PROGRAM DESIGN PRINCIPLES

THE STRUCTURED-UNSTRUCTURED PROGRAM STUDY

by

JORGE R. TABORGA
(Computing Science)

Submitted in Partial Fulfillment of the Requirements of the

University Undergraduate Fellows Program

1979-1980

Approved by:

[illegible]

Dr. William M. Lively III

April 1980

ABSTRACT

The sharply rising cost incurred in the production of quality software has brought with it the need for the development of techniques used during program design and implementation. The two most efficient techniques for the production of software are top-down design and structured programming.

The differences of these techniques and other software production methods can be quantitatively measured using the concepts of software science. Conclusions can then be made based on these measures, regarding the efficiency and the economic advantages of such techniques.

This paper is divided in two parts. The first part is dedicated to program design and structured programming. The second part introduces the concepts of software science and discusses in detail the structured-unstructured program study. This study has as an objective to measure the differences of structured and unstructured programs in regard to their size, amount of information contained, and programming effort.

ACKNOWLEDGMENTS

The author is deeply indebted to his parents for their tremendous help and encouragement. He also wishes to thank Dr. William M. Lively for his time and ideas.

This paper was typed by Charlene Helton.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PART I

1. INTRODUCTION

The two components of computers are hardware and software. The hardware is the actual device, composed of circuits and electro-mechanical parts, that make a computer work. Software, on the other hand, is the application programs and system control programs that we feed into the computer.

In the last decade, software engineering has emerged as a discipline concerned with the development and utilization of systematic methodologies and techniques. These methodologies and techniques are used for designing, implementing, and maintaining software systems. Software systems can be divided into two categories: those due to the nature of the computer (system programs), and those generated by the user community (application programs).

A study on information processing and data automation implications made by the Air Force Systems command [Boehm 1973], shows that for almost all applications software will have, as opposed to hardware, the biggest cost. Figure 1-1 shows the cost of software as compared to hardware for a span of 30 years. As exhibited in Figure 1-1, the estimated software expenditure in the Air Force and other similar organization will go over 90% of the total system cost by 1985.

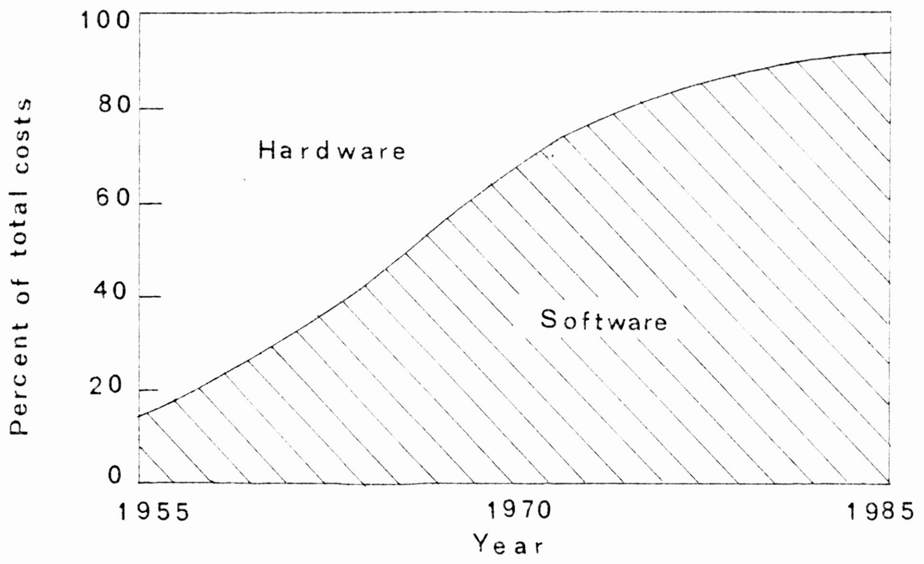


Figure 1-1. Hardware/software cost trends.

Software is big business. The overall software costs in the United States are probably over 10 billion dollars per year, over 1% of the gross national product. Consider what will happen in the years ahead when hardware gets less expensive because of a more advanced technology, and the software (people) costs go up. The relationship of software and hardware costs will then be even more dramatic.

Big as the direct costs of software are, the indirect costs are even bigger, because software generally is a major portion of the critical path in the overall system development. Consequently, any delays in the software schedule represent delays in the overall completion schedule of the system. What can be done in order to get software off the critical path, or at least to minimize the delays in software development? One opinion would be to add more people in hopes that a human wave of programmers will quickly overcome the problem. But measures of this kind usually make things worse rather than better [Brooks 1971]. There are several other measures that can be taken to minimize delays and to take software off the critical path. These measures can be divided into three categories [Boehm 1973]:

1. Increasing each individual's software productivity.
2. Improving project organization and management.
3. Initiating software development earlier in the system development cycle.

This paper mainly deals with increasing each individual's software productivity by means of software design strategies and structured programming. The other two measures to minimize delays and to take software off the critical path will not be developed here.

There are many factors that influence software productivity. One factor is computer response time. Many studies conducted in the subject show a 20% improvement in programming efficiency using an on-line system versus a batch system [Sackman 1970]. Another important factor is programming languages. Comparisons of high level and low level languages have resulted in significant differences in development time for the same program. The most important factor in software productivity is the selection of the right people to do the job. This task is not easy; consequently further work in the areas of personnel selection, training, and evaluation should be closely followed.

Now comes the problem of how to improve software productivity. Significant opportunities exist. The main one is concerned with the awareness of each individual programmer in regard to where his time is really going. The programmer should plan his design and develop thoughtful tests for the software he produces. Another opportunity to improve software productivity lies in the area of programming languages. As mentioned earlier, high level languages provide fast and accurate results; therefore their use is highly recommended. Tools and techniques are also very valuable in increasing software productivity, especially in the designing phase. The top-down approach in the design and structured programming in the implementation of a program constitute the most valuable of those tools and techniques.

The following two sections develop the ideas of top-down design and structured programming.

2. PROGRAM DESIGN

The program design begins once the problem is thoroughly defined. Actually, the distinction between the definition and the design is not sharp. Both activities are processes of discovery and resolution. As these processes advance, the designer discovers new requirements, irregularities in the system structure, errors in the work done so far, and opportunities for tradeoffs in cost, performance, and other types of measures. As the problem is defined and the major tradeoffs are made, the document of objectives is created. The definition phase is concluded when this document is complete and has been agreed to by the user and the developer.

The design phase is complete when the specification document of the design is finished. This is a document describing how the program was built. It is necessary to document the specification, in order to record the many details that could be lost otherwise. Equally important, the document serves as the basis for discussion with the user and the operators. It is also the only material back-up if, for any reason, the designer must be replaced by a substitute during the job. Finally, the specification document is the baseline against which the final program should be tested to verify that it does the things it is supposed to do.

DESIGN REPRESENTATIONS

In the design of a program, the representational scheme is of fundamental importance. Good notation can clarify the interrelationships and interactions in the program, where as poor notation can complicate the design. The commonly used representations for specifying the design of a program include structure charts, HIPOs, pseudocode, and structured flowcharts [Fairley 1976]. The first three design representation schemes are discussed in this section. The discussion on structured flowcharts is left for section three.

Structured Charts

Structure charts are useful during general program design as an aid in determining the functions, parameters, and interfaces of the system. A structure chart is different from a flowchart, because the former does not have decision boxes; nor does it have sequential ordering of tasks.

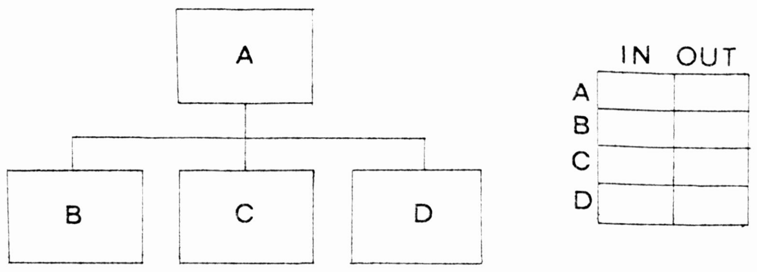


Figure 2-1. Structured chart with input/output table.

Figure 2-1 shows the application of structure charts to describe the structure of a hierarchical system. The chart can be augmented with a module by module description of the input and output parameters.

HIPOs

Hierarchy plus Input-Process-Output diagrams are formalized structure charts. As such, they emphasize the functional aspects of the design rather than the internal control flow mechanisms of a system.

A typical set of HIPO diagrams consists of a visual table of contents, overview diagrams and detail diagrams. The visual table of contents is a directory to the set of diagrams in the package. It consists of a tree structured table of contents, a summary of the contents of each of the overview diagrams, and a legend of symbol definitions (see Figure 2-2). Overview and detail diagrams describe the inputs, the process to support the function being described, and the outputs for the process. The format of an overview and detail diagram is represented in Figure 2-3.

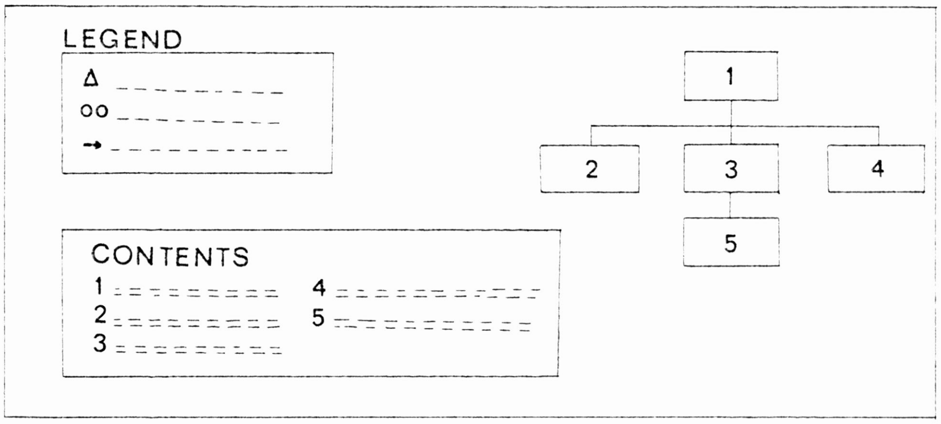


Figure 2-2. Visual table of contents.

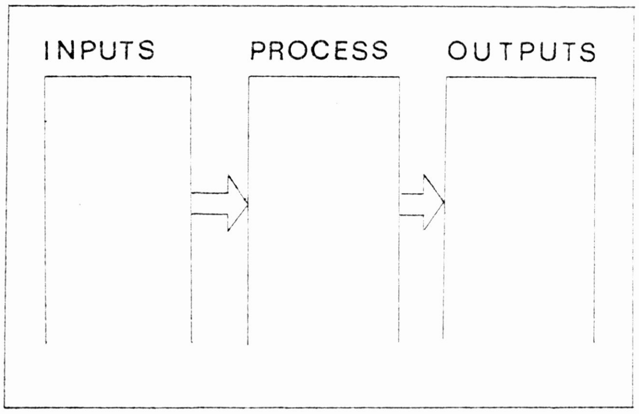


Figure 2-3. Overview and detail diagram.

Pseudocode

Using this notation, the designer describes the design using short concise English phrases that are structured by key words such as READ, INCREMENT, and DO WHILE. Key words and indentation describe the flow of control, while the English phrases describe the processing function.

As an example of pseudocode, assume that a sorting algorithm will be described. A list of numbers is sorted from smallest to largest. The pseudocode description of this algorithm might have the following form:

```
DO WHILE  Advancing position one by one until next to last
          record.
   INITIALIZE  Interchange flag to no interchange.
    DO WHILE  Advancing position one by one until next to
              last record.
       IF  First record greater or equal to second record
          THEN  Set interchange flag to interchange.
                Perform interchange of records.
       IF  Interchange flag is set to no interchange
          THEN  Exit.
    END LOOP
END LOOP
```

TOP-DOWN PROGRAM DESIGN

There are basically two approaches to the design of a program: the bottom-up approach and the top-down approach.

In the bottom-up approach, implementation begins after an initial design which identifies the tasks. The most elementary level functions are implemented first and then used as building blocks to compose more complicated tasks. One major advantage of the bottom-up approach is the concentration of effort in the high-risk, low-level components during the early development phase to determine whether or not they can be performed as specified. The major problem with the bottom-up approach is its lack of attention to the integration, verification, and validation of program modules [Ramamoorthy, Ho 1977].

Top-down design is considered one of the design methodologies for reducing the complexity of design and program analysis. The main idea of top-down design is to minimize logical errors and inconsistencies by having a structural specification of the development process.

The top-down approach begins by determining what overall functions will be performed by the program, in what order, and under what conditions. The design then continues with the development of a top-level module, containing all the logic controlling the sequencing between functions, but introducing dummy subprograms for the functions not yet implemented. These dummy subprograms are referred to as stubs. The succeeding steps consist of expanding the dummy subprograms into a lower-level sequence of control logic, computation, and subfunctions. As each subprogram is developed to replace a stub, it can be tested immediately; not only by itself, but also as part of the whole program. Figure 2-4 shows the graphical representation of a top-down design [Aron 1974].

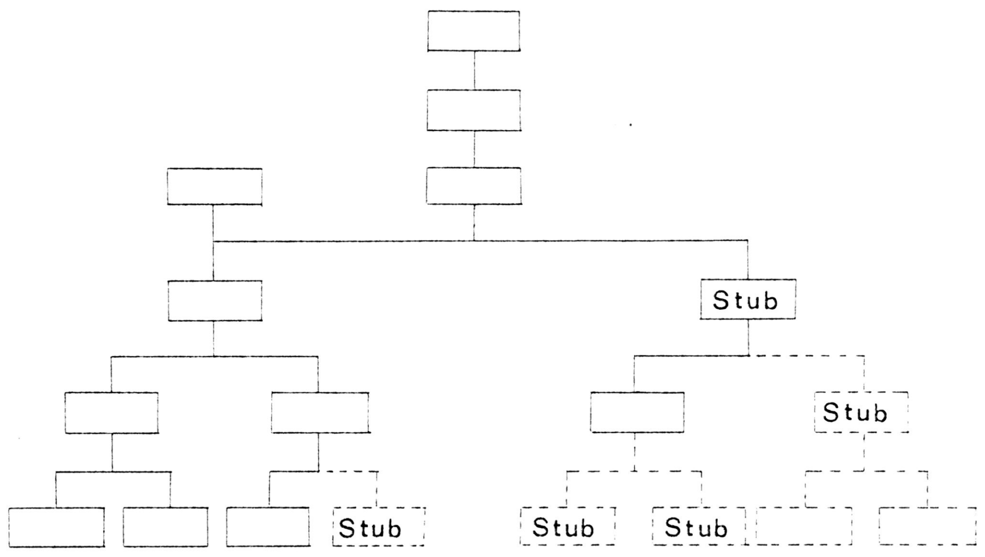


Figure 2-4. A top-down program design.

3. STRUCTURED PROGRAMMING

Structured programming is a coding technique that enables programs to be written in execution sequence. This permits a visible relationship between the program listing and the dynamic execution of the program.

A structured program is formed of three basic program structures. These structures are: sequences, choices or selections, and loops or repetitions. It has been shown that sequence, choice, and repetition structures are sufficient to solve any logic problem. The result is based on a theoretical foundation defined by Bohm and Jacopini [1966]. This proof only applies to proper programs. Proper programs have the following characteristics:

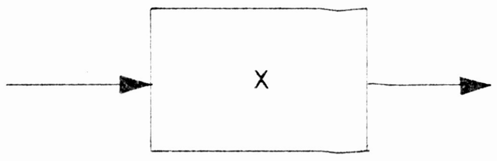
1. They have one entry point.
2. They have one exit point.
3. They do not have unreachable code.
4. They do not have infinite loops.

Structured programs are proper programs in the sense that they also have these characteristics.

PROGRAM STRUCTURES

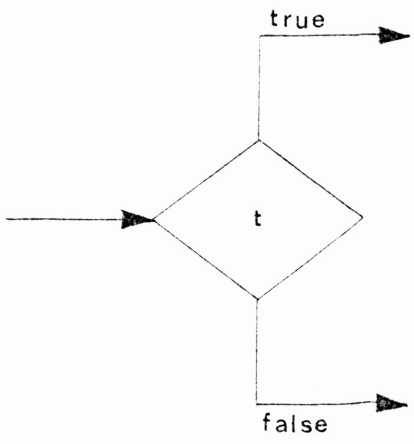
The symbols used to describe the program structures include: process, decision, connector symbols, and connector lines [Hughes, Michtom 1977].

Process Block. This symbol represents an operation that is to be performed. It consists of a rectangle with one control path leading into it and one leading out.

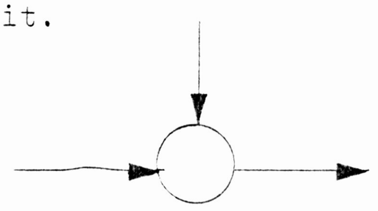


The process X may be a single executable statement, a call to and return from a subroutine, another logic structure, or a number of logic structures forming a subprogram or subroutine.

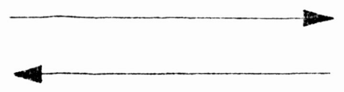
Decision Symbol. The decision symbol specifies a test operation. It consists of a decision box and is characterized by one control path leading in and two paths leading out. The specification t represents the expression to be tested. One or the other output path is taken as a result of the test.



Collector Symbol. The collector symbol is a circle where control paths converge. No operation is performed here. It is simply a junction which typically has two entries and one exit.

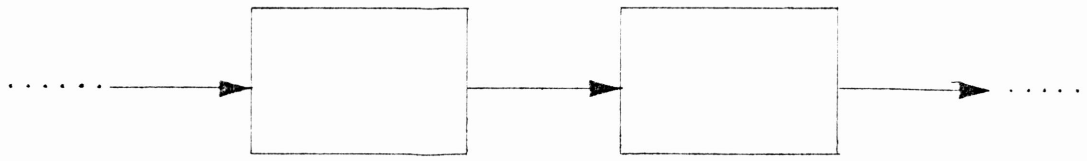


Connector Lines. Connector lines represent the passing of control from one of the above symbols to another in the direction indicated by the arrow.



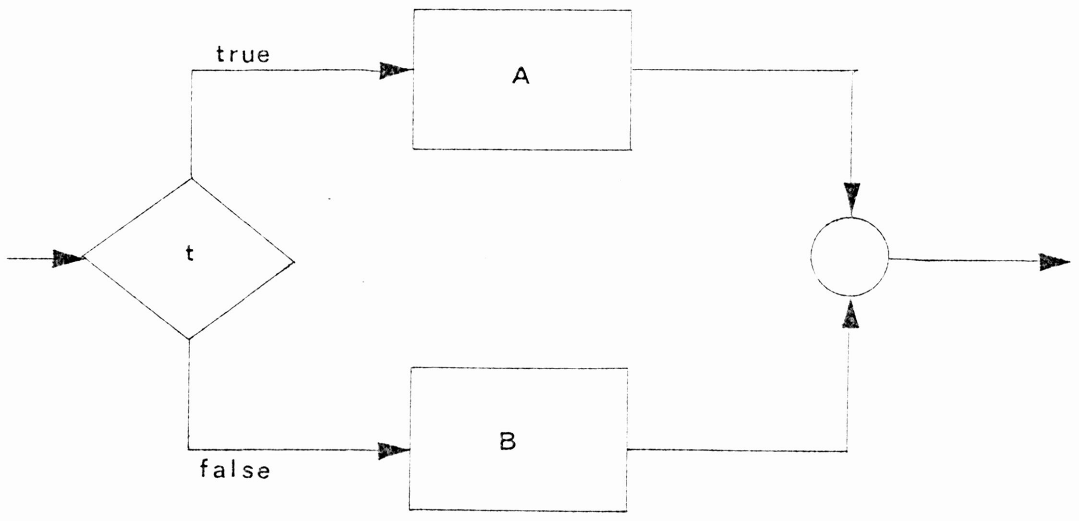
A. Sequence Structure

This structure indicates the flow of control from one process block to the next in sequence.



B. Choice Structure

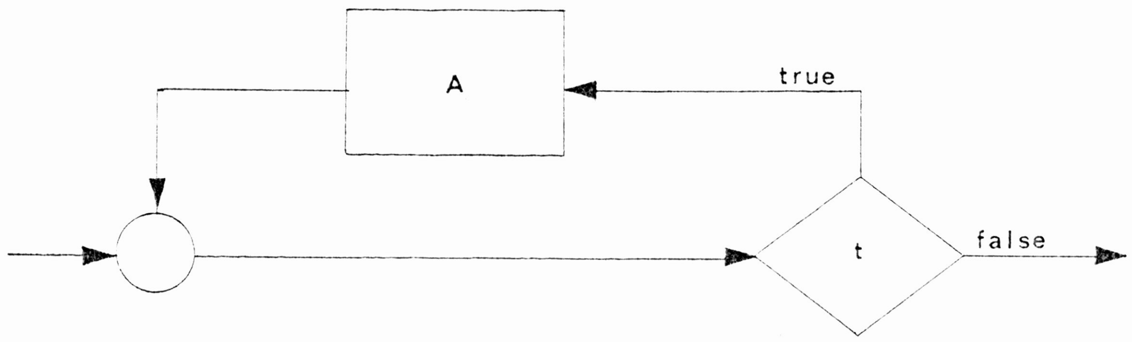
This program structure is also referred to as ifthen-else. It provides for a choice between two alternatives. In the structure, the test of t is performed and its evaluation results in a single binary condition. This condition may be a true/false, yes/no, on/off, or a similar resulting condition. The expression t could be a variable or a combination of variables. If the expression is true, then process A is executed. If it is false, process B is executed. Both processes A and B flow to a common point.



The two flow lines emerging from the process blocks meet at the collector node. The logic continues along the same line. The process blocks can be expanded to include as many statements as desired. Therefore, it is possible to have an ifthenelse structure within an ifthenelse structure.

C. Repetition

This structure, also referred to as dowhile, provides for the repetitive execution of process blocks.



In this structure the flow passes through the collector node to the decision symbol where a test is made. Here the logical expression t is evaluated. If the result of the evaluation is true, then process A is executed and t is evaluated again. If it is false, then A is not executed. If t contains a variable which would be initialized before the execution of the block, this variable would then be modified as a part of process A. The modification of the control variable affecting the test is a necessity; otherwise, the program would be in an endless loop. The process block of the dowhile could contain sequence structures in any combination to represent any kind of logic.

Figure 3-1 shows a structured flowchart of an algorithm that finds the largest number in a set of 100 numbers, and the position of that number in the set. The set of numbers is a vector array called A. The variable M points to the position in A where the largest value of the set is found.

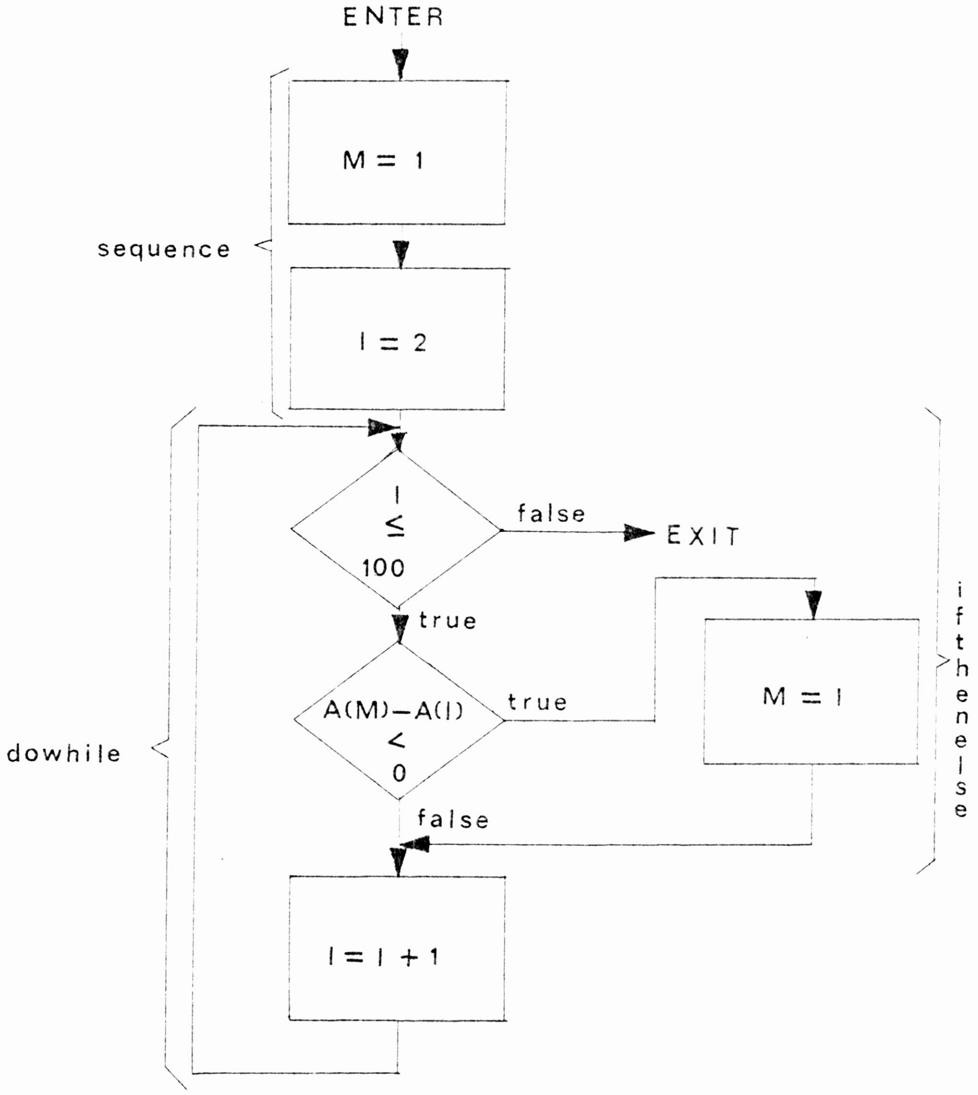


Figure 3-1. Structured flowchart example.

Using the basic structures just defined, it is possible to write programs that have no GO TO statements. For this reason, structured programming is sometimes referred to as GO TO less programming. However, this is a too narrow definition and impossible in some programming languages. Structured programming is not merely GO TO less programming. It is a discipline approach that encompasses a number of techniques in an effort to produce clear readable code [Hughes, Michtom 1977]. Structured programs that do not contain GO TO statements will have a logic flow from top to bottom. The specified program statements will be executed in the order in which they appear in the source program listing.

ADVANTAGES

When a program is designed and coded using the basic structures described in this section, the benefits are fourfold [Aron 1974]:

1. The program will contain fewer coding errors.
2. It will be easy to read.
3. It will be easy to debug.
4. It will be easy to maintain and modify.

Once the structured approach is mastered, it is possible to produce code that has few or no errors. The need for detailed flowcharts to explain a program is reduced or eliminated. Documentation time is also reduced because the programs themselves are self-documented. Programmer productivity is increased, thereby reducing personnel costs.

PART II

1. SOFTWARE SCIENCE AND ITS ELEMENTS

The rapid increase in the complexity of computer programs has increased the need for objective measures of the quality and complexity of software. To satisfy this need, a theory known as software science has been developed and refined by Maurice Halstead [1977]. Software science provides precise, objective measures of the complexity of the programs. It predicts the length of programs, and estimates the amount of time an average programmer would spend in the implementation of a given algorithm. Numerous statistical studies have shown very high correlations between the theory's predictions and the actual measures such as programming time. Software science does all these measures by simply counting the number of operators and operands in a program.

Halstead undertook a series of empirical studies of algorithms in order to prove the correlation of the count of operators and operands to the number of bugs encountered in a program. The correlation found was surprisingly strong. From this finding, Halstead hypothesized that algorithms may be characterized by a set of invariant laws.

The theory of software science has grown to include quantitative measures of:

1. Program level.
2. Intelligence content of a program.
3. Programming effort.
4. Language level.
5. Program purity.
6. Program clarity.
7. The effect of modularization.
8. Programming time.

In this section only the program length, volume, level, intelligence content, and effort are discussed. The remaining sections deal specifically with these five measures.

MEASURABLE PROPERTIES OF ALGORITHMS

An algorithm is a fixed, step by step procedure for accomplishing a given result. A program is an implementation of an algorithm using a list of coded instructions in a computer language.

The operands of a program, which are the variables and constants, can be identified and tabulated. The operators of a program, which affect the value or the ordering of operands, can be tabulated similarly. Halstead [1977] defined four basic measures which can be determined from these tabulations:

n_1 = The number of distinct operators appearing in a program.

n_2 = The number of distinct operands appearing in a program.

N_1 = The total number of occurrences of the operators in a program.

N_2 = The total number of occurrences of the operands in a program.

The size of the vocabulary is defined to be:

$$n = n_1 + n_2$$

To illustrate these metrics, a small subroutine written in FORTRAN is analyzed. Display 1 shows this program, and Tables 1 and 2 show the counts of operators and operands for the program. This example will be used throughout this section.

Display 4-1

```
   SUBROUTINE SORT (X,N)
   DIMENSION X(N)
   IF (N.LT.Z) RETURN
   DO 20 I = 2,N
      DO 10 J = 1,I
         IF (X(I).GE.X(J)) GO TO 10
         SAVE = X(I)
         X(I) = X(J)
         X(J) = SAVE
10 CONTINUE
20 CONTINUE
   RETURN
   END
```

Table 4-1. Operators of the Program of Display 4-1.

	Operator	Count
1	End of statement	7
2	Array subscript	6
3	=	5
4	IF ()	2
5	DO	2
6	,	2
7	End of program	1
8	.LT.	1
9	.GE.	1
n_1 = 10	GO TO 10	1
		28 = N_1

Table 4-2. Operands of the Program of Display 4-1.

	Operand	Count
1	X	6
2	I	5
3	J	4
4	N	2
5	2	2
6	SAVE	2
n_2 = 7	1	1
		22 = N_2

The concept that an algorithm consists of operators and operands only is most easily verified by considering simple digital computers whose instruction format consists of only two parts: an operation code and an operand address. Generalization of these concepts to computer languages is simply by induction [Halstead 1977].

Program Length

The program length can be obtained by simply adding the total number of operators and the total number of operands in a program. The program length equation is given by:

$$N = N_1 + N_2$$

For our example, the program length will be:

$$N = 28 + 22 = 50$$

Program Volume

The volume of an implementation of an algorithm is defined as:

$$V = N \log_2 n$$

This definition comes about, because for each of the N elements of a program, $\log_2 n$ bits must be specified to choose one of the operators or operands for that element. Thus, V measures the number of bits required to specify a program. If a program is translated into another language, its volume will change. For example, if a program is translated from PL/1 to assembler language, its volume will increase. It requires more operators and operands to express the same algorithm in assembler language than in a higher-level language.

Because of differences in the program volume of implementations in different languages, Halstead developed the concept of the potential volume. The potential volume is the most compact (highest-level) representation of an algorithm. It only considers the distinct number of operators and operands. The formula of the potential volume has the following configuration:

$$V^* = (n_1 + n_2) \log_2 (n_1 + n_2)$$

Going back to our example, the program volume and the potential volume are:

$$V = 204 \text{ bits}$$
$$V^* = 69 \text{ bits}$$

Program Level

The program level indicates how concise a program is. The implementation of a program in a high-level language produces a high program level.

Halstead hypothesized a conservation law between the level of a program and its volume:

$$LV = \text{constant}$$

From this hypothesis Halstead concluded that the program level L is the ratio of its potential volume to its actual volume.

$$L = V^*/V$$

In our example, we find that the program level is equal to:

$$L = 0.338$$

<u>Intelligence Content</u>

The intelligence content measure was developed to specify quantitatively how much detail was used in a program.

It is intuitively clear that a program written in machine language requires a greater amount of detail to say the same thing that it does in FORTRAN. In the past there was no way to measure how much has been said in either case. The intelligence content of a program answers the "how much" question.

Halstead defined the intelligence content as:

$$I = \frac{2N_2}{n_1 N_2} \times V$$

Referring back to our example, the resulting intelligence content for that particular program is:

$$I = 13$$

Programming Effort

The difficulty of programming increases as the volume of a program increases, and decreases as the program level increases. Thus, Halstead suggested the ratio

$$E = V/L$$

as a measure of the mental effort required to create a program. For the FORTRAN example,

$$E = 604$$

2. INTRODUCTION TO THE PROGRAM STUDY

Before algorithms could be measured, a comparative analysis of different implementations could not be performed. Differences between implementations was only a matter of speculation. For example, the length of a program could have been estimated by the number of statements in that program. However, this estimation was not accurate because some programming languages allow multiple instructions in a statement while some others do not.

Thanks to Maurice Halstead, who developed the concepts of software science, today it is possible to analyze different implementations of algorithms and to compare them numerically.

Many studies on the differences of implementations of algorithms have been conducted using the software science approach. Most of these studies were conducted in the area of programming languages. In these studies, the same algorithm was implemented in different languages, and the measuring equations of software science were applied to each implementation.

In this research, I followed these studies with the introduction of programming structures. The objective of my research was to measure the differences of structured and unstructured programs using the concepts of software science. The measures used in this study were:

1. the program length;
2. the program volume;
3. the program level;
4. the intelligence content of a program;
5. the programming effort.

PROCEDURE

For the structured-unstructured program study it was important to find well coded structured and unstructured programs that were implementations of the same algorithms. Because of the lack of resources and the time constraints imposed on this research, it was impossible to find a considerable number of programmers who could implement algorithms in both forms. For this reason, I have selected the option of finding unstructured algorithms already implemented, and converting these algorithms to structured code. For this purpose, I obtained 15 algorithms from the ACM communications written in unstructured FORTRAN, and translated these programs to structured PL/1. I kept the translation strictly oriented to the structure of the programs, not letting the differences of the languages interfere with the conversion. Then, I tabulated the measurable properties of each of the programs. The last part of the study consisted in the application of the software science equations and in the evaluation of the results.

This procedure was selected because FORTRAN does not support structured programming in its basic form, while PL/1 does. Therefore, a comparative analysis of program structures was possible using these two languages.

The next two sections explain in detail the procedure followed in the structured-unstructured program study.

3. THE PROCESS OF TRANSLATION AND COMPUTATION OF PROPERTIES

This section explains in detail the process of translation of the15 unstructured FORTRAN programs to structured PL/1. The computation of the measurable properties of both versions of the 15 programs is also described here.

