PROCEEDINGS OF THE PRINCETON SYMPOSIUM ON MATHEMATICAL PROGRAMMING

Edited by

HAROLD W. KUHN

Princeton University Press
Princeton, New Jersey, 1970

A COMPARATIVE STUDY OF NONLINEAR PROGRAMMING CODES

A. R. Colville

I. Introduction

Numerous algorithms have been developed over the years for the solution of nonlinear programming problems. Most of these algorithms and many variations of them have now been programmed for the computer. The comparative study described in this paper is an attempt to gather experimental evidence in order to evaluate the relative computational efficiency of many of these methods. Interest in this area stems from the need for the development of new large-scale nonlinear programming codes for third generation computer systems. In the past, a number of theoretical studies have been made describing and sometimes comparing different classes of nonlinear programming methods. A few attempts have also been made to actually test some of these methods on the same problems. Most of these attempts compare only a very few of the existing approaches. Originally, it was the intent of the author to perform as complete an analysis as possible on a large number of these algorithms. Most of the theoretically promising methods were going to becoded using the same programming language for the same computer. A standard set of test problems would then be solved by these codes in a controlled experimental environment. It quickly became obvious however, that to include enough of the better known techniques and variants of them in this type of study would be prohibitive with regard to the amount of time required.

In gathering information on what algorithms ought to be studied in this fashion, it was found that a number of people and organizations were interested in an evaluation of this type. It was decided, therefore, to make this study a cooperative venture where each participant would only be required to solve a set of test problems using his own method and computer program.

One advantage of this approach was that each method would be tested by someone thoroughly familiar with its vagaries and behavior, and by someone with an interest in its comparative standing. As news of this project spread, more and more interested parties volunteered. Eventually, 34 different computer codes were actively tested in this evaluation.

II. Purpose

The purpose of this study as it is currently organized is to gather statistics on the efficiency of various nonlinear programming codes and their behavior in solving a set of standard test problems. These statistics are gathered in such a way as to allow a comparison to be made between the various methods. It is hoped that one can then evaluate these methods and determine their relative efficiency

in solving various important classes of problems. This information should be of great value in planning future efforts in the nonlinear programming area. A secondary benefit of this work will be to provide a standardized set of test problems for use in future work of this sort. This set of test problems can be used by future developers of nonlinear programming codes in order to evaluate new ideas and new approaches. This study was originally designed for and oriented towards those classes of nonlinear programming problems involving many variables and constraints but containing reasonably smooth continuous nonlinear functions. However, a few of the test problems and a number of the nonlinear programming codes participating in this effort are more related to the smaller but highly complex nonlinear optimization area. Included here is the important class of problems known as "function minimization" or "unconstrained optimization." These different classes of problems may be illustrated graphically by Figure 1 which shows the "space" of nonlinear programming problems with problem size increasing from left to right and problem complexity increasing from top to bottom. Here size means dimensionality of the problem or number of variables, and complexity indicates the degree of nonlinearity in the objective function and constraints. Linear programming problems, for example, would lie along the top edge of this diagram. The two major important problem classes are shown by the shaded region. Those in the "upper right-hand corner" arise many times as an outgrowth of work on LP models. In addition to function minimization problems, the "lower left-hand corner" contains those complex problems involved in design optimization where the number of parameters to be varied is often rather small. Because of the wide deviation between these two problem areas and the methods for solving them, many of the comparisons attempted by this study are difficult. It has been suggested that a further effort of this sort be directed towards the specific area of unconstrained minimization in order to compare these methods on a more equitable basis.

III. Method

The first step in this study was the gathering of a representative set of test problems. For the most part, these were intended to be typical of the type of nonlinear programming problem being solved in the real world today. From a number of problems so collected, eight were chosen for the study. These represent a fairly varied group with regard to size and structure. None of the problems selected were large problems as it would have been difficult for the participants cooperating in this study to devote the time and resources necessary for the solution of very large problems.

