

1-1-1978

A comparison of several algorithms for maximum-likelihood estimation of parameters in unrestricted common factor analysis.

Michael Patrick Hagerty
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_1

Recommended Citation

Hagerty, Michael Patrick, "A comparison of several algorithms for maximum-likelihood estimation of parameters in unrestricted common factor analysis." (1978). *Doctoral Dissertations 1896 - February 2014*. 2100.
https://scholarworks.umass.edu/dissertations_1/2100

This Open Access Dissertation is brought to you for free and open access by ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations 1896 - February 2014 by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.



A COMPARISON OF SEVERAL ALGORITHMS FOR
MAXIMUM-LIKELIHOOD ESTIMATION OF PARAMETERS
IN UNRESTRICTED COMMON FACTOR ANALYSIS

A Dissertation Presented

by

MICHAEL PATRICK HAGERTY

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements of the degree of

DOCTOR OF EDUCATION

May 1978

Education

(c) Michael Patrick Hagerty 1978

All Rights Reserved

A COMPARISON OF SEVERAL ALGORITHMS FOR
MAXIMUM-LIKELIHOOD ESTIMATION OF PARAMETERS
IN UNRESTRICTED COMMON FACTOR ANALYSIS

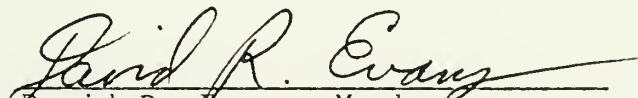
A Dissertation Presented


by

MICHAEL PATRICK HAGERTY

Approved as to style and content by:


Thomas E. Hutchinson, Chairman


David R. Evans, Member


Hariharan Swaminathan, Member


Mario Fantini, Dean

School of Education

ACKNOWLEDGEMENTS

In reviewing the process which has culminated in the production of this document, the support and guidance of a number of individuals has become apparent. Initially, my sincere thanks is offered to Dwight Allen, who made the process attainable; to Dave Evans, who encouraged the acquisition of 'hard skills'; to Tom Hutchinson and Jimmie Fortune, who assisted me in developing those skills; and to Swaminathan, for providing a problem on which they could be tested.

The staff and management of Abt Associates Inc. did so much to further this effort that a simple thank you does not seem adequate. It is appropriate that Sandy Friedman be singled out as the individual most responsible for pressuring me into finishing this document. For this, I owe him a debt of gratitude.

Lastly and most importantly, I must thank Donna, for without her constant encouragement in the darkest hours, this document would never have been completed.

ABSTRACT

A COMPARISON OF SEVERAL ALGORITHMS FOR
MAXIMUM-LIKELIHOOD ESTIMATION OF PARAMETERS
IN UNRESTRICTED COMMON FACTOR ANALYSIS

September 1978

Michael Patrick Hagerty

B.A., Southern Illinois University

Ed.D., University of Massachusetts

Directed by: Professor Thomas E. Hutchinson

The purpose of this study was to compare a set of nonlinear minimization routines within the context of the unrestricted factor analysis model. It was anticipated that the outcome of the study would provide researchers with recommendations concerning the efficiency of the various algorithms for minimization in this context.

To this end, eight routines which had either been used in factor analysis before, or had demonstrated high levels of efficiency in other problems, were collected and tested.

The set included Joreskog and van Thillo's NWTRAP, van der Voort's MINIM, Powell's VA06A, Fletcher's VA09A, Shanno's MINFUN, Browne's FACTOR, Gruvaeus and Joreskog's STEDE/FLEPOW, and the author's reworking of NWTRAP. The data used in the test procedure were matrices which had been previously factor analyzed in published reports.

A program was written to serve as the environment for the testing of the individual routines. The unrestricted factor analysis model proposed by Joreskog and van Thillo was implemented in the form of a standalone function to be invoked by each of the routines under test. The resultant solutions were compared against the solutions produced by the widely-used UFABY3 program. Information was collected on the robustness and accuracy of the routines, as well as the CPU time, number of iterations and evaluations, and amount of memory required.

The information collected by the test program was tabled and examined to determine the parameters which would allow the individual routines to be included as part of a general factor analysis package.

In summary, the study indicated that the choice of a minimization routine does make a difference. To select an inefficient algorithm is to guarantee the needless waste of

large sums of computer time. The recent availability of efficient algorithms for non-linear minimization provides the means by which efficient programs can be produced for increasingly complex factor analytic problems.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iv
ABSTRACT	v
TABLE OF CONTENTS	viii
LIST OF TABLES	ix
Chapter	
I INTRODUCTION	
History of Maximum-likelihood Factor Analysis	1
Advantages of Maximum-likelihood Estimates	3
Barriers to Utilization	4
II THE MINIMIZATION PROCESS	
Minimization Methods	6
Geometric Interpretation of Minimization	9
III THE NECESSITY OF COMPARISON	
Statement of the Problem	13
Refinements and Constraints	15
IV IMPLEMENTATION OF THE COMPARISON	
Details of the Selected Routines	18
The Function Generator	25
An Environment for Testing	28
The Testing Procedure	31
V RESULTS OF THE COMPARISON	
Interpretation of Tabular Results	34
Comments on Individual Routines	38
VI IMPLICATIONS OF THE STUDY	
General Conclusions	43
Recommendations for Future Research	44
TABLES	46
BIBLIOGRAPHY	76

LIST OF TABLES

Table		Page
1	Matrices Analyzed	46
2	Function Value at Reported Minimum	47
3	CPU Seconds to Reported Convergence	48
4	Epsilon at Reported Minimum	49
5	Memory Required for Program	50
6	Memory Required for Arrays	51
7	Number of Iterations Reported	52
8	Number of Function Evaluations	53
9	Number of Gradient Evaluations	54
10	Number of Approximate Hessian Evaluations	55
11	Number of Exact Hessian Evaluations	55
12	Source Listing of FROG (Main Driver)	56
13	Source Listing of INITIAL	59
14	Source Listing of ALLOCAT	61
15	Source Listing of OV1S2	63
16	Source Listing of NEWTON	65
17	Source Listing of MLF	67
18	Source Listing of JMLFN	68
19	Source Listing of CLAM	72
20	Source Listing of FLAM	74

C H A P T E R I

INTRODUCTION

History of Maximum-likelihood Factor Analysis

Factor analysis is a statistical technique which has been widely used, particularly in psychological research. Many methods of estimating factor loadings have been proposed, but the majority are of an approximate and non-statistical nature and little is known of the property of the estimates. By contrast, a statistically sound technique was first developed by Lawley [1940], using the method of maximum likelihood. These estimates of factor loadings are theoretically preferable to other estimates which have been proposed, as they are asymptotically efficient and there is a corresponding likelihood ratio test for assessing the fit of the factor analysis model. However, a great amount of computation is involved in solving the likelihood equations, and the earlier computational procedures suggested have been known to break down in certain cases [Lord, 1956], so that other methods for estimating factor loadings have been used in the majority of reported studies. [Browne, 1968a]

Since the above passage was written, several major advances have occurred that improve and refine the computational procedures required to solve the likelihood equations. However, these developments have gone virtually unnoticed in the fields of education and psychology. Mulaik [1972] points out that as a group, the early factor analysts were "not statisticians concerned with questions of

statistical inference," and were therefore reluctant to become entangled in the exceedingly complex problems which even mathematical statisticians were finding difficult to solve. Even after Lawley [1940] developed the equations for estimation of factor loadings using maximum-likelihood, interest remained unkindled. In this instance hesitation was not unwarranted, however, as the algorithm supplied by Lawley was not practical for problems of the large size to which psychologists were accustomed.

Mulaik further explains that when Howe [1955] first demonstrated that maximum-likelihood estimates (MLE) of the loadings could be derived without imposing any distributional assumptions on the variates and also provided a Gauss-Seidel method far superior to Lawley's, no serious attempt was made to implement the algorithm on the newly-emerging electronic computers. (In personal correspondence, Browne points out that this lack of effort was due mainly to ignorance as Howe's procedure was never formally published). The practitioners instead implemented a method developed by Rao [1955]. Jennrich and Robinson [1969] explain that although Rao's algorithm was better than Lawley's, it was not much better, "requiring extensive computing and many iterations with doubtful convergence." [Mulaik, 1972]

Mulaik concludes that psychologists "felt the statisticians had not provided the solutions needed, and so, while awaiting further developments, abandoned the (then popular) centroid method of computing and took up variations of components analysis."

Advantages of Maximum-likelihood Estimates

It might, at this point, be helpful to detail the advantages of using MLE. Mulaik [1972] states that ML estimators "are usually superior to other estimators in estimating population parameters." Jennrich and Robinson [1969] further explain that MLE is characterized by "asymptotic efficiency, invariance under changes in scale and the existence of a chi-square test for additional factors." Harman [1968], after providing definitions of statistical consistency, efficiency, sufficiency, and unbiasedness, states that

The method of maximum likelihood is a well-established and popular statistical procedure for estimating the unknown population parameters because such estimators satisfy the first three of the above (consistency, efficiency, and sufficiency) standards. Not all parameters have sufficient estimators but if one exists, the maximum-likelihood estimator is such a sufficient estimator. However, a maximum-likelihood estimator will generally not be unbiased. (While it is of some advantage to devise an unbiased estimate, it is not a very critical requirement.)

Barriers to Utilization

With the many advantages to be gained by the utilization of MLE, the question may be asked, "Why are they not more commonly used?" The answer is that for want of a computationally efficient and mathematically accurate technique for producing the estimates, Lawley's and Howe's procedures never gained a large following. In order to develop a technique that would satisfy these conditions it would be necessary to identify a numerically efficient method of minimizing the likelihood function. Basically, the issue reduces to a rather complicated nonlinear optimization problem, an area in which numerical analysts generally show far greater interest than social scientists. It is not surprising that years passed before an innovation in optimization theory found application in factor analysis.

For example, it was not until Joreskog [1967], upon Lawley's suggestion, used the algorithm developed by the physicist William Davidon, later refined by the numerical analysts Fletcher and Powell [1963] to minimize the function of the ML criterion in the common factor analysis model, that a relatively speedy convergence was obtained. Mulaik [1972] hails this application of an innovation in numerical analysis as the solution of the major computational obstacle barring the way to further use of MLE as well as the first

computationally feasible method.

C H A P T E R I I
THE MINIMIZATION PROCESS

Minimization Methods

It would be convenient if, for something so important as the solution of a minimization problem there existed a clearly superior computational method. Unfortunately this is not the case. Fiacco and McCormick [1968] have described four conceptually distinct types: derivative-free simple search procedures; gradient techniques; quasi-Newton and true Newton methods. An important method they do not include is to solve the equations giving necessary conditions for a minimum. In addition, there are a large number of variations that do not fit neatly into any one of the above categories.

The "search" and "conjugate direction" methods have the principal advantage of not requiring the calculation of derivatives, a difficult and time-consuming chore when the function is particularly complex [Powell, 1970]. In describing his implementation of Nelder and Mead's [1965] Simplex method, O'Neill [1971] explains that it is

essentially opportunistic, making use only of current information. The underlying strategy of this method is to create an $(n+1)$ -dimensional solid, where n is the number of variables in the problem, and then to compute the value of the objective function at each of the corners. The solid is expanded, reflected (flipped over) and/or contracted until all of the corners coincide in space at the point that represents the minimum of the function.

The gradient methods utilize the principle of "steepest descent (or ascent)," and inasmuch as the principle was first proposed by Cauchy in 1847, these methods can hardly be accorded the distinction of being recent advances. Although the underlying premise remains the same, a number of variations of this method exist. Instead of utilizing information only pertaining to each corner as in Simplex, these methods keep track of the direction that has been most successful in past iterations to point them in the most promising direction for the succeeding iteration. Using this vector of pointers (the gradient) a series of steps is taken until the minimum is reached.

Because the third type of computational method uses only an approximation to the Hessian (the matrix of second-order derivatives), these methods are termed "quasi-Newton." At each iteration, successively closer approximations to the

inverse of the Hessian are computed to be used in the next iteration. The original Davidon-Fletcher-Powell algorithm is the most widely known of this category.