THE TRANSLATION OF THE PROGRAMS

The 15 FORTRAN programs selected for this study were programs which allowed a structured translation without too much complication.

The translation of the programs was based on three objectives:

1. To maintain the flow of control of the structured versions sequential.
2. To use the three basic structures of structured programming at all times.
3. To avoid redundancies and unnecessary repetitions of statements.

In the translation of the programs, none of the statements were modified. Only their flow of control was altered with the introduction of the program structures of structured programming. When an IF statement was encountered in the

FORTRAN version of a program, it was replaced by the IF-THEN instruction of PL/1. When the true exit of the FORTRAN IF statement involved a GO TO instruction, the statements referred by that GO TO operator were attached to the THEN exit in the translation by means of a DO-END instruction. This procedure was followed only when the false exit of the FORTRAN IF statement was another GO TO instruction, or when the statements referred by the true exit of the FORTRAN IF statement involved some kind of program termination, such as a RETURN instruction. Display 3-1 shows this situation:

Display 3-1

<u>FORTRAN</u>

```
   IF (X.EQ.Y) GO TO 10
   -----
   -----
10 Z = X + Y
   -----
   RETURN
```

<u>PL/1</u>

```
IF X = Y THEN
  DO;
     Z = X + Y
     -----
     RETURN
  END;
-----
-----
```

In cases where the statements that followed the false exit of the FORTRAN IF statement met with the statements referred by the GO TO instruction in the true exit of the FORTRAN IF, the following procedure was taken: The test operator(s) of the FORTRAN IF statement were changed in order to alter the exits of this statement. Then the instructions in the false exit of the FORTRAN IF statement were placed in the THEN part of the PL/1 translation, and the statements referred by the GO TO instruction were placed after the IF-THEN structure. In display 3-2 we can see an example of this procedure.

Display 3-2

<u>FORTRAN</u>

```
   IF (X.GT.Y) GO TO 10
   Y = Y + 1
   -----
   -----
10 Z = X + Y
   -----
   -----
   -----
   -----
```

<u>PL/1</u>

```
IF (X <= Y) THEN
  DO;
    Y = Y + 1
    -----
    -----
  END;
Z = X + Y
-----
-----
-----
```

Another situation encountered with FORTRAN IF statements was when both exits involved GO TO instructions. For the translation of these type of structures, the ELSE operator was introduced. Display 3-3 shows a typical example of this situation.

Display 3-3

FORTRAN

```
   IF (X.EQ.Y) GO TO 10
   Y = Y + 1
   -----
   -----
   GO TO 15
10 Z = X + Y
   -----
   -----
   -----
15 X = X + 1
```

PL/1

```
IF X = Y THEN
  DO;
     Z = X + Y
     -----
     -----
     -----
  END;
ELSE
  DO;
     Y = Y + 1
     -----
     -----
  END;
X = X + 1
```

More complex cases came across in the translation of the programs. For example, GO TO instructions not found in IF statements, that referred to statements later in the program. To solve this situation, it was necessary to rewrite sets of statements several times where these statements were invoked. This measure was necessary in order to maintain a sequential flow of control. Another complex case encountered was when a GO TO instruction referred to statements earlier in the program. The procedure followed in this circumstance involved the creation of a dowhile structure. For this purpose, every time a dowhile structure was created, a new control variable was introduced. Some additional instructions that changed the value of the control variable were added inside the dowhile structure for a proper exit of the loop. A visual representation of this case is found in Display 3-4.

Display 3-4

FORTRAN

```
10 X = X + 1
   -----
   -----
   IF (X.GT.10) GO TO 15
   -----
   -----
   GO TO 10
15 -----
   -----
```

Display 3-4 (Continued)

PL/1

```
      KK = 1
LP1:  DO WHILE (KK = 1)
      -----
      -----
      IF X > 10 THEN KK = 0
      ELSE
          DO;
             -----
             -----
          END;
      END LP 1;
      -----
      -----
      -----
```

The remaining situations that required changes in the translation of the programs were combinations of the previously described cases.

As mentioned before, the objectives of the translation were to use the three basic structures of structured programming (sequence, ifthenelse, dowhile), to keep the programs sequential, and to avoid the repetition of statements whenever possible. A less important objective of the translation of the FORTRAN programs to structured PL/1 was the elimination of GO TO instructions. In some instances however, this objective was not fulfilled. Some of the programs contained GO TO instructions which exit loop structures, and whose elimination would have only produced long repetitions of statements and more unreadable code.

In any respect, the references of those GO TO instructions which were not eliminated corresponded to statements later in the program, thus the flow of control of the programs remained sequential.

The 15 FORTRAN programs can be found in Appendix A. Their translation to structured PL/1 are in Appendix B.

THE COMPUTATION OF THE MEASURABLE PROPERTIES

The measurable properties of algorithms consist of the number of distinct operators (n_1), the number of distinct operands (n_2), the total number of operators (N_1), and the total number of operands (N_2). The manner in which these properties were computed for both versions of each of the 15 programs is the topic of this subsection.

All the operators and operands of the 30 programs were computed by inspection. That is, every occurrence of each operator and operand was counted and tabulated in a tally form. During the computation of the measurable properties a computer environment was created. The variables in a calling statement of a subroutine were considered to be the same variables found in the parameter list of the called subroutine. Display 3-5 shows this situation.

Display 3-5

```
        CALL SUB1 (X,Y)
        SUBROUTINE SUB1 (A,B)
            or
SUB1:   PROC (A,B)
```

The names X and A, and Y and B represented the same variables. In the tabulation, if the calling and receiving names were the same, this name appeared only once. On the other hand, if the names were different, both names appeared in the tabulation separated by commas. However, they both corresponded to a single count. Local variables to each subroutine were treated as different variables. When the same name was used for more than one local variable, a subscript was attached to indicate the difference. For example, if the name "I" was used in two subroutines, and these names were independent, the first one was called "I(a)" and the second "I(b)".

Appendix C and D contain the tabulation of the operators and operands for the FORTRAN and PL/1 programs.

Table 3-1 summarizes the measurable properties of the 30 programs. In this table we can see that the number of distinct operators (n_1) for the structured programs decreases considerably. This decrease is due to the elimination of the GO TO instructions found in the FORTRAN programs. We can also see that the number of distinct operands (n_2)

Table 3-1

The Measurable Properties of the Programs

	Unstructured				Structured			
Algorithm #	n_1	N_1	n_2	N_2	n_1	N_1	n_2	N_2
365	34	230	38	169	21	294	41	234
286	27	179	21	148	17	189	21	155
424	27	717	101	662	26	732	104	680
500	52	717	69	589	25	720	71	600
502	60	740	80	591	29	720	81	634
505	26	159	22	121	16	200	23	171
509	34	273	45	246	25	298	46	271
512	13	443	34	417	13	443	34	417
513	36	255	31	201	22	269	34	226
515	20	87	16	79	19	94	16	86
518	24	183	38	152	19	162	37	139
521	53	400	77	303	27	459	74	362
523	60	667	65	652	36	673	65	694
529	33	377	55	348	24	401	57	378
533	54	653	70	608	39	682	77	602

slightly increases for the PL/1 programs. The introduction of control variables for the dowhile structures accounts for this increase. The total number of operators (N_1) and the total number of operands (N_2) also exhibit a noticeable increase. The reason for this difference is the repetition of statements necessary to preserve the sequentiality of the programs.

4. THE MEASURES OF COMPLEXITY AND THEIR MEANINGS IN THE STUDY

Going back to section two, we recall that the measures of complexity used in this study were the program length (N), the program volume (V), the program level (L), the intelligence content (I), and the programming effort (E). This section explains the results obtained from the application of the measures of complexity equations to the measurable properties described in the previous section. The final conclusions of the structured-unstructured program study are also described in this section.

Table 4-1 shows the measures of complexity results for both versions of the 15 programs. The results show that the program length for the structured versions has increased. This increase is as high as 30%. The reason for this increase is the repetition of statements necessary for maintaining the sequentiality of the structured programs. Similarly, the program volume of the PL/1 translations is greater than the volume of their FORTRAN counterparts. The difference between these two implementations is as high as 25%. As in the program length, the increase in the volume for the structured programs is due to the repetition of statements. Table 4-1 also indicates that the program level

Table 4-1

Measures of Complexity of the Programs

Algorithm #	Unstructured						Structured					
	N	V	V*	L	I	E	N	V	V*	L	I	E
365	399	2462	444	.180	32	13678	528	3144	369	.117	53	26872
386	327	1826	268	.147	20	12422	344	1805	199	.110	29	16409
424	1379	9653	896	.093	106	103796	1412	9916	913	.092	119	107783
500	1306	9036	837	.093	45	97116	1320	8692	632	.073	82	119068
502	1331	9489	998	.105	47	90371	1354	9182	746	.081	81	113358
505	280	1564	268	.171	22	9146	371	1961	206	.105	33	18676
509	519	3272	498	.152	35	21526	569	3499	437	.125	49	27992
512	860	4777	261	.055	62	86855	860	4777	261	.055	62	86855
513	456	2766	406	.147	25	18816	495	2875	325	.113	40	25442
515	166	858	186	.217	17	3954	180	923	180	.195	18	4733
518	335	1995	369	.185	42	10784	301	1748	325	.186	49	9398
521	703	4937	913	.185	49	26686	821	5466	672	.123	82	44439
523	1319	9188	871	.095	28	96716	1367	9102	672	.074	46	123000
529	725	4683	568	.121	47	38702	779	4939	514	.104	62	47490
533	1261	8769	862	.098	37	89480	1284	8806	796	.090	58	97844

of the translated programs has decreased. The largest drop encountered was of 39%. As we have seen in section one, the level of a program is the ratio of its potential volume to its actual volume. Since the number of distinct operators is less for the PL/1 programs, their corresponding potential volumes are also smaller. Therefore, the ratio of the potential to the actual volume for the strcutured programs is smaller than the same ratio for the unstructured programs. The amount of detailed imformation or the intelligence content of the PL/1 programs, as shown in Table 4-1, is greater than the intelligence content of the original programs. In some cases, the translated programs contain twice the amount of detailed information found in their FORTRAN versions. Again, the repetition of statements for maintaining the sequentiality of the structured programs is the reason for this increase. Finally, the table shows a noticeable increase in the programming effort for the structured PL/1 programs. Going back to the section on Software Science, we find that the programming effort is directly related to the program volume, and inversally related to the program level. Because the former increases and the latter decreases for the structured programs, the resulting programming effort for these programs is higher.

CONCLUSIONS

The objective of the structured-unstructured program study was to numerically measure the differences of structured and unstructured programs by measuring their program length, volume, level, intelligence content, and effort. Before this study was realized, there were no specific qualities which could have determined the difference of these two types of program implementation.

In order to enhance the meaning of the previously described results, averages were taken of the numbers in Table 4-1. The averages of the measures of complexity are summarized in Table 4-2.

Table 4-2

Average of the Measures of Complexity

	Unstructured	Structured
N	758	800
V	5,018	5,122
L	0.136	0.110
I	41	58
E	48,003	57,957

This table shows that on the average, the program length, volume, intelligence content, and programming effort of the unstructured programs are smaller than the same measures of complexity of the structured programs. Table 4-2 also shows that the program level of the original version of the programs is greater than the level of their translations.

Relating these averaged results to each other we obtain the comparative percentages shown in Table 4-3.

Table 4-3

Comparison of Structured and Unstructured Programs

Length (N)	6% higher for structured programs
Volume (V)	2% higher for structured programs
Level (L)	19% lower for structured programs
Intelligence (I)	41% higher for structured programs
Effort (E)	21% higher for structured programs

These comparative results may seem to indicate a preference in using unstructured programming. For this reason, it is important to emphasize the advantages that structured programming provides. Table 4-3 shows a considerable increase in the intelligence content for the structured programs. This is an indication that this type of programs are almost self-documented because of the amount of detailed information they contain. This self-documentation increases the readability of structured programs, and along with their sequentiality contributes to ease their modification, maintenance, and debugging. Thus in the long run, the implementation of algorithms in structured programming provides more economical and efficient results.

APPENDICES

APPENDIX A

THE FORTRAN PROGRAMS

ALGORITHM 365
COMPLEX ROOT FINDING [C5]

H. Bach (Recd. 18 Apr. 1968 and 15 July 1969)
Laboratory of Electromagnetic Theory, Technical
University of Denmark, Lyngby, Denmark

```
      SUBROUTINE CRF (ZS,HS,HM,DM,FUNC,DS,ZE,HE,DE,N)

      REAL W(3)
      COMPLEX ZO,ZS,ZE,ZD,ZZ,Z(3),CW,A,V,U(7),FUNC
      U(1)=(1.,0.)
      U(2)=(0.8660254,0.5000000)
      U(3)=(0.0000000,1.0000000)
      U(4)=(0.9659258,0.2588290)
      U(5)=(0.7071068,0.7071068)
      U(6)=(0.2588190,0.9659258)
      U(7)=(-0.2588190,0.9659258)
      H=HS
      ZO=ZS
      N=0
      CW=FUNC(ZO)
      WO=ABS(REAL(CW))+ABS(AIMAG(CW))
      DS=WO
      IF(WO-DM) 18,18,1
    1 K=1
      I=0
    2 V=(-1.,0.)
    3 A=(-0.5,0.866)
    4 Z(1)=ZO+H*V*A
      CW=FUNC(Z(1))
      W(1)=ABS(REAL(CW))+ABS(AIMAG(CW))
      Z(2)=ZO+H*V
      CW=FUNC(Z(2))
      W(2)=ABS(REAL(CW))+ABS(AIMAG(CW))
      Z(3)=ZO+H*CONJG(A)*V
      CW=FUNC(Z(3))
      W(3)=ABS(REAL(CW))+ABS(AIMAG(CW))
      N=N+1
      IF(W(1)-W(3)) 5,5,6
    5 IF(W(1)-W(2)) 7,8,8
    6 IF(W(2)-W(3)) 8,8,9
    7 NR=1
      GOTO 10
    8 NR=2
      GOTO 10
```

```
 9 NR=3
10 IF(WO-W(NR)) 11,12,12
11 GOTO (13,14,15),K
12 K=1
   I=0
   A=(0.707,0.707)
   V=(Z(NR)-ZO)/H
   WO=W(NR)
   ZO=Z(NR)
   IF(WO-DM) 18,18,4
13 K=2
   IF(H.LT.HM) GOTO 18
   H=H*0.25
   GOTO 3
14 K=3
   H=H*4.
   GOTO 2
15 I=I+1
   IF(I-7) 16,16,17
16 V=U(I)
   GOTO 3
17 IF(H.LT.HM) GOTO 18
   H=H*0.25
   I=0
   GOTO 2
18 ZE=ZO
   HE=H
   DE=WO
   RETURN
   END
```

ALGORITHM 386
GREATEST COMMON DIVISOR OF n INTEGERS AND MULTIPLIERS* [A1]

Gordon H. Bradley (Recd. 14 Oct. 1969, 28 Nov. 1969, and 26 Feb. 1970)
Administrative Sciences Department, Yale University, New Haven, CT 06520

```
      SUBROUTINE GCDN
     *  (IN,A,Z,IGCD)

      DIMENSION A(50),Z(50)
      INTEGER A,Z,C1,C2,Y1,Y2,Q
      DO 1 M = 1,N
      IF(A(M).NE.0) GO TO 3
    1 Z(M) = 0
    2 IGCD = 0
      RETURN
    3 IF(M.NE.N) GO TO 4
      IGCD = A(M)
      Z(M) = 1
      RETURN
    4 MP1 = M + 1
      MP2 = M + 2
      ISIGN = 0
      IF(A(M).GE.0) GO TO 5
      ISIGN = 1
      A(M) = -A(M)
    5 C1 = A(M)
      DO 30 I = MP1,N
      IF(A(I).NE.0) GO TO 7
      A(I) = 1
      Z(I) = 0
      GO TO 25
    7 Y1 = 1
      Y2 = 0
      C2 = IABS(A(I))
   10 Q = C2/C1
      C2 = C2 - Q*C1
      IF(C2.EQ.0) GO TO 20
      Y2 = Y2 - Q*Y1
      Q = C1/C2
      C1 = C1 - Q*C2
      IF(C1.EQ.0) GO TO 15
      Y1 = Y1 - Q*Y2
    GO TO 10
 15 C1 = C2
    Y1 = Y2
 20 Z(I) = (C1 - Y1*A(M))/A(I)
    A(I) = Y1
    A(M) = C1
 25 IF(C1.EQ.1) GO TO 60
 30 CONTINUE
 40 IGCD = A(M)
    DO 50 J = MP2,I
    K = I - J + 2
    KK = K + 1
    Z(K) = Z(K)*A(KK)
 50 A(K) = A(K)*A(KK)
    Z(M) = A(MP1)
    IF(ISIGN.EQ.0) GO TO 100
    Z(M) + -Z(M)
100 RETURN
 60 IP1 = I + 1
    DO 65 J = IP1,N
 65 Z(J) = 0
    GO TO 40
    END
```

ALGORITHM 424
CLENSHAW-CURTIS QUADRATURE [D-1]

W. Morven Gentleman (recd. 5 Oct. 1970 and 13 Aug. 1971)
University of Waterloo, Waterloo, Ontario, Canada

```
      REAL FUNCTION CCQUAD (F,A,B,TCLERR,LIMIT,ESTERR,
           USED,CSXFRM)
      REAL F,A,B,TCLERR
      INTEGER LIMIT
      REAL ESTERR,CSXFRM(LIMIT)
      INTEGER USED
      REAL PI,RT3,CENTRE,WIDTH,SHIFT,FUND,ANGLE,C,S
      REAL CIDINT,NEWINT
      REAL T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12
      INTEGER N,N2,N3,N LESS 1,N LESS 3,MAX,M,MAX,J,STEP
      INTEGER L(8),L1,L2,L3,L4,L5,L6,L7,L8
      INTEGER J1,J2,J3,J4,J5,J6,J7,J8,J REV
      EQUIVALENCE (L(1),L1),(L(2),L2),(L(3),L3),(L(4),L4),
         (L(5),L5),(L(6),L6),(L(7),L7),(L(8),L8),(J8,J REV)
      DATA PI,RT3/   3.14159265 3589E0    1.73205080 7568E0 /
      DATA M MAX/   8 /
      CENTRE=(A+B)*.5E0
      WIDTH = (B-A)*.5E0
      MAX=MINO(LIMIT,2*3**(M MAX+1)
      DO 10 J=1,M MAX
         L(J) = 1
   10 CONTINUE
      N=6
      CSXFRM(1)=F(A)
      CSXFRM(7)=F(8)
      SHIFT=WIDTH*RT3*.5E0
      CSXFRM(2)=F(CENTRE-SHIFT)
      CSXFRM(6)=F(CENTRE+SHIFT)
      SHIFT=WIDTH*.5E0
      CSXFRM(3)=F(CENTRE-SHIFT)
      CSXFRM(5)=F(CENTRE+SHIFT
      CSXFRM(4)=F(CENTRE)
      T1=CSXFRM(1)+CSXFRM(7)
      T2=CSXFRM(1)-CSXFRM(7)
      T3=2.E0*CSXFRM(4)
      T4=CSXFRM(2)+CSXFRM(6)
      T5=(CSXFRM(2)-CSXFRM(6)*RT3
      T6=CSXFRM(3)+CSXFRM(5)
      T7=CSXFRM(3)-CSXFRM(5)
      T8=T1+2.E0*T6
      T9=2.E0*T4+T3
      T10=T2+T7
      T11=T1-T6
      T12=T4-T3
      CSXFRM(1)=T8+T9
      CSXFRM(2)=T10+T5
      CSXFRM(3)=T11+T12
      CSXFRM(4)=T2-2.E0*T7
      CSXFRM(5)=T11-T12
      CSXFRM(6)=T10-T5
      CSXFRM(7)=T8-T9
      USED=7
      GO TO 200
  100 DO 110 J=2,M MAX
        L(J-1)=L(J)
  110 CONTINUE
      L(M MAX)=3*L(M MAX-1)
      J=USED
      FUND=PI/FLOAT(3*N)
      DO 120 J1=1,L1,1
        DO 120 J2=J1,L2,L1
          DO 120 J3=J2,L3,L2
            DO 120 J4=J3,L4,L3
              DO 120 J5=J4,L5,L4
                DO 120 J6=J5,L6,L5
                  DO 120 J7=J6,L7,L6
                    DO 120 J8=J7,L8,L7
                      ANGLE=FUND*FLOAT(3*J REV-2)
                      SHIFT=WIDTH*COS(ANGLE)
                      T1=F(CENTRE-SHIFT)
                      T3=F(CENTRE+SHIFT)
                      SHIFT=WIDTH*SIN(ANGLE)
                      T2=F(CENTRE+SHIFT)
                      T4=F(CENTRE-SHIFT)
                      T5=T1+T3
                      T6=T2+T4
                      CSXFRM(J+1)=T5+T6
                      CSXFRM(J+2)=T1-T3
                      CSXFRM(J+3)=T5-T6
                      CSXFRM(J+4)=T2-T4
                      J=J+4
  120 CONTINUE
      N2=2*N
      STEP=4
  150 J1=USED+STEP
      J2=USED+2*STEP
      CALL R3PASS (N2,STEP,N2-2*STEP,CSXFRM(USED+1),
         CSXFRM(J1+1),CSXFRM(J2+1))
      STEP=3*STEP
      IF (STEP .LT. N) GO TO 150
      T1=CSXFRM(1)
      T2=CSXFRM(USED+1)
      CSXFRM(1)=T1+2.E0*T2
      CSXFRM(USED+1)=T1-T2
      T1=CSXFRM(N+1)
      T2=CSXFRM(N2+2)
      CSXFRM(N+1)=T1+T2
      CSXFRM(N2+2)=T1-2.E0*T2
      N3=3*N
      N LESS 1=N-1
      DO 180 J=1,N LESS 1
         J1=N+J
         J2=N3-J
         ANGLE=FUND*FLOAT(J)
         C=COS(ANGLE)
         S=SIN(ANGLE)
         T1=C*CSXFRM(J1+2)-S*CSXFRM(J2+2)
         T2=(S*CSXFRM(J1+2)+C*CSXFRM(J2+2)*RT3
         CSXFRM(J1+2)=CSXFRM(J+1)-T1-T2
         CSXFRM(J2+2)=CSXFRM(J+1)-T1+T2
         CSXFRM(J+1)=CSXFRM(J+1)+2.E0*T1
  180 CONTINUE
      T1=CSXFRM(N2+1)
      T2=CSXFRM(N2+2)
      DO 190 J=1,N LESS 1
         J1=USED+J
         J2=N2+J
         CSXFRM(J2)=CSXFRM(J1)
         CSXFRM(J1)=CSXFRM(J2+2)
  190 CONTINUE
      CSXFRM(N3)=T1
      CSXFRM(N3+1)=T2
      N=N3
      USED=N+1
      GO TO 210
  200 OLDINT=(T1+2.E0*T3)/3.E0
  210 N LESS 3=N-3
      NEWINT=.5E0*CSXFRM(USED)/FLOAT(1-N**2)
      DO 220 J=1,N LESS 3,2
         J REV=N-J
         NEWINT=NEWINT+CSXFRM(J REV)/FLOAT(J REV*(2-J REV)
  220 CONTINUE
      NEWINT=NEWINT+.5E0*CSXFRM(1)
      ESTERR=ABS(OLDINT*3.E0-NEWINT)
      IF (ABS(NEWINT)*TCLERR .GE. ESTERR) GO TO 400
      OLDINT=NEWINT
      IF (3*N+1 .LE. MAX) GO TO 100
  400 CCQUAD=WIDTH*NEWINT/FLOAT(N/2)
      ESTERR=WIDTH*ESTERR/FLOAT(N/2)
      RETURN
      END

      SUBROUTINE P3PASS (N2,M,LENGTH,X0,X1,X2)
      INTEGER N2,M,LENGTH
      REAL X0(LENGTH),X1(LENGTH),X2(LENGTH)
      INTEGER HALF M,M3,K,K0,K1,J,J0,J1
      REAL TWOPI,HAFRT3,PSUM,RDIFF,PSUM2,ISUM,IDIFF,IDIFF2
      REAL FUND,ANGLE,C1,S1,C2,S2,R0,R1,R2,I0,I1,I2
      DATA TWOPI, HAFRT3/   6.283185307E0,    .866025403E0 /
      HALF M=(M-1)/2
      M3=M*3
      FUND=TWOPI/FLOAT(M3)
      DO 10 K=1,N2,M3
         RSUM=(X1(K)+X2(K))
         RDIFF=(X1(K)-X2(K))*HAFRT3
         X1(K)=X0(K)-RSUM*.5E0
         X2(K)=PDIFF
         X0(K)=X0(K)+RSUM
   10 CONTINUE
      J=M/2+1
      DO 20 K=J,N2,M3
         RSUM=(X1(K)+X2(K))*HAFRT3
         RDIFF=(X1(K)-X2(K))
         X1(K)=X0(K)-RDIFF
         X2(K)=RSUM
         X0(K)=X0(K)+RDIFF*.5E0
   20 CONTINUE
      DO 40 J=1,HALF M
         J0=J+1
         J1=M-J+1
         ANGLE=FUND*FLOAT(J)
         C1=COS(ANGLE)
         S1=SIN(ANGLE)
         C2=C1**2-S1**2
         S2=2.E0*S1*C1
         DO 30 K0=J0,N2,M3
            K1=Ko-J0+J1
            R0=X0(K0)
            I0=X0(K1)
            R1=C1*X1(K0)-S1*X1(K1)
            I1=S1*X1(K0)+C1*X1(K1)
            R2=C2*X2(K0)-S2*X2(K1)
            I2=S2*X2(K0)+C2*X2(K1)
            RSUM=P1+P2
```

```
          RDIFF=(R1-R2)*HAFRT3
          RSUM2=RO-.5EO*RSUM
          ISUM=I1+I2
          IDIFF=(I1-I2)*HAFRT3
          IDIFF2=IO-.5EO*ISUM
          XO(KO)=RO+RSUM
          XO(K1)=RSUM2+IDIFF
          X1(KO)=RSUM2-IDIFF
          X1(K1)=RDIFF+IDIFF2
          X2(KO)=RDIFF-IDIFF2
          X2(K1)=IO+ISUM
30      CONTINUE
40    CONTINUE
      RETURN
      END
```

ALGORITHM 500
MINIMIZATION OF UNCONSTRAINED MULTIVARIATE FUNCTIONS [E4]

D. F. Shanno and K. H. Phua
University of Toronto, Canada

```
      SUBROUTINE MINI(N, X, F, G, H, M, IH, STP, EPS, FEST,
     * MAXF, MODE, ITER, IFUN, IERR, CALCFG)
      DOUBLE PRECISION F, FEST, EPS, X(N), G(N), STP(N),
     * H(M)
      DOUBLE PRECISION D(1Ø), XX(1Ø), GG(1Ø), EPS2, DUM,
     * STEP, SUM, G1, G2
      DOUBLE PRECISION STPMAX, ALPHA, AL, BL, FG, F1, FM,
     * GM, AA, BB, CC
      DOUBLE PRECISION SS, DY, YHY, C1, C2, SQRT, DABS
      LOGICAL CONV
      CALL CALCFG(N, X, R, G)
      IFUM = 1
      ITER = Ø
      IERR = Ø
      EPS2 = EPS*EPS
      DUM = Ø.
      STEP = Ø
      DO 1Ø I=1,N
        DUM = DUM + G(I)*G(I)
        STEP = STEP + STP(I)*STP(I)
   1Ø CONTINUE
      IF (DUM.EQ.Ø.) RETURN
      IF (IH.EQ.Ø) GO TO 2Ø
      GO TO 5Ø
   2Ø DUM = 1ØØ.*DSQRT(STEP/DUM)
      IJ = 1
      DO 4Ø I=1,N
        DO 3Ø J=I,N
          H(IJ) = Ø.
          IF (I.EQ.J) H(IJ) = DUM
          IJ = IJ + 1
   3Ø   CONTINUE
   4Ø CONTINUE
   5Ø CONTINUE
   6Ø SUM = Ø.
      DO 11Ø I=1,N
        DUM = Ø.
        IJ = I
        IF (I.GT.1) GO TO 7Ø
        GO TO 9Ø
   7Ø   II = I - 1
```

```
        DO 8Ø J=1,II
          DUM = DUM - H(IJ)*G(J)
          IJ = IJ + N - J
  8Ø    CONTINUE
  9Ø    CONTINUE
        DO 1ØØ J=I,N
          DUM = DUM - H(IJ)*G(J)
          IJ = IJ + 1
 1ØØ    CONTINUE
        D(I) = DUM
        XX(I) = X(I)
        GG(I) = G(I)
        SUM = SUM + DUM*DUM
 11Ø  CONTINUE
      IF ((ITER.GE.1) .AND. (ITER.LE.N)) GO TO 12Ø
      GO TO 14Ø
 12Ø  DUM = DSQRT(STEP/SUM)
      DO 13Ø I=1,N
        D(I) = D(I)*DUM
 13Ø  CONTINUE
 14Ø  CONTINUE
      DO 17Ø J=1,2
        G1 = Ø.
        DO 15Ø I=1,N
          G1 = G1 + D(I)*G(I)
 15Ø    CONTINUE
        IF (G1.LT.Ø) GO TO 18Ø
        DO 16Ø I=1,N
          D(I) = -D(I)
 16Ø    CONTINUE
 17Ø  CONTINUE
 18Ø  IF (G1.GT.Ø.) GO TO 19Ø
      GO TO 2ØØ
 19Ø  IERR = 3
      RETURN
 2ØØ  CONTINUE
      STPMAX = 1.ØD+3Ø
      DO 21Ø I=1,N
        IF (DABS(D(I)).LT.1.D-3Ø) GO TO 21Ø
        DUM = DABS(STP(I)/D(I))
        IF (STPMAX.GT.DUM) STPMAX = .9*DUM
 21Ø  CONTINUE
      ALPHA = Ø.
      FG = G1
      AL = 1.
      IF (MODE.EQ.2) AL = 2.*(FEST-F)/G1
      IF (AL.GT.1.) AL = 1.
      IF (AL.GT.STPMAX) AL = Ø.8*STPMAX
      IF (AL.LT.Ø.) GO TO 22Ø
      GO TO 23Ø
```

```
220 IERR = 4
    RETURN
230 CONTINUE
240 F1 = F
    ALPHA = ALPHA + AL
    DO 250 I=1,N
      X(I) = XX(I) + ALPHA*D(I)
250 CONTINUE
    CALL CALCFG(N, X, F, G)
    IFUM = IFUM + 1
    IF (IFUN.GT.MAXF) GO TO 260
    GO TO 270
260 IERR = 1
    RETURN
270 CONTINUE
    G2 = 0.0
    DO 280 I=1,N
      G2 = G2 + G(I)*D(I)
280 CONTINUE
    CONV = (G2.GT.FG) .AND. (F1.GT.(F+.0001*G1))
    IF (CONV .AND. (MODE.EQ.2)) GO TO 290
    IF (F.GT.F1) .OR. (G2.GT.0.)) GO TO 290
    BL = AL
    AL = 0.5*G1BL*BL/(F1-F+BL*G1)
    IF ((MODE.EQ.2) .AND. (AL.LT.BL)) AL = 2.*(FEST-F)/G1
    IF (AL.GT.(10.*BL)) AL = 10.*BL
    IF (AL.GT.STPMAX) AL = 0.8*STPMAX
    IF (AL.LT.(1.001*BL)) AL = BL + BL
    G1 = G2
    GO TO 240
290 CONTINUE
    IF (.NOT.(CONV) .OR. (MODE.EQ.1)) GO TO 300
    GO TO 390
300 BL = AL
    AA = (G1+G2+2.*(F1-F)/AL)/(AL*AL)
    BB = (G2-3.*AA*AL*AL-G1)/(AL+AL)
    CC = BB*BB - 3.*AA*G1
    DUM = DABS(AA)*1.0D+05 - DABS(BB)
    AL = (.5*G1*BL*BL)/(F1-F+BL*G1)
    IF ((CC.GT.0.) .AND. (DUM.GT.0.)) AL = (-BB+DSQRT
   * (CC))/(3.*AA)
    FM = F
    GM = G2
    IF (AL.LE.(.001*BL)) AL = .1*BL
    IF (AL.GT.(.999*BL)) AL = .8*BL
    ALPHA = ALPHA - BL + AL
    DO 310 I=1,N
      X(I) = XX(I) + ALPHA*D(I)
310 CONTINUE
    CALL CALCFG(N, X, F, G)
```

```
      IFUN = IFUN + 1
      IF (IFUN.GT.MAXF) GO TO 32Ø
32Ø   IERR = 1
      RETURN
33Ø   CONTINUE
      G2 = Ø.
      DO 34Ø I=1,N
        G2 = G2 + D(I)*G(I)
34Ø   CONTINUE
      CONV = (G2.GT.FG) .AND. (F1.GT.(F+.ØØØ1*G1))
      IF (CONV) GO TO 38Ø
      IF ((G2.GT.FG) .AND. (BL.LE.EPS2)) GO TO 38Ø
      SS = Ø.
      DO 35Ø I=1,N
        SS = SS + DABS(G(I)*D(I))
35Ø   CONTINUE
      SS = SS/(1ØØØØØ.*FLOAT(N))
      IF ((G2.LT.SS) .AND. (G2.GT.FG)) GO TO 38Ø
      IF ((G2.LT.Ø.) .AND. (F1.GT.F)) GO TO 36Ø
      GO TO 37Ø
36Ø   G1 = G2
      G2 = GM
      F1 = F
      F = FM
      AL = BL - AL
      ALPHA = ALPHA + AL
37Ø   CONTINUE
      GO TO 3ØØ
38Ø   CONTINUE
39Ø   CONTINUE
      CONV = .TRUE.
      STEP = Ø.
      DO 4ØØ I=1,N
        D(I) = X(I) - XX(I)
        STEP = STEP + D(I)*D(I)
        IF ((G(I)*G(I)).GT.EPS2) CONV = .FALSE.
4ØØ   CONTINUE
      IF (CONV) RETURN
      IF (STEP.LE.(EPS2*EPS2)) GO TO 41Ø
      GO TO 42Ø
41Ø   IERR = 2
      RETURN
42Ø   CONTINUE
      DY = Ø.
      YHY = Ø.
      DO 43Ø I=1,N
        GG(I) = G(I) - GG(I)
        DY = DY + D(I)*GG(I)
43Ø   CONTINUE
      DO 48Ø I=1,N
```

```
          DUM = Ø.
          IJ = I
          IF (I.GT.1) GO TO 44Ø
          GO TO 46Ø
44Ø       II = I - 1
          DO 45Ø J=1,II
            DUM = DUM + H(IJ)*GG(J)
            IJ = IJ + N - J
45Ø       CONTINUE
46Ø       CONTINUE
          DO 47Ø J=I,N
            DUM = DUM + H(IJ)*GG(J)
            IJ = IJ + 1
47Ø       CONTINUE
          YHY = YHY + DUM*GG(I)
          XX(I) = DUM
48Ø     CONTINUE
        C1 = 1. + YHY/DY
        DO 49Ø I=1,N
          GG(I) = C1*D(I) - XX(I)
49Ø     CONTINUE
        IJ = 1
        DO 51Ø I=1,N
          C1 = D(I)/DY
          C2 = XX(I)/DY
          DO 5ØØ J=I,N
            H(IJ) = H(IJ) + C1*GG(J) - C2*D(J)
            IJ = IJ + 1
5ØØ       CONTINUE
51Ø     CONTINUE
        ITER = ITER + 1
        GO TO 6Ø
        END
```