In order to standardize the reporting, information on each participant's method and computer code was collected in the form of data sheets. These each contained a brief description of the technique, an indication of the computer system used, a list of pertinent references, and a discussion of any unique features of the code. Results from each problem solved were also submitted in the form of data sheets. They indicated the following items:

1. How much time was required to set up and program the problem for this code?
2. Were partial derivatives required for the solution of the problem, and if so, were they calculated analytically or numerically?
3. How many function evaluations and constraint evaluations were required for the solution of the problem?
4. How much time was required to execute the problem solution?
5. What were the solution values for the objective function and the independent variables?

In order to compare results obtained in this fashion from many different people using many different computers, it was necessary to decide upon a comparison criterion. One measure of the efficiency of a program is the number of function and constraint evaluations required for the solution of a problem. This measure is particularly meaningful for those problems which tend to have fewer variables but are quite complex (e.g., unconstrained minimization problems) where the time required for a function evaluation is many times greater than that required for the algorithm to determine which point to evaluate next. It is not always a meaningful criterion when applied to larger problems with many variables and constraints where the functions tend to be smoother and at times nearly linear. Indeed, most codes for problems of this sort do not even keep a count on these two items. Although the number of function and constraint evaluations was requested for each method of each problem, they were not found to be useful in analyzing the results obtained.

For this reason, solution time was chosen as the main basis for comparing results. It leads to a further difficulty however, in that different computers were used by many of the participants, each of them with a different computing power. In order to circumvent this problem, a standard timing program was sent out to all participants. This program, which simply inverts a matrix a number of times, was then timed to get an estimate of the computing power of the machine used. Written in FORTRAN, this program took about two minutes to execute on an IBM 7094. All of the problem solution times obtained in the study were then "standardized" by dividing

them by the amount of time required to execute the standard timing program.

IV. Classes of Methods

Rather than present the results solely in terms of individual participants or computer codes, an attempt has been made to group them into classes of nonlinear programming methods. These classes are as follows:

1. Search Methods or Zero Derivative Methods
2. Small Step Gradient Methods
3. Large Step Gradient Methods
4. Second Derivative Methods
5. Miscellaneous Methods

Table 1 indicates what methods were included in the study, and how they were categorized in each of the five classes. A few points of interest should first be noted in this table. Of those programs included in the first class, only Rosenbrock's method uses the penalty function concept for the solutions of constrained problems. In addition, only those methods attributable to Himmelblau and Greenstadt in the first two groups require the calculation of any derivatives. The rest all calculate these quantities numerically. On the other hand, all of the Large Step Gradient Methods and Second Derivative Methods required the use of analytically calculated derivatives. Of the twelve Large Step Gradient Methods, only one is a penalty function method whereas two out of the four Second Derivative Methods fall into this category.

Concerning the need for a class known as Miscellaneous Methods, Separable Programming, although it is considered to be an effective nonlinear programming tool for many important problems, was not expected to show up well on the type of problems proposed for this study. In addition, both it and the Method of Centers do not fall conveniently into any of the other four classes previously defined. In most instances, these two methods are not considered in the grouped comparisons.

V. Individual Problem Results

Problem 1. For problem one, which had a cubic objective function with five variables and five linear constraints, the following results were obtained:

Class	Number of Methods	Average Standard Time
Search Methods	7	0.378
Small Step Gradient Methods	4	0.055
Large Step Gradient Methods	11	0.023
Second Derivative Methods	4	0.023

As would be expected for a problem with a reasonably smooth objective function and linear constraints, the use of derivatives seems to have a great effect upon the efficiency of the solution method.

Problem 2. The results from Problem 2 indicate a similar trend in efficiencies over these four classes. The problem, which is the dual of Problem 1, contains a cubic objective function with fifteen variables and has twenty quadratic constraints. Listed below, the relative differences between each class are not nearly what they were for the first problem.