Lastly, there are pure Newton methods. On comparing various minimization algorithms, Fiacco and McCormick [1968] found this method, the Newton-Raphson, to be the most effective. To achieve the efficiency characteristic of this category, it is necessary to compute the actual Hessian at each iteration. Calculation of the Hessian increases in difficulty rather dramatically with increased complexity of the problem. Jennrich and Robinson [1969], Clarke [1971], and Joreskog and van Thillo [1971] all have specified the second derivatives for the classical unrestricted factor analysis model and using the Newton-Raphson procedure, achieved solutions. However, the manner in which the model was specified precludes the possibility of extension to other, more general models.

Joreskog and Goldberger [1971], prime champions of the Fletcher-Powell algorithm until 1971, say: "It has been found that the Newton-Raphson procedure is very efficient, generally requiring only a few iterations for convergence." Swaminathan [1971] explains that the success of the Newton-Raphson scheme "hinges on the availability of the matrix of second derivatives."

Geometric Interpretation of Minimization

While minimization is usually described in terms of calculus or algebra, a strong case can be made for adopting a geometric interpretation. Simply stated, finding the minimum of a function in several variables is analogous to finding the bottom of an unfathomed pond. The minimizer is analogous to a surveyor rowing a boat on the surface of this pond. It is the surveyor's responsibility, upon being provided with specific tools and instructions, to locate the coordinates of the deepest point.

In the case of direct search methods the surveyor is given a plumb line to fathom the depth under his boat (the value of the function at that point). The surveyor will then row in a straight line, testing the depth at fixed intervals until he begins to head into shallower water. As most direct search algorithms dictate the selection of a path perpendicular to the unsuccessful previous direction, the surveyor will then make a 90 degree turn to continue his search for deeper water.

In the case of the Simplex method, the surveyor is joined by a number of compatriots equal to the number of dimensions plus one (three surveyors in the two dimensional

case). At each iteration, the boat over the deepest point yet discovered is allowed to row a fixed distance perpendicular to a line connecting his colleagues. This strategy has the effect of expanding, contracting, or if the surveyor crosses the connecting line, reflecting the solid (triangle) formed by their positions. Assuming the bottom of the pond to be concave and smooth, once the lowest point has been encircled, convergence of the boats over this point is guaranteed. The method's only drawback to offset this extremely desirable feature, guaranteed convergence, is the large number of measurements which must be taken since only one surveyor may move at a time.

The gradient methods utilize single surveyors but are more successful than direct search methods because the surveyor remembers the direction from which he has come and has an indication of the most useful direction to take in the future. Hence, the surveyor tends not to strike off in directions which head into already explored shallow water. Because the surveyor is going to row a fixed distance before he is allowed to drop his line, he will have the greatest success around the edge of the pond, where the water becomes deep most rapidly, and will have the greatest difficulty in the neighborhood of the true bottom, where the slope is relatively flat. While above this flat area, the surveyor may row back and forth a fixed distance over the deepest

point without stopping, assuming the bottom to be flat, he reports coordinates of some point other than the true bottom. This difficulty may cause the frustrated surveyor crossing the deepest point over and over, each time going the same distance and never stopping in the middle, to give up.

The Newton approach involves remembering not only the most productive direction, but also the distance necessary to recognize a given increase in depth. So concerned are Newton methods with this distance/decrease ratio that they have difficulty at the outer edge, where the slope is most steep. Near the bottom, the surveyor goes shorter and shorter distances until the distance is practically zero, signifying the end of the search. Quasi-Newton methods attempt to approximate the Newton approach by varying the distance the surveyor rows in response to changes in the relative increase in depth. The major difference between strategies is that the quasi-Newtons determine step size approximately, while the Newton-Raphson procedure computes it directly.

Turning from the analogy to the current situation, we find that search methods have only the value of the function (drop line) available at each iteration; the gradient and quasi-Newton methods have function and gradient (direction

of greatest depth); and the Newton methods have function, gradient and Hessian (slope).

For convenience, these methods will be referred to by category throughout the remainder of this paper: direct search methods will be Category 0, gradient and quasi-Newton will be Category 1, and the pure Newton methods, Category 2.

C H A P T E R I I I
THE NECESSITY OF COMPARISON

Statement of the Problem

While it is not the purpose of this paper to examine the mathematical basis of either Maximum Likelihood estimation in factor analysis or the specific algorithms used in the minimization process, it is necessary to outline the import of minimization in ML factor analysis in order to bring the issue at hand into clearer focus. For a more complete (or mathematical) explanation the reader is referred to the outstanding books by Lawley and Maxwell [1971], Mulaik [1972], Adby and Dempster [1974], and Himmelblau [1972a]. The factor analysis problem ultimately reduces to an attempt to produce a matrix with fewer columns (factors) than rows (variables). When multiplied by its transpose and added to a diagonal matrix of error, the product should reproduce the original correlation matrix to a specified number of decimal places. The justification is simple parsimony: it is easier to understand what is happening in a set of data when the dimensionality is reduced.

In the current instance, this reduction in dimensionality is accomplished by beginning with some initial estimate of this space (principal components is a very common example), and adjusting the values of the matrix (and/or error) until the initial correlation matrix is reproduced. This result is achieved through the use of a minimization procedure. Customarily, the initial estimate is computed and passed on to a minimizer which in turn has access to a procedure which returns information about the current estimate. Given that an initial estimate of the factor loadings (the principal components of the correlation matrix) is easily computable, and that a routine to evaluate this estimate in terms of some model (e.g., function generator) exists, the only remaining difficulty lies in selecting a minimization algorithm which will accurately, certainly and efficiently guide the estimate to a solution.

Presently there exists much disagreement over the efficacy of the various methods; Fiacco and McCormick [1968] and Joreskog and Goldberger [1971] claim greater potential efficiencies with pure Newton methods, while Shanno disagrees in a private communication to McDonald, preferring instead gradient or quasi-Newton methods [Swaminathan, 1971]. This conflict could be resolved by comparing all of the methods and variations of solving identical problems in action. As there are possibly

hundreds of algorithms available, comparing them all would be a task of herculean proportions. For this reason, a project of more manageable size was undertaken.

Refinements and Constraints

A set of algorithms representative of the major categories and subject to either of the following constraints was selected for comparison:

- 1) The algorithm is
 - a) representative of its category,
 - b) reported as relatively efficient,
 - c) publically available, and
 - d) In common use (among numerical analysts);

or

- 2) The algorithm is of historical interest.

A thorough review of the literature, undertaken to identify algorithms satisfying the above constraints resulted in the selection of the following eight routines for comparison:

- NWTRAP - Joreskog and van Thillo's [1971] conditional Newton-Raphson;
- NEWTON - the author's reworking of the above algorithm;
- MINIM - van der Voort's [1972] three-step Newton;

VA06A - Powell's [1970c] quasi-Newton;
VA09A - Fletcher's [1972] quasi-Newton;
MINFUN - Shanno's [1970] quasi-Newton;
FACTOR - Browne's [1968b] Gauss-Seidel; and
SD/F-P - Gruvaeus and Joreskog's [1970]
steepest-descent and Fletcher-Powell
combination.

Simultaneously, an examination of the literature on algorithmic efficiency was initiated to select criteria for comparing the various routines. The criteria for evaluation taken from Himmelblau [1972a, 1972b] are as follows:

- 1) Robustness - success in obtaining a solution for a range of problems.
- 2) Accuracy - the degree of precision in the solution.
- 3) Number of function/gradient/Hessian evaluations.
- 4) Computer time to termination.
- 5) Amount of computer memory utilized by code and arrays.

In order to test the routines listed above using the specified evaluation criteria, it was necessary to obtain test data, preferably data which had already been subject to factor analysis. In the past, computer codes using both simulated and real data had been tested with mixed results. Hillstrom [1977], describing an evaluation of algorithms using simulated data, writes

The majority of the test problems currently used are somewhat artificial, being adaptations of problems originally designed to probe for weaknesses in optimization algorithms. This difficulty would be eased by collecting and employing test problems arising naturally in applications.

In a private communication subsequent to reviewing the prospectus for this paper, Browne (having used simulated data in his 1968a study) suggested, "to obtain a fair comparison between procedures, it would be advisable to vary p (variables) and q (factors) using different sets of empirical data." Therefore, a set of correlation matrices that had already been factor analyzed in the literature were sought. Given that computer memory requirements increase rapidly with the size of the matrices, the solicited set was to range from the smallest matrix from which multiple factors could be extracted (i.e., five variables), to a fourteen variable matrix. The set of ten matrices is shown in Table 1.

C H A P T E R I V
IMPLEMENTATION OF THE COMPARISON

Details of the Selected Routines

NWTRAP, the conditional Newton-Raphson routine written by Joreskog and van Thillo [1971] has achieved widespread distribution since it was released. Originally the kernel of a package known as UFABY3 (available from Educational Testing Service), and later as JFACTOR in Northwestern University's version of the widely-used statistical package, SPSS, this routine is now licensed by National Educational Resources as the core of EFAP, the Exploratory Factor Analysis Package. This code is widely believed to be the most efficient implementation of the unconditional factor analysis method proposed by Joreskog [1967]. NWTRAP, the driving routine of a collection of eight subroutines, contains the code necessary to produce factor analytic solutions by three different methods: Maximum Likelihood, Least Squares, and Generalized Least Squares. For the purposes of this analysis, only the ML portion of the code was exercised.

NWTRAP is organized in a rather curious form for a minimization program. Instead of following the usual form where the driving routine calls the minimizer which in turn iteratively invokes the function, gradient and Hessian generator, NWTRAP calls the generator which then invokes the minimizer. The mathematical process followed in the code, clearly defined in their 1971 Research Memorandum, is to iteratively locate the minimum of the diagonal matrix of error (referred to as uniqueness), computing a new (conditional) factor matrix at each successive conditional minimum. The final conditional factor matrix is the solution. Except for the inclusion of several counters to keep track of iterations and various evaluations, NWTRAP is program listed in the 1971 Research Memorandum.

NEWTON is the author's implementation of the NWTRAP model. In this code the minimizer/generator sequence is organized in the conventional order, where the minimizer calls the generator instead of vice-versa. The source listing (Table 13), demonstrates the straightforward nature of the N-R procedure. The routines INVS and MPYM are taken from the ESL Matrix Package [Bock and Repp, 1970] now available from National Educational Resources. MLF is the generator written by the author and corresponds to the function, gradient, and Hessian evaluator used in NWTRAP. The specific details of MLF will be described later under

Implementation Environment.

Aaby and Dempster [1974] have described several modifications to the N-R approach that increase the speed of convergence. One modification is to precede the N-R minimization process with several steepest descent iterations. The benefit derived from the initial steepest descent iterations is that the N-R algorithm is spared the task of finding its way down the relatively steep sides of the pond (to use the earlier analogy), which Category 1 routines do better anyway, and is allowed to search around in the neighborhood of the minimum. MINIM is an implementation of this idea by van der Voort and Dorpema [1972].

MINIM contains another feature designed to prevent the N-R algorithm from becoming completely disoriented when the next likely place to look for a minimum is outside the definition of the space. To use the pond analogy, the bottom is located under a cliff which protrudes out over the pond. Obviously the drop line is useless on dry land; it is therefore necessary to stop rowing and adopt an alternative strategy upon reaching the face of the cliff. In mathematical terms, the method devised by Fiacco and McCormick [1968] is to use directions that correspond to negative eigenvalues to converge to an area where the

Hessian is positive definite. The Hessian is non-positive definite when pointing to an area outside the range of the space (or surface of the pond). This situation occurs with considerable frequency in general minimization. Aaby and Dempster [1974] provide examples designed to trap unwary algorithms. Once again, with the exception of several counters, the code is exactly as shown in the original report [van der Voort & Dorpema, 1972].