ALGORITHM 502
DEPENDENCE OF SOLUTION OF NONLINEAR SYSTEMS ON A PARAMETER [C5]

Milan Kubicek
Prague Institute of Chemical Technology, Czechoslovakia

```
      SUBROUTINE DERPAR(N, X, XLOW, XUPP, EPS, W, INITAL,
     * ITIN, HH, HMAX, PREF, NDIR, E, MXADMS, NCORR,
     * NCRAD, NOUT, OUT, MAXOUT, NPRNT)

      DIMENSION X(11), XLOW(11), XUPP(11), W(11), HMAX(11),
     * PREF(11), NDIR(11), OUT(1ØØ,12), F(11), G(1Ø,11),
     * BETA(11), MARK(11), DXDT(11)
      DATA INDIC, INDSP /1H*,1H /
      N1 = N + 1
      LW = 3
      IF (INITAL) 1Ø, 6Ø, 1Ø
   1Ø DO 4Ø L=1,ITIN
        CALL FCTN(N, X, F, G)
        SQUAR = Ø.Ø
        DO 2Ø I=1,N
          SQUAR = SQUAR + F(I)**2
   2Ø   CONTINUE
        LL = L - 1
        SQUAR = SQRT(SQUAR)
        IF (NPRNT.NE.3) WRITE (LW,99999) LL, (X(I),I=1,N1),
     *  SQUAR
        CALL GAUSE(N, G, F, M, 1Ø, 11, PREF, BETA, K)
        IF (M.EQ.Ø) GO TO 31Ø
        P = Ø.Ø
        DO 3Ø J=1,N1
          X(J) = X(J) - F(J)
          P = P + ABS(F(J))*W(J)
   3Ø CONTINUE
      IF (P.LE.EPS) GO TO 5Ø
   4Ø CONTINUE
      WRITE (LW,99998) ITIN
      ITIN = -1
      IF (INITAL.EQ.1) RETURN
   5Ø IF (NPRNT.NE.3) WRITE (LW,99997) (X(I),I=1,N1)
      IF (INITAL.EQ.2) RETURN
   6Ø IF (NPRNT.NE.3) WRITE (LW,99996)
      KOUT = Ø
      NOUT = Ø
      MADMS = Ø
      NC = 1
      K1 = Ø
  7Ø  CALL FCTN(N, X, F, G)
      SQUAR = Ø.Ø
      DO 8Ø I=1,N
        SQUAR = SQUAR + F(I)**2
  8Ø  CONTINUE
      CALL GAUSE(N, G, F, M, 1Ø, 11, PREF, BETA, K)
      IF (M.EQ.Ø) GO TO 31Ø
      IF (K1.EQ.K) GO TO 9Ø
      MADMS = Ø
      K1 = K
  9Ø  SQUAR = SQRT(SQUAR)
      IF (NCRAD.EQ.1) SQUAR = -SQUAR
      P = Ø.Ø
      DO 1ØØ I=1,N1
        P = P + W(I)*ABS(F(I))
 1ØØ  CONTINUE
      IF (P.LE.EPS) GO TO 13Ø
      IF (NC.GE.NCORR) GO TO 12Ø
      DO 11Ø I=1,N1
        X(I) = X(I) - F(I)
 11Ø  CONTINUE
      NC = NC + 1
      GO TO 7Ø
 12Ø  IF (NCORR.EQ.Ø) GO TO 13Ø
      WRITE (LW,99995) NCORR, P
 13Ø  NC = 1
      IF (NCRAD.EQ.Ø) GO TO 15Ø
      DO 14Ø I=1,N1
        X(I) = X(I) - F(I)
 14Ø  CONTINUE
 15Ø  NOUT = NOUT + 1
      DO 16Ø I=1,N1
        MARK(I) = INDSP
 16Ø  CONTINUE
      MARK(K) = INDIC
      IF (NPRNT.EQ.3) GO TO 17Ø
      WRITE (LW,99994)
      WRITE (LW,99993) (X(I),MARK(I),I=1,N1), SQUAR
 17Ø  IF (NPRNT.EQ.1) GO TO 2ØØ
      IF (NOUT.LE.1ØØ) GO TO 18Ø
      WRITE (LW,99992)
      RETURN
 18Ø  DO 19Ø I=1,N1
        OUT(NOUT,I) = X(I)
 19Ø  CONTINUE
      OUT(NOUT,N+2) = SQUAR
      GO TO 21Ø
 2ØØ  IF (NOUT.EQ.1) GO TO 18Ø
```

```
  21Ø IF (NOUT.GE.MAXOUT) RETURN
      DO 22Ø I=1,N1
        IF (X(I).LT.XLOW(I) .OR. X(I).GT.XUPP(I)) GO TO 3ØØ
  22Ø CONTINUE
      IF (NOUT.LE.3) GO TO 24Ø
      P = Ø.Ø
      DO 23Ø I=1,N1
        P = P + W(I)*ABS(X(I)-OUT(1,I))
  23Ø CONTINUE
      IF (P.LE.E) GO TO 29Ø
  24Ø DXK2 = 1.Ø
      DO 25Ø I=1,N1
        DXK2 = DXK2 + BETA(I)**2
  25Ø CONTINUE
      DXDT(K) = 1.Ø/SQRT(DXK2)*FLOAT(NDIR(K))
      H = HH
      DO 27Ø I=1,N1
        NDIR(I) = 1
        IF (I.EQ.K) GO TO 26Ø
        DXDT(I) = BETA(I)*DXDT(K)
  26Ø   IF (DXDT(I).LT.Ø.Ø) NDIR(I) = -1
        IF (H*ABS(DXDT(I)).LE.HMAX(I)) GO TO 27Ø
        MADMS = Ø
        H = HMAX(I)/ABS(DXDT(I))
  27Ø CONTINUE
      IF (NOUT.LE.KOUT+3) GO TO 28Ø
      IF (H*ABS(DXDT(K)).LE.Ø.8*ABS(X(K)-OUT(1,K))) GO TO
     * 28Ø
      IF ((OUT(1,K)-X(K))*FLOAT(NDIR(K)).LE.Ø.Ø) GO TO 28Ø
      MADMS = Ø
      IF (H*ABS(DXDT(K)).LE.ABS(X(K)-OUT(1,K))) GO TO 28Ø
      H = ABS(X(K)-OUT(1,K))/ABS(DXDT(K))
      KOUT = NOUT
  28Ø CALL ADAMS(N, DXDT, MADMS, H, X, MXADMS)
      GO TO 7Ø
  29Ø WRITE (LW,99991)
      MAXOUT = -1
      RETURN
  3ØØ MAXOUT = -2
      RETURN
  31Ø WRITE (LW,9999Ø) (X(I),I=1,N1)
      MAXOUT = -3
      RETURN
99999 FORMAT (3X, 7H DERPAR, I3, 25H,INITIAL NEWTON
     * ITERATION/22X,11HX,ALFA,SQF=, 5F15.7/(33X, 5F15.7))
99998 FORMAT (/16H DERPAR OVERFLOW. I5, 2X, 18HINITIAL
     * ITERATIONS/)
99997 FORMAT (3X, 39H DERPAR AFTER INITIAL NEWTON
     * ITERATIONS/22X,7HX,ALFA=, 4X, 5F15.7/(33X, 5F15.7))
```

```
99996 FORMAT (/6X, 45H DERPAR RESULTS (VARIABLE CHOSEN AS
     * INDEPENDE, 18HNT IS MARKED BY *)/)
99995 FORMAT (/37H DERPAR NUMBER OF NEWTON CORRECTIONS=,
     * I5, 31H IS NOT SUFFICIENT,ERROR OF X=, F15.7/)
99994 FORMAT (6X, 27H DERPAR RESULTS X,ALFA,SQF=)
99993 FORMAT (33X, 5(F15.7, A1))
99992 FORMAT (/28H DERPAR OUTPUT ARRAY IS FULL//)
99991 FORMAT (/36H DERPAR CLOSED CURVE MAY BE EXPECTED/)
99990 FORMAT (/48H DERPAR SINGULAR JACOBIAN MATRIX FOR X
     * AND ALFA=, 5F12.6/(48X, 5F12.6))
      END


      SUBROUTINE GAUSE(N, A, B, M, NN, MM, PREF, BETA, K)

      DIMENSION A(NN,MM), B(MM), PREF(MM), BETA(MM), Y(11),
     * X(11), IRR(11), IRK(11)
      N1 = N + 1
      ID = 1
      M = 1
      DO 10 I=1,N1
        IRK(I) = 0
        IRR(I) = 0
   10 CONTINUE
   20 IR = 1
      IS = 1
      AMAX = 0.0
      DO 60 I=1,N
        IF (IRR(I)) 60, 30, 60
   30   DO 50 J=1,N1
          P = PREF(J)*ABS(A(I,J)
          IF (P-AMAX) 50, 50, 40
   40     IR = I
          IS = J
          AMAX = P
   50   CONTINUE
   60 CONTINUE
      IF (AMAX.NE.0.0) GO TO 70
      M = 0
      GO TO 150
   70 IRR(IR) = IS
      DO 90 I=1,N
        IF (I.EQ.IR .OR. A(I,IS).EQ.0.0) GO TO 90
        P = A(I,IS)/A(IR,IS)
        DO 80 J=1,N1
          A(I,J) = A(I,J) - P*A(IR,J)
   80   CONTINUE
        A(I,IS) = 0.0
        B(I) = B(I) - P*B(IR)
```

```
 9Ø CONTINUE
    ID = ID + 1
    IF (ID.LE.N) GO TO 2Ø
    DO 1ØØ I=1,N
      IR = IRR(I)
      X(IR) = B(I)/A(I,IR)
      IRK(IR) = 1
1ØØ CONTINUE
    DO 11Ø K=1,N1
      IF (IRK(K).EQ.Ø) GO TO 12Ø
11Ø CONTINUE
12Ø DO 13Ø I=1,N
      IR = IRR(I)
      Y(IR) = -A(I,K)/A(I,IR)
13Ø CONTINUE
    DO 14Ø I=1,N1
      B(I) = X(I)
      BETA(I) = Y(I)
14Ø CONTINUE
    B(K) = Ø.Ø
    BETA(K) = Ø.Ø
15Ø RETURN
    END


    SUBROUTINE ADAMS(N, D, MADMS, H, X, MXADMS)

    DIMENSION DER(4,11), X(11), D(11)
    N1 = N + 1
    DO 2Ø I=1,3
      DO 1Ø J=1,N1
        DER(I+1,J) = DER(I,J)
 1Ø   CONTINUE
 2Ø CONTINUE
    MADMS = MADMS + 1
    IF (MADMS.GT.MXADMS) MADMS = MXADMS
    IF (MADMS.GT.4) MADMS = 4
    DO 7Ø I=1,N1
      DER(1,I) = D(I)
      GO TO (3Ø, 4Ø, 5Ø, 6Ø), MADMS
 3Ø   X(I) = X(I) + H*DER(1,I)
      GO TO 7Ø
 4Ø   X(I) = X(I) + Ø.5*H*(3.Ø*DER(1,I)-DER(2,I))
      GO TO 7Ø
 5Ø   X(I) = X(I) + H*(23.Ø*DER(1,I)-16.Ø*DER(2,I)+5.Ø*
   *   DER(3,I))/12.Ø
      GO TO 7Ø
 6Ø   X(I) = X(I) + H*(55.Ø*DER(1,I)-59.Ø*DER(2,I)+37.Ø*
   *   DER(3,I)-9.Ø*DER(4,I)/24.Ø
 7Ø CONTINUE
```

```
RETURN
END
```

ALGORITHM 505
A LIST INSERTION SORT FOR KEYS WITH ARBITRARY KEY DISTRIBUTION [S20]

Wolfgang Janko
Hochschule für Welthandel, Austria

```
      SUBROUTINE SPN(K, L, II, JJ, MIN)

      INTEGER TAB(57)
      DIMENSION K(1), L(1)
      DATA TAB(1), TAB(2), TAB(3), TAB(4), TAB(5), TAB(6),
     * TAB(7), TAB(8), TAB(9), TAB(10), TAB(11), TAB(12),
     * TAB(13), TAB(14), TAB(15), TAB(16), TAB(17),
     * TAB(18), TAB(19), TAB(20), TAB(21), TAB(22),
     * TAB(23), TAB(24), TAB(25), TAB(26), TAB(27),
     * TAB(28), TAB(29), TAB(30), TAB(31), TAB(32),
     * TAB(33), TAB(34), TAB(35), TAB(36), TAB(37),
     * TAB(38), TAB(39), TAB(40), TAB(41), TAB(42),
     * TAB(43), TAB(44), TAB(45), TAB(46), TAB(47),
     * TAB(48), TAB(49), TAB(50), TAB(51), TAB(52),
     * TAB(53), TAB(54), TAB(55), TAB(56),
     * TAB(57) /3,5,10,17,26,37,50,65,82,101,122,145,170,
     * 197,226,257,290,325,362,401,442,485,530,577,626,
     * 677,730,785,842,901,962,1025,1090,1157,1226,1297,
     * 1370,1445,1522,1601,1682,1765,1850,1937,2026,2117,
     * 2210,2305,2402,2501,2602,2705,2810,2917,3026,3137,
     * 3250/
      MIN = II
      MAX = II
      L(II) = II
      IF (JJ-II) 170, 170, 10
   10 KMIN = K(MIN)
      KMAX = KMIN
      IA = II + 1
      IRT = 1
      ITAB = TAB(1) + II - 1
      ISTRT = II
      DO 160 J=IA,JJ
        IF (J-ITAB) 30, 20, 20
   20   IRT = IRT + 1
        ISTRT = ISTRT + 1
        ITAB = TAB(IRT) + II - 1
   30   KJ + K(J)
        IF (KJ-KMAX) 40, 60, 60
   40   IF (KJ-KMIN) 70, 50, 90
```

```
 50   MADIN = MIN
      MIN = J
      GO TO 130
 60   I = MAX
      MAX = J
      KMAX = K(J)
      GO TO 80
 70   I = MAX
      MIN = J
      KMIN = K(J)
 80   IPOI = L(I)
      L(I) = J
      L(J) = IPOI
      GO TO 160
 90   MADIN = MIN
      IPOI = J - 1
      I = KJ - KMIN
      DO 120 KEY=ISTRT,IPOI,IRT
        KDIFF = KJ - K(KEY)
        IF (KDIFF) 120, 140, 100
100     IF (I-KDIFF) 120, 120, 110
110     I = KDIFF
        MADIN = KEY
120   CONTINUE
130   IPOI = MADIN
      MADIN = L(IPOI)
      IF (KJ-K(MADIN)) 150, 130, 130
140   MADIN = KEY
      GO TO 130
150   L(IPOI) = J
      L(J) = MADIN
160   CONTINUE
170   MIN = L(MAX)
      L(MAX) = 0
      RETURN
      END
```

ALGORITHM 509
A HYBRID PROFILE REDUCTION ALGORITHM [F1]

Norman E. Gibbs
College of William and Mary

```
      SUBROUTINE PROFIT(NR, NDSTK, NEW, NDEG, LVLS2, LVLST,
     * LSTPT, NXTNUM)

      INTEGER NDSTK
      INTEGER NEW(1), NDEG(1), LVLS2(1), LVLST(1), LSTPT(1)
      DIMENSION NDSTK(NR,1)
      COMMON /GRA/ N, IDOTH, IDEG
      COMMON /LVLW/ S2(1ØØ), S3(1ØØ), Q(1ØØ)
      COMMON /CC/ CONECT(1ØØ)
      INTEGER S2, S3, Q, CONECT, S2SZE, S3SZE, QPTR, CONSZE
      NSTPT = 1
      DO 2Ø I=1,IDPTH
        LSTPT(I) = NSTPT
        DO 1Ø J=1,N
          IF (LVLS2(J).NE.I) GO TO 1Ø
          LVLST(NSTPT) = J
          NSTPT = NSPTP + 1
   1Ø   CONTINUE
   2Ø CONTINUE
      LSTPT(IDPTH+1) = NSTPT
      LEVEL = 1
      CALL FORMLV(S2, S2SZE, LSTPT, LVLST, LEVEL)
   3Ø CALL FORMLV(S3, S3SZE, LSTPT, LVLST, LEVEL+1)
   4Ø M = MINCON(S2,S2SZE,S3,S3SZE,CONECT,CONSZE,NDSTK,NR,
     * NDEG
      NEW(M) = NXTNUM
      NXTNUM = NXTNUM + 1
      CALL DELETE(S2, S2SZE, M)
      IF (CONSZE.LE.Ø) GO TO 6Ø
      DO 5Ø I=1,CONSZE
        QPTR = QPTR + 1
        Q(QPTR) = CONECT(I)
        CALL DELETE(S3, S3SZE, CONECT(I))
   5Ø CONTINUE
   6Ø IF (S2SZE.LE.Ø) GO TO 8Ø
      IF (S3SZE.GT.Ø) GO TO 4Ø
      DO 7Ø I=1,S2SZE
        NS2 = S2(I)
        NEW(NS2) = NXTNUM
        NXTNUM = NXTNUM + 1
   7Ø CONTINUE
```

```
      GO TO 1ØØ
  8Ø  IF (S3SZE.LE.Ø) GO TO 1ØØ
      DO 9Ø I=1,S2SZE
        QPTR = QPTR + 1
        Q(QPTR) = S2(I)
  9Ø  CONTINUE
 1ØØ  LEVEL = LEVEL + 1
      IF (LEVEL.GE.IDPTH) GO TO 12Ø
      DO 11Ø I=1,QPTR
        S2(I) = Q(I)
 11Ø  CONTINUE
      S2SZE = QPTR
      GO TO 3Ø
 12Ø  DO 13Ø I=1,QPTR
        IQ = Q(I)
        NEW(IQ) = NXTNUM
        NXTNUM = NXTNUM + 1
 13Ø  CONTINUE
      RETURN
      END


      FUNCTION MINCON(X, XSZE, Y, YSZE, CONLST, CONSZE,
     * NDSTK, NR, NDEG)

      INTEGER NDSTK
      DIMENSION NDSTK(NR,1)
      INTEGER X(1), XSZE, Y(1), YSZE, CONLST(1), CONSZE,
     * NDEG(1)
      INTEGER SMLST(1ØØ)
      CONSZE = YSZE + 1
      DO 5Ø I=1,XSZE
        LSTSZE = Ø
        IX = X(I)
        IROWDG = NDEG(IX)
        DO 2Ø J=1,YSZE
          DO 1Ø K=1,IROWDG
            IX = X(I)
            IF (NDSTK(IX,K).NE.Y(J)) GO TO 1Ø
            SMLST(LSTSZE+1) = Y(J)
            LSTSZE = LSTSZE + 1
            IF (LSTSZE.GE.CONSZE) GO TO 5Ø
            GO TO 2Ø
  1Ø      CONTINUE
  2Ø  CONTINUE
      IF (LSTSZE.GT.Ø) GO TO 3Ø
      MINCON = X(I)
      CONSZE = Ø
      RETURN
```

```
   3Ø CONSZE = LSTSZE
      DO 4Ø J=1,LSTSZE
        CONLST(J) = SMLST(J)
   4Ø CONTINUE
      MINCON = X(I)
   5Ø CONTINUE
      RETURN
      END


      SUBROUTINE DELETE(SET, SETSZE, ELEMNT)

      INTEGER SET(1), SETSZE, ELEMNT
      IF (SETSZE.GT.1) GO TO 1Ø
      IF (SETSZE.EQ.1 .AND. SET(1).NE.ELEMNT) GO TO 3Ø
      SETSZE = Ø
      RETURN
   1Ø DO 2Ø I=1,SETSZE
        IF (SET(I).EQ.ELEMNT) GO TO 4Ø
   2Ø CONTINUE
   3Ø WRITE (6,99999) ELEMNT, (SET(I),I=1,SETSZE)
      RETURN
   4Ø SETSZE = SETSZE - 1
      DO 5Ø J=I,SETSZE
        SET(J) = SET(J+1)
   5Ø CONTINUE
      RETURN
99999 FORMAT (1ØHØERROR -- , I6, 8H NOT IN , (2ØI5))
      END


      SUBROUTINE FORMLV(SET, SETSZE, LSTPT, LVLST, LEVEL)

      INTEGER SET(1), SETSZE, LSPTP(1), LVLST(1), UPPER
      LOWER = LSTPT(LEVEL)
      UPPER = LSTPT(LEVEL+1) - 1
      SETSZE = 1
      DO 1Ø I=LOWER,UPPER
        SET(SETSZE) = LVLST(I)
        SETSZE = SETSZE + 1
   1Ø CONTINUE
      SETSZE = SETSZE - 1
      RETURN
      END
```

ALGORITHM 512
A NORMALIZED ALGORITHM FOR THE SOLUTION OF POSITIVE DEFINITE SYMMETRIC QUINDIAGONAL SYSTEMS OF LINEAR EQUATIONS [F4]

A. Benson and D. J. Evans
Loughborough University of Technology, England

```
      SUBROUTINE FACTOR(A, B, C, G, H, N)

      DIMENSION A(N), B(N), C(N), G(N), H(N)
      N1 = N - 1
      N2 = N - 2
      N3 = N - 3
      N4 = N - 4
      C(1) = SQRT(C(1))
      V = B(1)/C(1)
      C(2) = SQRT(C(2)-V*V)
      B(1) = V/C(2)
      DO 1Ø I=3,N2
        I1 = I - 1
        I2 = I - 2
        U = A(I2)/C(I2)
        V = B(I1)/C(I1) - B(I2)*U
        C(I) = SQRT(C(I)-U*U-V*V)
        A(I2) = U/C(I)
        B(I1) = V/C(I)
   1Ø CONTINUE
      G(1) = A(N1)/C(1)
      G(2) = -B(1)*G(1)
      U = A(N3)/C(N3)
      V = B(N2)/C(N2) - B(N3)*U
      DO 2Ø I=3,N2
        G(I) = -B(I-1)*G(I-1) - A(I-2)*G(I-2)
   2Ø CONTINUE
      W = Ø.Ø
      DO 3Ø I=1,N4
        W = W + G(I)*G(I)
   3Ø CONTINUE
      C(N1) = SQRT(C(N1)-W-(G(N3)+U)**2-(G(N2)+V)**2)
      W = 1.Ø/C(N1)
      A(N3) = U*W
      B(N2) = V*W
      DO 4Ø I=1,N2
        G(I) = G(I)*W
   4Ø CONTINUE
```

```
      H(1) = B(N)/C(1)
      H(2) = A(N)/C(2) - B(1)*H(1)
      U = A(N2)/C(N2)
      V = B(N1)/C(N1) - B(N2)*U
      DO 5Ø I=3,N1
        H(I) = -B(I-1)*H(I-1) - A(I-2)*H(I-2)
   5Ø CONTINUE
      W = Ø.Ø
      DO 6Ø I=1,N2
        W = W + G(I)*H(I)
   6Ø CONTINUE
      H(N1) = H(N1) - W - G(N2)*U
      W = Ø.Ø
      DO 7Ø I=1,N3
        W = W + H(I)*H(I)
   7Ø CONTINUE
      C(N) = SQRT(C(N)-W-(H(N2)+U)**2-(H(N1)+V)**2)
      W = 1.Ø/C(N)
      A(N2) = U*W
      B(N1) = V*W
      DO 8Ø I=1,N1
        H(I) = H(I)*W
   8Ø CONTINUE
      RETURN
      END


      SUBROUTINE SOLVE(E, F, G, H, Q, N)

      DIMENSION E(N), F(N), G(N), H(N), Q(N)
      N1 = N - 1
      N2 = N - 2
      N3 = N - 3
      Q(2) = Q(2) - E(1)*Q(1)
      DO 1Ø I=3,N2
        Q(I) = Q(I) - E(I-1)*Q(I-1) - F(I-2)*Q(I-2)
   1Ø CONTINUE
      U = Ø.Ø
      V = Ø.Ø
      DO 2Ø I=1,N2
        U = U + G(I)*Q(I)
        V = V + H(I)*Q(I)
   2Ø CONTINUE
      Q(N1) = Q(N1) - E(N2)*Q(N2) - F(N3)*Q(N3) - U
      Q(N) = Q(N) - (E(N1)+H(N1))*Q(N1) - F(N2)*Q(N2) - V
      Q(N1) = Q(N1) - (E(N1)+H(N1))*Q(N)
      Q(N2) = Q(N2) - (E(N2)+G(N2))*Q(N1) - (F(N2)+H(N2))
     * *Q(N)
      DO 3Ø II=1,N3
        I = N3 - II + 1
      Q(I) = Q(I) - E(I)*Q(I+1) - F(I)*Q(I+2) -
   *    G(I)*Q(N1) - H(I)*Q(N)
30 CONTINUE
   RETURN
   END


   SUBROUTINE RHS(D, S, N)

   DIMENSION D(N), S(N)
   DO 10 I=1,N
     S(I) = S(I)/D(I)
10 CONTINUE
   RETURN
   END
```

ALGORITHM 513
ANALYSIS OF IN-SITU TRANSPOSITION [F1]

Esko G. Cate and David W. Twigg
Boeing Computer Services, Inc.

```
      SUBROUTINE TRANS(A, M, N, MN, MOVE, IWRK, IOK)

      DIMENSION A(MN), MOVE(IWRK)
      IF (M.LT.2 .OR. N.LT.2) GO TO 12Ø
      IF (MN.NE.M*N) GO TO 18Ø
      IF (IWRK.LT.1) GO TO 19Ø
      IF (M.EQ.N) GO TO 13Ø
      NCOUNT = 2
      K = MN - 1
      DO 1Ø I=1,IWRK
        MOVE(I) = Ø
   1Ø CONTINUE
      IF (M.LT.3 .OR. N.LT.3) GO TO 3Ø
      IR2 = M - 1
      IR1 = N - 1
   2Ø IRØ = MOD(IR2,IR1)
      IR2 = IR1
      IR1 = IRØ
      IF (IRØ.NE.Ø) GO TO 2Ø
      NCOUNT = NCOUNT + IR2 - 1
   3Ø I = 1
      IM = M
      GO TO 8Ø
   4Ø MAX = K - I
      I = I + 1
      IF (.GT.MAX) GO TO 16Ø
      IM = IM + M
      IF (IM.GT.K) IM = IM - K
      I2 = IM
      IF (I.EQ.I2) GO TO 4Ø
      IF (I.GT.IWRK) GO TO 6Ø
      IF (MOVE(I).EQ.Ø) GO TO 8Ø
      GO TO 4Ø
   5Ø I2 = M*I1 - K*(I1/N)
   6Ø IF (I2.LE.I .OR. I2.GE.MAX) GO TO 7Ø
      I1 = I2
      GO TO 5Ø
   7Ø IF (I2.NE.I) GO TO 4Ø
   8Ø I1 = I
      KMI = K - 1
      B = A(I1+1)
      I1C = KMI
      C = A(I1C+1)
  9Ø  I2 = M*I1 - K*(I1/N)
      I2C = K - I2
      IF (I1.LE.IWRK) MOVE(I1) = 2
      IF (I1C.LE.IWRK) MOVE(I1C) = 2
      NCOUNT = NCOUNT + 2
      IF (I2.EQ.I) GO TO 11Ø
      IF (I2.EQ.KMI) GO TO 1ØØ
      A(I1+1) = A(I2+1)
      A(I1C+1) = A(I2C+1)
      I1 = I2
      I1C = I2C
      GO TO 9Ø
 1ØØ  D = B
      B = C
      C = D
 11Ø  A(I1+1) = B
      A(I1C+1) = C
      IF (NCOUNT.LT.MN) GO TO 4Ø
 12Ø  IOK = Ø
      RETURN
 13Ø  N1 = N - 1
      DO 15Ø I=1,N1
        J1 = I + 1
        DO 14Ø J=J1,N
          I1 = I + (J-1)*N
          I2 = J + (I-1)*M
          B = A(I1)
          A(I1) = A(I2)
          A(I2) = B
 14Ø  CONTINUE
 15Ø  CONTINUE
      GO TO 12Ø
 16Ø  IOK = I
 17Ø  RETURN
 18Ø  IOK = -1
      GO TO 17Ø
 19Ø  IOK = -2
      GO TO 17Ø
      END
```

ALGORITHM 515
GENERATION OF A VECTOR FROM THE LEXICOGRAPHICAL INDEX [G6]

B. P. Buckles and M. Lybanon
Computer Sciences Corporation

```
      SUBROUTINE COMB(N, P, L, C)

      INTEGER N, P, L, C(P), K, R, P1, BINOM
      K = Ø
      P1 = P - 1
      DO 2Ø I=1,P1
        C(I) = Ø
        IF (I.NE.1) C(I) = C(I-1)
1Ø      C(I) = C(I) + 1
        R = BINOM(N-C(I),P-I)
        K = K + R
        IF (K.LT.L) GO TO 1Ø
        K = K - R
2Ø    CONTINUE
      C(P) = C(P1) + L - K
      RETURN
      END
      INTEGER FUNCTION BINOM(M, N)
      INTEGER M, N, P, I, N1, R
      N1 = N
      P = M - N1
      IF (N1.GE.P) GO TO 1Ø
      P = N1
      N1 = M - P
1Ø    R = N1 + 1
      IF (P.EQ.Ø) R = 1
      IF (P.LT.2) GO TO 3Ø
      DO 2Ø I=2,P
        R = (R*(N1+I))/I
2Ø    CONTINUE
3Ø    BINOM = R
      RETURN
      END
```

ALGORITHM 518
INCOMPLETE BESSEL FUNCTION I_0: THE VON MISES DISTRIBUTION [S14]

Geoffrey W. Hill
Centre de Morphologie Mathematique,
Fontainebleau, France

```
      FUNCTION VMISES(T, VK)

      DATA A1, A2, A3, A4, CK, C1 /12.0,0.8,8.0,1.0,10.5,
     * 56.0/
      Z = VK
      U = AMOD(T+PI,TPI)
      IF (U.LT.0.0) U = U + TPI
      Y = U - PI
      IF (Z.GT.CD) GO TO 30
      V = 0.0
      IF (Z.LE.0.0) GO TO 20
      IP = Z*A2 - A3/(Z+A4) + A1
      P = FLOAT(IP)
      S = SIN(Y)
      C = COS(Y)
      Y = P*Y
      SN = SIN(Y)
      CN = COS(Y)
      R = 0.0
      Z = 2.0/Z
      DO 10 N=2,IP
        P = P - 1.0
        Y = SN
        SN = SN*C - CN*S
        CN = CN*C + Y*S
        R = 1.0/(P*Z+R)
        V = (SN/P+V)*R
   10 CONTINUE
   20 VMISES = (U*0.5+V)/PI
      GO TO 40
   30 C = 24.0*Z
      V = C - C1
      R = SQRT((54.0/(347.0/V+26.0-C)-6.0+C)/6.0)
      R=SQRT((54.0/(347.0/V+26.0-C)-6.0+C)/12.0)
      Z = SIN(Y*0.5)*R
      S = Z*Z
      S=Z*Z*2.0
      V = V - S + 3.0
      Y = (C-S-S-16.0)/3.0
```

```
    Y = ((S+1.75)*S+83.5)/V - Y
    VMISES = GAUSS(Z-S/(Y*Y)*Z
    VMISES=ERF(Z-S/(Y*Y)*Z)*Ø.5+Ø.5
4Ø  IF (VMISES.LT.Ø.Ø) VMISES = Ø.Ø
    IF (VMISES.GT.1.Ø) VMISES = 1.Ø
    RETURN
    END
```

ALGORITHM 521
REPEATED INTEGRALS OF THE COERROR FUNCTION [S15]