Class	Number of Methods	Average Standard Time
Search Methods	1	0.765
Small Step Gradient Methods	3	0.325
Large Step Gradient Methods	6	0.256
Second Derivative Methods	2	0.265

This problem was apparently quite a bit more difficult than the first one. Five participants who attempted to solve it admitted that they did not converge. Two results reported for Search Methods obtained incorrect "optimal" solutions.

Problem 3. Problem 3 (quadratic objective function, five variables, twelve quadratic constraints) shows a considerable improvement in the ranking for Small Step Gradient Methods over that shown in previous problems. Search Methods average particularly poorly in this case.

Class	Number of Methods	Average Standard Time
Search Methods	7	0.279
Small Step Gradient Methods	3	0.017
Large Step Gradient Methods	6	0.017
Second Derivative Methods	4	0.020

Problem 4. Problem 4 on the other hand, showed Search Methods off to some advantage. This problem, a particularly messy unconstrained minimization problem with four variables and a quartic objective function, caused many methods to fail when they found a false local optimum. Two of the five Small Step Gradient Method had this difficulty, probably because of numeric calculation of derivatives.

Class	Number of Methods	Average Standard Time
Search Methods	5	0.025
Small Step Gradient Methods	5	0.048
Large Step Gradient Methods	8	0.021
Second Derivative Methods	2	0.014

Problem 5. Problem 5 was, in reality, a mixed-integer programming problem, as it contained a step function giving discontinuous derivatives. In essence, it contained a myriad of local optima with many different values. Each of the ten solutions submitted attained a different one of these optima. As this was a minimization problem, one method stands out above all others in that it found a solution whose objective function was one half as large as the best one found by all the other methods. This method happens to be a random Search Method known as OPTIM (developed by Mobil Oil Co.). In the course of its random probes, it apparently happened to light upon the best known local optimum discovered so far. Due to the discontinuous derivatives, methods requiring analytical calculation of derivatives usually did not attempt this problem.

Class	Number of Methods	Average Standard Time
Search Methods	4	0.433
Small Step Gradient Methods	5	0.134
Large Step Gradient Methods	0	0.000
Second Derivative Methods	1	0.031

Problem 6. This problem contained constraint equations with sine and cosine functions, and it had an objective function with discontinuities in its first derivatives.

Class	Number of Methods	Average Standard Time
Search Methods	4	0.277
Small Step Gradient Methods	3	0.060
Large Step Gradient Methods	2	0.028
Second Derivative Methods	1	0.071

Derivative Methods behaved appreciably better than Search Methods on this problem.

Problem 7. With a fourth-order polynomial objective function and eight linear equality constraints, this problem was solved nicely by nine different Large Step Gradient Methods.

Class	Number of Methods	Average Standard Time
Search Methods	1	0.463
Small Step Gradient Methods	3	0.299
Large Step Gradient Methods	9	0.094
Second Derivative Methods	2	0.223

Problem 8. This was a typical process optimization type problem where the functions are described in terms of a self-contained simulation type model. Analytic expressions for the partial derivatives were not explicitly stated.

Class	Number of Methods	Average Standard Time
Search Methods	2	0.167
Small Step Gradient Methods	4	0.053
Large Step Gradient Methods	4	0.028
Second Derivative Methods	1	0.035

VI. Summary Results

In order to compare the results from each method across all of the problems solved, some type of scoring system had to be devised. To do so, first the mean value and standard deviation of all standard time results were calculated for each problem. Next, each method was given a score for each problem depending upon the number of "standard deviation units" above or below the mean value it happened to fall. For each method, these scores were then added up for all problems solved. In order to favor those methods which solved more of the problems, a weighted average score was obtained by dividing the total score by the square root of the number of problems solved. These results are shown in Table II ranked according to the weighted average score within each class of methods. They include results for two different starting points for Problems 2, 3, and 5.