VA06A is Powell's [1970c] version of the quasi-Newton method. The "quasi" part of this method's name reflects the fact that it does not actually have the Hessian available to invert and use in the calculation of the next iteration's correction to the vector being minimized. Instead, an approximate to the inverse of the Hessian is built up using only function value and gradient information. The major benefit of this approach is that the Hessian need not be evaluated. Because this calculation can be extremely time-consuming, heightened efficiency is the result. Since the release of this code, "which has the advantage that convergence is guaranteed in theory, even if no good initial estimate of the required vector of variables is available [Powell, 1970]," a number of numerical analysts have evaluated it. In the broad range of problems analyzed, VA06A appeared to perform as well or better than Fletcher's 1972 routine VA09A, and much better than the original

Davidon-Fletcher-Powell routine [Adby and Dempster, 1974].

VA09A is Fletcher's 1972 version of the quasi-Newton technique. The major reason for selecting this routine was that Fletcher demonstrated it to be superior to Powell's VA06A for locating the minimum in a number of standard test problems. VA09A is occasionally referred to as Fletcher's Switching Policy as it switches back and forth between the original DFP formula and its complement to maintain positive definiteness of the approximated Hessian [Dixon, 1972]. The basic difference between the two routines, as reported by Fletcher [1972] is that VA09A is superior to VA06A in both efficiency and reliability; VA06A is more affected by the presence of round-off errors. A complete listing of VA09A is provided in Fletcher's 1972 AERE report.

MINFUN, Shanno's version of the quasi-Newton was secured through private sources. The improvement implicit in this code, an implementation of what is now referred to as the BFS (Broyden-Fletcher-Shanno) formula [Dixon, 1972], is that a class of approximating matrices can be generated as a function of a scalar which will speed up the convergence process. In a broad range of test problems, Dixon [1972] acknowledges that the BFS formula is somewhat faster than the Fletcher Switching Policy, and greatly more efficient than the original DFP algorithm. This improvement

"effects only the length, not the direction, of the search vector at each step [Shanno and Kettler, 1970]," and hence does very little to redirect the code once it begins to head toward an undefined region.

FACTOR, Michael Browne's [1968b] Gauss-Seidel driver and subroutine package was installed without modification since the theory under which it operates is, by and large, not compatible with the other minimization schemes. Forsythe, Malcolm and Moler [1977] describe this method, known variously as the Liebman process, the Gauss-Seidel, or the method of successive displacements, as an iterative solution of sets of equations in which all of the other variables, save the one under consideration, participate. A much more complete discussion of the the method is available in Forsythe and Moler [1967]. The rationale for including this algorithm is that it was the first practical (and published) use of ML in factor analysis.

SD/F-P is the steepest-descent and Fletcher-Powell (gradient and quasi-Newton) package developed by Gruvaeus and Joreskog [1970]. This combination is an early attempt to use steepest descent and a quasi-Newton algorithm to do what van der Voort and Dorpema [1972] accomplished with MINIM. The important difference between the two approaches is that MINIM uses the Newton-Raphson and modified N-R

algorithms once the neighborhood of the minimum has been located, while SD/F-P uses an early version of the DFP formula. Although tests by various authors have demonstrated that the original DFP algorithm does not display the efficiency and guarantee of convergence so desirable in a minimizer, it is included nevertheless because of its popularity.

Indeed, SD/F-P (a shortening of the original STEDE/FLEPOW name which separately identified its two components) is a frequently used routine. Joreskog has included the routine in his Analysis of Covariance Structures (ACOVs) series of programs, his Simultaneous Factor Analysis in Several Population (SIFASP) series, as well as the two new products Confirmatory Factor Analysis with Model Modification (COFAMM) and Linear Structural Relations (LISREL III). Gruvaeus and Joreskog advocate using the SD/F-P pair whenever exact second derivatives (Hessian) cannot be provided, and recommended several steepest descent iterations with STEDE before switching over to FLEPOW for the final approach to the minimum. Unlike MINIM, the mechanics of the switching are not performed automatically, requiring intervention on the part of the user.

The Function Generator

In order to compare the minimizers which were not already provided with their own function, gradient and Hessian generators, it was necessary to produce a routine which, upon invocation, would return the desired values. Using the unrestricted case ML formulae of Joreskog and van Thillo [1971] three routines were coded: JMLFN, the function, gradient and Hessian generator (Table 18); CLAM, the conditional lambda evaluator (Table 19); and FLAM, the final lambda evaluator (Table 20). In Joreskog's notation, the factor matrix extracted for a given uniqueness is referred to as lambda. In the Joreskog and van Thillo formulation of the unrestricted factor analysis problem, only the elements of the error vector (uniqueness) are manipulated by the minimization procedure. Lambda is produced as a product of the eigenvalues and eigenvectors of the uniqueness.

Since the conditional lambda must be computed within the generator at each iteration to provide the function, gradient and Hessian values, it was necessary to provide a mechanism capable of passing the large number of parameters, arrays and constants from the driver to JMLFN without going through the minimizer. The final result was the creation of MLF, shown in Table 17. By calling each of the minimizers

with only the parameters which it needed directly (the vector being minimized, the length of that vector, and the scratch space needed within the minimizer itself) the risk of introducing spurious errors was reduced. The minimizers called MLF with the current vector and its length, receiving on return the function value, the gradient, and/or the Hessian. MLF added to this information the scratch arrays, parameters and constants in a call to the real generator, JMLFN. The amount of information returned was indexed by MLEV, a parameter equal to the category of the minimizer less one.

There are several interesting ideas buried within the NWTRAP code which deserve close inspection. The NWTRAP model is designed to avoid those problems commonly encountered in factor analysis, specifically Heywood variables and boundary violations. A Heywood variable is one that does not contribute to any specific factor and may be visualized as a trench running across the pond in the earlier analogy. Once the surveyor is located above this trench, further progress toward the actual minimum is halted. In the code, when a diagonal element of the Hessian is found to be smaller than a certain value (EPSHEY), the gradient element and the off-diagonal elements of the Hessian are set to zero, and the diagonal element is set to one. This procedure has the effect of removing that

variable from the analysis entirely.

Boundary violations are attempts by the minimization algorithm to solicit information about coordinates outside of the space under investigation. They usually occur when the minimum is near the edge of the space and the minimizer takes a step beyond the edge. JMLFN, CLAM and FLAM follow the example provided by NWTRAP by adjusting the vector passed from the minimizer if it is smaller than the criterion. The statements which begin with `BND=ALOG(.005)` constitute the code for this adjustment.

The third interesting feature of the NWTRAP code also represented in JMLFN is the fact that the exact Hessian is computed only when the largest element of the gradient is smaller than a specified criterion (`EPSXCT`). Until the minimizer is in the neighborhood of the minimum, an approximate to the Hessian is supplied. NWTRAP's reduction in computational effort, achieved by decreasing the number of times the full set of eigenvectors must be extracted, is not realized in JMLFN. Since NWTRAP stores a flag specifying whether the Hessian is to be computed or approximated in the next iteration, the choice of number of eigenvectors to use is made before any are extracted. JMLFN, basing its decision on the current iteration's gradient (evaluated after eigenvectors are extracted),

requires the full set.

An Environment for Testing

In order to compare the eight different routines in a convenient manner, they were arranged together in a three-level overlaid package, the overall structure of which is depicted in Table 5. In this package, each of the routines was supplied with the required support for testing (timing, memory management, and communication routines) in an identical fashion. In general, the form was a driver program calling a minimizer which, in turn, invoked the generator.

The root level overlay contained the driver routine for the package, FROG (Table 12); MEMORF, the memory management routine from SPSS-6000; ALLOC, DPRNT, ADDM, INVS, and LOC, from Bock and Repp's 1970 ESL subroutine package; ALLOCAT, (Table 14); and PMSL and VARMAX, from Joreskog's UFABY3. FROG is responsible for reading in all of the parameters concerning the size, number of factors, number of cases, the value of the function evaluated at the minimum, the required tolerance, and a number of routine specific adjustments referred to as 'TWEAK' parameters. FROG, upon request, invokes the specific suboverlay containing the minimizer to be tested.

The test matrices are read in by INITIAL, a suboverlay unto itself. Here the initial solution is computed by extracting the first q principal components from the input correlation matrix. The initial estimate of the vector to be minimized is computed using Joreskog and van Thillo's [1971] formula (26). These estimates were then stored on a scratch file for later access by the various minimizers.

OV1 is a subroutine library used by minimizers which call JMLFN, the author's version of Joreskog and van Thillo's function generator. The routines included in this library are JMLFN, CLAM, FLAM and the eigensystem package from UFABY3. Nested within OV1 are the routines NWTRAP, MINIM, VA06A, VA09A, MINFUN, STEDE and FLEPOW, and their drivers and associated subroutines. A sample driver, the one used to control NEWTON, is shown in Table 15. The routine issues a salutation to the console displaying the time the test began and the size of the matrix being analyzed. ALLOCAT (Table 14) is called to allocate the arrays utilized by the function generator, and the various scratch arrays needed by the minimizer are set aside by calls to the entries in the ESL ALLOC routine. Once FIREUP has read the initial solution from the disk (saved by INITIAL) and initialized all of the allocated arrays to the machine specific invalid data value, the minimizer under test is called. The minimizer will iteratively invoke the

generator until convergence is achieved or a specified maximum number of iterations is exhausted. On return, the final lambda is computed and the results are printed out for later analysis.

The driver in OV2 is organized the same as the drivers in OV1; but only NWTRAP is available to be tested within this overlay. The subroutines called by NWTRAP constitute the remainder of this overlay.

OV3 is similar to OV2 in that only one minimizer, Michael Browne's FACTOR, is present. This routine was separated from the other minimizers as the equations on which it is based are incompatible with the NWTRAP/JMLFN model.

Thus we find within each overlay a driver routine responsible for the allocation and initialization of memory, the starting of the timing clocks and the invocation of the appropriate minimizer. Once the process has reached convergence, the driver is responsible for stopping the clocks and saving the results for later inspection.

The Testing Procedure

Once all of the various routines were assembled together and the linkages verified to be error free, the testing phase of the study began. The customary manner in which minimization algorithms are compared is to specify an identical convergence criterion (epsilon) for all of the routines and let the minimizers run until convergence is reported. The word "reported" is important in this instance, as the minimizer is incapable of determining whether the coordinates reported do in fact describe the true bottom (global minimum). As suggested earlier, unless precautions are taken to eliminate Heywood variables, the minimizer could terminate in a trench located in a region quite removed from the global minimum. Also, the gradient methods could find the neighborhood of the bottom too flat to provide sufficient guidance as to the direction and distance of the true bottom. Without an external criterion on which to compare the results, the process of evaluation remains fuzzy.

An alternate strategy was therefore adopted in which it was assumed that the results reported by NWTRAP (actually UFABY3) were correct, and that the answers from this algorithm would serve as a standard by which the others could be measured. Although this compromise to a certain

degree predestines the outcome of the analysis, the effect is less deleterious than might be expected. The NWTRAP code has, over a period of seven years, been exhaustively tested by the author and found to be comparable in every instance to other, correctly implemented factor analysis programs. Given that the function generator used by all but one of the other minimization codes would be JMLFN, the author's implementation of the generator used in NWTRAP, this small compromise appeared reasonable.

Thus the process used was first to run the matrix through UFABY3, and using the likelihood ratio test within it, determine the optimum number of factors. Using NWTRAP, the matrix was then analyzed to ascertain those values that described the minimum (i.e., function value, lambda and uniqueness at the minimum). Using the function value thus provided by NWTRAP as the termination criteria, the matrix was reanalyzed by each of the remaining routines. When the absolute difference between the function value calculated by the running program and the one supplied from NWTRAP was smaller than $1.0E-7$, the routine was judged to have converged to the true solution. The $1.0E-7$ value was selected after the author discovered that when function values produced by the different routines corresponded to seven places, the lambdas and uniquenesses corresponded to a minimum of five places. Thus the degree of comparability

between two factor solutions was between one and a half and two decimal places shy of the match between the final values of the function. This serendipitous heuristic markedly simplified the final evaluation. Because the final answer is supplied at the beginning and the individual routines run until they are within the $1.0E-7$ band around the identified minimum, information which would be useful in a practical, non-testing, situation must be collected. Therefore, once the routine has reached its termination point, the local epsilon is computed and saved. This value, the largest element of the gradient computed during the final iteration, would be specified as the stopping criterion were the evaluated routines to be included in a general analysis package.