Walter Gautschi
Purdue University

```
      SUBROUTINE INERFC(X, NMAX, ACC, FZERO, F, IFLAG)

      DIMENSION F(NMAX), RØ(5ØØ), R1 (5ØØ)
      DOUBLE PRECISION DEPS, DC, DX, DXSQ, DFM1, DFØ, DF1,
     * DN, DN1, DTE, DTERM, DSUM
      LOGICAL FRSTTM
      DATA FRSTTM, PREC, BOTEXP /.TRUE.,28.8989,-293./
      IFLAG = Ø
      IF (NMAX.LT.1) GO TO 23Ø
      IF (NMAX.GT.5ØØ) GO TO 24Ø
      TOL = PREC - ACC
      IF (TOL.LT.1) GO TO 25Ø
      I = 1
      EPS = .5*1Ø.**(-ACC)
      XSQ = X*X
      IF (.NOT.FRSTTM) GO TO 1Ø
      P = ATAN(1.)
      AL1Ø = ALOG(1Ø)
      C = SQRT(1./P)
      DC = DSQRT(1.DØ/DATAN(1.DØ))
      FRSTM = .FALSE.
   1Ø S = Ø
      IF (-XSQ.GT.AL1Ø*BOTEXP) S = EXP(-XSQ)
      B = .5*AL1Ø*TOL + .25*ALOG(2.*P)
      B1 = .5*AL1Ø*(TOL-1.)
      IF (XSQ.GT.B) GO TO 9Ø
      IF (X.LT.Ø.) GO TO 2Ø
      FØ = C
      IF (S.EQ.Ø.) GO TO 26Ø
      T = 1./S
      SQ = SQRT(2.*FLOAT(NMAX))
      IF ((X.GE.1.12 .AND. XSQ+SQ*X.GT.B) .OR. (X.LT.1.12
     * .AND. SQ*X.GT.B1)) GO TO 1ØØ
      GO TO 3Ø
   2Ø FØ = C*S
      T = 1.
   3Ø DEPS = .5DØ*1Ø.DØ**(-PREC)
      DX = DBLE(X)
      DXSQ = DX*DX
      DTE = DC*DX
```

```
    DSUM = 1.DØ - DTE
    DN = Ø.DØ
    DN1 = 1.DØ
 4Ø DN = DN + 1.DØ
    DN1 = DN1 + 2.DØ
    IF (DN.GT.2.D2) GO TO 27Ø
    DTE = -DTE*DXSQ/DN
    DTERM = DTE/DN1
    DSUM = DSUM - DTERM
    IF (DABS(DTERM).GT.DEPS*DABS(DSUM)) GO TO 4Ø
    IF (X.LT.Ø) GO TO 6Ø
    DFØ = DC
    DF1 = DEXP(DXSQ)*DSUM
    FZERO = SNGL(DF1)
    DO 5Ø N=1,NMAX
      DFM1 = DFØ
      DFØ = DF1
      DF1 = (-DX*DFØ+.5DØ*DFM1)/DBLE(FLOAT(N))
      F(N) = SNGL(DF1)
 5Ø CONTINUE
    RETURN
 6Ø F1 = SNGL(DSUM)
 7Ø FZERO = F1
    DO 8Ø N=1,NMAX
      FM1 = FØ
      FØ = F1
      F1 = (-X*FØ+.5*FM1)/FLOAT(N)
      F(N) = F1
 8Ø CONTINUE
    RETURN
 9Ø IF (X.LT.Ø.) GO TO 11Ø
1ØØ NØ = NMAX
    XØ = X
    GO TO 13Ø
11Ø IF (XSQ.LT.(ACC+1.)*AL1Ø-.572) GO TO 12Ø
    FØ = C*S
    F1 = 2.
    GO TO 7Ø
12Ø I = 2
    NØ = 1
    XØ = -X
13Ø DO 14Ø N=1,NØ
      R1(N) = Ø.
14Ø CONTINUE
    FNØ = FLOAT(NØ)
    NU = IFIX((SQRT(FNØ)+(2.3Ø26*ACC+1.3863)/(2.828*XØ))
   * **2-5.)
15Ø NU = NU + 1Ø
    IF (NU.GT.5ØØØ) GO TO 28Ø
    NØP1 = NØ + 1
    NUM = NU - NØP1
```

```
      DO 16Ø N=1,NØ
        RØ(N) = R1(N)
  16Ø CONTINUE
      R = Ø
      DO 17Ø K=1,NUM
        N = NU - K
        R = .5/(XØ+FLOAT(N+1)*R)
  17Ø CONTINUE
      DO 18Ø K=1,NØ
        N = NØP1 - K
        R = .5.(XØ+FLOAT(N+1)*R)
        R1(N) = R
  18Ø CONTINUE
      DO 19Ø N=1,NØ
        IF (ABS(R1(N)-RØ(N)).GT.EPS*ABS(R1(N))) GO TO 15Ø
  19Ø CONTINUE
      FØ = .5*C/(XØ+R1(1))
      FZERO = FØ
      FF = FØ
      DO 2ØØ N=1,NØ
        FF = R1(N)*FF
        F(N) = FF
  2ØØ CONTINUE
      GO TO (21Ø, 22Ø), I
  21Ø RETURN
  22Ø F1 = 2. - S*FØ
      FØ = C*S
      GO TO 7Ø
  23Ø IFLAG = 1
      RETURN
  24Ø IFLAG = 2
      RETURN
  25Ø IFLAG = 3
      RETURN
  26Ø IFLAG = 4
      RETURN
  27Ø IFLAG = 5
      RETURN
  28Ø IFLAG = 6
      RETURN
      END
```

ALGORITHM 523
CONVEX, A NEW CONVEX HULL ALGORITHM FOR PLANAR SETS [Z]

William F. Eddy
Carnegie-Mellon University

```
      COMMON NCOUNT
      DIMENSION XX(2,25),IN(25),IH(25)
      DIMENSION X(25),Y(25)
      INTEGER IWORK(5Ø)
      INTEGER IL(5Ø)
      DATA IWORK/5Ø*Ø/
      NCOUNT=Ø
      READ(5,1)N
1     FORMAT(I5)
      WRITE(6,1)N
      N1=N+1
      DO 2 I=1,N
      J=N1-I
2     IN(J)=I
      DO 3 I=1,N
3     READ(5,4)XX(1,I),XX(2,I)
4     FORMAT(2F1Ø.5)
      DO 5 I=1,N
      J=IN(I)
5     WRITE(6,4)XX(1,J),XX(2,J)
      DO 1Ø M=4,N
      CALL CONVEX(N,XX,M,IN,IWORK,IWORK(N+1),IH,NHULL,IL)
      IK=IL(1)
      DO 6 I=1,NHULL
      J=IH(IK)
      X(I)=XX(1,J)
      Y(I)=XX(2,J)
6     IK=IL(IK)
      WRITE(6,7)M,NHULL,NCOUNT
7     FORMAT(12HØSAMPLE SIZE , I5,9H VERTICES .I5,6H SPLIT
     * ,I5)
      DO 8 I=1,NHULL
8     WRITE96,9)X(I),Y(I)
9     FORMAT(1X,2F1Ø.5)
1Ø    CONTINUE
      STOP
      END
      SUBROUTINE SPLIT(N,X,M,IN,II,JJ,S,IABV,NA,MAXA,IBEL,
     1  NB,MAXB)

      DIMENSION X(2,N)
      DIMENSION IN(M),IABV(M),IBEL(M)
      INTEGER S
      LOGICAL T
      T=.FALSE.
      IF(X(1,JJ).NE.X(1,II))GOTO 1
      XT=X(1,II)
      DIR=SIGN(1.,X(2,JJ)-X(2,II))*SIGN(1.FLOAT(S))
      T=.TRUE.
      GOTO 2
1     A=(X(2,JJ)-X(2,II))/(X(1,JJ)-X(1,II))
      B=X(2,II)-A*X(1,II)
2     UP=Ø.
      NA=Ø
      MAXA=Ø
      DOWN=Ø.
      NB=Ø
      MAXB=Ø
      DO 6 I=1,M
        IS=IN(I)
      IF(T)GOTO 3
      Z=X(2,IS)-A*X(1,IS)-B
      GOTO 4
3     Z=DIR*(X(1,IS)-XT)
4     IF(Z.LE.Ø.)GOTO 5
      IF(S.EQ.-2)GOTO 6
      NA=NA+1
      IABV(NA)=IS
      IF(Z.LT.UP)GOTO 6
      UP=Z
      MAXA=NA
      GOTO 6
5     IF(S.EQ.2)GOTO 6
      IF(Z.GE.Ø.)GOTO 6
      NB=NB+1
      IBEL(NB)=IS
      IF(Z.GT.DOWN)GOTO 6
      DOWN=Z
      MAXB=NB
6     CONTINUE
      RETURN
      END


      SUBROUTINE CONVEX(N,X,M,IN,IA,IB,IH,NH,IL)

      DIMENSION X(2,N)
      DIMENSION IN(M),IA(M),IB(M),IH(M),IL(M)
      LOGICAL MAXE,MINE
      IF(M.EQ.1)GOTO 22
      IL(1)=2
      IL(2)=1
      KN=IN(1)
      KX=IN(2)
      IF(M.EQ.2)GOTO 21
      MP1=M+1
      MIN=1
      MX=1
      KX=IN(1)
      MAXE=.FALSE.
      MINE=.FALSE.
      DO 6 I=2,M
        J=IN(I)
        IF(X(1,J)-X(1,KX))3,1,2
1       MAXE=.TRUE.
        GOTO 3
2       MAXE=.FALSE.
        MX=I
        KX=J
3       IF(X(1,J)-X(1,KN))5,4,6
4       MINE=.TRUE.
        GOTO 6
5       MINE=.FALSE.
        MIN=I
        KN=J
6     CONTINUE
      IF(KX.EQ.KN)GOTO 18
      IF(MAXE.OR.MINE) GOTO 23
7     IH(1)=KX
      IH(2)=KN
      NH=3
      INH=1
      NIB=1
      MA=M
      IN(MX)=IN(M)
      IN(M)=KX
      MM=M-2
      IF(MIN.EQ.M)MIN=MX
      IN(MIN)=IN(M-1)
      IN(M-1)=KN
      CALL SPLIT(N,X,MM,IN,IH(1),IH(2),Ø,IA,MB,MXA,IB,
     1  IA(MA),MXBB)
8     NIB=NIB+IA(MA)
      MA=MA-1
9     IF(MXA.EQ.Ø)GOTO 11
      IL(NH)=IL(INH)
      IL(INH)=NH
      IH(NH)=IA(MXA)
```

```
       IA(MXA)=IA(MB)
       MB=MB-1
       NH=NH+1
       IF(MB.EQ.Ø)GOTO 1Ø
       ILINH=IL(INH)
       CALL SPLIT(N,X,MB,IA,IH((NH),IH(ILINH),1,IA,MBB,MXA,
      1  IB(NIB),IA(MA),MXB)
       MB=MBB
       GOTO 8
 1Ø    INH=IL(INH)
 11    INH=IL(INH)
       MA=MA+1
       NIB=NIG-IA(MA)
       IF(MA.GE.M)GOTO 12
       IF(IA(MA).EQ.Ø)GOTO 11
       ILINH=IL(INH)
       CALL SPLIT(N,X,IA(MA),IB(NIB),IH(INH),IH(ILINH),2,IA,
      1  MB,MXA,IB(NIB),MBB,MXB)
       IA(MA)=MBB
       GOTO 9
 12    MXB=MXBB
       MA=M
       MB=(IA)MA
       NIA=1
       IA(MA)=Ø
 13    NIA=NIA+IA(MA)
       MA=MA-1
 14    IF(MXB.EQ.Ø)GOTO 16
       IL(NH)=IL(INH)
       IL(INH)=NH
       IH(NH)=IB(MXB)
       IB(MXB)-IB(MB)
       MB=MB-1
       NH=NH+1
       IF(MB.EQ.Ø)GOTO 15
       ILINH=IL(INH)
       CALL SPLIT(N,X,MB,IB(NIB),IH(INH),IH(ILINH),-1,
      1  IA(NIA),IA(MA),MXA,IB(NIB),MBB,MXB)
       MB=MBB
       GOTO 13
 15    INH=IL(INH)
 16    INH=IL(INH)
       MA=MA+1
       NIA=NIA-IA(MA)
       IF(MA.EQ.MP1)GOTO 17
       IF(IA(MA).EQ.Ø)GOTO 16
       ILINH=IL(INH)
       CALL SPLIT(N,X,IA(MA),IA(NIA),IH(INH),IH(ILINH),-2,
      1  IA(NIA),MBB,MXA,IB(NIB),MB,MXB)
       GOTO 14
```

```
17      NH=NH-1
        RETURN
18      KX=IN(1)
        KN=IN(1)
        DO 2Ø I=1,M
          J=IN(I)
          IF(X(2,J).LE.X(2,KX))GOTO 19
          MX=I
          KX=J
19        IF(X(2,J).GE.X(2,KN))GOTO 2Ø
          MIN=I
          KN=J
2Ø      CONTINUE
        IF(KX.EQ.KN)GOTO 22
21      IH(1)=KX
        IH(2)=KN
        NH=3
        IF((X(1,KN).EQ.X(1,KX)).AND.(X(2,KN).EQ.X(2,KX)))NH=2
        GOTO 17
22      NH=2
        IH(1)=IN(1)
        IL(1)=1
        GOTO 17
23      IF(.NOT.MAXE)GOTO 25
        DO 24 I=1,M
          J=IN(I)
          IF(X(1,J).NE.X(1,KX))GOTO 24
          IF(X(2,J).LE.X(2,KX))GOTO 24
          MX=I
          KX=J
24      CONTINUE
25      IF(.NOT.MINE)GOTO 7
        DO 26 I=1,M
          J=IN(I)
          IF(X(1,J).NE.X(1,KN))GOTO 26
          IF(X(2,J).GE.X(2,KN))GOTO 26
          MIN=I
          KN=J
26      CONTINUE
        GOTO 7
        END
```

ALGORITHM 529
PERMUTATIONS TO BLOCK TRIANGULAR FORM [F1]

I. S. Duff and J. K. Reid
AERE Harwell

```
      SUBROUTINE MC13D(N, INC, LICN, IP, LENR, IOR, IB,
     * NUM, IW)

      INTEGER IP(N)
      INTEGER ICN(LICN), LENR(N), IOR(N), IB(N), IW(N,3)
      CALL MC13E(N, ICN, LICN, IP, LENR, IOR, IB, NUM,
     * IW(1,1), IW(1,2), IW(1,3)
      RETURN
      END


      SUBROUTINE MC13E(N, ICN, LICN, IP, LENR, ARP, IB,
     * NUM, LOWL, NUMB, PREV)
      INTEGER STP, DUMMY
      INTEGER IP(N)
      INTEGER ICN(LICN), LENR(N), ARP(N), IB(N), LOWL(N),
     * NUMB(N), PREV(N)
      ICNT = Ø
      NUM = Ø
      NNM1 = N + N - 1
      DO 1Ø J=1,N
        NUMB(J) = Ø
        ARP(J) = LENR(J) - 1
   1Ø CONTINUE
      DO 9Ø ISN=1,N
        IF (NUMB(ISN).NE.Ø) GO TO 9Ø
        IV = ISN
        IST = 1
        LOWL(IV) = 1
        NUMB(IV) = 1
        IB(N) = IV
        DO 8Ø DUMMY=1,NNM1
          I1 = ARP(IV)
          IF (I1.LT.Ø) GO TO 3Ø
          I2 = IP(IV) + LENR(IV) - 1
          I1 = I2 - I1
          DO 2Ø II=I1,I2
            IW = 1CN(II)
            IF (NUMB(IW).EQ.Ø) GO TO 7Ø
            IF (LOWL(IW).LT.LOWL(IV)) LOWL(IV) = LOWL(IW)
   2Ø     CONTINUE
```

```
          ARP(IV) = -1
 3Ø       IF (LOWL(IV).LT.NUMB(IV)) GO TO 6Ø
          NUM = NUM + 1
          IST1 = N + 1 - IST
          LCNT = ICNT + 1
          DO 4Ø STP=IST1,N
            IW = IB(STP)
            LOWL(IW) = N + 1
            ICNT = ICNT + 1
            NUMB(IW) = ICNT
            IF (IW.EQ.IV) GO TO 5Ø
 4Ø       CONTINUE
 5Ø       IST = N - STP
          IB(NUM) = LCNT
          IF (1ST.NE.Ø) GO TO 6Ø
          IF (1CNT.LT.N) GO TO 9Ø
          GO TO 1ØØ
 6Ø       IW = IV
          IV = PREV(IV)
          IF (LOWL(IW).LT.LOWL(IV)) LOWL(IV) = LOWL(IW)
          GO TO 8Ø
 7Ø       ARP(IV) = I2 - II - 1
          PREV(IW) = IV
          IV = IW
          IST = IST + 1
          LOWL(IV) = IST
          NUMB(IV) = IST
          K = N + 1 - IST
          IB(K) = IV
 8Ø     CONTINUE
 9Ø   CONTINUE
1ØØ   DO 11Ø I=1,N
        II = NUMB(I)
        ARP(II) = I
11Ø   CONTINUE
      RETURN
      END

      INTEGER IP(5Ø), ICN(1ØØØ), IOR(5Ø), IB(51), IW(150),
     * LENR(5Ø)
      INTEGER BLANK, EX, HOLD(1ØØ)
      LOGICAL A(5Ø,5Ø)
      DATA BLANK, EX, NOT /1H, 1HX,1HØ/
      MM = 5Ø
      LICN = 1ØØØ
 1Ø   READ (5,99999) N, IPP
      IF (N.EQ.Ø) GO TO 1ØØ
      WRITE (6,99998) N, IPP
      DO 3Ø J=1,N
        DO 2Ø I=1,N
```

```
          A(I,J) = .FALSE.
   2Ø   CONTINUE
        A(J,J) = .TRUE.
   3Ø CONTINUE
      IF (IPP.EQ.Ø) GO TO 6Ø
      DO 5Ø K9=1,IPP
   4Ø   CALL FAØ1BS(N, I)
        CALL FAØ1BS(N, J)
        IF (A(I,J)) GO TO 4Ø
        A(I,J) = .TRUE.
   5Ø CONTINUE
   6Ø CALL SETUP(N, A, MM, IP, ICN, LICN, LENR)
      CALL MC13D(N, ICN, LICN, IP, LENR, IOR, IB, NUM, IW)
      IF (NUM.EQ.1) WRITE (6,99997) NUM
      IF (NUM.NE.1) WRITE (6,99996) NUM
      IB(NUM+1) = N + 1
      INDEX = 1ØØ
      IBLOCK = 1
      DO 9Ø I=1,N
        DO 7Ø IJ=1,INDEX
          HOLD(IJ) = BLANK
   7Ø   CONTINUE
        IF (I.EQ.IB(IBLOCK)) WRITE (6,99995)
        IF (I.EQ.IB(IBLOCK)) IBLOCK = IBLOCK + 1
        JBLOCK = 1
        INDEX = Ø
        DO 8Ø J=1,N
          IF (J.EQ.IB(JBLOCK)) INDEX = INDEX + 1
          IF (J.EQ.IB(JBLOCK)) HOLD(INDEX) = BLANK
          IF (J.EQ.IB(JBLOCK)) JBLOCK = JBLOCK + 1
          INDEX = INDEX + 1
          II = IOR(I)
          JJ = IOR(J)
          IF (A(II,JJ)) HOLD(INDEX) = EX
          IF (.NOT.A(II,JJ)) HOLD(INDEX) = NOT
   8Ø   CONTINUE
        WRITE (6,99994) HOLD
   9Ø CONTINUE
      WRITE (6,99993) (IB(I),I=1,NUM)
      GO TO 1Ø
  1ØØ STOP
99999 FORMAT (2I4)
99998 FORMAT (1H1, 2ØH MATRIX IS OF ORDER , I3, 9H AND HAS ,
    * I3, 23H OFF-DIAGONAL NON-ZEROS)
99997 FORMAT (///31H THE REORDERED MATRIX WHICH HAS, I3,
    * 1ØH BLOCK IS , 11HOF THE FORM/)
99996 FORMAT (///31H THE REORDERED MATRIX WHICH HAS, I3,
    * 1ØH BLOCKS IS, 12H OF THE FORM/)
99995 FORMAT (3X)
99994 FORMAT (1X, 1ØØA1)
```

```
99993 FORMAT (/46H THE STARTING POINT FOR EACH BLOCK IS
     * GIVEN BY// 2Ø(2X, I4))
      END


      SUBROUTINE SETUP(N, A, MM, IP, ICN, LICN, LENR)
      LOGICAL A(MM,MM)
      INTEGER IP(N), ICN(LICN), LENR(N)
      DO 1Ø I=1,N
        LENR(I) = Ø
   1Ø CONTINUE
      IND = 1
      DO 3Ø I=1,N
        IP(I) = IND
        DO 2Ø J=1,N
          IF (.NOT.A(I.J)) GO TO 2Ø
          LENR(I) = LENR(I) + 1
          ICN(IND) = J
          IND = IND + 1
   2Ø   CONTINUE
   3Ø CONTINUE
      RETURN
      END
```

ALGORITHM 533
NSPIV, A FORTRAN SUBROUTINE FOR SPARSE GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING [F4]

Andrew H. Sherman
The University of Texas at Austin

```
      SUBROUTINE NSPIV (N,IA,HA,A,B,MAX,R,C,IC,X,ITEMP,
     * RTEMP, IERR)

      REAL A(1),B(1),X(1),RTEMP(1)
      INTEGER IA(1),JA(1),R(1),C(1),IC(1),ITEMP(1)
      INTEGER IU,JU,U,Y,P
      Y = 1
      U = Y + N
      P = 1
      IU = P + N + 1
      JU = IU + N + 1
      CALL NSPIVI (N,IA,JA,A,B,MAX,R,C,IC,X,RTEMP(Y),
     C             ITEMP(P),ITEMP(IU),ITEMP(JU),RTEMP(U),
     C             IERR)
      RETURN
      END

      SUBROUTINE NSPIVI (N,IA,JA,A,B,MAX,R,C,IC,X,Y,P,IU,
     * JU,U,IERR)

      REAL A(1),B(1),U(1),X(1),Y(1)
      REAL DK,LKI,ONE,XPV,XPVMAX,YK,ZERO
      INTEGER C(1),IA(1),IC(1),IU(1),JA(1),JU(1),P(1),R(1)
      INTEGER CK,PK,PPK,PV,V,VI,VJ,VK
      IF (N .EQ. Ø) GO TO 1ØØ1
      ONE = 1.Ø
      ZERO = Ø.Ø
      DO 1Ø J=1,N
        X(J) = ZERO
 1Ø     CONTINUE
      IU(1) = 1
      JUPTR = Ø
      DO 17Ø K=1,N
        P(N+1) = N+1
        VK = R(K)
        JMIN = IA(VK)
      JMAX = IA(VK+1) - 1
      IF (JMIN .GT. JMAX) GO TO 1ØØ2
      J = JMAX
 2Ø     JAJ = JA(J)
        VJ = IC(JAJ)
        X(VJ) = A(J)
        PPK = N+1
 3Ø     PK = PPK
        PPK = P(PK)
        IF (PPK - VJ)  3Ø,1ØØ3,4Ø
 4Ø     P(VJ) = PPK
        P)PK) = VJ
        J = J - 1
        IF (J .GE. JMIN) GO TO 2Ø
      VI = N+1
      YK = B(VK)
 5Ø   VI = P(VI)
      IF (VI .GE. K) GO TO 11Ø
      LKI = - X(VI)
      X(VI) = ZERO
      YK = YK + LKI * Y(VI)
      PPK = VI
      JMIN = IU(VI)
      JMAX = IU(VI+1) - 1
      IF (JMIN .GT. JMAX) GO TO 5Ø
      DO 1ØØ J=JMIN,JMAX
        JUJ = JU(J)
        VJ = IC(JUJ)
        IF (X(VJ) .NE. ZERO) GO TO 9Ø
        IF (VJ - PPK) 6Ø,9Ø,7Ø
 6Ø     PPK = VI
 7Ø     PK = PPK
        PPK = P(PK)
        IF (PPK - VJ) 7Ø,9Ø,8Ø
 8Ø     P(VJ) = PPK
        P(PK) = VJ
        PPK = VJ
 9Ø     X(VJ) = X(VJ) + LKI * U(J)
1ØØ     CONTINUE
      GO TO 5Ø
11Ø   IF (VI .GT. N) GO TO 1ØØ4
      XPVMAX = ABS(X(VI))
      MAXC = VI
      NZCNT = Ø
      PV = VI
12Ø     V = PV
        PV = P(PV)
        IF (PV .GT. N) GO TO 13Ø
```

```
          NZCNT = NZCNT + 1
          XPV = ABS(X(PV))
          IF (XPV .LE. XPVMAX) GO TO 12Ø
          XPVMAX = XPV
          MAXC = PV
          MAXCL = V
          GO TO 12Ø
13Ø     IF (XPVMAX .EQ. ZERO) GO TO 1ØØ4
        IF (VI .EQ. K) GO TO 14Ø
        IF (VI .EQ. MAXC) GO TO 14Ø
        P(MAXCL) = P(MAXC)
        GO TO 15Ø
14Ø     VI = P(VI)
15Ø     DK = ONE / X(MAXC)
        X(MAXC) = X(K)
        I = C(K)
        C(K) = C(MAXC)
        C(MAXC) = I
        CK = C(K)
        IC(CK) = K
        IC(I) = MAXC
        X(K) = ZERO
        Y(K) = YK * DK
        IU(K+1) = IU(K) + NZCNT
        IF (IU(K+1) .GT. MAX+1) GO TO 1ØØ5
        IF (VI .GT. N) GO TO 17Ø
        J = VI
16Ø       JUPTR = JUPTR + 1
          JU(JUPTR) = C(J)
          U(JUPTR) = X(J) * DK
          X(J) = ZERO
          J = P(J)
          IF (J .LE. N) GO TO 16Ø
17Ø     CONTINUE
      K = N
      DO 2ØØ I=1,N
        YK = Y(K)
        JMIN = IU(K)
        JMAX = IU(K+1) - 1
        IF (JMIN .GT. JMAX) GO TO 19Ø
        DO 18Ø J=JMIN,JMAX
          JUJ = JU(J)
          JUJ = IC(JUJ)
          YK = YK - U(J) 8 Y(JUJ)
18Ø       CONTINUE
19Ø     Y(K) = YK
        CK = C(K)
        X(CK) = YK
        K = X-1
 2ØØ    CONTINUE
      IERR = IU(N+1) - IU(1)
      RETURN
1ØØ1  IERR = Ø
      RETURN
1ØØ2  IERR = -K
      RETURN
1ØØ3  IERR = -(N+K)
      RETURN
1ØØ4  IERR = -(2*N+K)
      RETURN
1ØØ5  IERR = -(3*N+K)
      RETURN
      END

      INTEGER IA(1Ø1),JA(4ØØ),R(1ØØ),C(1ØØ),IC(1ØØ),
     * ITEMP(597)
      REAL A(4ØØ),B(1ØØ),X(1ØØ),RTEMP(495)
      DATA MAX/395/,NG/1Ø/,N/1ØØ/
      K = 1
      IA(1) = 1
      IAPTR = 1
      DO 5 I=1,NG
        DO 5 J=1,NG
          BK = Ø.
          IF (I. EQ. 1) GO TO 1
          JA(IAPTR) = K- NG
          A(IAPTR) = -1.
          BK = BK - 1.
          IAPTR = IAPTR + 1
  1       IF (J. EQ. 1) GO TO 2
          JA(IAPTR) = K - 1
          A(IAPTR) = -1.
          BK = BK -1.
          IAPTR = IAPTR + 1
  2       JA(IAPTR) = K
          A(IAPTR) = 4.
          BK = BK + 4.
          IAPTR = IAPTR + 1
          IF (I .EQ. NG) GO TO 4
          JA(IAPTR) = K + NG
          A(IAPTR) = -1.5
          BK = BK - 1.5
          IAPTR = IAPTR + 1
  4       B(K) = BK
          K = K + 1
          IA(K) = IAPTR
  5       CONTINUE
     CALL PREORD(N,IA,R,C,IC)
```

```
      CALL NSPIV(N,IA,JA,A,B,MAX,R,C,IC,X,ITEMP,RTEMP,
     * IERR)
      WRITE (6,1Ø1) IERR
 1Ø1  FORMAT (8H IERR = ,I1Ø)
      CALL RESCHK(N,IA,JA,A,B,X)
      STOP
      END

      SUBROUTINE PREFORD(N,IA,R,C,IC)
      INTEGER IA(1),R(1),C(1),IC(1)
      DO 1 I=1,N
        R(I) = I
        C(I) = I
        IC(I) = I
  1     CONTINUE
      DO 5 I = 1,N
        C(I) = Ø
      DO 1Ø k = 1,N
        KDEG = IA(K+1) - IA(K)
        IF (KDEG .EQ. Ø) KDEG = KDEG + 1
        IC(K) = C(KDEG)
        C(KDEG) = K
  1Ø    CONTINUE
      I = Ø
      DO 3Ø J = 1,N
        IF (C(J) .EQ. Ø) GO TO 3Ø
        K = C(J)
  2Ø    I = I + 1
        R(I) = K
        K = IC(K)
        IF (K .GT. Ø) GO TO 2Ø
  3Ø    CONTINUE
      DO 4Ø I = 1,N
        C(I) = I
        IC(I) = I
  4Ø    CONTINUE
      RETURN
      END

      SUBROUTINE RESCHK(N,IA,JA,A,B,X)
      INTEGER IA(1),JA(1)
      DOUBLE PRECISION RESID,RESIDM,ROWSUM
      RESID = Ø.
      RESIDM = Ø.
      DO 2Ø I=1,N
        ROWSUM = DBLE(B(I))
        JMIN = IA(I)
        JMAX = IA(I+1) - 1
     DO 1Ø J=JMIN,JMAX
       JAJ = JA(J)
       ROWSUM = ROWSUM - DBLE(A(J)) * DBLE(X(JAJ))
1Ø     CONTINUE
     IF (DABS(ROWSUM) .GT. RESIDM) RESIDM = DABS(ROWSUM)
     RESID = RESID + ROWSUM**2
2Ø   CONTINUE
   RESID = DSQRT(RESID)
   WRITE (6,25) RESID
25 FORMAT (22H 2-NORM OF RESIDUAL = ,D14.7)
   WRITE (6,3Ø) RESIDM
3Ø FORMAT(24H MAX NORM OF RESIDUAL = ,D14.7)
   RETURN
   END
```