As can be seen from Table II, there is a wide range of scores from very high to very low for both Search Methods and Small Step

Gradient Methods. On the other hand, both the Large Step Gradient Methods and the Second Derivative Methods seemed to be consistently high and above average. An over-all summary of these results for each class is shown in Table III. Obviously, for this group of problems, the Large Step Gradient and Second Derivative Methods, as a group, were much more efficient than either the Search Methods or the Small Step Gradient Methods. It should be emphasized that this observation is only for these classes of methods taken as a group and not individually. A number of the Search and Small Step Gradient Methods individually ranked high in the over-all standings.

VII. Study Difficulties

Before drawing conclusions from this work, a number of difficulties and disadvantages of this method of comparison must be pointed out. First of all, data obtained on the number of function and constraint evaluations proved to be an unsatisfactory basis for comparison. For this reason, timing results were used as the main comparative criterion. These timing results, unfortunately, are subject to numerous error possibilities. First of all, replicate results for the standard timing program obtained from identical computing hardware, showed discrepancies of as much as 10%. Possibly this is due to the manner in which the times were obtained. In addition, the standard timing program was not able to eliminate other discrepancies relating to the efficiency of various FORTRAN compilers available on different computers. However, although the standard time ratios obtained in this study may easily have contained an experimental error of 10 to 15%, the results obtained varied by as much as 100-fold from best to worst on each problem, and therefore, the comparison of results are still reasonably significant.

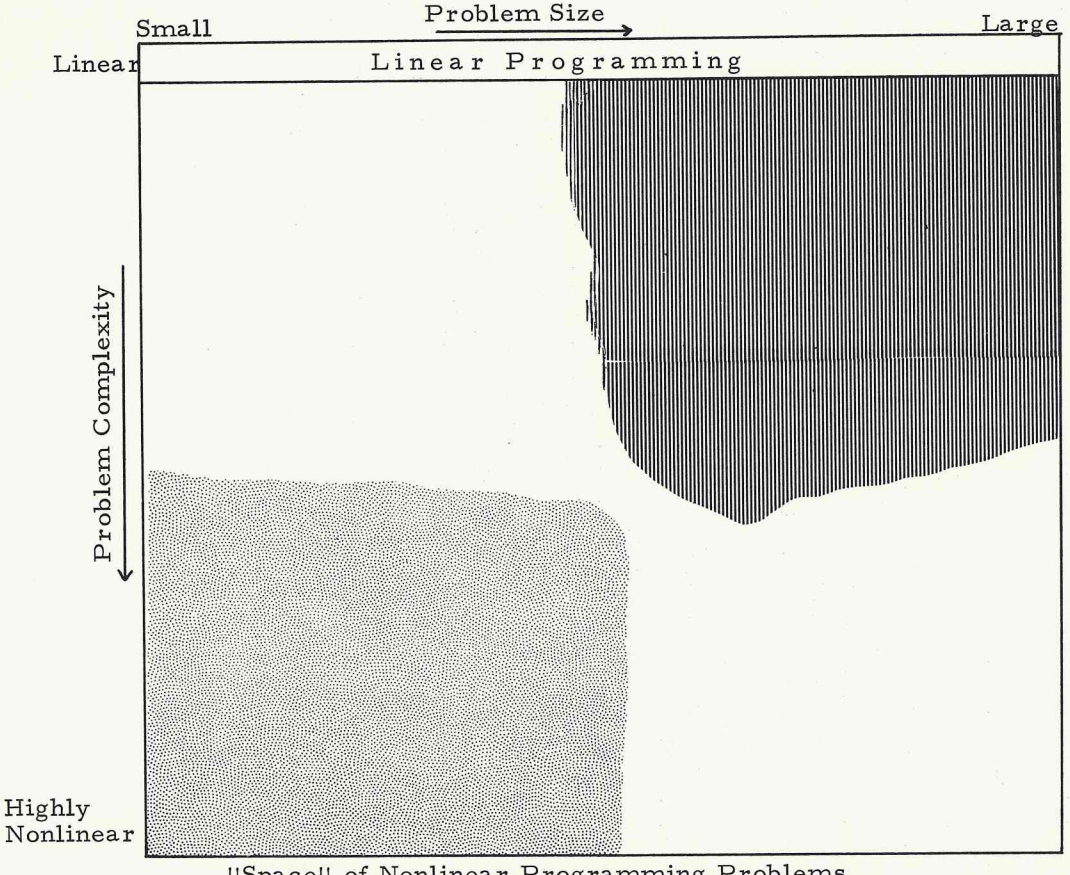
Another major source of error in this study results from the lack of a standard accuracy to which each problem should have been solved. Because of this, it is possible for a code to solve a problem rapidly but inaccurately and to look better than a code which solves the same problem more slowly but more accurately. Future efforts of this sort should include some methods for correcting this deficiency, such as requiring a specific degree of accuracy for each problem solution.