C H A P T E R V
RESULTS OF THE COMPARISON

Interpretation of Tabular Results

After each of the matrices had been run through all of the routines, the statistics collected by the driving program were organized in tabular form. These statistics included the value of the function and the epsilon at the reported minimum, the number of CPU seconds expended, the amount of memory used by program and arrays, the number of reported iterations, and the actual number of function, gradient and Hessian evaluations. The statistics are organized into a set of routine-by-test matrix tables, where the rows are the routines, and the columns are the matrices. The only exception to this procedure is SD/F-P. Because there are two distinct components to the SD/F-P package, the two parts, STEDE and FLEPOW, are broken out as separate lines. The lines identified as SD/F-P represent either the final value or the sum of the values of the two, whichever is appropriate.

Table 2 reports the value of the likelihood function,

evaluated at the reported minimum. Since all of the routines tested were to match the value given in the first row to at least 6 places (the internal criterion was .000001), values smaller than those given for NWTRAP represent improvements upon NWTRAP's performance. The cells in which the value is larger than that shown for NWTRAP represent failures by the routines to provide an adequate solution. A surprising result, noticed by Himmelblau [1972] as well, is that "the algorithms tended to cluster into groups." The Newton algorithms, NWTRAP, NEWTON, and MINIM, formed one group, while the quasi-Newtons, VA06A, VA09A, MINFUN and SD/F-P, formed another cluster. Browne's FACTOR appears to be consistently closer to the Newton-Raphson cluster. This tendency is most apparent in the 7 X 3 matrix, the SES Differences data. In this instance, it appears likely that the solution is confused by a Heywood variable which is filtered out by both the Newton techniques and the Gauss-Seidel. Since the quasi-Newton routines did not require the evaluation of the Hessian, the code for which contained the correction for Heywood variables, these routines were not notified of a condition which would preclude convergence to global minimum.

Table 3 details the actual CPU times (on a CDC 6400 using RUN FORTRAN). Specifically excluded from these times are all printing, peripheral processing and system

manipulation times, so that the times listed represent the number of seconds required to obtain reported convergence without interruption of any sort. It is here that the differences among the routines becomes apparent with the Newton procedures having a clear margin over all competitors. Of the quasi-Newton group, although VA09A appears to be the fastest, it should be remembered that it did not achieve the same accuracy as VA06A in Table 2. FACTOR turned in reasonably impressive times in these trials, being no worse than an order of magnitude from the best time. The poor performance of SD/F-P provides a ready explanation of why the current releases of Joreskog's programs incorporating SD/F-P allow the user to supply an upper bound for CPU time as one of the termination criteria. While usually terminating in less time than SD/F-P, MINFUN was a little slower than the other quasi-Newton routines, and much slower than the pure Newton procedures.

While the gradient and Hessian epsilon for NWTRAP, EPS and EPSE, were preset at .005 and .1 respectively, the values suggested by Joreskog and van Thillo [1971], the value reported in Table 4 as the epsilon value for each of the other routines is nothing more than the largest element present in the gradient during the final iteration. Because of the extreme variation in epsilon both within and across routines, this statistic is meaningless. That this

statistic was not consistent was extremely disappointing, for without a consistent criteria, the problems in implementing a stand-alone program using one of these routines are greatly compounded.

Tables 5 and 6 must be examined together for it is the sum of the value in the two tables that determines the amount of memory required to analyze a specific size matrix. Table 5 specifies the amount of physical memory required to execute the various levels of the overlay tree in decimal words (60 bit), and indicates that the difference among the eight routines in this dimension is very small. Table 6 illustrates that FACTOR requires the least amount of scratch space, while VA06A requires the most. This difference is not great, amounting to little more than a two to one ratio.

Table 7 is a list of reported iterations, and demonstrates one of the major obstacles in evaluating competing routines. To interpret this table correctly, it is necessary to look at the four tables which follow. The answers range from slight over-reporting of the number of evaluations to very great under-reporting. Keeping in mind that the most expensive portion of the minimization process, if the function is complicated to evaluate, is the invocation of the various levels of the function generator, it is necessary to count the actual number of times each

level of this generator is exercised. The counts thus collected that varied widely from the reported iterations value were common. One explanation of this discrepancy is the 'interior/exterior' iterations argument. The argument contends that 'interior' (or sub-) iterations are made along a vector that was selected by an 'exterior' iteration, and only the latter was counted. While this counter is then useful for estimating the interior/exterior iteration ratio, it serves only to confuse the current analysis. Tables 8 and 9 give the actual function and gradient evaluation counts for the problems. FACTOR has no numbers displayed as the counters within it recorded entirely different processes and the values were not comparable. Tables 10 and 11 display the number of approximate and exact Hessian calculations. Obviously, only those procedures which used this information were listed. Across all of the tables, it is apparent that the difference among the Newton procedures is not great, while the quasi-Newton procedures are spread over a wide range.

Comments on Individual Routines

Examination of the results discussed thus far can give only a partial view of the performance of the individual routines. There are many more variables which might have been considered for tabling. However, there is an absence

of generally accepted criteria quantifying the relative ease of installation, testing and evaluation of the various routines. Hence, the following series of comments point out the more salient advantages or disadvantages of the tested routines.

NWTRAP, the first routine evaluated, performed excellently and without apparent failure. While the overall size of the routine could be reduced by excising the code for the two methods not utilized in the current study, (unweighted and generalized least squares), the decrease in size would not be sufficient to compensate for the loss of generality in a very workable program. Were no other criteria available, the speed with which NWTRAP achieves convergence alone would suffice to recommend it.

NEWTON, being the equivalent of NWTRAP with the two unused methods removed, performed almost as well as NWTRAP. The difference between these two routines in CPU time is almost entirely attributable to the inability of the CDC RUN FORTRAN object compiler to produce efficient code for arrays with multiple dimensions. NWTRAP addressed all arrays as vectors with one subscript, and NEWTON used multiple subscripts.

Even though it was not written specifically for the

current problem, MINIM ran consistently well in time trials, moving rapidly toward the minimum and even beating NEWTON on two matrices. Of five general purpose minimization routines evaluated, MINIM, VA06A, VA09A, MINFUN and SD/F-P, MINIM required the smallest amount of array scratch space, and coincidentally converged in the smallest overall CPU time.

When compared to VA09A, the slightly poorer performance of VA06A in both speed and memory requirement put it in second place behind the routine written by Fletcher.

VA09A was the first of two routines that required some intervention in what was designed to be an automatic process. Specifically, upon switching from the DFP to its complement near the predefined minimum, VA09A decided that the minimum had been passed and exited. This occurrence should not be considered a flaw as VA09A was not designed to be informed externally that the minimum had been reached. In initial testing of this routine, before the scheme using the answer provided by NWTRAP was adopted, VA09A moved swiftly and consistently to a minimum in the neighborhood of the NWTRAP solution.

MINFUN was the other routine requiring outside intervention, but on a larger scale. As MINFUN moved closer toward the known minimum, the scalar used to adjust the

metric of the process became progressively smaller, eventually becoming zero. At this point a divide by zero check was activated and the program halted. In order to produce the results in the tables an alternative strategy was adopted. First the routine was given a tolerance band around the known minimum sufficiently large that the routine could not fail to report convergence. Then this band was tightened one decimal place at a time until the divide check occurred. At this point the criterion was modified until a decrease in the seventh place exceeded the bound beyond which a divide by zero was attempted. This procedure accounts for the differing values of the function reported by MINFUN in Table 2.

FACTOR performed much better than the author had anticipated or the literature had predicted. The actual factor solutions produced by FACTOR were not inconsistent with those produced by NWTRAP, although the handling of Heywood variables resulted in differing values of the function at termination. Unlike NWTRAP, which removes a Heywood variable from any iteration in which it is detected, FACTOR permanently discards a Heywood variable once it is encountered. Considering the length of time this routine has been available to factor analysts, it is surprising that it has not received far greater use. FACTOR's memory requirements were the smallest of the entire set, and the

CPU times were comparable to the quasi-Newton group.

The overall performance of SD/F-P was disappointing beyond expectation. It was not anticipated that the set of routines currently utilized by the most sophisticated factor analysis software on the market would converge so slowly. On first discovering the large number of iterations required by SD/F-P, the author assumed that a criterion value had been mis-specified, and all values were rechecked. When manipulating the values of the criterion failed to improve the performance, the routines were driven separately to ascertain whether the steepest descent routine was necessary. After several attempts in which FLEPOW demonstrated clearly the necessity of having a few steepest descent iterations to move it into the neighborhood of the minimum, the effort was abandoned.

C H A P T E R V I
I M P L I C A T I O N S O F T H E S T U D Y

General Conclusions

In reviewing the tables and the discussion presented above, it becomes clear that there are major differences among the various algorithms and among the routines which implement those algorithms. These differences range from small, in the case of computer memory requirements, to very great, in the case of speed and accuracy of convergence. It is therefore imperative for an analyst, upon deciding to utilize a nonlinear minimization routine in a factor analysis program, to select the desired routine with care.

In those instances in which the second order derivatives can be evaluated directly, the appropriate choice would be one of the Newton routines. The added features present in MINIM make it an ideal candidate for selection. When exact second derivatives cannot be evaluated directly, the appropriate choice is one of the quasi-Newton routines. In this study, Fletcher's VA09A was demonstrated to be no worse than the other tested

quasi-Newton routines. In most problems, VA09A was shown to be superior in both speed and memory requirements.

An interesting outcome of the current study is the finding that Michael Browne's FACTOR routine, the first publicly available ML factor analysis program, is equally fast or faster than any quasi-Newton routine. NWTRAP, Joreskog's successor to Browne's FACTOR, was demonstrated to be the fastest of all routines tested on every problem analyzed.

While the efficiency of NWTRAP was anticipated, the slowness of SD/F-P was not. Programs that utilize the relatively inefficient SD/F-P combination are widely available, which implies large sums of computer time are needlessly being wasted. In any environment where computer time is not available without charge, this inefficiency could discourage the general acceptance of these extremely useful programs.

Recommendations for Future Research

As mathematical statisticians provide new models for factor analysis such as multi-group longitudinal factor analysis, and as numerical analysts continue to produce ever more efficient minimization techniques, the need to evaluate

the match between theory and tools increases. It is imperative that statisticians become aware of advances in numerical analysis to select the optimal tools with which to implement their models. Therefore, upon defining a new model (and those constraints which are necessary to avoid Heywood variables, etc.), the analyst would find it beneficial to review the algorithms available, selecting an appropriate routine from those that appear at least equal to MINIM or VA09A in cost/performance.

Table 1: Matrices Analyzed

<u>Variables</u>	<u>Factors</u>	<u>Source</u>
5	2	Five Psychophysical Measurements, Lawley and Maxwell [1971], p. 19.
6	2	Six School Subject Correlations, Lawley and Maxwell [1971], p. 66.
7	3	SES Differences on Learning, Green and Rohwer [1971], p. 606.
8	3	Eight Physical Variables, Harmon [1967], p. 222.
9	3	Emmett's Nine Variables, Lawley and Maxwell [1971], p. 43.
10	4	Maxwell's Ten Variates, Lawley and Maxwell [1971], p. 44.
11	4	First 11 of 14 Rating Scales, Mulaik [1972], p. 11.
12	4	IQ and Achievement Tests on 5,495 Students, Crano, Kenny and Campbell [1972], (1st 12), p. 264-5.
13	4	Last 13 of 24 Psychological Tests, Harmon [1967], p. 125.
14	4	Seven Point Scale on 225 Trainees, Mulaik [1972], p. 11.