APPENDIX B

THE PL/I PROGRAMS

```
CRF:    PROC(ZS,HS,HM,DM,DS,ZE,HE,DE,N);
             DCL W(3) FLOAT DECIMAL(8);
             DCL (ZO,ZD,ZZ,Z(3),CW,A,V,U(7))
                   FLOAT DECIMAL(8) COMPLEX;
           U(1)=1.+0.I;
           U(2)=0.8660254+0.5000000I;
           U(3)=0.0000000+1.0000000I;
           U(4)=0.9659258+0.2588190I;
           U(5)=0.7071068+0.7071068I;
           U(6)=0.2588190+0.9659258I;
           U(7)=-0.2588190+0.9659258I;
           H=HS;
           ZO=ZS;
           N=0;
           CW=FUNC(ZO);
           WO=ABS(REAL(CW))+ABS(IMAG(CW));
           DS=WO;
           I1=1;
           I2=1;
           I3=1;
           IF WO-DM > 0  THEN
             DO;
                K=1;
                I=0;
LP1:            DO WHILE (I1=1);
                   V=-1.+0.I;
LP2:               DO WHILE (I2=1);
                      A=-0.5+0.866I;
LP3:                  DO WHILE (I3=1);
                         Z(1)=ZO+H*V*A;
                         CW=FUNC(Z(1));
                         W(1)=ABS(REAL(CW))+ABS(IMAG(CW));
                         Z(2)=ZO+H*V;
                         CW=FUNC(Z(2));
                         W(2)=ABS(REAL(CW))+ABS(IMAG(CW));
                         Z(3)=ZO+H*CONJG(A)*V;
                         CW=FUNC(Z(3));
                         W(3)=ABS(REAL(CW))+ABS(IMAG(CW));
                         N=N+1;
                         IF W(1)-W(3) > 0  THEN
                            IF W(2)-W(3) > 0  THEN  NR=3;
                            ELSE  NR=2;
                         ELSE
                            IF W(1)-W(2) < 0  THEN  NR=1;
                            ELSE  NR=2;
                         IF WO-W(NR) < 0  THEN
                            IF K = 1  THEN
                              DO;
                                 K=2;
                                 IF H < HM  THEN
                                   DO;
                                      I1=0;
                                      I2=0;
                                      I3=0;
                                    END;
                                 ELSE
                                    DO;
                              H=H*0.25;
                              I3=0;
                              I2=1;
                           END;
                     END;
                  ELSE
                  IF K = 2  THEN
                    DO;
                       K=3;
                       I3=0;
                       I2=0;
                       I1=1;
                    END;
                  ELSE
                  IF K = 3  THEN
                    DO;
                       I=I+1;
                       IF I-7 <=  0  THEN
                         DO;
                            V=U(I);
                            I3=0;
                            I2=1;
                         END;
                       ELSE
                           IF H < HM  THEN
                             DO;
                                I1=0;
                                I2=0;
                                I3=0;
                             END;
                           ELSE
                               DO;
                                  H=H*0.25;
                                  I=0;
                                  I2=0;
                                  I3=0;
                                  I1=1;
                               END;
                    END;
              ELSE
                  DO;
                     K=1;
                     I=0;
                     A=0.707+0.707I;
                     V=(Z(NR)-ZO)/H;
                     WO=W(NR);
                     ZO=Z(NR);
                     IF WO-DM <= 0  THEN
                       DO;
                          I1=0;
                          I2=0;
                          I3=0;
                       END;
                     ELSE I3=0;
                  END;
           END LP3;
        END LP2;
     END LP1;
  END;
ZE=ZO;
HE=H;
```

```
  DE=WD;
  RETURN;
END CRF;
```

```
GCDN:   PROC(N,A,Z,IGCD);
            DCL (A(*),Z(*),C1,C2,Y1,Y2,Q) FIXED BINARY(31);
LP1:      DO M=1 TO N;
            IF A(M) ¬= 0  THEN
              IF M = N  THEN
                DO;
                   IGCD=A(M);
                   Z(M)=1;
                   RETURN;
                END;
              ELSE
                  DO;
                     MP1=M+1;
                     MP2=M+2;
                     ISIGN=0;
                     IF A(M) < 0  THEN
                       DO;
                          ISIGN=1;
                          A(M)=-A(M);
                       END;
                     C1=A(M);
LP2:                 DO I=MP1 TO N;
                       IF A(I)  ¬= 0  THEN
                         DO;
                            Y1=1;
                            Y2=0;
                            C2=ABS(A(I));
                            KK=1;
LP3:                        DO WHILE (KK=1);
                               Q=C2/C1;
                               C2=C2-Q*C1;
                               IF C2 = 0  THEN  KK=0;
                               ELSE
                                   DO;
                                      Y2=Y2-Q*Y1;
                                      Q=C1/C2;
                                      C1=C1-Q*C2;
                                      IF C1 = 1  THEN
                                        DO;
                                           C1=C2;
                                           Y1=Y2;
                                           KK=0;
                                        END;
                                      ELSE
                                          Y1=Y1-Q*Y2;
                                   END;
                            END LP3;
                            Z(I)=(C1-Y1*A(M))/A(I);
                            A(I)=Y1;
                            A(M)=C1;
                         END;
                       ELSE
                           DO;
                              ISIGN=1;
                              A(M)=-A(M);
                           END;
                       IF C1 = 1  THEN
```

```
                    DO;
                       IP1=I+1;
                       DO J =IP1 TO N;
                          Z(J)=0;
                       END;
                    END;
                END LP2;
                IGCD=A(M);
                DO J=MP2 TO I;
                   K=I-J+2;
                   KK=K+1;
                   Z(K)=Z(K)*A(KK);
                   A(K)=A(K)*A(KK);
                END;
                Z(M)=A(MP1);
                IF ISIGN ¬= 0  THEN  Z(M)=-Z(M);
                RETURN;
             END;
         ELSE
             Z(M)=0;
   END LP1;
   IGCD=0;
END GCDN;
```

```
CCQUAD:  PROC(F,A,B,TOLERR,LIMIT,ESTERR,USED,CSXFRM)
               RETURNS (FLOAT DECIMAL(8));
           DCL (CENTRE,WIDTH,SHIFT,FUND,ANGLE,C,S,NEWINT,T1,T2,T3,T4,
                T5,T6,T7,T8,T9,T10,T11,T12,SCLINT,SCLERR,CSXFRM(*),
                F(*)) FLOAT DECIMAL(8);
           DCL (N,N2,N3,NLESS1,NLESS3,MAXX,J,STEP,L(8),L1,L2,L3,L4,L5,
               L6,L7,L8,J1,J2,J3,J4,J5,J6,J7,J8,JREV) FIXED BINARY(31);
           DCL (PI INIT(3.14159265359), RT3 INIT(1.73205080757))
               FLOAT DECIMAL(16);
           DCL MMAX INIT(8) FIXED BINARY(31);
           CENTRE=(A+B)*.5E0;
           WIDTH=(B-A)*.5E0;
           MAXX=MIN(LIMIT,2*3**(MMAX+1));
           DO J=1 TO MMAX;
              L(J)=1;
           END;
           N=6;
           CSXFRM(1)=F(A);
           CSXFRM(7)=F(B);
           SHIFT=WIDTH*RT3*.5E0;
           CSXFRM(2)=F(CENTRE-SHIFT);
           CSXFRM(6)=F(CENTRE+SHIFT);
           SHIFT=WIDTH*.5E0;
           CSXFRM(3)=F(CENTRE-SHIFT);
           CSXFRM(5)=F(CENTRE+SHIFT);
           CSXFRM(4)=F(CENTRE);
           T1=CSXFRM(1)+CSXFRM(7);
           T2=CSXFRM(1)-CSXFRM(7);
           T3=2.E0*CSXFRM(4);
           T4=CSXFRM(2)+CSXFRM(6);
           T5=(CSXFRM(2)-CSXFRM(6))*RT3;
           T6=CSXFRM(3)+CSXFRM(5);
           T7=CSXFRM(3)-CSXFRM(5);
           T8=T1+2.E0*T6;
           T9=2.E0*T4+T3;
           T10=T2+T7;
           T11=T1-T6;
           T12=T4-T3;
           CSXFRM(1)=T8+T9;
           CSXFRM(2)=T10+T5;
           CSXFRM(3)=T11+T12;
           CSXFRM(4)=T2-2.E0*T7;
           CSXFRM(5)=T11-T12;
           CSXFRM(6)=T10-T5;
           CSXFRM(7)=T8-T9;
           USED=7;
           OLDINT=(T1+2.E0*T3)/3.E0;
           KK=1;
LP1:       DO WHILE(KK=1);
              NLESS3=N-3;
              NEWINT=.5E0*CSXFRM(USED)/(1-N**2);
              DO J=1 TO NLESS3 BY 2;
                 JREV=N-J;
                 NEWINT=NEWINT+CSXFRM(JREV)/(JREV*(2-JREV));
              END;
              NEWINT=NEWINT+.5E0*CSXFRM(1);
              ESTERR=ABS(OLDINT*3.E0-NEWINT);
```

```
IF ABS(NEWINT)*TOLERR < ESTERR  THEN
  DO;
     OLDINT=NEWINT;
     IF  (3*N+1) <= MAXX  THEN
       DO;
          DO J=2 TO MMAX;
             L(J-1)=L(J);
          END;
          L(MMAX)=3*L(MMAX-1);
          J=USED;
          FUND=PI/(3*N);
          DO J1=1 TO L1;
             DO J2=J1 TO L2 BY L1;
                DO J3=J2 TO L3 BY L2;
                   DO J4=J3 TO L4 BY L3;
                      DO J5=J4 TO L5 BY L4;
                         DO J6=J5 TO L6 BY L5;
                            DO J7=J6 TO L7 BY L6;
                               DO J8=J7 TO L8 BY L7;
                               JREV=J8;
                               ANGLE=FUND*(3*JREV-2);
                               SHIFT=WIDTH*COS(ANGLE);
                               T1=F(CENTRE-SHIFT);
                               T3=F(CENTRE+SHIFT);
                               SHIFT=WIDTH*SIN(ANGLE);
                               T2=F(CENTRE+SHIFT);
                               T4=F(CENTRE-SHIFT);
                               T5=T1+T3;
                               T6=T2+T4;
                               CSXFRM(J+1)=T5+T6;
                               CSXFRM(J+2)=T1-T3;
                               CSXFRM(J+3)=T5-T6;
                               CSXFRM(J+4)=T2-T4;
                               J=J+4;
                               END;
                            END;
                         END;
                      END;
                   END;
                END;
             END;
          END;
          N2=2*N;
          STEP=4;
          KK2=1;
          DO WHILE(KK2=1);
             J1=USED+STEP;
             J2=USED+2*STEP;
             CALL R3PASS(N2,STEP,N2-2*STEP,CSXFRM(USED+1)
                         ,CSXFRM(J1+1),CSXFRM(J2+1));
             STEP=3*STEP;
             IF STEP >= N  THEN  KK2=0;
          END;
          T1=CSXFRM(1);
          T2=CSXFRM(USED+1);
          CSXFRM(1)=T1+2.E0*T2;
          CSXFRM(USED+1)=T1-T2;
          T1=CSXFRM(N+1);
          T2=CSXFRM(N2+2);
          CSXFRM(N+1)=T1+T2;
          CSXFRM(N+2)=T1-2.E0*T2;
```

```
                      N3=3*N;
                      NLESS1=N-1;
                      DO J=1 TO NLESS1;
                         J1=N+J;
                         J2=N3-J;
                         ANGLE=FUND*J;
                         C=COS(ANGLE);
                         S=SIN(ANGLE);
                         T1=C*CSXFRM(J1+2)-S*CSXFRM(J2+2);
                         T2=(S*CSXFRM(J1+2)+C*CSXFRM(J2+2))*RT3;
                         CSXFRM(J1+2)=CSXFRM(J+1)-T1-T2;
                         CSXFRM(J2+2)=CSXFRM(J+1)-T1+T2;
                         CSXFRM(J+1)=CSXFRM(J+1)+2.E0*T1;
                      END;
                      T1=CSXFRM(N2+1);
                      T2=CSXFRM(N2+2);
                      DO J=1 TO NLESS1;
                         J1=USED+J;
                         J2=N2+J;
                         CSXFRM(J2)=CSXFRM(J1);
                         CSXFRM(J1)=CSXFRM(J2+2);
                      END;
                      CSXFRM(N3)=T1;
                      CSXFRM(N3+1)=T2;
                      N=N3;
                      USED=N+1;
                   END;
                 ELSE
                     DO;
                        PUT SKIP EDIT(' ')(A);
                        KK=0;
                     END;
             END;
           ELSE KK=0;
        END LP1;
        RETURN(WIDTH*NEWINT/(N/2));
        ESTERR=WIDTH*ESTERR/(N/2);
      END CCQUAD;


3PASS:  PROC(N2,M,LENGT,X0,X1,X2);
        DCL (N2,M,LENGT,HALFM,M3,K,K0,K1,J,J0,J1)
             FIXED BINARY(31);
        DCL (TWOPI INIT(6.283185307), HAFRT3 INIT(.866025403))
             FLOAT DECIMAL(16);
        DCL (RSUM,RDIFF,RSUM2,ISUM,IDIFF,IDIFF2,FUNC,ANGLE,C1,
             S1,C2,S2,R0,R1,R2,I0,I1,I2)
             FLOAT DECIMAL(8);
        DCL (X0(*),X1(*),X2(*)) FLOAT DECIMAL(8);
        HALFM=(M-1)/2;
        M3=M*3;
        FUND=TWOPI/M3;
        DO K=1 TO N3 BY M3;
           RSUM=(X1(K)+X2(K));
           RDIFF=(X1(K)-X2(K))*HAFRT3;
           X1(K)=X0(K)-RSUM*.5E0;
           X2(K)=RDIFF;
           X0(K)=X0(K)+RSUM;
        END;
        J=M/2+1;
        DO K=J TO N2 BY M3;
      RSUM=(X1(K)+X2(K))*HAFRT3;
      RDIFF=X1(K)-X2(K);
      X1(K)=X0(K)-RDIFF;
      X2(K)=RSUM;
      X0(K)=X0(K)+RDIFF*.5E0;
   END;
   DO J=1 TO HALFM;
      J0=J+1;
      J1=M-J+1;
      ANGLE=FUND*J;
      C1=COS(ANGLE);
      S1=SIN(ANGLE);
      C2=C1**2-S1**2;
      S2=2.E0*S1*C1;
      DO K0=J0 TO N2 BY M3;
         K1=K0-J0+J1;
         R0=X0(K0);
         I0=X0(K1);
         R1=C1*X1(K0)-S1*X1(K1);
         I1=S1*X1(K0)+C1*X1(K1);
         R2=C2*X2(K0)-S2*X2(K1);
         I2=S2*X2(K0)+C2*X2(K1);
         RSUM=R1+R2;
         RDIFF=(R1-R2)*HAFRT3;
         RSUM2=R0-.5E0*RSUM;
         ISUM=I1+I2;
         IDIFF=(I1-I2)*HAFRT3;
         IDIFF2=I0-.5E0*ISUM;
         X0(K0)=R0+RSUM;
         X0(K1)=RSUM2+IDIFF;
         X1(K0)=RSUM2-IDIFF;
         X1(K1)=RDIFF+IDIFF2;
         X2(K0)=RDIFF-IDIFF2;
         X2(K1)=I0+ISUM;
     END;
  END;
END R3PASS;
```

```
 MINI:   PROC(N,X,F,G,H,M,IH,STP,EPS,FEST,MAXF,MODE,ITER,IFUN,IERR);
             DCL (F,FEST,EPS,X(*),G(*),STP(*),H(*),D(1),XX(1),GG(1),
                  EPS2,DUM,STEP,SUM,G1,G2,STPMAX,ALPHA,AL,BL,FG,F1,
                  FM,GM,AA,BB,CC,SS,DY,YHY,C1,C2) FLOAT DEC(16);
             DCL CONV BIT(1);
           CALL CALCFG(N,X,F,G);
           IFUN=1;
           ITER=0;
           IERR=0;
           EPS2=EPS*EPS;
           DUM=0.;
           STEP=0.;
           DO I=1 TO N;
              DUM=DUM+G(I)*G(I);
              STEP=STEP+STP(I)*STP(I);
           END;
           IF DUM = 0.  THEN RETURN;
           IF IH = 0  THEN
             DO;
                DUM=100.*SQRT(STEP/DUM);
                IJ=1;
                DO I=1 TO N;
                   DO J=I TO N;
                      H(IJ)=0;
                      IF I = J  THEN  H(IJ)=DUM;
                      IJ=IJ+1;
                   END;
                END;
             END;
LP1:         DO WHILE('1'B);
                SUM=0.;
                DO I=1 TO N;
                   DUM=0.;
                   IJ=I;
                   IF I > 1  THEN
                     DO;
                        II=I-1;
                        DO J=1 TO II;
                           DUM=DUM-H(IJ)*G(J);
                           IJ=IJ+N-J;
                        END;
                     END;
                   DO J=I TO N;
                      DUM=DUM-H(IJ)*G(J);
                      IJ=IJ+1;
                   END;
                   D(I)=DUM;
                   XX(I)=X(I);
                   GG(I)=G(I);
                   SUM=SUM+DUM*DUM;
                END;
                IF (ITER >= 1 ) & (ITER <= N)  THEN
                  DO;
                     DUM=SQRT(STEP/SUM);
                     DO I=1 TO N;
                        D(I)=D(I)*DUM;
                     END;
             END;
           DO J=1 TO 2;
              G1=0.;
              DO I=1 TO N;
                 G1=G1+D(I)*G(I);
              END;
              IF G1 >= 0.  THEN
                DO I=1  TO N;
                   D(I)=-D(I);
                END;
              ELSE  J=2;
           END;
           IF G1 > 0.  THEN
             DO;
                IERR=3;
                RETURN;
             END;
           STPMAX=1.0E+30;
           DO I=1 TO N;
              IF ABS(D(I)) >= 1.E-30  THEN
                DO;
                   DUM=ABS(STP(I)/D(I));
                   IF STPMAX > DUM  THEN  STPMAX=.9*DUM;
                END;
           END;
           ALPHA=0.;
           FG=G1;
           AL=1.;
           IF MODE = 2  THEN  AL=2.*(FEST-F)/G1;
           IF AL > 1.  THEN  AL=1.;
           IF AL > STPMAX  THEN  AL=0.8*STPMAX;
           IF AL < 0.  THEN
              DO;
                IERR=4;
                RETURN;
              END;
           KK=1;
L240:      DO WHILE(KK=1);
              F1=F;
              ALPHA=ALPHA+AL;
              DO I=1  TO N;
                 X(I)=XX(I)+ALPHA*D(I);
              END;
              CALL CALCFG(N,X,F,G);
              IFUN=IFUN+1;
              IF IFUN > MAXF  THEN
                DO;
                   IERR=1;
                   RETURN;
                END;
              G2=0.0;
              DO I=1 TO N;
                 G2=G2+G(I)*D(I);
              END;
              IF (G2 > FG) & (F1 > (F+0.0001*G1))  THEN
                 CONV='1'B;
              IF CONV & MODE = 2  THEN  KK=0;
              ELSE
                  IF (F > F1) | (G2 > 0.)  THEN  KK=0;
                  ELSE
                       DO;
```

```
                          BL=AL;
                          AL=0.5*G1*BL*BL/(F1-F+BL*G1);
                          IF (MODE = 2) & (AL < BL)  THEN
                             AL=2.*(FEST-F)/G1;
                          IF AL > (10.*BL)  THEN  AL=10.*BL;
                          IF AL > STPMAX  THEN  AL=0.8*STPMAX;
                          IF AL < (1.001*BL)  THEN  AL=BL+BL;
                          G1=G2;
                       END;
            END L240;
              IF ¬CONV | MODE = 1  THEN
                DO;
                   KK2=1;
L300:              DO WHILE(KK2=1);
                      BL=AL;
                      AA=(G1+G2+2.*(F1-F)/AL)/(AL*AL);
                      BB=(G2-3.*AA*AL*AL-G1)/(AL+AL);
                      CC=BB*BB-3.**AA*G1;
                      DUM=ABS(AA)*1.0E+05-ABS(BB);
                      AL=(.5*G1*BL*BL)/(F1-F+BL*G1);
                      IF (CC > 0.) & (DUM > 0.)  THEN
                         AL=(-BB+SQRT(CC))/(3.*AA);
                      FM=F;
                      GM=G2;
                      IF AL <= (0.001*BL)  THEN  AL=.1*BL;
                      IF AL > (0.999*BL)  THEN  AL=.8*BL;
                      ALPHA=ALPHA-BL+AL;
                      DO I=1 TO N;
                         X(I)=XX(I)+ALPHA*D(I);
                      END;
                      CALL CALCFG(N,X,F,G);
                      IFUN=IFUN+1;
                      IF IFUN > MAXF  THEN
                        DO;
                           IERR=1;
                           RETURN;
                        END;
                      G2=0.;
                      DO I=1 TO N;
                         G2=G2+D(I)*G(I);
                      END;
                      IF (G2 > FG) & (F1 > (F+.0001*G1))  THEN
                         CONV='1'B;
                      IF ¬CONV  THEN  KK2=0;
                      ELSE
                          IF (G2>FG) & (BL <= EPS2)  THEN  KK2=0;
                          ELSE
                              DO;
                                 SS=0.;
                                 DO I=1 TO N;
                                    SS=SS+ABS(G(I)*D(I));
                                 END;
                                 SS=SS/(100000.*N);
                                 IF (G2 < SS) & (G2 > FG)  THEN KK2=0;
                                 ELSE
                                     IF (G2<0.) & (F1>F)  THEN
                                       DO;
                                          G1=G2;
                                          G2=GM;
                                          F1=F;
                                          F=FM;
                                          AL=BL-AL;
                                          ALPHA=ALPHA+AL;
                                       END;
                              END;
                   END L300;
                END;
             CONV='1'B;
             STEP=0.;
             DO I=1 TO N;
                D(I)=X(I)-XX(I);
                STEP=STEP+D(I)*D(I);
                IF (G(I)*G(I)) > EPS2  THEN  CONV='0'B;
             END;
             IF CONV  THEN  RETURN;
             IF STEP <= (EPS2*EPS2)  THEN
               DO;
                  IERR=2;
                  RETURN;
               END;
             DY=0.;
             YHY=0.;
             DO I=1 TO N;
                GG(I)=G(I)-GG(I);
                DY=DY+D(I)*GG(I);
             END;
             DO I=1 TO N;
                DUM=0.;
                IJ=I;
                IF I > 1  THEN
                  DO;
                     II=I-1;
                     DO J=I TO II;
                        DUM=DUM+H(IJ)*GG(J);
                        IJ=IJ+N-J;
                     END;
                  END;
                DO J=1 TO N;
                   DUM=DUM+H(IJ)*GG(J);
                   IJ=IJ+1;
                END;
                YHY=YHY+DUM*GG(I);
                XX(I)=DUM;
             END;
             C1=1.+YHY/DY;
             DO I=1 TO N;
                GG(I)=C1*D(I)-XX(I);
             END;
             IJ=1;
             DO I=1 TO N;
                C1=D(I)/DY;
                C2=XX(I)/DY;
                DO J=I TO N;
                   H(IJ)=H(IJ)+C1*GG(J)-C2*D(J);
                   IJ=IJ+1;
                END;
             END;
             ITER=ITER+1;
          END LP1;
      END MINI;
```

```
DERPAR:   PROC(N,XLOW,XUPP,EPS,W,INITAL,ITIN,HH,HMAX,PREF,NDIR,E,
               MXADMS,NCORR,NCRAD,NOUT,OUT,MAXOUT,NPRNT);
            DCL  (X(11),XLOW(*),XUPP(*),W(*),HMAX(*),PREF(*),
                  NDIR(*),OUT(*,*),F(11),G(10,11),BETA(11)
                  ,DXDT(11)) FLOAT DECIMAL(8);
            DCL  (INDIC INIT('*'),INDSP INIT(' '),MARK(11)) CHAR(1);
          N1=N+1;
          LW=3;
          IF INITAL ¬= 0  THEN
            DO;
               DO L=1 TO ITIN;
                  CALL FCTN(N,X,F,G);
                  SQUAR=0.0;
                  DO I=1 TO N;
                     SQUAR=SQUAR+F(I)**2;
                  END;
                  LL=L-1;
                  SQUAR=SQRT(SQUAR);
                  IF NPRNT ¬= 3  THEN  PUT EDIT (LL,(X(I)DO I=1 TO N1)
                                              ,SQUAR) (R(L99999));
                  CALL GAUSE(N,G,F,M,10,11,PREF,BETA,K);
                  IF M = 0  THEN
                    DO;
                       PUT EDIT((X(I) DO I=1 TO N1)) (R(L99990));
                       MAXOUT=-3;
                       RETURN;
                    END;
                  P=0.0;
                  DO J=1 TO N1;
                     X(J)=X(J)-F(J);
                     P=P+ABS(F(J)*W(J));
                  END;
                  IF P <= EPS  THEN  GO TO L50;
               END;
               PUT EDIT(ITIN) (R(L99998));
               ITIN=-1;
               IF INITAL = 1  THEN  RETURN;
L50:           IF NPRNT ¬= 3  THEN PUT EDIT ((X(I) DO I=1 TO N1))
                                              (R(L99997));
               IF INITAL = 2  THEN  RETURN;
            END;
          IF NPRNT ¬= 3  THEN  PUT EDIT(' ') (R(L99996));
          KOUT=0;
          NOUT=0;
          MADMS=0;
          NC=1;
          K1=0;
P1:       DO WHILE('1'B);
             CALL FCTN(N,X,F,G);
             SQUAR=0.0;
             DO I=1 TO N;
                SQUAR=SQUAR+F(I)**2;
             END;
             CALL GAUSE(N,G,F,M,10,11,PREF,BETA,K);
             IF M = 0  THEN
               DO;
                  PUT EDIT((X(I) DO I=1 TO N1)) (R(L99990));
```

```
         MAXOUT=-3;
      END;
    IF K1 ¬= K  THEN
      DO;
         MADMS=0;
         K1=K;
      END;
    SQUAR=SQRT(SQUAR);
    IF NCRAD = 1  THEN  SQUAR=-SQUAR;
    P=0.0;
    DO I=1 TO N1;
       P=P+W(I)*ABS(F(I));
    END;
    IF P > EPS  THEN
     DO;
      IF NC >= NCORR  THEN
        IF NCORR ¬= 0  THEN  PUT EDIT(NCORR,P) (R(L99995));
        NC=1;
        IF NCRAD ¬= 0  THEN
          DO I=1 TO N1;
             X(I)=X(I)-F(I);
          END;
    NOUT=NOUT+1;
        DO I=1 TO N1;
           MARK(I)=INDSP;
        END;
        MARK(K)=INDIC;
        IF NPRNT ¬= 3  THEN
          DO;
          PUT EDIT (' ')(R(L99994));
             PUT EDIT((X(I),MARK(I) DO I=1 TO N1),SQUAR)
                     (R(L99993));
          END;
        IF NPRNT ¬= 1  THEN
         DO;
          IF NOUT > 100  THEN
            DO;
               PUT EDIT(' ')(R(L99992));
               RETURN;
            END;
          DO I=1 TO N1;
             OUT(NOUT,I)=X(I);
          END;
          OUT(NOUT,N+2)=SQUAR;
         END;
        ELSE
            IF NOUT = 1  THEN
              DO;
                 DO I=1 TO N1;
                    OUT(NOUT,I)=X(I);
                 END;
                 OUT(NOUT,N+2)=SQUAR;
              END;
        IF NOUT >= MAXOUT  THEN  RETURN;
        DO I=1 TO N1;
          IF X(I) < XLOW(I) | X(I) > XUPP(I)  THEN
            DO;
               MAXOUT=-2;
               RETURN;
            END;
         END;
```

```
                    IF NOUT > 3  THEN
                      DO;
                         P=0.0;
                         DO I=1 TO N1;
                            P=P+W(I)*ABS(X(I)-OUT(1,I));
                         END;
                         IF P <= E  THEN
                           DO;
                              PUT EDIT(' ')(R(L99991));
                              MAXOUT=-1;
                              RETURN;
                           END;
                      END;
                    DXK2=1.0;
                    DO I=1 TO N1;
                       DXK2=DXK2+BETA(I)**2;
                    END;
                    DXDT(K)=1.0/SQRT(DXK2)*NDIR(K);
                    H=HH;
                    DO I=1 TO N1;
                       NDIR(I)=1;
                       IF I ¬= K  THEN  DXDT(I)=BETA(I)*DXDT(K);
                       IF DXDT(I) < 0.0  THEN  NDIR(I)=-1;
                       IF H*ABS(DXDT(I)) > HMAX(I)  THEN
                         DO;
                            MADMS=0;
                            H=HMAX(I)/ABS(DXDT(I));
                         END;
                    END;
                    IF NOUT > KOUT+3  THEN
                      IF H*ABS(DXDT(K)) > 0.8*ABS(X(K)-OUT(1,K))  THEN
                        IF (OUT(1,K)-X(K))*NDIR(K) > 0.0  THEN
                          DO;
                              MADMS=0;
                              IF H*ABS(DXDT(K)) > ABS(X(K)-OUT(1,K))THEN
                                DO;
                                   H=ABS(X(K)-OUT(1,K))/ABS(DXDT(K));
                                   KOUT=NOUT;
                                END;
                          END;
                 END;
                ELSE
                    DO;
                       DO I=1 TO N1;
                          X(I)=X(I)-F(I);
                       END;
                       NC=NC+1;
                    END;
            END _P1;
L99999:     FORMAT(X(3),F(3),X(22),5 F(15,7));
L99998:     FORMAT(F(5),X(2));
L99997:     FORMAT(X(3),X(22),X(4),5 F(15,7));
L99996:     FORMAT(A);
L99995:     FORMAT(F(5),F(15,7));
L99994:     FORMAT(A);
L99993:     FORMAT(X(33),5 (F(15,7),A));
L99992:     FORMAT(A);
L99991:     FORMAT(A);
L99990:     FORMAT(5 F(12,6),X(48),5 F(12,6));
          END DERPAR;
```

```
AUSE: PROC(N,A,B,M,NN,MM,PREF,BETA,K);
        DCL (A(*,*),B(*),PREF(*),BETA(*),Y(11),X(11))
             FLOAT DECIMAL(8);
        DCL (IRR(11),IRK(11)) FIXED BINARY(31);
        N1=N+1;
        ID=1;
        M=1;
        DO I=1 TO N1;
           IRK(I)=0;
           IRR(I)=0;
        END;
        DO WHILE('1'B);
           IR=1;
           IS=1;
           AMAX=0.0;
           DO I=1 TO N;
              IF IRR(I) = 0  THEN
                DO J=1 TO N1;
                   P=PREF(J)*ABS(A(I,J));
                   IF P-AMAX > 0  THEN
                     DO;
                        IR=I;
                        IS=J;
                        AMAX=P;
                     END;
                END;
           END;
           IF AMAX = 0.0  THEN
             DO;
                M=0;
                RETURN;
             END;
           IRR(IR)=IS;
           DO I=1 TO N;
              IF ¬(I = IR | A(I,IS) = 0.0)  THEN
                DO;
                   P=A(I,IS)/A(IR,IS);
                   DO J=1 TO N1;
                      A(I,J)=A(I,J)-P*A(IR,J);
                   END;
                   A(I,IS)=0.0;
                   B(I)=B(I)-P*B(IR);
                END;
           END;
           ID=ID+1;
           IF ID > N  THEN
             DO;
                DO I=1 TO N;
                   IR=IRR(I);
                   X(IR)=B(I)/A(I,IR);
                   IRK(IR)=1;
                END;
                DO K=1 TO N1;
                   IF IRK(K) = 0  THEN
                     DO;
                        K1=K;
                        K=N1;
                     END;
                END;
                K=K1;
                DO I=1 TO N;
                   IR=IRR(I);
                   Y(IR)=-A(I,K)/A(I,IR);
                END;
                DO I=1 TO N1;
                   B(I)=X(I);
                   BETA(I)=Y(I);
                END;
                B(K)=0.0;
                BETA(K)=0.0;
                RETURN;
             END;
         END;
       END GAUSE;

ADAMS: PROC(N,D,MADMS,H,X,MXADMS);
         DCL (DER(4,11),X(*),D(*)) FLOAT DECIMAL(8);
         N1=N+1;
         DO I=1 TO 3;
            DO J=1 TO N1;
               DER(I+1,J)=DER(I,J);
            END;
         END;
         MADMS=MADMS+1;
         IF MADMS > MXADMS  THEN  MADMS=MXADMS;
         IF MADMS > 4  THEN  MADMS=4;
         DO I=1 TO N1;
            DER(1,I)=D(I);
            IF MADMS = 1  THEN  X(I)=X(I)+H*DER(1,I);
            IF MADMS = 2  THEN X(I)=X(I)+.5*H*(3.0*DER(1,I)-DER(2,I));
            IF MADMS = 3  THEN X(I)=X(I)+H*(23.0*DER(1,I)-16.0*DER(2,I)
                               +5.0*DER(3,I))/12.0;
            IF MADMS = 4  THEN X(I)=X(I)+H*(55.0*DER(1,I)-59.0*DER(2,I)
                               +37.0*DER(3,I)-9.0*DER(4,I))/24.0;
         END;
       END ADAMS;
```

```
SPN:    PROC(K,L,II,JJ,MINN);
            DCL (K(*),L(*)) FIXED BINARY(31);
            DCL TAB(57) INIT(3,5,10,17,26,37,50,65,82,101,122,145,170,
                             197,226,257,290,325,362,401,442,485,530,
                             557,626,677,730,785,842,901,962,1025,1090,
                             1157,1226,1297,1370,1445,1522,1601,1682,
                             1765,1850,1937,2026,2117,2210,2305,2402,
                             2501,2602,2705,2810,2917,3026,3137,3250)
                             FIXED BINARY(31);
        MINN=II;
        MAXX=II;
        L(II)=II;
        IF JJ-II > 0  THEN
          DO;
             KMIN=K(MINN);
             KMAX=KMIN;
             IA=II+1;
             IRT=1;
             ITAB=TAB(1)+II-1;
             ISTRT=II;
             DO J=IA TO JJ;
                IF J-ITAB >= 0  THEN
                  DO;
                     IRT=IRT+1;
                     ISTRT=ISTRT+1;
                     ITAB=TAB(IRT)+II-1;
                  END;
                KJ=K(J);
                IF KJ-KMAX >= 0  THEN
                  DO;
                     I=MAXX;
                     MAXX=J;
                     KMAX=K(J);
                     IPOI=L(I);
                     L(I)=J;
                     L(J)=IPOI;
                  END;
                ELSE
                    IF KJ-KMIN <= 0  THEN
                      DO;
                         I=MAXX;
                         MINN=J;
                         KMIN=K(J);
                         IPOI=L(I);
                         L(I)=J;
                         L(J)=IPOI;
                      END;
                    ELSE
                        IF KJ-KMIN = 0  THEN
                          DO;
                             MADIN=MINN;
                             MINN=J;
                             KK=1;
                             DO WHILE (KK=1);
                                IPOI=MADIN;
                                MADIN=L(IPOI);
                                IF KJ-K(MADIN) < 0  THEN
                             DO;
                                L(IPOI)=J;
                                L(J)=MADIN;
                                KK=0;
                             END;
                       END;
                    END;
                  ELSE
                      DO;
                          MADIN=MINN;
                          IPOI=J-1;
                          I=KJ-KMIN;
                          DO KEY=ISTRT TO IPOI BY IRT;
                             KDIFF=KJ-K(KEY);
                             IF KDIFF = 0  THEN
                                DO;
                                   MADIN=KEY;
                                   KEY=IPOI;
                                END;
                             ELSE
                                 IF KDIFF > 0  THEN
                                    IF I-KDIFF > 0  THEN
                                      DO;
                                         I=KDIFF;
                                         MADIN=KEY;
                                      END;
                          END;
                          KK=1;
                          DO WHILE (KK=1);
                             IPOI=MADIN;
                             MADIN=L(IPOI);
                             IF KJ-K(MADIN) < 0  THEN
                               DO;
                                  L(IPOI)=J;
                                  L(J)=MADIN;
                                  KK=0;
                               END;
                          END;
       END;
    END;
  MINN=L(MAXX);
  L(MAXX)=0;
  RETURN;
END SPN;
```

```
PROFIT:   PROC(NR,NDSTK,NEW,NDEG,LVLS2,LVLST,LSTPT,NXTNUM,S2,S3,Q,
               CONECT);
               DCL  (S2SZE,NEW(*),NDEG(*),LVLS2(*),LVLST(*),LSTPT(*),
                     NDSTK(*,*),S2(*),S3(*),Q(*),CONECT(*),
                     S3SZE,QPTR,CONSZE,M) FIXED BINARY(31);
            NSTPT=1;
            DO I=1 TO IDPTH;
               LSTPT(I)=NSTPT;
               DO J=1 TO N;
                  IF   LVLS2(J)  =  I   THEN
                    DO;
                       LVLST(NSTPT)=J;
                       NSTPT=NSTPT+1;
                    END;
               END;
            END;
              LSTPT(IDPTH+1)=NSTPT;
              LEVEL=1;
              CALL FORMLV(S2,S2SZE,LSTPT,LVLST,LEVEL);
 P1:          DO WHILE ('1'B);
                 CALL FORMLV(S3,S3SZE,LSTPT,LVLST,LEVEL+1);
                 QPTR=0;
                 KK=1;
 P2:             DO WHILE(KK=1);
                    M=MINCON(S2,S2SZE,S3,S3SZE,CONECT,CONSZE,NDSTK,NR,
                             NDEG);
                    NEW(M)=NXTNUM;
                    NXTNUM=NXTNUM+1;
                    CALL DLETE(S2,S2SZE,M);
                    IF CONSLE > 0  THEN
                      DO I=1 TO CONSZE;
                         QPTR=QPTR+1;
                         Q(QPTR)=CONECT(I);
                         CALL DLETE(S3,S3SZE,CONECT(I));
                      END;
                    IF S2SZE <= 0  THEN
                      DO;
                         IF S3SZE > 0  THEN
                           DO I=1 TO S3SZE;
                              QPTR=QPTR+1;
                              Q(QPTR)=S3(I);
                           END;
                         LEVEL=LEVEL+1;
                         IF LEVEL >= IDPTH  THEN
                           DO;
                              DO I=1 TO QPTR;
                                 IQ=Q(I);
                                 NEW(IQ)=NXTNUM;
                                 NXTNUM=NXTNUM+1;
                              END;
                              RETURN;
                           END;
                         DO I=1 TO QPTR;
                           S2(I)=Q(I);
                         END;
                         S2SZE=QPTR;
                         KK=0;
                    END;
                  ELSE
                      IF S3SZE <= 0   THEN
                        DO;
                           DO I=1 TO S2SZE;
                              NS2=S2(I);
                              NEW(NS2)=NXTNUM;
                              NXTNUM=NXTNUM+1;
                           END;
                           LEVEL=LEVEL+1;
                           IF LEVEL >= IDPTH   THEN
                             DO;
                                DO I=1 TO QPTR;
                                  IQ=Q(I);
                                  NEW(IQ)=NXTNUM;
                                  NEXTNUM=NEXTNUM+1;
                                END;
                                RETURN;
                             END;
                           DO I=1 TO QPTR;
                              S2(I)=Q(I);
                           END;
                           S2SZE=QPTR;
                           KK=0;
                        END;
               END LP2;
            END LP1;
          END PROFIT;


MINCON:   PROC(X,XSZE,Y,YSZE,CONLST,CONSZE,NDSTK,NR,NDEG)
               RETURNS (FIXED BINARY(31));
               DCL (NDSTK(*,*),X(*),XSZE,Y(*),YSZE,CONLST(*),CONSZE,
                    NDEG(*),SMLST(100),I) FIXED BINARY(31);
            CONSZE=YSZE+1;
            DO I=1 TO XSZE;
               LSTSZE=0;
               IX=X(I);
               IROWDG=NDEG(IX);
               DO J=1 TO YSZE;
                  DO K=1 TO IROWDG;
                     IF NDSTK(IX,K) = Y(J)   THEN