Also, in a study of this sort the cooperative nature of it prohibits, in a practical sense, the use of any large-scale test problems. Each participant in donating his own time and computer time for this study would probably not have the amount of time necessary for setting up and solving extremely large problems. Unfortunately, the results obtained on small problems are not always reliable for predicting performance on large problems. For instance, a number

of the computer programs participating in this study are production type codes designed for routine solution of large problems. At times, the systems overhead involved in solving a small problem can be greater than the actual algorithmic computation time. Other participating codes were rough experimental programs which have not yet been fully developed. All of these factors tend to cloud the interpretation of the results obtained.

VIII. Conclusions

With all the difficulties, inaccuracies and discrepancies present in this study, a number of worthwhile conclusions may still be drawn. First, it appears that the efficiency and performance of a nonlinear programming code can be greatly affected by the method of implementing it on a computer. This was particularly obvious in looking at the results from the Search Methods and the Small Step Gradient Methods, where some of these codes ranked among the leaders while others, with an identical mathematical approach, would be far, far behind. Second, the Large Step Gradient Methods and the Second Derivative Methods, on the average, behaved quite a bit better than the other two major classes. One factor in this success may be that all of these methods utilize analytically calculated derivative information whereas the Small Step Gradient Methods represented here almost all use numerical derivatives and the Search Methods use no derivatives at all. Finally, although certain methods or groups of methods may have performed badly when considering the average over all problems solved, many of these methods or classes were quite efficient with regard to one or more specific problems in this study. It does appear that a number of nonlinear programming codes have shown up rather well in this experiment and should be more fully exploited. Still further work needs to be done, particularly in the area of unconstrained minimization, before further more definite conclusions can be drawn.



"Space" of Nonlinear Programming Problems

Figure 1

TABLE I. PARTICIPATING PROGRAMS

	Participant	Affiliation	Derivatives
1. Search Methods			
OPTIM	Boas	Mobil	None
Sequential search	Cooper	Wash. Univ.	None
COMPLEX	Davies	ICI Ltd.	None
Rosenbrock	Davies	ICI Ltd.	None
Klingman & Himmelblau	Grace	P and G	Analytic
Mult. gradient summation technique	Himmelblau	U. of Texas	Analytic
CANDIDE	Himmelblau	U. of Texas	None
Simplex search	Miller	Shell Dev.	None
PROBE	Sullivan	IBM	None
2. Small Step Gradient Methods			
POP/360	Colville	IBM	Numeric
Richochet gradient	Greenstadt	IBM NYSC	Analytic
POP II/7094	Grigsby	Phillips	Numeric
Carbide optimization package	Hutton	Union Carbide	Numeric
Generalized gradient search	Kephart	Union Carbide	Numeric
Method of approx. programming	Miller	Shell Dev.	Numeric
Deflected ascent	Miller	Shell Dev.	Numeric
3. Large Step Gradient Methods			
Generalized reduced gradient	Abadie	EdF	Analytic
GRG II	Abadie	EdF	Analytic
Method of feasible directions	Anthony	IBM Research	Analytic
Davidon with CRST	Davies	ICI Ltd.	Analytic
Convex programming	Gauthier	IBM France	Analytic
Conjugate gradient	Goldfarb	Courant Institute	Analytic
Reduced gradient	Huard	EdF	Analytic
Gradient projection corrige	Kalfon	EdF	Analytic
Gradient projection	Miller	Shell Dev.	Analytic
Variable metric projection	Murtagh	Imperial College	Analytic
Revised reduced gradient	Ribiere	IBM France	Analytic
Modified feasible directions	Tzschach	IBM Germany	Analytic
4. Second Derivative Methods			
Courant	Ballot	CCSA	Analytic
Gauss-Newton-Carroll	Bard	IBM-CSC	Analytic
SUMT	McCormick	RAC	Analytic
SOLVER	Wilson	Stanford	Analytic
5. Miscellaneous Items			
Separable programming	Harvey	Std. Oil of Cal.	None
Method of centers	Huard	EdF	Analytic