Table 2: F at the Reported Minimum

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	.026781	.010867	.034183	.076412	.035017
NEWTON	.026781	.010867	.034191	.076412	.035017
MINIM	.026781	.010867	.034183	.076412	.035017
VA06A	.026781	.010867	.026184	.076412	.035017
VA09A	.026785	.010867	.026619	.076611	.035017
MINFUN	.026781	.010867	.026185	.076431	.035017
FACTOR	.026687	.010868	.034016	.075706	.035017
SD/F-P	.026781	.010867	.026184	.076413	.035017
STEDE	.033960	.011172	.036396	.082789	.035703
FLEPOW	.026781	.010867	.026184	.076413	.035017
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	.022848	.117335	.045798	.226498	.225907
NEWTON	.022848	.117342	.045798	.226498	.225907
MINIM	.022848	.117342	.045798	.226498	.225908
VA06A	.022848	.117436	.045798	.226498	.225908
VA09A	.022877	.118189	.045800	.226502	.226732
MINFUN	.022893	.117435	.045845	.226517	.226454
FACTOR	.022799	.110718	.045797	.226397	.220089
SD/F-P	.022848	.117301	.045798	.226498	.225909
STEDE	.029125	.128491	.048988	.233073	.237522
FLEPOW	.022848	.117301	.045798	.226498	.225909

Table 3: CPU Seconds to Reported Convergence

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	.378	.254	.740	.688	.790
NEWTON	.444	.326	1.106	.900	.884
MINIM	.744	.446	1.054	1.100	.978
VA06A	1.276	.870	22.100	3.820	3.106
VA09A	.994	.590	2.406	2.124	1.910
MINFUN	4.316	1.074	6.427	7.952	3.040
FACTOR	.904	.260	3.850	2.330	.902
SD/F-P	8.898	.888	58.976	25.620	2.726
STEDE	1.284	.292	3.726	2.650	.852
FLEPOW	7.614	.596	55.250	22.970	1.874
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	1.592	3.424	2.650	2.258	3.030
NEWTON	2.204	4.130	4.120	4.690	4.212
MINIM	2.456	3.956	4.358	4.842	4.254
VA06A	8.010	27.272	10.520	19.714	21.168
VA09A	5.170	3.058	5.512	7.064	4.498
MINFUN	13.648	14.612	6.316	12.028	6.300
FACTOR	10.718	6.126	5.268	9.930	4.348
SD/F-P	119.080	23.860	103.170	56.880	36.364
STEDE	3.476	8.996	6.616	6.008	15.492
FLEPOW	115.604	14.864	96.554	50.872	20.872

Table 4: Epsilon at Reported Minimum

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	.005000	.005000	.005000	.005000	.005000
NEWTON	.000581	.000004	.005893	.000011	.000051
MINIM	.000501	.000000	.000656	.000000	.000011
VA06A	.000000	.000000	.000000	.000000	.000000
VA09A	.000000	.000000	.030367	.002709	.000000
MINFUN	.002550	.000785	.838437	.028934	.002885
FACTOR	.000000	.000000	.000000	.000000	.000000
SD/F-P	.000095	.000083	.000030	.000742	.000031
STEDE	.500000	.500000	.500000	.500000	.500000
FLEPOW	.000095	.000083	.000030	.000742	.000031
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	.005000	.005000	.005000	.005000	.005000
NEWTON	.000104	.004389	.000771	.000019	.000002
MINIM	.000003	.000109	.000402	.000003	.000001
VA06A	.000000	.000027	.000000	.000000	.000020
VA09A	.000538	.177276	.000032	.000397	.012474
MINFUN	.098173	.073181	.094060	.406375	.307275
FACTOR	.000000	.000000	.000000	.000000	.000000
SD/F-P	.000049	.004183	.000117	.000102	.003797
STEDE	.500000	.500000	.500000	.500000	.500000
FLEPOW	.000049	.004183	.000117	.000102	.003797

Table 5: Memory Required for Program (Words)

<u>ROUTINE</u>	<u>LEVEL 0</u>	<u>LEVEL 1</u>	<u>LEVEL 2</u>	<u>TOTAL</u>
MAIN	10013			10013
INITIAL		1855		11868
OV1		1372		11385
NEWTON			665	12050
MINIM			1305	12690
VA06A			1326	12711
VA09A			879	12264
MINFUN			1231	12616
SD/F-P			1356	12741
OV2 - NWTRAP		2627		12640
OV3 - FACTOR		2223		12236

Table 6: Memory Required for Arrays (Words)

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	116	151	197	241	289
NEWTON	126	166	218	269	325
MINIM	141	184	239	293	352
VA06A	181	241	316	393	478
VA09A	136	178	232	285	343
MINFUN	166	217	281	345	415
FACTOR	87	111	153	184	218
SD/F-P	136	178	232	285	343
STEDE	136	178	232	285	343
FLEPOW	136	178	232	285	343
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	351	408	469	534	603
NEWTON	396	463	535	612	694
MINIM	426	496	571	651	736
VA06A	581	683	793	911	1037
VA09A	416	485	559	638	722
MINFUN	501	584	673	768	869
FACTOR	275	316	360	407	457
SD/F-P	416	485	559	638	722
STEDE	416	485	559	638	722
FLEPOW	416	485	559	638	722

Table 7: Number of Iterations Reported

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	10	4	9	6	5
NEWTON	9	4	11	6	4
MINIM	8	3	6	5	3
VA06A	30	14	281	35	22
VA09A	25	11	31	18	16
MINFUN	27	7	26	16	9
FACTOR	60	5	125	40	10
SD/F-P	44	9	100	40	13
STEDE	20	3	31	20	4
FLEPOW	24	6	69	20	9

	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	8	12	9	7	7
NEWTON	8	10	9	8	7
MINIM	7	9	8	7	6
VA06A	46	131	40	64	58
VA09A	31	11	19	22	10
MINFUN	22	15	9	14	6
FACTOR	130	150	40	35	45
SD/F-P	78	46	49	33	32
STEDE	14	39	17	15	28
FLEPOW	64	7	32	18	4

Table 8: Actual Function Evaluations

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	10	4	10	6	5
NEWTON	9	4	11	6	4
MINIM	9	4	8	6	4
VA06A	30	14	281	35	22
VA09A	31	11	37	24	16
MINFUN	152	22	78	102	28
FACTOR	0	0	0	0	0
SD/F-P	299	16	1051	331	23
STEDE	39	5	60	32	7
FLEPOW	260	11	991	299	16
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	8	13	9	7	7
NEWTON	8	10	9	8	7
MINIM	8	11	9	8	7
VA06A	46	131	40	64	58
VA09A	36	17	25	27	14
MINFUN	105	93	30	49	21
FACTOR	0	0	0	0	0
SD/F-P	951	149	526	245	129
STEDE	26	56	33	24	55
FLEPOW	925	93	493	221	74

Table 9: Actual Gradient Evaluations

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	10	4	10	6	5
NEWTON	9	4	11	6	4
MINIM	9	4	8	6	4
VA06A	30	14	281	35	22
VA09A	31	11	37	24	16
MINFUN	152	22	78	102	28
FACTOR	0	0	0	0	0
SD/F-P	299	16	1051	331	23
STEDE	39	5	60	32	7
FLEPOW	260	11	991	299	16
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	8	13	9	7	7
NEWTON	8	10	9	8	7
MINIM	8	11	9	8	7
VA06A	46	131	40	64	58
VA09A	36	17	25	27	14
MINFUN	105	93	30	49	21
FACTOR	0	0	0	0	0
SD/F-P	951	149	526	245	129
STEDE	26	56	33	24	55
FLEPOW	925	93	493	221	74

Table 10: Actual Approximate Hessian Evaluations

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	4	2	5	3	2
NEWTON	3	2	7	4	2
MINIM	3	2	6	4	2
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	4	4	5	5	4
NEWTON	4	5	4	3	5
MINIM	4	6	4	3	5

Table 11: Actual Exact Hessian Evaluations

	<u>5 X 2</u>	<u>6 X 2</u>	<u>7 X 3</u>	<u>8 X 3</u>	<u>9 X 3</u>
NWTRAP	5	1	3	2	2
NEWTON	6	2	4	2	2
MINIM	6	2	2	2	2
	<u>10 X 4</u>	<u>11 X 4</u>	<u>12 X 4</u>	<u>13 X 4</u>	<u>14 X 4</u>
NWTRAP	3	7	3	1	2
NEWTON	4	5	5	5	2
MINIM	4	5	5	5	2

Table 12: Source Listing of FROGS (Main Driver)

```

OVERLAY(FROG,0,0)
PROGRAM FROG (INPUT,OUTPUT,PUNCH,TAPE1,TAPE5=INPUT,
1          TAPE6=OUTPUT)
C
C   A IS THE SPACE BEYOND THE END OF A GIVEN OVERLAY, USED
C   AS SCRATCH FOR THE ARRAYS NEEDED IN THAT OVERLAY...
C
COMMON A(1)
C
C   POINTERS TO WHERE, RELATIVE TO THE FIRST WORD OF -A-
C   IN BLANK COMMON, THE VARIOUS ARRAYS ARE LOCATED...
C   ALSO MAINTAINS THE VALUES OF THE LAST AVAILABLE WORD
C   OF -A-, AND THE AVAILABLE CM REGISTER FOR MEMORF...
C
COMMON /INDEX/ S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1          H,FREEWDS,LAST,MINCM,CURCM
          INTEGER S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1          H,FREEWDS,LAST,MINCM,CURCM
C
C   MAINTAINS THE RUN TITLE, THE NAME AND NUMBER OF THE
C   ROUTINE BEING TESTED, THE TIME AND FINAL F, AS WELL AS
C   ALL OF THE KOUNTERS...
C
COMMON /OUTPUT/ IHEAD(8),NAMER,NPROG,NPROB,ITERS,TIME,
1          FINALF,F(4),JFMT(2)
C
C   PARAM DEFINES THE VARIOUS PARAMETERS (ORDER OF MATRIX,
C   NUMBER OF FACTORS TO BE EXTRACTED, ETC...)
C
COMMON /PARAM/ N,IP,IP2,IPP,IQ,IPQ,IPMQ,EPSHEY,EPSXCT,
1          IPARAT
C
C   NAMES OF ALL OF THE AVAILABLE OVERLAYS, AND POINTERS
C   TO WHERE (NUMBERS) THEY ARE TO BE FOUND...
C
COMMON /NEEDED/ NOVL,NSEG,NOVLTAB(3,15)
C
C   DEFINES THE INPUT AND OUTPUT UNITS
C
COMMON /UNITS/ IOCR,IOLP
C
DIMENSION LIST(15)
COMMON /MATCHF/ FTOBEAT,VARIOUS(4),TOLRNCE
COMMON /TWEAK/ NTWEAK,TWEAK(7)
DATA (IOCR=5),(IOLP=6)
DATA (JFMT=10H(5X,15,5X,,8H10F11.5))

```

```

DATA (NOVLTAB =
1   10HNEWTON      ,1,02B,   10HMINIM      ,1,03B,
2   10HVA06A      ,1,04B,   10HMINFUN   ,1,05B,
3   10HFLEPOW     ,1,06B,   10HNELMIN  ,1,10B,
4   10HVA09A      ,1,11B,   10HVA10A   ,1,12B,
5   10HWNTRAP     ,2,00B,   10HFACTORB ,3,01B,
6   10HFACTOR     ,3,02B,   12(0)      )

```

```

C
C   READ IN OVERALL PARAMETERS AND NAMES OF ROUTINES TO
C   EXERCIZE IN THIS RUN
C

```

```

NPROB=0
READ 5, LOOK,NLIST,MINCM,EPSHEY,EPSXCT,IPARAT,
1   (LIST(I),I=1,NLIST)
5  FORMAT (I1,I3,06,2F10.0,I3/(8A10))
PRINT 8, LOOK,NLIST,MINCM,EPSHEY,EPSXCT,IPARAT,
1   (LIST(I),I=1,NLIST)
8  FORMAT (*1INITIAL PARAMETERS:*//
1   * LOOK = *,I1/
2   * NLIST = *,I2/
3   * MINCM = *,06/
4   * EPSHEY = *,1F10.6/
5   * EPSXCT = *,1F10.6/
6   * IPARAT = .*,I3/
7   * ALGORITHMS: *,A10/
8   (13X,A10))

```

```

C
C   SET DEBUGGING FLAG IF NEEDED
C

```

```

IF (LOOK.EQ.1) CALL SLITE(4)