                       DO;
                          SMLST(LSTSZE+1)=Y(J);
                          LSTSZE=LSTSZE+1;
                          IF LSTSZE >= CONSZE   THEN GO TO L50;
                          K=IROWDG;
                       END;
                  END;
               END;
               IF LSTSZE <= 0   THEN
                 DO;
                    RETURN(X(I));
                    CONSZE=0;
                    RETURN;
                 END;
               CONSZE=LSTSZE;
               DO J=1 TO LSTSZE;
                  CONLST(J)=SMLST(J);
               END;
```

```
              RETURN(X(I));
L50:        END;
         END MINCON;


LETE:    PROC(SET,SETSZE,ELEMNT);
             DCL (SET(*),SETSZE,ELEMNT) FIXED BINARY(31);
           IF SETSZE > 1  THEN
             DO I=1 TO SETSZE;
                IF SET(I) = ELEMNT  THEN
                  DO;
                     SETSZE=SETSZE-1;
                     DO J=I TO SETSZE;
                        SET(J)=SET(J+1);
                     END;
                     RETURN;
                  END;
             END;
           ELSE
               IF ¬(SETSZE = 1 & SET(1) ¬= ELEMNT)  THEN
                 DO;
                    SETSZE=0;
                    RETURN;
                 END;
           PUT SKIP EDIT(ELEMNT,(SET(I) DO I=1 TO SETSZE))
                         (F(6),X(3),20 F(50));
         END DLETE;


ORMLV:   PROC(SET,SETSZE,LSTPT,LVLST,LEVEL);
             DCL (SET(*),SETSZE,LSTPT(*),LVLST(*),UPPER)
                 FIXED BINARY(31);
           LOWER=LSTPT(LEVEL);
           UPPER=LSTPT(LEVEL+1)-1;
           SETSZE=1;
           DO I=LOWER TO UPPER;
              SET(SETSZE)=LVLST(I);
              SETSZE=SETSZE+1;
           END;
           SETSZE=SETSZE-1;
         END FORMLV;
```

```
/*                          ALGORITHM 512                              */

FACTOR:   PROC(A,B,C,G,H,N);
             DCL (A(*),B(*),C(*),G(*),H(*)) FLOAT DECIMAL(8);
           N1=N-1;
           N2=N-2;
           N3=N-3;
           N4=N-4;
           C(1)=SQRT(C(1));
           V=B(1)/C(1);
           C(2)=SQRT(C(2)-V*V);
           B(1)=V/C(2);
L10:       DO I=3 TO N2;
              I1=I-1;
              I2=I-2;
              U=A(I2)/C(I2);
              V=B(I1)/C(I1)-B(I2)*U;
              C(I)=SQRT(C(I)-U*U-V*V);
              A(I2)=U/C(I);
              B(I1)=V/C(I);
           END L10;
           G(1)=A(N1)/C(1);
           G(2)=-B(1)*G(1);
           U=A(N3)/C(N3);
           V=B(N2)/C(N2)-B(N3)*U;
L20:       DO I=3 TO N2;
              G(I)=-B(I-1)*G(I-1)-A(I-2)*G(I-2);
           END L20;
           W=0.0;
L30:       DO I=1 TO N4;
              W=W+G(I)*G(I);
           END L30;
           C(N1)=SQRT(C(N1)-W-(G(N3)+U)**2-(G(N2)+V)**2);
           W=1.0/C(N1);
           A(N3)=U*W;
           B(N2)=V*W;
L40:       DO I=1 TO N2;
              G(I)=G(I)*W;
           END L40;
           H(1)=B(N)/C(1);
           H(2)=A(N)/C(2)-B(1)*H(1);
           U=A(N2)/C(N2);
           V=B(N1)/C(N1)-B(N2)*U;
L50:       DO I=1 TO N1;
              H(I)=-B(I-1)*H(I-1)-A(I-2)*H(I-2);
           END L50;
           W=0.0;
L60:       DO I=1 TO N2;
              W=W+G(I)*H(I);
           END L60;
           H(N1)=H(N1)-W-G(N2)*U;
           W=0.0;
L70:       DO I=1 TO N3;
              W=W+H(I)*H(I);
           END L70;
           C(N)=SQRT(C(N)-W-(H(N2)+U)**2-(H(N1)+V)**2);
           W=1.0/C(N);
           A(N2)=U*W;
           B(N1)=V*W;
```

```
L80:        DO I=1 TO N1;
               H(I)=H(I)*W;
            END L80;
         END FACTOR;


SOLVE:   PROC(E,F,G,H,Q,N);
               DCL (E(*),F(*),G(*),H(*),Q(*)) FLOAT DECIMAL(8);
            N1=N-1;
            N2=N-2;
            N3=N-3;
            Q(2)=Q(2)-E(1)*Q(1);
LP10:       DO I=3 TO N2;
               Q(I)=Q(I)-E(I-1)*Q(I-1)-F(I-2)*Q(I-2);
            END LP10;
            U=0.0;
            V=0.0;
LP20:       DO I=1 TO N2;
               U=U+G(I)*Q(I);
               V=V+H(I)*Q(I);
            END LP20;
            Q(N1)=Q(N1)-E(N2)*Q(N2)-F(N3)*Q(N3)-U;
            Q(N)=Q(N)-E((N1)+H(N1))*Q(N1)-F(N2)*Q(N2)-V;
            Q(N1)=Q(N1)-(E(N1)+H(N1))*Q(N);
            Q(N2)=Q(N2)-(E(N2)+G(N2))*Q(N1)-(F(N2)+H(N2)*Q(N));
LP30:       DO II=1 TO N3;
               I=N3-II+1;
               Q(I)=Q(I)-E(I)*Q(I+1)-F(I)*Q(I+2)-G(I)*Q(N1)-H(I)*Q(N);
            END LP30;
         END SOLVE;


RHS:     PROC(D,S,N);
               DCL (D(*),S(*)) FLOAT DECIMAL(8);
LP10:       DO I=1 TO N;
               S(I)=S(I)/D(I);
            END LP10;
         END RHS;
```

```
TRANS: PROC(A,M,N,MN,MOVE,IWRK,IOK);
          DCL (A(*),MOVE(*)) FLOAT DECIMAL(8);
          IF M < 2 | N < 2  THEN
            DO;
               IOK=0;
               RETURN;
            END;
          IF MN ¬= M*N  THEN
            DO;
               IOK=-1;
               RETURN;
            END;
          IF IWRK < 1  THEN
            DO;
               IOK=-2;
               RETURN;
            END;
          IF M = N  THEN
            DO;
               N1=N-1;
               DO I=1 TO N1;
                  J1=I+1;
                  DO J=J1 TO N;
                     I1=I+(J-1)*N;
                     I2=J+(I-1)*M;
                     B=A(I1);
                     A(I1)=A(I2);
                     A(I2)=B;
                  END;
               END;
               IOK=0;
               RETURN;
            END;
          NCOUNT=2;
          K=MN-1;
          DO I=1 TO IWRK;
             MOVE(I)=0;
          END;
          IF ¬(M < 3  | M < 3)  THEN
            DO;
               IR2=M-1;
               IR1=N-1;
               KK=1;
               DO WHILE (KK=1);
                 IR0=MOD(IR2,IR1);
                 IR2=IR1;
                 IR1=IR0;
                 IF IR0 = 0  THEN  KK=0;
               END;
               NCOUNT=NCOUNT+IR2-1;
            END;
          I=1;
          IM=M;
LP1:      DO WHILE ('1'B);
             I1=I;
             KMI=K-I;
             B=A(I1+1);
```

```
           I1C=KMI;
           C=A(I1C+1);
           KK1=1;
LP2:       DO WHILE (KK1=1);
              I2=M*I1-K*(I1/N);
              I2C=K-I2;
              IF I1 <= IWRK  THEN  MOVE(I1)=2;
              IF I1C <= IWRK  THEN  MOVE(I1C)=2;
              NCOUNT=NCOUNT+2;
              IF I2 = I  THEN  KK1=0;
              ELSE
                  IF I2 = KMI  THEN
                    DO;
                       D=B;
                       B=C;
                       C=D;
                       KK1=0;
                    END;
                  ELSE
                      DO;
                         A(I1+1)=A(I2+1);
                         A(I1C+1)=A(I2C+1);
                         I1=I2;
                         I1C=I2C;
                      END;
           END LP2;
           A(I1+1)=B;
           A(I1C+1)=C;
           IF NCOUNT < MN  THEN
LP3:         DO;
                KK2=1;
LP4:            DO WHILE (KK2=1);
                   MAXX=K-I;
                   I=I+1;
                   IF I > MAXX  THEN
                     DO;
                        IOK=I;
                        RETURN;
                     END;
                   IM=IM+M;
                   IF IM > K  THEN  IM=IM-K;
                   I2=IM;
                   IF I = I2  THEN  KK2=1;
                   ELSE
                       IF I > IWRK  THEN
                         DO WHILE (I2 <=  I | I2 >= MAXX);
                            I1=I2;
                            I2=M*I1-K*(I1/N);
                         END;
                       ELSE
                           IF MOVE(I) ¬= 0  THEN  KK2=1;
                END LP4;
             END LP3;
           ELSE
               DO;
                  IOK=0;
                  RETURN;
               END;
        END LP1;
     END TRANS;
```

```
COMB:  PROC(N,P,L,C);
          DCL (N,P,L,C(*),K,R,P1) FIXED BINARY(31);
       K=0;
       P1=P-1;
       DO I=1 TO P1;
          C(I)=0;
          IF I ¬= 1  THEN C(I)=C(I-1);
          KK=1;
          DO WHILE (KK=1);
             C(I) = C(I) + 1;
             R=BINOM(N-C(I),P-I);
             K=K+R;
             IF K < L  THEN  KK=1;
             ELSE
                DO;
                   K=K-R;
                   KK=0;
                END;
          END;
       END;
       C(P)=C(P1)+L-K;
     END COMB;

BINOM: PROC(M,N) RETURNS (FIXED BINARY(31));
          DCL (M,N,P,I,N1,R) FIXED BINARY(31);
       N1=N;
       P=M-N1;
       IF N1 < P  THEN
         DO;
            P=N1;
            N1=M-P;
         END;
       R=N1+1;
       IF P = 0  THEN  R=1;
       IF P >= 2  THEN
         DO I=2 TO P;
            R=(R*(N1+I))/I;
         END;
       RETURN(R);
     END BINOM;
```

```
VMISS:   PROC(T,VK,VMISES);
            DCL  (PI INIT(3.1415926535898),TPI INIT(6.2831853071760))
                 FLOAT DECIMAL(16);
            DCL  (A1 INIT(12.0),A2 INIT(0.8),A3 INIT(8.0),A4 INIT(1.0),
                  CK INIT(10.5),C1 INIT(56.0)) FLOAT DECIMAL(8);
            DCL  (VMISES,Z,S,Y) FLOAT DECIMAL(8);
          Z=VK;
          U=MOD(T+PI,TPI);
          IF U < 0.0   THEN  U=U+TPI;
          Y=U-PI;
          IF Z <= CK   THEN
            DO;
               V=0.0;
               IF Z > 0.0   THEN
                 DO;
                    IP=Z*AZ-A2-A3/(Z+A4)+A1;
                    P=IP;
                    S=SIN(Y);
                    C=COS(Y);
                    Y=P*Y;
                    SN=SIN(Y);
                    CN=COS(Y);
                    R=0.0;
                    Z=2.0/Z;
                    DO N=2 TO IP;
                      P=P-1.0;
                      Y=SN;
                      SN=SN*C-CN*S;
                      CN=CN*C+Y*S;
                      R=1.0/(P*Z+R);
                      V=(SN/P+V)*R;
                    END;
                 END;
                 VMISES=(U*0.5+V)/PI;
                 IF VMISES < 0.0   THEN   VMISES=0.0;
                 IF VMISES > 1.0   THEN   VMISES=1.0;
                 RETURN;
            END;
          C=24.0*Z;
          V=C-C1;
          R=SQRT((54.0/(347.0/V+26.0-C)-6.0+C)/6.0);
          Z=SIN(Y*0.5)*R;
          S=Z*Z;
          V=V-S+3.0;
          Y=(C-S-S-16.0)/3.0;
          Y=((S+1.75)*S+83.5)/V-Y;
          VMISES=GAUSS(Z-S/(Y*Y)*Z);
          IF VMISES < 0.0   THEN   VMISES=0.0;
          IF VMISES > 1.0   THEN   VMISES=1.0;
         END VMISS;
```

```
INERFC:  PROC(X,NMAX,ACC,FZERO,F,IFLAG);
            DCL (F(*),R0(500),R1(500)) FLOAT DECIMAL(8);
            DCL (DEPS,DC,DX,DXSQ,DFM1,DF0,DF1,DN,DN1,DTE,
                 DTERN,DSUM) FLOAT DECIMAL(16);
            DCL FRSTTM BIT(1) INIT('1'B);
            DCL (PREC INIT(28.8989),BOTEXP INIT(-293)) FLOAT DEC(8);
          IFLAG=0;
          IF NMAX < 1  THEN
            DO;
               IFLAG=1;
               RETURN;
            END;
          IF NMAX > 500  THEN
            DO;
               IFLAG=2;
               RETURN;
            END;
          TOL=PREC-ACC;
          IF TOL < 1  THEN
            DO;
               IFLAG=3;
               RETURN;
            END;
          I=1;
          EPS=.5*10.**(-ACC);
          XSQ=X*X;
          IF FRSTTM  THEN
            DO;
               P=ATAN(1.);
               AL10=LOG(10.);
               C=SQRT(1./P);
               DC=SQRT(1./ATAN(1.));
               FRSTTM='0'B;
            END;
          S=0.;
          IF -XSQ > AL10*BOTEX  THEN  S=EXP(-XSQ);
          B=.5*AL10*TOL+.25*LOG(2.*P);
          B1=.5*AL10*(TOL-1.);
          IF XSQ <= B  THEN
           DO;
             IF X >= 0  THEN
               DO;
                  F0=C;
                  IF S = 0.  THEN
                    DO;
                       IFLAG=4;
                       RETURN;
                    END;
                  T=1./S;
                  SQ=SQRT(2.*NMAX);
                  IF ¬((X >= 1.12 & XSQ+SQ*X > B) |
                      (X < 1.12 & SQ*X > B1))  THEN
                    DO;
                       IF X < 0.  THEN
                         DO;
                            F0=C*S;
                            T=1.;
               END;
             DEPS=.5*10.**(-PREC);
             DX=X;
             DXSQ=DX*DX;
             DTE=DC*DX;
             DSUM=1.-DTE;
             DN=0.0;
             DN1=1.0;
             KK=1;
             DO WHILE(KK=1);
                DN=DN+1.0;
                DN1=DN1+2.0;
                IF DN > 2.E2  THEN
                  DO;
                     IFLAG=5;
                     RETURN;
                  END;
                DTE=-DTE*DXSQ/DN;
                DTERM=DTE/DN1;
                DSUM=DSUM-DTERM;
                IF ABS(DTERM) <= DEPS*ABS(DSUM)  THEN KK=0;
             END;
             IF X >= 0.  THEN
               DO;
                  DF0=DC;
                  DF1=EXP(DXSQ)*DSUM;
                  FZERO=DF1;
                  DO N=1 TO NMAX;
                     DFM1=DF0;
                     DF0=DF1;
                     DF1=(-DX*DF0+.5*DFM1)/N;
                     F(N)=DF1;
                  END;
                  RETURN;
               END;
             F1=DSUM;
             FZERO=F1;
             DO N=1 TO NMAX;
                FM1=F0;
                F0=F1;
                F1=(-X*F0+.5*FM1)/N;
                F(N)=F1;
             END;
             RETURN;
          END;
        ELSE
            DO;
               NO=NMAX;
               X0=X;
            END;
     END;
 END;
ELSE
    IF X < 0.  THEN
       IF XSQ >= (ACC+1.)*AL10-.572  THEN
         DO;
            F0=C*S;
            F1=2.;
            FZERO=F1;
            DO N=1 TO NMAX;
               FM1=F0;
```

```
                              F0=F1;
                              F1=(-X*F0+.5*FM1)/N;
                              F(N)=F1;
                           END;
                           RETURN;
                        END;
                     ELSE
                         DO;
                            I=2;
                            N0=1;
                            X0=-X;
                         END;
                  ELSE
                      DO;
                         N0=NMAX;
                         X0=X;
                      END;
              DO N=1 TO N0;
                 R1(N)=0.;
              END;
              FN0=N0;
              NU=(SQRT(FN0)+(2.3026*ACC+1.3363)/(2.8284*X0))**2-5.;
              KK=1;
B1:           DO WHILE(KK=1);
                 NU=NU+I0;
                 IF NU > 5000  THEN
                   DO;
                      IFLAG=5;
                      RETURN;
                   END;
                 N0P1=N0+1;
                 NUM=NU-N0P1;
                 DO N=1 TO N0;
                    R0(N)=R1(N);
                 END;
                 R=0.;
                 DO K=1 TO NUM;
                    N=NU-K;
                    R=.5/(X0+(N+1)*R);
                 END;
                 DO K=1 TO N0;
                    N=N0P1-K;
                    R=.5/(X0+(N+1)*R);
                    R1(N)=R;
                 END;
                 DO N=1 TO N0;
                    IF ABS(R1(N)-R0(N)) > EPS*ABS(R1(N))  THEN
                      DO;
                         N=N0;
                         KK=0;
                      END;
                 END;
              END B1;
              F0=.5*C/(X0+R1(1));
              FZERO=F0;
              FF=F0;
              DO N=1 TO N0;
                 FF=R1(N)*FF;
                 F(N)=FF;
              END;
              IF I = 1  THEN  RETURN;
```

```
   IF I = 2  THEN
     DO;
        F1=2.-S*F0;
        F0=C*S;
        FZERO=F1;
      DO N=1 TO NMAX;
           FM1=F0;
           F0=F1;
           F1=(-X*F0+.5*FM1)/N;
           F(N)=F1;
        END;
        RETURN;
     END;
END INERFC;
```

```
MPROG: PROC OPTIONS(MAIN);
          DCL (XX(2,25),IN(25),IH(25),X(25),Y(25)) FLOAT DEC(8);
          DCL (IWORK(50) INIT((50) (0)),IL(50))FIXED BINARY(31);
        NCOUNT=0;
        GET EDIT(N)(F(5));
        PUT SKIP EDIT(N)(F(5));
        N1=N+1;
        DO I=1 TO N;
           J=N1-I;
           IN(J)=I;
        END;
        DO I=1 TO N;
           GET EDIT(XX(1,I),XX(2,I))(2 F(10,5));
        END;
        DO I=1 TO N;
           J=IN(I);
           PUT SKIP EDIT(XX(1,J),XX(2,J))(2 F(10,5));
        END;
        DO M=4 TO N;
           CALL CONVEX(N,XX,M,IN,IWORK,IWORK(N+1),IH,NHULL,IL);
           IK=IL(1);
           DO I=1 TO NHULL;
              J=IH(IK);
              X(I)=XX(1,J);
              Y(I)=XX(2,J);
              IK=IL(IK);
           END;
           PUT SKIP EDIT('SAMPLE SIZE  ',M,' VERTICES ',NHULL,
                        'SPLIT ',NCOUNT)
                       (A,F(5),A,F(5),A,F(5));
           DO I=1 TO NHULL;
              PUT SKIP EDIT(X(I),Y(I))(X(1),2 F(10,5));
           END;
        END;
      END MPROG;


SPLIT: PROC(N,X,M,IN,II,JJ,S,IABV,NA,MAXA,IBEL,NB,MAXB);
          DCL X(*,*) FLOAT DECIMAL(8);
          DCL (IN(*),IABV(*),IBEL(*),S)FIXED BINARY(31);
          DCL T BIT(1);
        T='0'B;
        IF X(1,JJ) ¬= X(1,II)  THEN
          DO;
             A=(X(2,JJ)-X(2,II))/(X(1,JJ)-X(1,II));
             B=X(2,II)-A*X(1,II);
          END;
        ELSE
            DO;
               XT=X(1,II);
               DIR=SIGN(1.,X(2,JJ)-X(2,II))*SIGN(1.,S);
               T='1'B;
            END;
        UP=0.;
        NA=0;
        MAXA=0;
        DOWN=0.;
```

```
         NB=0;
         MAXB=0;
         DO I=1 TO M;
            IS=IN(I);
            IF ¬T   THEN Z=X(2,IS)-A*X(1,IS)-B;
            ELSE   Z=DIR*(X(1,IS)-XT);
            IF Z > 0.   THEN
              DO;
               IF S ¬= -2   THEN
                 DO;
                    NA=NA+1;
                    IABV(NA)=IS;
                    IF Z >= UP   THEN
                      DO;
                         UP=Z;
                         MAXA=NA;
                      END;
                 END;
              END;
            ELSE
                IF S ¬= 2   THEN
                   IF Z < 0.   THEN
                     DO;
                        NB=NB+1;
                        IBEL(N)=IS;
                        IF Z <= DOWN   THEN
                          DO;
                             DOWN=Z;
                             MAXB=NB;
                          END;
         END;
       END SPLIT;


CONVEX:  PROC(N,X,M,IN,IA,IB,IH,NH,IL);
           DCL X(*,*) FLOAT DECIMAL(8);
           DCL (IN(*),IA(*),IB(*),IH(*),IL(*))FIXED BINARY(31);
           DCL (MAXE,MINE) BIT(1);
           IF M = 1   THEN
             DO;
                NH=2;
                IH(1)=IN(1);
                IL(1)=1;
                NH=NH-1;
                RETURN;
             END;
           IL(1)=2;
           IL(2)=1;
           KN=IN(1);
           KX=IN(2);
           IF M = 2   THEN
             DO;
                IH(1)=KX;
                IH(2)=KN;
                NH=3;
                IF X(1,KN) = X(1,KX) & X(2,KN) = X(2,KX)   THEN
                   NH=2;
                NH=NH-1;
                RETURN;
             END;
           MP1=M+1;
```

```
MINN=1;
MX=1;
KX=IN(1);
MAXE='0'B;
MINE='0'B;
DO I=2 TO M;
    J=IN(I);
    IF X(1,J)-X(1,KX) = 0   THEN   MAXE='1'B;
    ELSE   IF X(1,J)-X(1,KX) > 0   THEN
             DO;
                 MAXE='0'B;
                 MX=I;
                 KX=J;
             END;
    IF X(1,J)-X(1,KN) = 0   THEN   MINE='1'B;
    ELSE   IF X(1,J)-X(1,KN) < 0   THEN
             DO;
                 MINE='0'B;
                 MINN=I;
                 KN=J;
             END;
END;
IF KX = KN   THEN
   DO;
      KX=IN(1);
      KN=IN(1);
      DO I=1 TO M;
          J=IN(I);
          IF X(2,J) > X(2,KX)   THEN
            DO;
                MX=I;
                KX=I;
            END;
          IF X(2,J) < X(2,KN)   THEN
            DO;
                MINN=I;
                KN=J;
            END;
      END;
      IF KX = KN   THEN
         DO;
            NH=2;
            IH(1)=IN(1);
            IL(1)=1;
            NH=NH-1;
            RETURN;
         END;
      IH(1)=KX;
      IH(2)=KN;
      NH=3;
      IF X(1,KN) = X(1,KX) & X(2,KN) = X(2,KX)   THEN
          NH=2;
      NH=NH-1;
      RETURN;
   END;
   IF MAXE | MINE   THEN
     DO;
         IF MAXE   THEN
           DO I=1 TO M;
               J=IN(I);
               IF X(1,J) = X(1,KX)   THEN
```

```
                              IF X(2,J) > X(2,KX)   THEN
                                DO;
                                   MX=I;
                                   KX=J;
                                END;
                       END;
                     IF MINE  THEN
                       DO I=1 TO M;
                          J=IN(I);
                        IF X(1,J) = X(1,KN)   THEN
                             IF X(2,J) < X(2,KN)   THEN
                               DO;
                                  MINN=I;
                                  KN=J;
                               END;
                       END;
                 END;
               IH(1)=KX;
               IH(2)=KN;
               NH=3;
               INH=1;
               NIB=1;
               MA=M;
               IN(MX)=IN(M);
               IN(M)=KX;
               MM=M-2;
               IF MINN = M   THEN   MINN=MX;
               IN(MINN)=IN(M-1);
               IN(M-1)=KN;
               CALL SPLIT(N,X,MM,IN,IH(1),IH(2),0,IA,MB,MXA,IB,IA(MA),
                          MXBB);
LB8:           NIB=NIB+IA(MA);
               MA=MA-1;
B9:            IF MXA = 0   THEN   GO TO  L11;
               IL(NH)=IL(INH);
               IL(INH)=NH;
               IH(NH)=IA(MXA);
               IA(MXA)=IA(MB);
               MB=MB-1;
               NH=NH+1;
               IF MB = 0   THEN   GO TO L10;
               ILINH=IL(INH);
               CALL SPLIT(N,X,MB,IA,IH(INH),IH(ILINH),1,IA,MBB,MXA,
                          IB(NIB),IA(MA),MXB);
               MB=MBB;
               GO TO LB8;
10:            INH=IL(INH);
11:            INH=IL(INH);
               MA=MA+1;
               NIB=NIB-IA(MA);
               IF MA >= M   THEN   GO TO L12;
               IF IA(MA) = 0   THEN   GO TO L11;
               ILINH=IL(INH);
               CALL SPLIT(N,X,IA(MA),IB(NIB),IH(INH),IH(ILINH),2,IA,
                          MB,MXA,IB(NIB),MBB,MXB);
               GO TO LB9;
12:            MXB=MXBB;
               MA=M;
               MB=IA(MA);
               NIA=1;
               IA(MA)=0;
```

```
L13:       NIA=NIA+IA(MA);
           MA=MA-1;
 14:       IF MXB = 0  THEN  GO TO L16;
           IL(NH)=IL(INH);
           IL(INH)=NH;
           IH(NH)=IB(MXB);
           IB(MXB)=IB(MB);
           MB=MB-1;
           NH=NH+1;
           IF MB = 0  THEN  GO TO L15;
           ILINH=IL(INH);
           CALL SPLIT(N,X,MB,IB(NIB),IH(INH),IH(ILINH),-1,IA(NIA),
                      IA(MA),MXA,IB(NIB),MBB,MXB);
           MB=MBB;
           GO TO L13;
 15:       INH=IL(INH);
 16:       INH=IL(INH);
           MA=MA+1;
           NIA=NIA-IA(MA);
           IF MA = MP1  THEN
             DO;
                NH=NH-1;
                RETURN;
             END;
           IF IA(MA) = 0  THEN  GO TO L16;
           ILINH=IL(INH);
           CALL SPLIT(N,X,IA(MA),IA(NIA),IH(INH),IH(ILINH),-2,
                      IA(NIA),MBB,MXA,IB(NIB),MB,MXB);
           GO TO L14;
       END CONVEX;
```

```
MPROG: PROC OPTIONS(MAIN);
       DCL (IP(50),ICN(1000),IOR(50),IB(51),IW(50,50),LENR(50))
           FIXED BINARY(31);
       DCL HOLD(100) CHAR(1);
       DCL (EX INIT('X'),NOT INIT('O'),BLANK INIT(' '))CHAR(1);
       DCL A(50,50) BIT(1);
         MM=50;
         LICN=1000;
         DO WHILE('1'B);
            GET EDIT(N,IPP)(2 F(4));
            IF N = 0  THEN STOP;
            PUT SKIP EDIT(' MATRIX IS OF ORDER ',N,' AND HAS ',IPP,
                          ' OFF-DIAGONAL NON-ZEROS')(A,F(3),A,F(3),A);
            DO J=1 TO N;
               DO I=1 TO N;
                  A(I,J)='0'B;
               END;
               A(J,J)='1'B;
            END;
            IF IPP ¬= 0  THEN
              DO K9=1 TO IPP;
                 KK=1;
                 DO WHILE(KK=1);
                    CALL FA01BS(N,I);
                    CALL FA01BS(N,J);
                    IF ¬A(I,J)  THEN  KK=0;
                 END;
              END;
            CALL SETUP(N,A,MM,IP,ICN,LICN,LENR);
            CALL MC13D(N,ICN,LICN,IP,LENR,IOR,IB,NUM,IW);
            IF NUM = 1  THEN  PUT SKIP(3) EDIT(' THE REORDERED',
                                 ' MATRIX WHICH HAS ',NUM,
                                 ' BLOCK IS OF THE FORM')
                                 (A,A,F(3),A);
            IF NUM ¬= 1  THEN PUT SKIP(3) EDIT(' THE REORDERED',
                                 ' MATRIX WHICH HAS ',NUM,
                                 ' BLOCK IS OF THE FORM')
                                 (A,A,F(3),A);
            IB(NUM+1)=N+1;
            INDEX=100;
            IBLOCK=1;
            DO I=1 TO N;
               DO IJ=1 TO INDEX;
                  HOLD(IJ)=BLANK;
               END;
               IF I = IB(IBLOCK)  THEN  PUT SKIP EDIT(' ')(A);
               IF I = IB(IBLOCK)  THEN  IBLOCK=IBLOCK+1;
               JBLOCK=1;
               INDEX=0;
               DO J=1 TO N;
                  IF J = IB(JBLOCK)  THEN  INDEX=INDEX+1;
                  IF J = IB(JBLOCK)  THEN  HOLD(INDEX)=BLANK;
                  IF J = IB(JBLOCK)  THEN  JBLOCK=JBLOCK+1;
                  INDEX=INDEX+1;
                  II=IOR(I);
                  JJ=IOR(J);
                  IF A(II,JJ)  THEN HOLD(INDEX)=EX;
```

```
                IF ¬A(II,JJ)   THEN   HOLD(INDEX)=NOT;
             END;
             PUT SKIP EDIT(HOLD)(X(2),100 A(1));
          END;
          PUT SKIP EDIT(' THE STARTING POINT FOR EACH BLOCK IS',
                        ' GIVEN BY ',(IB(I) DO I=1 TO NUM))
                        (A,A,SKIP(2),COLUMN(46),20 (X(2),F(4)));
       END;
     END MPROG;

SETUP: PROC(N,A,MM,IP,ICN,LICN,LENR);
       DCL (IP(*),ICN(*),LENR(*)) FIXED BINARY(31);
       DCL A(*,*) BIT(1);
       DO I=1 TO N;
          LENR(I)=0;
       END;
       IND=1;
       DO I=1 TO N;
          IP(I)=IND;
          DO J=1 TO N;
          IF A(I,J)   THEN
             DO;
                LENR(I)=LENR(I)+1;
                ICN(IND)=J;
                IND=IND+1;
             END;
          END;
       END;
     END SETUP;

MC13D: PROC(N,ICN,LICN,IP,LENR,IOR,IB,NUM,IW);
       DCL (ICN(*),LENR(*),IOR(*),IB(*),IW(*,*),IP(*))
           FIXED BINARY(31);
       CALL MC13E(N,ICN,LICN,IP,LENR,IOR,IB,NUM,IW(1,1),
                  IW(1,2),IW(1,3));
     END MC13D;

C13E: PROC(N,ICN,LICN,IP,LENR,ARP,IB,NUM,LOWL,NUMB,PREV);
       DCL (STP,DUMMY,IP(*)) FIXED BINARY(31);
       DCL (ICN(*),LENR(*),ARP(*),IB(*),LOWL(*),NUMB(*),
           PREV(*)) FIXED BINARY(31);
       ICNT=0;
       NUM=0;
       NUM1=N+N-1;
       DO J=1 TO N;
          NUM(J)=0;
          ARP(J)=LENR(J)-1;
       END;
       DO ISN=1 TO N;
          IF NUMB(ISN) = 0   THEN
            DO;
               IV=ISN;
               IST=1;
               LOWL(IV)=1;
               NUMB(IV)=1;
               IB(N)=IV;
               DO DUMMY=1 TO NUM1;
                  I1=ARP(IV);
```

```
IF I1 >= 0  THEN
  DO;
     I2=IP(IV)+LENR(IV)-1;
     I1=I2-I1;
     DO II=I1 TO I2;
        IW=ICN(II);
        IF NUMB(IW) = 0  THEN
          DO;
             ARP(IV)=I2-II-1;
             PREV(IW)=IV;
             IV=IW;
             IST=IST+1;
             LOWL(IV)=IST;
             NUMB(IV)=IST;
             K=N+1-IST;
             IB(K)=IV;
             GO TO OUT1;
          END;
        IF LOWL(IW) < LOWL(IV)  THEN
           LOWL(IV)=LOWL(IW);
     END;
     ARP(IV)=-1;
  END;
IF LOWL(IV) < NUMB(IV)  THEN
  DO;
     IW=IV;
     IV=PREV(IV);
     IF LOWL(IW) < LOWL(IV)  THEN
        LOWL(IV)=LOWL(IW);
  END;
ELSE
    DO;
       NUM=NUM+1;
       IST1=N+1-IST;
       LCNT=ICNT+1;
       DO STP=IST1  TO N;
          IW=IB(STP);
          LOWL(IW)=N+1;
          ICNT=INCT+1;
          NUMB(IW)=ICNT;
          IF IW = IV  THEN
            DO;
               STP1=STP;
               STP=N;
            END;
       END;
       STP=STP1;
       IST=N-STP;
       IB(NUM)=LCNT;
       IF IST ¬= 0  THEN
         DO;
            IW=IV;
            IV=PREV(IV);
            IF LOWL(IW) < LOWL(IV)  THEN
               LOWL(IV)=LOWL(IW);
         END;
       ELSE
           IF ICNT < N  THEN  DUMMY=NUM;
           ELSE
               DO;
                  DO I=1 TO N;
```

```
                                                                                II=NUMB(I);
                                                                                ARP(II)=I;
                                                                          END;
                                                                          RETURN;
                                                                    END;
                                                 END;
UT1:                                END;
                              END;
                    END;
                END MC13E;
```

```
/*                     ALGORITHM 533                          */

PIVCHK:  PROC OPTIONS(MAIN);
           DCL (IA(101),JA(400),R(100),C(100),IC(100),ITEMP(597))
                FIXED BINARY(31);
           DCL (A(400),B(100),X(100),RTEMP(495))FLOAT DECIMAL(8);
           DCL (MAX INIT(395),NG INIT(10),N INIT(100))
                FIXED BINARY(31);
           K=1;
           IA(1)=1;
           IAPTR=1;
           DO I=1 TO NG;
              DO J=1 TO NG;
                 BK=0.;
                 IF I ¬= 1  THEN
                   DO;
                      JA(IAPTR)=K-NG;
                      A(IAPTR)=-1.;
                      BK=BK-1.;
                      IAPTR=IAPTR+1;
                    END;
                  IF J ¬= 1  THEN
                    DO;
                       JA(IAPTR)=K-1;
                       A(IAPTR)=-1.;
                       BK=BK-1.;
                       IAPTR=IAPTR+1;
                    END;
                  JA(IAPTR)=K;
                  A(IAPTR)=4.;
                  BK=BK+4.;
                  IAPTR=IAPTR+1;
                  IF I ¬= NG  THEN
                    DO;
                       JA(IAPTR)=K+NG;
                       A(IAPTR)=-1.5;
                       BK=BK-1.5;
                       IAPTR=IAPTR+1;
                    END;
                  B(K)=BK;
                  K=K+1;
                  IA(K)=IAPTR;
               END;
            END;
            CALL PREORD(N,IA,R,C,IC);
            CALL NSPIV(N,IA,JA,A,B,MAX,R,C,IC,X,ITEMP,RTEMP,IERR);
            PUT SKIP EDIT(' IERR = ',IERR)(A,F(10));
            CALL RESCHK(N,IA,JA,A,B,X);
          END PIVCHK;


RESCHK:  PROC(N,IA,JA,A,B,X);
            DCL (IA(*),JA(*)) FIXED BINARY(31);
            DCL (A(*),B(*),X(*)) FLOAT DECIMAL(8);
            DCL (RESID,RESIDM,ROWSUM) FLOAT DECIMAL(16);
            RESID=0.;
            RESIDM=0.;
            DO I=1 TO N;
```

```
              ROWSUM=B(I);
              JMIN=IA(I);
              JMAX=IA(I+1)-1;
              DO J=JMIN TO JMAX;
                 JAJ=JA(J);
                 ROWSUM=ROWSUM-A(J)*X(JAJ);
              END;
              IF ABS(ROWSUM) > RESIDM   THEN   RESIDM=ABS(ROWSUM);
              RESID=RESID+ROWSUM**2;
           END;
           RESID=SQRT(RESID);
           PUT SKIP EDIT(' 2-NORM OF RESIDUAL = ',RESID)(A,F(14,7));
           PUT SKIP EDIT(' MAX NORM OF RESIDUAL = ',RESIDM)(A,F(14,7));
         END RESCHK;


PREORD:  PROC(N,IA,R,C,IC);
           DCL (IA(*),R(*),C(*),IC(*)) FIXED BINARY(31);
           DO I=1 TO N;
              R(I)=I;
              C(I)=I;
              IC(I)=I;
           END;
           DO I=1 TO N;
              C(I)=0;
           END;
           DO K=1 TO N;
              KDEG=IA(K+1)-IA(K);
              IF KDEG = 0   THEN   KDEG=KDEG+1;
              IC(K)=C(KDEG);
              C(KDEG)=K;
           END;
           I=0;
           DO J=1 TO N;
              IF C(J) ¬= 0   THEN
                 DO;
                    K=C(J);
                    KK1=1;
                    DO WHILE(KK1=1);
                       I=I+1;
                       R(I)=K;
                       K=IC(K);
                       IF K <= 0   THEN   KK1=0;
                    END;
                 END;
           END;
           DO I=1 TO N;
              C(I)=I;
              IC(I)=I;
           END;
         END PREORD;


NSPIV:   PROC(N,IA,JA,A,B,MAX,R,C,IC,X,ITEMP,RTEMP,IERR);
           DCL (A(*),B(*),X(*),RTEMP(*))FLOAT DECIMAL(8);
           DCL (IA(*),JA(*),R(*),C(*),IC(*),ITEMP(*),IJ,
                 JU,U,Y,P) FIXED BINARY(31);
           Y=1;
           U=Y+N;
```

```
          P=1;
          IU=P+N+1;
          JU=IU+N+1;
          CALL NSPIV1(N,IA,JA,A,B,MAX,R,C,IC,X,RTEMP(Y),ITEMP(P),
                      ITEMP(IU),ITEMP(JU),RTEMP(U),IERR);
        END NSPIV;

SPIV1:  PROC(N,IA,JA,A,B,MAX,R,C,IC,X,Y,P,IU,JU,U,IERR);
          DCL (A(*),B(*),U(*),X(*),Y(*),DK,LKI,ONE,XPV,
               XPVMAX,YK,ZERO) FLOAT DECIMAL(8);
          DCL (C(*),IA(*),IC(*),IU(*),JA(*),JU(*),P(*),R(*),CK,
               PK,PPK,PV,V,VI,VJ,VK) FIXED BINARY(31);
          IF N = 0  THEN
            DO;
               IERR=0;
               RETURN;
            END;
          ONE=1.0;
          ZERO=0.0;
          DO J=1 TO N;
             X(J)=ZERO;
          END;
          IU(1)=1;
          JUPTR=0;
P1:       DO K=1 TO N;
             P(N+1)=N+1;
             VK=R(K);
             JMIN=IA(VK);
             JMAX=IA(VK+1)-1;
             IF JMIN > JMAX  THEN
               DO;
                  IERR=-K;
                  RETURN;
               END;
             J=JMAX;
             KK=1;
LP2:         DO WHILE(KK=1);
                JAJ=JA(J);
                VJ=IC(JAJ);
                X(VJ)=A(J);
                PPK=N+1;
                KK1=1;
P3:             DO WHILE(KK1=1);
                   PK=PPK;
                   PPK=P(PK);
                   IF PPK-VJ = 0  THEN
                     DO;
                        IERR=-(N+K);
                        RETURN;
                     END;
                   IF PPK-VJ > 0  THEN  KK1=0;
                END LP3;
                P(VJ)=PPK;
                P(PK)=VJ;
                J=J-1;
                IF J < JMIN  THEN  KK=0;
             END LP2;
             VI=N+1;
             YK=B(VK);
             KK2=1;
```

```
LP4:        DO WHILE(KK2=1);
               VI=P(VI);
               IF VI >= K  THEN GO TO RT1;
               LKI=-X(VI);
               X(VI)=ZERO;
               YK=YK+LKI*Y(VI);
               PPK=VI;
               JMIN=IU(VI);
               JMAX=IU(VI+1)-1;
               IF JMIN > JMAX  THEN  KK2=1;
               ELSE
                    DO J=JMIN TO JMAX;
                       JUJ=JU(J);
                       VJ=IC(JUJ);
                       IF X(VJ) = ZERO  THEN
                         DO;
                            PPKK=PPK;
                            IF VJ-PPKK < 0  THEN  PPK=VI;
                            IF VJ-PPKK > 0 | VJ-PPKK < 0  THEN
                              DO;
                                 KK3=1;
                                 DO WHILE(KK3=1);
                                    PK=PPK;
                                    PPK=P(PK);
                                    IF PPK-VJ >= 0  THEN  KK3=0;
                                 END;
                                 IF PPK-VJ > 0  THEN
                                   DO;
                                      P(VJ)=PPK;
                                      P(PK)=VJ;
                                      PPK=VJ;
                                   END;
                              END;
                         END;
                      X(VJ)=X(VJ)+LKI*U(J);
                    END;
            END LP4;
RT1:        IF VI > N  THEN
              DO;
                 IERR=-(2*N+K);
                 RETURN;
              END;
            XPVMAX=ABS(X(VI));
            MAXC=VI;
            NZCNT=0;
            PV=VI;
            DO WHILE('1'B);
               V=PV;
               PV=P(PV);
               IF PV > N  THEN  GO TO RT2;
               NZCNT=NZCNT+1;
               XPV=ABS(X(PV));
               IF XPV > XPVMAX  THEN
                 DO;
                    XPVMAX=XPV;
                    MAXC=PV;
                    MAXCL=V;
                 END;
            END;
RT2:        IF XPVMAX = ZERO  THEN
              DO;
```

```
                IERR=-(2*N+K);
                RETURN;
             END;
          IF VI = K   THEN   VI=P(VI);
          ELSE
               IF VI = MAXC   THEN   VI=P(VI);
               ELSE   P(MAXCL)=P(MAXC);
          DK=ONE/X(MAXC);
          X(MAXC)=X(K);
          I=C(K);
          C(K)=C(MAXC);
          C(MAXC)=I;
          CK=C(K);
          IC(CK)=K;
          IC(I)=MAXC;
          X(K)=ZERO;
          Y(K)=YK*DK;
          IU(K+1)=IU(K)+NZCNT;
          IF IU(K+1) > MAX+1   THEN
             DO;
                IERR=-(3*N+K);
                RETURN;
             END;
          IF VI <= N   THEN
             DO;
                J=VI;
                KK4=1;
                DO WHILE(KK4=1);
                   JUPTR=JUPTR+1;
                   JU(JUPTR)=C(J);
                   U(JUPTR)=X(J)*DK;
                   X(J)=ZERO;
                   J=P(J);
                   IF J > N   THEN   KK4=0;
                END;
             END;
       END LP1;
       K=N;
       DO I=1 TO N;
          YK=Y(K);
          JMIN=IU(K);
          JMAX=IU(K+1)-1;
          IF JMIX <= JMAX   THEN
             DO J=JMIN TO JMAX;
                JUJ=JU(J);
                JUJ=IC(JUJ);
                YK=YK-U(J)*Y(JUJ);
             END;
          Y(K)=YK;
          CK=C(K);
          X(CK)=YK;
          K=K-1;
       END;
       IERR=IU(N+1)-IU(1);
    END NSPIV1;
```