TABLE II. RESULT SUMMARY

	No. Problems Solved	Weighted Avg. Score
1. Search Methods		
Klingman & Himmelblau	1	1.08
PROBE	3	0.56
Simplex search	4	0.38
OPTIM	7	0.37
Sequential search	7	-0.67
Rosenbrock	2	-1.74
COMPLEX	2	-2.84
Multiple gradient summation technique	5	-4.54
CANDIDE	8	-5.00
2. Small Step Gradient Method		
POP/360	8	0.94
Richochet	5	0.70
Carbide optimization package	3	0.40
Deflected ascent	5	0.27
Generalized gradient search	10	-0.32
POP II/7094	3	-0.73
Method of approximation programming	1	-2.13
3. Large Step Gradient Method		
GRG II	9	2.64
Gradient projection corrige	6	1.56
Revised reduced gradient	3	1.49
Reduced gradient	2	1.21
Generalized reduced gradient	9	1.05
Modified feasible directions	3	0.95
Variable-metric projection	8	0.86
Davidon with CRST	8	0.70
Gradient projection	3	0.53
Method of feasible directions	1	0.28
Conjugate gradient	3	-0.11
Convex programming	1	-0.31
4. Second Derivative Methods		
COURANT	11	1.02
SOLVER	3	0.87
Sequential unconstrained minim. technique	6	0.85
Gauss-Newton-Carroll	3	0.67
5. Miscellaneous Methods		
Method of centers	1	-0.36
Separable programming	6	-2.46

TABLE III. OVER-ALL SUMMARY

	No. of Methods	No. Problems Solved	Weighted Avg. Score
Search methods	9	39	-1.38
Small step gradient methods	7	35	-0.03
Large step gradient methods	12	56	1.56
Second derivative methods	4	23	0.78

References

1. Y. Bard, and J. Greenstadt, "A Modified Newton Method for Optimization with Equality Constraints," IBM NY Scientific Center Report No. 320-2948, May 1968.

2. M. J. Box, "A New Method of Constrained Optimization and A Comparison with Other Methods," The Computer Journal, Vol. 8, pp. 42-52.

3. J. Carpentier et J. Abadie, "Generalisation de la methode du gradient reduit de Wolfe au cas de constraintes non lineaire," Proceedings of the IFORS Conference, Cambridge Massachusetts, (August 29-September 2, 1966).

4. C. W. Carroll,"The Created Response Surface Technique for Optimizing Nonlinear Restrained Systems," Operations Research, Vol. 9, No. 2 (March-April, 1961), pp. 169-184.

5. A. R. Colville, "A Comparative Study of Non-linear Programming Codes," IBM NY Scientific Center Report No. 320-2949, June 1968.