```

```

C
C   READ IN LIST OF ROUTINE SPECIFIC ADJUSTMENTS
C

```

```

READ 7, NTWEAK,TWEAK
7  FORMAT (A10,7F10.0)
PRINT 9, NTWEAK,TWEAK
9  FORMAT (//* TWEAKING *,A10,*WITH*,7F10.5)

```

```

C
C   MAIN LOOP - ONCE FOR EACH PROBLEM
C

```

```

10 NPROB=NPROB+1
   READ (5,20) IHEAD,N,IP,IQ,FTOBEAT,TOLRNCE
20  FORMAT (8A10/I4,2I2,2F10.0)
   IF (EOF,5) 80,30

```

```

C
C   CONSTANT INITIALIZATION FOR THIS MATRIX...
C

```

```

30  IP2=(IP*(IP+1))/2
    IPQ=IP*IQ
    IPP=IP*IP

```

```
      IPMQ=IP-IQ
C
C   FETCH THE REQUIRED AMOUNT OF MEMORY AND COMPUTE THE
C   INITIAL SOLUTION (PRINCIPAL COMPONENTS)
C
      IF (MEMORF(3,CURCM).LT.MINCM) CALL MEMORF (0,MINCM)
      CALL OVERLAY(4HFROG,4,0)
C
C   CALL EACH OF THE ROUTINES DESIRED (IN THE ORDER THEY
C   WERE SPECIFIED ON THE PARAMETER CARD)
C
      DO 70 I=1,NLIST
      DO 40 J=1,11
      IF (LIST(I).NE.NOVLTAB(1,J)) GO TO 40
      NOVL=NOVLTAB(2,J)
      NSEG=NOVLTAB(3,J)
      IF (MEMORF(3,CURCM).LT.MINCM) CALL MEMORF (0,MINCM)
      CALL OVERLAY(4HFROG,NOVL,0)
      GO TO 70
40 CONTINUE
C
C   DID NOT RECOGNIZE DESIRED ROUTINE...
C
50 PRINT 60, LIST(I)
60 FORMAT (*1UNKNOWN ROUTINE: *,A10)
70 CONTINUE
   GO TO 10
C
C   ALL FINISHED - ISSUE A STOP
C
80 STOP 55
   END
```

Table 13: Source Listing of INITIAL

```

C      SUBROUTINE INITIAL (IP2,IP,IQ,N,S,A,V,STORE)
C
C      MAINTAINS THE RUN TITLE, THE NAME AND NUMBER OF THE
C      ROUTINE BEING TESTED, THE TIME AND FINAL F, AS WELL AS
C      ALL OF THE KOUNTERS...
C
C      COMMON /OUTPUT/ IHEAD(8),NAMER,NPROG,NPROB,ITERS,TIME,
1      FINALF,NF(4),JFMT(2)
C      DIMENSION S(IP2),A(IP,IQ),V(IP),STORE(1),IFMT(8)
C
C      PRINT OUT THE HEADER AND INITIAL ESTIMATES...
C
C      PRINT 10, IHEAD,N,IP,IQ
10  FORMAT (1H1,24X,8A10//25X,17HNUMBER OF CASES =,I5,
1      8X,17HORDER OF MATRIX =,I3,
2      8X,19HNUMBER OF FACTORS =,I3)
C
C      READ INPUT FORMAT AND MATRIX
C
C      READ (5,20) IFMT
20  FORMAT (8A10)
C      READ (5,IFMT) (S(I),I=1,IP2)
C      REWIND 1
C      WRITE (1) (S(I),I=1,IP2)
C      CALL PMSL(IP,IP,S,JFMT,21HMATRIX TO BE ANALYZED,0,3,1)
C
C      EIGENVECTORS AND EIGENVALUES FOR COMPONENTS SOLUTION
C      CHAR RETURNS ROOTS AND VECTORS FOR THE SPECIFIED
C      NUMBER OF VARIABLES
C
C      DO 33 I=1,IP2
33  STORE(I)=S(I)
C      CALL CHAR (S,IP,V,A,3,IQ,0,3,IQ,0)
C      DO 40 I=1,IQ
40  V(I)=SQRT(V(I))
C      DO 50 I=1,IP
C      DO 50 J=1,IQ
50  A(I,J)=A(I,J)*V(J)
C      CALL PMSL (IP,IQ,A,JFMT,16HINITIAL SOLUTION,0,2,0)
C
C      COMPUTE COMMUNALITY ESTIMATES
C
C      CALL INVS (STORE,IP,DETS,V)
C      FT=1.0-FLOAT(IQ)/(2.0*FLOAT(IP))
C      K=0
C      DO 70 I=1,IP

```

```
K=K+I
70 V(I)=ALOG(FT/STORE(K))
   PRINT 80, (V(I),I=1,IP)
80 FORMAT (1H0,10X,23HINITIAL ESTIMATE OF PSI//
1       (15X,10F11.5))
```

C
C
C

```
SAVE THE INITIAL ESTIMATES ON DISK FOR ALL PROBLEMS
```

```
WRITE (1) A,V
WRITE (1) (STORE(I),I=1,IP2)
RETURN
END
```

Table 14: Source Listing of ALLOCAT

```

C      SUBROUTINE ALLOCAT (MLEV)
C
C      ALLOCATES CORE FOR JMLFN (AND MAKES CERTAIN THAT THERE
C      IS ENOUGH LYING AROUND FOR NWTRAP AND BROWNS ROUTINES
C      TO WORK).
C
C      A IS THE SPACE BEYOND THE END OF A GIVEN OVERLAY, USED
C      AS SCRATCH FOR THE ARRAYS NEEDED IN THAT OVERLAY...
C
C      COMMON A(1)
C
C      POINTERS TO WHERE, RELATIVE TO THE FIRST WORD OF -A-
C      IN BLANK COMMON, THE VARIOUS ARRAYS ARE LOCATED...
C      ALSO MAINTAINS THE VALUES OF THE LAST AVAILABLE WORD
C      OF -A-, AND THE AVAILABLE CM REGISTER FOR MEMORF...
C
C      COMMON /INDEX/ S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1      H,FREEWDS,LAST,MINCM,CURCM
C      INTEGER S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1      H,FREEWDS,LAST,MINCM,CURCM
C
C      MAINTAINS THE RUN TITLE, THE NAME AND NUMBER OF THE
C      ROUTINE BEING TESTED, THE TIME AND FINAL F, AS WELL AS
C      ALL OF THE KOUNTERS...
C
C      COMMON /OUTPUT/ IHEAD(8),NAMER,NPROG,NPROB,ITERS,TIME,
1      FINALF,F(4),JFMT(2)
C
C      PARAM DEFINES THE VARIOUS PARAMETERS (ORDER OF MATRIX,
C      NUMBER OF FACTORS TO BE EXTRACTED, ETC...)
C
C      COMMON /PARAM/ N,IP,IP2,IPP,IQ,IPQ,IPMQ,EPSHEY,EPSXCT,
1      IPARAT
C
C      GET SIZE OF AVAILABLE MEMORY...
C
C      CALL MEMORF (4,LWA)
C      FREEWDS=140000B-LWA
C      LAST=1
C
C      ALLOCATE THE SPACE FOR SIGMA, LAMBDA AND PSI...
C
C      CALL ALLOC (4,LAST,FREEWDS)
C      CALL ALLOC4 (S,IP,IP,1,LAM,IP,IQ,0,PSI,IP,IP,2,LAST,1,
1      1,0)
C      IF (MLEV.LT.-1) RETURN

```



```
C
C
C   ALLOCATE CORE FOR FUNCTION COMPUTATION
      CALL ALLOC (7, LAST, FREEWDS)
      CALL ALLOC5 (CLM, IP, IP, 1, VAL, IP, 1, 0, D1, IP, 1, 0, D2, IP, 1,
1         0, S1, IP, 1, 0)
      CALL ALLOC2 (S2, IP, 1, 0, LAST, 1, 1, 0)
      IF (MLEV.LT.0) RETURN
C
C
C   ALLOCATE CORE FOR GRADIENT COMPUTATION
      CALL ALLOC (3, LAST, FREEWDS)
      CALL ALLOC3 (VEC, IP, IP, 0, G, IP, 1, 0, LAST, 1, 1, 0)
      IF (MLEV.EQ.0) RETURN
C
C
C   ALLOCATE CORE FOR HESSIAN COMPUTATION
      CALL ALLOC (2, LAST, FREEWDS)
      CALL ALLOC2 (H, IP, IP, 1, LAST, 1, 1, 0)
C
      RETURN
      END
```

Table 15: Source Listing of OV1S2

```

PROGRAM OV1 S2
C
C
C
C
A IS THE SPACE BEYOND THE END OF A GIVEN OVERLAY, USED
AS SCRATCH FOR THE ARRAYS NEEDED IN THAT OVERLAY...

COMMON A(1)
C
C
C
C
C
C
POINTERS TO WHERE, RELATIVE TO THE FIRST WORD OF -A-
IN BLANK COMMON, THE VARIOUS ARRAYS ARE LOCATED...
ALSO MAINTAINS THE VALUES OF THE LAST AVAILABLE WORD
OF -A-, AND THE AVAILABLE CM REGISTER FOR MEMORF...

COMMON /INDEX/ S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1 H,FREEWDS,LAST,MINCM,CURCM
INTEGER S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1 H,FREEWDS,LAST,MINCM,CURCM
C
C
C
C
C
C
MAINTAINS THE RUN TITLE, THE NAME AND NUMBER OF THE
ROUTINE BEING TESTED, THE TIME AND FINAL F, AS WELL AS
ALL OF THE KOUNTERS...

COMMON /OUTPUT/ IHEAD(8),NAMER,NPROG,NPROB,ITERS,TIME,
1 FINALF,F(4),JFMT(2)
C
C
C
C
C
PARAM DEFINES THE VARIOUS PARAMETERS (ORDER OF MATRIX,
NUMBER OF FACTORS TO BE EXTRACTED, ETC...)

COMMON /PARAM/ N,IP,IP2,IPP,IQ,IPQ,IPMQ,EPSHEY,EPSXCT,
1 IPARAT
C
C
C
C
NEWTON-RAPHSON - DIRECT SOLUTION

NPROG=1
NAMER=10HNEWTON
CALL SAYHI(NAMER,IP,IQ)
EPS=.0005
MAXIT=IP**3
MLEV=1
CALL ALLOCAT (MLEV)
CALL ALLOC (2,LAST,FREEWDS)
CALL ALLOC2 (LSCR,IP,2,0,LAST,1,1,0)
CALL FIREUP (IP2,IPQ,IP,A(S),A(LAM),A(PSI),MLEV,LAST)
CALL NEWTON (IP,A(PSI),FINALF,A(G),A(H),A(LSCR),EPS,
1 ITERS,MAXIT)
CALL FLAM (IP,IP2,IQ,A(PSI),A(S),A(CLM),A(VAL),A(LAM),
1 A(D1),A(D2),A(S1),A(S2))

```

```
CALL RESULTS (IP, IQ, A(LAM), A(PHI), A(S), LAST)  
RETURN  
END
```

Table 16: Source Listing of NEWTON

```

C      SUBROUTINE NEWTON (IP,PSI,F,G,H,XX,EPS,NIT,MAXIT)
C
C      EXERCISES THE FUNCTION GENERATOR USING NEWTONS METHOD
C
C      IP - NUMBER OF VARIABLES
C
C      PSI - THE INITIAL ESTIMATE
C
C      F - THE FUNCTION VALUE
C
C      G - THE GRADIENT (LENGTH IP)
C
C      H - THE HESSIAN (MS=1 - LENGTH IP*(IP+1)/2)
C
C      XX - SCRATCH VECTOR OF LENGTH IP*2
C
C      EPS - CONVERGENCE CRITERION
C
C      NIT - NUMBER OF ITERATIONS COUNTER
C
C      MAXIT - MAXIMUM NUMBER OF ITERATIONS
C
C      COMMON /MATCHF/ FTOBEAT,VARIOUS(4),TOLRNCE
C      DIMENSION PSI(IP),G(IP),H(1),XX(IP,2)
C
C      TRANSFER THE INITIAL ESTIMATE TO THE SWITCHING VECTOR
C
C      DO 1 I=1,IP
1      XX(I,1)=PSI(I)
C      II=1
C      JJ=2
C
C      MINIMIZATION LOOP - USING MLFN TO COMPUTE F, G AND H
C
2      NIT=NIT+1
C      CALL MLF (IP,XX(1,II),1,F,G,H)
C      CALL INVS (H,IP,DET,PSI)
C      CALL MPYM (G,H,PSI,1,IP,0,1,IP)
C
C      CHECK FOR CONVERGENCE, COMPUTING NEW PSI IN PROCESS
C
C      ALARGE=0.0
C      DO 3 I=1,IP
C      IF (ABS(PSI(I)).GT.ALARGE) ALARGE=ABS(PSI(I))
3      XX(I,JJ)=XX(I,II)-PSI(I)
C      I=II