APPENDIX C

FORTRAN MEASURABLE PROPERTIES

ALGORITHM 365

COMPLEX ROOT FINDING

OPERATORS			OPERANDS		
1	Subscript	27	1	U	8
2	=	47	2	1	11
3	-	11	3	2	8
4	FUNC()	4	4	3	8
5	ABS()	8	5	4	1
6	REAL()	4	6	5	1
7	AIMAG()	4	7	6	1
8	IF()	9	8	7	2
9	GO TO 1	6	9	1.	3
10	GO TO 1	1	10	0.	3
11	GO TO 5	2	11	0.8660254	2
12	GO TO 6	1	12	0.5000000	2
13	GO TO 7	1	13	0.9659258	3
14	GO TO 8	4	14	0.2588190	3
15	GO TO 9	1	15	0.7071068	4
16	GO TO 10	2	16	H	14
17	GO TO 11	1	17	HS	1
18	GO TO 12	2	18	ZO	8
19	GO TO 13	1	19	ZS	1
20	GO TO 14	1	20	N	3
21	GO TO 15	1	21	O	4
22	GO TO 4	1	22	CW	12
23	GO TO 3	2	23	WO	7
24	GO TO 2	2	24	DS	1
25	GO TO 16	2	25	DM	2
26	GO TO 17	1	26	K	5
27	+	7	27	I	7
28	*	8	28	V	6
29	CONJG	1	29	A	4
30	,	1	30	Z	8
31	/	1	31	W	11
32	.LT.	2	32	NR	7
33	END[1]	1	33	HM	2
34	END[2]	63	34	0.25	2
		230	35	4.	1
			36	ZE	1
			37	HE	1
			38	DE	1
					169

$n_1 = 34$

$N_1 = 230$

$n_2 = 38$

$N_2 = 169$

[1]End of program or procedure

[2]End of statement

ALGORITHM 386

GREATEST COMMON DIVISOR OF N INTEGERS AND MULTIPLIERS

OPERATORS

1	DO	4
2	=	39
3	,	4
4	IF()	8
5	.NE.	3
6	GO TO 3	1
7	GO TO 4	1
8	GO TO 5	1
9	GO TO 7	1
10	GO TO 25	1
11	GO TO 20	1
12	GO TO 15	1
13	GO TO 10	1
14	GO TO 60	1
15	GO TO 100	1
16	GO TO 40	1
17	+	5
18	.GE.	1
19	-	8
20	IABS()	1
21	/	3
22	*	7
23	.EQ.	4
24	()	1
25	EOS	50
26	Subscript	29
27	END	1
		179

$n_1 = 27$

$N_1 = 179$

OPERANDS

1	M	18
2	N	4
3	1	3
4	A	19
5	0	12
6	Z	10
7	IGDC	3
8	MP1	3
9	2	2
10	MP2	2
11	ISIGN	3
12	C1	11
13	I	11
14	Y1	7
15	Y2	5
16	C2	8
17	Q	6
18	J	4
19	K	6
20	KK	3
21	IP1	2
		148

$n_2 = 21$

$N_2 = 148$

ALGORITHM 424

QUADRATURE (D-1)

OPERATORS

1	=	155
2	( )	14
3	+	74
4	*	59
5	-	48
6	MINO	1
7	,	31
8	**	4
9	DO	18
10	Subscript	110
11	GO TO 200	1
12	GO TO 150	1
13	GO TO 210	1
14	GO TO 400	1
15	GO TO 100	1
16	/	11
17	FLOAT()	9
18	COS()	3
19	SIN()	3
20	CALL R3PASS	1
21	IF()	3
22	.LT.	1
23	ABS	2
24	.GE.	1
25	.LE.	1
26	END	2
27	EOS	161
		717

$n_1 = 27$

$N_1 = 717$

ALGORITHM 424

QUADRATURE (D-1)

OPERANDS

1	CENTRE	10	36	2	30	71	M,STEP	12
2	A	3	37	3	14	72	N3	5
3	B	3	38	1	41	73	NLESS	1
4	WIDTH	7	39	6	5	74	N_LESS_1	2
5	MAX	2	40	7	4	75	C	3
6	LIMIT	1	41	5	4	76	S	3
7	M_MAX	5	42	4	6	77	OLDINT	3
8	J(a)	26	43	2.E0	9	78	N_LESS_3	2
9	L	5	44	3.0E0	2	79	NEWINIT	9
10	N	16	45	K0	10	80	ESTERR	4
11	CSXFRM	63	46	K1	9	81	TOLERR	1
12	F	10	47	R0	3	82	CCQUAD	1
13	SHIFT	12	48	I0	3	83	HALF_M	2
14	RT3	3	49	R1	3	84	M3	5
15	T1	19	50	IDIFF2	3	85	FUND(b)	2
16	T2	18	51	IDIFF	3	86	TWOPI	1
17	T3	7	52	ISUM	3	87	K	20
18	T4	6	53	I1	3	88	RSUM	8
19	T5	6	54	R2	3	89	X1	12
20	T6	6	55	I2	3	90	X2	12
21	T7	3	56	RSUM2	3	91	RDIFF	8
22	T8	3	57	J6	2	92	HAFRT3	4
23	T9	3	58	J7	2	93	X0	10
24	T10	3	59	J8	1	94	J(b)	6
25	T11	3	60	L1	2	95	J0	3
26	T12	3	61	L2	2	96	J1	2
27	USED	9	62	L3	2	97	ANGLE(b)	3
28	FUND(a)	3	63	L4	2	98	C1	5
29	PI	1	64	L5	2	99	S1	5
30	J1	11	65	L6	2	100	C2	3
31	J2	11	66	L7	2	101	S2	3
32	J3	2	67	L8	1			662
33	J4	2	68	ANGLE(a)	6			
34	J5	2	69	JREV	5			
35	.5E0	10	70	N2	12			

$n_2 = 101$

$N_2 = 662$

ALGORITHM 500

MINIMIZATION OF UNCONSTRAINED MULTIVARIATE FUNCTIONS

OPERATORS

1	CALL CALCFG	3	27	GO TO 390	1
2	=	134	28	GO TO 320	1
3	*	57	29	GO TO 330	1
4	DO	24	30	GO TO 380	3
5	,	24	31	GO TO 360	1
6	+	38	32	GO TO 370	1
7	Subscript	63	33	GO TO 410	1
8	IF()	33	34	GO TO 420	1
9	.EQ.	7	35	GO TO 440	1
10	GO TO 20	1	36	GO TO 460	1
11	GO TO 50	1	37	GO TO 60	1
12	GO TO 70	1	38	DSQRT()	3
13	GO TO 90	1	39	/	14
14	GO TO 120	1	40	.GT.	24
15	GO TO 140	1	41	-	24
16	GO TO 180	1	42	.GE.	1
17	GO TO 190	1	43	.AND.	9
18	GO TO 200	1	44	( )	39
19	GO TO 210	1	45	.LE.	4
20	GO TO 220	1	46	.LT.	7
21	GO TO 230	1	47	DABS	5
22	GO TO 260	1	48	.OR.	2
23	GO TO 270	1	49	.NOT.	1
24	GO TO 290	2	50	FLOAT()	1
25	GO TO 240	1	51	END	1
26	GO TO 300	2	52	EOS	170
					717

$n_1 = 52$

$N_1 = 717$

ALGORITHM 500

MINIMIZATION OF UNCONSTRAINED MULTIVARIATE FUNCTIONS

OPERANDS

1	N	28	36	0	3
2	X	7	37	0.	22
3	F	16	38	100.	1
4	G	15	39	2	5
5	IFUN	7	40	3	1
6	ITER	5	41	1.0D30	1
7	IERR	6	42	1.0D-30	1
8	EPS2	5	43	.9	1
9	EPS	2	44	1.	4
10	DUM	29	45	2.	3
11	STEP	15	46	.8	3
12	I	75	47	4	1
13	STP	3	48	.0001	2
14	IH	1	49	0.5	2
15	IJ	24	50	10.0	2
16	J	16	51	1.001	1
17	H	8	52	3.0	3
18	SUM	4	53	1.0D05	1
19	II	4	54	0.001	1
20	D	20	55	.1	1
21	XX	7	56	.999	1
22	GG	9	57	100000	1
23	G1	19	58	.TRUE.	1
24	STPMAX	7	59	.FALSE.	1
25	ALPHA	9	60	AA	5
26	FG	5	61	BB	5
27	AL	36	62	CC	3
28	MODE	4	63	FM	2
29	FEST	2	64	GM	2
30	F1	9	65	SS	6
31	MAXF	2	66	DY	6
32	G2	19	67	YHY	4
33	CONV	8	68	C1	4
34	BL	21	69	C2	2
35	1	40			589

$n_2 = 69$

$N_2 = 589$

ALGORITHM 502

DEPENDENCE OF SOLUTION OF NONLINEAR SYSTEMS ON A PARAMETER

OPERATORS

1	=	115	31	GO TO 120	1
2	+	21	32	GO TO 30	1
3	IF()	47	33	GO TO 40	1
4	GO TO 10	2	34	GO TO 50	1
5	GO TO 60	1	35	GO TO 60	1
6	GO TO 310	2	36	GO TO 70	2
7	GO TO 50	1	37	DO	25
8	GO TO 90	1	38	,	69
9	GO TO 130	2	39	CALL FCTN	2
10	GO TO 120	1	40	CALL GAUSE	2
11	GO TO 70	2	41	CALL ADAMS	1
12	GO TO 150	1	42	Subscript	111
13	GO TO 170	1	43	**	3
14	GO TO 200	1	44	-	23
15	GO TO 180	2	45	SQRT()	3
16	GO TO 210	1	46	.NE.	4
17	GO TO 300		47	WRITE	10
18	GO TO 240	1	48	( )	8
19	GO TO 290	1	49	.EQ.	15
20	GO TO 260	1	50	ABS()	12
21	GO TO 270	1	51	*	26
22	GO TO 280	4	52	.LE.	11
23	GO TO 60	2	53	.GE.	2
24	GO TO 30	1	54	.LT.	2
25	GO TO 50	2	55	.OR.	2
26	GO TO 40	1	56	.GT.	3
27	GO TO 70	1	57	/	8
28	GO TO 150	1	58	FLOAT()	2
29	GO TO 90	1	59	END	3
30	GO TO 20	1	60	EOS	166
					740

$n_1 = 60$

$N_1 = 740$

ALGORITHM 502

DEPENDENCE OF SOLUTION OF NONLINEAR SYSTEMS ON A PARAMETER

OPERANDS

1	N	16	42	MAXOUT	4
2	G,A	15	43	XLOW	1
3	F,B	17	44	XUPP	1
4	M	6	45	E	1
5	PREF	3	46	DXK2	4
6	BETA	6	47	NDIR	4
7	K	26	48	HH	1
8	DXDT	11	49	HMAX	2
9	MADMS	12	50	100	1
10	H	11	51	10	2
11	K1	3	52	11	2
12	MXADMS	3	53	1	59
13	N1 (a)	14	54	3	11
14	N1 (b)	6	55	0.0	13
15	N1 (c)	3	56	2	9
16	LW	1	57	0	15
17	INITA1	3	58	1.0	2
18	L	2	59	0.8	1
19	ITIN	3	60	4	3
20	X	30	61	0.5	1
21	SQUAR	15	62	3.0	1
22	I (a)	49	63	23.0	1
23	I (b)	28	64	16.0	1
24	I (c)	24	65	5.0	1
25	LL	2	66	12.0	1
26	NPRNT	5	67	55.0	1
27	P	19	68	59.0	1
28	J (a)	6	69	37.0	1
29	J (b)	8	70	9.0	1
30	J (c)	3	71	24.0	1
31	W	3	72	ID	4
32	EPS	2	73	IRK	3
33	KOUT	3	74	IRR	5
34	NOUT	11	75	IR	14
35	NC	5	76	IS	7
36	NCRAD	2	77	AMAX	4
37	NCORR	3	78	Y	2
38	MARK	3	79	DER	13
39	INDSP	1	80	D	1
40	INDIC	1			591
41	OUT	7			

$n_2 = 80$

$N_2 = 591$

ALGORITHM 505

A LIST INSERTION SORT FOR KEYS WITH ARBITRARY KEY DISTRIBUTION

OPERATORS

1	=	39
2	Subscript	16
3	IF()	7
4	GO TO 170	2
5	GO TO 10	1
6	GO TO 30	1
7	GO TO 20	2
8	GO TO 40	1
9	GO TO 60	2
10	GO TO 70	1
11	GO TO 50	1
12	GO TO 90	1
13	GO TO 130	4
14	GO TO 80	1
15	GO TO 160	1
16	GO TO 120	3
17	GO TO 100	1
18	GO TO 140	1
19	GO TO 110	1
20	GO TO 150	1
21	-	11
22	+	5
23	DO	2
24	,	3
25	END	1
26	EOS	50
		159

n_1 = 26

N_1 = 159

OPERANDS

1	MIN	7
2	II	9
3	MAX	6
4	L	9
5	JJ	2
6	KMIN	5
7	K	6
8	KMAX	3
9	IA	2
10	IRT	5
11	ITAB	3
12	TAB	2
13	ISTRT	4
14	J	13
15	KJ	6
16	MADIN	8
17	I	7
18	IPOI	7
19	KEY	4
20	KDIFF	4
21	1	8
22	0	1
		121

n_2 = 22

N_2 = 121

ALGORITHM 509

A HYBRID PROFILE REDUCTION ALGORITHM

OPERATORS

1	=	59	18	.LE.	3
2	DO	14	19	.GT.	3
3	,	18	20	.GE.	2
4	Subscript	35	21	END	4
5	IF()	12	22	EOS	78
6	.NE.	3	23	GO TO 10(b)	1
7	GO TO 10(a)	1	24	GO TO 50	1
8	GO TO 60	1	25	GO TO 20	1
9	GO TO 80	1	26	GO TO 30(b)	1
10	GO TO 40	1	27	GO TO 10(c)	1
11	GO TO 100	2	28	GO TO 30(c)	1
12	GO TO 120	1	29	GO TO 40(c)	1
13	GO TO 30(a)	1	30	.EQ.	2
14	+	14	31	.AND.	1
15	CALL FORMLV	2	32	WRITE	1
16	CALL DELETE	2	33	( )	1
17	MINCON()	1	34	-	3
					273

$n_1 = 34$

$N_1 = 273$

ALGORITHM 509

A HYBRID PROFILE REDUCTION ALGORITHM

OPERANDS

1	NSTPT	6	24	NSZ	2
2	I (a)	15	25	IQ	2
3	IDPTH	3	26	I(b)	5
4	LSTPT	4	27	LSTSZE	8
5	J (a)	3	28	IX	4
6	N	1	29	IROWOG	2
7	LVLSZ	1	30	J(b)	6
8	LVLST	6	31	K	2
9	LEVEL	8	32	SMLST	2
10	SZ, X	10	33	MINCON	2
11	SZSZE,XSZE	14	34	SETSZE (a)	8
12	S3, V	6	35	SET (a)	5
13	S3SZE, YSZE	7	36	ELEMNT	3
14	QPTR	10	37	I(c)	5
15	M	3	38	J(c)	3
16	CONECT, CONLST	3	39	LOWER	2
17	CONSZE	7	40	UPPER	2
18	NDSTK	2	41	I(d)	2
19	NR	1	42	SETSZE(b)	6
20	NDEG	2	43	SET (b)	1
21	NEW	3	44	0	9
22	NXT NOM	9	45	1	36
23	Q	4			246

$n_2 = 45$

$N_2 = 246$

ALGORITHM 512

SOLUTION OF POSITIVE DEFINITE SYMMETRIC QUINDIAGONAL

OPERATORS

1	=	70
2	-	57
3	Subscript	131
4	SQRT()	6
5	/	16
6	*	38
7	DO	12
8	,	12
9	+	16
10	( )	8
11	**	4
12	END	3
13	EOS	70
		443

n_1 = 13

N_1 = 443

OPERANDS

1	N	19
2	N1 (a)	14
3	N2 (a)	15
4	N3 (a)	7
5	N4	2
6	C	26
7	V (a)	15
8	B	17
9	I (a)	34
10	I1	4
11	I2	5
12	U	14
13	A	10
14	G	17
15	W	21
16	H	20
17	N1 (b)	12
18	N2 (b)	13
19	N3 (b)	5
20	Q	30
21	E	7
22	I (b)	21
23	F	5
24	U	4
25	V (b)	4
26	II	2
27	I (c)	4
28	S	2
29	1	32
30	2	22
31	3	6
32	4	1
33	0.0	5
34	1.0	2
		417

n_2 = 34

N_2 = 417

ALGORITHM 513

ANALYSIS OF IN-SITU TRANSPOSITION

OPERATORS

1	IF()	18
2	.LT.	5
3	.OR.	4
4	GO TO 120	2
5	GO TO 180	1
6	GO TO 190	1
7	GO TO 130	1
8	GO TO 30	1
9	GO TO 20	1
10	GO TO 80	2
11	GO TO 160	1
12	GO TO 40	4
13	GO TO 60	1
14	GO TO 70	1
15	GO TO 50	1
16	GO TO 110	1
17	GO TO 100	1
18	GO TO 90	1
19	GO TO 170	2
20	.NE.	3
21	*	7
22	.EQ.	5
23	=	51
24	-	15
25	DO	3
26	,	3
27	Subscript	16
28	MOD	1
29	+	15
30	.GT.	3
31	( )	4
32	/	2
33	.LE.	3
34	.GE.	1
35	END	1
36	EOS	73
		255

$n_1 = 36$

$N_1 = 255$

OPERANDS

1	M	10
2	N	10
3	MN	3
4	IWRK	5
5	NCOUNT	6
6	K	8
7	I	20
8	MOVE	2
9	IR2	4
10	IR1	4
11	IR0	3
12	IM	7
13	MAX	3
14	I2	16
15	I1	14
16	KMI	3
17	B	6
18	A	12
19	I1C	7
20	C	4
21	I2C	3
22	MOVE	2
23	D	2
24	IOK	4
25	N1	2
26	J1	3
27	J	3
28	2	7
29	1	22
30	0	4
31	3	2
		201

$n_2 = 31$

$N_2 = 201$

ALGORITHM 515

GENERATION OF A VECTOR

OPERATORS

1	=	19
2	-	8
3	DO	2
4	,	3
5	Subscript	8
6	IF ()	5
7	.NE.	1
8	+	5
9	BINOM ()	1
10	.LT.	2
11	GO TO 10(a)	1
12	GO TO 10(b)	1
13	GO TO 30	1
14	END	2
15	.GE.	1
16	.EQ.	1
17	()	2
18	*	1
19	/	1
20	EOS	22
		87

$n_1 = 20$

$N_1 = 87$

OPERANDS

1	K	7
2	P1	3
3	P(a)	4
4	I	12
5	C	10
6	R	3
7	N	3
8	L	2
9	N1	7
10	P(b)	7
11	R	5
12	I	3
13	BINOM	1
14	0	3
15	1	7
16	2	2
		79

$n_2 = 16$

$N_2 = 79$

ALGORITHM 518

INCOMPLETE BESSEL FUNCTION Io

OPERATORS

1	=	36
2	MOD (,)	1
3	+	16
4	IF()	5
5	.LT.	2
6	-	17
7	.GT.	2
8	GO TO 30	1
9	GO TO 20	1
10	GO TO 40	1
11	.LE.	1
12	*	21
13	/	14
14	()	13
15	FLOAT	1
16	SIN ()	3
17	COS ()	2
18	DO	1
19	,	1
20	SQRT ()	2
21	GAUSS ()	1
22	ERF ()	1
23	END	1
24	EOS	39
		183

n_1 = 24

N_1 = 183

OPERANDS

1	Z	18
2	VK	1
3	U	6
4	T	1
5	PI	3
6	TPI	2
7	Y	17
8	CK	1
9	V	10
10	IP	3
11	A2	1
12	A3	1
13	A4	1
14	A1	1
15	P	6
16	S	12
17	C	10
18	SN	5
19	CN	4
20	R	7
21	N	1
22	VMISES	7
23	C1	1
24	0.0	6
25	2.0	2
26	2	1
27	1.0	4
28	0.5	4
29	24.0	1
30	54.0	2
31	347.0	2
32	26.0	2
33	6.0	3
34	12.0	1
35	3.0	2
36	16.0	1
37	1.75	1
38	83.5	1
		152

n_2 = 38

N_2 = 152

ALGORITHM 521

REPEATED INTEGRALS OF THE COERROR FUNCTION

OPERATORS

1	=	86	31	SQRT()	3
2	IF()	15	32	/	11
3	.LT.	7	33	DSQRT()	1
4	GO TO 230	1	34	DATAN()	1
5	GO TO 240	1	35	EXP()	1
6	GO TO 250	1	36	+	16
7	GO TO 10	1	37	( )	1
8	GO TO 90	1	38	.EQ.	1
9	GO TO 20	1	39	FLOAT	6
10	GO TO 260	1	40	.GE.	1
11	GO TO 100	1	41	.AND.	2
12	GO TO 30	1	42	.OR.	1
13	GO TO 270	1	43	DBLE	2
14	GO TO 40	1	44	DABS	2
15	GO TO 60	1	45	DEXP	1
16	GO TO 110	1	46	SNGL	3
17	GO TO 130	1	47	DO	8
18	GO TO 120	1	48	,	9
19	GO TO 70	2	49	Subscript	12
20	GO TO 280	1	50	IFIX	1
21	GO TO 150	1	51	ABS	2
22	GO TO 210	1	52	END	1
23	GO TO 220	1	53	EOS	105
24	.GT.	9			400
25	-	18			
26	*	34			
27	**	3			
28	.NOT.	1			
29	ATAN()	1			
30	ALOG()	2			

$n_1 = 53$

$N_1 = 400$

ALGORITHM 521

REPEATED INTEGRALS OF THE COERROR FUNCTION

OPERANDS

1	IFLAG	7	41	DN	5
2	0	1	42	0.D0	1
3	NMAX	6	43	DN1	5
4	1	12	44	2.D0	1
5	500	1	45	DTERM	4
6	TOL	4	46	DF0	4
7	PREC	2	47	2.D2	1
8	ACC	4	48	DF1	5
9	1.	7	49	FZERO	3
10	I	3	50	N	23
11	EPS	2	51	DFM1	2
12	10.	2	52	F	4
13	XSQ	6	53	F1	7
14	X	13	54	FM1	2
15	FRSTTM	2	55	N0	9
16	P	2	56	X0	6
17	AL10	5	57	.572	1
18	C	6	58	2	3
19	DC	3	59	R1	7
20	1.D0	5	60	FN0	2
21	.FALSE.	1	61	NU	6
22	S	7	62	2.3026	1
23	0	7	63	1.3863	1
24	BOTEXP	1	64	2.8284	1
25	B	1	65	I0	1
26	.5	8	66	5000	1
27	.25	1	67	NOB1	3
28	2.	4	68	NUM	2
29	B1	2	69	R0	2
30	F0	11	70	R	6
31	T	2	71	K	4
32	SQ	3	72	FF	4
33	1.12	2	73	0.5	1
34	DEPS	2	74	3	1
35	.5D0	2	75	4	1
36	10.D0	1	76	5	1
37	DX	5	77	6	1
38	DXSQ	3			303
39	DTE	6			
40	DSUM	6			

$n_2 = 77$

$N_2 = 303$

ALGORITHM 523

CONVEX (FOR PLANAR SETS (Z))

OPERATORS

1	=	132	31	GO TO 22	2
2	READ	2	32	GO TO 21	1
3	WRITE	3	33	GO TO 3	2
4	+	10	34	GO TO 1	1
5	DO	11	35	GO TO 2	1
6	,	59	36	GO TO 5	1
7	-	21	37	GO TO 4	1
8	Subscript	107	38	GO TO 6	2
9	CALL CONVEX	1	39	GO TO 18	1
10	CALL SPLIT	5	40	GO TO 23	1
11	STOP	1	41	GO TO 11	2
12	END	3	42	GO TO 10	1
13	IF()	33	43	GO TO 8	1
14	.NE.	3	44	GO TO 12	1
15	GO TO 1	1	45	GO TO 9	1
16	GO TO 2	1	46	GO TO 16	2
17	GO TO 3	1	47	GO TO 15	1
18	GO TO 4	1	48	GO TO 13	1
19	GO TO 5	1	49	GO TO 17	3
20	GO TO 6	6	50	GO TO 14	1
21	SIGN()	2	51	GO TO 19	1
22	*	4	52	GO TO 20	1
23	FLOAT()	1	53	GO TO 25	1
24	( )	5	54	GO TO 24	2
25	/	1	55	GO TO 7	2
26	.LE.	3	56	GO TO 26	2
27	.EQ.	16	57	.OR.	1
28	.LT.	1	58	.AND.	1
29	.GE.	4	59	.NOT.	2
30	.GT.	1	60	EOS	187
					667