6. W. C. Davidon, "Argonne National Laboratory Report No. ANL-5990 (November 1959).

7. P. Faure et P. Huard, "Resolution des Programmes Mathematiques a fonction non-lineaire par la Methode du Gradient Reduit," Revue Francaise de Recherche Operationnelle, Vol. 9, (1965) pp. 167-205.

8. A. V. Fiacco and G. P. McCormick, "The Sequential Unconstrained Minimization Technique for Nonlinear Programming, A Primal-Dual Method," Management Science, Vol. 10, No. 2, (1964),pp. 360-366.

9. _____, "Computational Algorithm for the Sequential Unconstrained Minimization Technique for Nonlinear Programming," Management Science, Vol. 10, No. 4 (1964), pp. 601-617.

10. _____, "Programming Under Nonlinear Constraints by Unconstrained Minimization: A Primal-Dual Method," Research Analysis Corporation, RAC-TP-96, September, 1963.

11. _____, "The Sequential Unconstrained Minimization Technique for Convex Programming with Equality Constraints," Research Analysis Corporation, RAC-TP-155, April, 1965.

12. H. Glass and L. Cooper, "Sequential Search: A Method for Solving Constrained Optimization Problems, Journal of ACM, Vol. 12, No. 1 (January 1965), pp. 71-82.

13. J. Greenstadt, "On the Relative Efficiencies of Gradient Methods," Mathematics of Computation, Vol. 21, No. 99 (July 1967), pp. 360-367.

14. _____, "A Richocheting Gradient Method for Nonlinear Optimization," J. SIAM Appl. Math., Vol. 14, No. 3 (May 1966), pp. 429-445.

15. Griffith and Stewart, "A Nonlinear Programming Technique for Optimization of Continuous Processing Systems," Management Science, Vol. 7, No. 4 (July 1961), pp. 379-392.

16. P. Huard, "Resolution de programmes mathematiques a contraintes non lineaires par la Methods des Centres." Note interne de l'E.D.F., n° HR 5.690 du 6 Mai.

17. _____, "The Method of Centers - Nonlinear Programming: A Course," Edition, North-Holland Publishing Company (1966).

18. W. R. Klingman and D. M. Himmelblau, "Nonlinear Programming with the Aid of a Multiple Gradient Summation Technique," Journal of ACM, Vol. 11, No. 4 (October 1964), pp. 400-415.

19. R. A. Mugele, "A Nonlinear Digital Optimizing Program for Process Control Systems," Proceedings for the Spring Joint Computer Conference, San Francisco, May 1, 1962.

20. _____, "A Program for Optional Control of Nonlinear Processes," IBM Journal, Vol. 1 (1962), pp. 2-17.

21. _____, "The Probe and Edge Algorithms for Nonlinear Optimization," Recent Advances in Optimization Techniques, Lavi and Vogl (1966), pp. 131-144.

22. G. P. McCormick, W. C. Mylander and A. V. Fiacco, "Computer Programming Implementing the Sequential Unconstrained Minimization Technique for Nonlinear Programming," Research Analysis Corporation, RAC-TP-151, April, 1965.

23. J. B. Rosen, "The Gradient Projection Method for Nonlinear Programming - Part 1. Linear Constraints, Journal of the Society of Industrial Applied Mathematics, Vol. 8, No. 1, (March 1960), pp. 181-217.

24. H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," The Computer Journal, Vol. 3 (October 1960), pp. 175-184.

25. R. Wilson, "A Simplicial Algorithm for Concave Programming," unpublished doctoral dissertation, Harvard University Graduate School of Business Administration, Boston, 1963.

26 _____, "Subroutine SOLVER: Description, Instructions, Glossary, Bulletins 1, 2, 3," working papers of the Western Management Science Institute, University of California, Los Angeles, 1964.

27. P. Wolfe, "Methods of Nonlinear Programming," Recent Advances in Mathematical Programming, Graves and Wolfe, Editors, McGraw-Hill, 1963, pp. 67-86.