```

```
II=JJ  
JJ=I  
IF (NIT.GT.MAXIT) RETURN  
VARIOUS(1)=ALARGE  
IF (ABS(F-FTOBEAT).GT.TOLRNCE) GO TO 2
```

```
C  
C  
C  
C
```

```
REACHED CONVERGENCE - TRANSFER RESULT BACK TO PSI AND  
RETURN TO THE CALLING PROGRAM  
  
DO 4 I=1,IP  
4 PSI(I)=XX(I,II)  
RETURN  
END
```

Table 17: Source Listing of MLF

```

C      SUBROUTINE MLF (NN,X,MLEV,F,G,H)
C
C      MAIN CALLED ROUTINE FOR MINIMIZERS - SETS UP CALLING
C      SEQUENCE FOR JMLFN BY USING THE POINTERS IN INDEX TO
C      REFERENCE THE VARIOUS ARRAYS.
C
C      THIS ROUTINE MUST BE INSERTED AS A COMDECK WITHIN EACH
C      OVERLAY WHICH NEEDS IT TO ENABLE THE DYNAMIC MEMORY
C      ALLOCATION SCHEME...
C
C      A IS THE SPACE BEYOND THE END OF A GIVEN OVERLAY, USED
C      AS SCRATCH FOR THE ARRAYS NEEDED IN THAT OVERLAY...
C
C      COMMON A(1)
C
C      POINTERS TO WHERE, RELATIVE TO THE FIRST WORD OF -A-
C      IN BLANK COMMON, THE VARIOUS ARRAYS ARE LOCATED...
C      ALSO MAINTAINS THE VALUES OF THE LAST AVAILABLE WORD
C      OF -A-, AND THE AVAILABLE CM REGISTER FOR MEMORF...
C
C      COMMON /INDEX/ S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1         H,FREEWDS,LAST,MINCM,CURCM
C      INTEGER S,LAM,PSI,CLM,VAL,D1,D2,S1,S2,S3,VEC,G,
1         H,FREEWDS,LAST,MINCM,CURCM
C
C      MAINTAINS THE RUN TITLE, THE NAME AND NUMBER OF THE
C      ROUTINE BEING TESTED, THE TIME AND FINAL F, AS WELL AS
C      ALL OF THE KOUNTERS...
C
C      COMMON /OUTPUT/ IHEAD(8),NAMER,NPROG,NPROB,ITERS,TIME,
1         FINALF,F(4),JFMT(2)
C
C      PARAM DEFINES THE VARIOUS PARAMETERS (ORDER OF MATRIX,
C      NUMBER OF FACTORS TO BE EXTRACTED, ETC...)
C
C      COMMON /PARAM/ N,IP,IP2,IPP,IQ,IPQ,IPMQ,EPSHEY,EPSXCT,
1         IPARAT
C
C      CALL JMLFN (IP,IP2,IQ,X,MLEV,F,G,H,A(CLM),A(S),A(VAL),
1         A(VEC),NF,EPSHEY,EPSXCT,A(D1),A(D2),A(S1),
2         A(S2))
C
C      RETURN
C      END

```

Table 18: Source Listing of JMLFN

```

SUBROUTINE JMLFN (IP,IP2,IQ,X,MLEV,F,G,H,A,SINV,
1             EIGVAL,EIGVEC,NF,EPSHEY,EPSXCT,D1,
2             D2,S1,S2)

C
C COMPUTES THE MAXIMUM-LIKELIHOOD FUNCTION, THE GRADIENT
C AND THE HESSIAN, AS REQUIRED BY THE VALUE OF MLEV.
C
C IP - NUMBER OF VARIABLES
C
C IP2 - IP*(IP+1)/2 - THE LENGTH OF A, H AND S
C
C IQ - NUMBER OF FACTORS
C
C X - PSI VECTOR OF LENGTH IP (TO BE MINIMIZED)
C
C MLEV - FLAG INDICATING THE AMOUNT OF STUFF TO COMPUTE:
C   -1 = COMPUTE ONLY FUNCTION. VALUE
C   0 = COMPUTE FUNCTION AND GRADIENT
C   +1 = COMPUTE FUNCTION, GRADIENT AND HESSIAN
C
C F - THE LIKELIHOOD FUNCTION EVALUATED AT X
C
C G - THE GRADIENT OF LENGTH IP
C
C H - HESSIAN (SYMMETRIC MATRIX - IP*(IP+1)/2 - MS=1)
C
C A - THE CONDITIONAL LAMBDA
C
C SINV - THE INVERSE OF THE ORIGINAL CORRELATION MATRIX
C
C EIGVAL - VECTOR OF LENGTH IP TO HOLD EIGENVALUES
C
C EIGVEC - MATRIX (IP,IP) FOR EIGENVECTORS AS COLUMNS
C
C NF - VECTOR OF COUNTERS:
C   (1) - FUNCTION EVALUATIONS
C   (2) - GRADIENT COMPUTATIONS
C   (3) - APPROXIMATE HESSIAN COMPUTATIONS
C   (4) - EXACT HESSIAN COMPUTATIONS
C
C EPSHEY - SMALLEST LEGAL DIAGONAL ELEMENT IN HESSIAN -
C         USED TO DETECT HEYWOOD VARIABLES
C
C D1, D2, S1, S2 - SCRATCH VECTORS OF LENGTH IP
C
C DIMENSION X(IP),G(IP),H(IP2),A(IP2),SINV(IP2),

```

```

1          EIGVAL(IP),EIGVEC(IP,IP),NF(4),D1(IP),
2          D2(IP),S1(IP),S2(IP)
LOGICAL QEXACT
C
C      INITIALIZE CONSTANTS
C
      IQP1=IQ+1
      FF=0.0
C
C      COMPUTE A CONDITIONAL LAMBDA AND EXTRACT EIGENVALUES
C      AND THE REQUIRED EIGENVECTORS
C
      CALL CLAM (IP,IP2,X,SINV,A,EIGVAL,MLEV,EIGVEC,D1,D2,
1          S1,S2)
C
C      FUNCTION IS THE SUM OF RECIPROCALLS AND LOGS OF THE
C      LAST IP-IQ EIGENVALUES
C
      DO 10 I=IQP1,IP
10  FF=FF+1.0/EIGVAL(I)+ALOG(EIGVAL(I))
      F=FF-FLOAT(IP-IQ)
      NF(1)=NF(1)+1
      IF (MLEV) 150,20,20
C
C      TRANSFORM THE EIGENVALUES FOR THE GRADIENT COMPUTATION
C
20  DO 30 I=1,IP
30  D1(I)=1.0-1.0/EIGVAL(I)
C
C      ACCUMULATE PRODUCTS OF THE TRANSFORMED EIGENVALUES AND
C      THE LAST IP-IQ EIGENVECTORS AS THE GRADIENT
C      SET FLAG TO COMPUTE EXACT HESSIAN IF GRADIENT IS
C      SMALLER THAN THE SPECIFIED CRITERION
C
      QEXACT=.TRUE.
      DO 45 I=1,IP
      G(I)=0.0
      DO 40 J=IQP1,IP
40  G(I)=G(I)+D1(J)*EIGVEC(I,J)**2
      IF (ABS(G(I)).GT.EPSXCT) QEXACT=.FALSE.
45  CONTINUE
      NF(2)=NF(2)+1
      IF (MLEV) 150,150,50
C
C      DECIDE WHICH VERSION OF THE HESSIAN TO COMPUTE
C
50  IF (QEXACT) GO TO 80
C
C      COMPUTE APPROXIMATE TO THE HESSIAN - SAVES TIME
C

```



```

L=0
DO 70 I=1,IP
DO 70 J=1,I
S=0.0
DO 60 M=IQP1,IP
60 S=S+EIGVEC(I,M)*EIGVEC(J,M)
L=L+1
70 H(L)=S*S
NF(3)=NF(3)+1
GO TO 120

```

C
C
C

COMPUTATION OF EXACT HESSIAN - NOT ALWAYS NECESSARY

```

80 L=0
DO 110 I=1,IP
DO 110 J=1,I
U=0.0
DO 100 M=IQP1,IP
T=0.0
DO 90 N=1,IQ
S=(EIGVAL(M)+EIGVAL(N)-2.0)/(EIGVAL(M)-EIGVAL(N))
90 T=T+S*EIGVEC(I,N)*EIGVEC(J,N)
IF (I.EQ.J) T=T+1.0
100 U=U+EIGVEC(I,M)*EIGVEC(J,M)*T
IF (I.EQ.J) U=U-G(I)
L=L+1
110 H(L)=U
NF(4)=NF(4)+1

```

C
C
C
C
C
C

CHECK AND CORRECTION FOR HEYWOOD VARIABLES
ZERO GRADIENT AND ROW AND COLUMN OF HESSIAN
SET DIAGONAL ELEMENT TO 1.0
ADJUST THE ORIGINAL VECTOR FOR UNDESIREED ONES

```

120 L=0
BND=ALOG(.005)
DO 140 I=1,IP
D1(I)=1.0
IF (H(L+I).LT.EPSHEY) D1(I)=0.0
DO 130 J=1,I
L=L+1
IF (J.LT.I) GO TO 125
IF (D1(I).EQ.1.0) GO TO 125
X(I)=X(I)-G(I)/H(L)
IF (X(I).LT.BND) X(I)=BND
H(L)=1.0
GO TO 130
125 H(L)=H(L)*D1(I)*D1(J)
130 CONTINUE
G(I)=G(I)*D1(I)

```

```
140 CONTINUE  
C  
C     RETURN TO MAIN ROUTINE  
C  
150 RETURN  
    END
```

Table 19: Source Listing of CLAM

```

SUBROUTINE CLAM (IP,IP2,THETA,S,A,VAL,MLEV,VEC,D1,D2,
1          S1,S2)
C
C   COMPUTES A CONDITIONAL LAMBDA GIVEN A THETA
C   (CONVERTED TO PSI FOR CALCULATION)
C
C   IP - NUMBER OF VARIABLES
C
C   IP2 - IP*(IP+1)/2 - THE LENGTH OF A AND S
C
C   THETA - THE VECTOR BEING MINIMIZED (LENGTH IP)
C
C   S - INVERSE OF THE ORIGINAL CORRELATION MATRIX (MS=1)
C
C   A - SCRATCH SPACE USED TO COMPUTE CONDITIONAL LAMBDA
C
C   VAL - EIGENVALUES OF A
C
C   MLEV - THE MINIMIZATION LEVEL - IMPLIES WHETHER OR NOT
C           TO COMPUTE EIGENVECTORS (MLEV>0 MEANS COMPUTE
C           EIGENVECTORS)
C
C   VEC - EIGENVECTORS STORED AS COLUMNS
C
C   D1, D2, S1, S2 - SCRATCH VECTORS OF LENGTH IP
C
C   DIMENSION THETA(IP),S(IP2),A(IP2),VAL(IP),VEC(IP,IP),
1          D1(IP),D2(IP),S1(IP),S2(IP)
C
C   TRANSFORM THETA INTO PSI FOR COMPUTATION
C
C   BND=ALOG(.005)
C   L=0
C   DO 10 I=1,IP
C   IF (THETA(I).LT.BND) THETA(I)=BND
C   D1(I)=SQRT(EXP(THETA(I)))
C   DO 10 J=1,I
C   L=L+1
10 A(L)=D1(I)*S(L)*D1(J)
C
C   COMPUTE EIGENVALUES AND, IF NECESSARY, EIGENVECTORS
C
C   NEED=0
C   IF (MLEV.GE.0) NEED=IP
C   CALL HOUSE (IP,IP2,-1,NEED,A,VAL,VEC,D1,D2,S1,S2)
C