$n_1 = 60$

$N_1 = 667$

ALGORITHM 523

CONVEX (FOR PLANAR SETS (Z))

OPERANDS

1	NCOUNT	2	36	MXB	8
2	N	24	37	NIA	8
3	N1	2	38	0	14
4	I (a)	15	39	1	76
5	J (a)	10	40	2	32
6	IN	20	41	4	1
7	XX,X	43	42	3	2
8	M	20	43	-1	1
9	IWORK,IA,IB	35	44	-2	2
10	IH	19	45	1.	2
11	NHULL,NH	20	46	0.	4
12	IL	20	47	T	3
13	IK	4	48	JJ	4
14	X	5	49	II	7
15	Y	2	50	XT	2
16	KN Convex	16	51	DIR	2
17	KX	18	52	S	3
18	MP1	2	53	A	3
19	MIN	2	54	B	2
20	MX	6	55	UP	3
21	MAXE	5	56	NA	5
22	.FALSE.	5	57	MAXA	2
23	MINE	5	58	DOWN	3
24	I (b)	13	59	NB	5
25	J (b)	16	60	MAXB	2
26	.TRUE.	3	61	I (c)	2
27	INH	21	62	IS	6
28	NIB	11	63	Z	8
29	MA	26	64	IABV	1
30	MM	2	65	IBEL	1
31	MB	16			652
32	MXA	8			
33	MXBB	2			
34	ILINH	8			
35	MBB	7			

$n_2 = 65$

$N_2 = 652$

ALGORITHM 529

PERMUTATIONS TO BLOCK TRIANGULAR FORM

OPERATORS

1	CALL MC13E	1	18	GO TO 60 (b)	1
2	CALL FA01BS	2	19	GO TO 40	1
3	CALL SETUP	1	20	GO TO 10	1
4	CALL MC13D	1	21	GO TO 20	1
5	END	4	22	READ	1
6	DO	15	23	WRITE	6
7	,	19	24	( )	1
8	Subscript	59	25	STOP	1
9	IF()	22	26	=	77
10	GO TO 90	2	27	+	17
11	GO TO 30	1	28	-	10
12	GO TO 70	1	29	.NE.	3
13	GO TO 60 (a)	2	30	.LT.	5
14	GO TO 50	1	31	.EQ.	10
15	GO TO 100 (a)	1	32	.NOT.	2
16	GO TO 80	1	33	EOS	106
17	GO TO 100 (b)	1			377

$n_1 = 33$

$N_1 = 377$

ALGORITHM 529

PERMUTATIONS TO BLOCK TRIANGULAR FORM

OPERANDS

1	N	28	29	I (a)	3
2	ICN	5	30	MM	2
3	LICN	4	31	IPP	4
4	IP	5	32	J (b)	12
5	LENR	8	33	I (b)	10
6	IOR	4	34	A	8
7	IB	13	35	K9	1
8	NUM	12	36	INDEX	11
9	IW	4	37	IBLOCK	6
10	ICNT	6	38	IJ	2
11	NNM1	2	39	HOLD	5
12	J (a)	4	40	JBLOCK	5
13	NUMB	8	41	II	3
14	ARP	5	42	JJ	3
15	ISN	3	43	EX	1
16	IV	24	44	0	11
17	IST	9	45	1	42
18	LOWL	12	46	50	1
19	DUMMY	1	47	1000	1
20	I1	5	48	.FALSE.	1
21	I2	4	49	.TRUE.	2
22	II	5	50	100	1
23	IW	13	51	BLANK	2
24	IST1	2	52	NOT	1
25	LCNT	2	53	I(c)	7
26	STP	3	54	IND	5
27	PREV	2	55	J(c)	3
28	K	2			348

$n_2 = 55$

$N_2 = 348$

ALGORITHM 533

GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING

OPERATORS

1	IF()	26	28	()	4
2	-	27	29	END	9
3	GO TO 60	1	30	EOS	189
4	GO TO 90	2	31	GO TO 1	1
5	GO TO 70	2	32	GO TO 2	2
6	GO TO 80	1	33	GO TO 4	1
7	GO TO 50	1	34	CALL PREORD	1
8	GO TO 1004	2	35	CALL NSPV	2
9	GO TO 130	1	36	CALL RESCHK	1
10	GO TO 120	2	37	WRITE	3
11	GO TO 140	2	38	GO TO 30	1
12	GO TO 150	1	39	GO TO 20	1
13	GO TO 1005	1	40	GO TO 1001	1
14	GO TO 170	1	41	GO TO 1002	1
15	GO TO 160	1	42	GO TO 30	1
16	GO TO 190	1	43	GO TO 1003	1
17	=	125	44	GO TO 40	1
18	Subscript	104	45	GO TO 20	1
19	+	36	46	GO TO 110	1
20	*	9	47	GO TO 50	1
21	.GT.	9	48	GO TO 90	1
22	ABS()	2	49	.GE.	2
23	.LE.	2	50	.NE.	1
24	.EQ.	9	51	DABS()	2
25	/	1	52	**	1
26	DO	14	53	DSQRT	1
27	,	16	54	DBLE()	3
					653

$n_1 = 54$

$N_1 = 653$

ALGORITHM 533

GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING

OPERANDS

1	K (a)	9	36	0	9
2	IA	12	37	2	2
3	IAPTR	18	38	1.0	1
4	I (a)	4	39	3	1
5	NG	5	40	Y	8
6	J(a)	2	41	U	5
7	BK	10	42	P	155
8	JA	9	43	IU	13
9	A	9	44	JU	5
10	B	9	45	ONE	2
11	N	30	46	ZERO	7
12	R	6	47	J(d)	20
13	C	17	48	JUPTR	5
14	IC	12	49	K(c)	25
15	MAX	3	50	VK	4
16	X	20	51	JMIN	9
17	ITEMP	27	52	JMAX	9
18	RTEMP	11	53	JAJ	2
19	IERR	10	54	VJ	14
20	I(b)	18	55	PPK	13
21	K(b)	10	56	PK	6
22	KDEG	6	57	VI	20
23	J(b)	3	58	YK	9
24	RESID	6	59	LKI	3
25	RESIDM	4	60	JUJ	6
26	I(c)	3	61	XPVMAX	4
27	ROWSUM	6	62	MAXC	9
28	JMIN	2	63	NZCNT	4
29	JMAX	2	64	PV	7
30	J(c)	3	65	V	2
31	JAJ	2	66	XPV	3
32	1	57	67	MAXCL	2
33	0.	3	68	DK	3
34	4	2	69	I	4
35	1.5	2	70	CK	4
					608

$n_2 = 70$

$N_2 = 608$

APPENDIX D

PL/1 MEASURABLE PROPERTIES

ALGORITHM 365

COMPLEX ROOT FINDING

OPERATORS

1	Subscript	27	12	DO WHILE()	4
2	=	76	13	*	8
3	+	19	14	CONJG()	1
4	-	11	15	ELSE	10
5	FUNC ()	4	16	<	4
6	ABS()	8	17	<=	2
7	REAL ()	4	18	( )	1
8	IMAG ()	4	19	/	1
9	IF-THEN	12	20	END	1
10	>	3	21	EOS (;)	84
11	DO	10			294

$n_1 = 21$

$N_1 = 294$

ALGORITHM 365

COMPLEX ROOT FINDING

OPERANDS

1	U	8	22	ZE	1
2	H	14	23	HE	1
3	HS	1	24	DE	1
4	ZO	8	25	1	22
5	Z5	1	26	0.	4
6	N	3	27	1.	3
7	CW	12	28	0.8660254	2
8	WO	7	29	0.5000000	2
9	DS	1	30	0.9659258	3
10	I1	7	31	0.2588190	4
11	I2	9	32	0.7071068	4
12	I3	10	33	2	10
13	DM	2	34	3	9
14	K	7	35	4	1
15	I	7	36	5	1
16	V	6	37	6	1
17	A	4	38	7	2
18	Z	8	39	0	25
19	W	11	40	0.25	2
20	NR	8	41	0.4	1
21	HM	2			234

$n_2 = 41$

$N_2 = 234$

ALGORITHM 386

GREATEST COMMON DIVISOR OF N INTEGERS AND MULTIPLIERS

OPERATORS

1	DO	12
2	=	47
3	TO	4
4	IF-THEN	8
5	Subscript	29
6	¬=	3
7	ELSE	5
8	+	5
9	<	1
10	-	9
11	ABS()	1
12	DO WHILE()	1
13	/	3
14	*	7
15	( )	1
16	END	1
17	EOS(;)	52
		189

$n_1 = 17$

$N_1 = 189$

OPERANDS

1	M	20
2	N	4
3	A	20
4	IGCD	3
5	Z	9
6	MP1	3
7	MP2	2
8	ISIGN	4
9	C1	11
10	I	9
11	Y1	7
12	Y2	5
13	C2	8
14	KK	7
15	Q	6
16	IP1	2
17	J	4
18	K	6
19	1	12
20	0	11
21	2	2
		155

$n_2 = 21$

$N_2 = 155$

ALGORITHM 424

QUADRATURE (D-1)

OPERATORS

1	=	161	14	ABS ()	2
2	( )	20	15	IF THEN	3
3	+	74	16	<	1
4	*	59	17	<=	1
5	-	52	18	COS()	3
6	MIN()	1	19	SIN()	3
7	**	4	20	CALL R3PASS	1
8	DO	20	21	>=	1
9	TO	17	22	ELSE	2
10	Subscript	111	23	PUT	1
11	/	11	24	RETURN ()	1
12	DO WHILE()	2	25	END	2
13	BY	12	26	EOS (;)	167
					732

$n_1 = 26$

$N_1 = 732$

ALGORITHM 424

QUADRATURE (D-1)

OPERANDS

1	A	3	36	K0	10	71	S1	5
2	ANGLE(a)	6	37	K1	9	72	S2	3
3	ANGLE(b)	3	38	K	20	73	T1	19
4	B	3	39	LIMIT	1	74	T2	17
5	CSXFRM	63	40	L	5	75	T3	7
6	C	3	41	L1	2	76	T4	6
7	CCQUAD	1	42	L2	2	77	T5	6
8	C1	5	43	L3	2	78	T6	6
9	C2	3	44	L4	2	79	T7	3
10	CENTRE	10	45	L5	2	80	T8	3
11	ESTERR	3	46	L6	2	81	T9	3
12	FUND(a)	4	47	L7	2	82	T10	3
13	FUND(b)	1	48	L8	1	83	T11	3
14	F	11	49	MAX	2	84	T12	3
15	HALF-M	2	50	M-MAX	5	85	TOLERR	1
16	HAFRT3	4	51	M,STEP	12	86	TWOPI	1
17	I0	3	52	M3	4	87	USTED	10
18	IDIFF2	3	53	N	18	88	WIDTH	7
19	IDIFF	3	54	N2	9	89	X0	10
20	ISUM	3	55	N3	7	90	X1	12
21	I1	3	56	NLESS	1	91	X2	12
22	I2	3	57	N-LESS-1	3	92	0	3
23	J(a)	25	58	N-LESS-3	2	93	1	44
24	J1	11	59	NEWINIT	9	94	2	30
25	J2	11	60	OLDINT	3	95	3	14
26	J3	2	61	PI	1	96	4	6
27	J4	2	62	RT3	3	97	5	4
28	J5	2	63	R0	3	98	6	5
29	J6	2	64	R1	3	99	7	5
30	J7	2	65	R2	3	100	.5E0	10
31	J8	2	66	RSUM2	3	101	2.E0	9
32	J-REV	6	67	RSUM	8	102	3.E0	2
33	J(b)	6	68	RDIFF	8	103	KK	4
34	J0	3	69	S	3	104	KK2	3
35	J1	2	70	SHIFT	12			680

$n_2 = 104$

$N_2 = 680$

ALGORITHM 500

MINIMIZATION OF UNCONSTRAINED MULTIVARIATE FUNCTIONS

OPERATORS

1	CAL CALCFG()	3	14	( )	39
2	=	149	15	>=	3
3	*	57	16	&	9
4	DO	38	17	<=	4
5	TO	24	18	ELSE	6
6	+	37	19	ABS ()	5
7	Subscript	62	20	<	5
8	IF THEN	35	21	|	2
9	SQRT()	4	22	¬	2
10	/	15	23	**	1
11	DO WHILE()	3	24	END	1
12	>	23	25	EOS (;)	170
13	-	23			720

$n_1 = 25$

$N_1 = 720$

ALGORITHM 500

MINIMIZATION OF UNCONSTRAINED MULTIVARIATE FUNCTIONS

OPERANDS

1	N	28	37	AA	5
2	X	7	38	BB	5
3	F	16	39	CC	3
4	G	15	40	FM	2
5	IFUN	7	41	GM	2
6	ITER	5	42	SS	6
7	IERR	6	43	DY	6
8	EPS2	5	44	YHY	4
9	EPS	2	45	C1	4
10	DUM	29	46	C2	2
11	STEP	9	47	1	41
12	I	78	48	0	8
13	SUM	4	49	0.	19
14	STP	3	50	100.	1
15	IH	1	51	'1'B	4
16	IJ	23	52	2	6
17	J	16	53	3	1
18	H	7	54	1.0E+30	1
19	II	4	55	1.0E-30	1
20	D	20	56	.9	1
21	XX	7	57	1.	4
22	GG	9	58	2.	3
23	G1	19	59	0.8	3
24	STPMAX	7	60	4	1
25	ALPHA	9	61	0.0001	2
26	FG	5	62	0.5	2
27	AL	36	63	10.	2
28	MODE	4	64	1.001	1
29	FEST	2	65	3.0	3
30	KK	4	66	1.0E+05	1
31	F1	9	67	0.001	1
32	MAXF	2	68	0.1	1
33	G2	19	69	0.999	1
34	CONV	8	70	100000	1
35	BL	21	71	'0'B	1
36	KKZ	5			600

$n_2 = 71$

$N_2 = 600$

ALGORITHM 502

DEPENDENCE OF SOLUTION OF NONLINEAR SYSTEMS ON A PARAMETER

OPERATORS

1	=	137	16	ABS()	13
2	EOS(;)	162	17	<=	2
3	+	23	18	GO TO L50	1
4	IF THEN	42	19	DO WHILE()	2
5	¬=	9	20	>	13
6	DO	49	21	>=	2
7	TO	28	22	,	32
8	CALL FCTN	2	23	ELSE	2
9	Subscript	112	24	<	2
10	**	3	25	|	2
11	-	24	26	/	8
12	SQRT ()	3	27	( )	4
13	PUT	11	28	¬	1
14	CALL GAUSE	2	29	END	3
15	*	26			720

$n_1 = 29$

$N_1 = 720$

ALGORITHM 502

DEPENDENCE OF SOLUTION OF NONLINEAR SYSTEMS ON A PARAMETER

OPERANDS

1	AMAX	4	41	N1(b)	7
2	BETA	6	42	N1(c)	2
3	DXDT	9	43	NPRNT	5
4	DER	13	44	NOUT	13
5	DXK2	4	45	NC	5
6	E	1	46	NCRAD	2
7	EPS	2	47	NCORR	3
8	F,B	17	48	NDIR	4
9	G,A	15	49	OUT	9
10	H	10	50	PREF	3
11	HH	1	51	P	19
12	HMAX	2	52	SQUAR	16
13	I(a)	57	53	W	3
14	I(b)	30	54	X	33
15	I(c)	24	55	XLOW	1
16	INDSP	1	56	XUPP	1
17	INDIC	1	57	Y	2
18	ID	4	58	0	17
19	IRK	3	59	1	66
20	ITIN	3	60	2	11
21	IRR	5	61	3	13
22	IR	14	62	4	4
23	INITIAL	3	63	10,NN	2
24	IS	7	64	11,MM	2
25	J(a)	6	65	100	1
26	J(b)	8	66	0.0	13
27	J(c)	3	67	1.0	2
28	K	30	68	0.5	1
29	KOUT	3	69	3.0	1
30	K1	5	70	5.0	1
31	LW	1	71	9.0	1
32	L	2	72	12.0	1
33	LL	2	73	16.0	1
34	M	6	74	23.0	1
35	MADMS	14	75	0.8	1
36	MXADMS	2	76	37.0	1
37	MARK	3	77	55.0	1
38	MAXOUT	5	78	59.0	1
39	N	16	79	'1'B	2
40	N1(a)	18	80	D	1
			81	24.0	1
					634

$n_2 = 81$

$N_2 = 634$

ALGORITHM 505

A LIST INSERTION SORT FOR KEYS WITH ARBITRARY KEY DISTRIBUTION

OPERATORS

1	=	55
2	Subscript	24
3	IF THEN	10
4	-	13
5	>	3
6	DO	12
7	TO	2
8	+	5
9	>=	2
10	ELSE	4
11	<=	1
12	DO WHILE()	2
13	<	2
14	BY	1
15	END	1
16	EOS(;)	63
		200

$n_1 = 16$

$N_1 = 200$

OPERANDS

1	MINN	7
2	II	9
3	MAXX	6
4	L	15
5	JJ	2
6	KMIN	6
7	K	7
8	KMAX	3
9	IA	2
10	IRT	5
11	ITAB	3
12	TAB	2
13	ISTRT	4
14	J	17
15	KJ	8
16	I	9
17	IPOI	13
18	MADIN	12
19	KK	6
20	KEY	5
21	KDIFF	5
22	0	13
23	1	12
		171

$n_2 = 23$

$N_2 = 171$

ALGORITHM 509

A HYBRID PROFILE REDUCTION ALGORITHM

OPERATORS

1	DO	26	14	ELSE	2
2	=	72	15	END	4
3	TO	17	16	,	2
4	Subscript	32	17	GO TO L50	1
5	CALL FORMLV	2	18	RETURN	2
6	IF THEN	12	19	-	3
7	+	16	20	¬	1
8	DO WHILE	2	21	( )	2
9	MINCON	1	22	&	1
10	CALL DLETE	2	23	¬=	1
11	>	3	24	PUT	1
12	<=	3	25	EOS(;)	87
13	>=	3			298

$n_1 = 25$

$N_1 = 298$

ALGORITHM 509

A HYBRID PROFILE REDUCTION ALGORITHM

OPERANDS

1	I(a)	20	24	Q	6
2	IDPTH	4	25	IQ	4
3	LSTPT	6	26	NS2	2
4	NSTPT	5	27	I(b)	4
5	I (b)	3	28	LSTSZE	8
6	N	1	29	IX	3
7	LVLS2	1	30	IROWDG	3
8	LVLST	5	31	J	6
9	LEVEL	11	32	K	3
10	S2,X	10	33	SMLST	2
11	S2SZE,X52E	8	34	I(c)	5
12	S3,Y	6	35	SETSZE	7
13	S3SZE,YSZE	8	36	ELEMENT	3
14	QPTR	13	37	SET	5
15	KK	4	38	J	3
16	M	3	39	LOWER	2
17	CONECT CONLST	4	40	UPPER	2
18	CONSZE	7	41	SETSZE	6
19	NDSTK	2	42	I(d)	2
20	NR	1	43	SET	1
21	NDEG	2	44	1	43
22	NEW	4	45	'1'B	1
23	NXTNUM	12	46	0	10
					271

$n_2 = 46$

$N_2 = 271$

ALGORITHM 512

SOLUTION OF POSITIVE DEFINITE SYMMETRIC QUINDIAGONAL

OPERATORS

1	=	70
2	-	57
3	Sbuscript	131
4	SQRT()	6
5	/	16
6	*	38
7	DO	12
8	TO	12
9	+	16
10	( )	8
11	**	4
12	END	3
13	EOS(;)	70
		443

$n_1 = 13$

$N_1 = 443$

OPERANDS

1	N	19
2	N1(a)	14
3	N2(b)	15
4	N3(c)	7
5	N4	2
6	C	26
7	V(a)	15
8	B	17
9	I(a)	34
10	I1	4
11	I2	5
12	U	14
13	A	10
14	G	17
15	N	21
16	H	20
17	N1(a)	12
18	N2(b)	13
19	N3(c)	5
20	Q	30
21	E	7
22	I(b)	21
23	F	5
24	U	4
25	V(b)	4
26	II	2
27	I(c)	4
28	S	2
29	1	32
30	2	22
31	3	6
32	4	1
33	0.0	5
34	1.0	2
		417

$n_2 = 34$

$N_2 = 417$

ALGORITHM 513

ANALYSIS OF IN-SITU TRANSPOSITION

OPERATORS

1	IF THEN	16
2	<	6
3	'	3
4	DO	13
5	=	68
6	¬=	2
7	*	7
8	-	15
9	TO	3
10	+	5
11	()	5
12	Subscript	16
13	¬	1
14	DO WHILE	5
15	MOD ()	1
16	/	2
17	<=	3
18	ELSE	5
19	>	3
20	>=	1
21	END	1
22	EOS(;)	78
		269

$n_1 = 22$

$N_1 = 269$

OPERANDS

1	M	10
2	N	10
3	IOK	6
4	MN	3
5	IWRK	5
6	N1	2
7	I	19
8	J1	2
9	J	3
10	I1	15
11	I2	15
12	B	6
13	A	12
14	NCOUNT	6
15	K	8
16	MOVE	4
17	IR2	4
18	IR1	4
19	KK	3
20	IRO	3
21	IM	7
22	KMI	3
23	I1C	7
24	C	4
25	KK1	4
26	I2C	3
27	D	2
28	KK2	4
29	MAXX	3
30	2	7
31	0	9
32	1	30
33	3	2
34	'1'B	1
		226

$n_2 = 34$

$N_2 = 226$

ALGORITHM 515

GENERATION OF A VECTOR

OPERATORS

1	=	23
2	-	8
3	DO	4
4	TO	2
5	Subscript	8
6	IF - THEN	5
7	¬=	1
8	DO WHILE	1
9	+	5
10	BINOM()	1
11	<	2
12	ELSE	1
13	END	2
14	>=	1
15	()	2
16	*	1
17	/	1
18	RETURN ()	1
19	EOS(;)	25
		94

$n_1 = 19$

$N_1 = 94$

OPERANDS

1	K	7
2	P1	3
3	P(a)	5
4	I(a)	12
5	C	9
6	KK	4
7	R(a)	3
8	N	3
9	L	2
10	N1	7
11	P(b)	7
12	R(b)	5
13	I(b)	3
14	0	4
15	1	10
16	2	2
		86

$n_2 = 16$

$N_2 = 86$

ALGORITHM 518

INCOMPLETE BESSEL FUNCTION Io

OPERATORS

1	=	35
2	MOD(,)	1
3	+	12
4	IF THEN	7
5	<	3
6	-	13
7	<=	1
8	DO	3
9		3
10	*	16
11	/	11
12	()	10
13	SIN()	3
14	COS()	2
15	TO	1
16	SQRT()	1
17	GAUSS()	1
18	END	1
19	EOS(;)	38
		162

n_1 = 19

N_1 = 162

OPERANDS

1	Z	14
2	VK	1
3	U	6
4	T	1
5	PI	3
6	TPI	2
7	Y	15
8	CK	1
9	V	9
10	IP	3
11	A2	1
12	A3	1
13	A4	1
14	A1	1
15	P	6
16	S	10
17	C	8
18	SN	5
19	CN	4
20	R	6
21	N	1
22	VMISES	10
23	C1	1
24	0.0	8
25	2.0	1
26	2	1
27	1.0	6
28	0.5	2
29	24.0	1
30	54.0	1
31	347.0	1
32	26.0	1
33	6.0	2
34	3.0	2
35	16.0	1
36	1.75	1
37	83.5	1
		139

n_2 = 37

N_2 = 139

ALGORITHM 521

REPEATED INTEGRALS OF THE COERROR FUNCTION

OPERATORS

1	=	109	19	&	2
2	IF THEN	19	20	|	1
3	<	5	21	DO WHILE	2
4	DO	27	22	ABS	4
5	>	7	23	TO	10
6	-	21	24	Subscript	14
7	*	38	25	ELSE	4
8	**	3	26	END	1
9	()	19	27	EOS(;)	125
10	ATAN()	2			459
11	LOG()	2			
12	SQRT()	4			
13	/	13			
14	EXP	2			
15	+	18			
16	<=	3			
17	>=	3			
18	¬	1			

$n_1 = 27$

$N_1 = 459$

ALGORITHM 521

REPEATED INTEGRALS OF THE COERROR FUNCTION

OPERANDS

1	IFLAG	7	38	F1	15
2	NMAX	9	39	FM1	6
3	TOL	4	40	NO	11
4	PREC	2	41	XO	7
5	ACC	2	42	R1	7
6	I	4	43	FNO	2
7	EPS	2	44	NU	6
8	ACC	2	45	IO	1
9	XSQ	6	46	NOP1	3
10	X	17	47	NUM	2
11	FRSTTM	2	48	RO	2
12	P	3	49	R	6
13	AL10	5	50	K	5
14	C	6	51	FF	4
15	DC	3	52	0	4
16	S	8	53	1	24
17	BOTEXP	1	54	500	1
18	B	3	55	2	4
19	B1	2	56	3	1
20	FO	17	57	.5	11
21	T	2	58	10.	3
22	SQ	3	59	1.	11
23	DEPS	2	60	'O'B	1
24	DX	5	61	0.	8
25	DXSQ	3	62	.25	1
26	DTE	5	63	2.	5
27	DSUM	6	64	4	1
28	DN	5	65	1.12	2
29	DN1	4	66	2.E2	1
30	KK	5	67	5	1
31	DTERM	3	68	.572	1
32	DFO	4	69	2.3026	1
33	DF1	5	70	1.3863	1
34	FZERO	5	71	2.8484	1
35	N	30	72	5.	1
36	DFM1	2	73	5000	1
37	F	5	74	6	1
					362

n_2 = 74

N_2 = 362

ALGORITHM 523

CONVEX (FOR PLANAR SETS (Z))

OPERATORS

1	=	165	19	¬	1
2	GET	2	20	>	4
3	PUT	3	21	>=	2
4	+	10	22	<	4
5	DO	29	23	<=	1
6	TO	11	24	&	2
7	-	26	25	|	1
8	Subscript	113	26	CALL SPLIT	5
9	CALL CONVEX	1	27	GO TO L10	1
10	,	44	28	GO TO L11	2
11	END	3	29	GO TO LB8	1
12	IF THEN	36	30	GO TO L12	1
13	¬=	3	31	GO TO LB9	1
14	( )	3	32	GO TO L16	2
15	/	1	33	GO TO L15	1
16	*	4	34	GO TO L13	1
17	ELSE	5	35	GO TO L14	1
18	SIGN	2	36	EOS(;)	184
					673

$n_1 = 36$

$N_1 = 673$

ALGORITHM 523

CONVEX (FOR PLANAR SETS (Z))

OPERANDS

1	NCOUNT	2	33	MXB	8
2	N	15	34	NIA	8
3	N1	2	35	T	3
4	I(a)	16	36	JJ	4
5	J(a)	9	37	II	7
6	N	22	38	A	3
7	X,XX	57	39	B	2
8	M	20	40	XT	2
9	IK	4	41	DIR	2
10	IA,IB,IWORK	36	42	S	3
11	IH	22	43	UP	3
12	NH,NHULL	31	44	NA	5
13	IL	21	45	MAXA	2
14	Y	2	46	NB	4
15	KN	20	47	NB	4
16	KX	21	48	MAXB	2
17	MP1	3	49	I	2
18	MINN	7	50	IS	6
19	MX	6	51	Z	8
20	MAXE	5	52	IABV	1
21	MINE	5	53	IBEL	1
22	I(b)	13	54	0	18
23	J(b)	18	55	1	82
24	INH	21	56	2	38
25	NIB	11	57	4	1
26	MA	25	58	3	3
27	MM	2	59	'0'B	4
28	MB	16	60	'1'B	4
29	MXA	8	61	-1	1
30	MXBB	2	62	-2	1
31	ILINH	8	63	1.	2
32	MBB	6	64	0.	5
					694

$n_2 = 64$

$N_2 = 694$

ALGORITHM 529

PERMUTATIONS TO BLOCK TRIANGULAR FORM

OPERATORS

1	=	97	13	CALL SET UP	1
2	DO WHILE	2	14	CALL MC13D	1
3	GET	1	15	+	17
4	PUT	6	16	¬	2
5	IF-THEN	24	17	END	4
6	STOP	1	18	CALL MC13 E	1
7	DO	34	19	-	9
8	TO	15	20	>=	1
9	Subscript	63	21	GO TO OUT1	1
10	,	6	22	<	5
11	¬=	4	23	ELSE	3
12	CALL FA01BS	2	24	EOS(;)	111
					401

$n_1 = 24$

$N_1 = 401$

ALGORITHM 529

PERMUTATIONS TO BLOCK TRIANGULAR FORM

OPERANDS

1	MM	2	30	LOW1	16
2	LICN	4	31	NUMB	7
3	N	29	32	PREV	3
4	IPP	4	33	INCT	6
5	J(a)	11	34	NUM1	3
6	I(a)	10	35	J(c)	4
7	A	7	36	ISN	3
8	K9	1	37	IV	29
9	KK	3	38	IST	9
10	IP	5	39	DUMMY	2
11	ICN	5	40	I1	5
12	LENR	8	41	I2	4
13	ARP,IOR	9	42	II	5
14	IB	13	43	K	2
15	NUM	13	44	IST1	2
16	IW	20	45	LCNT	2
17	INDEX	10	46	STP	6
18	IBLOCK	5	47	STP1	2
19	IJ	2	48	I	3
20	HOLD	5	49	50	1
21	BLANK	2	50	1000	1
22	JBLOCK	6	51	'1'B	2
23	II	3	52	0	12
24	JJ	3	53	1	48
25	EX	1	54	'0'B	1
26	NOT	1	55	100	1
27	I(b)	7	56	2	1
28	IND	5	57	3	1
29	J(b)	3			378

$n_2 = 57$

$N_2 = 378$

ALGORITHM 533

GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING

OPERATORS

1	=	180	21	()	4
2	Subscript	101	22	<	3
3	DO	29	23	>=	2
4	TO	14	24	GO TO RT1	1
5	IF THEN	29	25	ELSE	3
6	¬=	4	26	|	1
7	-	31	27	GO TO RT2	1
8	+	37	28	/	1
9	CALL PREORD	1	29	EOS	198
10	CALL NSP1V	1	30	END	5
11	PUT	3			682
12	CALL RESCHK	1			
13	*	9			
14	ABS()	4			
15	>	11			
16	**	1			
17	SQRT	1			
18	DO WHILE	1			
19	<=	3			
20	CALL NSP1V1	1			

$n_1 = 30$

$N_1 = 682$

ALGORITHM 533

GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING

OPERANDS

1	A	9	40	J(d)	21
2	B	6	41	JAJ	2
3	BK	10	42	JUJ	6
4	C	17	43	JMIN	9
5	I(a)	3	44	JMAX	9
6	IA	13	45	JUPTR	5
7	IC,JU	15	46	K	28
8	ITEMP,IU,P	26	47	LKI	3
9	IERR	10	48	MAXC	9
10	IAPTR	18	49	MAXCL	2
11	J(a)	2	50	NZCNT	4
12	JA	9	51	ONE	2
13	K	10	52	PK	6
14	N	32	53	PPK	15
15	NG	5	54	PV	2
16	R	6	55	V	2
17	RTEMP,U,Y	10	56	VI	23
18	MAX	3	57	VJ	18
19	X	20	58	VK	4
20	I(b)	18	59	XPV	3
21	J(b)	3	60	XPVMAX	4
22	K	10	61	YK	9
23	KDEG	6	62	ZERO	7
24	I(c)	4	63	KK	3
25	J(c)	3	64	KK1(a)	3
26	JAJ	2	65	KK2	3
27	JMIN	2	66	PPKK	4
28	JMAX	2	67	KK3	3
29	RESID	6	68	KK4	3
30	RESIDM	4	69	KK1(b)	3
31	ROWSUM	6	70	0.	3
32	IU	13	71	0	16
33	JU	5	72	1.0	5
34	P	17	73	1.5	2
35	U	5	74	2	3
36	Y	8	75	3	1
37	CK	4	76	4	2
38	DK	3	77	'1'B	1
39	I(d)	4			602

$n_2 = 77$

$N_2 = 602$

LIST OF REFERENCES

1. Joel D. Aron, The Program Development Process, Part 1, Reading, Massachusetts: Addison-Wesley, 1974.

2. B. W. Boehm, "Software and Its Impact: Quantitative Assessment," Datamation, May 1973, pages 48-59.

3. Corrado Bohm and Guiseppe Jacopini, "Flow Diagrams, Turing Machines, and Languages with Only Two Formation Rules," Communications of Association for Computing Machinery, Vol. 9, May 1966, pages 366-371.

4. F. Brooks, "Why is the Software Late?" Data Management, August 1971.

5. Ole-J. Dahl, Edsger W. Dijkstra, and C. A. R. Noare, Structured Programming, New York, New York: Academic Press, 1972.

6. Edsger W. Dijkstra, "GO TO Statement Considered Harmful," Communications of the ACM, Volume 11, Number 3, March 1968, pages 147-148.

7. James L. Elshoff, "Measuring Commercial PL/1 Programs Using Halstead's Criteria," ACM SIGPLAN Notices, Volume 11, Number 5, May 1976, pages 38-46.

8. R. E. Fairley, "Modern Software Design Techniques," Proceedings of the Symposium on Computer Software Engineering, New York, New York: Polytechnic Press, 1976, pages 11-29.

9. Maurice H. Halstead, Elements of Software Science, New York, New York: Elsevier North-Holland, Inc., 1977.

10. Maurice H. Halstead, "Natural Laws Controlling Algorithm Structure?" ACM SIGPLAN Notices, Volume 7, Number 2, February 1972, pages 19-26.

11. Joan K. Hughes and Jay I. Michton, A Structured Approach to Programming, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977.

12. Michael A. Jackson, Principles of Program Design, New York, New York: Academic Press, 1975.

13. Donald E. Knuth, "Structured Programming With GO TO Statements," ACM Computing Surveys, Volume 6, Number 4, December 1974, pages 261-301.

14. C. V. Ramamoorthy and S. F. Ho, "Testing Large Software Systems with Automated Software Evaluation Systems," Current Trends in Programming Methodology, Volume 2, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977, pages 112-150.

15. H. Sackman, Man-Computer Problem Solving, New York New York: Averbach Publishers, Inc., 1970.