IF (JJ-N) 61,65,61
C
C     PERMUTATION OPERATION
C
   61 T1=XR(JJ)
      T2=XI(JJ)
      XR(JJ)=XR(N)
      XI(JJ)=XI(N)
      XR(N)=T1
      XI(N)=T2
      DO 68 K=1,N
      T1=AR(JJ,K)
      T2=AI(JJ,K)
      AR(JJ,K)=AR(N,K)
      AI(JJ,K)=AI(N,K)
      AR(N,K)=T1
   68 AI(N,K)=T2
      DO 62 K=1,N
      T1=AR(K,JJ)
      T2=AI(K,JJ)
      AR(K,JJ)=AR(K,N)
      AI(K,JJ)=AI(K,N)
      AR(K,N)=T1
   62 AI(K,N)=T2
C
C     DEFLATION
C
   65 N=N-1
      DO 66 I=1,N
      DO 66 J=1,N
      AR(I,J)=AR(I,J)-XR(I)*AR(N+1,J)+XI(I)*AI(N+1,J)
   66 AI(I,J)=AI(I,J)-XR(I)*AI(N+1,J)-XI(I)*AR(N+1,J)
      DO 600 I=1,N
      DO 600 J=1,N
```

```
      BR(I,J)=AR(I,J)
  600 BI(I,J)=AI(I,J)
C
C     COMPUTE EIGENVECTOR AND/OR DETERMINANT AS REQUIRED
C
  910 IF (IDET) 1000,527,700
  527 IF (IVEC) 1000,525,700
  700 DO 702 I=1,M
      DO 702 J=1,M
      AR(I,J)=CR(I,J)
      AI(I,J)=CI(I,J)
      IF (I-J) 702,701,702
  701 AR(I,I)=AR(I,I)-ALR
      AI(I,I)=AI(I,I)-ALI
  702 CONTINUE
      MM=M
      INTER=0
      ASSIGN 911 TO IA
      GO TO 535
  911 ASSIGN 530 TO IA
      IF (IDET) 1000,914,520
  912 WRITE(3,913) DETR,DETI
  913 FORMAT(1H ,'DETERMINANT=',2F18.8)
      ZLAG=SQRT (AR(1,1)**2+AI(1,1)**2)
      ZLIT=ZLAG
      DO 923 I=2,M
      ZMAGT=SQRT (AR(I,I)**2+AI(I,I)**2)
      IF (ZLAG-ZMAGT) 922,920,920
  920 IF (ZLIT-ZMAGT) 923,923,921
  921 ZLIT=ZMAGT
      GO TO 923
  922 ZLAG=ZMAGT
  923 CONTINUE
      WRITE(3,924) ZLAG,ZLIT
  924 FORMAT(1H ,'LARGEST AND SMALLEST MAGNITUDES OF DIAGONAL ELEMENTS O
     1F TRI. MATRIX=',2F18.8)
  914 ISL=-1
      IF (IVEC) 1000,916,915
  915 DO 703 I=1,M
      XR(I)=0.
  703 XI(I)=0.
      ASSIGN 753 TO IB
      ASSIGN 704 TO IC
      GO TO 530
  916 ASSIGN 525 TO IC
      GO TO 92
  704 WRITE(3,705) (XR(I),XI(I),I=1,M)
  705 FORMAT(1H-,'ASSOCIATED EIGENVECTOR IS', 2F20.8)
      ASSIGN 40 TO IB
  525 IF (N-1) 526,67,523
   67 ALR=AR(1,1)
      ALI=AI(1,1)
      SUMR=SUMR+ALR
      SUMI=SUMI+ALI
      T1=PRDR*ALR-PRDI*ALI
      PRDI=PRDR*ALI+PRDI*ALR
      PRDR=T1
      ZR(1)=ALR
      ZI(1)=ALI
      WRITE(3,320) ALR,ALI
```

```
320 FORMAT(1H-,'FINAL EIGENVALUE =',2F18.8)
    N=0
    GO TO 910
526 WRITE(3,321) SUMR,SUMI,PRDR,PRDI
321 FORMAT(1H-,'SUM OF EIGENVALUES =',2F18.8,5X,'PRODUCT OF EIGENVALUE
   1S =',2F18.8)
990 CONTINUE
    RETURN
    END
```

ADAPTING BORROWED COMPUTING SUBPROGRAMS FOR A SPECIFIC COMPUTING INSTALLATION

by

THOMAS MICHAEL RAWSON

B. S., Kansas State University, 1965
M. S., Kansas State University, 1967

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Statistics and Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1970

The ease with which many "library" computing subprograms are adaptable for use on a particular computing installation is usually overestimated by the typical computing center user. Because of the variations in programming languages and computing hardware, the immediate successful operation of a computing subprogram which was written for use at another installation is highly improbable. This project was undertaken to report the major problems encountered when a typical computing user attempts to make three borrowed subprograms operational on a particular computing installation.

The author chose to use subprograms which were written to compute the eigenvalues of real and complex matrices. Two of these subprograms, ALLMAT and EIG4, were obtained from SHARE while the third, EIVAL, was obtained from CO-OP. Both SHARE and CO-OP are cooperative computing users organizations which distribute library subprograms.

The author encountered different types of problems associated with each of the three subprograms. ALLMAT proved to be the easiest to adapt, the most accurate, and the fastest of the three subprograms. The adaptation of ALLMAT included writing an input-output program and correcting certain undefined variables. A total of four hours of programming and keypunching was utilized in making ALLMAT operational.

Subprogram EIG4 was the most difficult to adapt. This subprogram proved to be the least accurate of the three and also the slowest. Errors encountered in the adaptation of EIG4 included the correcting of undefined variables, checking for correct DO parameters, and correcting certain

FORTRAN IV library function references. The user spent a total of ten hours reprogramming EIG4 to make it operational.

EIVAL, the third subprogram selected by the author, presented similar problems to the user. These obstacles included the writing of an input program, the correcting of FORTRAN IV library function references and the initialization of certain variables. The user spent approximately six hours making EIVAL operational.

The three subprograms were tested with both real and complex test matrices. The user was able to convert EIVAL to double precision because of the programming techniques used in its construction. The double precision version proved to be highly accurate in its calculations. EIG4 was found to be inadequate in calculating the eigenvalues of symmetric real or complex matrices.

Several logistical problems were encountered by the user. These problems were specific ones faced by the user in the solution of the proposed problem.

Many barriers were encountered by the author in attempting to adapt the three subprograms for use on the Kansas State University computing installation. These barriers were not insurmountable but they must be taken into account when a computing center user considers adapting a borrowed subprogram for use on an available computing installation.