```

RETURN
END


```
DO 20 J=1,IQ  
20 VEC(I,J)=THETA(I)*VEC(I,J)*SQRT(1.0/VAL(J)-1.0)  
30 THETA(I)=THETA(I)*THETA(I)
```

C

```
RETURN  
END
```

Bibliography

- Adby, P. R., & Dempster, M. A. H. Introduction to optimization methods. New York: Wiley, 1974.
- Bergan, J. R., Zimmerman, B. J., & Ferg, M. Effects of variations in content and stimulus grouping on visual sequential memory. Journal of Educational Psychology, 1971, 62, 400-404.
- Brown, K. Model fitting and optimization by computer. Printed handouts from a 5-day workshop, Institute for Advanced Technology, New York, 1975.
- Browne, M. W. A comparison of factor analysis techniques. Psychometrika, 1968, 33, 267-223. (a)
- Browne, M. W. Gauss-Seidel computing procedures for a family of factor analytic solutions. Research Bulletin 68-61, Educational Testing Service, Princeton, N.J., 1968. (b)
- Browne, M. W. Fitting the factor analysis model. Psychometrika, 1969, 34, 375-394.
- Bock, R. D., & Repp, B. H. ESL matrix operations subroutines for the IBM System/360 computers. Research Memorandum Number 11, Statistical Laboratory, Department of Education, University of Chicago, 1970.
- Cauchy, A. Methode generale pour la resolution des systemes d'equations simultanees. Compt. Rend., 1847, 25, 536-538.
- Clark, M. R. B. A rapidly convergent method for maximum-likelihood factor analysis. British Journal of Mathematical and Statistical Psychology, 1970, 23(1), 43-52.
- Comrey, A. L. A first course in factor analysis. New York: Academic Press, 1973.
- Crano, W. D., Kenny, D. A., & Campbell, D. T. Does intelligence cause achievement?: a cross-lagged panel analysis. Journal of Educational Psychology, 1972, 63, 258-275.

- Davidon, W. C. Variable metric method for minimization. A.E.C. Research and Development Report ANL-5990, Argonne, 1959.
- Day, R. G. Study of a random search method for function minimization, Technical Report 70-6, Computer and Information Science Research Center, Ohio State University, Columbus, 1970.
- Dixon, L. C. W. The choice of step length, a crucial factor in the performance of variable metric algorithms. Numerical methods for non-linear optimization. New York: Academic Press, 1972.
- Edwards, A. W. F. Likelihood. Cambridge: University Press, 1972.
- Emmett, W. G. Factor analysis by Lawley's method of maximum likelihood. British Journal of Psychological Statistics, 1949, 2, 90-97.
- Fiacco, A. F., & McCormick, G. P. Nonlinear programming: Sequential unconstrained minimization techniques. New York: Wiley, 1968.
- Fletcher, R. A new approach to variable metric algorithms. Computer Journal, 1970, 13, 317-322.
- Fletcher, R. FORTRAN subroutines for minimization by quasi-Newton methods, Research Group Report AERE-R 7125, Atomic Energy Research Establishment, Harwell, Berkshire, 1972.
- Fletcher, R., & Powell, M. J. D. A rapidly convergent descent method for minimization. The Computer Journal, 1963, 2, 163-168.
- Forsythe, G. E., Malcolm, M. A., & Moler, C. B. Computer methods for mathematical computations. Englewood Cliffs, N.J.: Prentice-hall, 1977.
- Forsythe, G. E., & Moler, C. B. Computer solution of linear algebraic systems. Englewood Cliffs, N.J.: Prentice-Hall, 1967.
- Gebhardt, F. Maximum likelihood solution to factor analysis when some factors are completely specified. Psychometrika, 1971, 36, 155-163.

- Gentleman, W. M., & Johnson, S. C. Analysis of algorithms, a case study: determinants of matrices with polynomial entries. ACM Transactions on Mathematical Software, 1976, 2, 232-241.
- Gorsuch, R. L. Factor analysis. Philadelphia: Saunders, 1974.
- Gray, M. A survey of current optimization methods. Naval Ship Research and Development Center Report, Washington, 1971.
- Graybill, F. A. Introduction to matrices with applications in statistics. California: Wadsworth, 1969.
- Green, R. B., & Rohwer, W. D. SES differences on learning and ability tests in black children. American Educational Research Journal, 1971, 8, 601-609.
- Gruvaeus, G. T., & Joreskog, K. G. A computer program for minimizing a function of several variables. Research Bulletin 70-14, Educational Testing Service, Princeton, N.J., 1970.
- Guertin, W. H., & Bailey, J. P. Jr. Introduction to modern factor analysis. Ann Arbor: Edwards Brothers, 1970.
- Hamming, R. W. Introduction fo applied numerical analysis. New York: McGraw-Hill, 1971.
- Harman, H. H. Modern factor analysis. (2nd ed.) Chicago: University of Chicago Press, 1967.
- Hillstrom, K. E. A simulation test approach to the evaluation of nonlinear optimization algorithms. ACM Transactions on Mathematical Software, 1977, 3, 305-315.
- Himmelblau, D. M. Applied nonlinear programming. New York: McGraw-Hill, 1972. (a)
- Himmelblau, D. M. A uniform evaluation of unconstrained optimization techniques. Numerical methods for non-linear optimization, New York: Academic Press, 1972, 69-98. (b)
- Howe, W. G. Some contributions to factor analysis. Report #ONRL-1919, Oak Ridge National Laboratory, 1955.

- Jacoby, S. L. S., Kowalik, J. S., & Pizzo, J. T. Iterative methods for nonlinear optimization problems. New Jersey: Prentice-Hall, 1972.
- Jennrich, R. I., & Robinson, S. M. A Newton-Raphson algorithm for maximum likelihood factor analysis. Psychometrika, 1969, 34, 111-123.
- Joreskog, K. G. Some contributions to maximum likelihood factor analysis. Psychometrika, 1967, 32, 443-482.
- Joreskog, K. G. A general approach to confirmatory maximum likelihood factor analysis. Psychometrika, 1969, 34, 183-202.
- Joreskog, K. G., & Goldberger, A. S. Factor analysis by generalized least squares. Research Bulletin 71-26, Educational Testing Service, Princeton, N.J., 1971.
- Joreskog, K. G., & van Thillo, M. New rapid algorithms for factor analysis by unweighted least squares, generalized least squares and maximum likelihood. Research Memorandum 71-5, Educational Testing Service, Princeton, N.J., 1971.
- Kendall, M. G., & Stewart, A. Advanced theory of statistics, Volumes 2 & 3, New York: Hafner, 1973.
- Kuester, J. L., & Mize, J. H. Optimization techniques with FORTRAN. New York: McGraw-Hill, 1973.
- Kunzi, H. P., Tzschach, H. G., & Zehnder, C. A. Numerical methods of mathematical optimization, New York: Academic Press, 1971.
- Lawley, D. N. The estimation of factor loadings by the method of maximum likelihood. Proc. Royal Society Edinburgh, Section A, 1940, 60, 64-82.
- Lawley, D. N., & Maxwell, A. E. Factor analysis as a statistical method. (2nd ed.) New York: American Elsevier, 1971.
- Lord, F. M. A study of speed factors in tests and academic grades. Psychometrika, 1956, 21, 31-51.
- Maxwell, A. E. Recent trends in factor analysis. Journal of the Royal Statistical Society, 1961, A 124, 49-59.

- McDonald, R. P. Estimation and hypothesis-testing for a generalized factor analysis based on residual covariance matrices of prescribed structure. Research Bulletin RB-70-34, Educational Testing Service, Princeton, N.J., 1970.
- McDonald, R. P. Newton-Raphson methods in unrestricted factor analysis, A paper presented at the Psychometric Society meeting, 1972.
- McDonald, R. P., & Swaminathan, H. A simple matrix calculus with applications to multivariate analysis. General Systems, 1973, 18, 37-54.
- Merrifield, P. R. Factor analysis in educational research, Review of Research in Education, 1974, 2, 393-434.
- Montanelli, R. G. Jr. An investigation of the goodness of fit of the maximum likelihood estimation procedure in factor analysis. Report No. 451, Department of Computer Science, University of Illinois, Urbana, 1971.
- Mulaik, S. A. A note on some equations of confirmatory factor analysis. Psychometrika, 1971, 36, 63-70.
- Mulaik, S. A. The foundations of factor analysis. New York: McGraw-Hill, 1972.
- Mylander, W. C., Holmes, R. L., & McCormick, G. P. A guide to SUMT-version 4: the computer program implementing the sequential unconstrained minimization technique for nonlinear programming. Research Analysis Corporation Report RAC-P-63, Virginia, 1971.
- Nelder, J. A., & Mead, R. A Simplex algorithm for function minimization. The Computer Journal, 1965, 7, 308-313.
- O'Neill, R. Function minimization using a Simplex procedure. Applied Statistics, 1971, 20(3), 338-345.
- Ortega, J. M., & Rheinboldt, W. C. Iterative solution of nonlinear equations in several variables. New York: Academic Press, 1970.
- Parkinson, J. M., & Hutchinson, D. A consideration of non-gradient algorithms for the unconstrained optimization of functions of high dimensionality. Numerical methods for non-linear optimization, New York: Academic Press, 1972, 99-114.

- Parkinson, J. M., & Hutchinson, D. An investigation into the efficiency of variants on the Simplex method. Numerical methods for non-linear optimization, New York: Academic Press, 1972, 115-136.
- Powell, M. J. D. A survey of numerical methods for unconstrained optimization. SIAM Review, 1970, 12(1), 79-97. (a)
- Powell, M. J. D. Recent advances in unconstrained optimization. Research Group Report AERE-R 6430, Atomic Energy Research Establishment, Harwell, Berkshire, 1970. (b)
- Powell, M. J. D. A FORTRAN subroutine for unconstrained minimization, requiring first derivatives of the objective function. Research Group Report AERE-R 6469, Atomic Energy Research Establishment, Harwell, Berkshire, 1970. (c)
- Powell, M. J. D. Some properties of the variable metric algorithm. Numerical methods for non-linear optimization, New York: Academic Press, 1972.
- Powell, M. J. D. A view of unconstrained minimization algorithms that do not require derivatives. ACM Transactions on Mathematical Software, 1975, 1, 97-107.
- Press, S. J. Applied multivariate analysis, New York: Holt, Rinehart and Winston, 1972.
- Sargent, R. W. H., & Sebastian, D. J. Numerical experience with algorithms for unconstrained minimization. Numerical methods for non-linear optimization, New York: Academic Press, 1972, 45-68.
- Shanno, D. F. Conditioning of quasi-Newton methods for function minimization. Mathematics of Computation, 1970, 24, 647-656. (a)
- Shanno, D. F., & Kettler, P. C. Optimal conditioning of quasi-Newton methods. Mathematics of Computation, 1970, 24, 657-664. (b)
- Swaminathan, H. Structural analysis of dispersion matrices: Based upon a very general model with a rapidly convergent procedure for the estimation of parameters. Dissertation published by the Department of Measurement and Evaluation, Ontario Institute for Studies in Education, 1971.

- Tewarson, R. P. On the Davidon-Fletcher-Powell method for function minimization. Report No. 140, College of Engineering, State University of New York, Stony Brook, 1969.
- Tewarson, R. P. On the use of generalized inverses in function minimization. Computing, 1970, 6, 241-248.
- van der Voort, E., & Dorpema, B. A new algorithm to minimize functions. EURATOM Report EUR 4777e, Joint Nuclear Research Center, Ispra Establishment, Italy, 1972.
- Zeleznik, F. J. Quasi-Newton methods for nonlinear equations. Journal of the Association for Computing Machinery, 1968, 15, 265-271.

